

Василий Боев

МОДЕЛИРОВАНИЕ СИСТЕМ

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА GPSS WORLD

Санкт-Петербург

«БХВ-Петербург»

2004

УДК 681.3.06
ББК 32.973.26-018.2
Б72

Боев В. Д.

Б72 Моделирование систем. Инструментальные средства GPSS World:
Учеб. пособие. — СПб.: БХВ-Петербург, 2004. — 368 с.: ил.

ISBN 5-94157-515-7

В учебном пособии рассматриваются основы построения и принципы функционирования общецелевой системы моделирования GPSS World. На многочисленных примерах раскрываются методы построения имитационных моделей с применением инструментальных средств GPSS World. Значительное внимание уделяется особенностям использования языка GPSS и языка Plus для моделирования сложных систем. Описываются методы проведения экспериментов с моделями и принятия по их результатам решений.

В приложениях в систематизированном виде приводятся дополнительные справочные материалы, необходимые для построения блок-диаграмм, написания и отладки программ моделей на современных ПЭВМ.

Для студентов, курсантов военных учебных заведений, научных работников

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Людмила Еремеевская</i>
Зав. редакцией	<i>Григорий Добин</i>
Компьютерная верстка	<i>Дмитрия Боева</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн обложки	<i>Игоря Цырульников</i>
Зав. производством	<i>Николай Тверских</i>

Рецензенты:

К. В. Кумунжиев, заведующий кафедрой «Информационных технологий и систем» Ульяновского государственного университета, доктор технических наук, профессор; А. П. Бочков, профессор кафедры прикладной математики Санкт-Петербургского государственного политехнического университета, доктор технических наук, доцент; В. А. Кежаев, начальник кафедры управления войсками Военного артиллерийского университета, доктор технических наук, профессор

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 01.06.04.
Формат 70х100 $\frac{1}{16}$. Печать офсетная. Усл. печ. л. 29,67.

Тираж 3000 экз. Заказ № 3367
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-515-7

© Боев В. Д., 2004
© Оформление, издательство "БХВ-Петербург", 2004

Содержание

Введение	1
1. Основы построения языка имитационного моделирования	4
1.1. Объекты системы моделирования GPSS World	4
1.2. Объекты языка имитационного моделирования	5
1.3. Порядок представления моделей в виде блок-диаграмм	11
1.4. Структура модели на языке GPSS	15
1.5. Объекты вычислительной категории	19
1.5.1. Константы	19
1.5.2. Системные числовые атрибуты	19
1.5.3. Арифметические, условные и логические операторы	21
1.5.4. Библиотечные математические функции	22
1.5.5. Библиотечные генераторы случайных чисел	22
1.5.6. Переменные пользователя	26
1.5.7. Выражения в операторах GPSS	27
1.5.8. Сохраняемые ячейки	27
1.5.9. Матрицы сохраняемых ячеек	29
1.5.10. Арифметические переменные и арифметические выражения	32
1.5.11. Булевы переменные	34
2. Построение моделей с одноканальными устройствами, функционирующими в режиме занятия и освобождения устройства	37
2.1. Организация поступления транзактов в модель и удаления транзактов из нее	38
2.1.1. Поступление транзактов в модель	38
2.1.2. Удаление транзактов из модели и завершение моделирования	41
2.2. Изменение значений параметров транзактов	43
2.2.1. Блок ASSIGN	43
2.2.2. Блок PLUS	44
2.2.3. Блок INDEX	44
2.2.4. Блок MARK	45
2.3. Занятие одноканального устройства и его освобождение	45
2.4. Имитация обслуживания посредством задержки во времени	47

2.5. Проверка состояния одноканального устройства	49
2.5.1. Проверка состояния одноканального устройства блоком GATE	49
2.5.2. Проверка состояния булевой переменной и блоком TEST	50
2.6. Методы сбора статистики в имитационной модели	53
2.6.1. Регистратор очереди	53
2.6.2. Статистические таблицы	56
2.7. Методы изменения маршрутов движения транзактов в модели	57
2.7.1. Блок TRANSFER	58
2.7.1.1. Режим безусловной передачи	58
2.7.1.2. Режим статистической передачи	59
2.7.1.3. Режим BOTH	60
2.7.1.4. Режим ALL	61
2.7.1.5. Режим PICK	63
2.7.1.6. Функциональный режим	65
2.7.1.7. Параметрический режим	65
2.7.1.8. Подпрограммный режим	66
2.7.1.9. Одновременный режим	66
2.7.2. Блок LOOP	67
2.7.3. Блок DISPLACE	70
2.8. Примеры построения моделей с одноканальным устройством обслуживания	74
3. Разработка и эксплуатация моделей в GPSS World	80
3.1. Создание объекта «Модель»	80
3.1.1. Запуск GPSS World	80
3.1.2. Ввод операторов модели	82
3.2. Создание объекта «Процесс моделирования»	87
3.3. Команды GPSS World	91
3.3.1. Интерактивные операторы	91
3.3.2. Команды определения объектов GPSS	92
3.3.3. Команды управления процессом моделирования	92
3.3.3.1. Команда CLEAR	93
3.3.3.2. Команда CONDUCT	93
3.3.3.3. Команда CONTINUE	94
3.3.3.4. Команда EXIT	94
3.3.3.5. Команда HALT	95
3.3.3.6. Команда INCLUDE	95
3.3.3.7. Команда REPORT	95
3.3.3.8. Команда RESET	96
3.3.3.9. Команда SHOW	97
3.3.3.10. Команда STEP	98
3.3.3.11. Команда STOP	98

3.4. Окна GPSS World	99
3.5. Отладка моделей	100
3.6. Отчеты	111
4. Построение моделей систем с одноканальными устройствами, функционирующими в режимах прерывания и недоступности	115
4.1. Прерывание одноканального устройства	115
4.1.1. Прерывание в приоритетном режиме	116
4.1.2. Прерывание в режиме «захвата»	121
4.1.3. Проверка состояния одноканального устройства, функционирующего в приоритетном режиме	126
4.2. Недоступность одноканального устройства	132
4.2.1. Перевод в недоступное состояние и восстановление доступности	132
4.2.2. Проверка состояний недоступности и доступности одноканального устройства	134
4.3. Сокращение машинного времени и изменение дисциплин обслуживания методом применения списков пользователя	136
4.3.1. Ввод транзактов в список пользователя в безусловном режиме	136
4.3.2. Вывод транзактов из списка пользователя	138
4.3.3. Ввод транзактов в список пользователя в условном режиме	144
4.4. Примеры построения моделей систем с одноканальными устройствами, функционирующими в различных режимах	146
5. Построение моделей систем с многоканальными устройствами и переключателями	158
5.1. Занятие многоканального устройства и его освобождение ...	158
5.2. Перевод многоканального устройства в недоступное состояние и восстановление доступности	160
5.3. Проверка состояния многоканального устройства	163
5.3.1. Проверка состояния блоком GATE	163
5.3.2. Проверка состояния булевой переменной и блоком TEST	165
5.4. Моделирование неисправностей многоканальных устройств	169
5.5. Моделирование переключателей	178
5.5.1. Проверка состояния логического ключа блоком GATE	178
5.5.2. Проверка состояния логического ключа блоком TEST и булевой переменной	180
5.6. Примеры построения моделей систем с многоканальным устройством обслуживания	183

6. Методы применения функций в имитационных моделях ..	188
6.1. Генераторы псевдослучайных чисел	188
6.2. Типы функций	189
6.3. Методы розыгрыша псевдослучайных чисел для дискретных равномерных распределений	190
6.4. Методы розыгрыша псевдослучайных чисел для дискретных неравномерных распределений	193
6.5. Процедура розыгрыша псевдослучайных чисел при непрерывном равномерном распределении	196
6.6. Методы розыгрыша псевдослучайных чисел для непрерывных неравномерных распределений	198
6.6.1. Моделирование экспоненциального распределения	199
6.6.2. Моделирование нормального распределения	201
6.7. Функции типа E, L и M	204
6.8. Примеры построения моделей с использованием функций ...	205
7. Особенности применения GPSS для моделирования сложных систем	212
7.1. Применение в моделях блоков создания копий транзактов ...	212
7.2. Организация в моделях синхронизации движения транзактов	214
7.2.1. Блок ASSEMBLE	214
7.2.2. Блок GATHER	217
7.2.3. Блок MATCH	219
7.3. Уменьшение числа объектов в модели методом косвенной адресации	222
7.4. Обработка одновременных событий	227
7.5. Методы построения моделей с рациональными вариантами обслуживания	234
7.5.1. Блок COUNT	234
7.5.2. Блок SELECT	236
7.6. Организация прогонов при изменении версий модели	248
7.7. Применение текстовых объектов и потоков данных	255
7.7.1. Блок OPEN	255
7.7.2. Блок CLOSE	256
7.7.3. Блок READ	257
7.7.4. Блок WRITE	258
7.7.5. Блок SEEK	259
7.7.6. Пример применения текстовых объектов и потоков данных	259

7.8. Непрерывное моделирование	266
7.8.1. Команда INTEGRATE	266
7.8.2. Блок INTEGRATION	267
7.8.3. Фазы моделирования и ошибки интегрирования	268
7.8.4. Пример непрерывного моделирования	268
8. Язык PLUS	272
8.1. Алфавит	272
8.2. Имена	272
8.3. Выражения	273
8.4. Plus-операторы	275
8.5. Библиотека процедур	277
8.6. Подготовка текстовых объектов	282
8.7. Реализация собственного метода интегрирования	286
8.8. Отладка Plus-процедур	288
9. Эксперименты с имитационными моделями	291
9.1. Основные понятия теории планирования экспериментов	291
9.1.1. Необходимость планирования экспериментов	291
9.1.2. Стратегическое планирование эксперимента	293
9.1.3. Тактическое планирование эксперимента	295
9.2. Дисперсионный анализ (отсеивающий эксперимент)	297
9.3. Эксперименты пользователя	310
9.4. Регрессионный анализ (оптимизирующий эксперимент)	318
Список литературы	322
Приложение 1	324
Приложение 2	332
Приложение 3	336
Приложение 4	342
Элементы стандартного отчета	342
1. Общая информация о результатах работы модели	342
2. Информация об именах	342
3. Информация о блоках	343
4. Информация об объектах типа «устройство»	344
5. Информация об объектах типа «очередь»	344
6. Информация об объектах типа «многоканальное устройство»	345

7. Информация о таблицах	345
8. Информация о списках пользователя	346
9. Информация о группах транзактов	346
10. Информация о числовых группах	347
11. Информация о логических переключателях	347
12. Информация о сохраняемых величинах (ячейках)	347
13. Информация о матрицах	347
14. Информация о списках текущих и будущих событий	348

Введение

Моделирование — наиболее мощный универсальный метод исследования и оценки эффективности систем, поведение которых зависит от воздействия случайных факторов. Области применения методов имитации чрезвычайно широки и разнообразны. Однако по опыту научной работы и материалам диссертационных советов можно сделать вывод о том, что исследователи пока довольно редко используют в качестве инструментальных средств исследования системы моделирования, преимущества которых вполне очевидны [18]. Системы моделирования имеют специализированные средства, реализующие дополнительные возможности по организации модельных экспериментов на компьютере. Они также предоставляют возможность учитывать в моделях фактор времени, то есть строить динамические имитационные модели, что особенно важно для многих систем, в том числе и систем военного назначения.

Применение универсальных языков программирования при реализации имитационных моделей позволяет исследователю достигнуть гибкости при разработке, отладке и испытании модели. Однако языки моделирования, ориентированные на определенную предметную область, являются языками более высокого уровня, поэтому дают возможность с меньшими затратами создавать программы моделей для исследования сложных систем.

Специализированные языки моделирования делят на три группы, соответствующие видам имитации: для непрерывных, дискретных и комбинированных процессов. Для моделирования дискретных систем широкое распространение получил пакет моделирования дискретных систем (ПМДС) GPSS (General Purpose Simulation System — общецелевая система моделирования) [27]. Написанные с помощью языка GPSS модели получают более адекватными исследуемой системе, чем построенные с использованием известной теории массового обслуживания. В моделях, написанных на языке GPSS, можно учесть большое количество факторов и отказаться от многих ограничений и допущений.

Еще одним языком имитационного моделирования дискретных систем является Simpas [2, 5, 19, 20], который, по мнению разработчиков [19], «по своим возможностям полностью перекрывает специализированный язык GPSS, обладая более высоким быстродействием и гибкостью». Система Simpas оформлена как модуль и потому является открытой по сравнению с GPSS/PC.

Это дает возможность исследователю разрабатывать свои процедуры и включать их в Simpas или подстраивать что-то из уже имеющегося в нем под свои нужды. При разработке моделей на GPSS/PC исследователь ограничен его возможностями, в то время как Simpas позволяет использовать возможности Object Pascal.

Однако следует отметить, что GPSS/PC и Simpas, а также GPSS/H [4, 14] предназначены для работы в операционной системе MS DOS. Поэтому имеются ограничения, которые в ряде случаев не позволяют осуществить разработку и эксплуатацию моделей сложных систем с требуемой степенью детализации.

Отмеченных недостатков практически не имеет новая общецелевая система моделирования GPSS World, разработанная компанией Minuteman (США) [22]. Эта система является развитием GPSS/PC [1, 3, 4, 21], но приобрела комбинированный характер, т. е. может моделировать как дискретные, так и непрерывные процессы. Эти возможности обеспечиваются как новыми объектами языка GPSS, так и включением в состав GPSS World языка Plus — языка программирования низкого уровня. Этот язык сделал GPSS World более открытой системой и позволяет взаимодействовать с другими приложениями, а также создавать пользователям свои библиотеки процедур. Язык Plus вместе с другими инструментальными средствами GPSS World позволил автоматизировать весь цикл исследований от разработки модели до выработки рекомендаций за счет новых функций планирования экспериментов и обработки статистики. И наконец, система моделирования GPSS World работает в операционной системе Windows и максимально ориентирована на использование современных технологий, обеспечивающих высокую интерактивность и визуальное представление информации.

Вместе с тем в настоящее время недостаточно учебного материала для изучения языков моделирования вообще и в частности GPSS World. Источники [22, 26] недоступны для широкого круга обучаемых. В [15] рассматривается Simulink, входящий в состав MatLab и предназначенный для создания моделей технических устройств. Предложение автора создавать посредством Simulink такие же модели, как и на GPSS, нельзя признать приемлемым. Известная система Arena хотя и выполняет функции, аналогичные функциям GPSS, по своим возможностям и гибкости уступает GPSS. Книги [17, 24], посвященные GPSS/PC и GPSS World, не восполняют имеющийся пробел, поскольку не раскрывают многие новые возможности GPSS World и связанные с ними методы разработки моделей. В названии [23] есть слово «технологии», однако технологии, т. е. методов построения моделей, — там также нет. Книга содержит описание средств GPSS World практически без показа на примерах, как с их помощью строить модели. В [25] дается лишь обзор инструментальных средств реализации моделей, среди которых GPSS World нет.

Настоящее учебное пособие соответствует дисциплине «Моделирование систем». В нем раскрываются методы построения моделей с применением инструментальных средств современной мощной системы моделирования GPSS World.

В первой главе излагаются основы построения и принципы функционирования языка имитационного моделирования. Во второй, четвертой, пятой, шестой и седьмой главах на многочисленных примерах, разработанных автором, раскрываются методы построения моделей процессов, которые формализуются в виде схем массового обслуживания. В третьей главе описываются приемы создания объектов GPSS World и эксплуатации моделей. В восьмой главе описан язык Plus и показано его применение, в том числе и для построения моделей непрерывных процессов, а девятая глава посвящена методам проведения различных видов экспериментов.

Автор выражает благодарность кандидату технических наук Девяткову Владимиру Васильевичу, директору ООО «Элина-Компьютер», руководителю работ по локализации GPSS World на русский язык [22, 26], встреча с которым на презентации GPSS World в Санкт-Петербурге и послужила окончательным толчком к написанию данной книги. Важным для автора является и отзыв Владимира Васильевича: «Я прочитал присланные Вами материалы по Вашей книге. Мне они понравились, чувствуется, что это не просто написано, а все проверено на множестве моделей. С терминологией я согласен. Надеюсь, что книга будет положительно воспринята».

Замыслу структуры учебного пособия, методологии изложения материала некоторых глав, а также моральной поддержке в период работы над книгой автор обязан первому постановщику дисциплин «Моделирование систем» и «Компьютерное моделирование» [16] в Военном университете связи кандидату технических наук, доценту Сыпченко Р. П. Спасибо, Руслан Павлович!

Постоянную помощь в подготовке рукописи автору оказывал Боев Д. В., который выполнил и компьютерную верстку книги, за что автор благодарен ему.

Учебную версию GPSS World бесплатно можно получить на портале www.minutemansoftware.com/download, а русифицированную (русификация выполнена автором пособия) — у автора по адресу e-mail: kirasir@rambler.ru.

1. Основы построения языка имитационного моделирования

1.1. Объекты системы моделирования GPSS World

В системе моделирования GPSS World различают четыре вида объектов (рис. 1.1.): модель, процесс моделирования, отчет и текстовый.

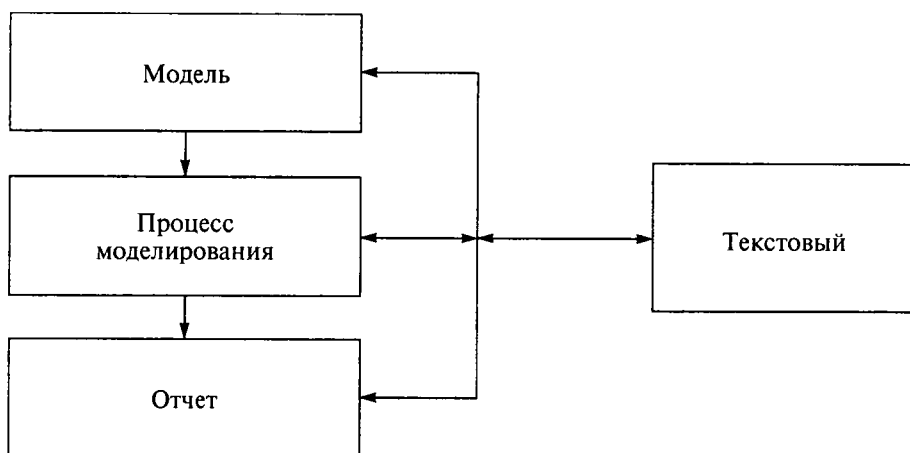


Рис. 1.1. Объекты GPSS World

Модель разрабатывается на языке GPSS и состоит из операторов, а объект «Модель» создается при помощи встроенного текстового редактора. Объект «Процесс моделирования» — это результат трансляции модели, получаемый после выполнения команды меню Create Simulation (Создать процесс моделирования). Далее процесс моделирования запускается с помощью команд GPSS. По завершении моделирования, как правило, автоматически создается объект «Отчет».

Текстовый объект (текстовый файл GPSS World) предназначен для упрощения разработки больших моделей и создания библиотеки исходных текстов, т. е. модель может быть разделена на наборы операторов, представляющие собой отдельные текстовые файлы, а затем объектом «Процесс моделирования» собрана из них. Объект «Процесс моделирования» может также создавать но-

вые текстовые файлы с фрагментами модели, результатами моделирования, а также считывать и записывать данные в текстовые файлы.

Модель, как уже отмечалось, разрабатывается на языке GPSS. Перейдем к его рассмотрению.

1.2. Объекты языка имитационного моделирования

Система GPSS World предназначена для имитационного моделирования систем с дискретными и непрерывными процессами. Языком моделирования в ней является язык GPSS, улучшенный встроенным языком программирования низкого уровня PLUS. Язык GPSS построен в предположении, что модель сложной системы можно представить совокупностью элементов и логических правил их взаимодействия в процессе функционирования моделируемой системы. Также предполагается, что для моделируемых систем можно выделить небольшой набор абстрактных элементов, называемых объектами. Причем набор логических правил ограничен и может быть описан небольшим числом стандартных операций. Комплекс программ, описывающих функционирование объектов и выполняющих логические операции, является основой для создания программной модели системы данного класса.

Кроме этого комплекса в составе системы GPSS World имеется специальная программа-планировщик, которая выполняет следующие функции:

- обеспечение продвижения по заданным разработчиком маршрутам динамических объектов, называемых транзактами;
- планирование событий, происходящих в модели, путем регистрации времени наступления каждого события и выполнения их в нарастающей временной последовательности;
- регистрация статистической информации о функционировании модели;
- продвижение модельного времени в процессе моделирования системы.

Чтобы обеспечить правильную последовательность обработки событий во времени, имеются системные часы, хранящие значения абсолютного модельного времени. Все отрезки времени описываются положительными числами.

Объекты в моделируемой системе предназначены для различных целей. Выбор объектов в моделируемой системе зависит от характеристик модели и, в некоторых случаях, от разработчика модели. Совершенно не обязательно, чтобы в одной модели участвовали все типы объектов. Необходимо лишь наличие блоков и транзактов, иначе модель работать не будет.

Объекты подразделяются на 7 *категорий* и 15 *типов*, которые представлены в табл. 1.1. Рассмотрим назначение объектов GPSS.

Динамическими объектами являются транзакты, которые создаются в определенных точках модели, продвигаются планировщиком через блоки, а затем уничтожаются. Транзакты являются аналогами единиц — потоков в реальной системе. Они могут представлять собой различные элементы даже в одной сис-

теме. Например, в модели боевых действий войск одни транзакты могут являться аналогами средств нападения противника, другие — зенитными управляемыми ракетами, третьи — вооружением и техникой войск, транспортными средствами, четвертые — донесениями. С каждым транзактом связаны параметры, которые используются для конкретных данных. В приведенных выше примерах параметрами транзактов могут быть: тип вооружения, принадлежность, техническое состояние, количество знаков в донесении и др. Каждый транзакт может иметь любое число параметров. Параметры нумеруются или им даются имена. Номера параметров и имена используются для ссылок на значения, присвоенные параметрам. Транзактам может присваиваться приоритет. Приоритет определяет предпочтение, которое получает транзакт, когда он и другие транзакты претендуют на один и тот же ресурс. Это может быть, например, категория срочности при передаче сообщений или очередность эвакуации и ремонта вооружения и техники в зависимости от их важности.

Таблица 1.1

Объекты GPSS

Категории	Типы объектов
Динамическая	Транзакты
Операционная	Блоки
Аппаратная	Одноканальные устройства, памяти (многоканальные устройства), логические ключи
Вычислительная	Переменные, функции, генераторы случайных чисел
Статистическая	Очереди, таблицы
Запоминающая	Ячейки, матрицы ячеек
Группирующая	Числовые группы, группы транзактов, списки

Объекты аппаратной категории — это абстрактные элементы, на которые может быть декомпозирована реальная система. Воздействуя на эти объекты, транзакты могут изменять их состояние и влиять на движение других транзактов. К объектам этого типа относятся одноканальные устройства, памяти (многоканальные устройства) и логические ключи.

Одноканальные устройства (ОКУ) представляют собой оборудование, которое в любой момент времени может быть занято только одним транзактом. Например, один канал передачи данных, одноканальный ремонтный орган, один узел связи.

Многоканальные устройства (МКУ) предназначены для имитации оборудования, осуществляющего параллельную обработку. Они могут быть использованы одновременно несколькими транзактами. МКУ можно использовать в качестве аналога, например, многоканального ремонтного органа, нескольких каналов связи.

Ранее происходившие события в системе могут заблокировать, изменить движение транзактов и наступление последующих событий. Например, один канал связи вышел из строя и все последующие заявки на передачу сообщений должны быть направлены на исправные каналы связи. Для моделирования таких ситуаций введены *логические ключи*. Транзакт может устанавливать эти ключи в положение «включено» или «выключено». Впоследствии состояние ключей может быть проверено другими транзактами для определения пути их дальнейшего следования.

Операционные объекты, т. е. блоки, задают логику функционирования модели системы и определяют пути движения транзактов между объектами аппаратной категории. В блоках могут происходить события четырех основных типов:

- 1) создание или уничтожение транзактов;
- 2) изменение числового атрибута объекта;
- 3) задержка транзакта на определенный период времени;
- 4) изменение маршрута движения транзакта в модели.

Версия GPSS, реализованная в системе GPSS World, содержит 53 типа блоков. В зависимости от назначения блоки подразделяются на несколько групп.

1. Блоки, осуществляющие модификацию атрибутов транзактов:
 - а) генерирование и уничтожение транзактов GENERATE, SPLIT, TERMINATE, ASSEMBLE;
 - б) временная задержка ADVANCE;
 - в) синхронизация движения двух MATCH и нескольких GATHER транзактов;
 - г) изменение параметров транзактов ASSIGN, INDEX, MARK, PLUS;
 - д) изменение приоритета транзакта PRIORITY.
2. Блоки, изменяющие последовательность движения транзактов (блоки передачи управления): DISPLACE, TRANSFER, LOOP, TEST, GATE.
3. Блоки, связанные с группирующей категорией: ADOPT, ALTER, EXAMINE, JOIN, REMOVE, SCAN.
4. Блоки, описывающие объекты аппаратной категории:
 - а) одноканальные устройства (технические средства) SEIZE, RELEASE, PRE-EMPT, RETURN, FUNAVAIL, FAVAIL;
 - б) многоканальные устройства (памяти) ENTER, LEAVE, SAVAIL, SUNAVAIL;
 - в) ключи (логические переключатели) LOGIC.
5. Блоки, сохраняющие необходимые значения для дальнейшего использования: SAVEVALUE, MSAVEVALUE.
6. Блоки, обеспечивающие получение статистических результатов:
 - а) очереди QUEUE, DEPART;
 - б) таблицы TABULATE.
7. Блоки для организации списка пользователя: LINK, UNLINK.
8. Блоки для организации ввода-вывода:

- а) открытие/закрытие файла: OPEN/CLOSE;
 - б) считывание/запись в файл: READ/WRITE;
 - в) установка позиции текущей строки: SEEK.
9. Специальные блоки: BUFFER, COUNT, EXECUTE, INTEGRATION, SELECT, TRACE, UNTRACE.

Вычислительная категория служит для описания таких ситуаций в процессе моделирования, когда связи между компонентами моделируемой системы наиболее просто и компактно выражаются в виде математических (аналитических и логических) соотношений. Для этих целей в качестве объектов вычислительной категории введены арифметические и булевы переменные и функции.

Переменные представляют собой сложные выражения, которые включают константы, системные числовые атрибуты (СЧА), библиотечные арифметические функции, арифметические и логические операции.

Выражения могут применяться в переменных и операторах GPSS. При применении в переменных выражения определяются командами GPSS. При применении в операторах GPSS выражения определяются как часть языка PLUS.

Каждому объекту соответствуют атрибуты, описывающие его состояние в данный момент времени. Они доступны для использования в течение всего процесса моделирования и называются *системными числовыми атрибутами* (СЧА). Например, объект вычислительной категории — генератор случайных чисел имеет СЧА RNn — число, вычисляемое генератором равномерно распределенных случайных чисел номер n ; у объекта динамической категории — транзакта СЧА: PR — приоритет обрабатываемого в данный момент транзакта; Pi — значение i -го параметра активного транзакта и др. Всего в GPSS World имеется свыше 50 СЧА.

Булевы переменные позволяют пользователю проверять в одном блоке GPSS одновременно несколько условий, исходя из состояния или значения этих условий и их атрибутов, т. е. в данном блоке производится обращение к булевой переменной, выражение которой содержит в себе проверку нескольких условий. Булевы переменные могут быть представлены комбинациями стандартных числовых атрибутов, связанных между собой посредством булевых операторов, включая и другие переменные. Булевы переменные определяются так же, как и арифметические, но вместо арифметических операций проверяются различные логические условия.

С помощью функций пользователь может производить вычисления непрерывных или дискретных функциональных зависимостей между аргументом функции (независимая величина) и зависимым значением функции. Все функции в GPSS задаются табличным способом с помощью команд описания функций. Как и переменные, функции не связаны с определенными блоками.

Кроме библиотечных арифметических функций GPSS World имеет 24 *встроенных генератора случайных чисел*.

Объекты запоминающей категории обеспечивают обращения к сохраняемым значениям. Ячейки сохраняемых величин и матрицы ячеек сохраняемых величин используются для сохранения некоторой числовой информации. Например, значение, занесенное в ячейку, может представлять собой коэффициент использования устройства в какой-то момент времени. Любой активный транзакт может произвести запись информации в эти объекты. Впоследствии записанную в эти объекты информацию может считать любой транзакт. Матрицы могут иметь до шести измерений.

К *статистическим объектам* относятся очереди и таблицы. В любой системе движение потока транзактов может быть задержано из-за недоступности устройств. Например, требуемое ОКУ (канал связи) может быть уже занято или МКУ (каналы связи), в которые нужно войти, уже заполнены. В этом случае задержанные транзакты ставятся в *очередь* — еще один тип объектов GPSS. Учет этих очередей составляет одну из основных функций планировщика. Планировщик автоматически накапливает определенную статистику относительно устройств и очередей. Кроме этого пользователь может собирать дополнительную статистическую информацию, указав специальные точки в модели.

Для облегчения табулирования статистической информации в GPSS предусмотрен специальный объект — *таблица*. Таблицы используются для получения выборочных распределений некоторых случайных величин. Таблица состоит из частотных классов (диапазонов значений), куда заносится число попаданий конкретного числового атрибута в каждый, тот или иной, частотный класс. Для каждой таблицы вычисляется также математическое ожидание и среднеквадратическое отклонение. Эта статистика является стандартной для всех таблиц. В конце эксперимента с моделью результаты, содержащиеся в таблицах, могут выводиться либо на дисплей, либо на печать.

К *группирующей категории* относятся три типа объектов: числовая группа, группа транзактов и списки.

В некоторых моделях транзакты представляют собой объекты, характеризующиеся общими СЧА, которые обычно представляются как параметры транзакты и меняются при прохождении транзактов через блок ASSIGN. Однако это дает ограниченный доступ к атрибутам, а иногда приходится одновременно менять некоторые атрибуты всех транзактов данного множества. Объекты такого типа предоставляют возможность пользователю обращаться к атрибутам транзактов данной *группы*. Они работают в одном из режимов: в режиме транзакта или числовом. Режим определяется первым обращением к определенной группе. После того как он установлен, необходимо его поддерживать. В режиме транзакта величины, поступающие в группу, представляются номерами по очереди транзактов данной группы. В другом режиме поступающие величины представляются группой числовых значений.

При моделировании транзакты хранятся в списках. Существует пять видов списков, только в одном из которых в любой момент времени может нахо-

даться транзакт: 1) текущих событий; 2) будущих событий; 3) задержки ОКУ или МКУ; 4) отложенных прерываний ОКУ; 5) пользователя.

В *списке текущих событий* (СТС) находятся транзакты, соответствующие событиям, время наступления которых меньше (транзакты, которые должны были начать двигаться в некоторый момент в прошлом, но были заблокированы) или равно текущему времени. Транзакты из СТС готовы к входу в блоки и должны войти в них до очередного изменения модельного времени. В СТС транзакты расположены в порядке убывания приоритета. Транзакты с одинаковыми приоритетами располагаются в порядке поступления их в список.

Список будущих событий (СБС) содержит транзакты, соответствующие событиям, время наступления которых больше текущего времени, т. е. событиям, которые должны произойти в будущем. Такие транзакты размещаются в списке строго в порядке возрастания времени начала движения. Приоритеты не влияют на порядок движения в этом списке.

В *список повторных попыток* помещаются транзакты, для которых не выполнены условия входа в следующий блок. Эти условия проверяются при попытке входа транзакта в блоки GATE, TEST, TRANSFER ALL и TRANSFER BOTH. Транзакты из списка повторных попыток ожидают изменения СЧА. Когда СЧА изменяется, транзакт активируется, проверяются условия входа его в следующий блок. Если условия выполняются, транзакт входит в следующий блок и автоматически исключается из списка повторных попыток.

Одноканальное устройство имеет:

- *список отложенных прерываний* — список транзактов, ожидающих занятия ОКУ по приоритету;
- *список прерываний* — список транзактов, обслуживание которых данным ОКУ было прервано;
- *список задержки* — список транзактов, ожидающих занятия ОКУ в порядке приоритета;
- *список повторных попыток* — список транзактов, ожидающих изменения состояния ОКУ.

Многоканальное устройство имеет:

- *список задержки* — список транзактов в порядке приоритета, ожидающих возможность занять освободившиеся каналы МКУ;
- *список повторных попыток* — список транзактов, ожидающих изменения состояния МКУ.

Список синхронизируемых транзактов имеют блоки ASSEMBLE, GATHER, MATH. Он содержит транзакты, находящиеся в данный момент времени в состоянии сравнения.

Список пользователя содержит транзакты, удаленные пользователем из списка текущих событий и помещенные в список пользователя как временно неактивные. Списки пользователя используются для организации очередей

с дисциплинами, отличными от дисциплины «первым пришел — первым обслужен».

Более подробно использование списков транзактов будет изложено в дальнейшем при рассмотрении методов построения моделей с использованием блоков GPSS World.

ЗАМЕЧАНИЕ

Термин CHAIN в [27, 17, 23] переведен с английского как «цепь». В [1, 3, 17, 22, 26], а также в данной книге вместо «цепь» используется «список». Поясним почему.

Представьте себе, например, что у вас есть золотая цепочка, состоящая из какого-то числа звеньев. За все время пользования этой цепочкой у вас даже и мысли не возникнет о том, чтобы каждые там день или два, а то и чаще разбирать ее и менять звенья местами. Да и при покупке цепочки вы не поставите жесткое условие: цепочка должна быть обязательно разборной. Скорее наоборот.

Проанализируем теперь процесс моделирования. Предположим, что перед ОКУ находится несколько транзактов, ожидающих его освобождения. В любой момент времени поступают новые транзакты, которые в общем случае вызывают перестроение уже ожидающих транзактов. То же самое может происходить и после занятия каким-либо транзактом освободившегося ОКУ. Кроме того, один или более из ожидающих транзактов могут удаляться по каким-либо причинам, не заняв ОКУ. И это повторяется многократно. Отсюда транзакты, находящиеся перед ОКУ (да и в любом другом случае, когда возникает нехватка ресурсов), ассоциируются, по нашему мнению, со списком, а не с цепью.

После того как система, модель которой нужно разработать, формализована, ее нужно описать на языке GPSS, используя блоки, которые выполняют соответствующие операции в модели.

1.3. Порядок представления моделей в виде блок-диаграмм

Модели систем на GPSS могут быть первично описаны в виде блок-диаграмм. *Блок-диаграмма* представляет собой набор с характерным очертанием блоков, соединенных между собой линиями. Вид каждого из блоков стандартен (приложение 1). Модель строится следующим образом. Из допустимого множества блоков выбирают необходимые и далее выстраивают их в диаграмму для того, чтобы в процессе функционирования модели они как бы взаимодействовали друг с другом. Использование блоков при построении моделей зависит от логических схем работы реальных систем, моделируемых на ЭВМ. При этом отдельные элементы модели и модель в целом имеют достаточно различимое подобие. Оно может быть также усилено разработчиком за счет более продуманного на этапе разработки разделения исследуемого объекта и модели на сегменты.

Типичный вид блок-диаграммы на GPSS представлен на рис. 1.2.

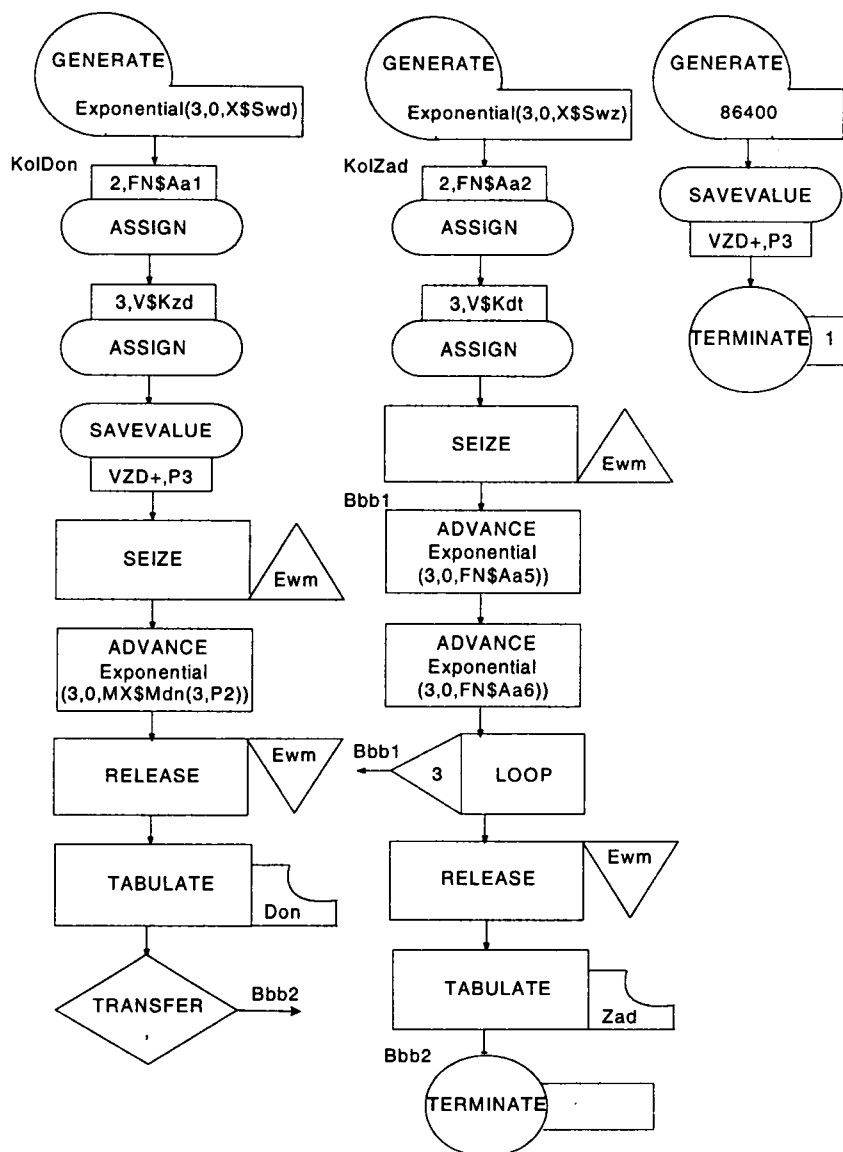


Рис. 1.2. Блок-диаграмма модели обработки информации

Конфигурация блок-диаграммы модели отражает направления, по которым происходит движение перемещающихся элементов — транзактов. Работа модели заключается в перемещении транзактов от блоков к блокам.

В самом начале моделирования в модели нет ни одного транзакта. В процессе моделирования транзакты входят в модель в определенные моменты времени в соответствии с теми логическими потребностями, которые возникают в мо-

делируемой системе. Подобным же образом транзакты покидают модель в определенные моменты времени в зависимости от специфики моделирования.

В общем случае в модели существует большое число транзактов, однако в определенный момент времени движется только один транзакт. Если транзакт начал движение, он перемещается от блока к блоку по пути, предписанному блок-диаграммой. Каждый блок можно рассматривать как некоторую точку, в которой происходит обращение к подпрограмме. В тот момент, когда транзакт входит в блок, на исполнение вызывается соответствующая подпрограмма, и далее транзакт пытается войти в следующий блок. Такое продвижение транзакта продолжается до тех пор, пока не произойдет одно из следующих событий:

- транзакт входит в блок, функцией которого является задержка транзакта на некоторое определенное модельное время;
- транзакт входит в блок, функцией которого является удаление транзакта из модели;
- транзакт пытается войти в следующий блок в соответствии с предписанной блок-диаграммой логикой, однако блок отказывается принять этот транзакт;
- транзакт остается в том блоке, в котором он в настоящее время находится, и будет повторять свои попытки войти в следующий блок.

Когда условия в модели изменятся, одна из попыток может оказаться успешной и транзакт сможет продолжить свое движение в модели.

Если возникло одно из перечисленных условий, транзакт остается на месте и начинается перемещение в модели другого транзакта. Выполнение модели заключается в последовательном обращении к подпрограммам, являющимся следствием входа в определенные блоки движущихся транзактов. Проходя через блоки, каждый транзакт вносит свой вклад в содержимое *счетчиков блоков*. Значения этих счетчиков доступны разработчику через СЧА блоков: *W* — текущее содержимое блока и *N* — общее количество входов в блок.

Каждое продвижение транзакта в модели является *событием*, которое должно произойти в определенный момент модельного времени. Для того, чтобы поддерживать правильную временную последовательность событий, планировщик имеет *таймер* модельного времени, который автоматически корректируется в соответствии с логикой, предписанной моделью.

Таймер GPSS имеет следующие особенности:

- регистрируются только положительные вещественные значения времени;
- единица модельного времени (масштаб) определяется разработчиком модели, который задает все временные интервалы в одних и тех же единицах;
- планировщик не анализирует состояние модели в каждый следующий момент модельного времени (отстоящий от текущего на единицу модельного времени), а продвигает таймер к моменту времени, когда должно произойти ближайшее следующее событие.

Значения таймера доступны разработчику через системные СЧА C1 (*относительное модельное время*) и AC1 (*абсолютное модельное время*).

Главной задачей, выполняемой планировщиком, является определение того, какой транзакт надо выбрать следующим для продвижения в модели, когда предыдущий транзакт прекратил свое продвижение. С этой целью планировщик рассматривает каждый транзакт как элемент некоторого списка.

Планировщик GPSS помещает транзакты в зависимости от условий в модели в тот или иной список, просматривает списки, выбирает следующий транзакт для обработки, корректирует таймер модельного времени после обработки всех транзактов в списке текущих событий.

Вернемся к рис. 1.2. На нем показаны три блока GENERATE, которые являются источниками транзактов. Такие блоки являются точками, в которых транзакты появляются в модели. Два блока TERMINATE являются конечными, т. е. блок-диаграмма на этих блоках завершается и из нее нет выхода. Эти блоки являются точками вывода транзактов из модели. Блок-диаграмма состоит из трех частей — сегментов. Один сегмент, состоящий из трех блоков, является независимым, а два других взаимосвязаны друг с другом. В процессе моделирования активным является тот из сегментов, в котором находится перемещающийся в настоящее время транзакт. Когда он блокируется, начинается движение следующего транзакта, который вполне возможно, принадлежит другому сегменту модели.

Содержательное значение транзактов определяет разработчик модели. Он устанавливает аналогию между транзактами и реальными элементами моделируемой им системы. Такая аналогия никогда не указывается планировщиком GPSS. Она присутствует только в памяти разработчика модели. Например, в модели системы технического обеспечения транзакты могут символизировать средства связи, автоматизации, отдельные транспортные средства, вооружение и технику, которые применяются, выходят из строя, эвакуируются при необходимости в ремонтный орган, ремонтируются и возвращаются в строй. В модели системы технического обеспечения на GPSS для обозначения всех этих процессов можно использовать различные блоки. Перемещение транзактов в этой модели от блока к блоку, таким образом, аналогично применению вооружения и техники.

На рис. 1.2 представлена блок-диаграмма модели системы сбора и обработки информации (донесений, заявок и др.) с применением средств автоматизации и выработке на основе этого управленческих воздействий. Помимо задач сбора информации на ЭВМ решаются различные задачи по ее обработке. Аналогами транзактов являются донесения, заявки и задачи. Построению блок-диаграммы должна предшествовать достаточно детальная разработка информационно-модели с указанием всех источников информации, различных документов, путей их передачи, средств и способов обработки. Степень детализации информационной модели зависит от целей моделирования.

Подробное описание системы сбора и обработки информации приведено в п. 6.8. Поэтому к блок-диаграмме на рис. 1.2 следует вернуться в ходе изучения материала п. 6.8.

1.4. Структура модели на языке GPSS

После того как блок-диаграмма построена, она должна быть записана в форме, удобной для создания объекта «Модель». Для этого информация об объектах GPSS записывается в виде последовательности операторов. Оператор модели в GPSS World — это либо оператор GPSS, либо PLUS-оператор. Операторы GPSS в свою очередь — это или операторы блоков, которые создают блоки, или команды, которые не создают блоков.

В отличие от PLUS-операторов, которые могут содержать несколько строк, операторы GPSS должны записываться одной текстовой строкой длиной не более 250 символов. Операторы GPSS состоят из частей, называемых полями. Поле — это набор символов, отделенный пробелами или ограничителем. В общем случае оператор состоит из следующих полей: номер строки (необязательно) | метка (необязательно) | глагол (обязательно) | операнды (зависит от глагола) | комментарий (обязательно).

Номера строк оставлены для совместимости с GPSS/PC и игнорируются GPSS World. Операторы, несмотря на установленный для каждого формат, имеют свободную форму записи в том смысле, что нет необходимости стремиться к выравниванию полей операторов по колонкам.

Для различения объектов GPSS, например, ОКУ, МКУ, очередей и т. д., которых в модели может быть несколько, им дают имена. Имена должны начинаться с буквы, могут содержать до 200 букв и цифр, а также символы подчеркивания. Имя не должно совпадать с ключевыми словами GPSS. Одно и то же имя можно использовать для различных объектов.

В процессе моделирования возможны переходы от одного объекта модели к другому. С целью осуществления этого объект можно идентифицировать меткой, для чего служит поле метки. Метки образуются так же, как и имена блоков. GPSS присваивает меткам номера, начиная с 10 000. Пользователь не может сразу записать вместо метки номер. Если он хочет это сделать, нужно перед обращением к объекту метке командой EQU присвоить номер, который не может быть больше 9999. В качестве меток также нельзя использовать ключевые слова GPSS.

В поле глагола пользователь должен записать ключевое слово GPSS, т. е. или блок, или команду. Задание исходных данных, необходимых для выполнения операций, соответствующих блоку или команде, производится в поле операндов. Операнды разделяются запятыми.

Команды используются для описания многоканальных устройств, переменных, матриц ячеек, функций, таблиц и управления процессом моделирования.

Ниже приведена модель, написанная согласно рассмотренной в п. 1.3 блок-диаграмме (см. рис. 1.2). Модель имеет один объект «Модель» и два текстовых объекта — текстовых файла DanZad.txt и DanDon.txt, подключаемых к сегменту задания исходных данных объекта «Модель» командой INCLUDE. Этот сегмент содержит команды описания функций, таблиц, матриц и арифметических переменных, а остальные сегменты — операторы блоков GPSS.

```
; Primer 6.1
; Сегмент задания исходных данных
VrMod EQU      86400      ; Время моделирования
;   Определение функций для расчета данных о задачах
INCLUDE      "DanZad.txt"; Подключение файла данных о задачах
;   Определение таблиц
Don  TABLE    M1,120,300,40 ; Таблица времени обработки донесений
Zad  TABLE    M1,3600,600,50 ; Таблица времени решения задач
ObVremTABLE    M1,1600,600,50 ; Таблица общего времени обработки
;   Инициализация матрицы – запись данных о донесениях
INCLUDE      "DanDon.txt"; Подключение файла данных о донесениях
;   Определение арифметических переменных для расчета
;   количества знаков в донесениях
Kzd  VARIABLE   INT(Normal(5,MX$TDon(1,P2),MX$TDon(2,P2)))
;   количество точек диалога
Kdt  VARIABLE   INT(Normal(7,FN$TZad2,FN$TZad3))
Skzd  VARIABLE   X$Vzd/N$KolDon ; Среднее число знаков в донесении
; Сегмент имитации обработки донесений
GENERATE      (Exponential(3,0,Swz)) ; Источник донесений
KolDonASSIGN  2,FN$TDon1 ; Определение вида донесения
ASSIGN        3,V$Kzd ; Определение количества знаков в донесении
SAVEVALUE     Vzd+,P3 ; Счет знаков в донесениях
SEIZE         Ewm ; Занятие ЭВМ
ADVANCE       (Exponential(3,0,MX$TDon(3,P2))); Обработка донесения
RELEASE       Ewm ; Освобождение ЭВМ
TABULATE      Don ; Табулирование времени обработки донесений
TRANSFER      ,Bbb2
; Сегмент имитации решения задач
GENERATE      (Exponential(3,0,Swz)) ; Источник задач
ASSIGN        2,FN$TZad1 ; Определение типа задачи
ASSIGN        3,V$Kdt ; Определение числа точек диалога
Bbb1 SEIZE     Ewm ; Занятие ЭВМ
ADVANCE       (Exponential(3,0,FN$TZad4)) ; Ведение диалога
ADVANCE       (Exponential(3,0,FN$TZad5)) ; Обработка после диалога
RELEASE       Ewm ; Освобождение ЭВМ
LOOP          3,Bbb1 ; Организация цикла по числу точек диалога
TABULATE      Zad ; Табулирование времени решения задач
Bbb2 TABULATE  ObVrem ; Табулирование общего времени обработки
; донесений и задач
TERMINATE     ; Вывод транзактов из модели
```

```
; Сегмент задания времени моделирования
; и расчета результатов моделирования
    GENERATE   VrMod ;Задание времени моделирования 1 ед. мод. вр.=1 с
    SAVEVALUE  Skzd, V$Skzd          ;Расчет среднего количество знаков
;                                     в донесении
    TERMINATE  1          ;Число, на которое уменьшается
;                             содержимое счетчика завершения
```

Текстовый файл DanZad.txt, содержащий данные о задачах.

```
; Данные о задачах
Swz   EQU      1800          ;Среднее время решения задач
.1, 1/.23, 2/.31, 3/.43, 4/.52, 5/.69, 6/.82, 7/1, 8
TZad1 FUNCTION  RN1, D6
.14, 1/.31, 2/.46, 3/.63, 4/.81, 5/1, 6
TZad2 FUNCTION  P2, D6
1, 23/2, 13/3, 29/4, 8/5, 19/6, 31
TZad3 FUNCTION  P2, D6
1, 4/2, 2/3, 4/4, 1/5, 3/6, 5
TZad4 FUNCTION  P2, D6
1, 25/2, 62/3, 78/4, 115/5, 93/6, 137
TZad5 FUNCTION  P2, D6
1, 9/2, 13/3, 7/4, 5/5, 11/6, 16
```

Текстовый файл DanDon.txt, содержащий данные о донесениях.

```
; Данные о донесениях
Swd   EQU      3600          ;Среднее время поступления донесений
Mdn   MATRIX    , 3, 8       ;Матрица данных о донесениях
INITIAL MX$TDon(1, 1), 420
INITIAL MX$TDon(1, 2), 510
INITIAL MX$TDon(1, 3), 630
INITIAL MX$TDon(1, 4), 920
INITIAL MX$TDon(1, 5), 1200
INITIAL MX$TDon(1, 6), 730
INITIAL MX$TDon(1, 7), 340
INITIAL MX$TDon(1, 8), 480
INITIAL MX$TDon(2, 1), 60
INITIAL MX$TDon(2, 2), 90
INITIAL MX$TDon(2, 3), 120
INITIAL MX$TDon(2, 4), 170
INITIAL MX$TDon(2, 5), 230
INITIAL MX$TDon(2, 6), 140
INITIAL MX$TDon(2, 7), 60
INITIAL MX$TDon(2, 8), 80
INITIAL MX$TDon(3, 1), 8
INITIAL MX$TDon(3, 2), 11
INITIAL MX$TDon(3, 3), 6
INITIAL MX$TDon(3, 4), 16
```

```

INITIAL      MX$TDon(3,5), 10
INITIAL      MX$TDon(3,6), 13
INITIAL      MX$TDon(3,7), 3
INITIAL      MX$TDon(3,8), 9

```

Объект «Модель» состоит из четырех сегментов: задания исходных данных; имитации обработки донесений; имитации решения задач; задания времени моделирования и расчета результатов моделирования.

Назначение и количество сегментов определяются разработчиком модели. Объект «Модель» является обязательным, а текстовые объекты — необязательными. Их назначение, содержание и количество также определяются разработчиком. Например, если сегменты имитации обработки донесений и решения задач представить также двумя текстовыми объектами — текстовыми файлами ObrDon.txt и ObrZad.txt соответственно, объект «Модель» будет представлен так.

```

; Primer 6.1
; Сегмент задания исходных данных
; Данные о задачах
    INCLUDE      "DanZad.txt"      ;Подключение файла данных о задачах
; Определение таблиц
Don  TABLE      M1,120,300,40      ;Таблица времени обработки донесений
Zad  TABLE      M1,3600,600,50     ;Таблица времени решения задач
ObvremTABLE      M1,1600,600,50     ;Таблица общего времени обработки
;                                     донесений и задач
; Данные о донесениях
    INCLUDE      "DanDon.txt"      ;Подключение файла данных о донесениях
; Определение арифметических переменных для расчета
;     количества знаков в донесениях
Kzd  VARIABLE     INT(Normal(5,MX$TDon(1,P2),MX$TDon(2,P2)))
;     количество точек диалога
Kdt  VARIABLE     INT(Normal(7,FN$TZad2,FN$TZad3))
Skzd VARIABLE     X$Vzd/N$KolDon    ;Среднее число знаков в донесении
; Сегмент имитации обработки донесений
    INCLUDE      "ObrDon.txt"      ;Подключение сегмента имитации
;                                     обработки донесений
; Сегмент имитации решения задач
    INCLUDE      "ObrZad.txt"      ;Подключение сегмента имитации
;                                     решения задач
; Сегмент задания времени моделирования
; и расчета результатов моделирования
    GENERATE      86400 ;Задание времени моделирования 1 ед. мод. вр.=1 с
    SAVEVALUE     Skzd,V$Skzd      ;Расчет среднего числа знаков в донесении
    TERMINATE     1                ;Число, на которое уменьшается содержимое
;                                     счетчика завершения

```

К данной модели нужно вернуться при изучении п. 6.8.

1.5. Объекты вычислительной категории

Как уже отмечалось, к объектам вычислительной категории относятся арифметические и булевы переменные и функции. Переменная — это обобщенное понятие, которое в конкретном случае представляет собой выражение. Выражение может включать следующие элементы:

- константы;
- системные числовые атрибуты;
- арифметические операции;
- логические операции;
- библиотечные функции;
- переменные пользователя.

Помимо применения в переменных, выражения в скобках могут также использоваться в операторах GPSS (некоторые операторы допускают применение и без скобок, но лучше использовать скобки).

Рассмотрим элементы выражений.

1.5.1. Константы

Константы — числа, представляются в одном из трех типов данных — целочисленном, вещественном или строковом. Так как арифметические операторы преобразуют свои операнды в подходящую форму, то в выражениях могут использоваться все три типа данных.

Целочисленные данные хранятся в виде 32-разрядных чисел. Если происходит переполнение, целое число преобразуется в вещественное.

Вещественные данные хранятся в виде 64-разрядных целых чисел с плавающей точкой двойной точности. Мантисса может изменяться от -308 до 308 , а точность ограничена 15 десятичными разрядами.

Строковые данные хранятся в массивах символов ASCII любой длины. Длина ограничена только значением Max Memory Request (Максимальный запрос памяти) (страница Simulation (Моделирование), настройки объекта «Процесс моделирования»). Эти настройки можно изменить с помощью меню Edit ► Settings (Правка ► Настройки).

1.5.2. Системные числовые атрибуты

В процессе моделирования GPSS World автоматически регистрирует и корректирует информацию, представленную системными числовыми атрибутами различных объектов, используемых в модели. Полный перечень СЧА, классифицированных по объектам GPSS, представлен в приложении 2.

Имена СЧА зарезервированы. По способу определения имен СЧА можно разбить на три группы.

Имя СЧА в первой группе состоит из двух частей. Первая часть указывает групповое имя, идентифицирующее тип объекта и тип информации об объекте. Вторая часть идентифицирует конкретного члена группы.

Групповое имя состоит из одной-двух букв, фиксированных для информации об объектах определенного типа, например, Q — ссылка на текущее значение длины очереди, FR — коэффициент использования устройства, SR — коэффициент использования памяти и т. д.

Объекты GPSS могут быть идентифицированы с помощью числовых или символьных имен. Идентификация СЧА зависит от вида адресации, которых в GPSS предусмотрено две: прямая и косвенная.

Рассмотрим *прямую адресацию*. Если объект идентифицирован с помощью номера, то ссылка на его СЧА записывается как СЧАj, где j — номер объекта (положительное целое число). Например, Q3 — текущее значение длины очереди номер 3, FR2 — коэффициент использования устройства номер 2. При указании имени объекта ссылка на его СЧА записывается как СЧА\$имя, где *имя* — имя объекта. Например, Q\$Basa — текущее значение длины очереди с именем Basa, SR\$Rem — коэффициент использования МКУ с именем Rem.

При косвенной адресации СЧА определяются как СЧА*параметр. Это означает следующее:

- СЧА*j, j — номер параметра активного транзакта (положительное целое число), содержащего номер нужного блока;
- СЧА*имя, имя — имя параметра активного транзакта, содержащего номер нужного блока.

Например, Q*3 — текущее значение длины очереди, номер которой является значением параметра 3 активного транзакта, SR*Rem — коэффициент использования памяти, номер которой содержится в параметре с именем Rem активного транзакта.

Вторую группу, как частный случай первой группы, составляют матричные СЧА. Матричный СЧА MX может содержать до трех идентификаторов (косвенных адресов). Например:

MX*Result(*Stroka,*Stolbez)

является ссылкой на матрицу, номер которой содержится в параметре активного транзакта с именем Result, а элемент матрицы — номер строки и номер столбца — определяется значениями параметров с именами Stroka и Stolbez соответственно того же активного транзакта.

Третью группу составляют СЧА A1, AC1, C1, M1, PR, TG1, XN1, Z1. В руководстве пользователя [22] они названы «атомарными». Эти СЧА, в отличие от СЧА первой и второй групп, не требуют указания номера или имени объекта.

СЧА по *признаку доступности к ним пользователя* можно разделить на две группы.

Первую группу составляют атрибуты, которые в части, касающейся их формирования, доступны только GPSS World, однако они могут быть использованы также и пользователем в выражениях, но не могут им изменяться в процессе моделирования. Эта группа содержит наибольшее число СЧА.

Во вторую группу входят СЧА вычислительных объектов, которые могут изменяться разработчиком модели и использоваться им в выражениях:

- FNj (FN\$Raspr) — вычисленное вещественное значение функции номер j (с именем Raspr);
- Vj (V\$Ver) — вычисленное значение вещественной или целочисленной переменной номер j (с именем Ver);
- BVj (BV\$Per) — вычисленное вещественное значение булевой переменной номер j (с именем Per).

1.5.3. Арифметические, условные и логические операторы

Арифметические, условные и логические операторы предназначены для указания операций, которые должны выполняться над элементами выражения. В GPSS World используются следующие операторы (табл. 1.2).

Если оператор требует определенного типа данных, данные преобразуются автоматически. Например, если над строковой величиной производится числовая операция, то будет использован числовой эквивалент этой строки символов.

Оператор «#» использован для операции умножения потому, чтобы применять оператор «*» для указания косвенной адресации. Однако пользователь может менять значение операторов «#» и «*». Для этого нужно воспользоваться флажком, имеющимся на странице Simulation (Моделирование) журнала настроек.

Операции над элементами выражения выполняются в следующей последовательности в порядке убывания приоритетов: $\wedge > \# / \backslash @ > - + > > = < = > < > > ! = > \& > |$.

Таблица 1.2

Оператор	Операция	Аргумент, значение
$\wedge$	Возведение в степень	A^B ; A, возведенное в степень B
#	Умножение	$A\#B$; произведение A на B
/	Деление	A/B ; частное от деления A на B
$\backslash$	Целочисленное значение	$A\backslash B$; частное от целочисленного деления A на B
@	Целый остаток	$A@B$; целый остаток от деления A на B
-	Вычитание	
+	Сложение	
$>='GE'$	Больше или равно	1, если $A \geq B$, иначе — 0
$<='LE'$	Меньше или равно	1, если $A \leq B$, иначе — 0
$>'G'$	Больше	1, если $A > B$, иначе — 0
$<'L'$	Меньше	1, если $A < B$, иначе — 0

Таблица 1.2 (продолжение)

Оператор	Операция	Аргумент, значение
= 'E'	Равно	1, если $A=B$, иначе — 0
!= 'NE'	Не равно	1, если $A \neq B$, иначе — 0
& 'AND'	Логическое «И»	1, если A и B не 0, иначе — 0
'OR'	Логическое «ИЛИ»	1, если A или B или оба не 0, иначе — 0

1.5.4. Библиотечные математические функции

Библиотека GPSS World имеет следующие математические функции (процедуры) (табл. 1.3).

Таблица 1.3

Оператор	Функции	Аргумент, значение
ABS(A)	Абсолютное значение	A^B ; A, возведенное в степень B
ATN(A)	Арктангенс	$A \# B$; произведение A на B
COS(A)	Косинус	A/B ; частное от деления A на B
EXP(A)	Экспонента	$A \setminus B$; частное от целочисленного деления A на B
INT(A)	Выделение целой части	$A @ B$; целый остаток от деления A на B
LOG(A)	Натуральный алгоритм	
SIN(A)	Синус	
SQR(A)	Квадратный корень	1, если $A \geq B$, иначе — 0
TAN(A)	Тангенс	1, если $A \leq B$, иначе — 0

Аргументы библиотечных функций во всех случаях автоматически преобразуются в числовые значения. Допустимые значения аргумента Expression (Выражение). Вычисляемые числовые значения функций — вещественного типа.

1.5.5. Библиотечные генераторы случайных чисел

Генераторы случайных чисел предназначены для моделирования случайных величин и случайных событий.

При пользовании табл. 1.4 нужно иметь в виду следующее:

- все аргументы генераторов имеют допустимое значение Expression;
- после вычисления аргумент преобразуется к соответствующему типу;
- символом «*» помечены аргументы, которые автоматически преобразуются в целое число;
- символом «**» помечены аргументы, автоматически преобразующиеся в вещественное число;

- аргумент А после вычисления выражения и преобразования в целое число используется в качестве номера генератора случайных чисел;
- генерирование случайных чисел производится по выражениям [22].

Таблица 1.4

Распределение, оператор, тип	Аргументы
Бета Beta(A,B,C,D,E) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; [B*,C*] — интервал генерируемых значений, $B > C$; D** и E** — первое и второе определяющие значения, использующиеся для выбора из семейства форм. Beta(A,B,C,1,1)=Uniform(A,B,C), Beta(A,B,C,1,2) — левый, Beta(A,B,C,2,1) — правый треугольники.
Биноминальное Binominal(A,B,C) Целый	A* — номер генератора случайных чисел, $A \geq 1$; B* — количество независимых испытаний Бернулли, $B > 0$; C** — вероятность успешных испытаний Бернулли, $C \in [0,1]$.
Дискретно-равномерное DUniform(A,B,C) Целый	A* — номер генератора случайных чисел, $A \geq 1$; [B*,C*] — интервал генерируемых значений, $B > C$.
Экспоненциальное Exponential(A,B,C) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$, B+C — среднее значение, при B=0 C — среднее значение.
Экстремального значения типа A ExtValA(A,B,C) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$.
Экстремального значения типа B ExtValB(A,B,C) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$.
Геометрическое Geometric(A,B) Целый	A* — номер генератора случайных чисел, $A \geq 1$; B** — вероятность успешных испытаний Бернулли, $B \in [0,1]$.

Таблица 1.4 (продолжение)

Распределение, оператор, тип	Аргументы
Гамма Gamma(A,B,C,D) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$; D** — определяющее значение, используемое для выбора из семейства форм, $D > 0$; Gamma(A,B,C,1)=Exponential(A,B,C).
Обратное Гаусса InvGauss(A,B,C,D) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$; D** — определяющее значение, используемое для выбора из семейства форм, $D > 0$.
Обратное Вейбулла InvWeibull(A,B,C,D) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$; D** — определяющее значение, используемое для выбора из семейства форм, $D > 0$.
Лапласа Laplace(A,B,C) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$.
Логлапласово LogLaplace(A,B,C,D) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$; D** — определяющее значение, используемое для выбора из семейства форм, $D > 0$.

Таблица 1.4 (продолжение)

Распределение, оператор, тип	Аргументы
Отрицательное биномиальное NegBinom(A,B,C) Целый	A* — номер генератора случайных чисел, $A \geq 1$; B* — количество успешных испытаний Бернулли, после которого возвращается число безуспешных испытаний, $B > 0$; C** — вероятность успешных испытаний Бернулли, $C \in [0,1]$.
Логистическое Logistic(A,B,C) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$.
Логлогистическое LogLogis(A,B,C,D) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$; D** — определяющее значение, используемое для выбора из семейства форм, $D > 0$.
Логнормальное LogNormal(A,B,C,D) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$; D** — определяющее значение, используемое для выбора из семейства форм, $D > 0$.
Парето Pareto(A,B,C) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$.
Пирсона типа 5 Pearson5(A,B,C,D) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$; D** — определяющее значение, используемое для выбора из семейства форм, $D > 0$.

Таблица 1.4 (продолжение)

Распределение, оператор, тип	Аргументы
Нормальное Normal(A,B,C) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — среднее значение распределения; C** — стандартное отклонение распределения, $C > 0$.
Пуассона Poisson(A,B) Целый	A* — номер генератора случайных чисел, $A \geq 1$; B** — среднее значение распределения.
Пирсона типа б Pearsonb(A,B,C,D,E) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; [B*,C*] — интервал генерируемых значений, $B > C$; D** и E** — первое и второе определяющие значения, использующиеся для выбора из семейства форм, $Beta(A,B,C,1,1) = Uniform(A,B,C)$.
Треугольное Triangular(A,B,C,D) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; [B*,C*] — интервал генерируемых значений, $B > C$; D** — мода распределения, $D \in [B,C]$.
Равномерное Uniform(A,B,C) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; [B**,C**] — интервал генерируемых значений, $B > C$, $Uniform(A,B,C) = Beta(A,B,C,1,1)$.
Вейбулла Weibull(A,B,C,D) Вещественный	A* — номер генератора случайных чисел, $A \geq 1$; B** — величина сдвига, определяющего местоположение распределения; C** — величина, используемая для сжатия или растяжения распределения, $C > 0$; D** — определяющее значение, использующееся для выбора из семейства форм, $D > 0$; $Weibull(A,B,C,1) = Exponential(A,B,C)$ $Weibull(A,B,C,2)$ — распределение Релея.

1.5.6. Переменные пользователя

GPSS предоставляет пользователю возможность иметь свои переменные для хранения нужных при моделировании как числовых, так и строковых данных. Переменные пользователя создаются с помощью команды EQU или присваивающих PLUS-процедур. Например:

Ver1	EQU	2.65
Stroka	EQU	11
Stolbez	EQU	8
Name	EQU	"Name"

В приведенных первых трех примерах переменным пользователя Ver1, Stroka, Stolbez присваиваются числовые значения, а переменной Name — строка.

С помощью переменных пользователя имена объектов GPSS можно заменять номерами. Для этого следует предварительно, как показано в примерах, определить переменные пользователя командой EQU.

Также имена, заданные командой EQU, но в модели использующиеся как метки объектов, считаются метками.

Переменные пользователя могут применяться в PLUS-выражениях, изменяться командами EQU или присваивающими операторами в PLUS-процедурах. При проведении эксперимента с имитационной моделью факторы, влияние которых на искомый показатель предполагается исследовать, обязательно должны быть переменными пользователя.

1.5.7. Выражения в операторах GPSS

Выражения в системе GPSS World определены как часть языка PLUS. Их можно использовать в операндах операторов GPSS и в PLUS-процедурах. В операндах, в которых допускается использование PLUS-выражений, они заключаются в скобки.

Следовательно, можно ввести фактор, какое-либо вычисление или вызов PLUS-процедуры прямо в операнд. Например:

```
ADVANCE    (VrUbit+(Exponential(3,Sred)))  
ASSIGN     3,((Normal(2,Sred,SrKv0tk))#2.45)
```

Кроме того, с помощью выражений, содержащих СЧА, можно получить доступ к переменным состояния объектов GPSS, таким, как глобальные переменные пользователя, из PLUS-процедур.

1.5.8. Сохраняемые ячейки

В GPSS используются ячейки памяти, начальные значения которых могут быть заданы перед моделированием и к которым можно обратиться из любого места модели во время моделирования. Эти ячейки называют *сохраняемыми*. Они являются СЧА. Значения сохраняемых ячеек изменяются только при прямом указании пользователя. Перед началом моделирования значения сохраняемых ячеек устанавливаются в нуль. По желанию пользователя отдельным сохраняемым ячейкам могут быть присвоены ненулевые начальные значения с помощью команды INITIAL. Команда INITIAL применяется при отсутствии активного транзакта или же когда нельзя использовать значения параметров транзактов. Она имеет следующий формат:

```
INITIAL    A,[B]
```

Операнд A при инициализации сохраняемой ячейки может быть *Хлопательное целое*, X\$имя. То есть в команде можно сразу указывать или имя, или номер, хотя, как узнаем позднее, не во всех командах это возможно.

Операнд B — присваиваемое первоначальное значение или UNSPECIFIED (не определено). Может быть именем, числом, строкой или UNSPECIFIED. Если операнд B не используется, то значение ячейки становится равным 1.

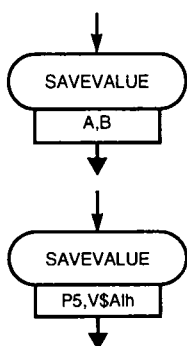
Например:

```
INITIAL  X32,3895.56
INITIAL  X$Def,Q$Pod
INITIAL  X$Cost,Zena1
INITIAL  X$Time,"Result number ..."
INITIAL  X1
```

В первом примере в ячейку номер 32 заносится число 3895.56, во втором — в ячейку с именем Def — текущая длина очереди с именем Pod, в третьем — в ячейку с именем Cost — значение переменной пользователя Zena1, в четвертом — в ячейку с именем Time заносится строковая константа, а в пятом — операнд B не используется, поэтому сохраняемой ячейке номер 1 присваивается значение 1.

В процессе моделирования значение сохраняемой ячейки изменяется при входе транзакта в блок SAVEVALUE (сохранить величину). Формат записи блока SAVEVALUE:

SAVEVALUE A, B



Когда транзакт входит в блок SAVEVALUE, величина, стоящая в операнде B, становится значением сохраняемой ячейки, номер или символическое имя которой записан в операнде A. Например, как только транзакт входит в этот блок, вычисляется значение переменной Alh. Полученный результат назначается сохраняемой ячейке, номер которой записан в параметре номер 5 активного транзакта. При этом старое значение сохраняемой ячейки уничтожается.

Операнд A может быть именем, положительным целым числом, выражением в скобках, СЧА*параметр или СЧА, а операнд B, кроме этого, может быть числом или строкой.

Блок SAVEVALUE может быть использован как в режиме замещения, так и в режиме накопления или уменьшения. В режиме накопления предыдущее значение сохраняемой ячейки увеличивается на значение, стоящее в операнде B.

В режиме уменьшения оно уменьшается на величину, стоящую в операнде B. Режимы накопления или уменьшения определяются введением соответственно знака + или — перед запятой, разделяющей операнды A и B. Например:

```
SAVEVALUE 5+,X2
SAVEVALUE Vad-,V$Hd1
SAVEVALUE Ais,-659
SAVEVALUE 3,(25.75#X$Num1+Q4)
SAVEVALUE 11,"Result numer..."
```

Значения операндов A изменяются только при входе транзактов в эти блоки. В первом примере значение сохраняемой ячейки номер 5 при входе транзакта в блок SAVEVALUE увеличивается на значение сохраняемой ячейки номер 2.

Во втором примере значение сохраняемой ячейки с именем Vad уменьшается на вычисленное значение арифметической переменной Hdl. В третьем примере значение сохраняемой ячейки с именем Ais замещается числом -659. В четвертом примере вычисляется выражение в скобках и присваивается ячейке номер 3. В пятом примере содержимое ячейки номер 11 замещается строкой. Если ячейки с таким именем не существует, она создается.

Сохраняемые ячейки имеют единственный СЧА с названием X, значением которого является текущее значение соответствующей сохраняемой ячейки. СЧА X в блоке SAVEVALUE при указании операнда A, как видно из примеров, не используется. При записи других операндов ссылка на СЧА X обязательна. Ссылка на этот СЧА применяется также в выражениях, функциях, в команде INITIAL.

1.5.9. Матрицы сохраняемых ячеек

Понятие сохраняемой ячейки в GPSS применяется и к матрицам. Эти объекты дают возможность упорядочить значения, которые нужно сохранить в виде *матриц*. Матрица должна быть предварительно описана. Команда описания матрицы имеет следующий формат:

```
Name      MATRIX      A, B, C, [D], [E], [F], [G]
```

Метка Name определяет имя матрицы и должна быть именем (не может быть числом). Операнд A не используется, так как оставлен для совместимости с более старыми версиями GPSS. Операнды B, C, D, E, F, G могут быть только целыми положительными числами.

Операндом B задается число строк матрицы — максимальное количество элементов в первом измерении, операндом C — число столбцов матрицы — максимальное количество элементов во втором измерении. Операнды D, E, F, G задают максимальное количество элементов в третьем, четвертом, пятом и шестом измерениях соответственно. Например, команда

```
TraSr     MATRIX      ,5,7
```

определяет матрицу с именем TraSr, содержащую 5 строк и 7 столбцов.

Матрицы имеют единственный СЧА с названием MX, с помощью которого можно обращаться к любому элементу матрицы. В качестве значений строк и столбцов могут использоваться только имена, целые числа и параметры транзактов. Например:

```
MX6(13,5)  
MX$Tab2(P4,P$Nal)  
MX$Spis(Usor,P3)
```

В первом примере определяется элемент матрицы номер 6, находящийся на пересечении 13-й строки и 6-го столбца. Во втором примере для определения элемента матрицы с именем Tab2 используются значения параметра номер 4 и параметра с именем Nal транзакта, а в третьем — имя переменной пользо-

вателя и параметр номер три транзакта. Переменной пользователя предварительно должно быть присвоено целое положительное значение командой EQU.

Следует обратить внимание на следующее. В первом примере матрица имеет не имя, а номер, хотя в команде определения матрицы, как отмечалось выше, в качестве метки Name может быть использовано только имя. Для присвоения матрице номера вместо имени нужно до записи команды определения матрицы использовать команду EQU. Например:

```
Sk1      EQU      6
Sk1      MATRIX   , 10, 7
```

В данном примере имя Sk1 матрицы заменено на номер 6, по которому MX6 теперь и нужно обращаться к матрице. Если в команде MATRIX вместо имени Sk1 сразу записать число 6, то при трансляции возникнет останов по ошибке «Пропущена метка оператора».

Перед началом моделирования или выполнения команды CLEAR ON значения всех элементов матрицы устанавливаются равными нулю. Некоторым или всем элементам по желанию разработчика могут быть присвоены ненулевые значения или они могут быть переведены в неопределенное состояние командой INITIAL. Например:

```
INITIAL  MX5(2,6), -648.237
INITIAL  MX$Ntf(4,7), 48
INITIAL  MX$Avt(13,24), Koef
INITIAL  Gros, 48
INITIAL  Pult
INITIAL  MainResult, UNCEPIFIED
```

В первом примере элементу матрицы номер 5, находящемуся на пересечении второй строки и шестого столбца, присваивается начальное значение -648.237 , во втором — элементу (4,7) матрицы с именем Ntf — значение 48, в третьем — элементу (13,24) матрицы с именем Avt — значение переменной пользователя Koef. В четвертом примере вместо операнда указано имя матрицы Gros, поэтому всем элементам этой матрицы присваивается значение 48. В пятом примере также вместо операнда A стоит имя матрицы Pult, но операнд B не используется, поэтому всем элементам матрицы присваивается значение 1. В шестом примере вместо операнда B записано ключевое слово UNCEPIFIED, поэтому все элементы предварительно созданной матрицы MainResult будут переведены в неопределенное состояние. Этим переводом матрица подготовлена для использования в эксперименте, в котором могут быть пропущенные, т. е. не полученные в ходе эксперимента данные. Неопределенное состояние элемента матрицы указывает на то, что данных нет. Если же элементы матрицы были бы переведены в нуль, то это бы воспринималось как получение данных эксперимента, равных нулю, что неверно.

После эксперимента матрица с его результатами передается библиотечной процедуре дисперсионного анализа (ANOVA), которая элементы, находящиеся

в состоянии UNPECIFIED, обрабатывает как пропущенные данные вместо нулевых значений.

Оператор MATRIX создает матрицу в текущей модели. Матрица не может быть удалена из текущей модели. Если оператор MATRIX удаляется из выполняемой модели, связь с матрицей в текущей модели остается. Матрица может быть переопределена повторно другим оператором MATRIX с той же меткой. Перед использованием матрица должна быть определена в операторе MATRIX. Матрица ограничена максимальным объемом памяти, который устанавливается в журнале настроек модели.

Для записи в процессе моделирования значений в матрицы, а также для увеличения или уменьшения значений, записанных в матрицы, служит блок MSAVEVALUE. Формат записи его следующий:

MSAVEVALUE A, B, C, D

В операнде A задается имя или номер матрицы. Крайним правым символом операнда может быть знак + (режим накопления) или знак – (режим вычитания). В операнде B задается номер строки, в операнде C — номер столбца матрицы. Операнд D определяет значение, которое должно храниться, прибавляться или вычитаться. Все операнды являются обязательными. Они могут быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр. Например:

```
MSAVEVALUE      2, 13, P21, 782.34
MSAVEVALUE      X$Bba-, 8, 23, V$Hit
MSAVEVALUE      Rta+, 11, (User#2+3), M1
MSAVEVALUE      (Rra+Q$Rem)+, (Rra-5), 12, P$Nht
MSAVEVALUE      4, Stroka, Stolbez, "Dlina Mars"
```

В первом примере при входе транзакта в блок MSAVEVALUE значение элемента матрицы номер 2, расположенного на пересечении строки 13 и столбца, номер которого содержится в параметре 21 вошедшего транзакта, замещается числом 782.34. Во втором примере значение элемента на пересечении строки 8 и столбца 23 матрицы, номер которой содержится в сохраняемой ячейке с именем Bba, уменьшается на вычисленное значение арифметической переменной Hit.

В третьем примере номер столбца находится как результат вычисления выражения в скобках (User#2+3), а затем значение определенного таким образом элемента матрицы Rta увеличивается на величину относительного времени M1 пребывания транзакта в модели.

В четвертом примере номер матрицы и номер строки элемента определяются как результаты вычисления выражений в скобках (Rra+Q\$Rem) и (Rra-5) соответственно. После этого значение найденного элемента увеличивается на величину, содержащуюся в параметре с именем Nht активного транзакта.

В пятом примере элементу матрицы номер 4, находящемуся на пересечении строки и столбца, определяемых переменными пользователя Stroka и Stolbez,

присваивается строка "Dlina Mars". Если переменным пользователя предварительно командой EQU не будут присвоены соответствующие значения, то возникнет останов по ошибке.

Останов по ошибке возникает в том случае, если при входе транзакта в блок SAVEVALUE не будет найдена матрица с нужным именем или номером, или не будет найден элемент с соответствующими номерами строки и столбца.

СЧА МХ в блоке MSAVEVALUE при указании операнда А, как и СЧА Х в блоке SAVEVALUE при указании операнда А, не используется. Ссылка на этот СЧА применяется также в выражениях, функциях, операторе INITIAL.

Из рассмотренного выше формата команды описания матрицы следует, что матрица GPSS может иметь до шести измерений. Однако, как видно из формата блока MSAVEVALUE, он предоставляет возможность использовать только двумерные матрицы, т. е. в нем доступны только первые два измерения. Остальные индексы равны 1.

Если нужно использовать матрицы более чем двух измерений, для этого следует создать одну или более PLUS-процедур. PLUS-процедуры имеют доступ ко всем элементам всех матриц. Матрицы, определенные командой MATRIX, являются глобальными и доступны всем PLUS-процедурам. На время выполнения PLUS-процедуры могут быть созданы временные матрицы с локальной областью видимости.

1.5.10. Арифметические переменные и арифметические выражения

Арифметическая переменная является СЧА, который определяется пользователем.

Если процесс моделирования выполняется не в режиме совместимости с GPSS/PC, арифметическая переменная в GPSS World может быть определена командой VARIABLE или командой FVARIABLE, так как действие этих команд, а следовательно, и вычисляемые значения одинаковы. Форматы команд:

Name	VARIABLE	A
Name	FVARIABLE	A

Операнд А — задаваемое пользователем выражение.

Name — имя арифметической переменной.

Ссылкой на арифметическую переменную является либо V\$имя, либо Vj, где j — номер переменной. Символическое имя может быть заменено номером — положительным целым числом, для чего нужно использовать команду EQU.

Например:

SrSkop	EQU	2
SrSkop	VARIABLE	X\$12/MX\$Vrem(3,P1)

Теперь на арифметическую переменную SrSkop можно ссылаться так: V2. Команды EQU нужно размещать в начале модели. В команде определения пе-

ременной, так же как и в команде определения матрицы, вместо имени сразу указывать номер нельзя. На этапе трансляции произойдет останов по ошибке «Пропущена метка оператора».

Величиной арифметической переменной является величина заданного пользователем арифметического выражения. Значение арифметической переменной может использоваться следующим образом:

- как операнд; в этом случае значение арифметической переменной может представлять: номер объекта j ; j логического атрибута (блок GATE); номер параметра транзакта (блоки ASSIGN, INDEX, LOOP, MARK, SPLIT); значение атрибута;
- как аргумент функции;
- как значение зависимой переменной атрибутивной функции;
- как аргумент таблицы;
- как операнд другой арифметической переменной или булевой переменной.

Приведем примеры арифметических выражений.

Pas	VARIABLE	R\$Nak+S\$Nak
Mak	VARIABLE	Q\$Parik-Q\$Box
Pak	VARIABLE	P5+Q\$Bul/5
Vse	VARIABLE	N\$Ewm@10

В первом примере переменная Pas представляет собой величину, равную сумме оставшейся емкости и текущего содержимого памяти Nak. Данный пример показывает, что разработчик может задавать арифметическую переменную, имеющую в качестве своей величины емкость памяти. Во втором примере переменная Mak определяется как текущее содержимое очереди Parik минус текущее содержимое очереди Box. В третьем примере переменная Pak подсчитывается вначале делением текущего содержимого очереди Bul на 5, а затем прибавлением результата к значению параметра P5 активного транзакта. В четвертом примере Vse — счетчик числа входов в блок с именем Ewm по модулю 10.

Значение арифметического выражения вычисляется, когда активный транзакт входит в блок, оператор которого среди своих операндов содержит одну или более ссылок на арифметические переменные. Вычисленные значения имеют вещественный тип.

В GPSS World промежуточные результаты и значения СЧА не округляются. Округление может задать только сам пользователь. Например:

```
Kom1      VARIABLE      MX$Plan(P$Stroka,P$Stolbez)\User
```

Вычисленное значение переменной Kom1 будет целым числом, так как использована операция деления нацело. Для округления можно использовать и библиотечную функцию INT(A) выделения целой части. Например:

```
Kom1      VARIABLE      INT(MX$Plan(P$Stroka,P$Stolbez)/User)
```

152 9156

Для деления в данном случае использована обычная операция.

Если же процесс моделирования выполняется в режиме совместимости с GPSS/PC, то вычисление одного и того же арифметического выражения, заданного командами VARIABLE и FVARIABLE, производится по-разному. Это связано с тем, что в GPSS/PC значения всех СЧА целого типа.

В режиме совместимости при описании арифметической переменной командой VARIABLE все промежуточные вычисления и конечный результат округляются до целого значения. Действительные переменные (с плавающей точкой) определяются командой FVARIABLE. В этом случае округляется до целого значения только конечный результат, а промежуточные вычисления не округляются, т. е. действительные переменные также целочисленны.

Арифметическая переменная и переменная с плавающей точкой в режиме совместимости с GPSS/PC не могут иметь одинаковые имена. Если они имеют одинаковые имена, то при вычислении используется более позднее из двух описаний. Происходит это потому, что СЧА V используется для обращения как к арифметическим переменным, так и к переменным с плавающей точкой. Способ же вычисления переменной V является оператором описания этой переменной.

Различие результатов, полученных при вычислении переменных с плавающей и фиксированной точками, можно увидеть из приведенных ниже примеров:

TrAvt1	FVARIABLE	30#(7/3)
TrAvt2	VARIABLE	30#(7/3)

Значение переменной TrAvt1 будет равно 40, поскольку константа 30 будет умножена на 1,34 и от результата 40,2 будет взята целая часть. Переменная TrAvt2 равна 30, поскольку результат промежуточной операции 1,34 будет округлен до 1.

ЗАМЕЧАНИЕ

Так как GPSS World предоставляет возможность использовать выражения и в качестве операндов, то для упрощения моделей рекомендуется следующее. Если одно и то же выражение используется в различных частях модели, его нужно определить как переменную только один раз, а затем лишь ссылаться на него. Где размещать однократно используемые выражения решает пользователь: может быть в сложной модели для облегчения возможных правок лучше иметь все выражения как арифметические и булевы переменные в одном месте с достаточными для правильного понимания комментариями.

1.5.11. Булевы переменные

Булевы переменные с использованием всего лишь одного блока дают возможность принимать решения в зависимости от состояния и значения атрибутов многих объектов GPSS. Булева переменная представляет собой логическое выражение, составленное из различных СЧА, в том числе и других булевых переменных. Выражение может также содержать библиотечные функции и Plus-процедуры.

Булева переменная определяется командой BVARIABLE. Формат команды:

Name	BVARIABLE	A

Name — имя булевой переменной, которое так же, как и имя арифметической переменной, по желанию пользователя командой EQU может быть заменено номером. Операнд A — логическое выражение.

В выражении можно использовать арифметические и логические операторы (см. табл. 1.2). Конечный результат преобразуется в целое значение 0, если равен нулю, или в целое значение 1, если отличен от нуля.

Логические операторы связаны с объектами аппаратной категории и используются для определения состояния этих объектов. Имеются следующие логические операторы:

- FVj равен 1, если устройство j доступно, иначе — 0;
- FIj равен 1, если устройство j обслуживает прерывание, иначе — 0;
- SFj равен 1, если МКУ j заполнено полностью, иначе — 0;
- SEj равен 1, если МКУ j пусто, иначе — 0;
- SVj равен 1, если МКУ j доступно, иначе — 0;
- LSj равен 1, если логический ключ j включен, иначе — 0.

Под j понимается номер или имя. Например:

Ok _y	BVARIABLE	FV\$Rem1
Can3	BVARIABLE	SF\$Use1
Prov1	BVARIABLE	SV\$Pusk
Prov2	BVARIABLE	LS2

В первом примере булева переменная Ok_y равна 1, если устройство Rem1 доступно, и — 0, если не доступно. Во втором примере булева переменная Can3 равна 1, если МКУ Use1 заполнено полностью. В третьем примере булева переменная Prov1 равна 1, если МКУ Pusk доступно. В четвертом примере булева переменная Prov2 равна 1, если логический ключ номер 2 включен.

Операторы отношения (см. табл. 1.2) производят алгебраическое сравнение операндов. Например:

Prov1	BVARIABLE	V\$VseAvt'G'16
Prov2	BVARIABLE	Q5\$'LE'P3
Prov3	BVARIABLE	MX\$Plan(Stroka,Stolbez)'GE'P2

Булева переменная Prov1 равна 1, если переменная VseAvt больше 16, иначе равна 0. Во втором примере булева переменная Prov2 равна 1, если текущая длина очереди номер 5 меньше или равна значению параметра 3 транзакта. В третьем примере булева переменная Prov3 равна 1, если значение определенного элемента матрицы Plan больше или равно значению параметра 2.

Булевых операторов два: OR — ИЛИ и AND — И. Оператор ИЛИ проверяет выполнение хотя бы одного из условий, а оператор И требует выполнения обоих условий. Например:

Con1	BVARIABLE	FI\$Rem' OR' SF4
Con2	BVARIABLE	FI\$Rem' AND' SF\$Pogr
Con3	BVARIABLE	(V3' G' 7)' AND' (FN\$Rav' OR' LS7)

Булева переменная Con1 равна 1, если выполняется одно из условий: устройство Rem обслуживает прерывание или память номер 4 не заполнена. Булева переменная Con2 равна 1, если выполняются оба условия: устройство Rem обслуживает прерывание и память с именем Pogr не заполнена. В третьем примере переменная Con3 равна 1, если выполняются оба условия: значение переменной номер 3 больше 7 и логический ключ номер 7 включен.

Скобки в третьем примере нужны только для задания определенных булевых соотношений. Скобки следует использовать только в тех случаях, когда они необходимы.

Если булева переменная задается СЧА, как например:

Stan	BVARIABLE	V\$TreAvt

то вычисляется значение арифметической переменной с именем TreAvt, и если оно отлично от нуля — значение булевой переменной Stan будет равно 1, в противном случае — 0.

Рассмотрены все элементы выражений, кроме функций. Методы применения функций будут рассмотрены в главе 6.

2. Построение моделей с одноканальными устройствами, функционирующими в режиме занятия и освобождения устройства

Для представления собственно обслуживания используются определенные элементы. Такими элементами могут быть либо люди, либо какие-то предметы. Независимо от этого подобные элементы в GPSS называют объектами аппаратной категории, к которой относят одноканальные (ОКУ) и многоканальные устройства (МКУ) и логические ключи (переключатели).

ОКУ характеризуются двумя основными свойствами:

- каждое ОКУ в любой момент времени может обслужить только один транзакт; если в процессе обслуживания появляется новый транзакт, он должен либо подождать своей очереди обслуживания, либо пойти куда-нибудь в другое место, либо, если он имеет более высокий приоритет, прервать протекающее обслуживание до его завершения;
- когда на ОКУ поступает транзакт, в модели надо пропустить время, необходимое для его обслуживания.

Поскольку ОКУ в модели может быть много, для различия им дают символические имена. Имена образуются так же, как и имена блоков.

При моделировании возможны следующие режимы организации функционирования ОКУ:

- занятие ОКУ и его освобождение;
- прерывание обслуживания ОКУ;
- недоступность ОКУ и восстановление доступности.

В данной главе мы рассмотрим режим занятия ОКУ и его освобождения. Остальные режимы работы ОКУ будут изложены в главе 4.

Прежде чем сразу рассматривать блоки, моделирующие ОКУ, вспомним, что потоки, существующие в реальных системах, в моделях имитируют транзакты. Поэтому сначала узнаем, как вводятся транзакты в модель и как выводятся из нее. А так как построение самых простейших моделей невозможно без некоторых блоков GPSS, такие блоки будут также рассмотрены в данной главе.

2.1. Организация поступления транзактов в модель и удаления транзактов из нее

2.1.1. Поступление транзактов в модель

GENERATE — это блок, через который транзакты входят в модель. Блок GENERATE имеет следующий формат записи:

GENERATE [A], [B], [C], [D], [E]

Скобки [] означают, что данный операнд является необязательным. Не существует ограничений на число различных блоков GENERATE в одной модели.

Интервалы времени между последовательными появлениями транзактов блока GENERATE называют интервалом поступления. Все разработчики должны задавать спецификацию распределений интервалов времени поступления в блоке GENERATE. Информация, необходимая для этого, задается операндами A и B. Все возможные виды распределения интервалов времени поступления в GPSS делят на *равномерно распределенные* и *все другие виды распределения*.

Операнд A — средний интервал времени между последовательными поступлениями транзактов в модель.

Операнд B — задает модификатор, который изменяет значения интервала генерации транзактов по сравнению с интервалом, указанным операндом A. Есть два типа модификаторов: модификатор-интервал и модификатор-функция.

С помощью *модификатора-интервала* задается *равномерный закон* распределения времени между генерацией транзактов.

Операнды A и B могут быть именем, положительным числом, выражением в скобках или непосредственно СЧА. Перечень блоков GPSS, их операндов и возможных значений операндов приведены в приложении 3.

При вычислении разности значений $(A-B)$, заданных операндами A и B, получается нижняя граница интервала, а при вычислении суммы $(A+B)$ — верхняя граница. После генерации очередного транзакта выбирается число из полученного интервала, это и будет значение времени, через которое следующий транзакт выйдет из блока GENERATE.

Когда операнды A и B задают в виде констант (B — модификатор-интервал), они должны быть неотрицательными числами, т. е. интервал времени может быть выражен числами, например, 4.1,...,12.7. Предположим, что транзакт входит в модель — блок GENERATE — в момент модельного времени 25.6. После того, как этот транзакт попадет в следующий блок модели, планировщик GPSS разыграет случайное значение из распределения интервалов времени, равного 8.4 ± 4.3 .

Пусть разыгранным значением будет число 9.7. Тогда планировщик планирует приход следующего транзакта в блок GENERATE в момент времени $25.6 + 9.7 = 35.3$.

Можно выбрать для розыгрыша генератор равномерно распределенных случайных чисел. Это устанавливается на странице Random Numbers (Случайные числа) в журнале настроек модели. Нужно выбрать Edit ► Settings (Правка ► Настройка) и страницу Random Numbers (Случайные числа), на которой в поле ввода GENERATE ввести номер генератора — любое положительное число. По умолчанию используется генератор номер 1.

Операнды A и B не обязательно должны быть заданы. Когда один или оба операнда не указаны, по умолчанию предполагается их нулевое значение. Например, $A=16.4$, $B=0$. Поскольку $B=0$, то интервалы времени распределены равномерно, 16.4 ± 0 , т. е. интервал времени поступления равен 16.4. Это пример того, как может быть задано детерминированное значение интервалов времени.

Более сложные интервалы времени поступления транзактов (не по равномерному закону) могут быть заданы с использованием *модификатора-функции*. Под действием модификатора-функции значение операнда A умножается на значение функции, заданной операндом B.

При любом способе вычисления интервала времени значение операнда B не должно превышать значения операнда A, в противном случае в блоке GENERATE может быть получен отрицательный интервал времени, который вызовет останов по ошибке «Отрицательное время задержки».

Рассмотрим три дополнительных операнда: C — смещение интервалов, D — ограничитель, E — уровень приоритета.

Смещение интервалов (первоначальная задержка) C — это момент времени, в который в блоке GENERATE должен появиться первый транзакт. После этого первого прихода все остальные приходы транзактов возникают в соответствии с распределением интервалов времени, задаваемых операндами A и B. Операнд C можно использовать как для ускорения, так и для замедления прихода первого транзакта или для указания прихода в нужный момент времени. Начальная задержка может быть меньше, равна или больше среднего времени, заданного операндом A. Когда операнд C не используется, интервалы генерирования определяются операндами A и B (они не оказывают влияния на задержку). Операнд C может быть таким же, как и операнды A и B.

Операнд D задает граничное значение общего числа транзактов, которые могут войти в модель через данный блок GENERATE в течение времени моделирования. Когда это число достигнуто, данный блок GENERATE перестает быть активным. Если не определено граничное значение (операнд D не используется), блок GENERATE остается активным в течение всего времени моделирования, т. е. по умолчанию ограничения на количество создаваемых транзактов нет.

Операнд E устанавливает класс приоритета каждого из транзактов, входящих в модель через данный блок GENERATE. Для задания приоритетов с целью повышения эффективности работы GPSS World рекомендуется использовать последовательность целых чисел 0, 1, 2, 3, ... вместо, например, 37, 43, 88, 122, ...

Чем выше число, тем выше приоритет. Если операнд E не используется, по умолчанию приоритет генерируемых данным блоком GENERATE транзактов равен нулю.

Операнды D и E могут задаваться так же, как и операнды A, B и C, но при этом принимать значения только целых положительных и целых чисел соответственно.

В любом блоке GENERATE должен быть обязательно задан либо операнд A, либо операнд D. *Нельзя использовать в качестве операнда параметры транзактов.* Необходимо также помнить, что транзакт не должен входить в блок GENERATE. Если транзакт пытается это делать, возникает ошибка выполнения.

Приведем примеры записи блоков GENERATE:

а) с операндом A

```
GENERATE      38.6
GENERATE      X$IntPostTran
GENERATE      MX$VrPost(3,6)
GENERATE      V$Prom
GENERATE      (Exponential(11,0,X$Mat))
GENERATE      IntPostTran
```

б) с операндами A и B

```
GENERATE      73.25,X$0tk
GENERATE      X$Sredne,FN2
GENERATE      Sredne,FN4
GENERATE      (V$Post+7.1),FN$Mod
```

в) с операндами A и C

```
GENERATE      7.3,,4.1
GENERATE      7.3,,X$VrSm
GENERATE      V$IntPost,,MX2(X$Stroka,X$Stolbez)
GENERATE      (Normal(3,X$Sre,X$Sre0tk)),,Sme
```

г) с операндами A, B, E

```
GENERATE      13.3,2.8,,,1
GENERATE      (Normal(8,X$Sred,X$SreKv0tk)),Post,,1
GENERATE      V$IntPostTran,(V1-12.3),,12
```

Приведенные примеры демонстрируют различные способы задания операндов блока GENERATE. Однако при этом нужно помнить следующее.

В начальный момент времени в каждом блоке GENERATE производится подготовка к выходу одного транзакта. На этой стадии модель еще полностью не инициализирована для выполнения. По этой причине все СЧА, описанные в блоке GENERATE, должны быть уже определены. В модели блоку GENERATE должны предшествовать команды определения EQU, INITIAL, FUNCTION, VARIABLE, FVARIABLE. Это делается для того, чтобы СЧА в блоке GENERATE, который ссылается на них, давали нужные результаты. Например:

SrIntPost	EQU	47.2
Stan0tk1	EQU	28.6
	INITIAL	X\$KolTrans, 43
	GENERATE	SrIntPost, St0tk, , X\$KolTrans

Как видно из примера, блоку GENERATE предшествуют присвоения командой EQU именам числовых значений и командой INITIAL начального значения сохраняемой ячейке с именем KolTrans.

2.1.2. Удаление транзактов из модели и завершение моделирования

Транзакты удаляются из модели, попадая в блок TERMINATE (завершить).

Блоки TERMINATE всегда позволяют войти всем транзактам, которые пытаются это сделать. В модели может быть любое число блоков TERMINATE. Блок имеет следующий формат записи:

TERMINATE [A]

Значением операнда A является число единиц, на которое блок TERMINATE уменьшает содержимое *счетчика завершения*, определяющего момент окончания моделирования. Операнд A может быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*<параметр>. По умолчанию значение операнда A равно нулю. В этом случае транзакт уничтожается, а значение счетчика завершения не меняется.

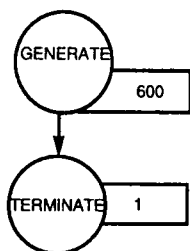
Счетчик завершения представляет собой ячейку памяти, которая хранит положительное целое число, записанное в начале процесса моделирования командой START.

В процессе моделирования транзакты попадают в блок TERMINATE и в соответствии со значением операнда A вычитают определенное число из счетчика завершения. При достижении содержимым счетчика нуля моделирование завершается. *В модели может быть много блоков TERMINATE, но счетчик завершения только один.*

Когда пользователь подготавливает модель, он задает время моделирования, указывая в команде START значение счетчика завершения. Поскольку пути транзактов в модели имеют различный смысл, каждый блок TERMINATE может либо уменьшать, либо не уменьшать содержимое счетчика завершения.

Рассмотрим пример, в котором блок TERMINATE и команда START используются для управления временем моделирования. Предположим, что разработчик выбрал в качестве единицы времени 1 мин. Он хочет промоделировать поведение системы в течение 10 часов, затем моделирование должно быть закончено. За единицу модельного времени возьмем 1 мин, тогда время моделирования равно $10 \cdot 60 = 600$ единицам.

Любая модель на GPSS состоит из одного или нескольких сегментов. Для управления временем моделирования разработчик (см. блок-диаграмму):



- 1) включает в модель сегмент из блоков GENERATE и TERMINATE;
- 2) во всех прочих блоках TERMINATE модели использует операнд A по умолчанию;
- 3) в команде START в качестве операнда A использует 1.

В процессе моделирования транзакты, которые двигаются в других сегментах модели, время от времени выводятся из модели в других блоках TERMINATE, но они не оказывают воздействия на счетчик завершения. В момент модельного времени 600 транзакт в приведенном выше сегменте попадает в блок GENERATE и сразу же переходит в следующий блок TERMINATE. Поскольку операнд этого блока содержит 1, то из счетчика завершения вычитается 1 и значение счетчика равно нулю. В результате планировщик прекращает моделирование.

Команда START используется для запуска процесса моделирования. Она имеет следующий формат записи:

START A, [B], C, [D]

Операнд A задает значение счетчика завершения, определяющего момент окончания прогона модели. Может быть только целым положительным числом. Операнд B — операнд вывода статистики. Он может быть NP («нет вывода данных») либо опущен. При задании NP стандартный отчет не выводится. По умолчанию выводится стандартный отчет. Операнд C не используется и сохранен для совместимости с описаниями ранних версий GPSS. Он использовался как счетчик «кадров» для периодического вывода отчета. Эта функция доступна при использовании в модели более одной команды START и REPORT. Операнд D определяет необходимость вывода содержимого списков событий. Если операнд D указать любым положительным целым числом, например, 1, списки текущих и будущих событий включаются в стандартный отчет и выводятся. Если операнд D опущен, по умолчанию списки в стандартном отчете не выводятся.

Команду START можно сразу указывать в конце программы модели при ее подготовке и в таком виде записывать на магнитный носитель. Тогда после трансляции модели, т. е. создания объекта «процесс моделирования», начинается моделирование. Эту же команду можно вводить в программу модели в интерактивном режиме.

Однако может возникнуть необходимость завершить моделирование не по истечении какого-то времени, а после обработки определенного количества транзактов, имитирующих, например, изготовленные детали, переданные по каналу связи сообщения и т. д. В этом случае сегмент задания времени моделирования не нужен. Для организации такого способа завершения моделирования необходимо сделать следующее. В блоках TERMINATE, которые выводят из модели транзакты, имеющие смысл тех же изготовленных деталей или переданных сообщений, указать число, на которое уменьшается счетчик завершения моделирования. В команде START также указать число, деление которого на указанное в блоке TERMINATE число даст требуемое количество из-

готовленных деталей или переданных сообщений. Например, требуется завершить моделирование после изготовления 100 деталей. В модели это может быть так:

```

      . . .
TERMINATE      1
      . . .
TERMINATE      1
      . . .
TERMINATE
START          100

```

Блоков TERMINATE, которые выводят из модели транзакты, соответствующие изготовленным деталям, может быть несколько. Все эти блоки должны иметь 1 в качестве операнда A. У остальных блоков TERMINATE, если они есть в модели, операнд A должен быть опущен.

Итак, транзакты введены в модель. Но аналоги транзактов — элементы потоков в реальных системах имеют различные характеристики. Рассмотрим, как эти же характеристики присваиваются транзактам.

2.2. Изменение значений параметров транзактов

2.2.1. Блок ASSIGN

Каждый транзакт может иметь любое число параметров. Интерпретация смысла параметров произвольная. В момент генерации транзактов все его параметры нулевые (только те, которые используются в модели). Блок ASSIGN является основным средством для задания значений параметров транзактов. Формат записи:

```

ASSIGN      A, B, [C]

```

Операндом A задается номер параметра, которому присваивается значение. Операнд A может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА*параметр и следующими за ними знаками + (если нужно увеличить), — (если нужно уменьшить).

Операнд B определяет значение, которое следует добавить, вычесть или которым следует заменить значение в параметре, заданном операндом A. Если такой параметр не существует, он создается со значением, равным 0. Операнд B может быть таким же, как и операнд A, кроме того, числом и строкой.

Операнд C задает номер модификатора-функции. При использовании операнда C значение операнда B умножается на значение модификатора-функции. Полученное произведение становится значением, которое изменяет значение параметра, заданного операндом A. Следует отметить, что операнд C определяет номер функции или ее имя (не нужно использовать СЧА FN или FN\$ перед ним). Если используется СЧА FN, например FN3, вычисляется функция номер 3 GPSS. Полученный результат используется для определения вто-

рой функции GPSS, которая также вычисляется и ее значение умножается на значение операнда B. Приведем примеры записи блока ASSIGN:

```

ASSIGN      1, 754.3
ASSIGN      4+, Q5
ASSIGN      3-, 5.85, 7
ASSIGN      Name, "Plan"
ASSIGN      Tr1-, (Normal(32, Sredn, SrKv0tk1), Expdis

```

В первом примере параметру 1 присваивается 754.3. Во втором примере к значению параметра 4 прибавляется значение длины текущей очереди номер 5. В третьем примере из значения параметра 3 вычитается произведение 5.85 на вычисленное значение функции номер 7. В четвертом примере параметру с именем Name присваивается строка Plan. В пятом примере вычисляются выражение в скобках и функция с именем Expdis, перемножаются и полученное произведение вычитается из значения параметра с именем Tr1.

2.2.2. Блок PLUS

В GPSS World значения параметров транзактов можно также изменять блоком PLUS. Блок PLUS вычисляет выражение и может записать его в параметр. Блок PLUS имеет следующий формат:

```
PLUS      A, [B]
```

Операнд A — выражение. Может быть именем, числом, строкой, выражением в скобках, СЧА или СЧА*параметр.

Операнд B — номер параметра транзакта, в котором сохраняется результат. Операнд может быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр.

Например:

```
PLUS      (Exponential(23, 0, 32.7)+48.75), Itog
```

При входе транзакта в блок PLUS вычисляется выражение в скобках, которым задан операнд A. Затем полученный результат присваивается параметру с именем Itog. Если такой параметр не существует, он создается.

2.2.3. Блок INDEX

Изменить значение параметра транзакта можно еще блоком INDEX следующего формата:

```
INDEX      A, B
```

Операнд A — номер параметра транзакта. Может быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр.

Операнд B — числовое значение, которое прибавляется к значению параметра, указанного операндом A. Может быть именем, числом, выражением в скобках, СЧА или СЧА*параметр.

Полученный результат заносится в первый параметр транзакта. Например:

INDEX Slag1, (Normal(21, Mat, Sre0tk)+X\$post)

При входе транзакта вычисляется выражение в скобках, которым указан операнд В, и складывается со значением параметра с именем Slag1. Результат сложения присваивается параметру номер один транзакта. Если этот параметр не существует, он создается. Если же нет параметра, указанного операндом А, происходит останов по ошибке «Обращение к несуществующему параметру».

2.2.4. Блок MARK

В GPSS имеется также блок MARK, который заносит в активный транзакт или его параметр значение абсолютного модельного времени. Блок MARK имеет следующий формат:

MARK [A]

Операнд А — номер параметра транзакта, в который записывается значение абсолютного модельного времени. Может быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр. Например:

MARK
MARK Vxod

В первом примере операнд А не используется. По умолчанию вошедшему в этот блок транзакту устанавливается время входа в систему, равное абсолютному модельному времени.

Во втором примере операнд А используется. Поэтому значение абсолютного модельного времени (СЧА AC1) заносится в параметр с именем Vxod вошедшего транзакта. Если этого параметра нет, он создается.

Блок MARK можно использовать, если необходимо, например, определить время обработки транзакта, которое складывается из времен обработки на нескольких этапах. С помощью блока MARK нужно записать абсолютное модельное время начала и конца обработки, а затем найти разность, которая и будет искомым временем.

ЗАМЕЧАНИЕ

В блоках ASSIGN, PLUS, INDEX, MARK и LOOP при записи операндов, которые являются параметром транзакта, СЧА Р не используется.

2.3. Занятие одноканального устройства и его освобождение

В GPSS элементами, которые требуют обслуживания, являются транзакты. Они перемещаются в модели от блока к блоку. Если в какой-то момент активности транзакт занимает ОКУ, то для этого он входит (или пытается вой-

ти) в соответствующий блок, описывающий это ОКУ. Блок должен обладать следующими свойствами:

- если ОКУ уже используют, транзакт не может войти в блок и должен ждать в очереди;
- если ОКУ не используют, транзакт может войти в блок. Статус ОКУ изменяется на «занято».

Блок, который обладает этими свойствами, является блоком SEIZE (занять). Вход транзакта в блок SEIZE моделирует занятие ОКУ.

После обслуживания (блок, имитирующий обслуживание, будет рассмотрен в п. 2.4) вход того же транзакта в другой блок моделирует освобождение ОКУ. Назначением этого блока является изменение состояния ранее занятого ОКУ с «занято» в «незанято». Этим блоком является блок RELEASE (освободить).
Форматы блоков:

SEIZE	A
RELEASE	A

В обоих блоках операнд A — это имя занимаемого (освобождаемого) ОКУ. Может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА*параметр. Планировщик автоматически обеспечивает возникновение транзактов и ОКУ, когда этого требует логика модели. В то время как транзакты находятся в модели временно, ОКУ, используемые в модели, существуют постоянно в течение всего процесса моделирования. Прежде чем освободить ОКУ, транзакт может пройти через неограниченное число блоков. Например:

SEIZE	Can
RELEASE	Can

Если ОКУ с именем Can не занято, активный транзакт занимает его. Если ОКУ занято, транзакт помещается в *список задержки* данного ОКУ позади транзактов с таким же приоритетом и не входит в блок SEIZE. Транзакту также отказывается во входе в блок SEIZE, если ОКУ с именем Can находится в недоступном состоянии (режим недоступности ОКУ рассматривается в п. 4.2).

ОКУ, как уже отмечалось, может иметь имя или номер. В данном случае разрешается записывать вместо операнда A номер непосредственно без предварительного присвоения его имени командой EQU. Например:

SEIZE	5
RELEASE	5

Блоки SEIZE и RELEASE при необходимости могут переопределяться. Для этого они должны иметь метки.

2.4. Имитация обслуживания посредством задержки во времени

Обычно транзакт занимает ОКУ для того, чтобы немедленно начать на нем обслуживание, которое длится некоторый промежуток модельного времени. В течение этого времени транзакт должен прекратить двигаться по модели. Только по истечении времени обслуживания он должен попасть в блок RELEASE для освобождения ОКУ.

Для задержки транзакта в течение некоторого интервала модельного времени используется блок ADVANCE. Чаще всего этот интервал задается случайной переменной. Как и при использовании блока GENERATE, информация, необходимая для описания соответствующего времени обслуживания и его распределения, задается операндами A и B. Формат записи блока ADVANCE следующий:

ADVANCE A, [B]

Здесь A — среднее время обслуживания, а B — способ модификации операнда A. Каждый из операндов A и B может быть именем, числом, выражением в скобках, СЧА или СЧА*параметр.

Как и в блоке GENERATE, модификаторы могут быть двух типов: модификатор-интервал и модификатор-функция, т. е. блок ADVANCE вычисляет время задержки (приращения модельного времени) такими же способами.

Если задан только операнд A, он вычисляется и используется в качестве времени задержки. Например:

ADVANCE (Normal(12, X\$\$srednee, X\$\$sre0tk))

Время задержки распределено по нормальному закону со средним значением и среднеквадратическим отклонением, предварительно записанными командой INITIAL в сохраняемые ячейки с именами Srednee и Sre0tk соответственно. Для генератора нормального распределения источником случайных чисел, равномерно распределенных в интервале [0, 1], является генератор номер 12 (RN12).

При задании операндов A и B, и B не определяет функцию, оба операнда вычисляются (если они не константы) и в качестве времени задержки выбирается случайное число, равномерно распределенное в интервале (A-B, A+B). Для розыгрыша может быть выбран любой генератор равномерно распределенных случайных чисел. Делается это так же, как и при выборе генератора для блока GENERATE. Только номер генератора на странице Random Numbers (Случайные числа) в журнале настроек модели нужно указать в поле ввода ADVANCE. По умолчанию используется генератор случайных чисел номер 1. Например:

ADVANCE 56.7, 23.2

В данном примере входящий транзакт задерживается на время, равномерно распределенное в интервале от 33.5 до 79.9.

Так же, как и в блоке GENERATE, при любых способах вычисления времени задержки значение операнда В не должно превышать значение операнда А. Если в приведенном выше примере операнд В взять равным 56.8, в процессе моделирования произойдет останов по ошибке «Отрицательное приращение времени».

Для задания времени задержки по другому закону, отличному от равномерного, в операнде В записывается модификатор-функция. При обращении к функции определяется некоторое число — значение функции. Это значение умножается на значение операнда А. Результат используется как время задержки. Например:

```
ADVANCE      Ring, FN$Exper
```

В примере вычисляется значение функции с именем Exper и умножается на значение переменной пользователя Ring, которой предварительно должно быть присвоено числовое значение командой EQU.

Как отмечалось ранее, в блоке GENERATE можно использовать функции и арифметические переменные, предварительно определенные командами FUNCTION и VARIABLE или FVARIABLE. Но в этих командах не должно быть ссылок на параметры транзактов, так как транзактов еще нет. В операндах блока ADVANCE ссылки на параметры транзактов возможны. Естественно, что этим параметрам пользователем должны быть предварительно присвоены соответствующие значения. Например:

```
ADVANCE      P1
ADVANCE      (Exponential(7, 0, MX$$Sred(P2, P$Stolb)))
ADVANCE      SrIntPost, FN*2
```

В первом примере транзакт задерживается на время, равное значению параметра 1. Во втором примере время задержки определяется по экспоненциальному закону. При этом среднее значение выбирается из элемента матрицы, номера строки и столбца которого содержатся в параметре 2 и параметре с именем Stolb соответственно. Для генератора экспоненциального распределения источником равномерно распределенных случайных чисел в интервале [0, 1] является генератор номер 7 (RN7).

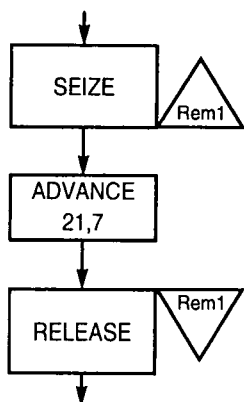
В третьем примере время задержки находится как произведение значения переменной пользователя SrIntPost и вычисленного значения функции, номер которой содержится в параметре 2 активного транзакта.

Блок ADVANCE никогда не препятствует входу транзакта. Любое число транзактов может находиться в этом блоке одновременно.

ЗАМЕЧАНИЕ

Только блоки GENERATE и ADVANCE позволяют поместить транзакты в список будущих событий. С помощью этих блоков моделируется продолжительность какого-либо события или промежутков времени между наступлением каких-либо событий.

Приведем пример использования блоков SEIZE, ADVANCE и RELEASE.



На схеме представлена последовательность этих блоков. Транзакт, войдя в блок SEIZE, займет ОКУ с символическим именем Rem1, задержится там благодаря блоку ADVANCE на 21 ± 7 единиц времени и затем покинет его. После того, как транзакт войдет в блок RELEASE, планировщик попытается продвинуть транзакт в следующий блок модели, и следующий транзакт попытается использовать ОКУ, называемое Rem1.

Блок ADVANCE можно располагать в любых местах модели, а не только между блоками SEIZE и RELEASE.

Время задержки может быть также равным нулю. Если время равно нулю, то транзакт в блоке ADVANCE не задерживается и переходит в следующий блок.

2.5. Проверка состояния одноканального устройства

Представим себе, что в какой-либо системе, модель которой вы собираетесь разработать, имеются, например, два канала передачи данных — два ОКУ. Естественно, что если один канал занят, то выясняется: свободен ли второй канал? Если он не занят, надо попытаться передать по этому каналу. Но как проверить состояние каналов в модели?

2.5.1. Проверка состояния одноканального устройства блоком GATE

В GPSS World блок GATE используется для определения состояния устройств без изменения их состояния. Формат записи:

GATE X A,[B]

Блок GATE работает в двух режимах:

- отказа во входе;
- разрешении во входе и альтернативном выходе.

При работе в режиме отказа блок GATE не пропускает транзакты, если соответствующий объект не находится в требуемом состоянии. Если же поставленное в блоке GATE условие удовлетворяется, то активный транзакт входит в него и затем переходит к следующему по порядку блоку. Операнд A определяет имя или номер ОКУ.

Условие задается одним из следующих условных операторов X, связанных с ОКУ:

- NU — ОКУ, заданное операндом A, свободно;
- U — ОКУ, заданное операндом A, занято.

Как видно, тип проверяемого объекта неявно задается условным оператором (позже мы рассмотрим применение блока GATE для проверки и других объектов аппаратной категории).

Операнд В содержит номер следующего блока для входящего транзакта, когда условный оператор имеет значение «ложь». Операнды А и В могут быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА*параметр. Если операнд В не используется, то проверка проводится в режиме отказа. Если результат этой проверки не будет «истина», то транзакт помещается в *список повторных попыток* проверяемого объекта. Когда состояние любого из объектов меняется, заблокированный транзакт снова активизируется, повторяется проверка заданного блоком GATE условия. Если это условие выполняется, транзакту разрешается войти в блок GATE и далее перейти к следующему по порядку блоку.

Например:

GATE	U	Zrk
GATE	NU	(FN\$Expert+4)
GATE	NU	Bat, Oper

В первом примере блок GATE не пропустит транзакт при условии незанятости ОКУ с именем Zrk. Во втором случае — когда занято ОКУ, номер которого определяется как результат вычисления и последующего округления выражения в скобках FN\$Expert+4. В третьем примере в случае занятости ОКУ с именем Bat транзакт будет направлен к блоку с именем Oper.

ЗАМЕЧАНИЕ

Несмотря на удобство применения блока GATE, в некоторых случаях это может привести при отсутствии операнда В к увеличению машинного времени за счет проведения проверок для заблокированных транзактов. Уменьшение числа проверок и исключение из проверяемых тех транзактов, для которых вряд ли когда-нибудь результат проверки будет истинным, может быть организовано с помощью блоков LINK и UNLINK. Методы применения этих блоков будут рассмотрены в п. 4.3.

Для проверки состояния различных объектов, в том числе и ОКУ, можно использовать также булевы переменные (см. п. 1.5.11) и блок TEST.

2.5.2. Проверка состояния булевой переменной и блоком TEST

Блок TEST описывает условие, которое проверяет при входе в него транзакт, и определяет номер следующего блока для вошедшего транзакта в зависимости от того, выполняется требуемое условие или нет. Блок TEST, как и блок GATE, также функционирует в двух режимах:

- отказа во входе;
- разрешении во входе и альтернативном выходе.

Формат записи:

TEST X A, B, [C]

Операнды А и В являются сравниваемыми величинами. Они могут быть именем, числом, строкой, выражением в скобках, СЧА, СЧА*параметр. Условный оператор X может быть выражен одним из следующих шести условных

операторов (см. табл. 1.2): L — меньше; LE — меньше или равно; E — равно; NE — не равно; G — больше; GE — больше или равно.

Если операнд C указан, транзакт всегда может войти в блок TEST и в зависимости от соотношения операндов будет передан либо в следующий блок, либо в блок, указанный операндом C. Если операнд C не указан, транзакт при невыполнении условия не сможет войти в блок TEST (режим отказа). Он помещается в *список повторных попыток* всех объектов, участвующих в проверке условия. Когда состояние любого из этих объектов изменяется, транзакт из списка повторных попыток активизируется и проверка проводится вновь. Если условие выполняется, транзакту разрешается войти в блок TEST.

Видно, что при функционировании блока TEST в режиме отказа, как и в таком же режиме функционирования блока GATE, возможно увеличение машинного времени на многократные проверки блокирующего условия. Сократить машинное время можно также с помощью блоков LINK и UNLINK, которые будут рассмотрены в п. 4.3.

В булевой переменной может использоваться СЧА F, равный 1, если ОКУ занято, в противном случае — 0.

А также следующие СЧА ОКУ:

- FC — количество транзактов, занимавших ОКУ с помощью блоков SEIZE и PREEMPT;
- FR — коэффициент использования ОКУ;
- FT — среднее время обслуживания ОКУ одного транзакта.

Приведем пример использования булевой переменной и блока TEST для проверки состояния ОКУ.

Prov	BVARIABLE	F\$Server
	TEST E	BV\$Prov, 0, PotZap
	SEIZE	Server
	ADVANCE	P\$Obrab
	RELEASE	Server

В данном примере при входе транзакта в блок TEST вычисляется булева переменная Prov. В булевой переменной использован СЧА ОКУ F (использование условных операторов NU и U недопустимо). Если ОКУ с именем Server занято, значение переменной Prov равно 1. Тогда условие, заданное блоком TEST, не выполняется, так как $1 \neq 0$, и транзакт не может занять ОКУ, а переходит к блоку с меткой PotZap, указанной операндом C. Если же ОКУ с именем Server свободно, то булева переменная Prov равна 0. Тогда проверяемое блоком TEST условие выполняется, транзакт переходит к следующему блоку, т. е. занимает ОКУ с именем Server.

Пример 2.1. Данный пример демонстрирует расширение возможностей по сравнению с блоком GATE проверки состояния ОКУ с помощью булевой переменной и блока TEST. В примере показывается распределение обслуживания

транзактов по трем ОКУ в зависимости от их коэффициентов использования (КИ): для ОКУ Сру1 он не должен превышать 0,5, а для ОКУ Сру2 — 0,6. Вся остальная нагрузка приходится на ОКУ Сру3. Причем, если коэффициент использования Сру3 составит 0,9, то транзакты не обслуживаются и теряются.

; Пример 2.1

; Задание булевых переменных

```
Prov1      BVARIABLE      F$Cpu1'AND'F$Cpu2'AND'F$Cpu3
Prov2      BVARIABLE      F$Cpu1'OR'(FR$Cpu1>500)
Prov3      BVARIABLE      F$Cpu2'OR'(FR$Cpu2>600)
Prov4      BVARIABLE      FR$Cpu3>900
```

; Сегмент имитации поступления и обработки заявок

```
GENERATE      (Exponential(310,0,6.3))
TEST E        BV$Prov1,0,Met1    ; Заняты ли все три ОКУ?
TEST E        BV$Prov2,0,Met2    ; Занято ли ОКУ1 или КИ1>0,5?
```

; Имитация работы ОКУ1

```
SEIZE         Cpu1              ; Нет, занять ОКУ1
ADVANCE       38,7.3            ; Имитация обслуживания
RELEASE       Cpu1              ; Освободить ОКУ1
TERMINATE     ; Обслуженные заявки
```

; Имитация работы ОКУ2

```
Met2          TEST E          BV$Prov3,0,Met3    ; Занято ли ОКУ2 или КИ2>0,6?
              SEIZE          Cpu2              ; Нет, занять ОКУ2
              ADVANCE       33.8,9.35          ; Имитация обслуживания
              RELEASE       Cpu2              ; Освободить ОКУ2
              TERMINATE     ; Обслуженные заявки
```

; Имитация работы ОКУ3

```
Met3          TEST E          BV$Prov4,0,Met1    ; КИ>0,9?
              SEIZE          Cpu3              ; Нет, тогда занять ОКУ3
              ADVANCE       (Exponential(310,0,12.3)) ;Имитация обслуживания
              RELEASE       Cpu3              ; Освободить ОКУ3
              TERMINATE     ; Обслуженные заявки
Met1          TERMINATE     ; Потерянные заявки
```

; Сегмент задания времени моделирования

```
GENERATE      3600
TERMINATE     1
```

Булева переменная Prov1 принимает значение 1 в случае, если ОКУ Сру1 и Сру2 заняты и коэффициент использования ОКУ Сру3 равен 0,9. Коэффициент использования в булевой переменной записывается в тысячных долях как целое число: 900 соответствует коэффициенту использования 0,9. В этом случае условие в первом блоке TEST не выполняется и транзакт направляется к блоку TERMINATE с меткой Met1 для уничтожения.

Во втором блоке TEST сравнивается значение булевой переменной Prov2 с нулем. Заданное условие не выполняется, если булева переменная принимает значе-

ние 1, т. е. ОКУ Cpu1 занято или его коэффициент использования равен 0,5. Тогда транзакт направляется на ОКУ Cpu2 к блоку TEST с меткой Met2. В этом блоке производится такая же проверка, как и в предыдущем случае, с той лишь разницей, что используется булева переменная Prov3. В случае невыполнения условия (значение булевой переменной равно 1), транзакт направляется на ОКУ Cpu3 к блоку TEST с меткой Met3.

Если коэффициент использования третьего ОКУ меньше 0,9 (значение булевой переменной Prov4 равно 0), условие в блоке TEST выполняется и транзакт направляется на обслуживание сразу или ожидает, если ОКУ Cpu3 занято. В противном случае транзакт направляется к блоку TERMINATE с меткой Met1 для уничтожения.

Приведем результаты моделирования.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND
CPU1	47373	0.500	37.997	1	573111	0
CPU2	63905	0.600	33.800	1	0	0
CPU3	263295	0.900	12.305	1	0	0

Коэффициенты использования оказались такими, как и требовалось разработчику. При этом вероятность обслуживания составила 0,81, т. е. 29% транзактов было потеряно.

Блок TEST можно использовать и для других проверок и в зависимости от их результатов изменять маршруты движения транзактов. Например:

TEST	L	P1,Gran
TEST	NE	FN1,X2
TEST	GE	Q\$Dlina,Nakor,Pered

В первом примере транзакту будет разрешено войти в блок TEST и перейти к следующему блоку лишь в том случае, когда значение его параметра 1 будет меньше значения переменной пользователя Gran.

Во втором примере транзакт будет проходить к следующему блоку только при неравенстве вычисленного значения функции номер 1 значению сохраняемой ячейки номер 2.

В третьем примере всегда входящие в блок TEST транзакты будут направлены к следующему блоку, если текущая длина очереди с именем Dlina будет больше или равна значению переменной пользователя Nakor. Если текущая длина очереди меньше переменной пользователя Nakor, транзакт направляется к блоку с меткой Pered.

2.6. Методы сбора статистики в имитационной модели

2.6.1. Регистратор очереди

В GPSS объекты типа «очередь» вводятся для сбора статистических данных. Эта статистика должна дать ответы на следующие вопросы:

- 1) Сколько раз транзакты приходили в очередь?
- 2) Сколько пришедших транзактов фактически присоединились к очереди и сколько их сразу без задержки заняло ОКУ?
- 3) Каково было максимальное значение длины очереди?
- 4) Каково было среднее число ожидающих транзактов в очереди?
- 5) Каково было среднее время ожидания тех транзактов, которым пришлось ждать?

GPSS обеспечивает возможность сбора такой статистики с помощью средства, называемого *регистратором очереди*. При использовании регистратора очереди в тех точках модели, где ресурсы ограничены, планировщик начинает автоматически собирать статистику, описывающее ожидание (если оно есть), возникающее в этих точках. Регистраторы очередей различают заданием имен. Условия задания имен те же, что и у устройств. Разработчик вносит регистратор очереди в модель с помощью пары взаимодополняющих блоков QUEUE (стать в очередь) и DEPART (покинуть очередь).

Формат записи блока QUEUE следующий:

QUEUE A, [B]

Блок QUEUE увеличивает длину очереди. Операнд A задает номер или имя очереди, к длине которой добавляются единицы. Операнд B определяет число единиц, на которое увеличивается текущая длина очереди. Если операнд B не используется, то прибавляется единица.

Рассмотрим примеры записи блока QUEUE

QUEUE RemQ

Увеличивает длину очереди RemQ на единицу при входе каждого транзакта

QUEUE P14, P1

Увеличивает длину очереди, номер которой задан в параметре P14 транзакта, на число единиц, заданных в параметре P1.

Формат записи блока DEPART имеет вид:

DEPART A, [B]

Блок DEPART служит для уменьшения длины очереди. Операнд A задает номер или имя очереди, длину которой надо уменьшить. Операнд B задает число единиц, на которое уменьшается длина очереди. Это число не должно превышать текущую длину очереди. Если операнд B не используется, то по умолчанию длина очереди уменьшается на 1. Операнды A и B в блоках QUEUE и DEPART могут быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр.

Приведем примеры записи блока DEPART.

DEPART RemQ

Уменьшает длину очереди RemQ на единицу

DEPART V3, (V4+3.7)

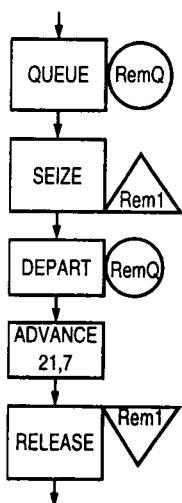
Операнды A и B заданы арифметической переменной V3 и выражением в скобках, которое также содержит арифметическую переменную V4. При входе транзакта в блок DEPART переменная и выражение в скобках вычисляются и округляются. После этого длина очереди, номер которой есть значение переменной V3, уменьшается на значение выражения в скобках ($V4 + 3.7$).

Очереди, как и ОКУ, разрешается записывать вместо операнда A номер без предварительного присвоения его имени командой EQU. Например:

QUEUE 5

DEPART 5

Рассмотрим использование блоков QUEUE и DEPART в модели на примере представленного ниже сегмента.



Ожидание может возникнуть ввиду занятости ОКУ с именем Rem1. Для сбора статистики об ожидании введем регистратор очереди и дадим ему имя RemQ. Если транзакт вошел в сегмент в момент, когда ОКУ Rem1 не занято, транзакт входит в блок QUEUE. Далее транзакт пытается войти в блок SEIZE и, поскольку Rem1 свободно, эта попытка оказывается успешной. Состояние Rem1 меняется на «занято», и далее транзакт сразу попадает в блок DEPART. Выполняется соответствующая подпрограмма и транзакт попадает в блок ADVANCE, где задерживается на некоторое время, вычисленное в соответствии с распределением 21 ± 7 .

Предположим теперь, что ОКУ Rem1 находится в занятом состоянии и следующий транзакт входит в сегмент модели. Он проходит в блок QUEUE и получает далее отказ, поскольку Rem1 находится в занятом состоянии. Транзакт перестает двигаться, оставаясь в блоке QUEUE.

Позднее, когда транзакт, находящийся на обслуживании в устройстве, покидает его, ожидающий транзакт опять попытается войти в блок SEIZE. На этот раз попытка окажется успешной. Двигаясь дальше, транзакт войдет в блок DEPART, уменьшая значение счетчика содержимого очереди на 1 и т. д.

Длина очереди не может быть отрицательной. Если такое может произойти, то происходит останов по ошибке «Запрещенная попытка сделать содержимое очереди отрицательным». Останов по ошибке «Недопустимое отрицательное число в операторе GPSS. Operand B.» происходит и тогда, когда при создании очереди делается попытка уменьшить ее длину. Например:

QUEUE Server,3

2.6.2. Статистические таблицы

Для получения плотности распределения, ее интегральных относительных частот, среднего и стандартного отклонения некоторых аргументов, которыми могут быть СЧА (например, времени нахождения транзакта в модели или задержки в ее отдельных частях, длин очередей, содержимого МКУ и т. д.), используются *статистические таблицы* TABLE и QTABLE. Команда описания таблицы TABLE имеет следующий формат:

Name TABLE A, B, C, D

Команда определяет аргумент, а также число и ширину частотных интервалов (классов). Метка Name определяет имя таблицы. Операндом A задается аргумент таблицы — элемент данных, чье частотное распределение будет табулироваться. Операнд может быть именем, выражением в скобках или СЧА. Операндом B задается верхний предел первого интервала. Операнд C задает ширину частотного интервала — разницу между верхней и нижней границей каждого частотного класса. Операнды B и C могут быть числами или string. Операндом D задается число частотных интервалов (положительное целое число).

Для сбора данных транзакт должен войти в блок TABULATE с тем же именем таблицы, которое определено в блоке TABLE. Блок помещается в ту точку модели, которая соответствует исследуемому объекту.

Блок TABULATE имеет следующий формат:

TABULATE A, [B]

Операндом A задается имя таблицы, в которую табулируется значение аргумента. Операндом B (*весовой коэффициент*) задается число единиц, которые должны быть занесены в тот частотный интервал, куда попало значение аргумента. Если операнд B отсутствует, то по умолчанию эта величина равна 1. Операнды A и B могут быть именем, выражением в скобках, СЧА или СЧА*параметр. Кроме того, операнд A может быть только положительным целым числом, а операнд B — положительным числом. Например:

```

      '      VrRem      TABLE      P$RealVs, 8.2, 5.5, 10
      . . .
      TABULATE      VrRem
  
```

Оператором TABLE описывается таблица с именем VrRem. Аргументом таблицы является СЧА P\$RealVs с верхним пределом первого интервала 8.2, шириной 5.5 и числом интервалов 10. Каждое значение табулируемого аргумента P\$RealVs, меньшее или равное 8.2, увеличивает частоту первого частотного класса таблицы на 1. Если аргумент таблицы не попадает в первый частотный класс, класс определяется делением значения аргумента на операнд C оператора TABLE. Например, значение P\$RealVs равно 30.25. Тогда $30.25/5.5=5.5$ и будет увеличена на 1 частота шестого класса. Если аргумент A таблицы больше $B+C\#D=8.2+5.5\#10=63.2$, изменен будет на 1 последний (десятый) класс. Од-

новременно корректируются текущие значения СЧА таблицы: счетчик входов в таблицу TC, среднее время ожидания TB и стандартное отклонение времени ожидания TD. Собранная в таблице статистика выводится в стандартный отчет. При использовании в операнде B блока TABULATE весового коэффициента, например:

TABULATE VrRem, 3

последний будет добавляться к значению частоты частотного класса. Весовой коэффициент применяется также для среднего и стандартного отклонения, что равносильно нескольким входам в блок TABULATE.

Таким образом, команда TABLE вместе с блоком TABULATE служат для табулирования любого СЧА.

Кроме таблиц TABLE могут использоваться Q-таблицы, являющиеся средством получения распределения только времени пребывания транзакта в очереди. Формат команды описания Q-таблицы такой же, как и TABLE. Отличие состоит в том, что *операндом A задается имя очереди*. Назначение операндов B, C, D такое же, что и в команде описания TABLE. Операнд B может быть нулем или положительным числом. Для создания в модели такой таблицы ее нужно предварительно определить с помощью команды QTABLE формата:

Name QTABLE A, B, C, D

Например:

VTime QTABLE Dlina, 18.2, 4.3, 6

Dlina — имя очереди к устройству, а не СЧА, как в случае использования TABLE.

При прохождении транзакта через блоки QUEUE и DEPART его время ожидания фиксируется, и к счетчику частотного интервала таблицы, в который попало это время, добавляется 1. Одновременно в таблице накапливается информация для вычисления среднего значения и среднеквадратического отклонения времени ожидания. Следует обратить внимание, что *при использовании QTABLE информация в таблицу заносится автоматически при входе транзакта в блоки QUEUE и DEPART и никаких специальных мер, т. е. блока TABULATE, при этом не требуется*. По окончании моделирования собранная в таблице информация также выводится в стандартном отчете GPSS.

2.7. Методы изменения маршрутов движения транзактов в модели

Для изменения маршрутов движения транзактов в модели применяются блоки DISPLACE, LOOP, GATE, TEST и TRANSFER. Здесь мы рассмотрим методы применения блоков TRANSFER, DISPLACE, LOOP, поскольку изложение порядка исполь-

зования блоков GATE и TEST уже начато и будет продолжено в последующих главах.

2.7.1. Блок TRANSFER

Блок TRANSFER (передать) предназначен для передачи входящего в него транзакта в любой другой блок модели. Он имеет следующий формат:

TRANSFER [A], [B], [C], [D]

Все режимы блока TRANSFER, кроме безусловного, *выборочные*, т. е. отличаются друг от друга способом выбора очередного блока, к которому должен быть направлен активный транзакт. Операнд A задает этот режим выбора. Существуют следующие девять режимов работы блока TRANSFER:

- , (по умолчанию) — безусловный;
- . — статистический, выбор случайным образом одного из двух блоков;
- BOTH — последовательный выбор одного из двух блоков;
- ALL — последовательный выбор одного из нескольких блоков;
- PICK — выбор случайным образом одного из нескольких блоков;
- FN — функциональный;
- P — параметрический;
- SBR — подпрограммный;
- SIM — одновременный.

Операнд A может принимать указанные выше значения, а также может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА*параметр.

Операнды B и C задают возможные значения номеров следующих блоков или их положение. Они могут быть такими же, как и операнд A. Использование этих значений будет описано ниже при рассмотрении указанных выше режимов работы. Если операнд B опущен, то планировщик записывает вместо него номер блока, следующего за блоком TRANSFER.

Рассмотрим режимы работы блока TRANSFER.

2.7.1.1. Режим безусловной передачи

В режиме безусловной передачи операнд A не используется. Операнд B указывает имя блока, в который транзакт должен попытаться войти. Блок TRANSFER не может отказать транзакту во входе. Например:

TRANSFER , Oper

После входа транзакт сразу же пытается войти в блок с меткой Oper. Если этот блок отказывает во входе, транзакт остается в блоке TRANSFER.

Рассмотрим еще один пример использования безусловного режима блока TRANSFER. Пусть требуется поток обслуженных транзактов перед удалением из модели разделить на четыре составляющие. Первый параметр каждого тран-

закта имеет одно из четырех присвоенных ранее значений: 1, 2, 3 или 4. Вначале для разделения потока используем блок TEST.

```

Met2      . . . . .
          TEST  E      P1, 1, Met2
          TERMINATE
          TEST  E      P1, 2, Met3
          TERMINATE
          TEST  E      P1, 3, Met4
          TERMINATE
Met4      TERMINATE

```

Теперь этот же фрагмент модели перепишем с использованием блока TRANSFER.

```

Met1      . . . . .
          TRANSFER      ,(Met1+P1)
          TERMINATE
          TERMINATE
          TERMINATE
          TERMINATE

```

В блоке TRANSFER в качестве операнда В указано выражение в скобках. При входе активного транзакта выражение вычисляется, т. е. к номеру, который присвоен планировщиком блоку с меткой Met1, прибавляется значение первого параметра. В итоге получается номер блока, к которому и направляется активный транзакт.

2.7.1.2. Режим статистической передачи

Когда операнд А используется и не является зарезервированным словом, блок TRANSFER работает в режиме передачи транзакта в один из двух блоков случайным образом.

Значение операнда А, записываемого после точки, рассматривается как трехзначное число, показывающее (в долях от тысячи), какая доля входящих в блок транзактов должна быть направлена в блок С. Остальные транзакты направляются в блок В или к следующему по номеру блоку, если операнд В опущен.

Числовое значение операнда А может быть задано любым СЧА. При этом возможны следующие случаи:

- значение операнда А меньше или равно нулю;
- значение операнда А равно или больше 1 000;
- значение операнда А больше нуля, но меньше 1 000.

Если вычисленное значение операнда А меньше или равно нулю, то будет производиться безусловная передача транзакта к блоку В. Если значение операнда А больше или равно 1 000, то будет осуществляться безусловная передача транзакта к блоку С. В третьем случае блок TRANSFER работает в обычном режиме.

Например:

```
TRANSFER      .P5,,Rrw
```

Трехзначное число, записанное в параметре 5 транзакта, входящего в блок TRANSFER, интерпретируется как вероятность (в долях от тысячи) того, что транзакт будет передаваться блоку Rrw, а в остальных случаях — следующему блоку, так как операнд В не используется.

Режим статистической передачи удобно использовать, например, в таких случаях. При моделировании работы цеха по производству деталей известно, что 12,5% изготовленных деталей бракуется. В модели это можно реализовать так:

```
TRANSFER      .125,Sam,Wzw
```

Транзакты, имитирующие изготовленные в цехе детали, в 12,5% случаев будут направлены к блоку с меткой Wzw, а в остальных 87,5% случаях — к блоку с меткой Sam.

Можно определить генератор — источник случайных чисел. Для этого нужно выбрать Edit ▶ Settings (Правка ▶ Настройки). Затем выбрать страницу Random Numbers (Случайные числа) и ввести номер генератора в поле ввода, отмеченное TRANSFER. После инсталляции по умолчанию используется генератор номер 1.

ЗАМЕЧАНИЕ

Блок TRANSFER в режиме статистической передачи моделирует единичное событие. Пусть известно, что некоторое событие Z (поступление заявки в систему массового обслуживания, отказ оборудования и т. д.) происходит с вероятностью $P(Z)$.

Моделью такого события является попадание значения x равномерно распределенной на отрезке $[0, 1]$ случайной величины в интервал $[0, P(Z)]$:

$$P[0 \leq x \leq P(Z)] = \int_0^{P(Z)} f(x) dx = \int_0^{P(Z)} 1 \cdot dx = x \Big|_0^{P(Z)} = P(Z).$$

Следовательно, свершилось или не свершилось реальное событие Z можно судить по значению x равномерно распределенной случайной величины. Если выбранное значение случайной величины удовлетворяет условию $x \leq P(Z)$, то событие Z свершилось. Если $x > P(Z)$, то свершившимся считают противоположное событие $\bar{Z}$.

2.7.1.3. Режим BOTH

Если в качестве операнда А используется ключевое слово BOTH (оба), то блок TRANSFER работает в режиме BOTH. В этом режиме каждый транзакт, вошедший в блок TRANSFER, проверяет два пути. Сначала проверяется возможность войти в блок В. Если транзакт не может войти в этот блок, он пытается войти в блок С.

Если транзакт не может войти и в этот блок, он задерживается в блоке TRANSFER. Опрос блоков В и С повторяется при каждом изменении текущего модельного времени до тех пор, пока транзакт не получит возможность войти в какой-либо блок. Если не используется операнд В, то проверяется возможность войти в следующий блок.

Пример 2.2. Заявки распределяются по двум ОКУ. Очередная заявка обслуживается первым освободившимся ОКУ.

```
; Пример 2.2
; Сегмент имитации поступления и обработки заявок
      GENERATE      (Exponential(315,0,11.7)); Источник заявок
      TRANSFER      BOTH,,Met2              ; Распределение заявок по ОКУ
;
      Имитация работы ОКУ1
      SEIZE          Rem1                    ; Занять ОКУ1
      ADVANCE        13.8,4.7                ; Обслуживание заявки
      RELEASE        Rem1                    ; Освободить ОКУ1
      TERMINATE      ; Обслуженные заявки ОКУ1
;
      Имитация работы ОКУ2
Met2   SEIZE          Rem2                    ; Занять ОКУ2
      ADVANCE        (Exponential(15,0,15.7)) ; Обслуживание заявки
      RELEASE        Rem2                    ; Освободить ОКУ2
      TERMINATE      ; Обслуженные заявки ОКУ2
; Сегмент задания времени моделирования
      GENERATE      3600
      TERMINATE      1
```

Транзакт, вошедший в блок TRANSFER, пытается перейти к следующему блоку SEIZE, так как операнд В не используется, и занять ОКУ Rem1. Если оно занято, то проверяется возможность входа транзакта в блок SEIZE с меткой Met1. Если и ОКУ Rem2 занято, транзакт остается в блоке TRANSFER. С изменением текущего модельного времени попытки продвижения транзакта, задержанного в блоке TRANSFER, повторяются.

Таким образом, блок TRANSFER в режиме BOTH выполняет роль «диспетчера», т. е. распределяет транзакты между двумя ОКУ.

В блоке TRANSFER операнды В и С могут быть и не метками, а номерами блоков. Конечно, это неудобно, так как разработчику нужно самому определять номера блоков в модели. В примере 2.2 можно записать

```
TRANSFER      BOTH,,7,4
```

2.7.1.4. Режим ALL

Для работы в режиме ALL необходимо в качестве операнда А использовать ключевое слово ALL. В этом режиме вошедший в блок TRANSFER транзакт проверяет возможность войти в любой блок, начиная с блока, указанного операндом В, и заканчивая блоком, указанным операндом С. Операнд D опреде-

ляет шаг d изменения номера проверяемого блока. Значение шага дает возможность опрашивать определенные блоки, расположенные между теми, которые заданы операндами В и С. Далее транзакт, вошедший в блок TRANSFER, пытается войти в блок В. Если блок В занят, транзакт пытается войти в блоки с номерами $v + d, v + 2d, \dots, z$, где v — номер блока В, z — номер блока С. При этом $z = v + nd$, где n — некоторое целое положительное число.

Если не используется операнд С, то проверяется только один блок.

Поскольку обычно в качестве операндов В и С записываются метки блоков, то блоки следует располагать таким образом, чтобы при присвоении номеров разность между номерами блоков, указанных в операндах В и С, была кратна шагу, указанному в операнде D. Например:

TRANSFER ALL, Met1, Met2, 4

Здесь режим ALL допустим в том случае, если разность между номерами, присвоенными блокам Met1 и Met2, кратна 4. В модели это может выглядеть следующим образом.

Пример 2.3. Заявки распределяются по трем ОКУ. Очередная заявка обслуживается первым освободившимся ОКУ.

; Пример 2.3

; Сегмент имитации поступления и обработки заявок

GENERATE (Exponential(315,0,11.7)); Источник заявок
TRANSFER All, Met1, Met2 ; Распределение заявок

; *Имитация работы ОКУ1*

Met1 SEIZE Apd1 ; Занять ОКУ1
ADVANCE 23.5, 12.1 ; Обслуживание заявки
RELEASE Apd1 ; Освободить ОКУ1
TERMINATE ; Обслуженные заявки ОКУ1

; *Имитация работы ОКУ2*

SEIZE Apd2 ; Занять ОКУ2
ADVANCE (Normal(357,28.9,13.8)) ; Обслуживание заявки
RELEASE Apd2 ; Освободить ОКУ2
TERMINATE ; Обслуженные заявки ОКУ2

; *Имитация работы ОКУ3*

Met2 SEIZE Apd3 ; Занять ОКУ3
ADVANCE (Exponential(15,0,15.7)) ; Обслуживание заявки
RELEASE Apd3 ; Освободить ОКУ3
TERMINATE ; Обслуженные заявки ОКУ3

; Сегмент задания времени моделирования

GENERATE 3600
TERMINATE 1

При входе транзакта в блок TRANSFER последовательно проверяется возможность входа сначала в блок SEIZE, помеченный меткой Met1. Потом, если этот

блок не позволяет войти, во второй блок SEIZE без метки, так как разность между номером этого блока и блока с меткой Met1 равна 4. При невозможности войти во второй блок SEIZE проверяется третий блок SEIZE с меткой Met2. Если и в него транзакт не может войти, он остается в блоке TRANSFER, и с каждым изменением текущего модельного времени проверки возможности входа вновь повторяются сначала, т. е. с блока SEIZE с меткой Met1.

В блоке TRANSFER операнды В и С также могут быть и не метками, а номерами блоков. Если в примере 2.3 записать

TRANSFER ALL, 3, 11, 4

результат моделирования будет тот же.

ЗАМЕЧАНИЕ

Условными являются только режимы BOTH и ALL. Во всех остальных режимах выбор следующего блока производится в момент входа транзакта в блок. Следует отметить, что когда планировщик при просмотре списка текущих событий обнаруживает транзакт, задержанный в блоках TRANSFER BOTH либо TRANSFER ALL, он пытается продвинуть транзакт, начиная с блока В. Следовательно, в режиме BOTH в тех случаях, когда возможен переход к обоим блокам В и С, блок В имеет некоторое преимущество. Аналогично и в режиме ALL в случае, когда возможен переход к нескольким блокам, блоки с меньшими номерами имеют некоторое преимущество перед блоками с большими номерами.

Подтвердим это результатами моделирования примера 2.3.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND
APD1	11503	0.750	23.474	1	30990	0
APD2	8638	0.693	28.864	1	0	0
APD3	10749	0.468	15.677	1	0	0

Результаты моделирования показывают, что наибольший коэффициент использования имеет ОКУ Apd1, а наименьший — ОКУ Apd3, хотя среднее время обслуживания ОКУ Apd3 меньше, чем ОКУ Apd1 и оно за одно и то же время могло бы обслужить большее число транзактов, т. е. быть более загруженным.

Блок TRANSFER в режиме ALL также выполняет роль «диспетчера», но в отличие от блока BOTH он распределяет транзакты между несколькими ОКУ.

2.7.1.5. Режим PICK

Для работы блока TRANSFER в режиме PICK необходимо в качестве операнда А использовать ключевое слово PICK. В этом режиме из последовательности блоков с номерами $v, v+1, v+2, \dots, n$, где v — номер блока, указанного операндом В, а n — номер блока, указанного операндом С, случайным образом выбирается номер одного блока, к которому должен быть направлен транзакт. Все блоки, включая и указанные операндами В и С, выбираются с равной вероятностью $1/[(n-v)+1]$. Транзакт пытается перейти только к выбранному для него блоку. Если транзакт не может сразу перейти к следующему блоку,

то он будет ждать в блоке TRANSFER до тех пор, пока не будет снято блокирующее условие. Номер блока C должен быть больше или равен $v + 1$.

Пример 2.4. Заявки распределяются по четырем ОКУ, имеющим одинаковое время обслуживания. Вероятность попадания на любое из ОКУ $P = 1/4 = 0,25$. Если в режимах BOTH и ALL очередная заявка занимала первое освободившееся ОКУ, то в режиме PICK заявка ожидает освобождения того ОКУ, на которое оно было распределено. Так как блоки, на которые следует направить заявки, в модели расположены не один за другим, использована последовательность блоков TRANSFER в режиме безусловной передачи.

; Пример 2.4

; Сегмент имитации поступления и обработки заявок

```

        GENERATE      (Exponential(315,0,11.7)); Источники заявок
        TRANSFER      PICK, Met1, Met2           ; Распределение заявок
Met1     TRANSFER      , Met3                     ; Направить на ОКУ1
        TRANSFER      , Met4                     ; Направить на ОКУ2
        TRANSFER      , Met5                     ; Направить на ОКУ3
Met2     TRANSFER      , Met6                     ; Направить на ОКУ4
;
;      Имитация работы ОКУ1
Met3     SEIZE        Apd1                       ; Занять ОКУ1
        ADVANCE       23.5, 12.1                 ; Обслуживание заявки
        RELEASE       Apd1                       ; Освободить ОКУ1
        TERMINATE     ; Обслуженные заявки ОКУ1
;
;      Имитация работы ОКУ2
Met4     SEIZE        Apd2                       ; Занять ОКУ2
        ADVANCE       23.5, 12.1                 ; Обслуживание заявки
        RELEASE       Apd2                       ; Освободить ОКУ2
        TERMINATE     ; Обслуженные заявки ОКУ2
;
;      Имитация работы ОКУ3
Met5     SEIZE        Apd3                       ; Занять ОКУ3
        ADVANCE       23.5, 12.1                 ; Обслуживание заявки
        RELEASE       Apd3                       ; Освободить ОКУ3
        TERMINATE     ; Обслуженные заявки ОКУ3
;
;      Имитация работы ОКУ4
Met6     SEIZE        Apd4                       ; Занять ОКУ4
        ADVANCE       23.5, 12.1                 ; Обслуживание заявки
        RELEASE       Apd4                       ; Освободить ОКУ4
        TERMINATE     ; Обслуженные заявки ОКУ4

```

; Сегмент задания времени моделирования

```

        GENERATE      3600
        TERMINATE     1

```

Приведем результаты моделирования примера 2.4.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND
APD3	2307513	0.502	23.499	1	0	0
APD2	2307303	0.502	23.501	1	9259811	0
APD4	2308180	0.502	23.492	1	9259817	0
APD1	2306819	0.502	23.500	1	9259812	0

По результатам моделирования (30 000 прогонов) видно, что коэффициенты использования всех ОКУ одинаковы и равны 0,502. Незначительные отличия в числе вошедших в ОКУ транзактов и времени обслуживания одного транзакта вызваны генераторами случайных чисел. Для вошедших в ОКУ транзактов относительное отклонение в среднем составит $[(2308180-2306819)/9259812]/2=[0,2493-0,2491]/2=0,0001$, а для времени обслуживания — 0,0045 единицы.

2.7.1.6. Функциональный режим

В функциональном режиме блок TRANSFER работает тогда, когда вместо операнда А используется ключевое слово FN. При входе транзакта в блок TRANSFER вычисляется значение функции, имя которой задано операндом В. Если результат не целый, то от него берется целая часть. Для определения номера следующего блока полученное целое складывается с операндом С (значение операнда С может быть записано равным нулю). Если блок с вычисленным номером занят, транзакт остается в блоке TRANSFER до тех пор, пока не сможет перейти именно к этому блоку. Например:

```
TRANSFER    FN, Vibor, P6
TRANSFER    FN, 7, P6
```

В первом примере транзакт направляется к блоку, номер которого получается как результат сложения вычисленного и округленного до целого значения функции с именем Vibor и значения параметра 6 вошедшего транзакта. Во втором примере все так же, только функция задана не именем, а номером.

ЗАМЕЧАНИЕ

Другие примеры применения блока TRANSFER в функциональном режиме, расширяющие возможности блока TRANSFER в режимах ALL и PICK, будут рассмотрены в главе 6.

2.7.1.7. Параметрический режим

Для задания параметрического режима необходимо в качестве операнда А указать ключевое слово Р. В параметрическом режиме активный транзакт направляется к блоку, номер которого определяется как сумма значения параметра и значения операнда С. Если операнд С не используется, значение параметра и является номером нового блока. Например:

```
TRANSFER    P, Novi, 2
TRANSFER    P, 5, 2
```

В первом примере транзакт направляется к блоку, номер которого равен сумме значения параметра с именем *Novi* и значения операнда *C*, т. е. 2. Во втором примере все так же, только параметр задан не именем, а номером.

2.7.1.8. Подпрограммный режим

Для задания подпрограммного режима необходимо в качестве операнда *A* указать ключевое слово *SBR*. В режиме *SBR* активный транзакт всегда направляется к блоку, номер которого указан операндом *B*. Номер блока *TRANSFER* помещается в параметр, указанный операндом *C*. Например:

```
TRANSFER      SBR, Novi, Ist
```

Вошедший в блок транзакт направляется к блоку с меткой *Novi*, а номер блока *TRANSFER* записывается в параметр с именем *Ist*. Если такого параметра нет, он создается.

Для возвращения из подпрограммы используется блок *TRANSFER* в параметрическом режиме (см. 2.7.1.7).

2.7.1.9. Одновременный режим

Режим *SIM* (одновременный) введен для случая, когда требуется одновременное выполнение нескольких условий. Каждый транзакт имеет свой индикатор задержки *SIM*. В этом индикаторе записывается результат любой попытки транзакта войти в следующий блок. Если планировщик обнаруживает условия, препятствующие входу транзакта в блок, индикатор *SIM* этого транзакта устанавливается в 1. Если все условия перехода к следующему блоку удовлетворяются, индикатор *SIM* остается равным нулю. Если не выполняется хотя бы одно из условий, индикатор *SIM* устанавливается в 1. В таких случаях операнд *C* указывает блок, в котором проверялось первое условие, и транзакт выполняет проверки всех условий, пока они не будут удовлетворяться одновременно. Блок *ADVANCE* также устанавливает индикатор *SIM* в 0. Проверка состояний связана с блоками, которые могут задерживать транзакты. Для проверки условий используется блок *GATE*.

Например:

```
Met1      GATE  NU      Mas1
           GATE  NU      Mas2
           GATE  NU      Mas3
           TRANSFER    SIM, ,Met1
           SEIZE        Rem1
```

В приведенном примере транзакт не может пройти к блоку *SEIZE*, если *OKY Mas1*, *Mas2* и *Mas3* не будут одновременно свободны. При входе транзакта в блок *TRANSFER* проверяется значение его индикатора *SIM*. Если транзакт был задержан в каком-либо из блоков *GATE*, его индикатор установлен в 1.

При проверке индикатора в блоке TRANSFER выясняется, что транзакт был задержан, он отсылается к тому блоку, где производилась первая проверка (Met1), и вся последовательность проверок повторяется. При задержке транзактов в блоке ASSEMBLE, GATHER или MATCH индикатор SIM в 1 не устанавливается.

ЗАМЕЧАНИЕ

Режим SIM используется редко, так как с помощью булевых переменных и блока TEST можно более эффективно управлять состоянием большого количества объектов.

2.7.2. Блок LOOP

Блок LOOP предназначен для организации в модели циклов. Он имеет следующий формат:

LOOP A, B

Операнд A — параметр транзакта или параметр цикла, в котором содержится число повторений какого-либо участка модели.

Операнд B — метка блока, с которого начинается цикл.

Операнды A и B могут быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр.

Например:

LOOP KolPovt, Start1

Пусть блок с меткой Start1 является началом цикла, т. е. расположен раньше блока LOOP. Когда транзакт, пройдя участок модели, начинающийся блоком с меткой Start1, войдет в блок LOOP, значение его параметра с именем KolPovt уменьшается на 1. Если это значение не равно нулю, транзакт переходит к блоку с меткой Start1, т. е. цикл повторяется. Если же после вычитания 1 значение параметра цикла равно нулю, следовательно, выполнено заданное число повторений, транзакт переходит к следующему блоку.

Если при входе транзакта в блок LOOP окажется, что транзакта с именем KolPovt нет, произойдет останов по ошибке «Обращение к несуществующему параметру».

ЗАМЕЧАНИЕ

При каждом входе транзакта в блок LOOP значение параметра цикла уменьшается на 1. Значит, значение параметра цикла можно использовать для выбора новых значений каких-либо факторов при очередном повторении. Однако это неудобно, так как величина параметра цикла изменяется от большего к меньшему значению, а не наоборот.

ЗАМЕЧАНИЕ

Функции, выполняемые блоком LOOP, могут быть реализованы, например, следующим образом:

	...	
	ASSIGN	2, 1
Met1	...	
	ASSIGN	2+, 1
	TEST G	P2, 10, Met1

Первым блоком ASSIGN в параметр 2 активного транзакта заносится 1. Блок с меткой Met1 является началом цикла. После выполнения последовательности блоков, составляющих тело цикла, второй блок ASSIGN увеличивает значение параметра 2 на 1. Блок TEST проверяет значение параметра 2. Если оно меньше десяти, транзакт направляется к блоку с меткой Met1 и вновь повторяется цикл, если больше — транзакт проходит к следующему по порядку блоку. Значения параметра 2 изменяются в порядке возрастания. Однако можно менять их точно так же, как и в блоке LOOP.

Пример 2.5. Одноканальная СМО с ожиданием и ограниченной очередью на три заявки. Требуется в любой момент времени знать номера транзактов, находящихся в очереди и на обслуживании в ОКУ.

; Пример 2.5

; Задание длины очереди и булевой переменной

Emk EQU 3
Kont1 BARIABLE (Q\$Dlina<Emk) ; Проверка длины очереди

; Сегмент имитации поступления и обслуживания заявок

GENERATE 2,,7 ; Источник заявок
TEST E BV\$Kont1,1, Met1 ; Есть ли место в очереди?
QUEUE Dlina ; Да, встать в очередь

; *Учет номеров транзактов, вошедших в очередь*

ASSIGN 1, Emk ; Запись в параметр цикла
Met2 TEST E X+1, 0, Met3 ; Есть ли свободное место?
SAVEVALUE P1, XN1 ; Да, записать номер транзакта
TRANSFER , Met4 ; Выйти из цикла
Met3 LOOP 1, Met2 ; Повторить или конец цикла

;

Met4 SEIZE Stan ; Занять ОКУ
DEPART Dlina ; Покинуть очередь

; *Учет номеров транзактов, покинувших очередь*

ASSIGN 1, Emk ; Запись в параметр цикла
Met5 TEST E X+1, XN1, Met6 ; Есть ли этот номер транзакта?
SAVEVALUE P1, 0 ; Да, тогда удалить
TRANSFER , Met7 ; Выйти из цикла
Met6 LOOP 1, Met5 ; Повторить или конец цикла

;


```
;      Учет номера транзакта, находящегося на обслуживании
Met7   SAVEVALUE      (Emk+1),XN1 ; Записать номер транзакта
;
      ADVANCE          4.5          ; Имитация обслуживания
      RELEASE          Stan         ; Освободить ОКУ
      TERMINATE        ; Обслуженные заявки
Met1   TERMINATE        ; Потерянные заявки
; Задание времени моделирования
      GENERATE          15
      TERMINATE         1
```

Блок GENERATE генерирует семь транзактов с интервалами 2 единицы модельного времени.

Результаты моделирования представлены в табл. 2.1.

Первый блок TEST пропускает транзакт к следующему блоку QUEUE, если в очереди на три заявки есть место. После постановки в очередь начинает работать сегмент учета номеров транзактов, вставших в очередь. Так как длина очереди ограничена тремя заявками, то в первый параметр — параметр цикла записывается число три. Далее организуется с помощью блока LOOP цикл, в котором отыскивается возможность записи номера очередного транзакта. Записать номер транзакта можно будет тогда, когда значение какой-нибудь одной из трех ячеек сохраняемых величин X1, X2 или X3 будет равно нулю. Таких ячеек может быть больше одной, а транзакт поступает один, поэтому после записи номера осуществляется выход из цикла.

Таблица 2.1

Результаты моделирования примера 2.5

События	Транзакты						
	1(1)*	2(3)	3(4)	4(5)	5(6)	6(7)	7(8)
Вход в модель	2	4	6	8	10	12	14
Занятие очереди	2	4	6	8	10	12	14
В очереди транзакты		3	3, 4	4, 5	4, 5, 6	5, 6, 7	5, 6, 7
Занятие ОКУ	2	6.5	11				
Освобождение ОКУ	6.5	11	15.5				
Вывод из модели	6.5	11	15.5				14

* Блок GENERATE сгенерировал семь транзактов, время прохождения которых и представлено в табл. 2.1. В их нумерации, данной в скобках, отсутствует номер два (XN1=2), так как этот номер присвоен единственному транзакту блока GENERATE сегмента задания времени моделирования.

Транзакт занимает блок SEIZE с меткой Met3 и покидает очередь с именем Dlina. Теперь начинает работать сегмент учета номеров транзактов, покинувших

очередь. Вначале в параметр цикла записывается число три (по длине очереди). Затем в цикле находится одна из сохраняемых ячеек X1, X2 или X3, в которой записан номер транзакта, покидающего очередь. Эта ячейка обнуляется. Так как учет номеров транзактов, находящихся в очереди и на обслуживании, ведется раздельно, то в сохраняемую ячейку с номером (E_{mk+1}), т. е. $3 + 1 = 4$ записывается номер транзакта, занявшего ОКУ Stan.

После завершения обслуживания ОКУ Stan освобождается, а сохраняемая ячейка с номером (E_{mk+1}) обнуляется, т. е. обслуженный транзакт списывается с учета и выводится из модели.

Анализ результатов моделирования подтверждает правильность работы модели. В $t = 14$ восьмой транзакт удален из модели, так как очередь заполнена тремя транзактами. Моделирование завершается в $t = 15$. В очереди находятся пятый, шестой и седьмой транзакты, а на обслуживании в ОКУ — четвертый.

2.7.3. Блок DISPLACE

Блок DISPLACE предназначен для нахождения любого транзакта и перемещения его к новому блоку. Блок DISPLACE имеет формат:

DISPLACE A, B, [C], [D]

Операнд A — номер транзакта, который необходимо переместить.

Операнд B — метка блока, к которому перемещается транзакт, указанный операндом A.

Операнд C — номер параметра перемещаемого транзакта, в который записывается оставшееся до конца его обслуживания время, если он находился в списке будущих событий.

Операнд D — метка альтернативного блока для активного транзакта.

Операнды A, B, C и D могут быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр. Например:

DISPLACE (P2+32), Term3, 0statok, Met2

Операнд A указан выражением в скобках. Это выражение вычисляется и округляется до целого. Полученный результат является номером транзакта, который следует переместить. Далее блок DISPLACE отыскивает этот транзакт. При этом возможны случаи:

- транзакт есть в модели и не находится в списке будущих событий;
- транзакт есть в модели и находится в списке будущих событий;
- транзакта с нужным номером нет в модели.

В первом случае транзакт перемещается к блоку с меткой Term3. Во втором случае определяется время, оставшееся до его повторного ввода в процесс моделирования, и записывается в параметр с именем 0statok. Если параметра с таким именем нет, он создается. Транзакт также перемещается к блоку с меткой Term3. В третьем случае, т. е. когда в модели нет транзакта с нужным

номером, активный транзакт, вошедший в блок DISPLACE, направляется к блоку с меткой Met2. Если операнда D нет, активный транзакт переходит к следующему блоку.

Когда транзакт перемещается к новому блоку, он исключается из списков:

- будущих событий;
- отложенных прерываний (для прерывающих транзактов);
- задержки (в порядке приоритета);
- пользователя;
- повторных попыток.

И не исключается из списков:

- текущих событий;
- прерываний (для прерванных транзактов);
- групп.

При перемещении прерванные выполнения в устройствах не сбрасываются. Это означает, что перемещаемый транзакт продолжает занимать устройство.

Приведем пример использования блока DISPLACE.

Пример 2.6. С узла связи, имеющего буфер, по каналу связи передаются сообщения. Во время передачи возникают помехи, в результате воздействия которых передача прекращается и сообщение теряется. После этого передается очередное сообщение. Кроме того, из находящихся в буфере и ожидающих передачи сообщений, некоторые могут быть удалены, например, вследствие ошибок и т. д.

Для построения модели воспользуемся примером 2.5, в который добавим сегмент имитации помех. Для лучшего понимания механизма работы блока DISPLACE предположим, что помехи возникают дважды в $t = 4,5$ и в $t = 9$.

При этом нужно удалить транзакты номер четыре и пять соответственно.

Модель представлена ниже.

```
; Пример 2.6
; Задание длины очереди и булевой переменной
Emk      EQU      3
Kont1    BARIABLE (Q$Dlina<Emk)    ; Проверка длины очереди
; Сегмент имитации и передачи сообщений
          GENERATE  2,,,7            ; Источник сообщений
          TEST E    BV$Kont1,1,Met1   ; Есть ли место в очереди?
          QUEUE     Dlina             ; Встать в очередь
;
;      Учет номеров транзактов, вставших в очередь
          ASSIGN    1,Emk             ; Запись в параметр цикла
Met2      TEST E    X*1,0,Met3        ; Есть ли свободное место?
          SAVEVALUE P1,XN1            ; Да, записать номер транзакта
          TRANSFER  ,Met4             ; Выйти из цикла
Met3      LOOP      1,Met2            ; Повторить или конец цикла
;
```

```

Met4      SEIZE      Stan      ; Занять канал
          DEPART     Dlina     ; Покинуть очередь
;         Учет номеров транзактов, покинувших очередь
          ASSIGN     1,Emk      ; Запись в параметр цикла
Met5      TEST E     X*1,XN1,Met6 ; Есть ли этот номер транзакта?
          SAVEVALUE  P1,0       ; Да, тогда снять с учета
          TRANSFER   ,Met7      ; Выйти из цикла
Met6      LOOP       1,Met5     ; Повторить или конец цикла
;
;         Учет номера транзакта, находящегося на обслуживании
Met7      SAVEVALUE  (Emk+1),XN1 ; Записать номер транзакта
;
          ADVANCE    4.5        ; Имитация передачи
          RELEASE    Stan       ; Освободить канал
          SAVEVALUE  (Emk+1),0 ; Удалить номер обслуженного транзакта
          TERMINATE                ; Переданные сообщения
Met1      TERMINATE                ; Потерянные сообщения
; Сегмент имитации помех
          GENERATE   4.5,,,2     ; Источник помех
;         Удаление транзактов
          ASSIGN     1,Emk      ; Запись в параметр цикла
Met8      TEST E     X*1,4,Met9  ; Есть ли транзакт номер 4?
          DISPLACE   X*1,Met10   ; Да, тогда удалить из очереди
          SAVEVALUE  P1,0       ; Снять с учета
          TRANSFER   ,Met11      ; Выйти из цикла
Met9      LOOP       1,Met8     ; Начало или конец цикла
          TEST E     X4,5,Met11  ; Есть ли транзакт номер 5?
          DISPLACE   X4,Met12    ; Да, тогда удалить из ОКУ
          SAVEVALUE  (Emk+1),0   ; Снять с учета
Met11     TERMINATE                ; Вывод транзактов – имитаторов помех
Met10     DEPART     Dlina     ; Покинуть очередь
          TERMINATE                ; Потерянные сообщения
Met12     RELEASE    Stan       ; Освободить канал
          TERMINATE                ; Потерянные сообщения
; Сегмент задания времени моделирования
          GENERATE   20
          TERMINATE   1

```

Нам известно из примера 2.5, как производится учет номеров транзактов, находящихся в очереди и на обслуживании. Рассмотрим работу модели с включенным в нее сегментом имитации помех.

Результаты моделирования представлены в табл. 2.2.

В $t = 4,5$ блок GENERATE сегмента имитации помех генерирует транзакт. Сначала отыскивается транзакт с нужным номером в очереди. Для этого блоком ASSIGN в первый параметр активного транзакта — параметр цикла заносится число три (по длине очереди). Далее поиск производится в цикле, организованном с помощью блока LOOP. Если транзакт найден в очереди (в данном случае транзакт номер четыре находится в очереди), блоком DISPLACE он перемещается к блоку DEPART с меткой Met10. Транзакт снимается с учета путем обнуления соответствующей сохраняемой ячейки.

Если транзакт с нужным номером в очереди не найден, проверяется его нахождение на обслуживании. Если он обслуживается, блок DISPLACE удаляет его к блоку RELEASE с меткой Met12. В рассматриваемом примере так и произошло в $t = 9$ с транзактом номер пять.

Моделирование закончилось в $t = 20$. В это время транзакт номер девять находился на обслуживании, а транзакт номер десять — в очереди (см. табл. 2.2).

Таблица 2.2

Результаты моделирования примера 2.6

События	Транзакты						
	1(1)*	2(4)	3(5)	4(7)	5(8)	6(9)	7(10)
Вход в модель	2	4	6	8	10	12	14
Занятие очереди	2	4	6	8	10	12	14
В очереди транзакты		4	5	7	8	8, 9	10
Занятие ОКУ	2		6.5	9	13.5	18	
Перемещение транзакта		4.5	9				
Освобождение ОКУ	6.5	4.5	9	13.5	18		
Вывод из модели	6.5	4.5	9	13.5	18		

* Даны порядковые номера транзактов, которые должны обслуживаться ОКУ Stan. В скобках даны номера этих же транзактов, присвоенные системой GPSS. Номера два (XN1=2) и шесть (XN1=6) имеют транзакты блока GENERATE сегмента имитации неисправностей, а номер три (XN1=3) — блока GENERATE сегмента задания времени моделирования.

ЗАМЕЧАНИЕ

Ранее отмечалось, что при перемещении прерванные обслуживания в ОКУ не сбрасываются, т. е. транзакт продолжает занимать ОКУ. Поэтому если необходимо по логике модели освободить устройство, транзакт должен пройти через соответствующий блок RELEASE, что и показано в модели.

Аналогично в случае удаления из очереди: транзакт должен пройти через соответствующий блок DEPART, иначе он будет считаться находящимся в очереди, что нарушит логику работы модели.

2.8. Примеры построения моделей с одноканальным устройством обслуживания

Пример 2.7. В ремонтное подразделение с одним каналом обслуживания поступают вышедшие из строя средства связи (СС), требующие текущего ремонта (ТР). Интервалы времени поступления неисправных СС распределены равномерно в интервале 18 ± 6 часов. Время ремонта также распределено равномерно в интервале 16 ± 4 часа. Ремонт производится по мере поступления: первым поступило — первым отремонтировано.

Необходимо промоделировать функционирование ремонтного подразделения в течение 3 суток (72 часов). Модель на GPSS должна обеспечить сбор статистики об очереди.

Построим модель в виде последовательности блоков, добавив к ней элементы, предназначенные для управления моделированием. Последовательность блоков в модели соответствует последовательности ситуаций, в которых оказывается неисправное СС в реальной системе. Неисправные СС поступают, если необходимо, то ждут очереди, затем передаются в ремонтное подразделение непосредственно для ремонта. Единица модельного времени равна 1 мин, тогда время моделирования составит $72 \cdot 60 = 4320$ единиц.

Для построения модели введем следующие идентификаторы:

- Rem1 — ремонтное подразделение с одним каналом обслуживания;
- RemQ — очередь неисправных средств связи в ремонтное подразделение;
- VrRem — имя таблицы, в которой будет табулироваться время пребывания неисправного средства связи в ремонтном подразделении.

Ниже приведена программа на GPSS.

Программа имеет два сегмента. Первый из них имитирует процесс ремонта неисправных СС, а второй определяет длительность процесса моделирования.

; Пример 2.7

; Задание таблицы

VrRem TABLE M1,800,50,10 ; Определение таблицы

; Сегмент моделирования процесса ремонта

GENERATE 1080,480 ; Источник неисправных СС

QUEUE RemQ ; Встать в очередь

SEIZE Rem1 ; Занять ремонтное подразделение

DEPART RemQ ; Покинуть очередь

ADVANCE 960,240 ; Имитация ремонта

RELEASE Rem1 ; Освободить ремонтное подразделение

TABULATE VrRem ; Данные в таблицу

TERMINATE ; Отремонтированные СС

; Сегмент задания времени моделирования

GENERATE 4320

TERMINATE 1

START 1000

Моделирование начинается сразу после трансляции, так как в модели имеется команда START.

ЗАМЕЧАНИЕ

В приводимой модели непосредственно указываются числовые значения используемых переменных. В этом случае для моделирования при других исходных данных потребуется внести соответствующие изменения. При большом числе переменных это затруднительно. Поэтому целесообразно исключить числовые значения переменных непосредственно из той части модели, которая описывает функционирование системы с помощью блоков, и представить их в виде сохраняемых величин, переменных пользователя и т. д.

Как уже отмечалось, это описание следует осуществить перед блоковой частью модели. При таком подходе к написанию программы исходные данные будут сосредоточены по возможности в одном обозримом месте, что облегчит в последующем их корректировку.

Пример 2.8. В ремонтное подразделение с одним каналом обслуживания могут поступать неисправные СС двух типов. СС первого и второго типов ремонтируются одними и теми же мастерами ремонтного отделения. Интервалы времени поступления СС первого типа распределены равномерно 20 ± 10 часов. Распределение интервалов времени поступления СС второго типа 15 ± 8 часов. Поступающие СС ремонтируются в последовательности: первым поступило — первым отремонтировано. На ремонт СС первого типа затрачивается 6 ± 2 часа, второго типа — 8 ± 4 часов.

Функционирование ремонтного подразделения можно промоделировать посредством двух независимых сегментов или последовательностей блоков. Первая последовательность моделирует ремонт СС первого типа, а вторая — второго типа. В качестве единицы модельного времени взята 1 мин. Время моделирования — 3 суток (72 часа).

При построения модели используем следующие идентификаторы:

- Rem1 — ремонтное подразделение с одним каналом обслуживания;
- RemQ — общая очередь неисправных СС в ремонт;
- RemQ1 — очередь неисправных СС первого типа;
- RemQ2 — очередь неисправных СС второго типа;
- VrRem — имя таблицы, в которой будет табулироваться общее время нахождения неисправного СС в ремонте;
- VrRem1 — имя таблицы, в которой будет табулироваться время нахождения в ремонте неисправного СС первого типа;
- VrRem2 — имя таблицы, в которой будет табулироваться время нахождения в ремонте неисправного СС второго типа.

; Пример 2.8

; Определение таблиц

VrRem1 TABLE M1,420,180,5

VrRem2 TABLE M1,420,180,5

VrRem TABLE M1,420,180,5

; Сегмент имитации ремонта СС первого типа

GENERATE 600,300 ; Источник СС первого типа

QUEUE RemQ ; Встать в общую очередь

QUEUE RemQ1 ; Встать в очередь СС первого типа

SEIZE Rem1 ; Занять ремонтное подразделение

DEPART RemQ ; Покинуть общую очередь

DEPART RemQ1 ; Покинуть очередь СС первого типа

ADVANCE 360,120 ; Имитация ремонта

RELEASE Rem1 ; Освободить ремонтное подразделение

TABULATE VrRem1 ; Данные о СС первого типа

TRANSFER , Met1 ; Отремонтированные СС первого типа

; Сегмент имитации ремонта СС второго типа

GENERATE 720,480 ; Источник СС второго типа

QUEUE RemQ ; Встать в общую очередь

QUEUE RemQ2 ; Встать в очередь СС второго типа

SEIZE Rem1 ; Занять ремонтное подразделение

DEPART RemQ ; Покинуть общую очередь

DEPART RemQ2 ; Покинуть очередь СС второго типа

ADVANCE 480,240 ; Имитация ремонта

RELEASE Rem1 ; Освободить ремонтное подразделение

TABULATE VrRem2 ; Данные о СС второго типа в таблицу

Met1 TABULATE VrRem ; Данные о СС обоих типов в таблицу

TERMINATE ; Отремонтированные СС первого и второго типов

; Сегмент задания времени моделирования

GENERATE 4320

TERMINATE 1

В данном примере очередь с именем RemQ и ОКУ с именем Rem1 используются в двух сегментах модели. Следовательно, одну и ту же очередь или ОКУ можно использовать в разных частях модели столько раз, сколько этого требует логика функционирования моделируемой системы.

Однако возможен и другой вариант построения этой же модели. Рассмотрим такой вариант в примере 2.9.

Пример 2.9. Условия те же, что и в примере 2.8. Только время ремонта неисправных СС первого и второго типов предполагается распределенным по экспоненциальному закону со средними значениями 600 и 480 мин соответственно. Выделим, как это и есть на практике, два источника поступления неисправных СС. Действия ремонтного подразделения в отличие от предыдущего варианта моделируются с помощью одного ОКУ.

; Пример 2.9

; Определение таблиц

VrRem1 TABLE M1,420,180,5

VrRem2 TABLE M1,420,180,5

VrRem TABLE M1,420,180,5

; Сегмент имитации поступления неисправных СС первого типа

GENERATE 600,300 ; Источник СС первого типа

ASSIGN 1,1 ; Код СС первого типа

ASSIGN 2,(Exponential(271,0,600)); Время ремонта

; Сегмент имитации действий ремонтного подразделения

Met1 QUEUE RemQ ; Встать в общую очередь

QUEUE P1 ; Встать в очередь СС по типу

SEIZE Rem1 ; Занять ремонтное подразделение

DEPART RemQ ; Покинуть общую очередь

DEPART P1 ; Покинуть очередь СС по типу

ADVANCE P2 ; Имитация ремонта

RELEASE Rem1 ; Освободить ремонтное подразделение

TEST E 1,P2,Met2 ; СС первого типа?

TABULATE VrRem1 ; Да, данные о СС первого типа в таблицу

TRANSFER ,Met3 ; Отремонтированные СС первого типа

Met2 TABULATE VrRem2 ; Данные о СС второго типа в таблицу

Met3 TABULATE VrRem ; Данные о СС обоих типов в таблицу

TERMINATE ; Отремонтированные СС первого и второго типов

; Сегмент имитации поступления неисправных СС второго типа

GENERATE 720,480 ; Источник СС второго типа

ASSIGN 1,2 ; Код СС второго типа

ASSIGN 2,(Exponential(371,0,480)); Время ремонта

TRANSFER ,Met1

; Сегмент задания времени моделирования

GENERATE 4320

TERMINATE 1

Транзакты, имитирующие неисправные СС, генерируются блоками GENERATE соответствующих сегментов. В первый параметр транзакта в зависимости от типа СС записывается 1 или 2. Это нужно в последующем для занесения данных раздельно в таблицы по типам СС. Во второй параметр транзактов после вычисления выражения в скобках заносится время ремонта также согласно типу неисправного СС. Поэтому в качестве операнда А блока ADVANCE, имитирующего ремонт, записывается P2. Арифметические выражения не используются. Вместо очередей с именами RemQ1 и RemQ2 используются очереди номер 1 и номер 2. Номера очередям задаются значениями первого параметра транзактов, которые являются кодами типа СС.

Результаты моделирования такие же, как и в примере 2.8. Однако модель получилась более компактной и по своей структуре в большей степени отражает

структуру реальной системы ремонта, в качестве которой выступает ремонтное подразделение.

Пример 2.10. Сервер обрабатывает запросы, поступающие с автоматизированных рабочих мест (АРМ) с интервалами, распределенными по экспоненциальному закону со средним значением 2 мин. Вычислительная сложность запросов распределена по нормальному закону с математическим ожиданием $6 \cdot 10^7$ операций и среднеквадратическим отклонением $2 \cdot 10^5$ операций. Производительность сервера $5 \cdot 10^5$ операций/с.

Построить имитационную модель для определения вероятности обработки запросов за 1 час.

Сервер в модели будет представлять ОКУ, а запросы — транзакты. Исходные данные зададим командой EQU так, чтобы в блоковой части модели они были представлены только переменными пользователя. Для построения модели используем следующие идентификаторы.

Переменные пользователя:

- IntZap — среднее значение интервала времени поступления запросов;
- Mat, Sre0tk — математическое ожидание и среднеквадратическое отклонение вычислительной сложности запросов;
- Koef — коэффициент изменения математического ожидания и среднеквадратического отклонения вычислительной сложности запросов;
- ProSer — производительность сервера;
- VrMod — время моделирования.

Арифметические выражения:

- Ver0br — вероятность обработки запросов;
- Vr0br — время обработки запроса;
- Server — сервер, ОКУ.

; Пример 2.10

; Задание исходных данных

```
IntZap      EQU      120
Koef        EQU      1
Mat         EQU      60000000
Sre0tk      EQU      200000
ProSer      EQU      500000
VrMod       EQU      3600
Ver0br      FVARIABLE N$0brZap/N$KolZap
Vr0br       FVARIABLE (Normal(2,Mat#Koef,Sre0tk#Koef))/ProSer
```

; Сегмент имитации обработки запросов

```
GENERATE    (Exponential(123,0,IntZap)); Источники запросов
KolZap      GATE NU   Server,PotZap      ; Сервер свободен?
SEIZE       Server    ; Да, тогда занять сервер
ADVANCE     V$Vr0br    ; Обработка запроса
```

```
                                RELEASE      Server          ; Освободить сервер
ObrZap      TERMINATE          ; Обработанные запросы
PotZap      TERMINATE          ; Потерянные запросы
; Сегмент задания времени моделирования
; и расчета результатов моделирования
                                GENERATE      VrMod
                                SAVEVALUE      VerObr,V$VerObr ; Вероятность обработки запросов
                                TERMINATE      1
```

Если сервер свободен, блок GATE, стоящий сразу после блока GENERATE, пропускает транзакт на сервер. Транзакт занимает сервер на время, которое вычисляется с помощью арифметической переменной VrObr. В арифметической переменной использована наряду с другими переменными пользователя и переменная с именем Koef. Эта переменная введена для удобства изменения, при необходимости, математического ожидания и стандартного отклонения нормального распределения. Особенно целесообразно использование этой переменной пользователя при проведении экспериментов (п. 9.2).

Вероятность обработки запросов рассчитывается как отношение количества обработанных запросов N\$ObrZap к количеству всех поступивших N\$KolZap запросов.

Теперь мы умеем строить модели систем с одноканальными устройствами, функционирующими в режиме занятия в порядке очереди и освобождения после обслуживания без прерываний в течение требуемого времени.

Приступим к отладке и эксплуатации таких моделей в GPSS World.

3. Разработка и эксплуатация моделей в GPSS World

3.1. Создание объекта «Модель»

3.1.1. Запуск GPSS World

Установка GPSS World на компьютер производится достаточно просто. Для этого нужно вначале загрузить с сайта MinutemanSoftware.com самораспаковывающийся файл (если у вас его нет), а затем запустить его. Предположим, что когда-то все это вы сделали.

Запуск GPSS World производится двойным щелчком мыши на его значке, который после установки программы располагается на рабочем столе.

Если прошло значительное время с момента последней загрузки «Заметок» с веб-сайта Minuteman Software, появляется окно (рис. 3.1), напоминающее о необходимости обновления этих заметок.

При нажатии кнопки Download (Загрузить) открывается страница в Интернет, с помощью которой можно загрузить текущие «Заметки». Практически эти заметки можно загрузить в любое время командой меню File ► Internet ► Download Notices (Файл ► Интернет ► Загрузить заметки). В обоих случаях необходимо указать папку, в которую будет помещен загружаемый файл, и папку модулей, в которую были установлены исполняемые файлы GPSS World. Всегда нужно убедиться, что заменяется существующий файл GPSSW.NTC.

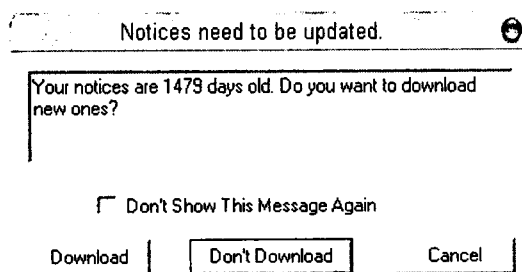


Рис. 3.1. Окно напоминания о необходимости обновления «Заметок»

Окно Notices (Заметки) появляется самостоятельно. Оно (рис. 3.2) содержит текущую информацию об обновлениях и другие полезные сведения.

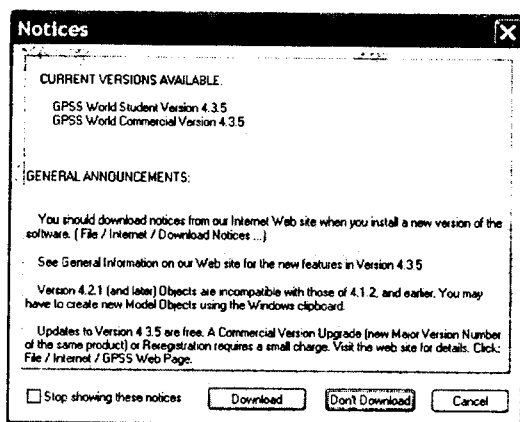


Рис. 3.2. Окно Notices (Заметки)

После закрытия окна Notices появляется главное окно (рис. 3.3) GPSS World.

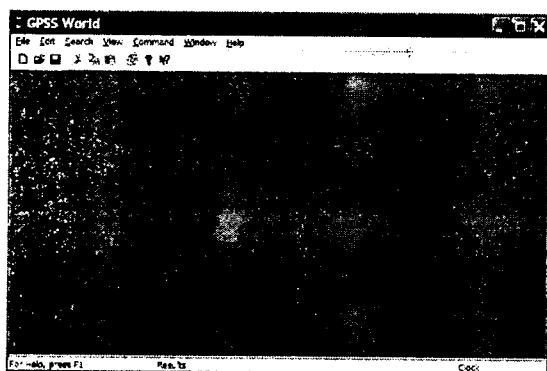


Рис. 3.3. Главное окно GPSS World

Главное окно состоит из нескольких компонентов. В верхней части расположена строка заголовка. Ниже находится основное меню, а еще ниже — панель инструментов, за которой расположена клиентская область.

В самой нижней части главного окна размещена строка состояния, разделенная на три части. Левая часть строки состояния показывает подсказки с информацией об используемых пунктах меню. Средняя часть строки состояния показывает сообщения об ошибках. Правая часть строки состояния имеет небольшую область, используемую для отображения модельного времени в процессе выполнения модели. Включение или выключение показа модельного времени (таймера) для каждого объекта «Процесс моделирования» производится установкой или сбросом флажка View ► Simulation Clock (Вид ► Модельное время).

Все объекты GPSS World имеют меню в их собственных окнах. Однако большинство действий запускается из главного окна с помощью основного меню

(File, Edit, View, Windows, Help (Файл, Правка, Поиск, Вид, Окно, Справка)). Каждый пункт основного меню имеет команды подменю, большинство из которых доступны для использования только в определенных случаях. Это сделано для оказания помощи в выборе правильных действий. В недоступном (отключенном) состоянии команды меню отображаются серым цветом и не могут быть выбраны.

Существуют три варианта работы с моделью:

- выбор с помощью мыши последовательности пунктов меню;
- нажатие клавиши [Alt] для активизации меню с последующим нажатием подчеркнутого символа в названии каждого пункта меню;
- использование клавиш быстрого доступа, указанных справа от пунктов меню.

Нам необходимо создать новый объект «Модель». Операторы модели написаны. Теперь нужно их ввести.

3.1.2. Ввод операторов модели

Для обработки текста GPSS World имеет текстовый редактор. Выберем File ► New (Файл ► Новый). После этого появится меню:

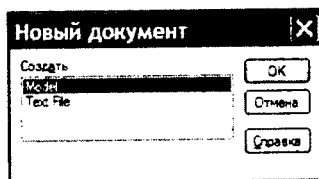


Рис. 3.4. Меню выбора объекта типа «Модель» или текстового объекта

Так как нужно создать объект «Модель», выберем Model. Откроется окно текстового редактора:

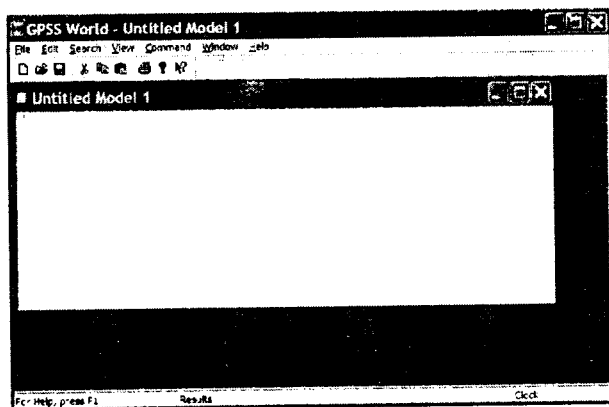


Рис. 3.5. Окно текстового редактора

Модель примера 2.8 начинается с заголовка, выделенного жирным. В текстовом редакторе GPSS World по умолчанию используется шрифт Courier New. Данный шрифт облегчает выравнивание колонок текста модели и особенно стандартного отчета, так как все символы этого шрифта имеют одинаковую ширину. Однако этот шрифт можно изменить. Изменим его, например, для заголовков сегментов модели. Выберем File ► Font (Файл ► Шрифт) и получим окно установки шрифта. Установим Times New Roman, жирный, 10 (рис. 3.6).

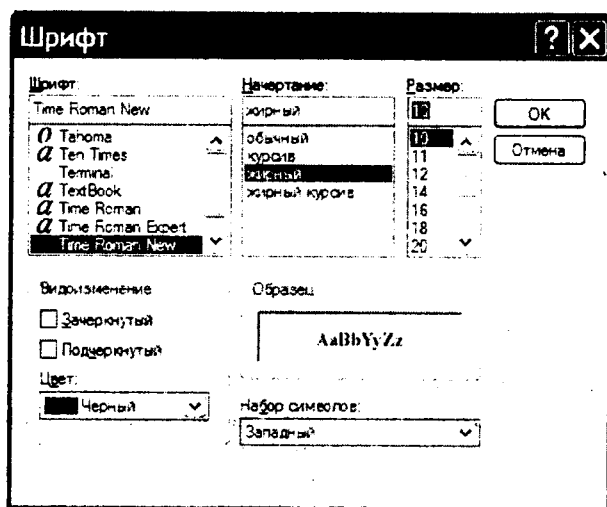


Рис. 3.6. Окно установки шрифта

Для набора второй строки изменим шрифт жирный на курсив жирный. После набора этой строки вернемся к шрифту Courier New и приступим к вводу команд определения таблиц.

Введем имя VrRem1. Для перехода к вводу TABLE следует воспользоваться клавишей [Tab]. Интервалы табуляции установлены по умолчанию. Но они также могут быть изменены в настройках объекта «Модель» на странице Function Keys (поле Model Tabstops).

Оставим интервалы табуляции без изменения и продолжим набор команды, определяющей таблицу с именем VrRem1. Следующие две команды определения таблиц отличаются от первой команды только последними символами в именах. Поэтому можно воспользоваться копированием первой команды и последующим ее редактированием. Выделим строку и поместим ее в буфер обмена.

Существуют три способа выделения текста. При первом способе можно поместить курсор в начале строки и «протаскать» его вдоль страницы, удерживая левую клавишу мыши нажатой. Подсвеченный текст считается выделенным. При втором способе нужно щелкнуть мышью слева от первого символа V имени VrRem1. Затем, нажав и удерживая клавишу [Shift], щелкнуть мышью справа от последнего символа (5) M1,420,180,5. Третий способ состоит в копировании отдельного слова. Для этого нужно два раза щелкнуть мышью на слове.

Для помещения копии выделенного текста выберем Edit ► Copy (Правка ► Копировать). Теперь поместим курсор в начало следующей (пустой) строки и выберем Edit ► Paste (Правка ► Вставить).

Для вырезания, копирования и вставки выделенного фрагмента текста можно также использовать комбинацию клавиш [Ctrl]+[X], [Ctrl]+[C], [Ctrl]+[V].

После редактирования имен таблиц получим (рис. 3.7).

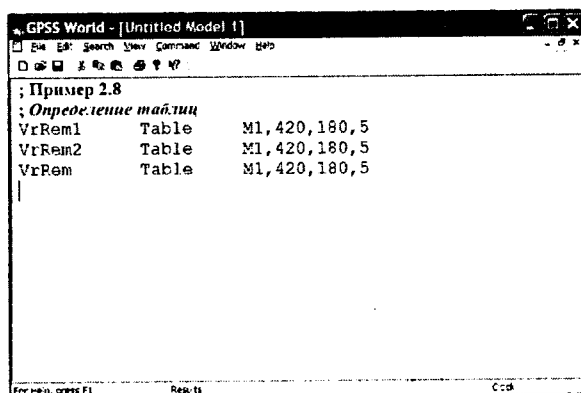


Рис. 3.7. Окно текстового редактора с командами определения таблиц

Можно таким же образом продолжить ввод остальных операторов модели. Однако мы воспользуемся окном Insert GPSS Blocks (Вставить блок GPSS) и окнами создания блоков.

Выберем Edit ► Insert GPSS Blocks... (Правка ► Вставить блок GPSS...). В окне (рис. 3.8) выберем GENERATE и щелкнем левой кнопкой мыши.

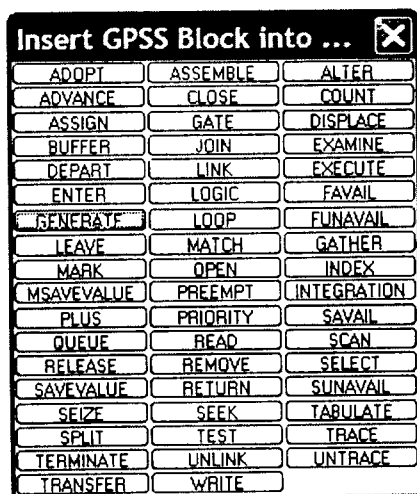


Рис. 3.8. Меню блоков GPSS

Появится окно создания блока GENERATE (рис. 3.9) с курсором в окошке операнда А. Введем 600. Для перехода в окошко операнда В нажмем клавишу [Tab] и затем введем 300. Далее последовательным нажатием клавиши [Tab] либо мышью переведем курсор в окошко Comment. Введем комментарий: Источник СС первого типа (рис. 3.10).

Рис. 3.9. Окно создания блока GENERATE (пустое)

Рис. 3.10. Окно создания блока GENERATE (заполненное)

Теперь нужно нажать OK. В модели появится оператор

GENERATE 600,300; Источник СС первого типа

Продолжим ввод операторов модели и получим (рис. 3.11).

```

GPSS World - [Untitled Model 1]
File Edit Search View Command Window Help
; Пример 2.8
; Определение таблиц
VrRem1 Table M1,420,180,5
VrRem2 Table M1,420,180,5
VrRem Table M1,420,180,5
; Сегмент имитации ремонта СС первого типа
GENERATE 600,300 ; Источник СС первого типа
QUEUE RemQ ; Встать в общую очередь
QUEUE RemQ1 ; Встать в очередь СС первого типа
SEIZE Rem1 ; Занять ремонтное подразделение
DEPART RemQ ; Покинуть общую очередь
DEPART RemQ1 ; Покинуть очередь СС первого типа
ADVANCE 360,120 ; Имитация ремонта
RELEASE Rem1 ; Освободить ремонтное подразделение
TABULATE VrRem1 ; Данные о СС первого типа
TRANSFER ,Met1 ; Отремонтированные СС первого типа
    
```

Рис. 3.11. Фрагмент модели примера 2.8 в окне текстового редактора

Для внесения изменений в текст используются и другие опции меню Edit (Правка). Для отмены неверно сделанного изменения, т. е. возвращения файла к предыдущему состоянию, используется пункт меню Undo (Отмена) ([Ctrl]+[Z]). Команда Delete Line (Удалить строку) ([Ctrl]+[D]) удаляет строку, на которой расположен курсор. Команда Insert Line (Вставить строку) ([Ctrl]+[I]) помещает новую строку ниже строки, на которой расположен курсор.

Рассмотрим команды меню Search (Поиск), которые также могут использоваться для редактирования текста. Команда Find ► Replace (Найти ► Заменить) и диалоговое окно Find Replace (рис. 3.12) подобны аналогичным в других текстовых редакторах.

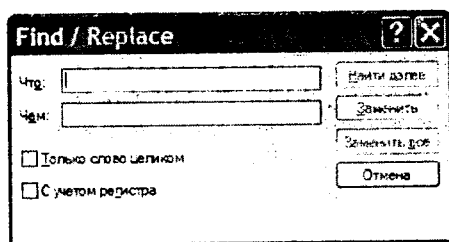


Рис. 3.12. Диалоговое окно Find Replace

ЗАМЕЧАНИЕ

Поиск работает только в одном направлении и останавливается при достижении конца файла. Поэтому перед поиском нужно поместить курсор в начало файла или в ту его часть, выше которой нет того, что ищется.

Для сохранения объекта «Модель» создадим папку МОДЕЛЬ. Далее выберем File ► Save As... (Файл ► Сохранить как...). В появившемся окне (рис. 3.13) вместо Untitled укажем Пример_2.8 и нажмем Сохранить ([Ctrl]+[S]). В последующем для открытия этой модели или какой-либо другой необходимо использовать команду File ► Open (Файл ► Открыть) ([Ctrl]+[O]).

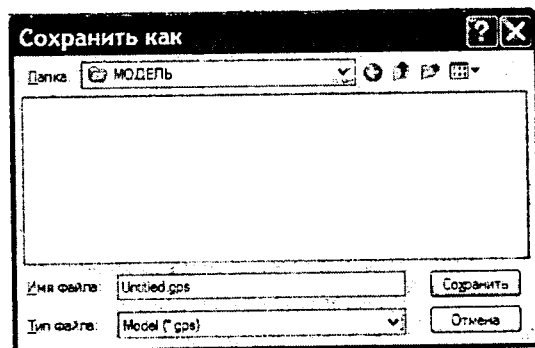


Рис. 3.13. Диалоговое окно Save As... (Сохранить как...)

После этого приступим к созданию объекта «Процесс моделирования».

3.2. Создание объекта «Процесс моделирования»

Объект «Процесс моделирования» представляет собой оттранслированный объект «Модель». Для трансляции объекта «Модель» необходимо выбрать Command ► Create Simulation (Команда ► Создать процесс моделирования) ([Ctrl]+[Alt]+[S]). По этой команде транслятор GPSS проверяет модель на наличие синтаксических ошибок. При отсутствии синтаксических ошибок транслятор активизирует все интерактивные команды и окна для того, чтобы можно было следить за состоянием процесса моделирования и управления им.

Однако чаще всего синтаксические ошибки в объекте «Модель» имеются. Предположим, что есть они и в модели Пример 2.8. Пусть при вводе третьей строки вместо Table набрано Tale и, поскольку следующие две строки (четвертая и пятая) были введены посредством копирования, эта ошибка оказалась и в них. Вследствие ошибок объект «Процесс моделирования» не может быть создан. В окне JOURNAL (Журнал) транслятор выдаст список сообщений об ошибках трансляции (рис. 3.14).

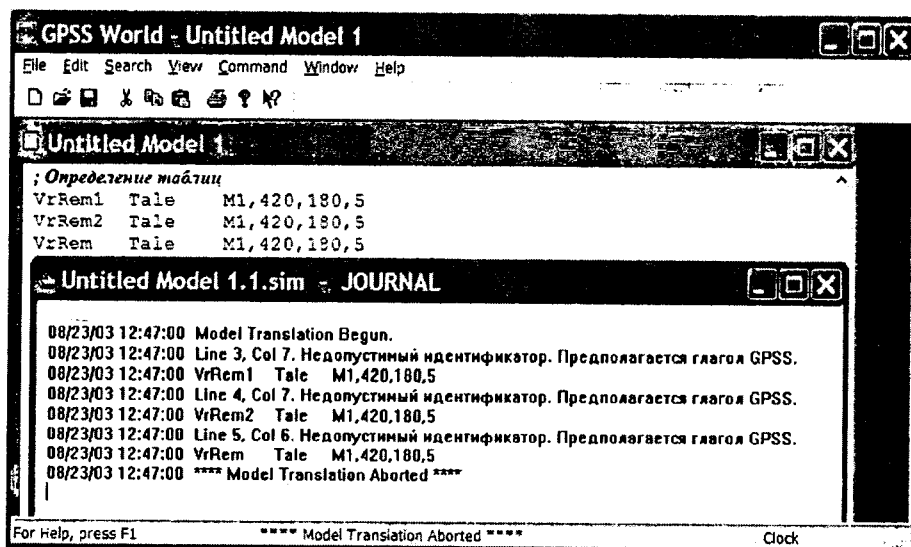


Рис. 3.14. Окно JOURNAL со списком сообщений об ошибках трансляции

Для поиска ошибок и их исправления следует воспользоваться командой Search ► Next Error (Поиск ► Следующая ошибка) ([Ctrl]+[Alt]+[N]). При первом выполнении этой команды курсор мыши помещается в строке текста модели с ошибкой (рис. 3.15).

ЗАМЕЧАНИЕ

В данной книге использовалась русифицированная версия GPSS World, в которой сообщения об ошибках выдаются на русском языке. Русификация выполнена согласно [22].

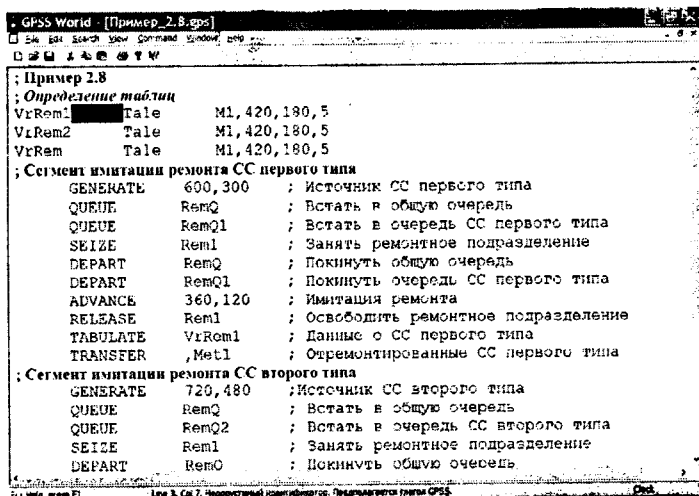


Рис. 3.15. Курсор мыши в строке текста модели с ошибкой

Ошибка исправляется и, поскольку ошибок три, то еще дважды применяется команда Search ► Next Error. Следует иметь в виду, что помимо помещения курсора мыши в строку с ошибкой каждая ошибка при этом также описывается в строке состояния главного окна (см. рис. 3.15). Команда Search ► Previous Error (Поиск ► Предыдущая ошибка) ([Ctrl]+[Alt]+[P]) позволяет просмотреть список ошибок трансляции в тексте модели в обратном направлении.

После исправления ошибок повторно выполняется трансляция. Ее можно также выполнить командой Command ► Retranslate (Команда ► Повторная трансляция) ([Ctrl]+[Alt]+[R]). В случае отсутствия ошибок, обнаруживаемых транслятором, в окне JOURNAL появляется об этом сообщение (рис. 3.16).

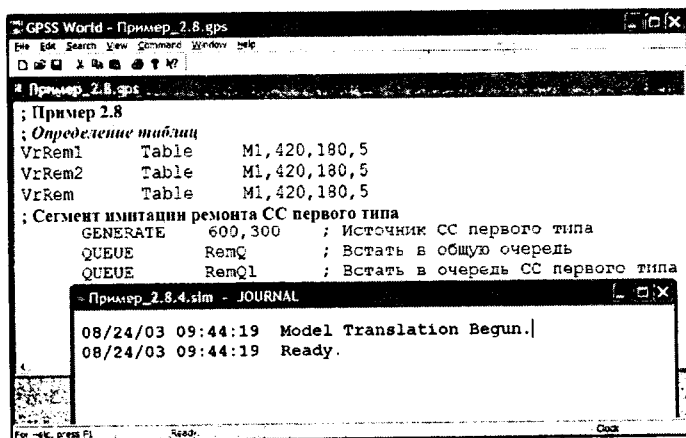


Рис. 3.16. Окно JOURNAL с сообщением об отсутствии ошибок

Теперь можно запустить процесс моделирования. Для этого нужно выбрать Command ▶ Start (Команда ▶ Старт). Появится диалоговое окно (рис. 3.17), в котором имеется возможность заменить 1 на любое количество прогонов, например, на 100 или 100,NP (NP — стандартный отчет не выводится).

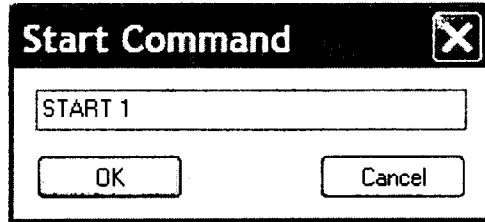


Рис. 3.17. Диалоговое окно команды Start

После нажатия OK произойдет *останов по ошибке выполнения* и в окне JOURNAL будет выдано об этом сообщение (рис. 3.18).

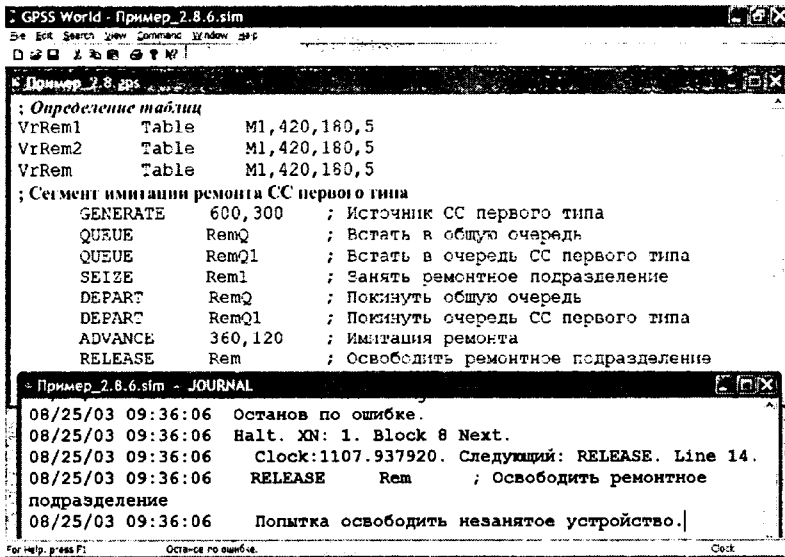


Рис. 3.18. Окно JOURNAL с сообщением об ошибке выполнения

Сообщение указывает, что транзакт пытается освободить ОКУ, которое не было им занято, т. е. в блоке RELEASE в данной модели нужно было указать Rem1, а не Rem. При останове по ошибке транслятор не создает список ошибок выполнения, подобный списку ошибок трансляции. Поэтому использовать команду Search ▶ Next Error нельзя. Строку с ошибкой нужно найти самостоятельно. Для этого следует воспользоваться командой Search ▶ Go To Line... (Поиск ▶ Перейти к строке...) ([Ctrl]+[Alt]+[G]). Номер строки, которую нужно найти, указан в сообщении об ошибке выполнения (см. рис. 3.18, Line 14).

ЗАМЕЧАНИЕ

Иногда бывает, что найденная строка по указанному в сообщении об ошибке номеру не содержит ошибки. Ошибка может быть в других строках текста модели. Для иллюстрации этого предположим, что требуется определить относительную долю поступающих в ремонт СС первого типа. В модель примера 2.8 дополнительно введем две строки:

Dola	VARIABLE	QC\$RemQ1/QC\$Rem
	SAVEVALUE	Dolja,V\$Dolja

Первую строку, определяющую арифметическую переменную с именем Dola, поместим после команд определения таблиц, а вторую строку — после блока GENERATE сегмента задания времени моделирования. После применения команды Command ▶ Start произойдет останов и будет выдано сообщение об ошибке выполнения (рис. 3.19).

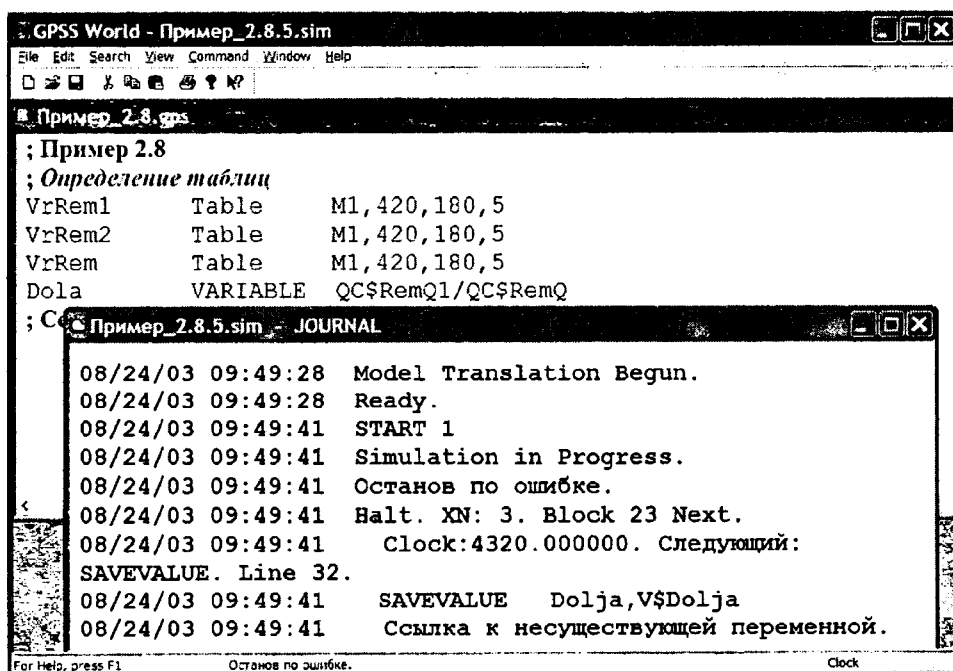


Рис. 3.19. Окно JOURNAL с сообщением об ошибке выполнения

В сообщении указана строка с номером 32. Однако в строке с данным номером ошибки нет. Ошибка находится в строке, определяющей арифметическую переменную: вместо имени Dolja указано имя Dola.

Объект «Процесс моделирования» создан. Прежде чем приступить к дальнейшей отладке, тестированию модели и ее эксплуатации, рассмотрим средства GPSS World, которые используются при этом.

3.3. Команды GPSS World

3.3.1. Интерактивные операторы

Когда успешно завершается первоначальная трансляция модели, создаются:

- последовательность блоков;
- зарегистрированный список процедур;
- список команд.

Все это передается объекту «Процесс моделирования». Последовательность блоков и список команд сохраняются отдельно.

После первоначальной трансляции модели созданному объекту «Процесс моделирования» можно передавать любые операторы модели. Такие операторы называются *интерактивными*. Это могут быть:

- команды;
- определения Plus-процедур;
- блоки.

Последовательность блоков, созданная во время первоначальной трансляции, остается неизменной до следующей повторной трансляции. Интерактивные блоки не включаются в эту последовательность. Они транслируются независимо и передаются для выполнения объекту «Процесс моделирования». Для этого после их поступления создается временный блок, который используется в режиме ручного моделирования и затем уничтожается.

Интерактивные блоки называются *операторами ручного режима моделирования*. Только блоки GENERATE не могут использоваться в этом режиме. Вместо них для создания транзактов могут быть применены блоки SPLIT.

Во время первоначальной трансляции модели оттранслированные процедуры помещаются в регистрационный список процедур. Когда объект «Процесс моделирования» получает интерактивный оператор процедуры, он присоединяет оттранслированную процедуру к библиотеке процедур процесса моделирования, заменяя любую имеющуюся процедуру с тем же именем.

В процессе первоначальной трансляции также создается список команд, образующих очередь. Команды делятся на *срочные* и *несрочные*. К срочным относятся только команды HALT и SHOW. Все остальные команды — несрочные. Срочные команды выполняются сразу же после того, как были переданы объекту «Процесс моделирования». Все остальные команды ставятся в очередь после команд, которые не были еще выполнены.

Существуют несколько способов передать оператор модели процессу моделирования. Можно использовать меню Command (Команда), комбинации «горячих клавиш» и закреплять с помощью журнала установок модели оператор GPSS за функциональной клавишей.

Для отправки объекту «Процесс моделирования» списка команд нужно выбрать Command ► Custom... (Команда ► Ввести...), набрать список команд и нажать кнопку OK. Список команд будет оттранслирован и передан для выполнения.

Можно поместить список наиболее употребляемых команд в текстовый файл. Затем применить команду INCLUDE для передачи списка команд объекту «Процесс моделирования». Или можно закрепить команду INCLUDE за функциональной клавишей и иметь готовый список команд, выполняемый одним нажатием клавиши.

Команды GPSS World в соответствии со своим назначением делятся на две группы:

- команды определения объектов GPSS;
- команды управления процессом моделирования.

3.3.2. Команды определения объектов GPSS

- BVariable — определяет булеву переменную;
- EQU — присваивает значение переменной пользователю;
- FUNCTION — определяет функцию;
- FVariable — определяет переменную;
- INITIAL — инициализирует или изменяет значение логического ключа, ячейки или матрицы;
- MATRIX — определяет матрицу ячеек сохраняемых величин;
- QTable — определяет таблицу статистики очереди (Q-таблицы);
- RMULT — устанавливает параметры первых семи генераторов случайных чисел;
- START — устанавливает значение счетчика завершения и запускает процесс моделирования;
- STORAGE — определяет емкость памяти (МКУ);
- Variable — определяет вещественную переменную;
- Table — определяет таблицу.

Команды BVariable, EQU, FVariable, INITIAL, MATRIX, QTable, Table, Variable рассмотрены в главе 1, START — в главе 2. Команда STORAGE будет рассмотрена в главе 5, а команды FUNCTION и RMULT — в главе 6.

3.3.3. Команды управления процессом моделирования

- CLEAR — сбрасывает накопленную статистику в ноль и удаляет из модели все транзакты;
- CONDUCT — запускает эксперимент;
- CONTINUE — продолжает процесс моделирования;
- EXIT — завершает работу с GPSS World;
- HALT — останавливает процесс моделирования и удаляет все команды из очереди;
- INCLUDE — считывает и транслирует дополнительный файл модели;
- INTEGRATE — автоматически интегрирует по времени переменную пользователя;

- REPORT — устанавливает имя файла отчета или запрашивает немедленный отчет;
- RESET — сбрасывает статистику в ноль без сброса генераторов случайных чисел и таймера модельного времени;
- SHOW — вычисляет и отображает выражение;
- STEP — устанавливает ограниченное количество входов транзактов в блок;
- STOP — устанавливает условие остановки, основанное на количестве входов транзактов в блок.

Рассмотрим команды управления процессом моделирования.

3.3.3.1. Команда CLEAR

При проведении эксперимента необходимо выполнить последовательно несколько прогонов, лишь слегка изменяя версию модели. Было бы удобнее, если бы эти изменения производились за один этап моделирования. В GPSS это достигается за счет использования команды CLEAR. Формат записи команды:

CLEAR [A]

Операнд A может быть ON или OFF. По умолчанию — ON.

Команда CLEAR возвращает процесс моделирования в исходное состояние. Она сбрасывает всю накопленную статистическую информацию, удаляет все транзакты из процесса моделирования и заполняет все блоки GENERATE первым транзактом. ОКУ и МКУ становятся доступными, устанавливаются в незанятое состояние. Содержимое всех блоков становится нулевым. Генераторы случайных чисел не сбрасываются. Пример применения команды CLEAR приведен в п. 7.6.

ЗАМЕЧАНИЕ

Если в команде CLEAR операнд A равен OFF, то ячейки, логические ключи и элементы матриц остаются без изменений.

3.3.3.2. Команда CONDUCT

Команда CONDUCT предназначена для запуска эксперимента (глава 9). Формат команды:

CONDUCT [A]

Операнд A — вызов процедуры Plus-эксперимента. Эксперимент должен быть предварительно зарегистрирован в объекте «Процесс моделирования».

Если в объекте «Процесс моделирования» только один эксперимент без аргументов, то операнд A не используется. Например:

CONDUCT MyExperiment(NumberOfTellers, StartingReplicateNumber)

Plus-эксперимент запускается как и любая другая процедура. Глобальные переменные пользователя NumberOfTellers и StartingReplicateNumber используются, чтобы указать эксперименту, где начать или продолжить прогоны. Аргументы вычисляются и передаются вызванному эксперименту.

Во время эксперимента доступна только команда HALT, т. е. взаимодействие пользователя с процессом моделирования ограничено. Пользователь может только установить отображение абсолютного модельного времени View ► Simulation Clock (Модельное время), а для взаимодействия необходимо приостановить эксперимент.

3.3.3.3. Команда CONTINUE

Команда CONTINUE предназначена для возобновления приостановленного процесса моделирования. Она имеет формат:

CONTINUE

Процесс моделирования приостанавливается при выполнении условия завершения, при поступлении команды HALT, в случае возникновения ошибки или нормального завершения. Условия устанавливаются командой STOP или STEP.

Если процесс моделирования приостанавливается по условию завершения, команда CONTINUE пропускает это условие, но не удаляет его. Если то же самое условие произойдет снова, процесс моделирования снова будет приостановлен. Условия завершения могут быть исключены из модели командой STOP OFF или явно удалены в окне Blocks (Блоки).

Команда CONTINUE может быть использована, когда процесс моделирования был прерван командой HALT. Так как команда HALT удаляет все команды из очереди команд, то будет продолжен только процесс моделирования.

При обработке команды CONTINUE сначала находится счетчик завершения. Если его значение не равно нулю, следовательно, выполнение предыдущей команды START не было завершено. Тогда вызывается планировщик для продолжения обработки транзактов. Если значение счетчика завершения равно нулю, формируется стандартный отчет, а планировщик транзактов не вызывается.

3.3.3.4. Команда EXIT

Команда EXIT предназначена для немедленного завершения сеанса работы с GPSS World. Формат команды:

EXIT [A]

Операнд A — код выхода из GPSS World. Операнд A используется для управления записью объектов «Модель» и «Процесс моделирования» в файлы. Допустимые значения:

- 1 — все объекты сохраняются;
- -1 — все объекты не сохраняются;
- 0 или по умолчанию — каждый измененный файл вызывает появление окна с запросом о его сохранении.

Для закрытия окна приложения служит также библиотечная процедура Exit(). Например, Exit(-1) — все объекты не сохранять.

GPSS World поддерживает фоновый пакетный режим работы, использующийся, когда требуется длительное моделирование без взаимодействия с пользователем. Команда EXIT применяется в пакетном режиме для завершения сеанса работы без оконных операций. Для этого в модель включается команда EXIT 1 или вызов процедуры Exit(1), чтобы закрыть сеанс работы и сохранить все объекты.

3.3.3.5. Команда HALT

Команда HALT предназначена для прерывания процесса моделирования и удаления из очереди всех команд. Команда не имеет операндов:

HALT

Команда HALT является срочной, поэтому она не помещается в очередь команд, а выполняется немедленно. Ее выполнение останавливает процесс моделирования и удаляет все оставшиеся команды из очереди команд. Для продолжения процесса моделирования нужно использовать команду CONTINUE.

3.3.3.6. Команда INCLUDE

Команда INCLUDE предназначена для считывания и трансляции дополнительного файла модели с операторами или списком команд. Формат команды:

INCLUDE A

Операнд A — спецификация файла (полный путь доступа к файлу). Допустимые значения — String. Например:

```
INCLUDE            "DanDon.txt"
INCLUDE            "D:\Primer\DanZad.txt"
```

В первом примере путь доступа к файлу не приводится, так как предполагается, что файл с указанным именем находится в папке модели. Во втором примере указан путь доступа к файлу.

Команда INCLUDE является срочной командой. При трансляции она заменяется файлом. Поэтому располагать в модели команду INCLUDE нужно там, где должны быть операторы или команды, содержащиеся в файле.

Все дополнительно вводимые файлы нумеруются транслятором целыми числами, начиная с 0. Номер 0 присваивается объекту-модели. Нескольким вводам одного файла также присваиваются уникальные номера, т. е. каждый ввод файла приводит к созданию отличающихся наборов блоков.

Команда INCLUDE допускает пять уровней вложенности файлов модели. Нельзя помещать команду INCLUDE в Plus-процедуру.

3.3.3.7. Команда REPORT

Команда REPORT вызывает немедленное создание стандартного отчета. Формат команды:

REPORT [A], [B]

оставлен для совместимости. Поэтому команды

REPORT
REPORT , NOW

идентичны и обе приводят к немедленному созданию стандартного отчета. Когда объект «Процесс моделирования» встречает команду REPORT, немедленно начинается создаваться стандартный отчет в соответствии с параметром In Windows (В окне), который находится на странице Reports (Отчеты) в журнале настроек.

Если параметр In Windows установлен, создается новый объект-отчет, содержащий новый стандартный отчет. Затем он может быть не сохранен или сохранен в файле.

Если параметр In Windows не установлен, создается стандартный отчет, которому присваивается следующий порядковый номер и он помещается в файл.

ЗАМЕЧАНИЕ

Обычно нет необходимости использовать команду REPORT, так как стандартный отчет создается автоматически, посылается окнам Report (Отчет) и потом может быть сохранен в файле. Установленный параметр In Windows напрямую посылает стандартный отчет в файл, имя которого должен указать пользователь.

3.3.3.8. Команда RESET

Начальные условия модели могут заметно отличаться от тех условий, которые имеют место при достижении системой устойчивого состояния. Не все системы нужно изучать в установившемся режиме, который характеризуется некоторыми средними, почти постоянными характеристиками, и на которые оказывают влияние многочисленные мелкие факторы. Если моделируемая система работает в циклическом режиме или является некоторым промежуточным звеном, то более важным может оказаться изучение функционирования системы при увеличении или уменьшении нагрузки. Такими системами будут являться большинство моделируемых систем военного назначения.

Примерами систем, работу которых нужно оценивать в установившемся режиме, являются системы, которые более или менее равномерно функционируют в течение всего моделируемого периода времени. Для получения достоверной статистики существуют три подхода.

1. Модель разрабатывается так, что условия функционирования системы типичны с самого начала.
2. Система моделируется так долго, что любые нетипичные статистические данные, собранные в самом начале процесса моделирования, «утонут» среди типичных данных, собранных в последующее время.
3. Можно поступать следующим образом: моделировать до возникновения типичных условий работы системы; отбросить собранную до этого времени статистику без изменения состояния модели; продолжить моделирование, собирая статистику, на которую уже не влияют нетипичные ситуации.

Первый подход требует от разработчика знания типичных условий работы и умения внести в модель эти условия. В моделях сложных систем это требование вряд ли выполнимо.

При втором подходе требуется слишком долгое моделирование до наступления такого состояния, когда исчезает влияние собранных неверных данных. Стоимость такого моделирования для сложных систем может оказаться слишком высокой, что делает этот подход нежелательным.

Третий подход оказывается наиболее удобным. Нужно на определенный стадии моделирования отбросить статистику с последующим продолжением моделирования без каких-либо модификаций модели.

Сброс в ноль статистики без удаления транзактов из процесса моделирования осуществляет команда RESET. Команда RESET не имеет операндов:

RESET

В отличие от команды CLEAR, команда RESET не удаляет транзакты из процесса моделирования. Она не сбрасывает генераторы случайных чисел, не влияет на абсолютное модельное время и нумерацию транзактов. Относительное модельное время (время после последней команды RESET) устанавливается равным нулю. Таким образом, с помощью оператора RESET обеспечивается удобное средство сбора статистических данных в типичных условиях функционирования систем.

Подразумевается, что если моделирование выполняется долго, то система приходит в стационарное состояние. Сколько времени следует вести моделирование, чтобы достичь стационарного состояния? Часто ответ на этот вопрос можно получить из опыта экспериментирования с моделью. Оператор RESET, в частности, служит для этого. Пример применения команды RESET приведен в п. 7.6.

3.3.3.9. Команда SHOW

Команда SHOW предназначена для передачи объекту «Процесс моделирования» выражения для вычисления и записи результата в строку состояния окна Model. Формат команды:

SHOW X

X — арифметическое или логическое выражение.

Команда SHOW является срочной командой и выполняется немедленно. Если в выражении имеются СЧА, они вычисляются относительно активного транзакта. При отсутствии активного транзакта, т. е. когда процесс моделирования еще не был запущен, будет выдано сообщение об ошибке. Переменным пользователя, если они применяются в выражении, должны быть предварительно заданы значения с помощью команды EQU. Например:

SHOW	XN1
SHOW	N\$Met1/N\$Met2
SHOW	LOG(AC1)

В первом примере в строке состояния показывается номер активного транзакта. Во втором примере — результат деления количества транзактов, вошедших в блок с меткой Met1, к количеству транзактов, вошедших в блок с меткой Met2. В третьем примере вычисляется и выводится в строку состояния натуральный логарифм абсолютного модельного времени.

3.3.3.10. Команда STEP

Команда STEP предназначена для организации выполнения процессом моделирования заданного количества входов транзактов в блоки. Формат команды:

STEP [A]

Операнд A — количество входов транзактов в блоки. Команда:

STEP 1

используется для пошагового выполнения процесса моделирования, т. е. процесс моделирования приостанавливается после одного входа транзакта в блок. При этом в окне JOURNAL появляется сообщение, содержащее время, номер активного транзакта, текущий блок, в котором находится активный транзакт, и следующий блок для входа активного транзакта.

Команда STEP удобна при тестировании (отладке) процесса моделирования. Пошаговое выполнение процесса моделирования кроме окна JOURNAL динамически отображается в окне Blocks (Блоки), что позволяет проследить за путями движения транзактов и, естественно, за логикой работы модели. Команда STEP может вводиться нажатием комбинации [Ctrl]+[Alt]+[1] или с помощью мыши в окне Blocks.

3.3.3.11. Команда STOP

Команда STOP предназначена для установки или снятия условия прерывания процесса моделирования. Формат команды:

STOP [A],[B],[C]

Операнд A — номер транзакта.

Операнд B — номер блока.

Операнд C — флаг состояния команды, может быть ON или OFF, по умолчанию — ON.

Например:

STOP 34,26

Данная команда вводит условие, которое приведет к остановке процесса моделирования, когда транзакт номер 34 попытается войти в блок номер 26. В строку состояния и все открытые окна JOURNAL передается время, номер активного транзакта, текущий блок, в котором находится активный транзакт, и следующий для его входа блок.

Для продолжения моделирования используется команда CONTINUE. Команда CONTINUE не снимает условие остановки. Если это условие вновь выполнится, процесс моделирования снова остановится. Для снятия всех условий остановки необходимо использовать команду

STOP , , OFF

Приведем еще примеры команд:

STOP 83
STOP 83, , OFF
STOP , Met2
STOP , Met2, OFF

Первая команда останавливает процесс моделирования, когда транзакт номер 83 становится активным. Вторая команда снимает условие, установленное первой командой. Третья команда приводит к остановке процесса моделирования, когда любой транзакт пытается войти в блок с меткой Met2. Четвертая команда снимает это условие.

С помощью команд STOP можно проследить на этапе тестирования и верификации правильность выполнения процесса моделирования. Команда STOP вводится с помощью мыши в окне Blocks.

3.4. Окна GPSS World

Для наблюдения за процессом моделирования и действием на него команд на этапе тестирования и верификации используются десять графических окон. Окна подразделяются по типам объектов:

- блоки (Blocks);
- выражения (Expression);
- устройства (Facilities);
- логические ключи (Logicswitches);
- матрица (Matrix);
- график (Plot);
- очереди (Queues);
- ячейки (Savevalues);
- памяти (Storages);
- таблицы (Table).

Окно Blocks (Блоки) показывает входы транзактов в блоки. Оно позволяет с помощью мыши или клавиатуры устанавливать и удалять контрольные точки и визуально отслеживать передвижение транзактов.

Окно Expression (Выражения) предназначено для наблюдения за изменениями любого количества Plus-выражений.

С помощью окна Plot (График) одновременно можно наблюдать любое количество многоцветных графиков. Эти графики можно сохранить или распечатать. Каждое окно может выводить от одного до восьми графиков переменных, включая интегрируемые переменные. Графики могут масштабироваться в любом направлении.

Окно Table (Таблица) представляет собой динамическую гистограмму, полезную для наблюдения за сбором данных, поиска выбросов и оценки сходимости к порождающему вероятностному распределению.

Окна Blocks (Блоки), Facilities (Устройства), Logicswitches (Логические ключи), Queues (Очереди), Savevalues (Ячейки), Storages (Памяти) имеют подробный обзор и общий обзор. Открываются они всегда с подробным обзором. Переход к общему обзору производится с помощью View ► Entity Details (Вид ► Подробности).

Кроме динамических окон в GPSS World доступны кадры состояния для:

- списка текущих событий (CEC Snapshot);
- списка будущих событий (FEC Snapshot);
- отдельного транзакта (Transaction Snapshot);
- числовых групп (Numeric Groups Snapshot);
- групп транзактов (Transaction Groups Snapshot);
- списков пользователя (Userchains Snapshot).

ЗАМЕЧАНИЕ

Динамические окна обновляются оперативно с целью соответствия изменяющемуся состоянию процесса моделирования. В отличие от динамических окон кадры состояния представляют собой статические отображения изменяющихся объектов, зафиксированные в какой-то момент времени выполняющегося процесса моделирования, т. е. кадры состояния динамически не обновляются.

Теперь перейдем к приемам применения рассмотренных средств GPSS World при отладке моделей.

3.5. Отладка моделей

Откройте ранее сохраненную модель Пример_2.8. Для этого выберите File ► Open (Файл ► Открыть), затем Пример_2.8.gps и нажмите Open (Открыть). Закрепите команды SHOW за функциональными клавишами. Откройте журнал настроек модели. Выберите Edit ► Settings (Правка ► Настройки) (рис. 3.20) и перейдите на страницу Function Keys (Функциональные клавиши) (рис. 3.21). Рядом с клавишами [F8] и [F9] наберите следующие две команды:

SHOW	Q\$RemQ1
SHOW	Q\$RemQ2

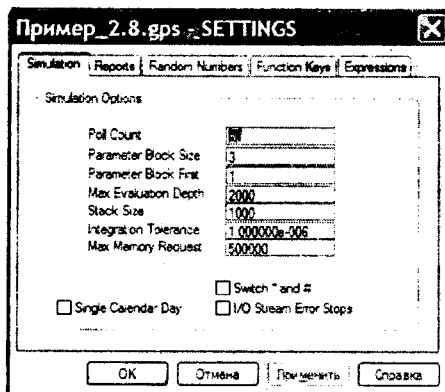


Рис. 3.20. Страница «Simulation» окна журнала настроек модели

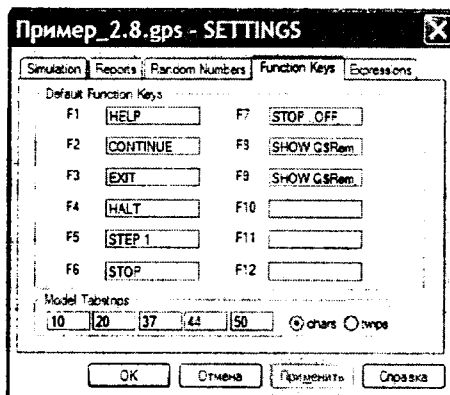


Рис. 3.21. Страница «Function Keys» окна журнала настроек модели

Нажмите ОК. Если допущена ошибка, например, введено только Q\$RemQ1, после нажатия ОК появляется сообщение «Синтаксическая ошибка в операторе. Недопустимый СЧА. Предполагается метка или глагол GPSS». Если ошибок нет, команды закреплены за функциональными клавишами. Во время выполнения процесса моделирования можно интерактивно исследовать текущие длины очередей RemQ1 и RemQ2 нажатием одной соответствующей клавиши.

Запустите процесс моделирования. Выберите Command ► Create Simulation (Команда ► Процесс моделирования). Откройте также окно Blocks (Блоки). В главном меню выберите Window ► Simulation ► Blocks Window (Окно ► Окно процесса моделирования ► Окно «Блоки») (рис. 3.22).

ЗАМЕЧАНИЕ

Команда File ► Print (Файл ► Печать) распечатывает окно, которое было активно, когда была выполнена эта команда. Однако, если используется несколько принтеров, нужно прежде выполнить команду File /Print Setup... (Файл / Настройки принтера...). Если вы попытаетесь распечатать сейчас открытое окно, то оно должно вы-

глядеть так, как на рис. 3.22. Распечатывать можно и в процессе моделирования, однако, лучше его приостановить.

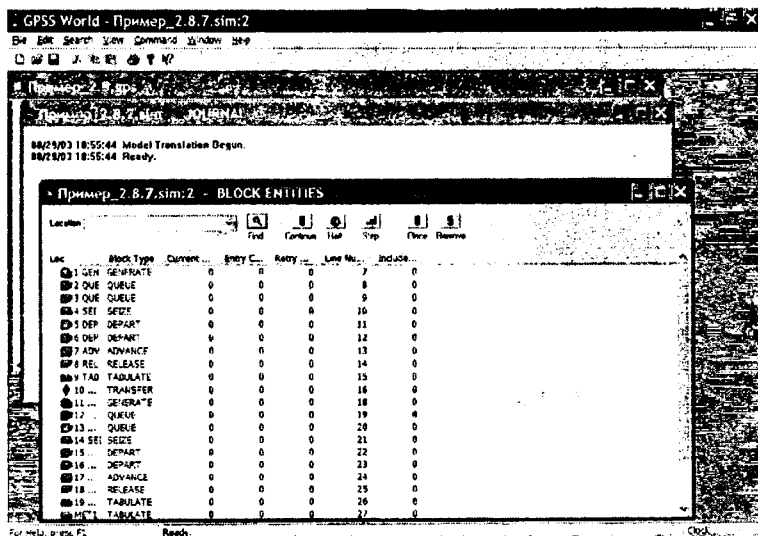


Рис. 3.22. Окна Blocks (Блоки) и Journal (Журнал) модели Пример_2.8

Поставьте условие остановки на активный транзакт номер 20. Для этого выберите Command ► Custom... (Команда ► Ввести...) и в диалоговом окне (рис. 3.23) наберите STOP 20 и нажмите OK.

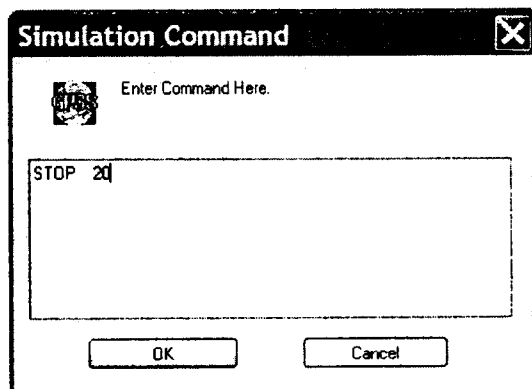


Рис. 3.23. Окно Command ► Custom... (Команда ► Ввести...)

В главном меню выберите Command ► START и в диалоговом окне вместо 1 наберите 100, нажмите OK. Появится сообщение (рис. 3.24) о том, что в процессе моделирования транзакт номер 20 остановился при первой попытке войти в блок. В данном случае это был первый блок GENERATE.

Теперь начните выполнять процесс моделирования в пошаговом режиме, используя функциональную клавишу [F5] с закрепленной за ней в журнале настроек модели командой STEP 1 (см. рис. 3.21). Прежде снимите условие остановки. Выберите Window ► Simulation Snapshot ► User Stops (Окно ► Кадр процесса моделирования ► Условия остановки), щелкните на 20, нажмите Remove (Удалить) и OK.

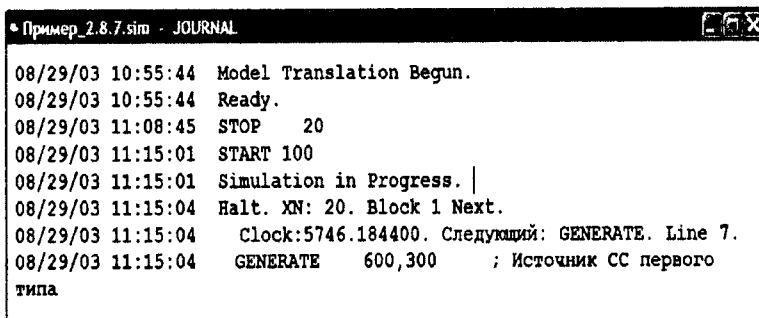


Рис. 3.24. Сообщение об остановке в окне Journal (Журнал)

Вызовите команду STEP 1, нажав функциональную клавишу [F5]. Можно было также выполнить эту команду нажатием комбинации клавиш [Ctrl]+[Alt]+[S] или выбрав Command ► STEP (Команда ► STEP) в главном меню. Транзакт номер 20 войдет в блок GENERATE. Это можно увидеть в трассировочном сообщении окна Journal (Журнал).

Исследуйте некоторые показатели процесса моделирования. Выберите Command ► SHOW... (Команда ► SHOW...) и в диалоговом окне (рис. 3.25) наберите Q\$RemQ, нажмите OK.

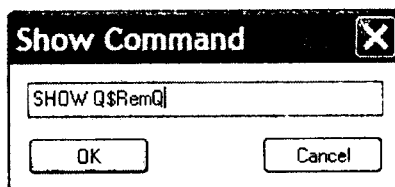
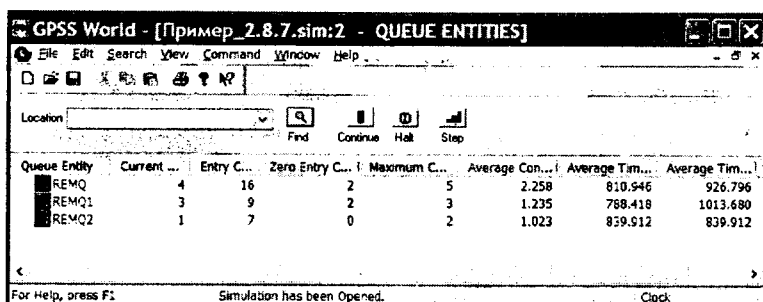


Рис. 3.25. Диалоговое окно Command ► SHOW... (Команда ► SHOW...)

Длина (4) общей очереди RemQ неисправных СС отображается в строке состояния главного окна и в окне Journal (Журнал). Введите команды, которые были закреплены за функциональными клавишами [F8] и [F9]. При нажатии [F8] в строке состояния главного окна и в окне Journal (Журнал) отобразится длина (3) очереди неисправных СС первого типа, а при нажатии [F9] — длина (1) очереди неисправных СС второго типа.

Выберите Window ► Simulation Window ► Queues Window (Окно ► Окно процесса моделирования ► Окно «Очереди»). Откроется окно (рис. 3.26) с подробным

обзором состояния очередей. Для наблюдения за изменением параметров очередей нажмите несколько раз [F5]. Закройте окно.



GPSS World - [Пример_2.8.7.sim:2 - QUEUE ENTITIES]

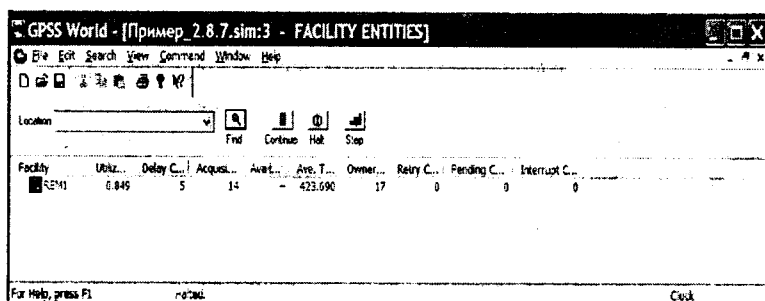
Location: [] Find Continue Halt Stop

Queue Entity	Current ...	Entry C...	Zero Entry C...	Maximum C...	Average Con...	Average Tim...	Average Tim...
REMQ	4	16	2	5	2.258	810.946	926.796
REMQ1	3	9	2	3	1.235	788.418	1013.680
REMQ2	1	7	0	2	1.023	839.912	839.912

For Help, press F1 Simulation has been Opened. Clock

Рис. 3.26. Окно Queues (Очереди). Подробный обзор

Выберите Window ► Simulation Window ► Facilities Window (Окно ► Окно процесса моделирования ► Окно «Устройств»). Откроется окно (рис. 3.27) с подробным обзором устройств. Теперь наблюдайте за изменением состояния ОКУ. Нажмите клавишу [F2] (CONTINUE), а затем [F4] (HALT).



GPSS World - [Пример_2.8.7.sim:3 - FACILITY ENTITIES]

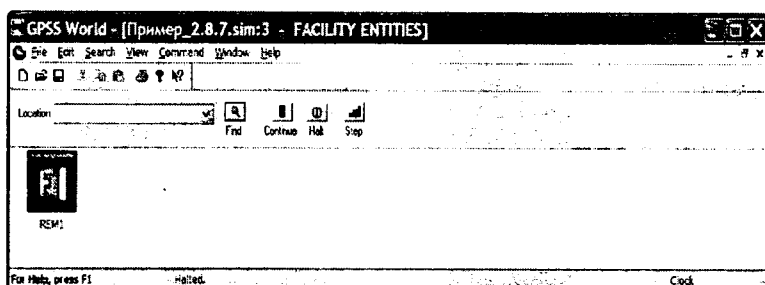
Location: [] Find Continue Halt Stop

Facility	Utilz...	Delay C...	Acquis...	Await...	Avail. T...	Owner...	Retry C...	Pending C...	Interrupt C...
REM1	0.845	5	14	-	423.090	17	0	0	0

For Help, press F1 not running Clock

Рис. 3.27. Окно Facilities (Устройства). Подробный обзор

Для перехода к общему обзору окна Facilities (Устройства) выберите View ► Entity Details (Вид ► Подробности). Откроется окно (рис. 3.28).



GPSS World - [Пример_2.8.7.sim:3 - FACILITY ENTITIES]

Location: [] Find Continue Halt Stop

Facility	Utilz...	Delay C...	Acquis...	Await...	Avail. T...	Owner...	Retry C...	Pending C...	Interrupt C...
REM1	0.845	5	14	-	423.090	17	0	0	0

For Help, press F1 Halted. Clock

Рис. 3.28. Окно Facilities (Устройства). Общий обзор

Нажмите функциональную клавишу [F2] (CONTINUE), а затем функциональную клавишу [F4] (HALT). Обратите внимание на изменение состояния устройства Rem1. Цвет значка устройства меняется с серого, когда оно свободно, на красный, когда оно занято. Маленькая белая полоска в правой части значка показывает небольшую очередь. Большая белая полоска означает, что очередь превышает десять транзактов. Закройте окно Facilities (Устройства).

ЗАМЕЧАНИЕ

Обе формы окна Blocks (Блоки) могут быть полезными. Иногда достаточно знать, где накапливаются транзакты, а иногда нужно точно учесть количество транзактов в различных блоках.

Откройте окно Expression (Выражения). Выберите Window ► Simulation Window ► Expression Window (Окно ► Окно процесса моделирования ► Окно «Выражения»). Появляется окно Edit Expression (Редактор выражения) (рис. 3.29).

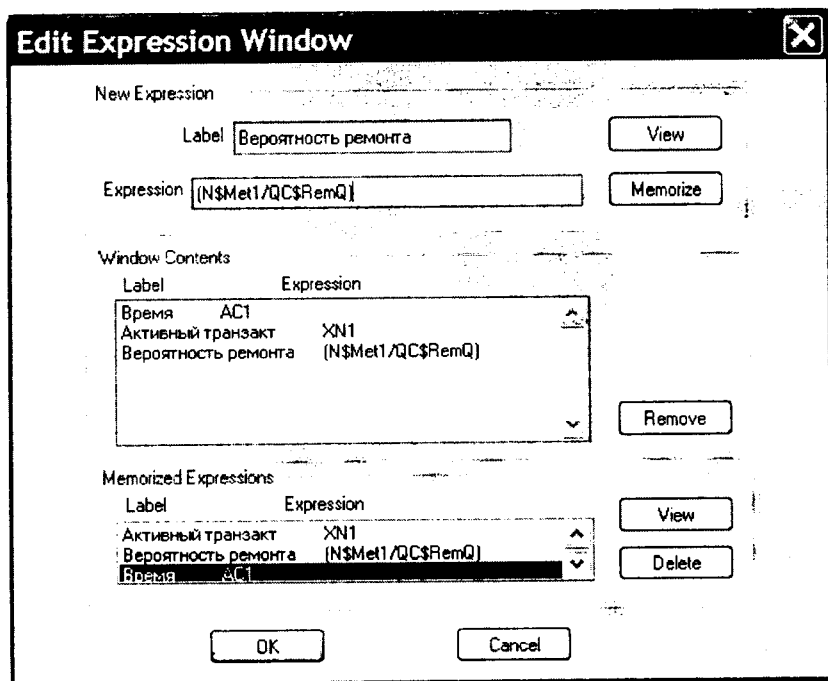


Рис. 3.29. Окно Edit Expression (Редактор выражения) с введенными выражениями и их метками

Редактор выражения запрашивает метку и выражение, которое вы хотите вычислить. В диалоговом окне в поле Label (Метка) введите:

Время

В поле Expression (Выражение) введите:

AC1

Нажмите кнопку View (Просмотр) и кнопку Memorize (Запомнить). Это нужно для того, чтобы показать значения этого выражения в текущем процессе моделирования и просматривать это выражение после сохранения процесса моделирования и повторного его запуска.

В диалоговое поле Label (Метка) вместо Время введите:

Активный транзакт

Вместо AC1 введите:

XN1

Снова нажмите кнопки View (Просмотр) и Memorize (Запомнить). Эти кнопки нажимаются после каждого ввода метки и выражения.

Введите метку и третье выражение:

Вероятность ремонта
(N\$Met1/QC\$RemQ)

Нажмите кнопки View (Просмотр) и Memorize (Запомнить). Нажмите OK. Вы увидите (рис. 3.30) абсолютное модельное время, номер активного транзакта и вероятность ремонта СС.

Label	Expression	Value
Время	AC1	555945.341
Активный тр...	XN1	1858.000
Вероятность ...	(N\$Met1/QC\$RemQ)	0.770

Рис. 3.30. Окно Expression (Выражения)

Понаблюдайте за изменением значений выражений. Нажмите клавишу [F2] (CONTINUE), а затем клавишу [F4] (HALT).

Теперь закройте окно Expression и снова откройте. В открытом окне Edit Expression (Редактор выражения) (см. рис. 3.28) выражения и их метки сохранятся только в списке Memorized Expression (Запомненные выражения). Можно выбрать выражения, которые необходимо отобразить. Для этого нужно выбрать выражение в списке Memorized Expression (Запомненные выражения) и нажать кнопку View (Просмотр). Выберите Время и нажмите кнопку View (Просмотр), затем — Вероятность ремонта и снова нажмите кнопку View (Просмотр). Оба выражения появятся в поле Window Contents. Нажмите OK. Вы увидите окно Expression (Выражения), которое показывает абсолютное модельное время и вероятность ремонта неисправных СС обоих типов. Закройте окно.

ЗАМЕЧАНИЕ

Для перехода от поля к полю в окне редактора выражений нельзя использовать клавишу [Enter], так как это закроет диалоговое окно и выдаст ошибку о недопустимом выражении, если не введены данные до конца. Для перехода можно воспользоваться мышью или клавишей [Tab].

Завершите процесс моделирования, закройте все окна, кроме окна с объектом «Модель». Снова выберите Command ► Create Simulation (Команда ► Создать процесс моделирования).

Выберите Window ► Simulation Window ► Plot Window (Окно ► Окно процесса моделирования ► Окно «График»). Появляется окно Edit Plot (Редактор графика) (рис. 3.31). Вы увидите, что окно похоже на окно Edit Expression (Редактор выражения). Окно Edit Plot предназначено для задания осей, выражений и других общих данных, включаемых в график.

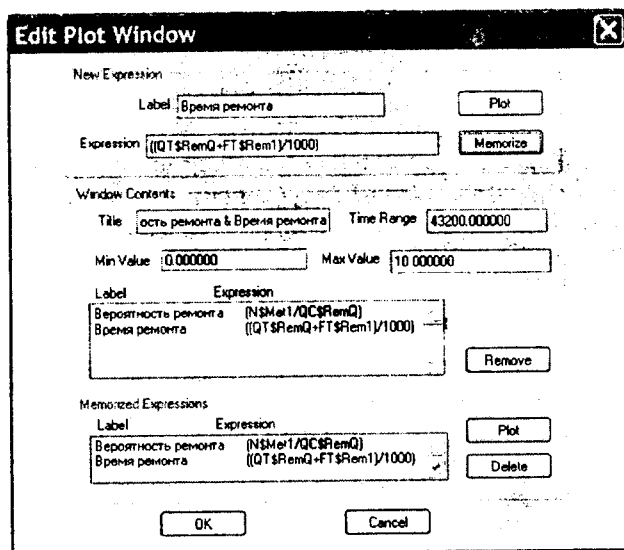


Рис. 3.31. Диалоговое окно Edit Plot (Редактор графика) с введенными данными

Поле Label (Метка) должно содержать метку, которая используется в пояснениях к графику. В это поле введите:

Вероятность ремонта

В поле Expression (Выражение) введите первую переменную для построения графика:

$(N\$Met1/QC\$RemQ)$

Ожидаемая вероятность ремонта определяется как отношение отремонтированных СС к поступившим всего в ремонт. Количество поступивших СС всего в ремонт считается равным общему числу входов в общую очередь.

В поле Title (Заголовок) задайте имя графика, которое описывало бы обе переменные: вероятность выполнения ремонта и время ремонта. Введите:

Вероятность ремонта и время ремонта

Для каждого графика должен быть указан интервал времени. Необходимо сделать его продолжительность достаточно большой, чтобы график не мелькал и при этом его можно было бы разглядеть. Введите временной интервал для оси X. В поле Time Rang (Временной интервал) наберите 43 200 (произведение времени моделирования на количество прогонов).

Для оси Y по умолчанию берутся значения от 0 до 100. Установите 10. Нажмите кнопку Plot (Построить) и кнопку Memorize (Запомнить).

Введите вторую переменную. В поле Label (Метка) введите:

Время ремонта

В поле Expression (Выражение) введите:

$((QT\$RemQ+FT\$Rem1)/1000)$

Среднее время ремонта определяется как сумма среднего времени пребывания в очереди и среднего времени пребывания непосредственно в ремонте. Так как вероятность не может быть больше 1, а время ремонта может превысить 1000, то для времени ремонта введен масштабный множитель 0,001. В этом случае возможно наблюдать оба графика в одном окне без перелистывания.

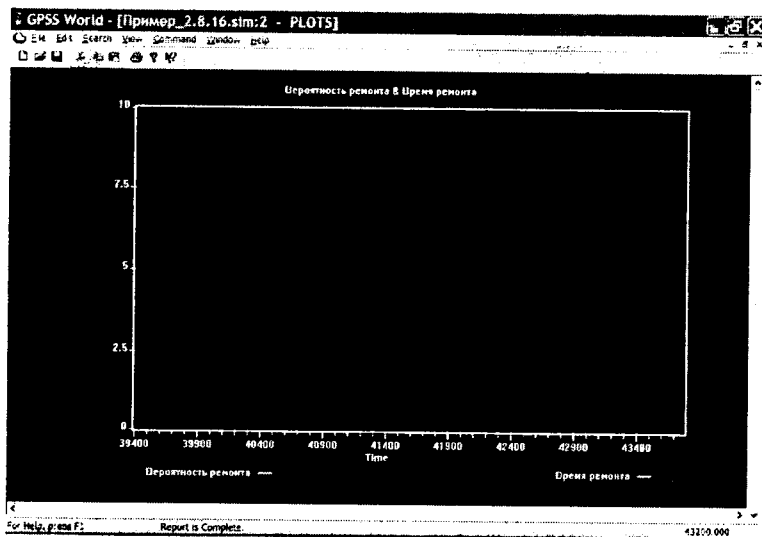


Рис. 3.32. Окно Plot (График)

Нажмите кнопку Plot (Построить) и кнопку Memorize (Запомнить). Нажмите ОК. Окно Plot (График) должно выглядеть так, как показано на рис. 3.32 (если вы

открываете окно не перед запуском процесса моделирования, а в ходе его, то вид рис. 3.32 может быть и другим). Теперь запустите процесс моделирования. Количество прогонов укажите 10. По завершению указанного числа прогонов окно Plot (График) должно выглядеть так, как показано на рис. 3.33.

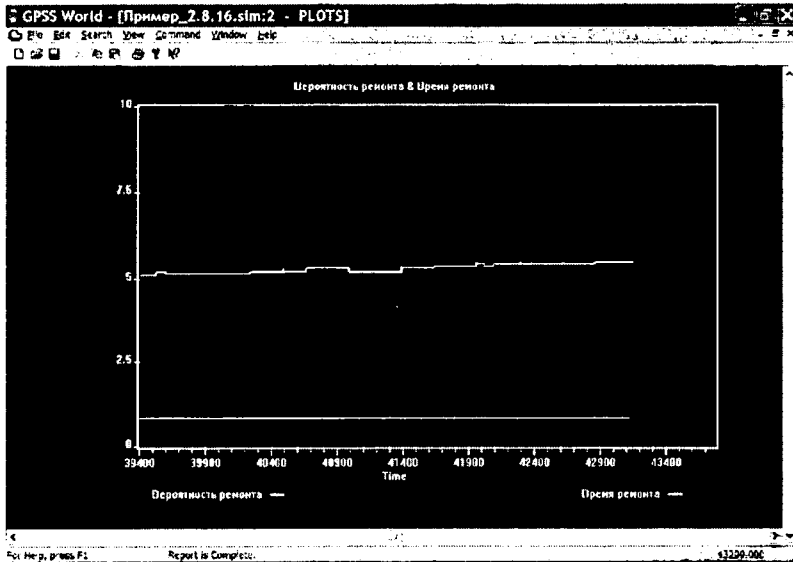


Рис. 3.33. Окно Plot (График) с двумя построенными графиками

В одном окне можно строить до восьми графиков. Для этого нужно ввести подобным образом соответствующее число меток и выражений.

Закройте окно Plot (График). Выберите File ► Save (Файл ► Сохранить). Если раньше файл не сохраняли, по умолчанию будет задано имя Пример_2.8.1.sim. Нажмите кнопку Save (Сохранить). Закройте все открытые окна.

Продолжите сохраненный процесс моделирования. Выберите File ► Open (Файл ► Открыть). В выпадающем списке File of type (Тип файла) в нижней части окна выберите тип файла Simulation (Процесс моделирования). Затем в верхней части диалогового окна выберите Пример_2.8.1.sim и нажмите Open (Открыть).

Открылся сохраненный процесс моделирования. Выберите Window ► Simulation Window ► Plot Window (Окно ► Окно процесса моделирования ► Окно «График»). Появляется окно Edit Plot (Редактор графика). В списке Memorized (Сохраненные выражения) выделите первую метку и нажмите Plot (Построить), затем вторую метку и также нажмите Plot (Построить).

Введите в поле Title (Заголовок) заголовок графика «Вероятность ремонта и время ремонта», в поле Time Range (Временной интервал) — 43 200, в поле Max Value (Максимальное значение) — 10. Теперь все готово для построения графиков.

Процесс моделирования был сохранен после завершения указанных в команде START 10 прогонов, поэтому запустить его снова можно, выбрав Command ► START и указав требуемое число прогонов.

Если же процесс моделирования будет остановлен и сохранен не завершенным, его можно продолжить, нажав функциональную клавишу [F2], с того момента времени, в какой он был остановлен и сохранен. Закройте Plot (График).

Выберите Window ► Simulation Window ► Table Window (Окно ► Окно процесса моделирования ► Окно «Таблица»). В окне Open Table (Выбрать таблицу) (рис. 3.34) выберите VrRem и нажмите OK.

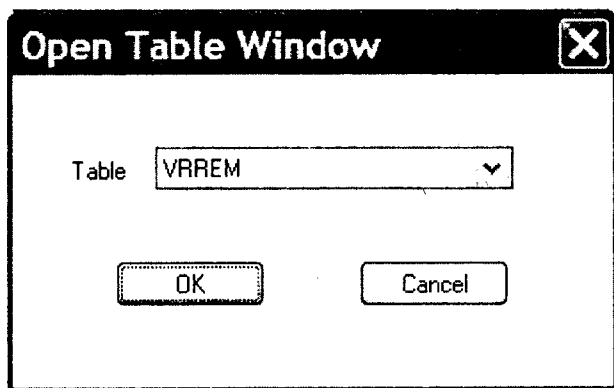


Рис. 3.34. Диалоговое окно Open Table (Выбрать таблицу)

Вы увидите отображение таблицы VrRem, но кроме осей X и Y на диаграмме ничего нет. Запустите процесс моделирования, указав в команде START 10. Наблюдайте за изменением диаграммы. По завершению десяти прогонов в появившейся гистограмме (рис. 3.35) видно, что среднее время ремонта СС обоих типов составляет 5490,823 со стандартным отклонением 2521,028. Второй график (см. рис. 3.33) соответствует этим результатам моделирования.

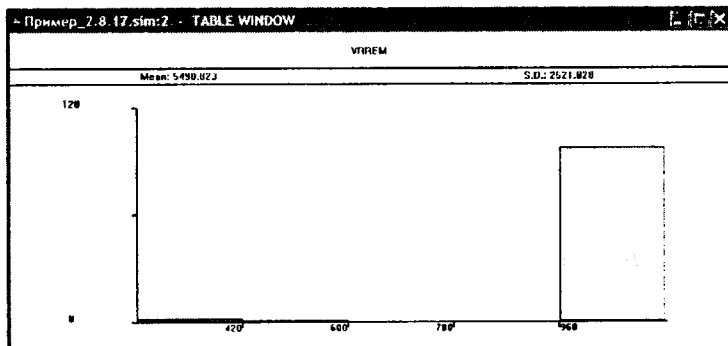


Рис. 3.35. Окно Table (Таблица)

Закройте все окна и снова создайте процесс моделирования. Выберите Window ► Simulation Window ► Blocks Window (Окно ► Окно процесса моделирования ► Окно «Блоки»). В окне Blocks (Блоки) выберите блок TRANSFER (блок 10). Это можно сделать, переместив курсор мыши к блоку 10 и щелкнув левой кнопкой мыши. Щелкните на блоке TRANSFER, затем нажмите кнопку Place (Поместить) на панели инструментов окна Blocks (Блоки). В блок 10 было помещено условие остановки. То же самое можно сделать и через командное меню главного окна с помощью интерактивной команды. Но с помощью панели инструментов отладки это сделать проще.

Запустите процесс моделирования. Выберите Command ► START (Команда ► START) и в диалоговом окне вместо 1 наберите 10, нажмите OK. Процесс моделирования остановится, когда следующий транзакт попытается войти в блок TRANSFER.

Удалите условие остановки из блока TRANSFER. Выберите Window ► Simulation Snapshot ► User Stops (Окно ► Кадр процесса моделирования ► Условия остановки), нажмите кнопку Remove All (Удалить все) (рис. 3.36) и кнопку OK.

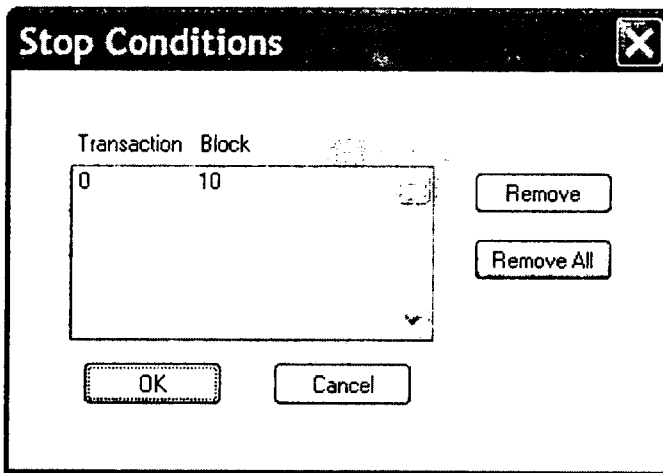


Рис. 3.36. Окно Stop Conditions (Условия остановки)

При отладке и тестировании модели используются также отчеты. Перейдем к их рассмотрению.

3.6. Отчеты

По окончании работы модели система автоматически создает отчет. Статистическая информация выводится только о тех объектах, которые присутствуют в модели. В общем же случае стандартный отчет содержит кроме общей характеристики модели следующие сведения:

- об именах объектов модели;
- блоках модели;
- устройствах;
- многоканальных устройствах;
- очередях;
- списках пользователя;
- статистических таблицах;
- матрицах;
- группах транзактов;
- числовых группах;
- логических переключателях;
- сохраняемых величинах;
- списке текущих событий;
- списке будущих событий.

Необходимость в выводе списков текущих и будущих событий определяет операнд D команды START. Если операнд D не равен нулю, списки выводятся.

Вывод отчета может быть заблокирован. Для этого в поле операнда B команды START необходимо записать NP. Создание стандартных отчетов можно также отключить, сняв флажок Create Standard Report (Создавать стандартные отчеты) на странице Reports (Отчеты) журнала настроек модели.

Каждому отчету присваивается имя. Это имя по умолчанию составляется из имени файла модели, порядкового номера процесса моделирования, созданного из этого файла, и номера отчета данного процесса моделирования. Нумерация отчета начинается с 1. У файлов с отчетами имеется расширение .qpr. Например, Пример_2.8.1.1.qpr, Пример_2.8.1.2.qpr и т. д. При удалении некоторых отчетов их порядковые номера в сеансе работы будут непоследовательными.

Имя отчета, формируемое системой, пользователь изменить не может. Отчеты являются специально отформатированными файлами. Если возникает необходимость сохранить какой-либо отчет с именем пользователя, его нужно скопировать в буфер обмена, а из него — в соответствующий файл. Любой отчет можно распечатать из окна Report (Отчет) с помощью команды File ▶ Print (Файл ▶ Печать) в главном меню.

Отчет может настраиваться. Откройте модель Пример_2.8. Выберите Edit ▶ Settings (Правка ▶ Настройки). Перейдите на страницу Reports (Отчеты) (рис. 3.37). Вы увидите перечень объектов GPSS, данные о которых по умолчанию включаются в отчет. По желанию пользователя перечень объектов можно изменить, сняв или установив соответствующие флажки.

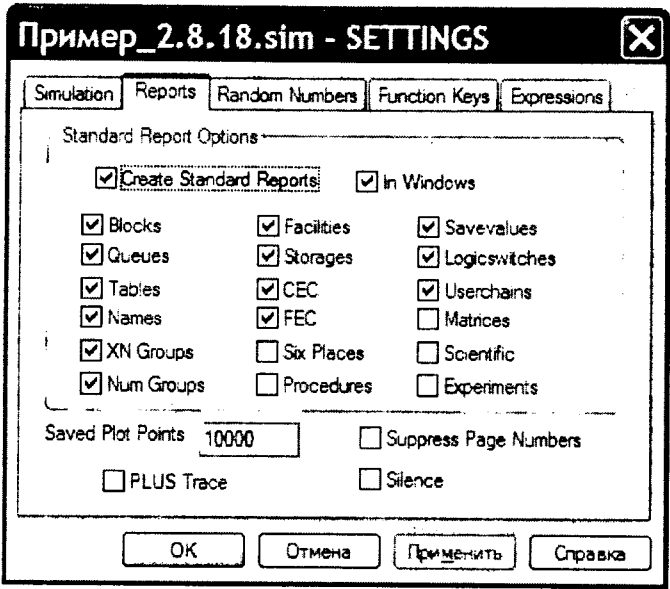


Рис. 3.37. Страница Reports (Отчеты) журнала настроек модели

Запустите процесс моделирования. По его завершению автоматически появляется окно Report (Отчет) (рис. 3.38). Подробное описание элементов отчета приводится в приложении 4. Окно Report (Отчет) предоставляет возможности редактирования текста, изменения шрифта, цвета текста и печати. Закройте окно отчета. Ответьте No (Нет) на вопрос о сохранении файла отчета.

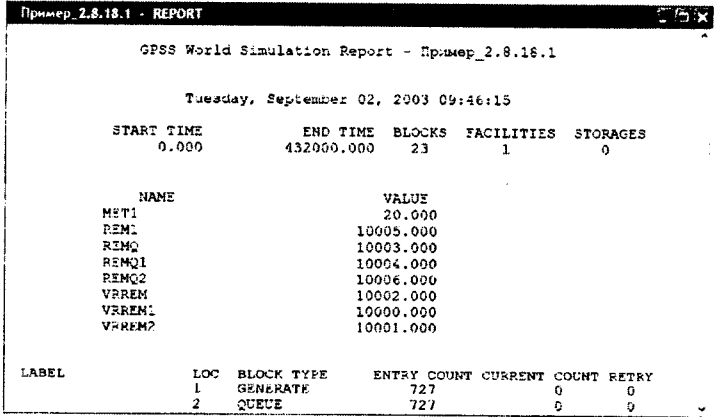


Рис. 3.38. Окно Report (Отчет)

Используйте команду REPORT для получения отчета в момент времени, когда транзакт пытается войти в интересующий вас блок. Щелкните мышью в любом месте окна Journal (Журнал), которое вы не закрыли. Выберите сначала

Command ► CLEAR (Команда ► CLEAR), потом Window ► Simulation Window ► Blocks Window (Окно ► Окно процесса моделирования ► Окно «Блоки»). В окне Blocks (Блоки) выберите блок TERMINATE (блок 21), затем нажмите кнопку Place (Поместить) на панели инструментов окна Blocks (Блоки). В блок 21 было помещено условие остановки. Выберите Command ► START (Команда ► СТАРТ). Вместо 1 наберите 100 и нажмите OK. Процесс моделирования остановится и транзакт будет задержан в блоке 20 с меткой Met1. Выберите Command ► Custom... (Команда ► Выбрать...). Введите команду REPORT и нажмите OK. Сразу же появится окно с отчетом для момента времени, в который было остановлено моделирование.

ЗАМЕЧАНИЕ

Метод применения команды CLEAR изложен в п. 7.6. Команда CLEAR применяется для приведения процесса моделирования в исходное состояние путем сброса статистики, собранной за период неустановившегося режима (количество необходимых для этого прогонов указывается пользователем). При этом генераторы случайных чисел в исходное состояние не возвращаются. После сброса сбор статистики возобновляется, но уже в установившемся режиме. Во всех примерах, кроме указанных, автор посчитал возможным команду CLEAR не применять. Пользователь может, если это ему необходимо, применять команду CLEAR самостоятельно так, как показано в примере 6.2.

На простой модели были рассмотрены приемы работы со средствами, предоставляемыми GPSS World для создания, отладки, тестирования и эксплуатации моделей. Перейдем к рассмотрению методов построения с помощью объектов GPSS других моделей.

4. Построение моделей систем с одноканальными устройствами, функционирующими в режимах прерывания и недоступности

4.1. Прерывание одноканального устройства

Если на входе ОКУ образуется очередь, то выбор транзакта для его занятия после освобождения происходит:

- в порядке поступления (FIFO) — для транзактов с равными приоритетами;
- с учетом приоритета, указанного операндом E блока GENERATE.

При этом очередной транзакт с большим приоритетом ждет окончания обслуживания предыдущего транзакта независимо от его приоритета. Приоритет учитывается только в образующейся очереди. В ней транзакты выстраиваются в приоритетном порядке.

Такой режим функционирования ОКУ организуется блоками SEIZE и RELEASE, рассмотренными в главе 2.

Однако может возникнуть необходимость смоделировать ситуацию, когда очередной транзакт должен занять ОКУ, прервав обслуживание предыдущего транзакта. Такое прерывание называется «захватом» ОКУ и моделируется блоком PREEMPT (захватить). Формат блока:

PREEMPT A, [B], [C], [D], [E]

Операнд A — имя или номер захватываемого ОКУ.

Когда ОКУ свободно, блок PREEMPT работает так же, как и блок SEIZE. При занятии ОКУ блок PREEMPT функционирует либо в *приоритетном режиме*, либо в *режиме прерывания*. Режимы определяются операндом B:

- PR — приоритетный;
- по умолчанию (операнд B не используется) — прерывания.

В приоритетном режиме прервать обслуживание предыдущего (обслуживаемого) транзакта, т. е. «захватить» ОКУ, может только транзакт с большим приоритетом. Если приоритет претендующего на занятие ОКУ транзакта равен или ниже приоритета обслуживаемого транзакта, он помещается в *список задержки* ОКУ последним в своем приоритете.

Что делать с транзактом, обслуживание которого прерывается? Это определяют операнды C, D и E.

Операнд C — имя или номер блока, куда должен быть направлен прерванный транзакт.

Операнд E при значении RE определяет режим удаления прерванного транзакта.

Операнд D — номер параметра прерванного транзакта, в который записывается оставшееся до завершения обслуживания время.

Операнды A, C, D и E могут быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр.

Транзакт, захвативший ОКУ, освобождает его от захвата вхождением в блок RETURN. Формат блока:

RETURN A

Операнд A — имя или номер освобождаемого ОКУ. Например:

RETURN Rem1

Освободить от захвата ОКУ Rem1.

Приведем примеры использования блоков PREEMPT и RETURN.

4.1.1. Прерывание в приоритетном режиме

Пример 4.1. Блок PREEMPT функционирует в приоритетном режиме, так как операнд B — PR. Операнд C не используется, т. е. прерванный транзакт к другому блоку не направляется. Поэтому после прерывания он с автоматически сохраненным временем до конца обслуживания автоматически возвращается в список задержки ОКУ.

; Пример 4.1

	GENERATE	5,,,3	; Источник транзактов
	PRIORITY	XN1	; Присвоить приоритет согласно номеру
	PREEMPT	Rem1,PR	; Занять ОКУ в приоритетном режиме
	ADVANCE	11	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
	TEST E	PR,1,Met2	; Приоритет равен 1?
Met1	TERMINATE	1	; Да, тогда уничтожить
Met2	TEST E	PR,2,Met4	; Приоритет равен 2?
Met3	TERMINATE	1	; Да, тогда уничтожить
Met4	TERMINATE	1	; Уничтожить транзакт с приоритетом 3

Блок GENERATE генерирует только три транзакта в моменты 5, 10 и 15 модельного времени (обозначим *t*) и перестает быть активным. Каждый очередной транзакт, благодаря блокам SAVEVALUE и PRIORITY, имеет более высокий приоритет, чем предыдущий: 1, 2 и 3 соответственно.

Первый транзакт с приоритетом 1 в $t = 5$ занимает ОКУ Rem1. В $t = 10$ генерируется второй транзакт с приоритетом 2. Он прерывает обслуживание первого транзакта, которому осталось до завершения обслуживания $5 + 11 - 10 = 6$ единиц модельного времени. Первый транзакт, ввиду отсутствия операнда C, помещается в список задержки ОКУ Rem1.

В $t = 5$ генерируется третий транзакт с приоритетом 3. Он прерывает обслуживание второго транзакта, так как у него более низкий приоритет, равный 2. До завершения обслуживания второму транзакту осталось $10 + 11 - 15 = 6$ единиц модельного времени. Он также помещается в список задержки ОКУ Rem1 впереди первого транзакта, у которого приоритет ниже и равен 1.

Таким образом, в соответствии с приоритетами будут обслужены третий, второй и первый транзакты. Проследить это можно, выполнив пошаговое моделирование: транзакты последовательно войдут в блоки TERMINATE с метками Met4, Met3 и Met1 соответственно.

При прерывании время «дообслуживания» прерванного транзакта автоматически запоминается и затем при возобновлении обслуживания используется. Это следует из суммарного времени обслуживания трех транзактов в данном примере: $5 + 11 + (5 + 6) + (5 + 6) = 38$ единиц модельного времени. Причем обслуживание третьего транзакта завершится в $t = 15 + 11 = 26$, второго — в $t = 26 + 6 = 32$, первого — в $t = 32 + 6 = 38$ единиц модельного времени.

Из примера 4.1 видно, что в *приоритетном режиме ОКУ может быть прервано любое количество раз*.

ЗАМЕЧАНИЕ

Последние пять блоков (два блока TEST и три блока TERMINATE с метками) включены только лишь для удобства наблюдения за последовательностью обработки транзактов при пошаговом моделировании.

Пример 4.2. Блок PREEMPT функционирует в приоритетном режиме. Используется операнд C, т. е. указана метка блока, к которому направляется прерванный транзакт.

; Пример 4.2

	GENERATE	5,,,3	; Источник транзактов
	PRIORITY	XN1	; Присвоить приоритет согласно номеру
	PREEMPT	Rem1,PR,Met6	; Занять ОКУ в приоритетном режиме
Met1	ADVANCE	11	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
Met2	TRANSFER	,(Met2+XN1)	; Распределить транзакты по приоритету
Met3	TERMINATE	1	; Уничтожить транзакт с приоритетом 1
Met4	TERMINATE	1	; Уничтожить транзакт с приоритетом 2
Met5	TERMINATE	1	; Уничтожить транзакт с приоритетом 3
Met6	ADVANCE	10	; Задержать
	TRANSFER	,Met1	; Отправить на дообслуживание

Блок GENERATE также вырабатывает три транзакта и у каждого следующего транзакта приоритет выше: 1, 2 и 3 соответственно.

Первый транзакт с приоритетом 1 в $t = 5$ занимает ОКУ Rem1.

В $t = 10$ второй транзакт с приоритетом 2 прерывает обслуживание первого транзакта и направляет его к блоку с меткой Met6.

В $t = 15$ третий транзакт с приоритетом 3 прерывает обслуживание второго транзакта и также направляет его к блоку с меткой Met6. Меткой Met6 помечен блок ADVANCE задержки, который предотвращает немедленную передачу прерванного транзакта на только что покинутое ОКУ. После этого блока находится блок TRANSFER, который направляет первый и второй прерванные транзакты на блок ADVANCE ОКУ Rem1.

Обслуживание транзактов завершится:

- третьего в $t = 15 + 11 = 26$;
- второго в $t = 26 + 10 + 11 = 47$;
- первого в $t = 47 + 10 + 11 = 68$.

Видно, что приоритетность обслуживания сохранилась, хотя первый транзакт поступил в модель раньше и раньше был удален с устройства. При этом отсчет времени задержки блоком ADVANCE с меткой Met6 начался только после того, как завершилось обслуживание третьего транзакта. Также прерванные транзакты ОКУ Rem1 обслуживались не 6 единиц модельного времени (столько им оставалось до завершения обслуживания), а с самого начала, т. е. 11 единиц модельного времени.

Если блоком TRANSFER прерванные транзакты направить к блоку PREEMPT, то возникнет ошибка: «Транзакт пытается занять (занять с прерыванием) уже занятое им устройство». Поэтому, казалось бы, удаление прерванных транзактов из модели, т. е. направление их в блок TERMINATE, должно было бы вызывать ошибку, однако этого не происходит. Например, направим прерванные первый и второй транзакты в блок TERMINATE с меткой Met3. Модель благополучно завершит работу через 26 единиц, т. е. после обслуживания третьего транзакта, а первый и второй будут удалены из модели в $t = 5$ и $t = 10$ соответственно.

ЗАМЕЧАНИЕ

Во всем этом можно убедиться, выполнив пошаговое моделирование. Для удобства наблюдения также введены блок TRANSFER с меткой Met2 и три блока TERMINATE с метками. Ими продемонстрирована возможность адресации относительно какого-либо блока. В данном случае относительно блока TRANSFER с меткой Met2. При входе транзакта в этот блок вычисляется выражение в скобках, т. е. номер блока, к которому должен перейти транзакт, становится равным номеру блока TRANSFER с меткой Met2 плюс номер транзакта. Вследствие этого первый, второй и третий транзакты войдут в блоки TERMINATE с метками Met3, Met4 и Met5 соответственно.

Пример 4.3. Блок PREEMPT функционирует в приоритетном режиме и также используется операнд C. Данный пример отличается от примера 4.2 тем, что меткой Met6 помечен не блок ADVANCE, а добавленный перед ним блок RETURN, относящийся также к устройству Rem1. Кроме того, меткой Met1 помечен не блок ADVANCE, имитирующий обслуживание в устройстве Rem1, а блок PREEMPT. Что изменилось от этого?

; Пример 4.3

	GENERATE	5, , , 3	; Источник транзактов
	PRIORITY	XN1	; Присвоить приоритет согласно номеру
Met1	PREEMPT	Rem1, PR, Met2	; Занять ОКУ в приоритетном режиме
	ADVANCE	11	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
	TERMINATE	1	; Обслуженные транзакты
Met2	RETURN	Rem1	; Освободить ОКУ прерванным транзактом
	ADVANCE	10	; Задержать
	TRANSFER	, Met1	; Вернуть на обслуживание сначала

Прерванные первый и второй транзакты, входя в блок RETURN с меткой Met2, исключаются из списка прерываний ОКУ Rem1. Сразу же начинается отсчет времени задержки ADVANCE на 10 единиц модельного времени. После задержки в приоритетном порядке прерванные транзакты направляются на ОКУ Rem1 и вновь занимают его. Обслуживание длится не оставшееся до конца время, а с начала в течение 11 единиц модельного времени.

Обслуживание транзактов завершится:

- третьего в $t = 15 + 11 = 26$;
- второго в $t = 26 + 10 + 11 = 47$;
- первого в $t = 47 + 10 + 11 = 68$.

Время обработки трех транзактов по сравнению с примером 4.2 (68 единиц) уменьшилось на 20 единиц модельного времени. Произошло это вследствие параллельного обслуживания третьего и задержки второго, обслуживания второго и задержки первого транзактов.

Если оставить как в примере 4.2 метку Met1 у блока ADVANCE, т. е. не переносить ее к блоку PREEMPT, возникнет ошибка «Попытка освободить незанятое устройство».

ЗАМЕЧАНИЕ

В примере 4.3 и последующих примерах блоки для удобства наблюдения при пошаговом моделировании исключены. При необходимости они могут быть добавлены пользователем.

Пример 4.4. В этом примере блок PREEMPT использует операнды A, B, C и E. Операндом E указано RE. Второй блок RETURN, относящийся к ОКУ Rem1, удален и меткой Met2 вновь помечен, как в примере 4.2, блок ADVANCE.

; Пример 4.4

	GENERATE	5,,,3	; Источник транзактов
	PRIORITY	X1	; Присвоить приоритет согласно номеру
Met1	PREEMPT	Rem1, PR, Met2,, RE	; Занять ОКУ в приоритетном режиме
	ADVANCE	11	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
	TERMINATE	1	; Обслуженные транзакты
Met2	ADVANCE	10	; Задержать
	TRANSFER	, Met1	; Отправить на обслуживание сначала

Время обработки трех транзактов такое же, как и в примере 4.3:

- третьего в $t = 15 + 11 + 26$;
- второго в $t = 26 + 11 = 37$;
- первого в $t = 37 + 11 = 48$.

То есть если вместо операнда С указать RE, действие аналогично прохождению прерванного транзакта через блок RETURN или RELEASE. В обоих случаях прерванный транзакт теряет право на ОКУ, на котором его обслуживание было прервано. Если, например, меткой Met1 пометить не блок PREEMPT, а блок ADVANCE, возникнет ошибка «Попытка освободить незанятое устройство».

Направление прерванных транзактов в блок TERMINATE, т. е. удаление их из модели, ошибки не вызывает.

В примерах 4.2, 4.3 и 4.4 прерванные первый и второй транзакты, вновь получившие право занять ОКУ Rem1, не завершали обслуживание, а обслуживались с начала. Как организовать «дообслуживание»? Покажем это в примере 4.5.

Пример 4.5. В примере показано, как пользователь может организовать дообслуживание прерванных транзактов. Для того чтобы прерванный транзакт смог в последующем «дообслужиться», не нужно использовать операнд E. Следует операндом С указать номер или метку блока, где может это «дообслуживание» произойти (вход в блок ADVANCE), а для записи оставшегося до конца обслуживания времени использовать операнд D, которым указать имя или номер параметра прерванного транзакта. «Дообслуживание» может быть завершено на том же ОКУ, на котором обслуживание было прервано.

; Пример 4.5

	GENERATE	5,,,3	; Источник транзактов
	ASSIGN	VrObr, 11	; Занести в параметр время обслуживания
	PRIORITY	XN1	; Присвоить приоритет согласно номеру
Met1	PREEMPT	Rem1, PR, Met2, VrObr	; Занять ОКУ в приоритетном режиме
	ADVANCE	P\$VrObr	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
	TERMINATE	1	; Обслуженные транзакты
Met2	ADVANCE	10	; Задержать
	TRANSFER	, Met1	; Отправить на дообслуживание

Поскольку время обслуживания и «дообслуживания» может быть разным, в блоке ADVANCE, имитирующем обслуживание ОКУ Rem1, указан параметр Vr0br. Полное время обслуживания записывается в него блоком ASSIGN непосредственно перед занятием ОКУ.

Время окончания обслуживания транзактов в порядке приоритета:

- третьего в $t = 15 + 11 = 26$;
- второго в $t = 26 + 6 = 32$;
- первого в $t = 32 + 6 = 38$.

Второй и первый транзакт после задержки на 10 единиц модельного времени «дообслуживались» на том же ОКУ Rem1, на котором их обслуживание было прервано.

4.1.2. Прерывание в режиме «захвата»

В режиме «захвата», если ОКУ уже используется, активный транзакт помещается в список *отложенных прерываний* или «захватывает» ОКУ. Прерывание обслуживания сразу, а не помещение в список, происходит тогда, когда список отложенных прерываний пуст и обслуживаемый транзакт сам не является «захватчиком». Транзактам из списка отложенных прерываний представляется право занять ОКУ ранее, чем прерванным транзактам или транзактам из списка задержки ОКУ.

Пример 4.6. Блок PREEMPT функционирует в режиме «захвата», так как не используется операнд В. Остальные операнды тоже не используются.

; Пример 4.6

	GENERATE	5, , , 3	; Источник транзактов
	SAVEVALUE	1, (4-XN1)	; Получить номера в обратном порядке
	PRIORITY	X1	; Присвоить приоритет в обратном порядке
	PREEMPT	Rem1	; Захватить ОКУ
	ADVANCE	11	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
Met1	TRANSFER	, (Met1+XN1)	; Распределение транзактов по приоритету
Met2	TERMINATE	1	; Уничтожить транзакт с приоритетом 1
Met3	TERMINATE	1	; Уничтожить транзакт с приоритетом 2
Met4	TERMINATE	1	; Уничтожить транзакт с приоритетом 3

Блок GENERATE генерирует три транзакта. Приоритеты каждого следующего транзакта ниже приоритета предыдущего: 3, 2 и 1 соответственно. Это обеспечивают два блока SAVEVALUE и блок PRIORITY. Такая приоритетность выбрана для того, чтобы показать, что в режиме «захвата» приоритеты игнорируются.

В $t = 5$ блок GENERATE генерирует первый транзакт с приоритетом 3. ОКУ свободно и его займет первый транзакт.

В $t = 10$ в модель вводится второй транзакт с приоритетом 2. Несмотря на то, что его приоритет ниже, чем обслуживаемого первого транзакта, он «захва-

тывает» ОКУ, а второй транзакт, которому осталось до конца обслуживания $11 - 5 = 6$ единиц модельного времени, помещается в список прерываний.

В $t = 15$ вводится третий транзакт с приоритетом 1, но он не может «захватить» ОКУ, так как там уже обслуживается «захватчик». Третий транзакт помещается в список отложенных прерываний.

В $t = 10 + 11 = 21$ завершается обслуживание второго транзакта. Третий транзакт, находящийся в списке отложенных прерываний, имеет преимущество перед первым транзактом, и он занимает ОКУ, хотя приоритет первого транзакта выше. Обслуживание третьего транзакта завершается в $t = 21 + 11 = 32$.

В $t = 32$ ОКУ занимает первый транзакт. Его «дообслуживание» завершается в $t = 32 + 6 = 38$ единиц модельного времени.

Таким образом, приоритеты игнорированы. Проследить работу модели можно в пошаговом режиме. Транзакты покинут модель, войдя в блоки TERMINATE с метками Met4, Met3 и Met2 в $t = 21$, $t = 32$ и $t = 38$ соответственно.

Пример 4.7. Блок PREEMPT функционирует в режиме «захвата». Используется операнд C, т. е. указана метка блока, к которому направляются прерванные в результате «захвата» транзакты.

; Пример 4.7

	GENERATE	5,,3	; Источник транзактов
	SAVEVALUE	1,(4-XN1)	; Получить номера в обратном порядке
	PRIORITY	X1	; Присвоить приоритет в обратном порядке
	PREEMPT	Rem1,,Met6	; Захватить ОКУ
Met1	ADVANCE	11	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
Met2	TRANSFER	,(Met2+XN1)	; Распределение транзактов по приоритету
Met3	TERMINATE	1	; Уничтожить транзакт с приоритетом 1
Met4	TERMINATE	1	; Уничтожить транзакт с приоритетом 2
Met5	TERMINATE	1	; Уничтожить транзакт с приоритетом 3
Met6	ADVANCE	10	; Задержать
	TRANSFER	,,Met1	; Отправить на обслуживание сначала

В $t = 5$ генерируется первый транзакт с приоритетом 3, который и занимает свободное ОКУ Rem1.

В $t = 10$ второй транзакт с приоритетом 2 «захватывает» ОКУ, прерывая обслуживание первого транзакта с более высоким приоритетом. Первый транзакт направляется к блоку ADVANCE с меткой Met6, не теряя права на занятие в последующем ОКУ Rem1.

В $t = 15$ третий транзакт не может «захватить» ОКУ, уже обслуживающее «захватчика», и помещается в список отложенных прерываний.

В $t = 10 + 11 = 21$ завершается обслуживание второго транзакта.

Третий транзакт был в списке отложенных прерываний. Ему предоставляется право занять освободившееся ОКУ. Обслуживание третьего транзакта завершается в $t = 21 + 11 = 32$.

ОКУ становится свободным и начинается отсчет времени задержки первого транзакта блоком ADVANCE с меткой Met6. После задержки он направляется к блоку ADVANCE с меткой Met1, имитирующему обслуживание в ОКУ Rem1. Завершается обслуживание первого транзакта в $t = 21 + 10 + 11 = 32$. Как видно, обслуживание его также начинается с начала, хотя «дообслужиться» ему оставалось 6 единиц модельного времени.

Если блоком TRANSFER прерванный первый транзакт направить к блоку PREEMPT, а не к блоку ADVANCE, то возникнет ошибка: «Транзакт пытается занять (занять с прерыванием) уже занятое им устройство».

Пример 4.8. Блок PREEMPT функционирует в режиме «захвата». Используются все операнды, кроме операнда B. Операнд E — RE, т. е. прерванные в результате «захвата» транзакты удаляются. «Дообслуживание» этих транзактов происходит на одноканальном устройстве Rem2, определенном блоками SEIZE (метка Met6), ADVANCE и RELEASE.

; Пример 4.8

	GENERATE	5,,3	; Источник транзактов
	ASSIGN	Vr0br,11	; Записать время обслуживания в параметр
	SAVEVALUE	1,(4-XN1)	; Получить номера в обратном порядке
	PRIORITY	X1	; Присвоить приоритет в обратном порядке
	PREEMPT	Rem1,,Met2,Vr0br,RE	; Захватить ОКУ
	ADVANCE	P\$Vr0br	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
Met1	TERMINATE	1	; Обслуженные транзакты
Met2	SEIZE	Rem2	; Занять ОКУ Rem2
	ADVANCE	P\$Vr0br	; Дообслужить прерванный транзакт
	RELEASE	Rem2	; Освободить Rem2
	TRANSFER	,Met1	; Отправить на уничтожение

В $t = 5$ генерируется первый транзакт с приоритетом 3, который и занимает свободное ОКУ Rem1.

В $t = 10$ второй транзакт с приоритетом 2 «захватывает» ОКУ, прерывая обслуживание первого транзакта с более высоким приоритетом. Первый транзакт, освобождая ОКУ Rem1, направляется на «дообслуживание» (до конца обслуживания осталось $5 + 11 - 10$) к блоку SEIZE с меткой Met2. ОКУ Rem2 свободно и первый транзакт занимает его.

В $t = 15$ третий транзакт «захватывает» ОКУ Rem1 и прерывает обслуживание второго транзакта, которому до конца обслуживания оставалось $10 + 11 - 15 = 6$ единиц модельного времени. Второй транзакт также направляется на «дообслуживание» на ОКУ Rem2. Так как оно занято первым транзактом, второй транзакт помещается в список задержки этого ОКУ.

В $t = 10 + 6 = 16$ завершается «дообслуживание» первого транзакта на ОКУ Rem2 и его занимает второй транзакт, находившийся в списке задержки.

«Дообслуживание» второго транзакта заканчивается в $t = 16 + 6$.

В $t = 15 + 11 = 26$ ОКУ Rem1, будет обслужен третий транзакт.

Таким образом, если используется операнд E и задает режим удаления (RE), то при занятом ОКУ «захват» происходит и в том случае, когда ОКУ уже было «захвачено», в то время как при не использовании операнда E такое недопустимо. Еще раз убедимся в этом.

Давайте внесем небольшие изменения в пример 4.8. Пусть блок GENERATE генерирует 5 транзактов и перестает быть активным. Заменяем строку

SAVEVALUE 1, (6-XN1)

добавим две строки с TERMINATE, а все остальное оставим без изменения.

Приведем результаты моделирования.

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	5	0	0
	2	ASSIGN	5	0	0
	3	SAVEVALUE	5	0	0
	4	PRIORITY	5	0	0
	5	PREEMPT	5	0	0
	6	ADVANCE	5	0	0
	7	RETURN	1	0	0
MET2	8	TRANSFER	5	0	0
MET3	9	TERMINATE	1	0	0
MET4	10	TERMINATE	1	0	0
MET5	11	TERMINATE	1	0	0
MET6	12	TERMINATE	1	0	0
MET7	13	TERMINATE	1	0	0
MET8	14	SEIZE	4	0	0
	15	ADVANCE	4	0	0
	16	RELEASE	4	0	0
	17	TRANSFER	4	0	0

Видно, что обслуживание ОКУ Rem1 первых четырех транзактов было прервано «захватом» и продолжено на ОКУ Rem2. Каждый следующий транзакт прерывал обслуживание предыдущего «захватом», несмотря на то, что приоритет его был ниже и в ОКУ уже находился «захватчик» (кроме первого транзакта). Последний из «захватчиков» — пятый транзакт обслуживался ОКУ Rem1 без прерывания. То, что это именно он, подтверждает вход одного транзакта в блок TERMINATE с меткой Met6. Обслуживание пятого транзакта завершено в $t = 25 + 11$ (на Rem1) = 36.

Время «дообслуживания» других транзактов:

- первого с приоритетом 5 в $t = 10 + 6$ (на Rem2) = 16;
- второго с приоритетом 4 в $t = 16 + 6$ (на Rem2) = 22;
- третьего с приоритетом 3 в $t = 22 + 6$ (на Rem2) = 28;
- четвертого с приоритетом 2 в $t = 28 + 6$ (на Rem2) = 34.

«Дообслуживание» произведено в приоритетном порядке. Однако это не всегда так. В данном случае ОКУ Rem2 функционирует в режиме без прерываний.

Пример 4.9. Пусть теперь ОКУ Rem2 функционирует в режиме «захвата». Используются только один операнд A.

; Пример 4.9

	GENERATE	5,,5	; Источник транзактов
	ASSIGN	VrObr,11	; Записать время обслуживания в параметр
	SAVEVALUE	1,(6-XN1)	; Получить номера в обратном порядке
	PRIORITY	X1	; Присвоить приоритет в обратном порядке
Met1	PREEMPT	Rem1,,Met3,VrObr,RE	; Захватить ОКУ
	ADVANCE	P\$VrObr	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
Met2	TERMINATE	1	; Обслуженные транзакты
Met3	PREEMPT	Rem2	; Захватить ОКУ Rem2
	ADVANCE	P\$VrObr	; Дообслужиться
	RETURN	Rem2	; Освободить ОКУ Rem2
	TRANSFER	,Met2	; Отправить на уничтожение

Как и в примере 4.8, обслуживание ОКУ Rem1 первых четырех транзактов было прервано «захватом» и продолжено на ОКУ Rem2. Рассмотрим как происходило «дообслуживание».

Первый транзакт с приоритетом 5 в $t = 10$ занял ОКУ Rem2. Его обслуживание в $t = 15$ было прервано вторым транзактом, а до конца оставалась $10 + 6 - 15 = 1$ единица модельного времени.

Третий транзакт поступил в $t = 20$. Прервать обслуживание второго транзакта ОКУ Rem2 он не мог, так как второй транзакт был «захватчиком». Третий, а в $t = 25$ и четвертый транзакты были помещены в соответствии с приоритетом в список отложенных прерываний.

Время обслуживания транзактов:

- второго с приоритетом 4 в $t = 10 + 5$ (на Rem1) + 6 (на Rem2) = 21;
- третьего с приоритетом 3 в $t = 15 + 5$ (на Rem1) + 6 (на Rem2) + 1 (ожидание к Rem2) = 27;
- четвертого с приоритетом 2 в $t = 27 + 6$ (на Rem2) = 33;
- первого с приоритетом 5 в $t = 28 + 5$ (на Rem2) + 1 (на Rem2) = 34.

Обслуживание пятого транзакта завершится на ОКУ Rem1 в $t = 25 + 11 = 36$. Как видно транзакт с наивысшим приоритетом обслужится предпоследним в $t = 36$, хотя поступит в модель первым в $t = 5$.

4.1.3. Проверка состояния одноканального устройства, функционирующего в приоритетном режиме

Проверка состояния ОКУ в приоритетном режиме может проводиться блоком GATE, а также с использованием булевой переменной и блока TEST.

Рассмотрим проверку состояния ОКУ блоком GATE.

Условие проверки задается одним из следующих условных операторов X:

- I — ОКУ, заданное операндом A, прервано;
- NI — ОКУ, заданное операндом A, не прервано.

Например:

```
GATE I      Stan
GATE NI     (V$Rasp-3)
GATE I      Print,Udal
```

В первом примере блок GATE пропустит транзакт, когда ОКУ Stan будет прервано. Во втором примере транзакт пройдет к следующему блоку, когда не прервано ОКУ, номер которого определяется как результат вычисления и последующего округления до целого выражения в скобках (V\$Rasp-3). В третьем примере в случае прерывания ОКУ Print транзакт будет направлен к блоку с меткой Udal.

В первом и втором примерах блок GATE работает в режиме отказа во входе в случае невыполнения условия. Здесь также остается справедливым замечание, сделанное в п. 2.5: отсутствие операнда B может привести к увеличению машинного времени моделирования. Однако в некоторых случаях такой режим, наверное, можно использовать. Рассмотрим это на примере.

Пример 4.10. Блок PREEMPT функционирует в приоритетном режиме и используются также операнды A, C и D. Блок GATE проверяет: обслуживает ли прерывание ОКУ Rem1?

; Пример 4.10

```
GENERATE 5,,5      ; Источник транзактов
ASSIGN   VrObr,11  ; Записать время обслуживания в параметр
PRIORITY XN1       ; Присвоить приоритет согласно номеру
GATE I   Rem1,Met1 ; Прервано ОКУ?
PREEMPT  Rem1,PR,Met1,VrObr,RE ; Занять ОКУ по приоритету
ADVANCE  P$VrObr   ; Обслуживание
RETURN   Rem1      ; Освободить ОКУ
Met1     TERMINATE 1      ; Обслуженные транзакты
```

Приведем результаты моделирования.

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	25.000	10	1	0

LABEL	LOC	BLOCK TYPE	ENTRY	COUNT	CURRENT	COUNT	RETRY
	1	GENERATE	5		0		0
	2	ASSIGN	5		0		0
	3	PRIORITY	5		0		0
	4	GATE	5		0		0
	5	PREEMPT	0		0		0
	6	ADVANCE	0		0		0
	7	RETURN	0		0		0
MET1	8	TERMINATE	5		0		0

Блок GENERATE генерирует пять транзактов и перестает быть активным. Каждый следующий транзакт имеет более высокий приоритет, чем предыдущий: 1, 2, 3, 4 и 5 соответственно.

Однако ни один из пяти транзактов не пропущен блоком GATE к блоку PREEMPT. Произошло это потому, что OKU Rem1 свободно и условие, заданное в блоке GATE, не выполнилось.

Если вместо условного оператора I в блоке GATE указать NI, то каждый последующий транзакт прервет обслуживание предыдущего. Первые четыре транзакта будут направлены к блоку TERMINATE. Один лишь пятый транзакт будет обслужен до конца без прерывания, что следует из приведенных ниже результатов моделирования.

START TIME		END TIME	BLOCKS	FACILITIES		STORAGES	
0.000		36.000	10	1		0	
LABEL	LOC	BLOCK TYPE	ENTRY	COUNT	CURRENT	COUNT	RETRY
	1	GENERATE	5			0	0
	2	ASSIGN	5			0	0
	3	PRIORITY	5			0	0
	4	GATE	5			0	0
	5	PREEMPT	5			0	0
	6	ADVANCE	5			0	0
	7	RETURN	1			0	0
MET1	8	TERMINATE	5			0	0

Это происходит потому, что приоритет следующего транзакта выше приоритета предыдущего транзакта. Условие, указанное в блоке GATE, выполняется: ведь прерывание на блоке PREEMPT сгенерировал транзакт с меньшим приоритетом.

Пример 4.11. Блок GENERATE с интервалами пять единиц модельного времени генерирует пять транзактов и перестает быть активным. Первые два транзакта имеют приоритеты 1 и 2 соответственно, а третий, четвертый и пятый — нулевые. Такие приоритеты позволяют получить блоки TEST, PRIORITY и TRANSFER, стоящие перед блоком GATE.

; Пример 4.11

```

GENERATE      5,,,5      ; Источник транзактов
ASSIGN        VrObr,11    ; Записать время обслуживания в параметр
TEST LE       XN1,2,Met1  ; Номер транзакта ≤ 2?
PRIORITY      XN1         ; Присвоить приоритет согласно номеру
TRANSFER      ,Met2       ; Отправить на ОКУ
Met1          GATE I      Rem1,Met4 ; ОКУ Rem1 прервано?
Met2          PREEMPT     Rem1,PR,Met4,VrObr,RE ; Занять ОКУ по приоритету
ADVANCE       P$VrObr     ; Обслуживание на Rem1
RETURN        Rem1        ; Освободить ОКУ Rem1
Met3          TERMINATE   1      ; Обслуженные транзакты
Met4          SEIZE       Rem2    ; Занять ОКУ Rem2
ADVANCE       P$VrObr     ; Дообслужиться
RELEASE       Rem2        ; Освободить ОКУ Rem2
TRANSFER      ,Met3       ; Отправить на уничтожение

```

Первые два транзакта поступают непосредственно на блок PREEMPT, минуя блок GATE с помощью блока

```
TRANSFER      ,Met2
```

а остальные три транзакта — на блок GATE.

Приведем результаты моделирования.

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES	
0.000	49.000	19	2	0	
LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	5	0	0
	2	ASSIGN	5	0	0
	3	TEST	5	0	0
	4	PRIORITY	2	0	0
	5	TRANSFER	2	0	0
MET1	6	GATE	3	0	0
MET2	7	PREEMPT	2	0	0
	8	ADVANCE	2	0	0
	9	RETURN	1	0	0
MET3	10	TERMINATE	5	0	0
MET4	11	SEIZE	4	0	0
	12	ADVANCE	4	0	0
	13	RELEASE	4	0	0
	14	TRANSFER	4	0	0

В $t = 5$ первый транзакт с приоритетом 1 занимает свободное ОКУ Rem1.

В $t = 10$ второй транзакт с приоритетом 2 прерывает обслуживание ОКУ Rem1 первого транзакта за $5 + 11 - 10 = 6$ единиц модельного времени до его за-

вершения. Первый транзакт направляется к блоку SEIZE с меткой Met4 и занимает свободное ОКУ Rem2.

В $t = 15$ третий транзакт с нулевым приоритетом входит в блок GATE с меткой Met1 и далее направляется также на ОКУ Rem2, так как ОКУ Rem1 обслуживает второй транзакт с более высоким приоритетом. ОКУ Rem2 занято, но через 1 единицу модельного времени, т. е. в $t = 16$ закончит «дообслуживание» первого транзакта и приступает к обслуживанию третьего транзакта, которое должно завершиться в $t = 16 + 6 = 22$.

В $t = 20$ четвертый транзакт с нулевым приоритетом входит в блок GATE с меткой Met1 и потому же, что и третий транзакт, направляется на ОКУ Rem2. В $t = 22$ завершается обслуживание третьего транзакта и начинается обслуживание четвертого транзакта. Оно должно завершиться в $t = 22 + 11 = 33$.

В $t = 25$ пятый транзакт с нулевым приоритетом также направляется на ОКУ Rem2.

Таким образом, ОКУ Rem1 будет обслужен до конца без прерывания один (второй) транзакт, а остальные четыре будут обслужены и «дообслужены» ОКУ Rem2:

- первый с приоритетом 1 в $t = 5 + 5 \text{ (Rem1)} + 6 \text{ (Rem2)} = 16$;
- второй с приоритетом 2 в $t = 10 + 11 \text{ (Rem1)} = 21$;
- третий с приоритетом 0 в $t = 15 + 1 \text{ (задержка)} + 11 \text{ (Rem2)} = 27$;
- четвертый с приоритетом 0 в $t = 20 + 7 \text{ (задержка)} + 11 \text{ (Rem2)} = 38$;
- пятый с приоритетом 0 в $t = 25 + 13 \text{ (задержка)} + 11 \text{ (Rem2)} = 49$.

В пошаговом режиме выполнения модели можно проследить, что только второй транзакт будет обслужен ОКУ Rem1.

По времени обработки транзактов видно, что *пятый транзакт поступил в $t = 25$, когда ОКУ Rem1 13 единиц модельного времени было свободно, но пятый транзакт не занял его*. Произошло это по той же причине, что и в примере 4.10: не выполнилось условие, заданное в блоке GATE.

ЗАМЕЧАНИЕ

Из примеров 4.10 и 4.11 следует, что блоком GATE с условным оператором I проверяется только нахождение ОКУ в состоянии прерывания. Для определения же занятости или незанятости ОКУ нужно использовать блок GATE (см. п. 2.5) с условными операторами U или NU соответственно.

Пример 4.12. Блок PREEMPT функционирует в приоритетном режиме и в нем используются все операнды. Блок GATE проверяет ОКУ Rem1 на нахождении в состоянии прерывания.

; Пример 4.12

GENERATE	5, , , 5, 3	; Источник транзактов
ASSIGN	VrObr, 11;	; Записать время обслуживания в параметр
TEST LE	XN1, 2, Met1	; Номер параметра ≤ 2 ?

	SAVEVALUE	1, (3-XN1)	; Получить номера в обратном порядке
	PRIORITY	X1	; Присвоить приоритет в обратном порядке
	TRANSFER	,Met2	; Отправить на ОКУ
Met1	GATE I	Rem1, Met4	; ОКУ Rem1 прервано?
Met2	PREEMPT	Rem1, PR, Met4, VrObr, RE;	; Занять ОКУ по приоритету
	ADVANCE	P\$VrObr	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
Met3	TERMINATE	1	; Обслуженные транзакты
Met4	SEIZE	Rem2	; Занять ОКУ Rem2
	ADVANCE	P\$VrObr;	; Обслуживание
	RELEASE	Rem2	; Освободить ОКУ Rem2
	TRANSFER	,Met3	; Отправить на уничтожение

Блок GENERATE генерирует пять транзактов с приоритетом 3. Приоритеты первого и второго транзактов изменяются на 2 и 1 соответственно, а третьего, четвертого и пятого остаются неизменными. Такую приоритетность позволяют получить блоки TEST, PRIORITY и TRANSFER, стоящие перед блоком GATE. Первые два транзакта, минуя блок GATE, поступают непосредственно на блок PREEMPT, а остальные — на блок GATE.

Приведем результаты моделирования.

START TIME		END TIME	BLOCKS	FACILITIES	STORAGES		
0.000		48.000	20	2	0		
LABEL	LOC	BLOCK TYPE	ENTRY	COUNT	CURRENT	COUNT	RETRY
	1	GENERATE	5		0	0	
	2	ASSIGN	5		0	0	
	3	TEST	5		0	0	
	4	SAVEVALUE	2		0	0	
	5	SAVEVALUE	2		0	0	
	6	PRIORITY	2		0	0	
	7	TRANSFER	2		0	0	
MET1	8	GATE	3		0	0	
MET2	9	PREEMPT	2		0	0	
	10	ADVANCE	2		0	0	
	11	RETURN	2		0	0	
MET3	12	TERMINATE	5		0	0	
MET4	13	SEIZE	3		0	0	
	14	ADVANCE	3		0	0	
	15	RELEASE	3		0	0	
	16	TRANSFER	3		0	0	

В $t = 5$ первый транзакт с приоритетом 2 занимает свободное ОКУ Rem1.

В $t = 10$ второй транзакт с приоритетом 1 помещается в список задержки ОКУ Rem1, так как оно обслуживает первый транзакт с более высоким приоритетом.

В $t = 15$ третий транзакт с приоритетом 3 входит в блок GATE. Так как ОКУ Rem1 обслуживает прерывание, третий транзакт направляется на ОКУ Rem2. Это ОКУ свободно и третий транзакт занимает его.

Четвертый и пятый транзакты в $t = 20$ и $t = 25$ соответственно также будут направлены на ОКУ Rem2 и помещены в список его задержки, так как оно обслуживает третий транзакт и закончит обслуживание в $t = 15 + 11 + 26$.

Обслуживание завершится:

- первого транзакта с приоритетом 2 в $t = 5 + 11$ (Rem1) = 16;
- второго транзакта с приоритетом 1 в $t = 10 + 6$ (задержка) + 11 (Rem1) = 27;
- третьего транзакта с приоритетом 3 в $t = 15 + 11$ (Rem2) = 26;
- четвертого транзакта с приоритетом 3 в $t = 20 + 6$ (задержка) + 11 (Rem2) = 37;
- пятого транзакта с приоритетом 3 в $t = 20 + 17$ (задержка) + 11 (Rem2) = 48.

Если в блоке GATE условный операнд I изменить на NI, время обслуживания пяти транзактов составит 59 единиц. При этом ОКУ Rem1 будет обслужен без прерываний также второй транзакт.

Теперь приведем примеры использования булевой переменной и блока TEST для проверки состояния ОКУ в приоритетном режиме.

Пример 4.13. Данный пример аналогичен примеру 4.11, только в нем введена булева переменная Prov и блок GATE заменен блоком TEST:

Prov	BVARIABLE	FI\$Rem1
Met1	TEST	BV\$Prov, 1, Met7

В булевой переменной Prov используется СЧА FI — ОКУ прервано. СЧА FI равен 1, если ОКУ занято транзактом с более высоким приоритетом, иначе — 0.

; Пример 4.13

Prov	BVARIABLE	FI\$Rem1	; Определение булевой переменной
	GENERATE	5, , , 5	; Источник транзактов
	ASSIGN	Vr0br, 11	; Записать время обслуживания в параметр
	TEST LE	XN1, 2, Met1	; Номер параметра ≤ 2 ?
	PRIORITY	XN1	; Присвоить приоритет согласно номеру
	TRANSFER	, Met2	; Отправить на ОКУ
Met1	TEST E	BV\$Prov, 1, Met4	; ОКУ прервано?
Met2	PREEMPT	Rem1, PR, Met4, Vr0br, RE	; Занять ОКУ по приоритету
	ADVANCE	P\$Vr0br	; Обслуживание
	RETURN	Rem1	; Освободить ОКУ
Met3	TERMINATE	1	; Обслуженные транзакты
Met4	SEIZE	Rem2	; Занять ОКУ Rem2
	ADVANCE	P\$Vr0br	; Дообслужить
	RELEASE	Rem2	; Освободить ОКУ Rem2
	TRANSFER	, Met3	; Отправить на уничтожение

В $t = 5$ первый транзакт с приоритетом 1 занимает свободное ОКУ Rem1.

В $t = 10$ второй транзакт с приоритетом 2 прерывает обслуживание ОКУ Rem1 первого транзакта за $5 + 11 - 10 = 6$ единиц модельного времени до его завершения. Первый транзакт направляется к блоку SEIZE с меткой Met4 и занимает свободное ОКУ Rem2.

В $t = 15$ третий транзакт с нулевым приоритетом входит в блок TEST с меткой Met1. Заданное в блоке TEST условие не выполняется и транзакт направляется на ОКУ Rem2. Также четвертый транзакт в $t = 20$ и пятый транзакт в $t = 25$ будут направлены на ОКУ Rem2.

Обслуживание всех пяти транзактов завершится за то же время, в те же временные промежутки и на тех же ОКУ, что и в примере 4.11:

- первый с приоритетом 1 в $t = 5 + 5 \text{ (Rem1)} + 6 \text{ (Rem2)} = 11$;
- второй с приоритетом 2 в $t = 10 + 11 \text{ (Rem1)} = 21$;
- третий с приоритетом 0 в $t = 15 + 1 \text{ (задержка)} + 11 \text{ (Rem2)} = 27$;
- четвертый с приоритетом 0 в $t = 20 + 7 \text{ (задержка)} + 11 \text{ (Rem2)} = 38$;
- пятый с приоритетом 0 в $t = 25 + 13 \text{ (задержка)} + 11 \text{ (Rem2)} = 49$.

4.2. Недоступность одноканального устройства

Для моделирования неисправностей ОКУ и других ситуаций в GPSS World предусмотрены блоки, реализующие недоступность и доступность ОКУ. При использовании этих блоков *статистика ОКУ не искажается*. Здесь имеется в виду следующее. Для моделирования, например, неисправностей можно использовать и режим прерывания (PREEMPT). Однако при этом транзакты, вызывающие прерывания (имитирующие отказы ОКУ), учитываются в статистике как и транзакты, обслуженные при исправном функционировании ОКУ. А это неправильно, вследствие чего и искажается статистика ОКУ.

4.2.1. Перевод в недоступное состояние и восстановление доступности

Недоступность ОКУ моделируется блоком FUNAVAIL (символ F означает ОКУ, UNAVAIL — недоступный). При использовании этого блока статистика ОКУ не искажается. Формат блока:

FUNAVAIL A, [B], [C], [D], [E], [F], [G], [H]

Блок делает недоступным ОКУ с именем или номером, указываемым операндом A.

Все транзакты, обрабатываемые блоком FUNAVAIL, разделяются на три класса, которые и определяют назначение операндов:

- транзакт, занимающий ОКУ (по SEIZE или PREEMPT) в момент перевода его в недоступное состояние (операнды B, C, D);
- ранее прерванные транзакты, находящиеся в списке прерываний (операнды E, F);

- транзакты, находящиеся в *списке отложенных прерываний* и в *списке задержки ОКУ* (операнды G, H).

Операндом B задаются режимы обработки транзакта, занимающего ОКУ в момент перевода его в недоступное состояние:

- C0 — *режим продолжения*: продолжить обработку занимающего ОКУ транзакта во время недоступности;
- RE — *режим удаления*: удалить и направить занимающий ОКУ транзакт к блоку, метка которого должна быть указана операндом C;
- по умолчанию — *прервать обработку и поместить в список прерываний ОКУ*; после восстановления доступности этот транзакт может занять ОКУ и «дообслужиться».

Операнд C — метка блока, в который будет направлен в режиме удаления транзакт, занимавший ОКУ в момент перевода его в недоступное состояние.

Операнд D — номер или имя параметра транзакта, занимавшего ОКУ в момент перевода его в недоступное состояние; если он будет удален (режим RE), т. е. исключен из СБС, в этот параметр будет записано время, оставшееся удаленному транзакту до конца обслуживания.

Операндом E задаются режимы обработки транзактов, находящихся к моменту перевода ОКУ в недоступное состояние в списке прерываний, т. е. тех транзактов, обслуживание которых на данном ОКУ было ранее прервано:

- C0 — *режим продолжения*: продолжить работу ОКУ во время недоступности — обслуживать транзакты из списка прерываний;
- RE — *режим удаления*: удалить и направить транзакты из списка прерываний к новому блоку, метка которого должна быть указана операндом F;
- по умолчанию — *оставить ранее прерванные транзакты в списке прерываний ОКУ и запретить им занимать его во время недоступности*.

Операнд F указывает метку блока, к которому будут направлены транзакты из списка прерываний ОКУ, вследствие чего они не могут находиться в СБС, поэтому для них нет возможности занесения в их параметры времени, оставшегося до конца обслуживания.

Операнд F может использоваться и тогда, когда отсутствует операнд E (по умолчанию). В этом случае для перемещенных к новому блоку транзактов прерывание обслуживания сохраняется.

Операндом G задаются режимы обработки транзактов, находящихся к моменту перевода ОКУ в недоступное состояние в списке отложенных прерываний, т. е. ожидающих выполнения с прерыванием, и в списке задержки:

- C0 — *режим продолжения*: продолжить работу ОКУ во время недоступности — обслуживать транзакты из списка отложенных прерываний и списка задержки;
- RE — *режим удаления*: удалить и направить транзакты из списка отложенных прерываний и списка задержки к новому блоку, метка которого должна быть указана операндом H;

○ по умолчанию — *оставить транзакты в списке отложенных прерываний и списке задержки ОКУ и запретить им занимать его во время недоступности.*

Операндом Н указывается метка нового блока, к которому в режиме удаления (RE) направляются транзакты из списка отложенных прерываний и списка задержки. Когда операнд G не используется, нельзя использовать и операнд Н. Недоступность ОКУ сохраняется до тех пор, пока транзакт, вызвавший переход в недоступное состояние, не войдет в блок

FAVAIL A

Блок FAVAIL изменяет состояние ОКУ на доступное, т. е. восстанавливает обычный режим вхождения транзактов в ОКУ. Все транзакты, ожидающие доступного состояния ОКУ, указанного операндом А, активизируются и могут попытаться занять его.

ЗАМЕЧАНИЕ

Операнды В...Н относятся только к транзактам указанных ранее трех классов. Другие транзакты, которые пытаются прервать ОКУ, уже находящееся в недоступном состоянии, в эти классы не входят и операнды В...Н не имеют к ним никакого отношения.

ЗАМЕЧАНИЕ

Перевод ОКУ в недоступное состояние и разрешение продолжать обработку транзактов из указанных трех классов дает возможность имитировать не только отказы, но и различные дисциплины обслуживания.

**4.2.2. Проверка состояний недоступности и доступности
одноканального устройства**

Проверка состояния ОКУ в режиме недоступности проводится блоком GATE. Формат блока см. п. 2.5.

Условие проверки задается одним из следующих условных операторов X:

- FNV — ОКУ, заданное операндом А, недоступно;
- FV — ОКУ, заданное операндом А, доступно.

Например:

GATE	FNV	Stan
GATE	FV	(FN\$Rasp-X\$Col)
GATE	FNV	Print,Udal

В первом примере блок GATE пропустит транзакт, когда ОКУ Stan будет недоступно. Во втором примере транзакт пройдет к следующему блоку, когда доступно одноканальное ОКУ, номер которого определяется как результат вычисления и последующего округления до целого выражения в скобках (FN\$Rasp-X\$Col). В третьем примере в случае доступности ОКУ Print, т. е. не вы-

полнения заданного в блоке GATE условия, транзакт будет направлен к блоку с меткой Udal.

В первом и втором примерах блок GATE работает в режиме отказа во входе в случае невыполнения условия. Здесь также остается справедливым замечание, сделанное в п. 2.5: отсутствие операнда В может привести к увеличению машинного времени моделирования.

Пример 4.14. Данный пример представляет сегмент имитации выхода из строя и восстановления работоспособности ОКУ.

; Пример 4.14

```

GENERATE      (Exponential(32,0,287))      ;Источник транзактов
FUNAVAIL      Cpu1,RE,Ter1,,RE,Ter2,RE,Ter3 ; Перевод в недоступность
ADVANCE       Normal(21,25,4))              ; Восстановление доступности
FAVAIL        Cpu1                          ; Перевод в доступное состояние
TERMINATE

```

Блок GENERATE генерирует транзакты через случайные интервалы времени. Транзакты входят в блок FUNAVAIL и переводят ОКУ Cpu1 в недоступное состояние, что соответствует выходу его из строя. Все транзакты, находившиеся к этому моменту в списке прерываний, списке отложенных прерываний, списке задержки ОКУ Cpu1, а также занимавший это ОКУ транзакт (если такой был) удаляются. Блок ADVANCE имитирует процесс восстановления (ремонта), а блок FAVAIL переводит ОКУ в доступное состояние, т. е. фиксирует момент перехода в исправное после ремонта состояние.

Возможен и другой вариант этого же сегмента.

```

GENERATE      ...,1                        ; Источник транзакта
Met1 ADVANCE   (Exponential(32,0,287)) ; Время доступности ОКУ
FUNAVAIL      Cpu1,RE,Ter1,,RE,Ter2,RE,Ter3 ; Перевод в недоступность
ADVANCE       Normal(21,25,4))              ; Восстановление доступности
FAVAIL        Cpu1                          ; Перевод в доступность
TRANSFER      ,Met1                        ; Направить на очередной цикл

```

Блок GENERATE генерирует один транзакт. Далее этот транзакт входит в первый блок ADVANCE, имитирующий время между отказами. Затем перевод в недоступное и доступное состояния производится так же, как и в предыдущем варианте. Блоком TRANSFER транзакт возвращается к блоку ADVANCE с меткой Met1 и имитация выхода из строя и восстановления ОКУ повторяется вновь. В данном варианте процесс моделирования упрощается, поскольку не нужно генерировать новые и, возможно, в большом количестве, транзакты.

Другие примеры применения блоков FUNAVAIL и FAVAIL будут рассмотрены в п. 4.4.

4.3. Сокращение машинного времени и изменение дисциплин обслуживания методом применения списков пользователя

При движении по модели транзакты могут быть заблокированы, например, как отмечалось ранее, при проверке состояния ОКУ блоками GATE и TEST. Если заблокированные транзакты находятся в СТС, то при большом количестве их планировщик расходует много времени на просмотр СТС с целью выбора очередного транзакта для продвижения. Для экономии машинного времени заблокированные транзакты целесообразно помещать в *списки пользователя* и оставлять их там до тех пор, пока не будут выполнены условия, позволяющие дальнейшее продвижение этих транзактов. Кроме того, размещение ожидающих транзактов в списках пользователя позволяет организовать различные дисциплины очередей, отличающиеся от дисциплины «первым пришел — первым обслужен», реализованной в списке текущих событий.

Список пользователя представляет собой некоторый буфер, в который могут временно помещаться транзакты, выведенные из СТС. *В отличие от списков текущих и будущих событий транзакты вводятся в список пользователя и выводятся из него не автоматически, а по решению пользователя в соответствии с логикой модели при помощи специальных блоков.*

Для ввода транзактов в список пользователя служит блок LINK (ввести в список), который может быть использован в двух режимах: условном и безусловном.

4.3.1. Ввод транзактов в список пользователя в безусловном режиме

Рассмотрим *безусловный режим*, в котором блок LINK имеет следующий формат записи:

[имя] LINK A, B

Операндом A задается имя или номер списка пользователя, в который безусловно помещается транзакт, вошедший в блок LINK.

Операнд B определяет, в какое место списка пользователя следует поместить вошедший транзакт. Допустимые значения:

- FIFO — транзакт помещается в конец списка;
- LIFO — транзакт помещается в начало списка;
- PR — транзакты упорядочиваются по убыванию приоритета;
- P — транзакты помещаются позади тех транзактов, значения соответствующего параметра которых меньше (в порядке возрастания значения параметра);
- M1 — транзакты помещаются в порядке возрастания относительного времени пребывания в модели.

В качестве операнда B могут использоваться и другие СЧА, кроме указанных ранее СЧА транзактов: арифметическая переменная, функция, а также вы-

ражение в скобках. В этом случае выполняется вычисление указанного операндом В для активного транзакта и для всех остальных транзактов, уже находящихся в списке пользователя, начиная с начала очереди. После этого производится упорядочивание транзактов в списке пользователя по убыванию вычисленного значения. Например, блок

```
LINK          3, FIFO
```

помещает транзакты в конец списка пользователя с номером 3 в порядке их поступления в блок. Блок

```
LINK          0tst, P$Pol
```

помещает транзакты в список пользователя с именем 0tst, упорядочивая их по возрастанию значения параметра с именем Pol.

Условия, при которых транзакт помещается в список пользователя, в безусловном режиме проверяются средствами, предусмотренными разработчиком модели. Например, направить транзакт в список пользователя в случае занятости ОКУ можно так, как показано ниже.

```
      . . .
      GATE  NU      Rem1, Wait
      SEIZE      Rem1
      . . .
Wait  LINK          0tst, FIFO
      . . .
```

Если ОКУ Rem1 занято, то блок GATE не впускает транзакт в блок SEIZE, а направляет его в блок LINK с именем Wait, и транзакт вводится в конец списка пользователя с именем 0tst.

В том же фрагменте модели список пользователя можно разместить и иначе.

```
      . . .
      GATE  U      Rem1, Met1
      LINK          0tst, FIFO
Met1  SEIZE      Rem1
      . . .
```

Здесь ОКУ Rem1 проверяется на занятость. Если ОКУ занято, транзакт проходит к следующему блоку LINK и помещается в список пользователя с именем 0tst. В случае незанятости ОКУ, транзакт направляется к блоку SEIZE с меткой Met1 и занимает свободное ОКУ.

В только что рассмотренных примерах предполагается, что список пользователя неограничен, т. е. в него может помещаться любое количество транзактов. При моделировании реальных систем список пользователя может использоваться для имитации, например, входного накопителя, емкость которого, как правило, ограничена. Это ограничение можно реализовать следующим образом.

Emk	EQU	10
	...	
	GATE NU	Rem1,Wait
	SEIZE	Rem1
	...	
Wait	TEST L	CH\$Otst,Emk,Term1
	LINK	Otst,LIFO

Если ОКУ Rem1 занято, то блок GATE не впускает транзакт в блок SEIZE, а направляет его в блок TEST с меткой Wait, находящийся перед блоком LINK. Если текущее содержимое списка пользователя с именем Otst меньше заданной емкости Emk, транзакт проходит в список пользователя, в противном случае направляется к блоку с меткой Term1.

Приведем и другой возможный вариант этого же фрагмента модели.

Emk	EQU	10
	...	
	GATE U	Rem1, Met1
	TEST L	CH\$Otst,Emk,Term1
	LINK	Otst,LIFO
Met1	SEIZE	Rem1

Если ОКУ Rem1 занято, блок GATE пропускает транзакт к блоку TEST. Если текущее содержимое списка пользователя с именем Otst меньше заданной емкости Emk, транзакт проходит в список пользователя, в противном случае направляется к блоку с меткой Term1. Если ОКУ незанято, транзакт направляется к блоку SEIZE с меткой Met1 и занимает свободное ОКУ Rem1.

4.3.2. Вывод транзактов из списка пользователя

Для вывода одного или нескольких транзактов из списка пользователя и помещения их обратно в список текущих событий служит блок UNLINK (вывести из списка), имеющий следующий формат:

[имя] UNLINK X A,B,C,[D],[E],[F] \

- Операндом А указывается имя или номер списка пользователя.
- Операнд В — метка блока, в который переходят выведенные из списка пользователя транзакты.
- Операндом С указывается число выводимых транзактов или ключевое слово ALL для вывода всех находящихся в списке транзактов. По умолчанию, т. е. когда не используется операнд С, берется ALL.
- Операнды D и E вместе с условным оператором X определяют способ и условия вывода транзактов из списка пользователя. Значения оператора X те же, что и в блоке TEST. В случае, когда условный оператор X должен использоваться, но не указан, по умолчанию он принимает значение E (равенство). Если операнды D и E не используются, не указывается и условный оператор X. В этом

случае транзакты выводятся с начала списка, а количество выводимых транзактов определяется обязательным операндом С.

Операнд D может быть:

- булевой переменной;
- номером параметра транзакта;
- ключевым словом BACK.

Если операнд D является булевой переменной, операнд E и оператор X не используются. Булева переменная вычисляется относительно транзакта, находящегося в списке пользователя. Если результат не нуль, т. е. условие вывода выполняется, транзакт выводится. Количество выводимых транзактов определяется операндом С. Однако выведено может быть и меньше, чем указано операндом С: по числу ненулевых результатов вычисления булевой переменной. Кроме того, и транзактов в списке пользователя может быть меньше, чем указано операндом С.

Если операндом D указано ключевое слово BACK, также операнд E и условный оператор X не используются, а транзакты выводятся с конца списка в количестве, определяемом обязательным операндом С.

Если операнд D не булева переменная и не ключевое слово BACK, должны быть указаны операнд E и условный оператор X. Операнд D вычисляется относительно транзакта, находящегося в списке пользователя, и используется в качестве номера параметра, значение которого сравнивается с результатом вычисления операнда E.

Если операнд D задает параметр, а операнд E не используется, значение параметра транзакта из списка пользователя сравнивается со значением такого же параметра выводящего транзакта. Если они равны, транзакт выводится из списка пользователя. И в этом случае количество выводимых транзактов определяется операндом С.

Операндом F указывается имя блока, куда переходит транзакт, выходящий из блока UNLINK, если из списка пользователя не выведен ни один транзакт. Если операнд F не используется, выходящий транзакт переходит в следующий блок независимо от количества выведенных транзактов. Например, блок

UNLINK 4, App, 1

выводит из списка пользователя с номером 4 один транзакт с начала списка и направляет его в блок с меткой App. Блок

UNLINK Otst, Mars, 1, BACK

выводит из списка пользователя с именем Otst один транзакт с конца списка и направляет его в блок с меткой Mars. Блок

UNLINK E P\$Wiw, App1, ALL, App2, P\$App2, App3

выводит из списка пользователя, номер которого записан в параметре Wiw выводящего транзакта, и направляет в блок с меткой App1 все транзакты, содер-

жимое параметра App2 которых равно содержимому одноименного параметра выводящего транзакта. Если таких параметров в списке не окажется, то выводящий транзакт будет направлен в блок с меткой App3, в противном случае — к следующему блоку.

ЗАМЕЧАНИЕ

Отметим следующие особенности выполнения блока UNLINK. Во-первых, если операнды D и E содержат ссылки на СЧА транзактов, операнд D вычисляется относительно транзактов в списке пользователя, а операнд E — относительно активного транзакта. Во-вторых, после вывода транзактов из списка планировщик продолжает или начинает продвижение транзакта с наивысшим приоритетом, а при равенстве приоритетов отдает предпочтение транзакту-инициатору вывода.

Рассмотрим более детально на других примерах механизм работы блока LINK в безусловном режиме и блока UNLINK, а следовательно, и формирования различных дисциплин обслуживания.

Пример 4.15. В блоке LINK в качестве операнда B используется СЧА класса P, следовательно, транзакты в списке пользователя будут размещаться в порядке возрастания значения параметра P1.

; Пример 4.15

	GENERATE	5, , , 5	; Источник транзактов
	ASSIGN	1, XN1	; Записать номер транзакта в параметр
	GATE NU	Rem1, 0tst	; Незанято ОКУ?
App1	SEIZE	Rem1	; Занять ОКУ
	ADVANCE	11	; Обслуживание
	RELEASE	Rem1	; Освободить ОКУ
	UNLINK	Pol, App1, 1	; Вывести из списка пользователя
	TERMINATE	1	; Обслуженные транзакты
0tst	LINK	Pol, P1	; Список пользователя

Блок GENERATE генерирует с интервалом 5 единиц модельного времени 5 транзактов. В параметр P1 каждого следующего транзакта записывается большее число: 1, 2, 3, 4 и 5 соответственно.

В $t = 5$ первый транзакт занимает свободное ОКУ Rem1. Его обслуживание закончится в $t = 5 + 11 = 16$.

В $t = 10$ второй транзакт, так как ОКУ Rem1 занято, направляется в блок LINK с меткой 0tst и помещается в пустой список пользователя Pol.

В $t = 15$ третий транзакт, так как ОКУ Rem1 по-прежнему занято, также направляется в список пользователя Pol. В списке находится второй транзакт, значение параметра P1 которого, равное 2, меньше значения параметра P1, равное 3, третьего транзакта. Поэтому третий транзакт размещается за вторым.

В $t = 16$ завершается обработка первого транзакта. Первый транзакт, войдя в блок UNLINK, выводит из списка пользователя второй транзакт и направляет его на освободившееся ОКУ Rem1. Обслуживание второго транзакта закончится в $t = 16 + 11 = 27$. В списке пользователя остался третий транзакт.

В $t = 20$ и $t = 25$ четвертый и пятый транзакт вследствие занятости ОКУ Rem1 также будут размещены после третьего транзакта друг за другом.

В $t = 27$, $t = 38$ и $t = 49$ из списка пользователя будут выведены третий, четвертый и пятый транзакты соответственно, т. е. в моменты освобождения ОКУ Rem1. Обслуживание последнего пятого транзакта закончится в $t = 49 + 11 = 60$.

Проследить последовательность размещения в списке пользователя и обслуживания можно в пошаговом режиме: транзакты согласно своим номерам XN1 войдут в блоки TERMINATE.

Рассмотренный пример позволяет организовать обслуживание согласно номерам транзактов, т. е. приоритет отдается транзакту с наименьшим временем пребывания в модели. Такую же дисциплину обслуживания можно организовать, если в блоке LINK в качестве операнда В указать M1 или выражение в скобках (6-XN1).

Если заменить блок

ASSIGN 1, (6-XN1)

в параметр P1 транзактов будут записаны 5, 4, 3, 2 и 1 соответственно. Вследствие этого при размещении в списке пользователя приоритет будет отдаваться транзакту с наибольшим временем пребывания в модели. Такая же дисциплина обслуживания будет получена, если в блоке LINK вместо операнда В указать XN1 или M1.

Пример 4.16. В одноканальной СМО с ожиданием требуется организовать дисциплину обслуживания, при которой приоритет отдается заявкам с наименьшим временем обслуживания.

Программа такой модели приведена ниже.

; Пример 4.16

	GENERATE	(Exponential(23,0,200))	; Источник транзактов
	ASSIGN	Nb1, (Exponential(34,0,120))	; Записать в параметр
	GATE NU	Evm, Otst	; ОКУ свободно?
App1	SEIZE	Evm	; Занять ОКУ
	ADVANCE	P\$Nb1	; Обслуживание
	RELEASE	Evm	; Освободить ОКУ
	UNLINK	Pol, App1, 1	; Вывести из списка пользователя
	TERMINATE		; Обслуженные транзакты
Otst	LINK	Pol, P\$Nb1	; Список пользователя
	GENERATE	3600	; Время моделирования
	TERMINATE	1	

В параметр Nb1 поступающих в модель транзактов в блоке ASSIGN записывается случайное время обслуживания, вычисляемое с использованием встроенного генератора экспоненциального распределения. Если ОКУ Evm свободно, то блок GATE впускает транзакт в блок SEIZE, и ОКУ занимается на время P\$Nb1. Если же в момент поступления транзакта ОКУ занято, то блок GATE направ-

ляет транзакт в блок LINK с меткой Otst, который вводит транзакт в список пользователя Pol, упорядочивая транзакты по возрастанию времени обслуживания, записанного в параметре P\$Nbl. Блок UNLINK по освобождению ОКУ выводит с начала списка транзакт с наименьшим временем обслуживания, обеспечивая тем самым заданную дисциплину.

Пример 4.17. В одноканальной СМО с ожиданием требуется организовать дисциплину обслуживания, при которой заявки не обслуживаются, если они находятся в очереди более Vrem единиц модельного времени.

; Пример 4.17

Vrem	EQU	250	; Время ожидания в очереди
Preb	BVARIABLE	(AC1-P1) 'LE' Vrem	; Определение булевой переменной
	GENERATE	(Exponential(23,0,120))	; Источник заявок
	ASSIGN	Nbl,(Exponential(34,0,150));	Записать в параметр
	GATE NU	Evm,Otst	; ОКУ свободно?
App1	SEIZE	Evm	; Занять ОКУ
	ADVANCE	P\$Nb	; Обслуживание
	RELEASE	Evm	; Освободить ОКУ
	UNLINK	Pol,App1,1,BV\$Preb,,Term2;	Вывести из списка
Term1	TERMINATE		; Обслуженные заявки
Term2	TERMINATE		; Потерянные заявки
Otst	ASSIGN	1,AC1	; Записать в параметр абсолютное модельное время
	LINK	Pol,P1	; Список пользователя
	GENERATE	3600	; Время моделирования
	TERMINATE	1	

В параметр Nbl поступающих в модель транзактов в блоке ASSIGN записывается случайное время обслуживания. В список пользователя с именем Pol транзакты помещаются так же, как и в примере 4.16. Перед блоком LINK вставлен блок ASSIGN с меткой Otst, который в первый параметр P1 транзакта записывает абсолютное модельное время в момент помещения его в список пользователя.

В качестве операнда D блока UNLINK используется булева переменная Preb, в которой сравнивается время пребывания транзакта в очереди с заданным ограничением Vrem. Время нахождения в очереди определяется как разность между абсолютным модельным временем в момент входа выводящего транзакта в блок UNLINK и абсолютным модельным временем помещения транзактов в список пользователя, хранящимся в параметре P1. При входе выводящего транзакта булева переменная вычисляется относительно транзакта, находящегося в списке пользователя. Если результат вычисления не нулевой, т. е. заданное в булевой переменной условие выполняется, выводится один транзакт, так как операнд C равен 1.

По количеству транзактов, вошедших в блок TERMINATE с меткой Met2, можно увидеть в скольких случаях в списке пользователя не оказалось транзактов,

время обслуживания которых было меньше или равно заданному временному ограничению пребывания в очереди.

Если в данном примере в блоке UNLINK записать условие LE или использовать операнд E, то произойдет останов по ошибке «Запрещенная комбинация операндов блока UNLINK».

ЗАМЕЧАНИЕ

Ранее мы рассмотрели как можно использовать список пользователя для имитации накопителя (буфера) ограниченной емкости какой-либо реальной системы. Емкость накопителя, т. е. списка пользователя, ограничивалась таким количеством транзактов, которые могли находиться в нем одновременно. Если, например, список пользователя ограничен пятью транзактами, то это может соответствовать, например, пяти ожидаемым передачи по каналу связи сообщениям. А что если длина сообщения измеряется, например, в байтах? Значит, и объем накопителя должен измеряться в байтах. Покажем это на примере.

Пример 4.18. Вместо СЧА СН — текущей емкости списка пользователя, измеряемой в транзактах, вводится сохраняемая ячейка с именем EmkNak, в которой фиксируется текущая емкость списка пользователя, например, в байтах.

; Пример 4.18

Emk	EQU	10000	; Максимальная емкость в байтах
	GENERATE	(Exponential(12,0,12.5))	; Источник транзактов
	ASSIGN	P\$Dlina,(Normal(13,5450,350)	; Записать в параметр
	GATE U	Evm, Met1	; Занято OKY?
	TEST LE	P\$Dlina,(Emk-X\$EmkNak), Met3	; Есть место в очереди
	SAVEVALUE	EmkNak+, P\$Dlina	; Увеличить текущую емкость, байт
	LINK	Otst, P\$Dlina	; Список пользователя
Met1	SEIZE	Evm	; Занять OKY
	ADVANCE	(Exponential(12,0,32.5))	; Обслуживание
	RELEASE	Evm	; Освободить OKY
	UNLINK	Otst, Met2, 1	; Вывести из списка пользователя
	TERMINATE		; Обслуженные заявки
Met2	SAVEVALUE	EmkNak-, P\$Dlina	; Уменьшить текущую емкость, байт
	TRANSFER	, Met1	; Отправить на OKY
Met3	TERMINATE		; Потерянные сообщения

Если OKY Evm занято, транзакт, в параметр которого с именем Dlina записана длина сообщения, входит в блок TEST. Проверяется заданное в блоке TEST условие. Если длина сообщения P\$Dlina меньше или равна свободной емкости накопителя (Emk-X\$EmkNak), транзакт проходит к следующему блоку SAVEVALUE. Текущая емкость накопителя EmkNak увеличивается на длину сообщения.

Транзакт, освобождающий OKY Evm, входит в блок UNLINK и выводит из списка пользователя один транзакт ко второму блоку SAVEVALUE с меткой Met2. Текущая емкость накопителя уменьшается на длину выводимого сообщения,

после чего выведенный транзакт блоком TRANSFER направляется на ОКУ Evm и занимает его. Выводящий транзакт входит в блок TERMINATE и уничтожается.

Для перевода одноканального ОКУ в недоступное состояние используется блок FUNAVAIL. В момент перевода, если необходимо, он позволяет удалить все транзакты, находившиеся в списке прерываний, списке отложенных прерываний, списке задержки ОКУ, а также удаляет занимавший это ОКУ транзакт. А как быть с теми транзактами, которые ждали обслуживания на этом ОКУ и находились в списке пользователя, если, предположим, они все или частично должны быть удалены? Рассмотрим на примере возможный вариант удаления транзактов из списка пользователя в таких случаях.

Пример 4.19. Воспользуемся одним из приведенных ранее в примере 4.14 сегментов имитации выхода из строя и восстановления работоспособности ОКУ. Добавим в этот сегмент блок UNLINK после блока FUNAVAIL.

; Пример 4.19

```

GENERATE      , , 1                      ; Источник транзакта
Met1 ADVANCE   (Exponential(32,0,287))    ; Задержать
FUNAVAIL      Cpu1, RE, Ter1, , RE, Ter2, RE, Ter3;; Недоступность
UNLINK        NakCpu1, Met2, ALL          ; Удалить все из списка пользователя
ADVANCE       Normal(21,25,4))           ; Восстановление доступности
FAVAIL        Cpu1                        ; Перевод в доступное состояние
TRANSFER      , Met1
```

Транзакт после перевода ОКУ Cpu1 в недоступное состояние и удаления транзактов из указанных ранее списков входит как выводящий транзакт, в блок UNLINK. Так как на месте операнда C стоит ALL, то из списка пользователя с именем NakCpu1 выводятся все находившиеся в нем транзакты.

ЗАМЕЧАНИЕ

Предположим, что нужно вывести не все транзакты, а, согласно логике модели, меньшее количество. Для этого нужно задать другие, рассмотренные ранее значения операнда C. Пусть, например, нужно вывести половину имеющихся в списке пользователя транзактов. На месте операнда C запишем выражение в скобках ($CH\$NakCpu1/2$). При входе выводящего транзакта это выражение будет вычислено и округлено до целого. Из списка пользователя будет выведено полученное число транзактов, в том числе и ни одного, если результатом, в том числе и после округления, окажется нуль.

Теперь, когда знаем и работу блока UNLINK, рассмотрим ввод транзактов в список пользователя в условном режиме.

4.3.3. Ввод транзактов в список пользователя в условном режиме

Для ввода транзактов в список пользователя в условном режиме используется блок LINK формата

```
[имя]      LINK      A, B, [C]
```

Операндом *C* указывается метка блока, к которому переходит активный транзакт в случае, если индикатор (флаг) списка пользователя установлен в 0 (выключен).

Индикатор имеется у каждого списка пользователя. Когда операнд *C* не используется, т. е. блок LINK работает в безусловном режиме, индикатор устанавливается в 1 (включается) и все вошедшие в блок LINK транзакты помещаются в список пользователя.

Если поместить блок LINK в модель

	LINK	Nak, M1
Met1	SEIZE	Can
	ADVANCE	V\$Zad
	RELEASE	Can
	UNLINK	Nak, Met1, 1

непосредственно перед ОКУ с именем Can, то модель работать не будет. Все транзакты будут входить в блок LINK, помещаться в список пользователя с именем Nak и ни один транзакт не попадет на ОКУ Can. Следовательно, блок UNLINK также работать не будет.

Приведем тот же фрагмент модели, но в блоке LINK используем операнд *C*.

	LINK	Nak, M1, Met1
Met1	SEIZE	Can
	ADVANCE	V\$Zad
	RELEASE	Can
	UNLINK	Nak, Met1, 1

Индикатор списка пользователя управляется блоками LINK и UNLINK. Если в блоке LINK используется операнд *C*, блок UNLINK при обнаружении того, что список пользователя пуст, сбрасывает индикатор в 0 (выключает). Первый транзакт, вошедший в блок LINK, находит индикатор в выключенном состоянии, поэтому не помещается в список пользователя, а направляется к блоку с меткой Met1, указанной операндом *C*. После этого блок LINK устанавливает индикатор в 1 (включает).

Следующий транзакт может войти в блок LINK либо раньше, чем тот закончит обслуживание предыдущего транзакта, либо после того, как это обслуживание завершится.

Если следующий транзакт войдет в блок LINK раньше, чем будет закончено обслуживание ОКУ с именем Can предыдущего транзакта, он будет помещен в список пользователя.

Пусть теперь обслуживание предыдущего транзакта ОКУ с именем Can закончится раньше, чем очередной транзакт войдет в блок LINK. Обслуженный

транзакт, он же выводящий, войдет в блок UNLINK и будет обнаружено, что список пользователя пуст. Блок UNLINK установит индикатор списка пользователя в 0 (выключит). Поэтому следующий транзакт вновь будет направлен к блоку с меткой Met1, а не помещен в список пользователя.

Так блок LINK в условном режиме и блок UNLINK управляют помещением транзактов в список пользователя: *если список пуст, транзакт в него не помещается, а направляется к блоку с меткой, указанной операндом С.*

Блоки LINK и UNLINK предоставляют возможность пользователю формировать свои списки в динамике вне зависимости от списков задержки, которые автоматически управляются системой GPSS World.

ЗАМЕЧАНИЕ

Блоки QUEUE — DEPART для сбора статистики об очереди не используются при применении блоков LINK и UNLINK, так как почти все те же данные можно получить из статистики о списке пользователя.

Рассмотрим примеры построения моделей систем с ОКУ, функционирующими в различных режимах.

4.4. Примеры построения моделей систем с одноканальными устройствами, функционирующими в различных режимах

Пример 4.20. Направление связи состоит из двух каналов (основного и резервного) и общего накопителя емкостью на E_{mk} сообщений. Два потока поступления сообщений простейшие со средними значениями интервалов времени $T_1=3$ мин и $T_2=4$ мин. При нормальной работе сообщения передаются по основному каналу в течение времени, распределенного по экспоненциальному закону со средним значением $T_3=2$ мин. В основном канале происходят сбои через интервалы времени, распределенные по экспоненциальному закону со средним значением $T_4=15$ мин. Если сбой происходит во время передачи, то за время $T_5=1$ мин запускается резервный канал, который передает прерванное сообщение с самого начала в течение времени, распределенного по экспоненциальному закону со средним значением $T_6=3$ мин. Время восстановления канала подчинено экспоненциальному закону со средним значением $T_7=8$ мин. После восстановления резервный канал выключается и основной канал продолжает работу с очередного сообщения.

Необходимо промоделировать функционирование направления связи в течение 1 ч. Определить:

- рациональную емкость E_{mk} накопителя;
- загрузку основного Os_{nk} и резервного Res_{nk} каналов связи;
- вероятность безотказной работы $VerO_{tk}$ основного канала связи;
- вероятность V_{Nak} отказа в передаче сообщений вследствие полного заполнения накопителя;

- вероятности передачи сообщений потока 1 и потока 2 соответственно;
- вероятность Ver передачи сообщений направлением связи в целом;
- вероятность V0tk отказа в передаче сообщений направлением связи.

; Пример 4.20

; Задание исходных данных

Emk	EQU	5	; Емкость накопителя
VrMod	EQU	3600	; Время моделирования
T1	EQU	180	; Средний интервал поступления сообщений потока 1
T2	EQU	240	; Средний интервал поступления сообщений потока 2
T3	EQU	120	; Среднее время передачи по 0snK
T4	EQU	900	; Средний интервал времени выхода из строя 0snK
T5	EQU	60	; Время включения ResK
T6	EQU	180	; Среднее время передачи по ResK
T7	EQU	480	; Среднее время восстановления 0snK

; Описание арифметических выражений

; Вероятность передачи сообщений потока 1

Ver1 VARIABLE (N\$Term1+N\$Term3)/N\$Soob1

; Вероятность передачи сообщений потока 2

Ver2 VARIABLE (N\$Term2+N\$Term4)/N\$Soob2

; Вероятность передачи сообщений потоков 1 и 2

Ver VARIABLE (V\$Ver1+V\$Ver2)/2

; Вероятность отказа в передаче сообщений потоков 1 и 2

V0tk VARIABLE 1-X\$Ver

; Вероятность безотказной работы

Ver0tk VARIABLE (AC1-X\$Vr0tk)/AC1

; Сегмент имитации сообщений потока 1

	GENERATE	(Exponential(12,0,T1));	Генератор сообщений потока 1
Soob1	ASSIGN	1,1	; Код 1 в P1 – сообщения потока 1
	TRANSFER	,Nakop	; Направить на ResK

; Сегмент имитации сообщений потока 2

	GENERATE	(Exponential(15,0,T2));	Генератор сообщений потока 2
Soob2	ASSIGN	1,2	; Код 2 в P1 – сообщения потока 2

; Сегмент имитации работы накопителя и 0snK

Nakop	GATE FV	0snK,KRes	; Доступен ли 0snK? Если нет, на ResK
	GATE NU	0snK,Spis	; Свободен ли 0snK? Если нет, в накопитель
Prov3	SEIZE	0snK	; Занять 0snK
	ADVANCE	(Exponential(11,0,T3))	; Обслуживание
	RELEASE	0snK	; Освободить 0snK
	UNLINK	Nak,Prov3,1	; Вывод из накопителя одного транзакта на 0snK
	TEST E	P1,1,Term2;	Сообщение потока 1 или 2 передано по 0snK?
Term1	TERMINATE		; Счет переданных сообщений потока 1 по 0snK
Term2	TERMINATE		; Счет переданных сообщений потока 2 по 0snK

```

;      Список пользователя Nak
Spis   TEST L      CH$Nak,Emk,Term7 ; Есть ли место в накопителе?
      LINK      Nak,FIFO ; Если да, поместить сообщение в накопитель
Term7  TEST E      P1,1,Term6; Сообщение потока 1 или потока 2 потеряно?
Term5  TERMINATE          ; Счет потерянных сообщений потока 1
Term6  TERMINATE          ; Счет потерянных сообщений потока 2
; Сегмент имитации работы Resk
KRes   GATE NU     ResK,Spis ; Свободен Resk? Сообщение в накопитель
      ADVANCE     T5          ; Включение Resk
Prov1  SEIZE       ResK      ; Занять Resk
      ADVANCE     (Exponential(12,0,T6)) ; Передача по Resk
      RELEASE     ResK      ; Освободить Resk
      GATE FNV     OsnK,Prov2 ; Доступен ли OsnK?
      UNLINK      Nak,Prov1,1 ; Если нет, из накопителя сообщение на Resk
Prov2  TEST E      P1,1,Term4 ; Сообщение потока 1 или 2 передано по Resk?
Term3  TERMINATE          ; Счет переданных сообщений потока 1
Term4  TERMINATE          ; Счет переданных сообщений потока 2
; Сегмент имитации выхода из строя OsnK
      GENERATE     ,,,1
Term8  ADVANCE     (Exponential(12,0,T4)) ; Расчет времени до отказа
      FUNAVAIL     OsnK      ; Выход из строя OsnK
      ASSIGN       1,(Exponential(12,0,T7)) ; Время восстановления OsnK
      ADVANCE     P1          ; Имитация восстановления OsnK
      SAVEVALUE    Vr0tk+,P1 ; Учет времени восстановления OsnK
      FAVAIL       OsnK      ; Окончание восстановления OsnK
      TRANSFER     ,Term8
; Сегмент задания времени моделирования
; и вычисления результатов моделирования
      GENERATE     VrMod
      SAVEVALUE    Ver1,V$Ver1 ; Вероятность передачи сообщений потока 1
      SAVEVALUE    Ver2,V$Ver2 ; Вероятность передачи сообщений потока 2
      SAVEVALUE    Ver,V$Ver ; Вероятность передачи сообщений
      SAVEVALUE    V0tk,V$V0tk ; Вероятность отказа в передаче сообщений
      SAVEVALUE    Ver0tk,V$Ver0tk; Вероятность безотказной работы OsnK
      TERMINATE    1

```

Результаты моделирования, полученные при 5 000 прогонов, приведены ниже.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER
OSNK	91312	0.605	119.220	1	179442	0	0
RESK	39087	0.392	180.377	1	0	0	0

USER	CHAIN	SIZE	RETRY	AVE.CONT	ENTRIES	MAX	AVE.TIME
NAK		4	0	3.009	107207	5	505.278

SAVEVALUE	RETRY	VALUE
VER1	0	0.747
VER2	0	0.748
VER	0	0.748
VOTK	0	0.252
VEROTK	0	0.653

Пример 4.21. Локальная вычислительная сеть (ЛВС) имеет три автоматизированных рабочих места (АРМ) и сервер, соединенных общей шиной. В локальной вычислительной сети организована распределенная обработка данных. Запросы поступают от АРМ, которые имеют свои базы данных (БД). Поступающие запросы первично обрабатываются на АРМ. С вероятностью P_i ($i=1..3$, порядковые номера АРМ) требующаяся информация обнаруживается в БД АРМ, после чего продолжается дальнейшая обработка запроса. В противном случае необходима посылка запроса на сервер. После посылки запроса на сервер АРМ обрабатывает другие поступающие на него запросы. Получив ответ с сервера, АРМ завершает обработку запроса.

Интервалы времени поступления запросов на АРМ распределены по показательному закону со средним значением $T1_i$.

Канал передачи данных (КПД), соединяющий АРМ и сервер, не имеет накопителей и передача по нему возможна только тогда, когда он свободен. Когда он занят, АРМ находится в режиме ожидания и только после освобождения канала передает запрос. При передаче данных с сервера по запросу АРМ этим данным присваивается более высокий приоритет по сравнению с поступающими на АРМ запросами. Этим обеспечивается дисциплина обслуживания «раньше пришел — раньше обслужен».

На сервере имеются два накопителя. Накопитель 1 предназначен для поступающих запросов, а второй — для передаваемых ответов на запросы. Емкость накопителя 1 ограничена на $Emk1$ запросов, т. е. поступающие с АРМ запросы могут теряться в случае полного заполнения накопителя 1. Емкость накопителя 2 практически бесконечна, т. е. ответы на запросы не теряются.

Время первичной обработки запроса, его передачи при надобности на сервер, обработки на сервере и обратной передачи на нужное АРМ подчинены экспоненциальному закону со средними значениями соответственно $T2_i$, $T3_i$, $T4_i$ и $T5_i$. Значения исходных данных приведены в табл. 4.1.

Таблица 4.1

Характеристики	АРМ		
	1	2	3
Средний интервал времени поступления запросов на АРМ, $T1_i$, сек	68	61	64

Таблица 4.1 (продолжение)

Вероятность P_i нахождения информации в базе данных АРМ	0,75	0,6	0,4
Среднее время $T2_i$ первичной обработки запроса на АРМ, сек	3	3,5	4
Среднее время $T3_i$ передачи запроса на сервер, сек	2,5	3,5	2
Среднее время $T4_i$ обработки запроса на сервере, сек	26	31	28
Среднее время $T5_i$ передачи данных с сервера на АРМ, сек	2,7	3	2,2
Среднее время $T6_i$ обработки запроса на АРМ, сек	26	20	13,5

Необходимо промоделировать функционирование ЛВС в течение 1 часа.

Определить:

- рациональную емкость накопителя 1 на сервере, при которой не происходит потерь запросов с АРМ;
- вероятность отказа в ответе на запрос с АРМ вследствие полного заполнения накопителя 1 на сервере;
- вероятности обработки запросов, поступающих на АРМ;
- загрузку АРМ, канала передачи данных и сервера.

; Пример 4.21

MT MATRIX ,6,3 ; Описание матрицы

; Задание исходных данных

Emk1 EQU 2 ; Емкость накопителя 1 на сервере

; Средний интервал поступления запросов на:

IntArm1 EQU 32 ; АРМ1

IntArm2 EQU 25 ; АРМ2

IntArm3 EQU 30 ; АРМ3

; Вероятность нахождения информации в базе данных на:

VerArm1 EQU .750 ; АРМ1

VerArm2 EQU .600 ; АРМ2

VerArm3 EQU .400 ; АРМ3

INITIAL X\$VrMod,3600 ; Время моделирования

; Среднее время первичной обработки запросов на:

INITIAL Mx\$MT(2,1),3 ; АРМ1

INITIAL Mx\$MT(2,2),3.5 ; АРМ2

INITIAL Mx\$MT(2,3),4 ; АРМ3

```

;           Среднее время передачи запросов на сервер с:
INITIAL    MX$MT(3,1),2.5    ; APM1
INITIAL    MX$MT(3,2),3.5    ; APM2
INITIAL    MX$MT(3,3),2      ; APM3

;           Среднее время обработки запросов на сервере с:
INITIAL    MX$MT(4,1),26     ; APM1
INITIAL    MX$MT(4,2),31     ; APM2
INITIAL    MX$MT(4,3),28     ; APM3

;           Среднее время передачи данных с сервера на:
INITIAL    MX$MT(5,1),2.7    ; APM1
INITIAL    MX$MT(5,2),3      ; APM2
INITIAL    MX$MT(5,3),2.2    ; APM3

;           Среднее время обработки запросов на:
INITIAL    MX$MT(6,1),26     ; APM1
INITIAL    MX$MT(6,2),20     ; APM2
INITIAL    MX$MT(6,3),13.5   ; APM3

; Описание арифметических выражений
;           Вероятность обработки запросов с:
VArm1      VARIABLE    (N$Ter1+N$Ter7)/N$NS1    ; APM1
VArm2      VARIABLE    (N$Ter2+N$Ter8)/N$NS2    ; APM2
VArm3      VARIABLE    (N$Ter3+N$Ter9)/N$NS3    ; APM3

;           Вероятность потери запросов:
VNaK       VARIABLE    N$Met8/N$PZ              ; ЛВС
VNaK1      VARIABLE    N$Ter4/N$NS1             ; APM1
VNaK2      VARIABLE    N$Ter5/N$NS2             ; APM2
VNaK3      VARIABLE    N$Ter6/N$NS3             ; APM3

;           Среднее время обработки запросов (реакция ЛВС):
RArm1      VARIABLE    (X$VArm11+X$VArm12)/(N$Ter1+N$Ter7) ; APM1
RArm2      VARIABLE    (X$VArm21+X$VArm22)/(N$Ter2+N$Ter8) ; APM2
RArm3      VARIABLE    (X$VArm31+X$VArm32)/(N$Ter3+N$Ter9) ; APM3

; Сегмент имитации APM1
NS1        GENERATE    (Exponential(1,0,IntArm1)); Генератор запросов на APM1
ASSIGN     1,1          ; Код 1 – запрос с APM1
QUEUE      ArmQ1        ; Встать в очередь запросов на APM1
SEIZE      Arm1          ; Занять Arm1
DEPART     ArmQ1        ; Покинуть очередь ArmQ1
ADVANCE    (Exponential(1,0,MX$MT(2,P1))); Обработка запроса
TRANSFER   VerArm1,Met2,Met1 ; Есть информация в базе данных?
Met1       ADVANCE      (Exponential(1,0,MX$MT(6,P1))); Обработка запроса
RELEASE    Arm1          ; Освободить APM1
SAVEVALUE  VArm11+,M1    ; Суммарное время обработки запросов
TER1       TERMINATE    ; Запросы, обработанные на APM1 без запроса на сервер
Met2       RELEASE      Arm1          ; Освободить APM1

```

PRIORITY 1 ; Запросу на сервер – более высокий приоритет
 TRANSFER ,Met7 ; Отправка запроса в КПД

; Сегмент имитации APM2

GENERATE(Exponential(2,0,IntArm2)) ; Генератор запросов на APM2
 NS2 ASSIGN 1,2 ; Код 2 – запрос с APM2
 QUEUE ArmQ2 ; Встать в очередь запросов на APM2
 SEIZE Arm2 ; Занять APM2
 DEPART ArmQ2 ; Покинуть очередь ArmQ2
 ADVANCE (Exponential(2,0,MX\$MT(2,P1))); Обработка запроса
 TRANSFER VerArm2,Met5,Met3 ; Есть информация в базе данных?
 Met3 ADVANCE (Exponential(2,0,MX\$MT(6,P1))); Обработка запроса
 RELEASE Arm2 ; Освободить APM2
 SAVEVALUE VArm21+,M1 ; Суммарное время обработки запросов
 Ter2 TERMINATE ; Запросы, обработанные на APM2 без запроса на сервер
 Met5 RELEASE Arm2 ; Освободить APM2
 PRIORITY 1 ; Запросу на сервер – более высокий приоритет
 TRANSFER ,Met7 ; Отправка запроса в КПД

; Сегмент имитации APM3

GENERATE (Exponential(3,0,IntArm3)); Генератор запросов на APM3
 NS3 ASSIGN 1,3 ; Код 3 – запрос с APM3
 QUEUE ArmQ3 ; Встать в очередь запросов на APM3
 SEIZE Arm3 ; Занять Arm3
 DEPART ArmQ3 ; Покинуть очередь ArmQ3
 ADVANCE (Exponential(3,0,MX\$MT(2,P1))); Обработка запроса
 TRANSFER VerArm3,Met6,Met4 ; Есть информация в базе данных?
 Met4 ADVANCE (Exponential(3,0,MX\$MT(6,P1))); Обработка запроса
 RELEASE Arm3 ; Освободить APM3
 SAVEVALUE VArm31+,M1 ; Суммарное время обработки запросов
 Ter3 TERMINATE ; Запросы, обработанные на APM3 без запроса на сервер
 Met6 RELEASE Arm3 ; Освободить APM3
 PRIORITY 1 ; Запросу на сервер – более высокий приоритет

; Сегмент имитации работы канала передачи данных

Met7 GATE NU Can ; Проверка: свободен ли КПД?
 SEIZE Can ; Да, занять КПД
 ADVANCE (Exponential(4,0,MX\$MT(3,P1))); Передача данных
 PZ RELEASE Can ; Освободить КПД

; Сегмент имитации работы накопителя 1 и сервера

TEST L Q\$SerQ1,Emk1,Met8 ; Есть место в накопителе?
 QUEUE SerQ1 ; Встать в очередь в накопителе 1 на сервере
 SEIZE \ Server ; Занять сервер
 DEPART SerQ1 ; Покинуть очередь в накопителе 1
 ADVANCE (Exponential(5,0,MX\$MT(4,P1))); Обработка запроса
 RELEASE Server ; Освободить сервер

; Сегмент имитации работы накопителя 2 и канала передачи данных

```

QUEUE      SerQ2      ; Занять очередь в накопителе 2 на сервере
GATE NU    Can        ; Свободен ли КПА?
SEIZE      Can        ; Да, занять КПА
DEPART     SerQ2      ; Покинуть очередь в накопителе 2
ADVANCE    (Exponential(6,0,MX$MT(5,P1))); Передача запроса
RELEASE    Can        ; Освободить КПА

```

; Сегмент обработки запросов на АРМ1 после получения данных с сервера

```

TEST E     P1,1,Met11 ; Запрос с АРМ1?
SEIZE      Arm1       ; Занять АРМ1
ADVANCE    (Exponential(7,0,MX$MT(6,P1))); Обработка запроса
RELEASE    Arm1       ; Освободить АРМ1
SAVEVALUE  VArm12+,M1 ; Суммарное время обработки запросов

```

Ter7 TERMINATE ; Запросы, обработанные с запросами на сервер

; Сегмент обработки запросов на АРМ2 после получения данных с сервера

```

Met11     TEST E     P1,2,Met12 ; Запрос с АРМ2?
SEIZE      Arm2       ; Занять АРМ2
ADVANCE    (Exponential(7,0,MX$MT(6,P1))); Обработка запроса
RELEASE    Arm2       ; Освободить АРМ2
SAVEVALUE  VArm22+,M1 ; Суммарное время обработки запросов

```

Ter8 TERMINATE ; Запросы, обработанные с запросами на сервер

; Сегмент обработки запросов на АРМ3 после получения данных с сервера

```

Met12     SEIZE      Arm3       ; Занять АРМ3
ADVANCE    (Exponential(7,0,MX$MT(6,P1))); Обработка запроса
RELEASE    Arm3       ; Освободить АРМ3
SAVEVALUE  VArm32+,M1 ; Суммарное время обработки запросов

```

Ter9 TERMINATE ; Запросы, обработанные с запросами на сервер

; Сегмент уничтожения запросов при заполнении накопителя 1 на сервере

```

Met8      TEST E     P1,1,Met9

```

Ter4 TERMINATE ; Уничтоженные запросы с АРМ1

```

Met9      TEST E     P1,2,Ter6

```

Ter5 TERMINATE ; Уничтоженные запросы с АРМ2

Ter6 TERMINATE ; Уничтоженные запросы с АРМ3

; Сегмент задания времени моделирования и вывода результатов моделирования

```

GENERATE   X$VrMod

```

; **Вероятность обработки запросов с:**

```

SAVEVALUE  VArm1,V$VArm1 ; АРМ1

```

```

SAVEVALUE  VArm2,V$VArm2 ; АРМ2

```

```

SAVEVALUE  VArm3,V$VArm3 ; АРМ3

```

; **Вероятность потери запросов:**

```

SAVEVALUE  V$Nak,V$V$Nak ; ЛВС

```

```

SAVEVALUE  V$Nak1,V$V$Nak1 ; АРМ1

```

```

SAVEVALUE  V$Nak2,V$V$Nak2 ; АРМ2

```

```

SAVEVALUE  V$Nak3,V$V$Nak3 ; АРМ3

```

```

;               Среднее время обработки запросов с:
SAVEVALUE      RArm1,V$RArm1      ; APM1
SAVEVALUE      RArm2,V$RArm2      ; APM2
SAVEVALUE      RArm3,V$RArm3      ; APM3
TERMINATE      1
    
```

Выполним 1 000 прогонов и приведем полученные результаты.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER
Arm1	130063	0.828	22.912	1	377783	0	0
Arm2	179774	0.820	16.417	1	377838	0	0
CAN	257134	0.187	2.617	1	377847	0	0
SERVER	99186	0.793	28.786	1	377841	0	0
Arm3	166184	0.487	10.544	1	0	0	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)
ArmQ1	42	10	112531	19391	3.946	126.231	152.511
ArmQ2	48	1	144178	25815	3.701	92.404	112.558
SERQ1	2	1	99187	27350	0.897	32.565	44.963
ArmQ3	16	0	120139	61429	0.431	12.908	26.415
SERQ2	4	0	99185	80870	0.019	0.676	3.658

SAVEVALUE	RETRY	VALUE
VArm1	0	0.906
VArm2	0	0.848
VArm3	0	0.782
VNAK	0	0.372
VNAK1	0	0.094
VNAK2	0	0.152
VNAK3	0	0.218
RArm1	0	170.336
RArm2	0	143.216
RArm3	0	66.116

Пример 4.22. В примере 4.21 запросы со всех АРМ имели равные приоритеты. Однако на практике приоритеты запросов, как правило, разные. Рассмотрим, как можно модифицировать модель примера 4.21 в случае неодинаковых приоритетов запросов.

Пусть запросы с АРМ имеют относительные приоритеты, причем высший приоритет имеют запросы с АРМ1, а низший — с АРМ3. Для установления таких приоритетов в блоки PRIORITY сегментов, имитирующих работу АРМ1, АРМ2 и АРМ3, операндом А укажем 3, 2 и 1 соответственно.

Выполним 1 000 прогонов и приведем полученные результаты.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER
ARM1	129976	0.827	22.912	1	0	0	0
ARM2	179296	0.814	16.342	1	376212	0	0
CAN	256246	0.186	2.619	1	0	0	0
SERVER	98993	0.791	28.783	1	376205	0	0
ARM3	164931	0.482	10.525	1	376203	0	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)
ARMQ1	54	0	112071	19472	3.913	125.688	152.118
ARMQ2	58	1	143486	26742	3.453	86.639	106.484
SERQ1	2	1	98994	27341	0.895	32.533	44.946
ARMQ3	14	3	119658	61910	0.414	12.446	25.789
SERQ2	5	0	98992	80416	0.019	0.706	3.761

SAVEVALUE	RETRY	VALUE
VARM1	0	0.909
VARM2	0	0.848
VARM3	0	0.781
VNAK	0	0.370
VNAK1	0	0.091
VNAK2	0	0.152
VNAK3	0	0.219
RARM1	0	168.653
RARM2	0	135.546
RARM3	0	69.158

Из результатов следует, что для системы с данными параметрами введение относительных приоритетов не привело к каким-либо значительным изменениям. Например, уменьшение реакции ЛВС на запросы с АРМ1 составляет 1,7, а увеличение ее реакции на запросы с АРМ2 и АРМ3 — примерно 8 и 3 единицы модельного времени соответственно (1 с = 1 ед. модельного времени).

Пусть теперь запросы с АРМ1 имеют абсолютный приоритет, а запросы с АРМ2 — относительный приоритет. Поскольку запросы с АРМ1 имеют абсолютный приоритет, т. е. они могут прерывать обработку запросов с АРМ2 и АРМ3, необходимо использовать для организации канала передачи данных и сервера, передающих и обрабатывающих запросы с АРМ1, блоки PREEMPT и RETURN.

Дополним модель примера 4.21 следующими сегментами.

; Сегмент имитации работы канала передачи данных с АРМ1

```
Met13      GATE NI      Can      ; КПД не прерван?
           PREEMPT      Can      ; Да, занять КПД
           ADVANCE      (Exponential(4,0,MX$MT(3,P1))); Передача данных
PZ1        RETURN      Can      ; Освободить КПД
```

; Сегмент имитации работы накопителя 3 и сервера

```
TEST L      Q$SerQ3,Emk3,Met8 ; Есть место в накопителе?
QUEUE      SerQ3 ; Встать в очередь в накопителе 3 на сервере
PREEMPT    Server ; Занять сервер
DEPART     SerQ3 ; Покинуть очередь в накопителе 3
ADVANCE    (Exponential(5,0,MX$MT(4,P1))); Обработка запроса
RETURN     Server ; Освободить сервер
```

; Сегмент имитации работы накопителя 4 и канала передачи данных

```
QUEUE      SerQ4 ; Занять очередь в накопителе 4 на сервере
GATE NI    Can ; Свободен ли КПД?
PREEMPT    Can ; Да, занять КПД
DEPART     SERQ4 ; Покинуть очередь в накопителе 4
ADVANCE    (Exponential(6,0,MX$MT(5,P1))); Передача запроса
RETURN     Can ; Освободить КПД
TRANSFER ,Met14
```

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER
ARM1	140519	0.908	23.258	1	377488	0	0
ARM2	177133	0.805	16.365	1	377497	0	0
CAN	261032	0.190	2.618	1	0	0	0
SERVER	103583	0.819	28.475	1	377483	0	0
ARM3	162425	0.473	10.476	1	0	0	0

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)
ARMQ1	71	2	112395	10298	8.148	260.981	287.305
ARMQ2	41	6	144270	28126	3.347	83.514	103.738
SERQ1	2	1	75458	18879	0.971	46.311	61.763
ARMQ3	15	0	119838	63393	0.404	12.149	25.793
SERQ3	5	0	28126	20669	0.080	10.181	38.401
SERQ4	2	0	28126	27755	0.000	0.036	2.701
SERQ2	4	0	75456	60636	0.015	0.739	3.763

SAVEVALUE	RETRY	VALUE
VARM1	0	1.000
VARM2	0	0.830
VARM3	0	0.755
VNAK	0	0.342
VNAK1	0	0.000
VNAK2	0	0.170
VNAK3	0	0.245
RARM1	0	308.153
RARM2	0	136.150
RARM3	0	77.527

Читателю рекомендуется самостоятельно получить такие же результаты и проанализировать их. Это будет способствовать закреплению и более глубокому пониманию материала первых четырех глав, а также приобретению новых и совершенствованию имевшихся навыков интерпретации результатов имитационного моделирования.

Рассмотрены методы построения моделей систем с одноканальными устройствами, функционирующими в различных режимах. Перейдем к методам построения моделей систем с многоканальными устройствами и переключателями.

5. Построение моделей систем с многоканальными устройствами и переключателями

Два или более обслуживающих устройств могут быть промоделированы на GPSS двумя или более ОКУ, располагаемыми рядом, т. е. параллельно. Так нужно поступать, когда отдельные устройства являются разнородными, то есть характеризуются различными свойствами, например, различной интенсивностью обслуживания.

Однако часто различные параллельно работающие устройства являются однородными. GPSS представляет для моделирования однородных параллельных устройств специальное средство, именуемое многоканальным устройством. МКУ может быть использовано несколькими транзактами одновременно. Ограничений на число МКУ в модели нет. Для различия им дают имена. МКУ определяется до его использования командой STORAGE. Формат команды:

Name	STORAGE	A

Name — имя МКУ. Символическому имени может быть поставлен в соответствие номер командой EQU. Это необходимо, если требуется обращаться к нескольким МКУ в блоках SELECT и COUNT. Операнд A может быть только целым положительным числом. Иные способы задания емкости вызывают ошибку.

В модели можно организовать функционирование МКУ в двух режимах:

- занятие и освобождение МКУ;
- недоступность МКУ.

5.1. Занятие многоканального устройства и его освобождение

Занятие и освобождение МКУ имитируется блоками ENTER (войти) и LEAVE (выйти). Форматы блоков:

ENTER	A, [B]
LEAVE	A, [B]

Операнд A в обоих блоках используется для указания имени, соответствующего МКУ. Операнд B задает число устройств (элементов памяти), которое

должно быть занято в блоке ENTER или освобождено в блоке LEAVE. По умолчанию операнд $B = 1$. При $B = 0$ блок считается неработоспособным.

Когда транзакт входит в блок ENTER, операнд A используется для нахождения МКУ с указанным именем. Если такого МКУ нет, происходит останов по ошибке «Обращение к несуществующей памяти». Если МКУ существует и задан операнд B, он вычисляется, округляется до целого и полученный результат используется для оценки свободной емкости. Транзакт может войти в блок ENTER, если МКУ находится в доступном состоянии и достаточно емкости для выполнения запроса. В противном случае транзакт помещается в список задержки устройства в соответствии с приоритетом.

Когда транзакт входит в блок ENTER, планировщик выполняет следующие действия:

- увеличивает на 1 счетчик входов МКУ;
- увеличивает на значение операнда B (по умолчанию на 1) текущее содержимое МКУ;
- уменьшает на значение операнда B (по умолчанию на 1) доступную емкость МКУ.

Если транзакт при входе в блок ENTER запрашивает больше устройств (элементов памяти), чем определено командой STORAGE, т. е. ее операнд A меньше операнда B блока ENTER, возникает ошибка «Запрос элементов памяти превышает ее общую емкость».

МКУ никогда не может быть удалено из текущей модели, даже если команда STORAGE удаляется из рабочей программы. МКУ можно переопределить, т. е. изменить емкость другой командой STORAGE с тем же самым именем.

Например:

Batr	STORAGE	18

Повторное описание

Batr	STORAGE	24

Имитация обслуживания в течение какого-то промежутка времени также осуществляется блоком ADVANCE.

Пример 1.

Nak	STORAGE	20
	ENTER	Nak, 2
	ADVANCE	120, 40
	LEAVE	Nak, 2

Командой STORAGE определяется МКУ с именем Nak емкостью 20 единиц. При входе транзакта в блок ENTER занимается 2 единицы и столько же освобождается в блоке LEAVE при выходе из МКУ.

Пример 2.

Nak	STORAGE	20
	ENTER	Nak, 21
	ADVANCE	120, 40
	LEAVE	Nak, 2

При входе транзакта в блок ENTER произойдет останов по ошибке, так как транзакт будет пытаться занять больше каналов (21), чем определено (20) командой STORAGE. То же самое произойдет, если при выходе из блока LEAVE транзакт будет пытаться освободить каналов больше, чем определено командой STORAGE.

Пример 3.

Pun1	EQU	1
Pun2	EQU	2
Pun3	EQU	3
Pun1	STORAGE	6
Pun2	STORAGE	5
Pun3	STORAGE	3
	ENTER	*1
	ADVANCE	MX\$NorVr(P2, P3)
	LEAVE	*1

В данном примере определены три МКУ с именами Pun1, Pun2, Pun3 и емкостями 6, 5 и 3 соответственно. Именам командами EQU поставлены в соответствие номера 1, 2 и 3. Предполагается, что при входе транзакта в блок ENTER в его первом параметре содержится какой-либо один из трех номеров. Согласно этому номеру и занимается МКУ, а затем освобождается. Операнд В в блоках ENTER и LEAVE не используется, поэтому транзактом занимается и освобождается одна единица емкости МКУ.

5.2. Перевод многоканального устройства в недоступное состояние и восстановление доступности

Недоступность МКУ моделируется блоком SUNAVAIL (символ S означает МКУ, UNAVAIL — недоступный). Формат блока:

SUNAVAIL A

Операнд A — имя или номер МКУ, может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА*параметр.

Например:

SUNAVAIL Batr

Когда транзакт входит в этот блок, МКУ *Batr* становится недоступным. Если при переводе в недоступное состояние в МКУ находились транзакты, т. е. текущее содержимое МКУ не равнялось нулю, то обслуживание этих транзактов продолжается, пока текущее содержимое не станет равным нулю. Транзакты, которые пытаются занять МКУ во время нахождения его в недоступном состоянии, не входят в блок *ENTER* и помещаются в список задержки МКУ.

Нахождение в недоступном состоянии продолжается до тех пор, пока транзакт не войдет в блок *SAVAIL*. Формат блока:

SAVAIL A

Операнд *A* — имя или номер МКУ. Может быть таким же, как операнд *A* в блоке *SUNAVAIL*.

Если в момент перевода МКУ в доступное состояние в его списке задержки были транзакты, им предоставляется возможность занять МКУ в соответствии с дисциплиной «first-fit-with-skip» (первый подходящий с пропусками). Транзакты, которым будет отказано в занятии МКУ, остаются в списке задержки.

; Пример 5.1

; Определение МКУ

Can STORAGE 3

; Сегмент имитации поступления и обслуживания транзактов

GENERATE	, , , 3	; Источник транзактов
ENTER	Can	; Занять МКУ
ADVANCE	10	; Обслуживание
LEAVE	Can	; Освободить каналы МКУ
TERMINATE		; Обслуженные транзакты

; Сегмент имитации недоступности

GENERATE	5, , , 1	; Транзакт-инициатор недоступности
SUNAVAIL	Can	; Перевод в состояние недоступности
ADVANCE	6	; Восстановление доступности
SAVAIL	Can	; Перевод в состояние доступности
TERMINATE	1	

Блок *GENERATE* первого сегмента генерирует в $t = 0$ три транзакта, которые занимают МКУ *Can* емкостью 3, определенной командой *STORAGE*.

В $t = 5$ блок *GENERATE* второго сегмента генерирует транзакт, который входит в блок *SUNAVAIL* и переводит МКУ *Can* в недоступное состояние.

Приведем результаты моделирования.

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	11.000	10	0	1

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
1		GENERATE	3	0	0
2		ENTER	3	0	0
3		ADVANCE	3	0	0
4		LEAVE	3	0	0
5		TERMINATE	3	0	0
6		GENERATE	1	0	0
7		SUNAVAIL	1	0	0
8		ADVANCE	1	0	0
9		SAVAIL	1	0	0
10		TERMINATE	1	0	0

МКУ Кан в недоступном состоянии находилось 6 единиц модельного времени и в $t = 11$ блоком SAVAIL была восстановлена его доступность. Время обслуживания транзактов равно 10 единицам. По результатам моделирования видно, что и в период недоступности обслуживание находившихся в МКУ трех транзактов продолжалось еще 5 единиц модельного времени и завершилось в $t = 10$. Все три обслуженных транзакта вошли в блок TERMINATE первого сегмента.

Пример 5.2. В примере 5.1 обслуживание трех транзактов закончилось до того, как устройство вновь стало доступным. Рассмотрим случай, когда обслуживание некоторых транзактов не закончилось до восстановления доступности.

; Пример 5.2

; Определение МКУ

Kan STORAGE 3

; Сегмент 1

GENERATE 3,,3 ; Источник транзактов

ENTER Kan ; Занять МКУ

ADVANCE 4 ; Обслуживание

LEAVE Kan ; Освободить каналы МКУ

TERMINATE ; Обслуженные транзакты

; Сегмент 2

GENERATE 11,,1 ; Транзакт-инициатор недоступности

SUNAVAIL Kan ; Перевод в состояние недоступности

ADVANCE 1 ; Восстановление доступности

SAVAIL Kan ; Перевод в состояние доступности

ADVANCE 1 ; Задержать

TERMINATE 1

Блок GENERATE первого сегмента генерирует три транзакта с интервалами в 3 единицы модельного времени. Первый транзакт поступит в модель в $t = 3$, займет один канал МКУ и обслужится в $t = 7$.

Второй транзакт займет второй канал МКУ в $t = 6$. Его обслуживание закончится в $t = 10$.

Третий транзакт в $t = 9$ займет один из двух свободных каналов МКУ. В $t = 11$ блок GENERATE второго сегмента сгенерирует транзакт, который войдет в блок SUNAVAIL и МКУ станет недоступным. Обработка третьего транзакта продолжается. В $t = 12$ МКУ вновь будет доступным. Обработка третьего транзакта продолжится и завершится в $t = 13$.

Приведем результаты моделирования.

START TIME		END TIME	BLOCKS	FACILITIES	STORAGES	
0.000		13.000	11	0	1	
LABEL	LOC	BLOCK TYPE	ENTRY	COUNT	CURRENT	COUNT RETRY
	1	GENERATE	3		0	0
	2	ENTER	3		0	0
	3	ADVANCE	3		0	0
	4	LEAVE	3		0	0
	5	TERMINATE	3		0	0
	6	GENERATE	1		0	0
	7	SUNAVAIL	1		0	0
	8	ADVANCE	1		0	0
	9	SAVAIL	1		0	0
	10	ADVANCE	1		0	0
	11	TERMINATE	1		0	0

Они точно такие же, как и в примере 5.1. То есть недоступность только за-прещает поступление транзактов в МКУ. Обработка же их продолжается, как будто и не было никакой недоступности.

Таким образом, осуществить имитацию выхода МКУ из строя, при котором все транзакты, находившиеся в МКУ на обслуживании теряются, блоками SUNAVAIL и SAVAIL невозможно.

5.3. Проверка состояния многоканального устройства

5.3.1. Проверка состояния блоком GATE

Состояние МКУ, как и состояние ОКУ, проверяется блоком GATE такого же формата:

GATE X A,[B]

Отличие состоит в значениях условного оператора X, которые могут быть следующими:

- SE — МКУ, заданное операндом A, пусто;
- SF — МКУ, заданное операндом A, заполнено;
- SNE — МКУ, заданное операндом A, не пусто;
- SNF — МКУ, заданное операндом A, не заполнено;
- SNV — МКУ, заданное операндом A, не доступно;

○ SV — МКУ, заданное операндом A, доступно.

Блок GATE также работает в двух режимах:

○ отказа во входе;

○ разрешении во входе и альтернативном выходе.

Например:

GATE SNF Can, Met5

Если МКУ с именем Can не заполнено, т. е. имеются свободные каналы (элементы памяти), заданное в блоке GATE условие выполняется и транзакт будет направлен к следующему блоку. Если МКУ будет заполнено, транзакт будет направлен к блоку с меткой Met5.

Пример 5.3. МКУ переводится в недоступное состояние. Обслуживание транзактов, находившихся в МКУ к моменту перевода его в недоступное состояние, продолжается, но вновь поступающие транзакты не помещаются в список задержки МКУ. Восстановление доступности МКУ начинается после того, как МКУ будет пусто.

; Пример 5.3

; Определение МКУ

Sist STORAGE 7

; Сегмент имитации поступления и обслуживания транзактов

GENERATE 2,,,5 ; Источник транзактов

GATE SV Sist, Met1 ; МКУ доступно?

GATE SNF Sist, Met1 ; Да, тогда МКУ не заполнено? Нет, тогда

ENTER Sist, 3 ; Занять, если есть свободные каналы МКУ

ADVANCE 5 ; Обслуживание

LEAVE Sist, 3 ; Освободить каналы МКУ

TERMINATE 1 ; Обслуженные транзакты

Met1 TERMINATE 1 ; Потерянные транзакты

; Сегмент имитации недоступности МКУ

GENERATE ,,, 1 ; Транзакт-инициатор недоступности

Met2 ADVANCE 7 ; Время нахождения в доступном состоянии

SUNAVAIL Sist ; Перевод в состояние недоступности

GATE SE Sist ; МКУ пусто?

ADVANCE 1 ; Да, тогда восстановление доступности

SAVAIL Sist ; Перевод в состояние доступности

TRANSFER , Met2 ; Отправить на очередной цикл

В $t = 0$ блок GENERATE сегмента 2 генерирует один транзакт (XN1=2), который входит в блок задержки ADVANCE.

В $t = 2$ первый транзакт (XN1=1) блока GENERATE сегмента 1 проходит первый и второй блоки GATE и занимает три канала из семи МКУ Sist. Обслуживание этого транзакта закончится в $t = 2 + 5 = 7$.

В $t = 4$ второй транзакт ($XN1=3$) также проходит оба блока GATE сегмента 1 и занимает следующие три канала МКУ Sist. Остается свободным один канал. Обслуживание второго транзакта закончится в $t = 4 + 5 = 9$.

В $t = 6$ третий транзакт ($XN1=4$) пройдет первый блок GATE сегмента 1 и войдет во второй блок GATE. Так как МКУ Sist имеет один свободный канал, т. е. не заполнено, условие во втором блоке GATE выполняется и транзакт пройдет к блоку ENTER. Но одного свободного канала недостаточно для удовлетворения запроса, требуется три, поэтому третий транзакт помещается в список задержки МКУ Sist.

В $t = 7$ МКУ Sist блоком SUNAVAIL переводится в недоступное состояние. В это же время закончится обслуживание первого транзакта, но третий транзакт не займет МКУ, а так и останется в списке задержки.

Так как в МКУ Sist находится один второй транзакт, т. е. оно не пусто, то транзакт, вызвавший перевод в недоступное состояние, будет задержан блоком GATE сегмента 2, потому что заданное в нем условие не выполняется.

В $t = 8$ четвертый транзакт ($XN1=5$) первым блоком GATE сегмента 1 вследствие недоступности МКУ Sist направляется к блоку TERMINATE с меткой Met1 и уничтожается.

В $t = 9$ закончится обслуживание второго транзакта. МКУ станет пусто, условие в блоке GATE сегмента 2 выполнится и сгенерировавший недоступность транзакт войдет в блок задержки ADVANCE.

В $t = 10$ пятый транзакт ($XN1=6$) первым блоком GATE сегмента 1 вследствие недоступности МКУ Sist направляется к блоку TERMINATE с меткой Met1 и уничтожается.

Также в $t = 9 + 1 = 10$ блок SAVAIL переведет МКУ в доступное состояние. Третий транзакт из списка задержки МКУ займет три канала МКУ. Обслуживание третьего транзакта ($XN1=4$) закончится в $t = 10 + 5 = 15$.

ЗАМЕЧАНИЕ

Как видно из примера, блок GATE позволяет только определить состояние незаполненности МКУ, т. е. наличие свободных каналов, но достаточно ли их для удовлетворения запроса он не определяет. Поэтому если в блоке ENTER операнд $B \neq 1$ и емкость МКУ не делится на значение операнда B без остатка, транзакт проходит через блок GATE и помещается в список задержки МКУ. При $B \neq 1$ и делении емкости МКУ на значение операнда B нацело или $B = 1$ список задержки МКУ всегда пуст.

5.3.2. Проверка состояния булевой переменной и блоком TEST

При проверке состояния МКУ блок TEST имеет тот же формат записи и те же условные операторы, что и при проверке состояния ОКУ.

В булевой переменной допускается использовать следующие логические операторы, связанные с МКУ:

- SF равен 1, если МКУ заполнено полностью, иначе — 0;
- SE равен 1, если МКУ пусто, иначе — 0;

- SV равен 1, если МКУ доступно, иначе — 0,
- а также СЧА МКУ:
- S — число занятых каналов МКУ;
- SA — среднее значение занятых каналов МКУ;
- SC — счетчик использования МКУ;
- SR — коэффициент использования МКУ;
- SM — максимальное число занятых каналов МКУ;
- ST — среднее время использования одного канала МКУ.

Как видно, SE, SF и SV можно использовать как в блоке GATE, так и в булевых переменных. Если же в блоке GATE в качестве условного оператора X указать, например, SA, то при трансляции будет ошибка «Недопустимое использование спецификатора класса СЧА». Если же в булевой переменной указать, например, SNV, также будет ошибка «Предполагается фактор выражения».

Пример 5.4. На трех параллельно работающих МКУ требуется организовать обслуживание так, чтобы их коэффициенты использования были одинаковыми.

; Пример 5.4

; Определение МКУ и булевых переменных

```

Use11    STORAGE    2                ; Емкость МКУ1
Use12    STORAGE    2                ; Емкость МКУ2
Use13    STORAGE    2                ; Емкость МКУ3
Prov1     BVARIABLE  BV$Prov2'AND'BV$Prov3'AND'BV$Prov4
Prov2     BVARIABLE  S$Use11'OR'(SR$Use11>400)
Prov3     BVARIABLE  S$Use12'OR'(SR$Use12>400)
Prov4     BVARIABLE  S$Use13'OR'(SR$Use13>400)
Prov5     BVARIABLE  BV$Prov2'AND'BV$Prov3

```

; Сегмент имитации МКУ1

```

GENERATE  (Exponential(310,0,2.3)) ; Источник транзактов
TEST E    BV$Prov1,0,Met1           ; Все МКУ заполнены?
TEST E    BV$Prov5,0,Met3           ; МКУ1 и МКУ2 заполнены?
TEST E    BV$Prov2,0,Met2           ; МКУ1 заполнено?
ENTER     Use11                     ; Занять МКУ1
ADVANCE   12,4,3                    ; Обслуживание МКУ1
LEAVE     Use11                     ; Освободить каналы МКУ1
TERMINATE                                ; Обслуженные транзакты МКУ1

```

; Сегмент имитации МКУ2

```

Met2      TEST E    BV$Prov3,0,Met3 ; МКУ2 заполнено?
          ENTER     Use12            ; Занять МКУ2
          ADVANCE   14.8,12.35       ; Обслуживание МКУ2
          LEAVE     Use12            ; Освободить каналы МКУ2
          TERMINATE                                ; Обслуженные транзакты МКУ2

```

; Сегмент имитации МКУ3

```

Met3      TEST E    BV$Prov4,0,Met1 ; МКУ3 заполнено?
          ENTER     Use13            ; Занять МКУ3

```

```
ADVANCE      (Exponential(310,0,22.3)) ; Обслуживание МКУ3
LEAVE        Usl13                      ; Освободить каналы МКУ3
TERMINATE    ; Обслуженные транзакты МКУ3
Met1         TERMINATE                  ; Потерянные транзакты
; Сегмент задания времени моделирования
GENERATE     3600
TERMINATE    1
```

При входе транзакта в первый блок TEST вычисляется булева переменная Prov1. Она равна 1 тогда, когда все три МКУ заполнены или их коэффициенты использования больше 0,4. В этом случае заданное в блоке TEST условие не выполняется и транзакт направляется к блоку TERMINATE с меткой Met1 для уничтожения.

Булева переменная Prov5 вычисляется при входе транзакта во второй блок TEST. Она равна 1, если МКУ Usl1 и Usl2 не имеют свободных каналов или их коэффициенты использования больше 0,4. При Prov5 = 1 условие во втором блоке TEST также не выполняется и транзакт направляется к блоку TEST с меткой Met3 на МКУ Usl3.

Если МКУ Usl3 не заполнено и его коэффициент использования меньше 0,4 (значение булевой переменной Prov4 равно 0), условие в блоке TEST выполняется, транзакт проходит к следующему блоку и занимает один из свободных каналов МКУ Usl3. При невыполнении условия в блоке TEST (значение булевой переменной Prov4 равно 1) транзакт направляется к блоку TERMINATE с меткой Met1 для уничтожения.

Если при входе транзакта в первый и второй блоки TEST булевы переменные Prov1 и Prov5 равны 0, транзакт войдет в третий блок TEST, который пропустит его для обслуживания на МКУ Usl1 в случае, если МКУ Usl1 имеет свободные каналы или его коэффициент использования меньше 0,4. В противном случае транзакт направляется к блоку TEST с меткой Met2.

Блок TEST с меткой Met2 проверяет наличие свободных каналов у МКУ Usl2 и величину его коэффициента использования. Если булева переменная Prov3 равна 0, транзакт пропускается к следующему блоку ENTER и занимает один из свободных каналов МКУ Usl2.

Если МКУ Usl2 не имеет свободных каналов или его коэффициент использования больше 0,4, транзакт направляется к блоку TEST с меткой Met3, где осуществляются аналогичные проверки состояния МКУ Usl3.

Приведем результаты моделирования.

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.
CPU1	2	1	0	1	2402	1	0.800	0.400
CPU2	2	2	0	1	1965	1	0.800	0.400
CPU3	2	2	0	1	1233	1	0.800	0.400

Все три МКУ имеют равные коэффициенты использования.

ЗАМЕЧАНИЕ

Использование булевой переменной и блока TEST не только расширяет возможности, но и позволяет сократить машинное время за счет того, например, что сразу выводятся из модели транзакты, для которых выполняются условия в булевой переменной Prov1. Для данного примера таких транзактов оказалось 65%.

Пример 5.5. В условиях примера 5.3 организуем такую дисциплину обслуживания, при которой в случае незаполненности МКУ, но недостаточности свободных каналов для удовлетворения запроса, транзакт не пропускается к блоку ENTER и, следовательно, не помещается в список задержки.

; Пример 5.5

; Определение МКУ и булевых переменных

```
Zap      EQU      3
Sist      STORAGE  7
Kont1     BARIABLE SV$Sist'AND'((7-$Sist)>=Zap))
Kont2     BARIABLE SE$Sist
```

; Сегмент имитации поступления и обслуживания транзактов

```
GENERATE  2,,,5      ; Источник транзактов
TEST E    BV$Kont1,1,Met1 ; МКУ свободно?
ENTER     Sist,3      ; Занять МКУ
ADVANCE   5           ; Обслуживание
LEAVE     Sist,3      ; Освободить каналы МКУ
TERMINATE 1           ; Обслуженные транзакты
Met1      TERMINATE 1 ; Потерянные транзакты
```

; Сегмент имитации недоступности

```
GENERATE  ,,,1      ; Транзакт-инициатор недоступности
Met2      ADVANCE   7 ; Время нахождения в состоянии недоступности
SUNAVAIL  Sist      ; Перевод в состояние доступности
TEST E    BV$Kont2,1 ; МКУ пусто?
ADVANCE   1         ; Восстановление доступности
SAVAIL    Sist      ; Перевод в состояние доступности
TRANSFER  ,Met2     ; Отправить на очередной цикл
```

В $t = 0$ блок GENERATE сегмента 2 генерирует один транзакт ($XN1=2$), который входит в блок задержки ADVANCE.

В $t = 2$ первый транзакт ($XN1=1$) блока GENERATE сегмента 1 входит блок TEST. Так как МКУ свободно и доступно, булева переменная $Kont1=1$, условие в блоке TEST выполняется, транзакт его проходит и занимает три канала из семи МКУ Sist. Обслуживание первого транзакта закончится в $t = 2 + 5 + 7$.

В $t = 4$ второй транзакт ($XN1=3$) также проходит блок TEST сегмента 1 и занимает следующие три канала МКУ Sist. Остается свободным один канал. Обслуживание второго транзакта закончится в $t = 4 + 5 = 9$.

В $t = 6$ третий транзакт ($XN1=4$) не пройдет блок TEST сегмента 1. Значение булевой переменной будет равно 0, так как МКУ Sist имеет один свободный канал. Но одного канала недостаточно для удовлетворения запроса, требуется

три, поэтому третий транзакт направляется в блок TERMINATE с меткой Met1 и уничтожается.

В $t = 7$ МКУ Sist блоком SUNAVAIL переводится в недоступное состояние. В это же время закончится обслуживание первого транзакта.

Так как в МКУ Sist находится один второй транзакт, т. е. оно не пусто, то транзакт, вызвавший перевод в недоступное состояние, будет задержан блоком TEST сегмента 2, потому что заданное в нем условие не выполняется (булева переменная Kont2 равна 0).

В $t = 8$ четвертый транзакт ($XN1 = 5$) блоком TEST сегмента 1 вследствие недоступности МКУ Sist (булева переменная Kont1 равна 0) направляется к блоку TERMINATE с меткой Met1 и уничтожается.

В $t = 9$ закончится обслуживание второго транзакта. МКУ станет пусто, условие в блоке TEST сегмента 2 выполнится и сгенерировавший недоступность транзакт войдет в блок задержки ADVANCE.

В $t = 10$ пятый транзакт ($XN1 = 6$) первым блоком GATE сегмента 1 вследствие недоступности МКУ Sist направляется к блоку TERMINATE с меткой Met1 и уничтожается.

Также в $t = 9 + 1 = 10$ блок SAVAIL переведет МКУ в доступное состояние. Транзакт, генерирующий недоступность и доступность, блоком TRANSFER будет направлен к блоку ADVANCE с меткой Met2.

Таким образом, в примере 5.3 обслуживаются три транзакта, а в данном примере — только два транзакта, так как третий транзакт из-за недостаточности свободной емкости не пропускается блоком TEST к блоку ENTER и, естественно, не помещается в список задержки МКУ.

5.4. Моделирование неисправностей многоканальных устройств

Ранее отмечалось, что осуществить имитацию выхода МКУ из строя, при котором все транзакты, находившиеся в МКУ на обслуживании, теряются, блоками SUNAVAIL и SAVAIL невозможно.

Рассмотрим, как это можно осуществить с применением блока DISPLACE.

Пример 5.6. Многоканальная СМО с отказами без очереди. СМО имеет конечную надежность. При выходе СМО из строя заявки, находившиеся на обслуживании, теряются. Новые заявки на обслуживание не принимаются.

; Пример 5.6

; Определение МКУ и булевых переменных

Emk	EQU	7
Zap	EQU	2
Sist	STORAGE	7
Kont1	BVARIABLE	SV\$Sist AND ((7-S\$Sist)>=Zap)
Kont2	BVARIABLE	SE\$Sist

```

; Сегмент имитации поступления и обслуживания заявок
      GENERATE      2,,,7      ; Источник заявок
Met12  TEST E      BV$Kont1,1,Met1 ; Есть ли место в МКУ?
      SAVEVALUE     KolPovt,(INT(Emk/Zap))
      ENTER         Sist,Zap    ; Занять МКУ
;      Учет номеров транзактов, занявших МКУ
      ASSIGN        KolPovt,X$KolPovt ; Запись в параметр цикла
Met5   TEST E      X*KolPovt,0,Met4 ; Есть ли в списке место?
      SAVEVALUE     P$KolPovt,XN1 ; Да, записать номер транзакта
      TRANSFER      ,Met6        ; Выйти из цикла
Met4   LOOP        KolPovt,Met5 ; Повторить или конец цикла
;
Met6   ADVANCE     5            ; Имитация обслуживания
      LEAVE         Sist,Zap    ; Освободить МКУ
;      Учет номеров транзактов, освободивших МКУ
      ASSIGN        KolPovt,X$KolPovt ; Запись в параметр цикла
Met8   TEST E      X*KolPovt,XN1,Met7 ; Есть ли этот номер транзакта?
      SAVEVALUE     P$KolPovt,0 ; Да, тогда удалить
      TRANSFER      ,Met9        ; Выйти из цикла
Met7   LOOP        KolPovt,Met8 ; Повторить или конец цикла
;
Met9   TERMINATE   1            ; Обслуженные заявки
Met1   TERMINATE   1            ; Потерянные заявки
; Сегмент имитации неисправностей
      GENERATE      ,,,,1
Met2   ADVANCE     7.5          ; Интервал отказов
      SUNAVAIL      Sist        ; Перевод в недоступность
;      Удаление транзактов из МКУ
      ASSIGN        KolPovt,X$KolPovt ; Запись в параметр цикла
Met10  TEST NE     X*KolPovt,0,Met11 ; Есть ли транзакт в МКУ
      DISPLACE      X*KolPovt,Met3 ; Да, тогда удалить
      SAVEVALUE     P$KolPovt,0 ; Снять с учета
Met11  LOOP        KolPovt,Met10 ; Повторить или конец цикла
;
      TEST E      BV$Kont2,1 ; МКУ пуст?
      ADVANCE     1            ; Да, тогда ремонтируем
      SAVAIL      Sist        ; Перевод в доступность
      TRANSFER      ,Met2
Met3   LEAVE       Sist,2      ; Освободить МКУ
      TERMINATE   1            ; Удаленные из МКУ транзакты

```

Блок GENERATE с интервалами две единицы модельного времени генерирует семь транзактов. Следующий за ним блок TEST проверяет МКУ Sist на исправность (доступность) и наличия в нем свободных каналов, достаточных для удовлетворения запроса. Если булева переменная Kont1 равна 1, транзакт пропускается и занимает МКУ Sist. Но перед этим вычисляется и заносится целое

число $INT(Emk/Zap) = INT(7/2) = 3$ в сохраняемую ячейку с именем KolPovt, которое определяет, сколько транзактов может одновременно находиться в МКУ.

После выхода транзакта из блока ENTER начинает работать сегмент учета номеров транзактов, занявших МКУ. Блоком ASSIGN в параметр с именем KolPovt — параметр цикла заносится число, находящееся в сохраняемой ячейке с именем KolPovt. Далее в цикле, тело которого начинается с блока TEST с меткой Met5 и заканчивается блоком LOOP с меткой Met4, находится свободное место в списке для записи номера транзакта. Свободное место определяется блоком TEST как равенство нулю значения какой-либо одной из трех сохраняемых ячеек X1, X2 или X3 (по числу транзактов, одновременно находящихся в МКУ). Такая ячейка всегда есть и блоком SAVEVALUE в нее записывается номер занявшего МКУ транзакта. А так как сохраняемых ячеек, значения которых равны нулю, может быть несколько (особенно вначале работы модели), после записи номера транзакта осуществляется выход из цикла.

Транзакт направляется к блоку ADVANCE с меткой Met6 и входит в него. Имитируется обслуживание заявки. После обслуживания транзакт освобождает МКУ Sist, пройдя блок LEAVE.

Начинает работать сегмент учета номеров транзактов, освободивших МКУ. Его работа аналогична рассмотренной ранее работе сегмента учета номеров транзактов, занявших МКУ. Отличие состоит в том, что отыскивается сохраняемая ячейка X1, X2 или X3, которая содержит номер обслуженного транзакта. Найденная ячейка обнуляется, т. е. обслуженный транзакт снимается с учета. Так как номер записывается только в одну ячейку, то после ее обнуления осуществляется выход из цикла, тело которого начинается с блока TEST с меткой Met8 и заканчивается блоком LOOP с меткой Met7.

Теперь рассмотрим работу модели при возникновении неисправности МКУ. Транзакт, инициирующий неисправность, после задержки блоком ADVANCE сегмента имитации неисправностей, входит в блок SUNAVAIL, который переводит МКУ Sist в недоступное (неисправное) состояние.

Далее начинается работа сегмента удаления транзактов из МКУ. Суть ее заключается в следующем. В цикле, также организованном с помощью блока LOOP, просматривается список номеров транзактов, занявших МКУ. Если значение какой-либо сохраняемой ячейки X1, X2 или X3 не равно нулю, значит, в ней записан номер транзакта, находящегося в данный момент в МКУ. Блоком DISPACE этот транзакт перемещается к блоку LEAVE с меткой Met3, освобождает МКУ и выводится из модели.

После удаления из МКУ всех транзактов, транзакт-инициатор неисправности входит в следующий за блоком LOOP с меткой Met11 блок TEST. Так как Sist пусто (булева переменная Kont2 равна нулю), блок TEST пропускает его и начинается имитация восстановления работоспособности МКУ.

Результаты моделирования представлены в табл. 5.1. Из результатов следует, что обслужены только две заявки (первый и шестой транзакты). Пять заявок не обслужены: пятый транзакт поступил, когда МКУ было неисправным,

а третий и четвертый, седьмой и восьмой транзакты потеряны: в моменты возникновения неисправностей ($t = 7,5$ и $t = 16$ соответственно) они находились в МКУ.

Коэффициент использования (Util.) МКУ Sist равен 0,375. Он рассчитывается по формуле:

$$k = \frac{\sum_{i=1}^n \alpha_i \cdot \Delta t_i}{T \cdot E}, \quad (5.1)$$

где n — число транзактов, занимавших МКУ за время T моделирования;

E — число каналов МКУ (емкость);

α_i — число каналов, занимаемых i -м транзактом;

Δt_i — промежуток времени, в течение которого i -м транзактом было занято α_i каналов, рассчитывается как

$$\Delta t_i = t_{2i} - t_{1i}, \quad (5.2)$$

t_{1i} и t_{2i} — абсолютное модельное время занятия и освобождения соответственно α_i каналов МКУ.

Таблица 5.1

Результаты моделирования примера 5.6

События	Транзакты						
	1	3	4	5	6	7	8
Вход в модель	2	4	6	8	10	12	14
Занятие МКУ	2	4	6		10	12	14
В МКУ транзакты	1	1, 3	1, 3, 4		6	6, 7	6, 7, 8
Удаление		7.5	7.5			16	16
Освобождение МКУ	7	7.5	7.5		15	16	16
Вывод из модели	7	7.5	7.5	8	15	16	16

Для рассматриваемого примера 5.6 (см. табл. 5.1) имеем:

$$E = 7, T = 16, \alpha = 2;$$

$$\Delta t_1 = 7 - 2 = 5, \Delta t_2 = 7,5 - 4 = 3,5; \Delta t_3 = 7,5 - 6 = 1,5; \Delta t_4 = 0;$$

$$\Delta t_5 = 15 - 10 = 5; \Delta t_6 = 16 - 12 = 4; \Delta t_7 = 16 - 14 = 2;$$

$$k_1 = \frac{2(5+3,5+1,5+0+5+4+2)}{16 \cdot 7} = \frac{2 \cdot 21}{16 \cdot 7} = \frac{3}{8} = 0,375.$$

Таким образом, коэффициент использования МКУ рассчитан с учетом времени занятия его и теми транзактами, обслуживание которых было прервано.

Если в результате моделирования требуется определить коэффициент использования МКУ без учета удаленных из него транзактов, т. е. транзактов, обслуживание которых было прервано, пользователь может это сделать самостоятельно. Для этого нужно знать, сколько заявок будет обслужено полностью. Введем в модель следующие строки:

```
KIsp      FVARIABLE      (Zap#N$Met9#5)/(Emk#16)
          SAVEVALUE      KIsp,V$KIsp
```

Получим

$$k_2 = \frac{2 \cdot (5+5)}{16 \cdot 7} = \frac{5}{28} = 0,179.$$

Пример 5.7. Многоканальная СМО с ограниченной очередью обладает конечной надежностью. При выходе из строя обслуживание находящихся в каналах заявок прекращается, но они не теряются, а после восстановления работоспособности обслуживаются сначала. Новые заявки принимаются, если в очереди есть место.

Так как заявки, находящиеся на обслуживании, при выходе СМО из строя не должны теряться, то их нужно сохранять в очереди до тех пор, пока они не будут обслужены.

Модель функционирования такой СМО представлена ниже.

; Пример 5.7

; Определение МКУ, арифметических и булевых переменных

```
Emk      EQU      7      ; Число каналов МКУ
Zap      EQU      2      ; Число занимаемых одним транзактом каналов
Smo      STORAGE  7      ; Задание числа каналов МКУ
Dlina    EQU      3      ; Длина очереди
Kont1     BVARIABLE (Dlina-($Smo/Zap))>0
Kont2     BVARIABLE SV$Smo'AND'(R$Smo>Zap)
Kont3     BVARIABLE SV$Smo
Rasm      VARIABLE Dlina+1+P$KolTp
KIsp      VARIABLE (N$Met18#7#Zap)/(Emk#19)
```

; Сегмент имитации поступления и обслуживания заявок

```
GENERATE  2,,7      ; Источник заявок
TEST E    BV$Kont1,1,Met1      ; Есть ли место в очереди?
SAVEVALUE KolTp,(INT(Emk/Zap)) ; Расчет числа повторений

;      Учет номеров транзактов, вставших в очередь
SSIGN     Dlina,Dlina      ; Запись в параметр цикла
Met4      TEST E    X*Dlina,0,Met5      ; Есть ли в списке место?
SAVEVALUE P$Dlina,XN1      ; Да, записать номер транзакта
TRANSFER  ,Met6      ; Выйти из цикла
Met5      LOOP     Dlina,Met4      ; Повторить или конец цикла
;
Met6      TEST E    BV$Kont2,0,Met3      ; Есть ли в очереди место?
```

```

LINK      Nakop, LIFO      ; Да, поставить транзакт в очередь
Met3      ENTER      Smo, Zap      ; Занять МКУ
;      Учет номеров транзактов, занявших МКУ
ASSIGN    KolTp, X$KolTp      ; Запись в параметр цикла
Met8      ASSIGN    1, V$Rasm      ; Расчет индекса
TEST E    X*1, 0, Met7      ; Есть ли в списке место?
SAVEVALUE P1, XN1      ; Да, записать номер транзакта
TRANSFER  , Met9      ; Выйти из цикла
Met7      LOOP      KolTp, Met8      ; Повторить или конец цикла
;
Met9      ADVANCE    7      ; Имитация обслуживания
LEAVE     Smo, Zap      ; Освободить МКУ
;      Учет номеров транзактов, освободивших МКУ
ASSIGN    KolTp, X$KolTp      ; Запись в параметр цикла
Met10     ASSIGN    1, V$Rasm      ; Расчет индекса
TEST E    X*1, XN1, Met11      ; Есть ли этот номер транзакта?
SAVEVALUE P1, 0      ; Да, удалить
TRANSFER  , Met12      ; Выйти из цикла
Met11     LOOP      KolTp, Met10      ; Повторить или конец цикла
;      Учет номеров транзактов, покинувших очередь
Met12     ASSIGN    Dlina, Dlina      ; Запись в параметр цикла
Met15     TEST E    X*Dlina, XN1, Met16 ; Есть ли этот номер транзакта?
SAVEVALUE P$Dlina, 0      ; Да, тогда удалить
TRANSFER  , Met17      ; Выйти из цикла
Met16     LOOP      Dlina, Met15      ; Повторить или конец цикла
;
Met17     UNLINK    Nakop, Met3, 1      ; Одна заявка из очереди в МКУ
TERMINATE      ; Обслуженные заявки
Met1      TERMINATE      ; Потерянные заявки
;      Сегмент имитации неисправностей
GENERATE  , , , 1
Met20     ADVANCE    9.5      ; Имитация времени между отказами
SUNAVAIL  Smo      ; Перевод в недоступность
;      Удаление транзактов из МКУ
SAVEVALUE UdTp, 0      ; Обнуление счетчика
ASSIGN    KolTp, X$KolTp      ; Запись в параметр цикла
Met21     ASSIGN    1, V$Rasm      ; Расчет индекса
TEST NE   X*1, 0, Met22      ; Есть ли транзакт в МКУ?
DISPLACE  X*1, Met23      ; Да, тогда удалить
SAVEVALUE P1, 0      ; Снять с учета
Met22     LOOP      KolTp, Met21      ; Повторить или конец цикла
;
TEST E    SE$Smo, 1      ; МКУ пусто?
ADVANCE   1      ; Да, тогда ремонт МКУ

```

	SAVAIL	Smo	; Перевод в доступность
	TEST NE	X\$UdTp, X\$KolTp, Met20	; Выводить из Накор?
	ASSIGN	1, (X\$KolTp - X\$UdTp)	; Да, записать в параметр цикла
Met24	UNLINK	Nakor, Met3, 1	; Одна заявка из очереди в МКУ
	LOOP	1, Met24	; Повторить или конец цикла
	TRANSFER	, Met20	
Met23	LEAVE	Smo, Zap	; Освободить МКУ
	SAVEVALUE	UdTp+, 1	; Счет удаленных транзактов
	TEST E	BV\$Kont3, 1	; Исправно ли МКУ?
	TRANSFER	, Met3	; Да, удаленные транзакты в МКУ
; Сегмент задания времени моделирования			
	GENERATE	19	
	SAVEVALUE	KIsp, V\$KIsp	; Расчет коэффициента использования
	TERMINATE	1	

Блок TEST, стоящий сразу после блока GENERATE сегмента имитации поступления и обслуживания заявок, проверяет наличие свободного места в очереди. При отсутствии свободного места транзакт направляется к блоку TERMINATE с меткой Met1 и уничтожается. Если свободное место есть (булева переменная Kont1 равна 1), учитывается номер транзакта и сохраняется в одной из ячеек X1...X3 (число ячеек равно длине очереди).

После этого, если СМО неисправно и свободное число каналов не может удовлетворить запрос (булева переменная Kont2 равна 0), блок TEST с меткой Met6 пропускает транзакт. Транзакт помещается блоком LINK в список пользователя с именем Накор. В противном случае булева переменная Kont2 равна 1, транзакт направляется к блоку ENTER с меткой Met3 и занимает Zap каналов МКУ Smo.

При дальнейшем продвижении транзакта до блока ADVANCE с меткой Met9 учитывается его номер, как занявшего МКУ.

После обслуживания транзакт освобождает Zap каналов МКУ. Его номер снимается с учета номеров транзактов, занимающих МКУ и находящихся в очереди. При входе транзакта в блок UNLINK с меткой Met17 (этот транзакт является выводящим) из списка пользователя с именем Накор выводится очередной транзакт и направляется к блоку ENTER с меткой Met3. Поскольку в МКУ есть свободные каналы для удовлетворения запроса, транзакт занимает их. Выводящий, он же обслуженный транзакт, входит в блок TERMINATE с меткой Met18 и уничтожается.

При возникновении неисправности транзакт сегмента имитации неисправностей войдет в блок SUNAVAIL и переведет МКУ Smo в недоступное состояние. Проходя далее через последовательность блоков до блока LOOP с меткой Met22, транзакт вызывает их срабатывание, вследствие чего все транзакты, которые находятся в это время в МКУ, удаляются из него и направляются к блоку LEAVE с меткой Met23. Удаленные транзакты проходят этот блок и освобождают МКУ Smo. Но так как МКУ недоступно, то блок TEST не пропускает дальше удаленные из МКУ транзакты.

В то же время, так как МКУ стало пустым, блок TEST пропускает транзакт-инициатор неисправности к блоку ADVANCE. Начинается имитация ремонта МКУ Smo. По истечении необходимого для этого времени МКУ Smo блоком SAVAIL переводится в доступное состояние.

Как только МКУ Smo стало доступным, удаленные из МКУ транзакты пропускаются блоком TEST (булева переменная Kont3 равна 1), направляются блоком TRANSFER к блоку ENTER с меткой Met3, подсчитываются блоком SAVEVALUE и занимают МКУ Smo. Обслуживание этих транзактов начинается сначала.

Во время недоступности МКУ при наличии места в очереди могут приниматься новые заявки, которые помещаются в список пользователя с именем Накор. МКУ после восстановления доступности может иметь свободные каналы. Поэтому из списка пользователя нужно вывести, если есть, транзакты и направить их в МКУ. Если количество удаленных транзактов равно количеству транзактов, которые одновременно могут находиться в МКУ, то после восстановления доступности в МКУ нет свободных каналов для удовлетворения запроса. Вывод из списка пользователя не производится. В противном случае транзакты из списка пользователя выводятся. В качестве выводящего транзакта используется транзакт-инициатор неисправности.

Результаты пошагового моделирования сведены в табл. 5.2.

Таблица 5.2

Результаты моделирования примера 5.7

События	Транзакты						
	1(1)*	2(4)	3(5)	4(6)	5(7)	6(8)	7(9)
Вход в модель	2	4	6	8	10	12	14
В очереди транзакты	1	1, 4	1, 4, 5	1, 4, 5	4, 5, 7	4, 5, 7	4, 5, 7
Занятие МКУ	2	4/10,5	6/10,5		10,5	17,5	17,5
Удаление транзактов		9,5	9,5				
Освобождение МКУ	9	9,5/ 17,5	9,5/ 17,5		17,5		
Вывод из модели	9	17,5	17,5	8	10	12	14

* *Примечание.* Даны порядковые номера транзактов, которые должны обслуживаться МКУ Smo. В скобках даны номера этих же транзактов, присвоенные системой GPSS. Номера два (XN1=2) и три (XN1=3) имеют транзакты блоков GENERATE сегмента имитации неисправностей и сегмента задания времени моделирования соответственно.

За 19 единиц модельного времени обслужены четыре транзакта (первый, четвертый, пятый и седьмой). Обслуживание четвертого и пятого транзактов было прервано в $t = 9,5$ и они обслуживались повторно с $t = 10,5$ до $t = 17,5$. Седьмой транзакт обслуживался одновременно с ними.

Шестой транзакт был удален, так как в $t = 8$ в очереди находились первый, четвертый и пятый транзакты, т. е. в очереди не было места.

В $t = 9$ закончилось обслуживание первого транзакта и в очереди остались два транзакта. В $t = 9,5$ МКУ стало неисправным. В $t = 10$ поступил седьмой транзакт. В очереди было одно свободное место и седьмой транзакт поместили в список пользователя с именем Накор. В очереди стало три транзакта (четвертый, пятый и седьмой). Поэтому из-за отсутствия свободных мест в очереди восьмой и девятый транзакты были удалены в $t = 12$ и $t = 14$ соответственно.

Седьмой транзакт был помещен в список пользователя с именем Накор в $t = 10$. Когда в $t = 10,5$ МКУ стало исправным, его заняли только два удаленные ранее транзакта (четвертый и пятый). Вывод седьмого транзакта из списка пользователя осуществляется в цикле транзактом-инициатором неисправности. Число повторений цикла равно разности между количеством транзактов, которое может быть одновременно в МКУ, и количеством удаленных из МКУ транзактов.

Коэффициент использования МКУ S_{mo} равен 0,556. Системой GPSS он рассчитывался по формуле:

$$k = \frac{\sum_{i=1}^n \sum_{j=1}^m \alpha_i \cdot \Delta t_{ij}}{T \cdot E}, \quad (5.3)$$

$$\Delta t_{ij} = \Delta t_{2ij} - \Delta t_{1ij}, \quad (5.4)$$

Δt_{1ij} и Δt_{2ij} — абсолютное модельное время j -го занятия и освобождения МКУ соответственно i -м транзактом.

Для рассматриваемого примера 5.7 (см. табл. 5.2) получим:

$$E = 7, \quad T = 16, \quad \alpha = 2;$$

$$\Delta t_{11} = 9 - 2 = 7;$$

$$\Delta t_{21} = 9,5 - 4 = 5,5; \quad \Delta t_{22} = 17,5 - 10,5 = 7;$$

$$\Delta t_{31} = 9,5 - 6 = 3,5; \quad \Delta t_{32} = 17,5 - 10,5 = 7;$$

$$\Delta t_{41} = 0; \quad \Delta t_{51} = 17,5 - 10,5 = 7; \quad \Delta t_{61} = 0; \quad \Delta t_{71} = 0;$$

$$k_1 = \frac{2[7 + (5,5 + 7) + (3,5 + 7) + 7]}{7 \cdot 19} = \frac{74}{133} = 0,556.$$

Коэффициент использования МКУ без учета времени незавершенного обслуживания составит:

$$k_2 = \frac{2 \cdot 4 \cdot 7}{7 \cdot 19} = \frac{8}{19} = 0,421.$$

Увеличим длину очереди на одну заявку. В этом случае шестой транзакт не должен быть удален, поскольку в очереди будет одно место. Коэффициент k_2

использования МКУ останется неизменным, а k_1 увеличится, так как $\Delta t_{s1} = 17,5 - 10,5 = 7$, а не нуль:

$$k_1 = \frac{2[7 + (5,5 + 7) + (3,5 + 7) + 7 + 0,5 + 1,5]}{7 \cdot 19} = \frac{78}{133} = 0,586.$$

Рассмотрены методы моделирования неисправностей МКУ. Их применение позволит разработчику строить интересующие его модели МКУ с отказами.

5.5. Моделирование переключателей

Для моделирования такого оборудования, как переключатели, имеющие только два состояния, в GPSS используются логические ключи. Логический ключ может находиться в одном из двух состояний:

- включен (ON или 1);
- выключен (OFF или 0).

В зависимости от состояния ключа изменяется направление движения транзактов.

Логический ключ моделируется блоком LOGIC. Формат блока:

LOGIC X A

Операнд A — имя или номер логического ключа. Может быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр. Логический оператор X — состояние логического ключа устанавливается в зависимости от следующих его значений:

- S — логический ключ, заданный операндом A, включается;
- R — логический ключ, заданный операндом A, выключается;
- I — логический ключ инвертируется, т. е. состояние его меняется на противоположное, например, если был включен, будет выключен.

5.5.1. Проверка состояния логического ключа блоком GATE

Блок GATE имеет такой же формат, как и при проверке состояний ОКУ и МКУ, и такие же два режима работы:

GATE X A, [B]

Операнд A — имя или номер проверяемого ключа. Может быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр.

Операнд B — метка блока, к которому будет направлен транзакт в случае невыполнения условия, заданного условным оператором X.

Условный оператор может принимать значения:

- LS — равен 1, если логический ключ, заданный операндом A, включен;
- 0 — если выключен;

○ LR — равен 1, если логический ключ, заданный операндом A, выключен;
0 — если включен.

Пример 5.8. В условиях примера 5.3 используем логический ключ для организации такой же дисциплины обслуживания, как и с блоками SUNAVAIL и SAVAIL.

; Пример 5.8

; Определение МКУ

Sist STORAGE 7

; Сегмент имитации поступления и обслуживания транзактов

GENERATE 2,,,5 ; Источник транзактов

GATE SV Klu, Met1 ; Включен ключ

GATE SNF Sist, Met1 ; МКУ свободно?

ENTER Sist, 3 ; Занять МКУ

ADVANCE 5 ; Обслуживание

LEAVE Sist, 3 ; Освободить каналы МКУ

TERMINATE 1 ; Обслуженные транзакты

Met1 TERMINATE 1 ; Потерянные транзакты

; Сегмент имитации недоступности

GENERATE,,,1 ; Транзакт-инициатор недоступности

LOGIC S Klu ; Включить ключ

Met2 ADVANCE 7 ; Время нахождения в доступном состоянии

LOGIC R Klu ; Выключить ключ

GATE SE Sist ; МКУ пусто

ADVANCE 1 ; Восстановление доступности

LOGIC S Klu ; Включить ключ

TRANSFER , Met2 ; Направить на очередной цикл

В $t = 0$ блок GENERATE сегмента имитации поступления и обслуживания транзактов генерирует один транзакт ($XN1=1$) и перестает быть активным. Этот транзакт входит в блок LOGIC, который включает ключ Klu, и далее задерживается блоком ADVANCE.

Блок GENERATE сегмента имитации поступления и обслуживания транзактов генерирует с интервалом 2 единицы модельного времени пять транзактов и также перестает быть активным.

В $t = 2$ первый транзакт ($XN1=2$) проходит первый и второй блоки GATE, так как ключ Klu включен и МКУ свободно, и занимает три свободных канала. Обслуживание этого транзакта закончится в $t = 2 + 5 = 7$.

Второй транзакт ($XN1=3$) в $t = 4$ также пройдет оба блока GATE и займет следующие три канала МКУ Sist. Свободным останется один канал. Обслуживание второго транзакта закончится в $t = 4 + 5 = 9$.

В $t = 6$ третий транзакт пройдет первый блок GATE сегмента имитации поступления и обслуживания транзактов и войдет во второй блок GATE. Поскольку МКУ Sist имеет один канал свободным, условие, заданное в блоке GATE, выполнится и транзакт пройдет к блоку ENTER. Но одного канала недостаточно

для удовлетворения запроса, поэтому транзакт помещается в список задержки МКУ.

В $t = 7$ ключ Klu будет выключен и МКУ будет закрыто для вновь поступающих транзактов. В это же время закончится обслуживание первого транзакта и третий транзакт из списка задержки займет МКУ.

Так как в МКУ *Sist* находятся второй и третий транзакты, т. е. оно не пусто, то транзакт, вызвавший выключение ключа, будет задержан блоком GATE сегмента имитации недоступности, потому что заданное в нем условие не выполняется.

В $t = 8$ четвертый транзакт ($XN1=5$) первым блоком GATE сегмента имитации поступления и обслуживания транзактов, так как ключ Klu выключен, направляется к блоку TERMINATE с меткой Met1 и уничтожается.

В $t = 9$ закончится обслуживание второго транзакта. В МКУ останется один транзакт, условие в блоке GATE сегмента имитации недоступности снова не выполнится и вызвавший выключение ключа транзакт не войдет в блок задержки ADVANCE.

В $t = 10$ пятый транзакт ($XN1=6$) первым блоком GATE сегмента 1 также направляется к блоку TERMINATE с меткой Met1 и уничтожается.

В $t = 7 + 5 = 12$ закончится обслуживание третьего транзакта. Транзакт, вызвавший выключение ключа, пройдет через блок GATE к блоку задержки ADVANCE и в $t = 12 + 1 = 13$ будет включен ключ Klu . Транзактам откроется путь на МКУ *Sist*.

Таким образом, в данном примере будут обслужены три транзакта (первый, второй и третий), в примере 5.3 — также три транзакта (первый, второй и третий). При этом время окончания обслуживания составит 12 и 15 единиц модельного времени соответственно, т. е. при использовании логического ключа время обслуживания первых трех транзактов заканчивается раньше.

ЗАМЕЧАНИЕ

Если до поступления в модель первого транзакта от блока GENERATE сегмента имитации поступления и обслуживания транзактов не включить ключ Klu , то модель работать будет неверно, так как все транзакты будут направляться к блоку TERMINATE с меткой Met1 и уничтожаться. Ключ Klu будет включен лишь после первого перевода из недоступного состояния в доступное. Поэтому в сегменте имитации недоступности предусмотрен блок LOGIC, включающий в $t = 0$ ключ Klu .

5.5.2. Проверка состояния логического ключа блоком TEST и булевой переменной

При проверке состояния логического ключа блок TEST имеет такой же формат записи и те же условные операторы, что и при проверке состояний ОКУ и МКУ.

В булевой переменной используется только СЧА LS. Если в булевой переменной записать LR, возникнет ошибка «Предполагается фактор выражения».

Пример 5.9. МКУ переводится в недоступное состояние через случайные интервалы времени. Организуем две дисциплины обслуживания и исследуем как они влияют, например, на вероятность обслуживания запросов.

Для организации недоступности МКУ при первой дисциплине обслуживания используем логический ключ и блок GATE, а не блоки SUNAVAIL и SAVAIL. Так как емкость МКУ Sist равна 8 и не делится на 3 без остатка, то блок GATE будет пропускать транзакт при недостаточной для удовлетворения запроса свободной емкости МКУ. Восстановление доступности МКУ начинается после того, как МКУ будет пусто.

; Пример 5.9. Вариант 1

; Определение МКУ и арифметической переменной

```
Zap      EQU      3
Sist      STORAGE  8
Ver       FVARIABLE N$Per/N$Vse
```

; Сегмент имитации поступления и обслуживания запросов

```
GENERATE  (Exponential(310,0,10.3)) ; Источник запросов
Vse       GATE LS   Klu, Met1        ; Ключ включен?
          GATE SNF  Sist, Met1       ; МКУ свободно
          ENTER     Sist, Zap        ; Занять МКУ
          ADVANCE   (Normal(231,25.3,1.7)) ; Обслуживание
          LEAVE     Sist, Zap        ; Освободить каналы МКУ
Per       TERMINATE                                ; Обслуженные запросы
```

; Сегмент имитации недоступности

```
GENERATE  ,, 1 ; Транзакт-инициатор недоступности
LOGIC S   Klu ; Включить в ключ
Met2     ADVANCE (Exponential(11,0,600.7)); Время доступности
LOGIC R   Klu ; Выключить ключ
          GATE SE  Sist ; МКУ пусто?
          ADVANCE 3.2, 1.3 ; Восстановление доступности
LOGIC S   Klu ; Включить ключ
TRANSFER , Met2 ; Направить на очередной цикл
Met1     TERMINATE
```

; Сегмент задания времени моделирования

```
GENERATE  3600
SAVEVALUE Ver, V$Ver ; Расчет вероятности обслуживания
TERMINATE 1
```

Блок GATE сегмента имитации поступления и обслуживания транзактов, стоящий перед блоком ENTER, проверяет состояние ключа Klu. Если ключ включен, транзакт проходит ко второму блоку GATE. Этот блок определяет наличие свободных каналов в МКУ Sist. Если свободные каналы имеются, транзакт проходит к следующему блоку ENTER и занимает три из восьми каналов МКУ Sist. Если условия в обоих блоках GATE не выполняются, транзакт направляется к блоку TERMINATE с меткой Met1 для уничтожения.

Перевод МКУ Sist в недоступное состояние производится блоками сегмента имитации недоступности. Единственный транзакт, сгенерированный блоком GENERATE сегмента 2, после включения ключа Klu и задержки поступает в блок

LOGIC, который переводит логический ключ Klu в состояние выключено. Далее на пути транзакта находится блок GATE. Если MKY Sist пусто, транзакт проходит к блоку ADVANCE и начинается имитация восстановления доступности, например, технического обслуживания. В противном случае транзакт ожидает завершения обслуживания транзактов, находившихся в MKY Sist к моменту перевода его в недоступное состояние.

После включения ключа Klu транзакт блоком TRANSFER направляется к блоку ADVANCE с меткой Met2 и процесс имитации доступности и недоступности MKY Sist повторяется.

Приведем результаты моделирования.

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.
SIST	8	2	0	6	8049	1	5.658	0.707

SAVEVALUE	RETRY	VALUE
VER	0	0.759

Вторая дисциплина обслуживания предусматривает, что в случае незаполненности MKY, но недостаточности свободной емкости для удовлетворения запроса, транзакт не помещается в список задержки MKY. Для организации второй дисциплины обслуживания используем логический ключ, булеву переменную и блок TEST.

; Пример 5.9. Вариант 2

; Определение MKY, арифметической и булевых переменных

```
Zap      EQU      3
Sist      STORAGE  8
Ver       VARIABLE N$Per/N$Vse
Kont1     BVARIABLE LS$Klu'AND'((8-S$Sist)>=Zap)
Kont2     BVARIABLE SE$Sist
```

; Сегмент имитации поступления и обслуживания запросов

```
GENERATE  (Exponential(310,0,10.3)) ; Источник запросов
Vse       TEST E   BV$Kont1,1,Met1   ; Доступно и есть каналы MKY
          ENTER    Sist,Zap          ; Занять MKY
          ADVANCE  (Normal(231,25.3,1.7)) ; Обслуживание
          LEAVE    Sist,Zap          ; Освободить каналы MKY
Per       TERMINATE ; Обслуженные запросы
```

; Сегмент имитации недоступности

```
GENERATE  ,,1 ; Транзакт-инициатор недоступности
LOGIC S   Klu ; Включить ключ
Met2     ADVANCE (Exponential(11,0,600.7)); Время доступности
LOGIC R   Klu ; Выключить ключ
TEST E    BV$Kont2,1 ; MKY пусто?
ADVANCE   3.2,1.3 ; Восстановление доступности
LOGIC S   Klu ; Включить ключ
TRANSFER  ,Met2 ; Направить на очередной цикл
Met1     TERMINATE ; Обслуженные запросы
```

; Сегмент задания времени моделирования

GENERATE 3600

SAVEVALUE Ver, V\$Ver

; Расчет вероятности

TERMINATE 1

Блок TEST сегмента 1 пропускает транзакт на МКУ Sist в случае, если булева переменная Kont1 равна 1. Это будет тогда, когда ключ Klu включен и когда оставшейся свободной емкости МКУ достаточно для удовлетворения запроса.

Блок TEST сегмента 2 пропускает транзакт к следующему блоку задержки ADVANCE, если булева переменная Kont2 равна 1, т. е. когда МКУ Sist пусто.

В остальном работа модели аналогична. Приведем результаты моделирования.

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.
SIST	8	5	0	6	5529	1	3.891	0.486

SAVEVALUE	RETRY	VALUE
VER	0	0.521

Из сравнения полученных результатов видно, что при одних и тех же данных и второй дисциплине обслуживания коэффициент использования МКУ и вероятность обслуживания уменьшились в 1,52 раза.

5.6. Примеры построения моделей систем с многоканальным устройством обслуживания

Пример 5.10. Пункты наблюдения (ПН) ведут мониторинг средств связи (СС). По мере сбора данных с интервалом времени $T1 \pm T2$ мин ПН передают их для обработки на ЭВМ. На обработку затрачивается $T3 \pm T4$ мин машинного времени. При этом ЭВМ выполняет другие заявки, поступающие с интервалом времени $T5 \pm T6$ мин. Эти заявки обрабатываются в течение времени $T7 \pm T8$ мин. ПН приступают к ведению мониторинга СС практически одновременно.

Построить модель мониторинга СС в течение 10 часов с целью определения среднего времени одного этапа мониторинга одним ПН.

Вариант данных, необходимых для разработки модели, примем следующий: $T1 = 30$ мин, $T2 = 5$ мин, $T3 = 4$ мин, $T4 = 2$ мин, $T5 = 15$ мин, $T6 = 10$ мин, $T7 = 10$ мин, $T8 = 4$ мин, количество ПН — 15.

В данном случае есть два вида ограничивающих условия. Во-первых, имеется лишь одна ЭВМ. Во-вторых, имеется фиксированное число ПН. Тогда ЭВМ — одноканальное устройство, а ПН — транзакт. В реальной системе после того как ПН обработает данные на ЭВМ, он возвращается к сбору данных. В модели этот возврат может быть обеспечен блоком TRANSFER. Для того, чтобы ограничить число транзактов, циркулирующих в модели, числом, равным числу ПН, необходимо использовать операнд D блока GENERATE. Для имитации сбора данных целесообразно использовать МКУ с емкостью, равной числу ПН.

Программа модели и результаты моделирования после 5 000 прогонов приведены ниже.

; Пример 5.10

TR1 STORAGE 15 ; Количество ПН

; Сегмент имитации мониторинга

Rasv GENERATE ,,,15 ; Количество транзактов = количеству ПН
 ASSIGN 1,AC1 ; Время начала этапа мониторинга
 ENTER Tr1 ; Занятие МКУ – начало этапа сбора данных
 ADVANCE 30,5 ; Сбор данных
 LEAVE Tr1 ; Окончание этапа сбора данных
 QUEUE Dlina ; Встать в общую очередь заявок на ЭВМ
 QUEUE Dlina1 ; Встать в очередь передачи данных
 SEIZE Ewm ; Занять ЭВМ
 DEPART Dlina1 ; Покинуть очередь передачи данных
 DEPART Dlina ; Покинуть общую очередь заявок на ЭВМ
 ADVANCE 4,2 ; Передача данных
 RELEASE Ewm ; Освободить ЭВМ
 ASSIGN 2,AC1 ; Время конца этапа мониторинга
 SAVEVALUE 1+,(P2-P1) ; Суммарное время этапов
 KolDok TRANSFER ,Rasv ; На очередной этап мониторинга

; Сегмент имитации выполнения других заявок

GENERATE 15,10 ; Источник других заявок
 QUEUE Dlina ; Встать в общую очередь заявок на ЭВМ
 SEIZE Ewm ; Занять ЭВМ
 DEPART Dlina ; Покинуть общую очередь заявок на ЭВМ
 ADVANCE 10,4 ; Выполнение запроса
 RELEASE Ewm ; Освободить ЭВМ
 TERMINATE ; Выполненные запросы

; Сегмент задания времени моделирования

GENERATE 600
 SAVEVALUE Vrem,(X1/N\$KolDok) ; Среднее время этапа мониторинга
 TERMINATE 1

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE. (-0)
DLINA	34	20	450076	1	21.905	146.007	146.008
DLINA1	15	13	249954	0	12.167	146.030	146.030

SAVEVALUE	RETRY	VALUE
VREM	0	180.037

Среднее время одного этапа мониторинга одним ПН составляет 3 часа: значение сохраняемой ячейки с именем VREM равно 180,037 мин, т. е. 3 часа.

В приведенной модели предполагается, что число ПН остается неизменным. Однако на практике это может быть не так. Например, будут потери или вследствие какой-либо необходимости потребуется уменьшить или увели-

чить число ПН. В случае моделирования с учетом потерь можно, например, добавить блок TRANSFER в статистическом режиме. При заданной доле потерь, в соответствии с ней будет уменьшаться число ПН, т. е. не все они будут возвращаться и передавать данные.

Для имитации восполнения потерь ПН или их числа целесообразно ввести дополнительно блок GENERATE, задав в нем закон пополнения ПН, и направлять эти ПН с помощью еще одного дополнительного блока TRANSFER.

Пример 5.11. Средства связи (СС) после сборки проходят проверку на функционирование на двух пунктах контроля (ПК). Если они функционируют нормально, то отправляются по назначению. В противном случае они возвращаются на настройку, которая выполняется на одном пункте, а затем снова на ПК для проверки на функционирование. Интервалы первичного поступления на ПК, время проверки и время настройки распределены по экспоненциальному закону со средними значениями 50, 90 и 200 мин соответственно. Примерно 85% СС проходят проверку успешно. После настройки около 15% СС бракуются.

Построить модель с целью определения количества СС, проверенных за 8 часов и готовых к применению, а также отправленных в брак.

Программа модели представлена ниже.

Блоки ENTER и LEAVE образуют МКУ с именем Kont, имитирующее работу двух ПК. Блоки SEIZE и RELEASE представляют в модели ОКУ — пункт настройки СС. Первая пара блоков QUEUE и DEPART осуществляет сбор информации об очереди Ar1 к МКУ Kont, а вторая пара — об очереди Ar2 к ОКУ Nastr. Первый блок TRANSFER осуществляет статистическую передачу транзактов, движение которых в модели предположительно соответствует движению СС в реальной системе. Второй блок TRANSFER также в статистическом режиме осуществляет передачу транзактов — СС в брак и на ПК.

; Пример 5.11

Kont STORAGE 2 ; Число пунктов контроля

; Сегмент имитации работы пунктов контроля

	GENERATE	(Exponential(47,0,8Q))	; Источник СС
Bac	QUEUE	Ar1	; Встать в очередь на пункт контроля
	ENTER	Kont	; Занять один пункт контроля
	DEPART	Ar1	; Покинуть очередь на пункт контроля
	ADVANCE	(Exponential(67,0,90))	; Контроль
	LEAVE	Kont	; Освободить один пункт контроля
	TRANSFER	.15, Rem1	; 15% отправить на проверку
SrSv	TERMINATE		; 85% готовых СС

; Сегмент имитации работы пункта проверки

Rem1	QUEUE	Ar2	; Встать в очередь на проверку
	SEIZE	Nastr	; Занять пункт проверки
	DEPART	Ar2	; Покинуть очередь на пункт проверки
	ADVANCE	(Exponential(77,0,200))	; Проверка

```

                RELEASE   Nastr      ; Освободить пункт проверки
                TRANSFER  .8,,Bac    ; 20% отправить в брак
Brak            TERMINATE          ; Бракованные СС

```

; Сегмент задания времени моделирования

```

                GENERATE   4800
                SAVEVALUE  IspSS,(INT(N$SrSv/5000))
                SAVEVALUE  BrakSS,(INT(N$Brak/5000))
                TERMINATE  1
                START      5000

```

По результатам 5 000 прогонов за 8 часов проверено 58 и забраковано 2 СС. Модель запускается сразу после трансляции, так как указана команда START. Количество прогонов (5 000) используется в расчетах при определении количества исправных и забракованных СС в блоках SAVEVALUE. При изменении числа прогонов нужно внести изменения и в операнд В блоков SAVEVALUE.

Пример 5.12. На узел связи, имеющий три канала, поступают заявки на передачу сообщений с интервалами, распределенными по экспоненциальному закону со средним значением 120 сек. Время передачи сообщения также имеет экспоненциальное распределение со средним значением 300 сек. Если все три канала заняты, то с вероятностью 0,15 в дальнейшем не будет предпринято повторных попыток передать эти же сообщения, и они будут потеряны. С вероятностью 0,85 будут предприняты попытки снова передать эти сообщения. Причем с вероятностями 0,35 и 0,65 повторные попытки будут предприняты через интервалы времени, распределенные по экспоненциальному закону со средними значениями 370 и 425 сек соответственно.

Построить модель с целью определения относительного количества заявок, по которым были переданы сообщения со второй и более попыток.

Программа модели системы с отказами и повторными попытками передачи сообщений приведена ниже.

; Пример 5.12

; Сегмент имитации работы узла связи

```

                GENERATE   (Exponential(54,0,120)) ; Источник заявок
Met6            ASSIGN    1,1                      ; Код 1 – первая попытка
Met1            TRANSFER  B0TH,,Met2                ; Не заполнен МКУ?
                ENTER      Usl3                      ; Нет, тогда занять канал
                ADVANCE    (Exponential(45,0,300)) ; Передача
                LEAVE      Usl3                      ; Освободить канал
                TEST E     P1,1,Met5                ; Первая попытка?
                TERMINATE          ; Переданные сообщения с первой попытки
Met5            TERMINATE          ; Переданные сообщения не с первой попытки
Met2            TRANSFER  .15,,Met3                  ; 0,15 заявок не повторяются
                INDEX      1,1                      ; Фиксация очередной попытки
                TRANSFER  .35,,Met4                  ; Деление потока повторных заявок
                ADVANCE    (Exponential(54,0,425)) ; Время очередной попытки

```

```

Met4      TRANSFER      ,Met1                ; Снова на узел связи
          ADVANCE      (Exponential(54,0,370)) ; Время очередной попытки
          TRANSFER      ,Met1                ; Снова на узел связи
Met3      TERMINATE
; Сегмент задания времени моделирования
          GENERATE      4320
          SAVEVALUE     VPovt,(N$Met5/N$Met6) ; Доля повторов
          TERMINATE     1

```

В параметр 1 транзактов, поступающих в модель, заносится код 1 — признак первой попытки передать сообщение. Далее транзакты попадают в блок TRANSFER с меткой Met1, работающий в режиме BOTH. Если в момент поступления транзакта в МКУ Use13 хотя бы один канал свободен, то блок TRANSFER направит транзакт в следующий блок, т.е. в блок ENTER. Если же в момент поступления три канала МКУ заняты, и поэтому блок ENTER отказывает во входе, транзакт будет направлен в блок TRANSFER с именем Met2, работающий в статистическом режиме. С вероятностью 0,85 транзакты из этого блока передаются в следующий блок INDEX, который увеличивает значение параметра 1 на 1 — фиксируется очередная попытка передать сообщение. Далее поток повторных попыток блоком TRANSFER в статистическом режиме разбивается на две части. Каждая из частей задерживается в соответствующем блоке ADVANCE на случайное время и с помощью блока TRANSFER, работающего в безусловном режиме, передается вновь на вход модели в блок с именем Met1. С вероятностью 0,15 транзакты из блока с меткой Met1 передаются в блок TERMINATE с меткой Met3 для уничтожения.

По результатам 5 000 прогонов относительное количество сообщений, переданных со второй и более попыток, составляет 0,316.

Примеры моделей с МКУ и переключателями будут приведены еще в последующих главах.

6. Методы применения функций в имитационных моделях

6.1. Генераторы псевдослучайных чисел

В общем случае источником «случайности» в моделях являются одна или несколько функций, при обращении к которым можно получить случайные значения из выборки равномерно распределенных в интервале $[0, 1]$ чисел. Розыгрыш значения в соответствии с некоторым распределением обычно включает два этапа:

- разыгрывается случайное число из выборки чисел, равномерно распределенных в интервале $[0, 1]$;
- полученное число каким-то образом преобразуется в эквивалентное ему значение, но уже принадлежащее интересующей нас выборке.

В GPSS World может быть любое число генераторов равномерно распределенных псевдослучайных чисел в интервале $[0, 1]$. Обращаются к генератору без предварительного его объявления посредством СЧА RN_n , где n — номер генератора. Начальное число генератора случайных чисел совпадает с номером n генератора. Например, $RN2127$. Генератор номер 2127 запускается начальным числом 2127.

При моделировании может возникнуть необходимость исследовать:

- различные варианты модели при одном и том же входном потоке случайных чисел;
- один или несколько вариантов модели при различных потоках случайных чисел.

Первая необходимость легко реализуется, поскольку генераторы позволяют воспроизводить одни и те же последовательности равномерно распределенных случайных чисел.

Во втором случае GPSS предоставляет возможность изменять начальные числа первых семи $RN1 \dots RN7$ генераторов, а следовательно, и формировать различные последовательности. Для этого в GPSS имеется команда $RMULT$ следующего формата:

$RMULT \quad [A], [B], [C], [D], [E], [F], [G]$

Операнд A — начальное число для RN1, B — для RN2, C — для RN3, D — для RN4, E — для RN5, F — для RN6, G — для RN7.

Например:

RMULT , 112, 17, . . , 295

Устанавливаются начальные числа генераторов 2, 3, 6. Остальные значения остаются без изменений.

В GPSS World псевдослучайные числа генерируются по 32-битному мультипликативному конгруэнтному алгоритму. Алгоритм позволяет получить 2 147 483 646 уникальных псевдослучайных чисел, т. е. с периодом $2^{31} - 2$ не включающим 0.

СЧА RN принимает целые значения от 0 до 999 включительно, а при вычислении случайных функций используется случайное число, выбираемое из интервала от 0 до 0,999999 включительно.

В GPSS World генераторы случайных чисел используются при обработке одновременных событий (п. 7.4), в блоках TRANSFER, работающих в режиме статистической передачи (п. 2.7), в блоках GENERATE (п. 2.1) и ADVANCE (п. 2.4). Пользователь может выбрать любой генератор на странице Random Numbers (Случайное число) журнала настроек модели (по умолчанию — RN1).

6.2. Типы функций

Для применения в GPSS функция должна быть задана таблицей. Правила поиска значения функции по таблице определяет команда FUNCTION, за которой следуют пары возможных значений аргумента и функции. Команда FUNCTION имеет следующий формат:

Name	FUNCTION	A, B

Name — имя функции.

Операнд A — аргумент функции. Может быть именем, положительным целым числом, строкой, выражением в скобках, СЧА или СЧА*параметр.

Операнд B — состоит из одной буквы, определяющей тип функции, и целого положительного числа, задающего количество пар возможных значений аргумента и функции (количество точек функции).

В GPSS World можно применять следующие пять типов функций:

- C — непрерывная числовая;
- D — дискретная числовая;
- E — дискретная атрибутивная;
- L — списковая числовая;
- M — списковая атрибутивная.

Существует еще один тип функции, которая применяется в блоках GENERATE и ADVANCE в качестве операнда B. Она называется модификатор функции. Зна-

чение этой функции вычисляется с двойной точностью и умножается на значение операнда A. Полученный результат используется как требуемая временная задержка в данном блоке.

Значения, заданные командой FUNCTION, хранятся в форме чисел с плавающей запятой двойной точности. Эти значения ограничены 306 десятичными разрядами с точностью 15 десятичных разрядов.

ЗАМЕЧАНИЕ

Для некоторых функций удобнее использовать не функции, а Plus-процедуры. Однако функции GPSS лучше подходят для применения их в качестве табличных функций. Кроме того, функция GPSS может вычисляться за меньшее время, чем аналогичная Plus-процедура.

Рассмотрим методы получения случайных чисел различных распределений с помощью имеющихся средств GPSS World.

6.3. Методы розыгрыша псевдослучайных чисел для дискретных равномерных распределений

Ранее было рассмотрено, как получить интервалы поступления и обслуживания заявок, распределенные по равномерному закону. Напомним, что в блоках GENERATE и ADVANCE равномерный закон распределения задается с помощью модификатора-интервала (см. п. 2.1). Например:

GENERATE	3600,900
ADVANCE	3600,900

Можно использовать также встроенный генератор дискретно-равномерного распределения DUniform(n,min,max), где n — номер генератора равномерно распределенных случайных чисел, min и max — минимальное и максимальное генерируемые значения соответственно. Перепишем примеры:

GENERATE	DUniform(1,2700,4500)
ADVANCE	DUniform(1,2700,4500)

Теперь используем оба этих метода в следующих простых примерах и сравним полученные результаты.

GENERATE	3600,900
QUEUE	1
SEIZE	Can
DEPART	1
ADVANCE	3600,900
RELEASE	Can
TERMINATE	
GENERATE	144000
TERMINATE	1

```

GENERATE      (DUniForm (1,2700,4500))
QUEUE         1
SEIZE         Can
DEPART        1
ADVANCE       (DUniForm (1,2700,4500))
RELEASE       Can
TERMINATE
GENERATE      144000
TERMINATE     1

```

При использовании модификатора-интервала за 10 000 прогонов блок GENERATE сгенерировал 400 005 транзактов, из которых ОКУ Can было обслужено 399 981. Применение встроенного генератора DUniForm(1,2700,4500) дало 399 926 и 399 863 соответственно. Относительная разница составляет 0,0002. В обоих примерах в качестве генератора равномерно распределенных случайных чисел использовался RN1.

В моделях требуется получать и другие случайные величины, а не только интервалы поступления и обслуживания заявок. Например, количество знаков в сообщении, вычислительную сложность алгоритма и т. д. Для розыгрыша таких случайных величин также можно использовать встроенный генератор дискретно-равномерного распределения. Например:

```

Znak          VARIABLE      DUniForm(211,5000,9000)
               ASSIGN       2,V$Znak

```

Генерируемое DUniForm случайное число присваивается арифметической переменной Znak, на которую потом и ссылаются в модели.

Возможен и такой вариант:

```

ASSIGN        2,(DUniForm(211,5000,9000))

```

В качестве операнда В блока ASSIGN записано выражение в скобках, что исключает необходимость применения арифметической переменной.

Для получения равномерно распределенных случайных чисел можно применять СЧА RN. Как уже отмечалось, СЧА RN принимает значения от 0 до 999 включительно. Значение, соответствующее равномерному распределению $A \pm B$, рекомендуется вычислять как выражение

$$A - B + (2 \# B \# RNn) / 999 \quad (6.1)$$

Данное выражение получается путем добавки к меньшему значению выборки «случайной доли» ширины выборки. Наименьшим значением является $A - B$, а ширина выборки равна $2B$. Значение, полученное от генератора случайных чисел, рассматривается как «случайная доля». Таким образом, новое значение $A - B + RNn \# 2 \# B$. Однако возникает следующая проблема. Новое значение может совпасть с максимально допустимым значением только в том случае,

если RNn выдаст значение 1 000. Но так как это невозможно, может быть только 999, то максимальное разыгрываемое значение $A+2B$ никогда не получить. Для исправления этого и рекомендуется выражение (6.1).

Используем выражение (6.1) в следующем примере.

; Пример 6.1

```

Sred      EQU      3600          ; Среднее значение
St0tk     EQU      900          ; Стандартное отклонение
DRavR     VARIABLE  Sred-St0tk+(2#St0tk#RN733)/999
          GENERATE  ,, ,10000000
          SAVEVALUE 1,V$DRavR    ; Случайное число
          TEST E     Sred-St0tk),X1,Met1 ; Минимальное значение?
          TERMINATE  1           ; Да
Met1      TEST E     (Sred+St0tk),X1,Met2 ; Максимальное значение?
          TERMINATE  1           ; Да
Met2      TERMINATE  1           ; Другие значения

```

Блок GENERATE генерирует 10 000 000 транзактов. В блоке SAVEVALUE вычисляется арифметическая переменная с именем DRavR и полученное значение записывается в сохраняемую ячейку номер один. Далее первым блоком TEST проверяется равенство значения арифметической переменной минимальному генерируемому значению дискретного равномерного распределения, а вторым блоком TEST — максимальному генерируемому значению.

Приведем результаты моделирования.

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	10000000	0	0
	2	SAVEVALUE	10000000	0	0
	3	TEST	10000000	0	0
	4	TERMINATE	10004	0	0
MET1	5	TEST	9989996	0	0
	6	TERMINATE	9991	0	0
MET2	7	TERMINATE	9980005	0	0

В первый и второй блоки TERMINATE вошли 10 004 и 9 991 транзактов соответственно, т. е. 0,0001 в каждый. Точно такой же результат будет получен, если вместо переменной DRavR использовать просто RN733, что подтверждает целесообразность применения выражения (6.1).

В рассмотренных случаях результатами розыгрыша являлись дробные числа. В некоторых случаях требуется получить целые числа. Например, при розыгрыше количества заявок в сообщении, количества деталей, поступающих на обработку, автомобилей в транспорте и т. д. Язык Plus (глава 8) имеет процедуру INT выделения целой части с отбрасыванием дробной части. Этой процедурой и нужно воспользоваться. В приведенном выше примере арифметическое выражение DRavR запишется так:

```
DRavI      VARIABLE      INT(Sred-St0tk+(2#St0tk#RN733)/999)
```

При использовании встроенного генератора DUniform процедура INT применяется также, например:

```
Znak      VARIABLE      INT(DUniform(211, 5000, 9000))
```

6.4. Методы розыгрыша псевдослучайных чисел для дискретных неравномерных распределений

При определении интервалов поступления и обслуживания заявок или других величин часто бывает необходимо разыгрывать случайные числа для неравномерных распределений. Рассмотрим методы розыгрыша таких чисел для дискретных распределений.

Пусть функция, которую нужно применять в модели, задана табл. 6.1. Согласно этой таблице случайная величина должна получать значения 2, 5, 8, 9, 12 с частотой 0,15; 0,2; 0,25; 0,22; 0,18 соответственно. Графическая интерпретация функции представлена на рис. 6.1.

Таблица 6.1

Задание функции				
Значение случайной величины	Относительная частота	Суммарная частота	Диапазон	Интервал
2	0,15	0,15	0,0...0,15	1
5	0,2	0,35	0,15...0,35	2
8	0,25	0,6	0,35...0,6	3
9	0,22	0,82	0,6...0,82	4
12	0,18	1,0	0,82...1,0	5

Разыграем число в интервале [0, 1]. Пусть им будет 0,523664. Тогда, просматривая таблицу, видим, что число попадает в интервал 3. Искомым значением случайной величины является 8.

Таким образом, когда необходимо разыграть случайное число в соответствии с некоторым дискретным распределением, планировщик должен автоматически выполнить процедуру просмотра таблицы, которой задана функция. В данном случае для этого ему необходимо иметь:

- источник случайных чисел в интервале [0, 1];
- значения случайной величины;
- суммарную частоту принятия значений случайной величиной.

Эти данные пользователь задает с помощью функции типа D — дискретного типа. В списке данных функции типа D значения аргумента X должны быть типа

Expression, а значения функции Y — типа integer, real или Name. Для нашего примера функция, заданная табл. 6.1, будет определена командой FUNCTION так:

```
Rasp      FUNCTION      RN6, D5
. 15, 2/. 35, 5/. 6, 8/. 82, 9/1, 12
```

Функция имеет имя Rasp. В качестве источника случайных чисел выступает RN6. Случайная величина может иметь пять различных значений. Суммарные частоты и соответствующие им пять значений функции записаны как пять пар чисел.

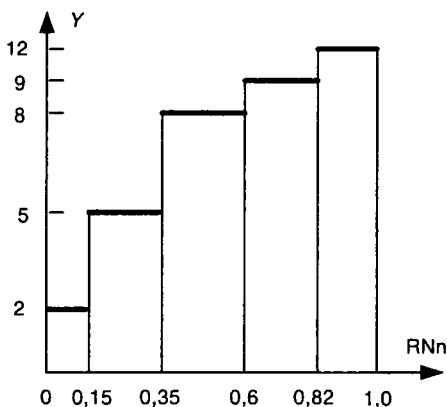


Рис. 6.1. Графическая интерпретация дискретной функции

При написании пар чисел, определяющих координаты X и Y точки не только дискретной, но и функции любого типа, необходимо выполнять следующие правила:

- значения координат X и Y одной точки функции разделяются запятой;
- наборы координат разделяются знаком / или переводом строки;
- каждое последующее значение X должно быть больше предыдущего;
- каждая функция должна иметь, по крайней мере, две описанные точки;
- количество пар данных, указанных в операнде B команды FUNCTION, должно совпадать с количеством пар, разделенных символом / в списке данных функции;
- списки данных функции не имеют полей комментария.

Как известно, блок TRANSFER имеет такие режимы работы, как выборочный (PICK) и функциональный (FN). В режиме PICK выбор блока, к которому направляется активный транзакт, производится случайно. Блоки должны быть расположены друг за другом и переходы к каждому из них равновероятны. А как быть в том случае, когда блоки расположены не друг за другом и выбор каждого из них не равновероятен? В этом случае следует использовать функцию и блок TRANSFER в режиме FN. В модели это будет выглядеть так:

```

Per      FUNCTION      RN13, D4
0.2, Met4/0.5, Met2/0.76, Met3/1, Met4
      GENERATE      (Exponential(305, 0, 17.3))

      TRANSFER      FN, Per
. . . . .
Met1. . . . .
. . . . .
Met2. . . . .
. . . . .
Met3. . . . .
. . . . .
Met4. . . . .
. . . . .

```

Блок GENERATE генерирует транзакты. При входе транзакта в блок TRANSFER вычисляется функция Per, аргументом которой является RN13. Выбирается значение функции. В данном примере значения функции — метки блоков. В соответствии с выбранной меткой и производится безусловный переход активного транзакта к блоку, помеченному этой меткой.

Функцию типа D можно использовать в блоке GENERATE для генерирования транзактов с дискретными произвольными интервалами. Например:

```

DInt     FUNCTION      RN831, D4
.15, 3.2/.55, 4.7/.75, 7.1/1, 1.5
      GENERATE      FN$DInt
      TERMINATE      1

```

Блок GENERATE генерирует транзакты с интервалами 3,2, 4,7, 2,1 и 1,5, вероятности появления которых составляют 0,15, 0,4, 0,2 и 0,25 соответственно. В этом можно убедиться на приведенном примере, задавая в команде START операнд A равным 1, 2, 3, 4 и анализируя время завершения моделирования.

Точно так же функцию типа D можно применять в блоке ADVANCE.

Часто необходимо блоком ADVANCE организовать на одном и том же ОКУ или МКУ обслуживание заявок с различными временными характеристиками. В этом случае в качестве аргумента функции можно использовать какой-либо параметр транзакта. Например:

```

Obs      FUNCTION      P1, D4
1, 3.75/2, 4.3/3, 2.1/4, 5.6
      GENERATE      FN$DInt
      TERMINATE      1
. . . . .
      ADVANCE      FN$Obs
. . . . .

```

При входе активного транзакта в блок ADVANCE вычисляется функция с именем Obs, аргументом которой является параметр P1. Полученный результат и используется как время задержки.

ЗАМЕЧАНИЕ

Если в функции типа D аргумент RN, его запись больше 1 или не в строгом порядке возрастания вызывает ошибку «Вероятность не может быть больше единицы» или «Ошибка: убывающие интегральные вероятности» соответственно.

ЗАМЕЧАНИЕ

Если в функции типа D аргумент не RN, его запись не в строгом порядке возрастания не вызывает ошибку, т. е. не обнаруживается, и результаты вычисления функции получаются неверными. Например в функции

Obs	FUNCTION	P3, D4
2. 15, 3. 2/3. 55, 4. 7/11. 75, 7. 1/4. 2, 5. 6		

система ошибку пользователя не обнаруживает. При P3 = 4 будет выдано значение функции 7,1, что неверно.

6.5. Процедура розыгрыша псевдослучайных чисел при непрерывном равномерном распределении

Для получения случайных чисел при непрерывном равномерном распределении используется функция типа C. Непрерывная функция определяется с помощью символа C (в отличие от символа D для дискретной функции), записываемого в качестве первого символа операнда B. В списке данных функции X и Y должны быть типа integer или real.

Например:

VrObs	FUNCTION	RN23, C5
. 1, 14. 2/. 35, 30. 4/. 6, 50/. 85, 64. 7/1, 80		

В чем различие между непрерывной и дискретной функциями?

Вначале непрерывная функция вычисляется так же, как и дискретная. При обращении к функции разыгрывается случайное число, которое далее используется как аргумент функции. Затем просматривается таблица для определения интервала значений суммарной вероятности, на который выпало случайное число. Если функция была определена как дискретная, берут второй элемент соответствующей пары, его значение считается значением функции. Если функция определена как непрерывная, выполняется линейная интерполяция для пары точек X_i и X_{i+1} , находящихся по краям того интервала значений вероятности, в который попало случайное число X , являющееся аргументом. Интерполяция производится согласно выражению:

$$Y(X) = Y(X_i) + \frac{Y(X_{i+1}) - Y(X_i)}{X_{i+1} - X_i} \cdot (X - X_i), \quad i = \overline{1, m}. \quad (6.2)$$

где m — число точек функции.

Пусть в приведенном выше примере получили RN23=0,43. Это число попадает в интервал [0,36; 0,6]. Если бы функция была дискретной, ее значение рав-

нялось бы 50. Непрерывная же функция вычисляется с учетом интерполяции по формуле (6.2):

$$Y(0,43) = Y(0,36) + \frac{Y(0,6) - Y(0,36)}{0,6 - 0,36} \cdot (0,43 - 0,36) =$$

$$= 30,4 + \frac{50 - 30,4}{0,24} \cdot 0,07 = 30,4 + 5,7166 = 36,1166.$$

Приведем еще один пример:

```
Skor      FUNCTION      V$Put, C3
2. 1, 18. 3/4. 6, 24. 1/7. 4, 32. 6
```

В качестве аргумента функции используется арифметическая переменная с именем Put. Когда функция вызывается, сначала вычисляется эта переменная. Если значение аргумента выходит за пределы области определения функции, т. е. меньше 2,1 и больше 7,4, то функции присваиваются значения в ближайших предельных точках: 18,3 и 32,6 соответственно. При попадании аргумента в область определения функции выполняется линейная интерполяция по формуле (6.2).

Ввиду того, что непрерывная функция задается с помощью линейной интерполяции, для всех точек на одном интервале значений суммарной частоты существует одинаковая вероятность их использования. Это делает процедуру розыгрыша случайной величины при равномерном непрерывном распределении чрезвычайно простой. Предположим, что некоторая случайная переменная распределена равномерно и непрерывно на интервале от 2 до 6 (исключительно). Вероятность того, что значение переменной меньше 2, равно 0, а вероятность того, что она меньше 6, равна 1. Эти два значения суммарной вероятности используются для определения непрерывной функции. Например:

```
Ww2      FUNCTION      RN32, C2
0. 2/1, 6
```

Несмотря на то, что значением второй пары, использованной при определении функции, является 6, функция никогда не получает значение 6, потому что СЧА RN при вычислении функции не может быть больше 0,999999. Таким образом, наибольшей целой частью получаемого значения является 5.

Преимущество непрерывной функции по сравнению с дискретной функцией очевидно, если в результате розыгрыша требуется получать целые числа. Предположим, что нужно разыгрывать случайные числа в интервале от 100 до 475. Функцию можно описать так:

```
Ww5      FUNCTION      RN61, C2
0. 100/1, 476
```

Если в этих же целях воспользоваться дискретной функцией, то нужно будет рассчитать 376 пар значений суммарных частот и задать 376 пар для определения функции.

ЗАМЕЧАНИЕ

Вторая пара 1, 476, а не 1, 475. Сделано это для получения максимального генерируемого значения.

6.6. Методы розыгрыша псевдослучайных чисел для непрерывных неравномерных распределений

В некоторых случаях возникает необходимость розыгрыша псевдослучайных чисел согласно некоторому непрерывному распределению, задаваемому изначально какой-либо кривой.

Для генерации таких чисел используется метод кусочной аппроксимации плотности распределения.

Пусть случайная величина, которую требуется моделировать, задана плотностью распределения $f(Y)$ в интервале $[a, b]$. Аппроксимируем функцию $f(Y)$ (рис. 6.2).

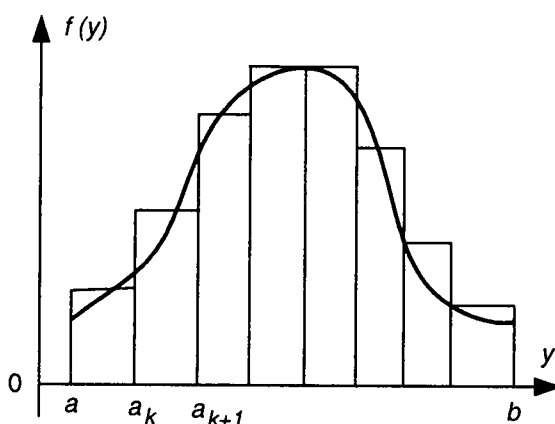


Рис. 6.2. Кусочная аппроксимация плотности распределения

Разбивать $[a, b]$ на отрезки следует так, чтобы вероятность попадания случайной величины Y на любой из отрезков $[a_k, a_{k+1}]$ была постоянной, т. е. не зависела бы от расположения на $[a, b]$. Если число отрезков m , то для любого отрезка $[a_k, a_{k+1}]$ должно быть выполнено равенство

$$\int_{a_k}^{a_{k+1}} f(Y) dY = \frac{1}{m}, \quad (6.3)$$

на основе которого можно определить аппроксимирующее значение функции на отрезке $[a_k, a_{k+1}]$.

Процедура моделирования одного значения случайной величины Y содержит этапы:

1. Формирование случайного числа с равномерным распределением в интервале $[0, 1]$.

2. Выбор по значению соответствующего ему отрезка $[a_k, a_{k+1}]$ на $[a, b]$.

Так, если $[a, b]$ содержит 10 отрезков, то значениям $0 \leq X_i \leq a_1$ соответствует первый отрезок, значениям $a_1 < X_i \leq a_2$ — второй отрезок и т. д.

3. Вычисление значения случайной величины с заданным законом распределения:

$$Y_i = a_k + (a_{k+1} - a_k) \cdot X_{i+1}. \quad (6.4)$$

Рассмотрим применение метода кусочной аппроксимации к моделированию экспоненциального и нормального распределений.

6.6.1. Моделирование экспоненциального распределения

Когда имеет место пуассоновский поток (простейший поток, удовлетворяющий требованиям ординарности, стационарности, отсутствия последствия), поступление заявок описывается законом Пуассона, т.е. определяется уравнением

$$P_k(T) = \frac{(\lambda T)^k}{k!} e^{-\lambda T}, \quad k = 0, 1, \dots, \quad (6.5)$$

где $P_k(T)$ — вероятность того, что k заявок придет за время T ;

λ — средняя интенсивность поступления заявок.

При моделировании пуассоновского потока нет необходимости в определении вероятностей $P_k(T)$. Достаточно иметь лишь значения интервалов времени поступления требований. Расчет значения времени прихода следующего транзакта в блок GENERATE производится посредством сложения текущего значения системного времени и разыгранной по распределению интервалов времени величины. Это уравнение можно выразить относительно времени. Результатом является экспоненциальное распределение.

Экспоненциальное распределение методом обратной функции может быть задано в такой форме, что если имеются равномерно распределенные в интервале $[0, 1]$ числа, то соответствующие интервалы времени могут быть вычислены непосредственно из выражения

$$T_p = T_{cp} [-\text{LN}(1 - \text{RNn})], \quad (6.6)$$

где T_p — разыгранный интервал времени;

T_{cp} — средний интервал времени;

RNn — имя генератора равномерно распределенных случайных чисел.

При использовании этого уравнения для моделирования пуассоновского входящего потока в блоке GENERATE следует:

получить число от RNn и взять логарифм величины $(1 - RNn)$ со знаком минус, затем полученный результат умножить на T_{cp} .

Первый этап можно выполнить путем задания в GPSS функции, использующей в качестве аргумента RNn и принимающей значения $[-LN(1 - RNn)]$. В GPSS функцию логарифма можно определить следующим способом. Распределение вида $[-LN(1 - RNn)]$ следует аппроксимировать отрезками, накладываемыми на действительную функцию. Такая аппроксимация для GPSS осуществлена 23 отрезками, которые охватывают значения экспоненциальной функции случайной переменной на интервале $[0, 8]$. Определение такой функции, имеющей имя $XpDis$ (EXPONENTIAL DISTRIBUTION), состоит из 24 пар значений, используемых для преобразования значений RNn в значение $[-N(1 - RNn)]$:

```
XpDis      FUNCTION      RN200,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
```

Теоретически значения выражения $[-LN(1 - RNn)]$ могут находиться в диапазоне чисел от 0 до ∞ . Практически, поскольку наибольшим значением RNn является 0,999999, наибольшим значением выражения является 13,8155. Так как последним значением функции является пара .9998, 8, функция выдает в качестве своего значения величину 8, а не 13,8155 (рис. 6.3).

Однако RNn получает значения больше чем 0,9998 только в 0,0199% случаев. В большинстве случаев такая аппроксимация является вполне достаточной. На втором этапе значение функции нужно умножить на T_{cp} , чтобы получить значение интервала времени, соответствующего пуассоновскому потоку.

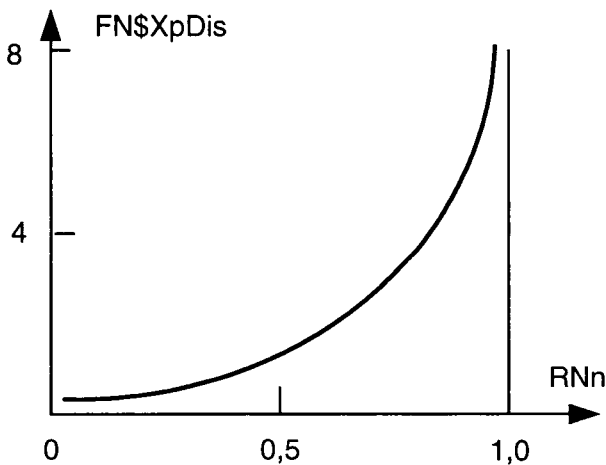


Рис. 6.3. Графическая интерпретация функции $FN\$XpDis$

Для автоматического выполнения этого блок GENERATE имеет специальное свойство. Пуассоновский поток моделируется с помощью блока GENERATE путем выполнения следующих действий:

- в качестве операнда A используется значение интервалов времени;
- в качестве операнда B используется запись FN\$XpDis.

Значение интервала времени вычисляется в блоке посредством умножения значения функции на значение операнда A.

Пример 1. Предположим, что необходимо промоделировать пуассоновский входящий поток со средним значением интенсивности, равным 4, в течение каждых 24 часов. Единица времени — 1 мин. Когда транзакт входит в блок и переходит к следующему блоку, планировщик определяет время поступления следующего транзакта путем вычисления FN\$XpDis, умножения результата на 360 и добавления его к текущему времени таймера. Для вычисления функции FN\$XpDis необходимо определить значение $RN23 = 0,35$, $FN\$XpDis = 0,432$ (среднее между 0,355 и 0,509). Произведение равно 155,52. Интервал времени — 155,52.

При использовании переменной, например:

Rys FVARIABLE 360#FN\$XpDis

блоки GENERATE 360, FN\$XpDis и GENERATE V\$Rys эквивалентны.

Пример 2. Все, что было изложено выше относительно моделирования пуассоновского потока блоком GENERATE, остается в силе для моделирования экспоненциального времени задержки в блоке ADVANCE. Например, если ОКУ с экспоненциальным обслуживанием в среднем требует 75 единиц модельного времени на одно обслуживание, его можно промоделировать следующими блоками ADVANCE:



6.6.2. Моделирование нормального распределения

Случайная величина с нормальным распределением полностью описывается заданием двух значений: математического ожидания и стандартного отклонения. Нормированная случайная величина с нормальным распределением имеет математическое ожидание, равное нулю, и стандартное отклонение, равное 1. Как правило, нормальные случайные величины, представляющие интерес, являются ненормированными.

Для того, чтобы сделать выборку из ненормированного нормального распределения, удобно вначале осуществить выборку из нормированного нормального распределения и произвести арифметические действия с полученной величиной, преобразуя ее в ненормированную:

$$G\text{Norm}_{\text{выб}} = (G\text{Norm}_{\text{сот}})(S\text{Norm}_{\text{выб}}) + G\text{Norm}_{\text{мож}},$$

где $S\text{Norm}_{\text{выб}}$ — величина, выбранная из нормированного нормального распределения;

$G\text{Norm}_{\text{мож}}$ и $G\text{Norm}_{\text{сот}}$ — математическое ожидание и стандартное отклонение соответственно ненормированного нормального распределения.

Случайная величина для нормированного нормального распределения может быть найдена из выражения

$$RNn = \int_{-\infty}^{S\text{Norm}_{\text{выб}}} \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} dx,$$

где RNn — случайная величина любого из равномерных распределений GPSS от 0 до 1;

$S\text{Norm}_{\text{выб}}$ — верхний предел интегрирования.

Уравнение нельзя решить аналитически для нахождения точного значения $S\text{Norm}_{\text{выб}}$ для RNn . Как следствие, величины $S\text{Norm}_{\text{выб}}$, определяемые из уравнения для RNn , должны быть протабулированы.

На рис. 6.4 показана зависимость между RNn и $S\text{Norm}_{\text{выб}}$.

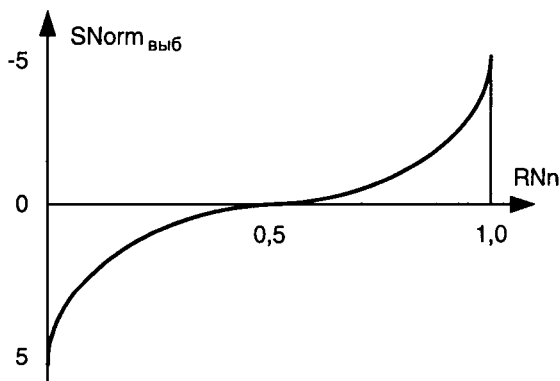


Рис. 6.4. Графическая интерпретация $S\text{Norm}_{\text{выб}}$

Эта кривая аппроксимируется с помощью непрерывной функции GPSS, составленной из 24 соответствующих ей прямолинейных сегментов. Определение функции, имеющей имя $S\text{NORM}$, состоит из 25 пар значений:

SNorm FUNCTION RN123, C25
0, -5/.00003, -4/.00135, -3/.00621, -2.5/.02275, -2/.06681, -1.5/.11507, -1.2/.15866, -1
.21186, -.8/.27245, -.6/.34458, -.4/.42074, -.2/.5, .0/.57926, .2/.65542, .4/.72575, .6
.78814, .8/.84134, 1/.88493, 1.2/.93319, 1.5/.97725, 2/.99379, 2.5/.99865, 3/.99997, 4/1, 5

Заметим, что хотя величины $S_{Norm_{выб}}$ могут лежать в пределах от $-\infty$ до $+\infty$, величины, взятые для $FN\$S_{Norm}$, могут меняться только в интервале от -5 до $+5$. Для практики это расхождение является приемлемым.

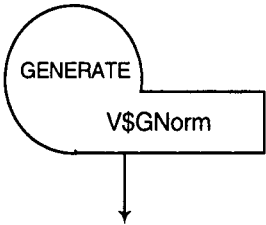
Рассмотрим примеры применения нормальных распределений в GPSS.

Пример 1. Время между поступлениями заявок на передачу сообщений по каналу связи распределено по нормальному закону с математическим ожиданием 24 ч и стандартным отклонением 2 ч. За единицу времени принимается 1 ч.

Блок GENERATE, имитирующий процесс поступления заявок, может быть представлен следующим образом.

Переменная GNorm, используемая в операнде A блока GENERATE (см. схему), описана так:

GNorm FVARIABLE 2#FN\$SNorm + 24



Таким образом, всякий раз, когда транзакт выходит из блока GENERATE, планировщик случайным образом определяет величину времени выхода следующего транзакта, удовлетворяющего способу определения ненормированной случайной величины, распределенной по нормальному закону.

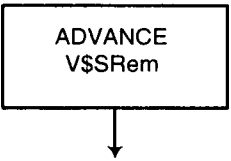
Пример 2. Время обслуживания в некоторой точке модели имеет нормальное распределение, а математическое ожидание и стандартное отклонение в действительности зависят от вида осуществляемого обслуживания. Например, ремонт различных типов средств связи в одном и том же ремонтном органе одними и теми же мастерами (табл. 6.2).

Таблица 6.2

Задание закона распределения в зависимости от вида обслуживания

Вид обслуживания	Математическое ожидание	Стандартное отклонение
1	30	5
2	20	3
3	36	6

Предположим, что вид обслуживания, необходимый транзакту, записан в его параметре 4. Тогда блок ADVANCE можно представить так, как показано справа. Определение функций и арифметической переменной, обеспечивающих функционирование блока, записываются так:



Mean	FUNCTION	P4, D3
1, 30/2, 20/, 3, 36		
StDe	FUNCTION	P4, D3
1, 5/2, 3/, 3, 6		
Srem	FVARIABLE	FN\$StDe#FN\$SNorm+FN\$Mean

Функции в приведенных примерах в качестве своих аргументов используют значение параметра P4 активного транзакта. При обращении к функции SRem вначале вычисляются функции Mean, StDe, SNorm. Полученные значения функций используются при вычислении функции SRem.

ЗАМЕЧАНИЕ

Наименьшее значение функции равно -5 . Может оказаться, что соответствующее значение выборки будет отрицательным. Однако на практике некоторые отрицательные величины не имеют смысла (например, время). Разработчик должен быть уверен, что *математическое ожидание по крайней мере в пять раз больше стандартного отклонения*. В противном случае произойдет останов по ошибке «Отрицательное время задержки».

6.7. Функции типа E, L и M

Функция типа E. Как известно, в дискретной числовой функции типа D значения аргумента X должны быть типа *Expression*, а значения функции Y — типа *integer*, *real* или *Name*. В дискретной атрибутивной функции типа E аргумент X также должен быть типа *Expression*, а Y может быть типа *integer*, *real*, *Name*, СЧА или выражение в скобках. Например:

Vibor	FUNCTION	P3, E4
1, V\$Norm1/7, V\$Norm2/12, R\$Use1/14, FN\$Sprav		

Функция типа E вычисляется так же, как и функция типа D, с той лишь разницей, что после нахождения аргумента X вычисляется соответствующий СЧА или выражение в скобках. Например, при P3 равном 13 вычисляется функция с именем Sprav.

ЗАМЕЧАНИЕ

Если в функции типа E Y имеет значения типа *integer*, *real* или *Name*, то она идентична функции типа D.

Функция типа L. В функции типа E аргумент X может быть типа *Expression*, т. е. после вычисления принимать любые значения. В списковой числовой функции типа L аргумент X может быть только типа *integer*. Причем аргумент рассматривается как порядковый номер. Значения функции — *integer*, *real* или *Name*. Например:

Blok	FUNCTION	Q\$Dlina, L5
1, Met1/2, Met2/3, Met3/4, Met4/5, Met5		

По порядковому номеру и находится значение функции. Если в результате вычисления аргумент меньше 1 или больше его заданного максимального значения, происходят остановы по ошибкам «Неположительный аргумент списка функции» или «Аргумент списковой функции слишком велик» соответственно. Значения аргумента должны начинаться с 1 и увеличиваться на 1 для каждой следующей пары данных. Если аргумент будет иметь значения не с единицы, а, например, с двух, или будет нецелым, например 1,5, возникнет ошибка «Недопустимый порядковый номер в списке». В ходе моделирования вычисленное значение аргумента автоматически преобразуется в целое, поэтому может иметь нецелые значения.

Функция типа М. Списковая атрибутивная функция типа М отличается от списковой числовой функции типа L тем, что значения функции могут быть не только типа *integer*, *real* или *Name*, а и СЧА или выражением в скобках. Аргумент также целого типа и рассматривается как порядковый номер. Поэтому он должен начинаться с 1 и возражать на 1 для каждой следующей пары данных.

Например:

```
Pos1      FUNCTION      X$Kvart, M5  
1, 0$Stan/2, N$Term3/3, S$Usl/4, V$Zirk/5, FN$Susp
```

Вычисляется функция типа М так же, как функция типа L, только после нахождения аргумента — порядкового номера вычисляется соответствующий СЧА или выражение в скобках. Например, при $X = 4$ вычисляется арифметическая переменная с именем *Zirk*. При $1 > X > 5$ также происходят остановы по ошибкам «Неположительный аргумент списка функции» и «Аргумент списковой функции слишком велик» соответственно.

ЗАМЕЧАНИЕ

Если в функции типа М Y имеет значения типа *integer*, *real* или *Name*, то она аналогична функции типа L.

6.8. Примеры построения моделей с использованием функций

Пример 6.1. В АСУ осуществляется сбор информации о состоянии системы ТО. Информация (донесения, заявки и др.) может быть 8 видов. Поток поступления заявок на обработку простейший со средним временем $T_1 = 60$ мин. Относительная частота поступления информации (в дальнейшем — донесений) того или иного вида i . Объем (в знаках) каждого донесения распределен по нормальному закону с математическим ожиданием Q_i и стандартным отклонением S_i . Время обработки вычислительной системой i -го донесения распределено по экспоненциальному закону со средним значением T_{2i} .

Вариант исходных данных для построения модели приведен в табл. 6.3 и табл. 6.4.

Таблица 6.3

Данные о донесениях

Характеристика донесений	Вид донесения							
	1	2	3	4	5	6	7	8
Относительная частота, ν_i	0,1	0,13	0,08	0,1	0,09	0,17	0,13	0,18
Математическое ожидание количества знаков в донесении, Q_i	420	510	630	920	1200	730	340	480
Стандартное отклонение количества знаков в донесении, S_i	60	90	120	170	230	140	60	80
Среднее время обработки донесения, T_{2i} , с	8	11	6	16	10	13	3	9

На основе поступающей информации принимаются решения. При этом со средним интервалом времени Lam1 решается одна задача из шести различных задач. Относительная частота решения j -й задачи λ_j . Каждая задача решается в диалоговом режиме. Число диалоговых точек при решении j -й задачи распределено по нормальному закону с математическим ожиданием N_j и стандартным отклонением m_j . Время диалога в jk -й точке подчиняется экспоненциальному закону со средним значением T_{3j} . Непосредственное время работы ЭВМ после jk -го диалога также распределено по экспоненциальному закону со средним значением T_{4j} .

Таблица 6.4

Данные о задачах

Характеристики задач	Тип задачи					
	1	2	3	4	5	6
Относительная частота решения, λ_j	0,14	0,17	0,15	0,17	0,18	0,19
Математическое ожидание числа диалоговых точек, N_j	23	13	29	8	19	31
Стандартное отклонение числа диалоговых точек, m_j	4	2	4	1	3	5
Среднее время диалога в jk -й точке, T_{3i} , с	25	62	78	115	93	137
Среднее время работы ЭВМ после диалога в jk -й точке, T_{4i} , с	9	13	7	5	11	16

Построить модель системы сбора и обработки данных по ТО войск с применением АСУ. Функционирование автоматизированной системы необходимо промоделировать в течение 24 часов. За единицу модельного времени возьмем 1 с. Тогда время одного прогона модели будет равно $24 \times 60 \times 60 = 86400$ единицам модельного времени.

Первый вариант модели, соответствующий данной постановке, был описан в п. 1.4. Второй вариант этой же модели приводится ниже. Он отличается тем, что в нем применены не встроенные функции экспоненциального и нормального распределений, а функции XpDis и SNorm. Кроме того, исходные данные о донесениях также записаны с использованием функций, а не матрицы. За счет этого количество строк, потребовавшихся для записи табл. 6.3, сократилось с 24 до 6.

Большая часть программы посвящена описанию исходных данных. После ввода генератором очередного транзакта-донесения в модель с помощью функции TDon1 разыгрывается вид донесения и его код заносится в P2 транзакта. Далее также случайным образом определяется количество знаков в донесении и присваивается P3 транзакту. После этого транзакт-донесение направляется для обработки на ЭВМ. Аналогичным образом имитируется поступление заявок-задач в модель, разыгрываются их характеристики и осуществляется решение на ЭВМ в зависимости от ее загруженности и числа диалоговых точек

Блоковая часть программы модели, которая непосредственно имитирует функционирование АСУ, не содержит числовых значений исходных данных. Все они описаны и заданы предварительно, что в последующем облегчает при необходимости их корректировку.

; Пример 6.1

Swd	EQU	3600	; Среднее время поступления донесений
Swz	EQU	1800	; Средний интервал времени решения задач
VrMod	EQU	86400	; Время моделирования

; *Определение функции экспоненциального распределения*

```
Xpdis    FUNCTION    RN121,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
```

; *Определение функции нормального распределения*

```
SNorm    FUNCTION    RN123,C25
0,-5/.00003,-4/.00135,-3/.00621,-2.5/.02275,-2/.06681,-1.5/.11507,-1.2/.15866,-1
.21186,-.8/.27245,-.6/.34458,-.4/.42074,-.2/.5,.0/.57926,.2/.65542,.4/.72575,.6
.78814,.8/.84134,1/.88493,1.2/.93319,1.5/.97725,2/.99379,2.5/.99865,3/.99997,4/1,5
```

; *Определение функций для расчета данных о задачах*

```
TZad1    FUNCTION    RN1,D6
.14,1/.31,2/.46,3/.63,4/.81,5/1,6
TZad2    FUNCTION    P2,D6
```

```

1, 23/2, 13/3, 29/4, 8/5, 19/6, 31
TZad3      FUNCTION      P2, D6
1, 4/2, 2/3, 4/4, 1/5, 3/6, 5
TZad4      FUNCTION      P2, D6
1, 25/2, 62/3, 78/4, 115/5, 93/6, 137
TZad5      FUNCTION      P2, D6
1, 9/2, 13/3, 7/4, 5/5, 11/6, 16
;          Определение функций для расчета данных о донесениях
TDon1      FUNCTION      RN1, D8
. 1, 1/. 23, 2/. 31, 3/. 43, 4/. 52, 5/. 69, 6/. 82, 7/1, 8
TDon2      FUNCTION      P2, D8
1, 420/2, 510/3, 630/4, 920/5, 1200/6, 730/7, 340/8, 480
TDon3      FUNCTION      P2, D8
1, 60/2, 90/3, 120/4, 170/5, 230/6, 140/7, 60/8, 80
TDon4      FUNCTION      P2, D8
1, 8/2, 11/3, 6/4, 16/5, 10/6, 13/7, 3/8, 9
;          Определение таблиц
Don         TABLE       M1, 120, 300, 40; Таблица времени обработки донесений
Zad         TABLE       M1, 3600, 600, 50 ; Таблица времени решения задач
Obvrem      TABLE       M1, 1600, 600, 50 ; Таблица общего времени обработки
;          Определение арифметических переменных для расчета
;          количества знаков в донесении
Kzd         VARIABLE      INT(FN$Tdon2+FN$Tdon3#FN$SNorm)
;          количества диалоговых точек
Kdt         VARIABLE      INT(FN$TZad2+FN$TZad3#FN$SNorm)
Skzd        VARIABLE      INT(X$Vzd/N$KolDon); Среднее число знаков в донесении
; Сегмент имитации обработки донесений
KolDon      GENERATE      (Swd#FN$XpDis) ; Источник донесений
            ASSIGN        2, FN$TDon1 ; Определение вида донесения
            ASSIGN        3, V$Kzd ; Определение количества знаков в донесении
            SAVEVALUE      Vzd+, P3 ; Счет количества знаков в донесениях
            SEIZE          Ewm ; Занятие ЭВМ
            ADVANCE        (FN$TDon4#FN$XpDis) ; Обработка донесения
            RELEASE        Ewm ; Освобождение ЭВМ
            TABULATE       Don ; Табулирование времени обработки донесений
            TRANSFER        ,Bbb2
; Сегмент имитации обработки задач
            GENERATE      (Swz#FN$XpDis) ; Источник задач
            ASSIGN        2, FN$TZad1 ; Определение типа задачи
            ASSIGN        3, V$Kdt ; Определение числа точек диалога
Bbb1        SEIZE          Ewm ; Занятие ЭВМ
            ADVANCE        (FN$TZad4#FN$XpDis) ; Ведение диалога
            ADVANCE        (FN$TZad5#FN$XpDis) ; Обработка на ЭВМ после диалога
            RELEASE        Ewm ; Освобождение ЭВМ

```

```

      LOOP      3,Bbb1      ; Организация цикла по числу точек диалога
      TABULATE  Zad         ; Табулирование времени решения задач
Bbb2  TABULATE  ObVrem      ; Табулирование общего времени обработки
      TERMINATE                ; Вывод транзактов из модели
; Сегмент задания времени моделирования
      GENERATE  VrMod ; Задание времени моделирования, 1 ед. мод. вр.=1 с
      SAVEVALUE Skzd,V$Skzd ; Расчет среднего количества знаков в донесении
      TERMINATE 1

```

Выполним по 5 000 прогонов каждого варианта и сравним полученные результаты. Коэффициент использования ЭВМ, среднее время обработки одного транзакта и среднее количество знаков в одном донесении для первого и второго вариантов модели составили 1; 98,101; 631 и 1; 98,126; 631 соответственно, т. е. практически они одинаковы. Затраты машинного времени для первого варианта равны 5 мин 35 с. Для второго варианта, в котором не использовались встроенные генераторы случайных чисел, это время равно 5 мин 14 с, т. е. на 21 с меньше.

Пример 6.2. На вход СМО с тремя каналами обслуживания и ограниченными длинами очередей поступает пуассоновский поток заявок со средним интервалом времени 2,5 мин. Заявки могут быть трех категорий. Вероятности поступления заявок 1, 2 и 3-й категорий 0,25, 0,45 и 0,3 соответственно. Среднее время обслуживания заявок составляет 3,5, 4,2 и 7 мин соответственно. Чем меньше среднее время обслуживания заявки, тем выше ее приоритет.

Построить имитационную модель с целью определения в течение 3 ч функционирования СМО среднего времени ожидания обслуживания заявками каждой категории, а также вероятностей их обслуживания и потерь в зависимости от длин очередей.

Программа модели приведена ниже.

; Пример 6.2

; Сегмент задания исходных данных

```

Can3      STORAGE      3          ; Задания емкости МКУ
;
; описание функций задания
Kat       FUNCTION      RN5,D3     ; Вероятностей поступления заявок
.25,1/.7,2/1,3
WrObs     FUNCTION      P1,D3      ; Среднего времени обслуживания заявок
1,180/2,252/3,420
Dlina     FUNCTION      P1,D3      ; Длин очередей
1,4/2,4/3,4
Clas      FUNCTION      P1,D3      ; Определение приоритета заявки
1,3/2,2/3,1
VMod      EQU           10800.0    ; Время моделирования, 1 ед. мод. времени = 1 с
SrInt     EQU           60.0       ; Средний интервал времени поступления заявок

```

```

;      Описание арифметических выражений
;      определения вероятностей обработки заявок
Ver      FVARIABLE   N$Wse/N$Wxod      ; Любой категории
Ver1     FVARIABLE   N$Term1/X1        ; 1-й категории
Ver2     FVARIABLE   N$Term2/X2        ; 2-й категории
Ver3     FVARIABLE   N$Term3/X3        ; 3-й категории
;      определения вероятностей потерь заявок
Vpot     FVARIABLE   N$PotWse/N$Wxod   ; Любой категории
Vpot1    FVARIABLE   N$Pot1/X1         ; 1-й категории
Vpot2    FVARIABLE   N$Pot2/X2         ; 2-й категории
Vpot3    FVARIABLE   N$Pot3/X3         ; 3-й категории
; Сегмент имитации СМО
Wxod     GENERATE    (Exponential(3,0,SrInt)); Источник потока заявок
Wxod     ASSIGN      1,FN$Kat           ; Розыгрыш категории заявки
Wxod     PRIORITY    FN$Clas           ; Изменение приоритета заявки
Wxod     SAVEVALUE   P1+,1             ; Счет заявок по категориям
;      определение категорий и проверка длин очередей заявок
Met3     TEST E      P1,1,Met3          ; 1-й категории?
Met3     TEST L      Q1,FN$Dlina,PotWse ; Есть место в очереди?
Met3     TRANSFER    ,Met5              ; Отправка заявки на обслуживание
Met3     TEST E      P1,2,Met4          ; 2-й категории?
Met3     TEST L      Q2,FN$Dlina,PotWse ; Есть место в очереди?
Met3     TRANSFER    ,Met5              ; Отправка заявки на обслуживание
Met4     TEST L      Q3,FN$Dlina,PotWse ; Есть место в очереди?
Met4     QUEUE       Oth                ; Встать в общую очередь заявок
Met5     QUEUE       P1                 ; Встать в очереди по категориям
Met5     ENTER       Can3               ; Занять свободный канал
Met5     DEPART      Oth                ; Покинуть общую очередь заявок
Met5     DEPART      P1                 ; Покинуть очереди по категориям
Met5     ADVANCE     (Exponential(1,0,FN$WrObs)) ; Обслуживание
Met5     LEAVE       CAN3               ; Освободить канал обслуживания
;      счет обслуженных заявок и удаление их из модели
Wse      TEST E      P1,1,Met1          ; 1-й категории?
Term1    TERMINATE   ; Удаление обслуженных заявок 1-й категории
Met1     TEST E      P1,2,Term3         ; 2-й категории?
Term2    TERMINATE   ; Удаление обслуженных заявок 2-й категории
Term3    TERMINATE   ; Удаление обслуженных заявок 3-й категории
;      счет потерянных заявок и удаление их из модели
PotWse   TEST E      P1,1,Met2          ; 1-й категории?
Pot1     TERMINATE   ; Удаление потерянных заявок 1-й категории
Met2     TEST E      P1,2,Pot3          ; 2-й категории?
Pot2     TERMINATE   ; Удаление потерянных заявок 2-й категории
Pot3     TERMINATE   ; Удаление потерянных заявок 3-й категории

```

```
; Сегмент задания времени моделирования и расчета результатов
GENERATE VMod ; Время одного прогона модели
; расчет вероятностей обслуживания заявок
SAVEVALUE Ver11,V$Ver1 ; 1-й категории
SAVEVALUE Ver21,V$Ver2 ; 2-й категории
SAVEVALUE Ver31,V$Ver3 ; 3-й категории
SAVEVALUE VerWse,V$Ver ; Любой категории
; расчет вероятностей потерь заявок
SAVEVALUE VPot11,V$VPot1 ; 1-й категории
SAVEVALUE VPot21,V$VPot2 ; 2-й категории
SAVEVALUE VPot31,V$VPot3 ; 3-й категории
SAVEVALUE VsePot,V$VPot ; Любой категории
TERMINATE 1
START 1000,NP
CLEAR
START 5000
```

Приведем результаты моделирования, полученные после 5000 прогонов.

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
1	4	1	223084	3180	0.499	120.683	122.428	0
2	4	0	320973	5757	1.925	323.858	329.773	0
3	4	1	94596	3731	3.284	1874.888	1951.872	0
OTH	12	2	638653	12668	5.708	482.624	492.390	0

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.UTIL.	RETRY	DELAY
CAN3	3	0	0	3	638651	1	2.981	0.994	0 2

SAVEVALUE	RETRY	VALUE
1	0	225202.000
2	0	405930.000
3	0	269676.000
VER11	0	0.991
VER21	0	0.791
VER31	0	0.351
VERWSE	0	0.709
VPOT11	0	0.009
VPOT21	0	0.209
VPOT31	0	0.649
VSEPOT	0	0.291

Право интерпретировать их предоставим пользователю с помощью приложения 4. Это будет способствовать приобретению навыков быстрого и грамотного чтения стандартных отчетов, формируемых системой GPSS World.

7. Особенности применения GPSS для моделирования сложных систем

Рассмотрим назначение и методы применения тех блоков, кроме уже известных, которые могут быть необходимы при построении имитационных моделей.

7.1. Применение в моделях блоков создания копий транзактов

Блок GENERATE является основным средством создания транзактов и ввода их в модель. Вход транзактов в блок GENERATE не допускается. Кроме блока GENERATE для создания транзактов используется блок SPLIT (расщепить). Но, в отличие от GENERATE, блок SPLIT не создает самостоятельных транзактов, а лишь генерирует заданное число копий входящего в него транзакта — порождающего. Блок SPLIT имеет следующий формат:

SPLIT A, [B], [C]

Операнд A — число создаваемых копий одного семейства. Он может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА*параметр. Все копии формируются в момент входа порождающего транзакта в блок SPLIT.

Операнд B — номер блока, к которому переходят копии порождающего транзакта. Значение операнда B вычисляется для каждой копии отдельно.

После прохождения блока SPLIT порождающий транзакт направляется в следующий блок, а все копии направляются по адресу, указанному операндом B (в том числе и к следующему блоку, если операнд B не указан — по умолчанию). Таким образом, если операнд A равен k , то из блока выйдут $k+1$ транзакт. Далее порождающий транзакт и копии являются равноправными и могут проходить снова через любое число блоков SPLIT.

Операндом C может быть задан номер параметра, используемого для присвоения копиям последовательных номеров (нумерации). Если операнд C не используется, по умолчанию номеров выходящих из блока SPLIT транзактов в их параметрах нет.

Операнды B и C могут быть такими же, как и операнд A.

Приведем примеры.

SPLIT	6
SPLIT	9, Met11
SPLIT	4, Met1, 5

В первом примере из блока SPLIT выйдет $1+6=7$ транзактов. Шесть копий вслед за порождающим транзактом будут направлены к следующему блоку, так как операнд В не используется. Записи номеров в параметр порождающего и копий транзактов также не будет — операнд С опущен. Копии имеют тот же приоритет, значения параметров и время входа в модель, что и порождающий транзакт.

Во втором примере блок SPLIT сгенерирует $1+9=10$ транзактов. Порождающий транзакт перейдет к следующему блоку, а девять копий — к блоку с меткой Met11. Записи номеров в параметры транзактов также нет.

В третьем примере блок SPLIT при входе порождающего (активного) транзакта генерирует четыре копии, которые направляются к блоку с меткой Met1. Каждая копия имеет тот же приоритет, время входа в модель и значения параметров, что и порождающий транзакт, за исключением параметра номер 5. В параметр номер пять каждой очередной копии будет записан порядковый номер. Если параметр номер 5 порождающего транзакта не определен заранее, он будет создан, и ему будет присвоено значение 0.

Предположим, что в данном примере значение параметра номер 5 порождающего транзакта равно 0. После выхода из блока SPLIT в параметр номер 5 порождающего транзакта будет записано 1, а в параметр номер 5 копий 2, 3, 4 и 5 соответственно.

Пусть теперь параметр номер 5 порождающего транзакта определен заранее и его значение равно n . Тогда в параметре номер 5 порождающего транзакта и копий после выхода из блока SPLIT будут записаны $n+1$, $n+2$, $n+3$, $n+4$ и $n+5$ соответственно.

Приведем еще один пример, демонстрирующий возможность направления копий транзактов к последовательно расположенным в модели блокам.

SPLIT	3, P\$Adr, Adr

Блок, к которому должны направляться копии транзактов, указывается операндом В. Значение операнда В, т. е. номер блока, вычисляется для каждой вновь создаваемой копии. Если параметр с именем Adr содержит номер блока, предположим n , то первая копия будет направлена к блоку $n+2$, вторая — к $n+3$, третья — к $n+4$. Эти же номера будут записаны в параметр с именем Adr копий. Порождающий транзакт со значением $n+1$ параметра с именем Adr перейдет к следующему блоку.

Каждая новая копия, создаваемая блоком SPLIT, становится членом семейства (ансамбля) транзактов, порожденных исходным транзактом блока GENERATE. Транзакты, принадлежащие к одному семейству, объединяются планиров-

щиком в список. Если блоком SPLIT не производить нумерацию порождающего транзакта и его копий, то внутри семейства нельзя установить, какой из транзактов является порождающим. Число транзактов в семействе произвольно. Каждый транзакт, созданный блоком GENERATE, является отдельным семейством (номер транзакта равен номеру семейства, т. е. сколько сгенерировано блоком GENERATE в модели транзактов, столько и семейств). Таким образом, число семейств (ансамблей) в системе произвольно и семейство существует до тех пор, пока в нем находится хотя бы один транзакт.

К семейству транзактов можно применять блоки, осуществляющие обработку транзактов, например, MATCH, ASSEMBLE, GATHER.

7.2. Организация в моделях синхронизации движения транзактов

7.2.1. Блок ASSEMBLE

Блок ASSEMBLE (объединить) используется для объединения заданного числа транзактов, принадлежащих одному семейству, в один транзакт (т. е. осуществляет сборку заданного числа транзактов). После сборки из блока ASSEMBLE выходит только один транзакт, который переходит в следующий по номеру блок. Блок ASSEMBLE имеет следующий формат:

ASSEMBLE A

Операнд A — число транзактов одного семейства, участвующих в сборке (начальное число счетчика сборки). Может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА*параметр.

В одном и том же блоке ASSEMBLE возможна одновременная сборка транзактов нескольких семейств. Так как сборка длится в течение какого-то промежутка модельного времени, блок ASSEMBLE имеет список синхронизации. В список синхронизации помещаются первые вошедшие в блок ASSEMBLE транзакты каждого собираемого семейства. Они ожидают прихода транзактов своих семейств.

При входе транзакта в блок ASSEMBLE проверяется: есть ли в списке синхронизации транзакт такого же семейства? Возможны два случая: транзакта данного семейства в списке синхронизации нет или он есть.

Пусть транзакта такого же семейства в списке синхронизации нет, т. е. вошедший транзакт первый из собираемого семейства. Тогда вычисляется операнд A, округляется до целого числа и уменьшается на 1 (один транзакт уже прибыл). В зависимости от полученного результата S, сохраняемого в ячейке (счетчике сборки) первого транзакта собираемого семейства, выполняются следующие действия:

- $S < 0$ — происходит останов по ошибке «Счетчик сборки не положителен»;
- $S = 0$ — транзакт сразу пытается войти в следующий блок (т. е. нужно было собрать только один транзакт);

○ $S \geq 1$ — транзакт исключается из списка текущих событий и помещается в список синхронизации блока ASSEMBLE.

Теперь предположим, что в списке синхронизации уже есть транзакт того же семейства, что и вошедшего в блок ASSEMBLE. Прибывший транзакт уничтожается, а счетчик сборки, сохраненный в ячейке находящегося в списке синхронизации транзакта, уменьшается на 1. Когда этот счетчик сборки становится равным нулю, т. е. в блок ASSEMBLE вошло заданное число транзактов, ожидающий транзакт выводится из списка синхронизации. Если обслуживание этого транзакта не было прервано ни одним из устройств, он пытается перейти к следующему блоку и в случае успеха возвращается в список текущих событий.

Например:

ASSEMBLE 7

собирает 7 транзактов одного семейства, 6 уничтожаются, один переходит в следующий блок.

ASSEMBLE *1

собирает число транзактов, равное значению параметра 1 первого пришедшего в блок ASSEMBLE транзакта семейства. Этим способом задания счетчика сборки удобно пользоваться, когда неизвестно заранее количество собираемых транзактов. Оно определяется в ходе моделирования. Следовательно, нужно в параметр номер один всех собираемых транзактов записать значение счетчика сборки, так как неизвестно, какой транзакт придет первым и обновит счетчик сборки.

ЗАМЕЧАНИЕ

Прерванному транзакту запрещается покидать блок ASSEMBLE до тех пор, пока все его прерванные обслуживания не будут закончены. Поэтому рекомендуется строить модель так, чтобы транзакты, обслуживание которых было прервано без удаления (без освобождения устройства), не входили в блок ASSEMBLE.

ЗАМЕЧАНИЕ

Записи операнда A

ASSEMBLE	0.6
ASSEMBLE	-0.6
ASSEMBLE	5.6

на этапе трансляции не обнаруживаются. Первые две записи в ходе моделирования приводят к останову по ошибке «Счетчик сборки не положителен».

Рассмотрим пример совместного использования блоков SPLIT и ASSEMBLE.

Пример 7.1. На пункт управления (ПУ) соединения поступают из частей группы сообщений. На ПУ имеются 4 автоматизированных рабочих места (АРМ). Интервалы времени поступления групп сообщений подчинены экспоненциальному закону со средним значением 5 часов. Количество сообщений в груп-

не распределено по нормальному закону с математическим ожиданием 20 сообщений и стандартным отклонением 3 сообщения. Время обработки одного сообщения на одном АРМ распределяется по экспоненциальному закону со средним значением 25 мин. После обработки сообщения вновь собираются и хранятся вместе. Ограничений на количество обрабатываемых сообщений нет.

Необходимо построить модель функционирования ПУ в течение трех суток с целью определения коэффициента загрузки АРМ.

Для имитации группы сообщений и одного сообщения следует использовать транзакты. АРМ в модели будет представлять МКУ.

Общая идея построения модели заключается в следующем. Блок GENERATE имитирует поступление транзактов. Один транзакт — одна группа. Затем случайным образом определяется количество сообщений в группе. Далее поступившая группа сообщений — исходный (порождающий) транзакт копируется блоком SPLIT и результат копирования — число транзактов по числу сообщений в группе, поступают на АРМ — МКУ и обрабатываются. После обработки сообщения, входящие в группу, собираются блоком ASSEMBLE. Когда все транзакты-сообщения будут обработаны и поступят в блок ASSEMBLE, из него выйдет один транзакт, имитирующий одну группу сообщений.

Приведем программу модели.

; Пример 7.1

; Задание исходных данных

Tpt	EQU	300	; Средний интервал поступления групп сообщений
Moka	EQU	30	; Матожидание числа сообщений в группе
Soka	EQU	5	; Стандартное отклонение числа сообщений в группе
Tpoa	EQU	25	; Среднее время обработки одного сообщения
VrMod	EQU	4320	; Время моделирования
P0br	STORAGE	4	; Число каналов МКУ

; Сегмент имитации поступления и обработки сообщений

GENERATE	(Exponential(13,0,Tpt))	; Источник групп сообщений
ASSIGN	1,(Normal(451,Moka,Soka))	; Определение числа сообщений в группе
;		
SPLIT	*1	; Деление группы на сообщения
QUEUE	Rr1	; Занять очередь
ENTER	P0br	; Занять канал МКУ
DEPART	Rr1	; Покинуть очередь
ADVANCE	(Exponential(13,0,Tpoa))	; Обработка сообщения
LEAVE	P0br	; Освободить один канал МКУ
ASSEMBLE	*1	; Собрать сообщения в группу
TERMINATE		; Обработанные группы сообщений

; Сегмент задания времени моделирования

GENERATE	VrMod
TERMINATE	1

В блоке ASSIGN случайным образом определяется количество сообщений в поступившей группе и запоминается в первом параметре транзакта. Сделано это потому, что в блоке ASSEMBLE использована запись ASSEMBLE *1, значит, он собирает число транзактов, равное значению параметра 1 первого пришедшего в блок ASSEMBLE транзакта семейства. А какой транзакт придет первым после обработки на МКУ PP1 — исходный или копия — неизвестно.

По результатам моделирования коэффициент использования АРМ составляет 0,631.

7.2.2. Блок GATHER

Блок GATHER (собирать) предназначен для сбора заданного количества транзактов одного семейства, движущихся по одному и тому же пути. Отличие блока GATHER от блока ASSEMBLE состоит в том, что в нем транзакты не уничтожаются, а после сбора все направляются к следующему блоку.

Формат блока:

GATHER A

Операнд A задает число транзактов, принадлежащих одному семейству, которое нужно собрать при их движении по одному пути (начальное число счетчика сбора). Операнд может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА*параметр.

Блок GATHER также может одновременно производить сбор транзактов нескольких семейств и также имеет список синхронизации.

Пусть при входе транзакта в блок GATHER в результате проверки оказалось, что в его списке синхронизации нет транзактов этого же семейства, т. е. вошедший транзакт из данного семейства является первым. Тогда вычисляется операнд A, округляется до целого числа, уменьшается на 1 и запоминается полученный результат S в ячейке транзакта — счетчике сбора. В зависимости от результата S возможны следующие действия:

- $S < 0$ — происходит останов по ошибке «Счетчик сбора не положителен»;
- $S = 0$ — транзакт сразу пытается войти в следующий блок (надо было собрать только один транзакт);
- $S \geq 1$ — транзакт помещается в список синхронизации блока для ожидания прибытия других транзактов своего семейства.

Теперь предположим, что проверкой в списке синхронизации обнаружены транзакты семейства, к которому принадлежит только что вошедший в блок GATHER транзакт. Он также помещается в список синхронизации, а счетчик сбора уменьшается на 1. Если его значение становится равным нулю, т. е. в блок GATHER вошло заданное число транзактов данного семейства, все собранные транзакты исключаются из списка синхронизации и помещаются в список текущих событий.

ЗАМЕЧАНИЕ

Если среди собранных транзактов одного семейства обнаруживаются прерванные транзакты, то после окончания сбора они не переводятся в СТС. Прерванные транзакты будут находиться в блоке GATHER, но не в списке синхронизации, до тех пор, пока все прерванные обслуживания не будут закончены. Поэтому в модели нужно предусмотреть, чтобы прерванные без удаления, т. е. без освобождения устройства, транзакты, если в этом нет необходимости, не входили в блок GATHER.

ЗАМЕЧАНИЕ

Записи операнда A

GATHER	-0.6
GATHER	-0.6
GATHER	5.6

на этапе трансляции не обнаруживаются. Первые две записи в ходе моделирования приводят к останову по ошибке «Счетчик сборки не положителен».

Рассмотрим пример использования блоков SPLIT, ASSEMBLE и GATHER.

Пример 7.2. В ремонтной части производится ремонт средств связи (СС) путем замены неисправных блоков исправными. Всего среднее количество неисправных блоков различных типов в одном образце СС равно 17. Количество неисправных блоков в образце СС распределено по нормальному закону со стандартным отклонением 3 блока. Среднее время интервалов поступления образцов СС, распределенное по экспоненциальному закону, равно 300 мин. При поступлении образца СС первой группой специалистов-ремонтников производится контроль всех неисправных блоков, и только потом осуществляется их замена второй группой специалистов-ремонтников. Время контроля блока и время его замены распределены по экспоненциальному закону. Среднее время контроля блока равно 15 мин, а среднее время его замены — 10 мин. Необходимо построить модель функционирования ремонтной части при ремонте СС с целью определения коэффициентов загрузки каждой группы специалистов-ремонтников.

В модели транзакты будут представлять СС и блоки, а одноканальное устройство — одну группу специалистов-ремонтников.

За единицу модельного времени взята 1 сек. Моделирование осуществляется в течение трех суток.

Ниже представлена программа модели.

; Пример 7.2

; Задание исходных данных

Tt1	EQU	18000	; Средний интервал поступления СС
Tt2	EQU	17	; Среднее число неисправных блоков в СС
Tt3	EQU	3	; Стандартное отклонение числа неисправных блоков в СС
Tt4	EQU	900	; Среднее время замены блока первой группой
Tt5	EQU	600	; Среднее время замены блока второй группой
VrMod	EQU	259200	; Время моделирования, 1 сек = 1 ед. мод. времени

; Сегмент имитации поступления и обработки неисправных СС

```
GENERATE (Exponential(427,0,Tt1)); Источник СС
ASSIGN 1,(Normal(521,Tt2,Tt3)) ; Розыгрыш неисправных блоков
SPLIT P1 ; Выделение неисправных блоков
SEIZE Gr1 ; Занять первую ремонтную группу
ADVANCE (Exponential(427,0,Tt4)); Контроль первой группой
RELEASE Gr1 ; Освободить первую ремонтную группу
GATHER *1 ; Собрать отремонтированные блоки одного СС
SEIZE Gr2 ; Занять вторую ремонтную группу
ADVANCE (Exponential(427,0,Tt5)); Ремонт второй группой
RELEASE Gr2 ; Освободить вторую ремонтную группу
ASSEMBLE *1 ; Собрать СС
TERMINATE ; Отремонтированные СС
```

; Сегмент задания времени моделирования

```
GENERATE VrMod
TERMINATE 1
```

Блок GATHER задерживает все подвергнувшиеся контролю блоки (транзакты), количество которых хранится в параметре P1 транзакта, и одновременно направляет их на одноканальное устройство Gr2, моделирующее работу второй группы специалистов-ремонтников. Блок ASSEMBLE собирает все транзакты (СС) и на его выходе появляется один транзакт — отремонтированный образец средства связи.

Коэффициенты загрузки первой и второй групп ремонтников по результатам моделирования равны 0,847 и 0,524 соответственно.

7.2.3. Блок MATCH

Блок MATCH (синхронизировать) предназначен для синхронизации движения транзактов одного семейства, продвигающихся по разным путям. Для синхронизации необходимы два блока MATCH, находящихся в соответствующих местах модели и называемых сопряженными. Формат записи:

```
Name MATCH A
```

Операндом А каждого блока MATCH указывается метка или номер сопряженного ему блока. Операнд может быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*параметр. Например:

```
Kont1 MATCH Kont2
Kont2 MATCH Kont1
```

В модели эти блоки будут помещены отдельно в параллельных путях продвижения транзактов.

При входе транзакта в блок MATCH операнд А вычисляется и округляется до целого числа. По полученному значению определяется сопряженный блок MATCH. При его отсутствии происходит останов по ошибке.

Если сопряженный блок есть, проверяется наличие в нем транзакта из того же семейства. Если в блоке нет ни одного транзакта данного семейства, поступивший транзакт помещается в список синхронизации и будет ожидать в нем входа транзакта своего семейства в сопряженный блок MATCH.

При поступлении такого транзакта в сопряженный блок MATCH оба транзакта одного семейства будут исключены из списка синхронизации и одновременно будут пропущены в следующие за блоком MATCH блоки.

Одна и та же пара блоков MATCH может одновременно синхронизировать любое число пар транзактов из разных семейств. Транзакты одного семейства также можно синхронизировать в любом числе пар блоков MATCH.

ЗАМЕЧАНИЕ

Если один из синхронизируемых транзактов будучи прерванным вошел в блок MATCH, ему не разрешается выйти из него до тех пор, пока все его прерванные обслуживания не будут закончены. Поэтому строить модель нужно так, чтобы транзакты, обслуживание которых прервано без удаления (без освобождения устройств), не входили в блоки MATCH.

ЗАМЕЧАНИЕ

Блок MATCH может быть сопряжен сам себе. Например:

Six MATCH Six

При этом его действие будет аналогично действию блока GATHER с операндом A, равным 2.

Приведем пример применения блока MATCH.

Пример 7.3. Техническое обслуживание (ТО) средств связи (СС) проводят в два этапа две группы специалистов. Первый этап выполняется специалистами первой и второй групп параллельно со средним временем соответственно 80 мин и 70 мин. После первого этапа производится уточнение мероприятий ТО, время выполнения которого принимается равным нулю. После уточнения специалистами первой и второй групп параллельно выполняется второй этап со средним временем 30 мин и 40 мин соответственно. Затем третья группа специалистов производит комплексную проверку СС со средним временем 60 мин. Интервалы поступления СС и время их ТО подчиняются экспоненциальному закону.

Построить модель ТО средств связи с целью определения коэффициентов занятости групп специалистов.

В модели группы специалистов целесообразно представить ОКУ, а СС — транзактами. Пусть первой группе специалистов соответствует ОКУ Gto1, второй группе — Gto2, третьей группе — Gto3.

Программа модели представлена ниже.

; Пример 7.3

; Задание исходных данных

Post	EQU	11400 ; Интервал поступления СС на ТО
Obs2	EQU	4800 ; Среднее время ТО первой группой на этапе 1


```

Obs3      EQU      4200 ; Среднее время T0 второй группой на этапе 1
Obs4      EQU      1800 ; Среднее время T0 первой группой на этапе 2
Obs5      EQU      2400 ; Среднее время T0 второй группой на этапе 2
Obs6      EQU      3600 ; Среднее время T0 третьей группой на этапе 3
; Сегмент имитации поступления средств связи на T0
      GENERATE      (Exponential(215,0,Post)); Источник средств связи
      SPLIT         1,Anb ; Разделение на два транзакта
; Сегмент имитации работы первой группы специалистов
      SEIZE         Tg1 ; Занять первую группу специалистов
      ADVANCE       (Exponential(215,0,Obs2)); Первый этап обслуживания
Par1      MATCH     Par2 ; Уточнение мероприятий T0
      ADVANCE       (Exponential(215,0,Obs4)); Второй этап обслуживания
      RELEASE       Tg1 ; Освободить первую группу специалистов
      TRANSFER      ,Ank
; Сегмент имитации работы второй группы специалистов
Anb       SEIZE     Tg2 ; Занять вторую группу специалистов
      ADVANCE       (Exponential(215,0,Obs3)); Первый этап обслуживания
Par2      MATCH     Par1
      ADVANCE       (Exponential(215,0,Obs5)); Второй этап обслуживания
      RELEASE       Tg2 ; Освободить вторую группу специалистов
Ank       ASSEMBLE  2 ; Передача CC третьей группе специалистов
; Сегмент имитации работы третьей группы специалистов
      SEIZE         Tg3 ; Занять третью группу специалистов
      ADVANCE       (Exponential(215,0,Obs6)); Третий этап обслуживания
      RELEASE       Tg3 ; Освободить третью группу специалистов
      TERMINATE     1 ; Готовые CC

```

После поступления в модель исходного транзакта — одного образца средства связи — осуществляется расщепление его с помощью блока SPLIT на два транзакта. Первый из них направляется на цепочку блоков, имитирующих работу специалистов первой группы, а второй (копия) — на блоки, моделирующие работу специалистов второй группы.

После выполнения первого этапа обслуживания специалистами обеих групп оба транзакта задерживаются блоками MATCH и одновременно начинается второй этап работ. После его выполнения два транзакта блоком ASSEMBLE объединяются в один и далее со CC имитируется проведение комплексной проверки.

ЗАМЕЧАНИЕ

Программа модели не имеет второго сегмента, определяющего время моделирования. Она завершит свою работу после имитации T0 такого количества CC, какое будет указано в команде START. Это число занесется в счетчик завершений, а каждый транзакт, входя в блок TERMINATE, будет уменьшать его содержимое на 1.

7.3. Уменьшение числа объектов в модели методом косвенной адресации

В рассматривавшихся до сих пор примерах моделей ссылки на различные объекты GPSS производились в основном по данным им именам. Такая *прямая* адресация удобна, когда речь идет о небольшом числе объектов каждого типа. Если же число объектов некоторого типа велико, то с целью уменьшения программы модели используют ссылки на эти объекты по их номерам, применяя косвенную адресацию.

Идея *косвенной адресации* состоит в том, что каждый транзакт в некотором своем параметре содержит номер того или иного объекта, а в операндах блоков, адресующихся к объектам, записывается ссылка на этот параметр транзакта. Общий вид косвенной адресации был дан в п. 1.5.2: **СЧА*параметр**.

Например:

SEIZE P*X1

Занять ОКУ, номер которого содержится в параметре; номер этого параметра определяется значением ячейки X1.

SAVEVALUE 1, X*P2

Поместить в ячейку с номером 1 значение, содержащееся в ячейке, номер которой определяется значением параметра 2 транзакта.

Проиллюстрируем применение косвенной адресации на примерах.

Пример 7.4. На вход многоканальной СМО с тремя каналами обслуживания поступает экспоненциальный поток заявок со средним интервалом поступления 140 единиц модельного времени. Каждая заявка с равной вероятностью 0,2 относится к одному из пяти видов: 1, 2, 3, 4 или 5. Среднее время обслуживания заявок каждого типа составляет соответственно 90, 100, 110, 120 и 130 единиц модельного времени. Чем меньше среднее время обслуживания заявки, тем выше ее приоритет. Необходимо построить модель, позволяющую оценить средние значения времени ожидания заявок каждого вида, а также распределения общего времени ожидания в очереди и общего времени пребывания в системе.

Программа модели представлена ниже.

; Пример 7.4

; Задание исходных данных

Wid	FUNCTION	RN21,05	; Вид заявки
.2, 1/.4, 2/.6, 3/.8, 4/1, 5			
Mean	FUNCTION	P\$WidZ, L5	; Среднее время обслуживания
1, 90/2, 100/3, 110/4, 120/5, 130			
PriZ	VARIABLE	6-P\$WidZ	; Приоритет заявки
Pered	STORAGE	3	; Емкость МКУ
VTime	QTABLE	Ocher, 50, 50, 10	; Определение таблицы
TTime	TABLE	M1, 100, 100, 15	; Определение таблицы

; Сегмент поступления и обслуживания заявок

```

GENERATE      (Exponential(32,0,140)); Источник заявок
ASSIGN        WidZ, FN$Wid      ; Определение вида заявки
PRIORITY      V$PriZ            ; Определение приоритета заявки
QUEUE         Ocher             ; Встать в общую очередь
QUEUE         P$WidZ            ; Встать в очередь по видам заявок
ENTER         Pered             ; Занять канал МКУ
DEPART        P$WidZ            ; Покинуть очередь по видам заявок
DEPART        Ocher             ; Покинуть общую очередь
ADVANCE       (Exponential(45,0, FN$Mean)); Обслуживание
LEAVE         Pered             ; Освободить канал МКУ
TABULATE      TTime            ; Поместить в таблицу
TERMINATE     ; Обслуженные заявки

```

; Сегмент задания времени моделирования

```

GENERATE      4320
TERMINATE     1

```

В модели функция Wid задает вид заявки, а функция Mean — среднее время обработки заявки в зависимости от ее вида. Переменная PriZ служит для вычисления приоритета транзакта как функции ее вида, содержащегося в параметре с именем WidZ. Транзакты вида 1 ($P\$WidZ=1$) получают приоритет 5, транзакты вида 2 — приоритет 4, транзакты вида 3 — приоритет 3, транзакты вида 4 — приоритет 2, транзакты вида 5 — приоритет 1.

В блоке ASSIGN в параметр WidZ транзактов записывается вид заявки, разыгрываемый с помощью функции Wid. В следующем блоке PRIORITY посредством переменной PriZ определяется приоритет транзактов, первоначально равный 0 (отсутствует операнд E в блоке GENERATE). Далее каждый транзакт «отмечается» в блоках QUEUE в двух очередях. Очередь с именем Ocher является общей для транзактов всех видов. Входя в следующий блок QUEUE, транзакт «отмечается» в очереди с номером 1, 2, 3, 4 или 5 в зависимости от вида заявки, записанного в параметре WidZ. Аналогичным образом фиксируется уход из очередей в блоках DEPART. Таким образом, в модели создается шесть объектов типа «очередь»: одна очередь с именем Ocher и пять с номерами 1, 2, 3, 4 и 5. При этом последние пять очередей создаются одной парой блоков QUEUE — DEPART. В этом и состоит эффект косвенной адресации.

Как уже отмечалось ранее, каждому имени объекта планировщик сам ставит в соответствие некоторый номер. При ссылках на объекты одного и того же типа одновременно по именам и номерам, как это имеет место в рассматриваемом примере, существует опасность параллельной адресации к одному и тому же объекту вместо двух разных или, наоборот, к двум разным вместо одного. Так, в рассматриваемой модели мы вообще-то не знаем, какой именно номер поставит сам планировщик в соответствие имени очереди Ocher. Если этот номер будет от 1 до 5, то это приведет к ошибке, так как в модели ока-

жется не шесть очередей, а пять, причем в одну из них будет заноситься информация как обо всех транзактах, так и дополнительно о транзактах одного из пяти видов. Как избежать такой ситуации?

В большинстве случаев об этом можно не заботиться, поскольку планировщик ставит в соответствие именам объектов номера, начиная с 10 000. Например, в данной модели имя Ocher получит номер 10 005. При необходимости же можно воспользоваться командой EQU и самостоятельно назначить имени объекта нужный номер. Например, в рассматриваемой модели для того, чтобы очередь с именем OCHER имела номер 6, достаточно записать команду

Ocher EQU 6

Покажем преимущество косвенной адресации еще на одном примере.

Пример 7.5. На ПУ объединения из пяти соединений поступают группы сообщений. На ПУ имеются 13 отделов обработки различных видов сообщений. Для обработки каждого вида сообщений в одном отделе могут быть развернуты несколько АРМ (табл. 7.1).

Таблица 7.1

Данные о ПУ объединения

Характеристики	Коды видов сообщений												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Количество АРМ, шт.	2	3	4	3	3	5	4	5	3	6	4	3	5
Среднее время обработки, мин	6	2	3	2.5	3.5	8	7	3	3	4	6	5	4.5

Интервалы времени поступления групп сообщений каждого из пяти соединений распределены по экспоненциальному закону со средними значениями 6, 5, 3,5 и 2,5 часа соответственно. Количество видов сообщений, которые могут быть в одной группе, распределено по нормальному закону с математическим ожиданием 8 и стандартным отклонением 1. Для одного вида сообщений в одной группе их количество распределено по нормальному закону с математическим ожиданием 14 и стандартным отклонением 2. Время обработки сообщения одного вида распределено по экспоненциальному закону и зависит также от вида сообщения (см. табл. 7.1).

Построить модель функционирования АРМ пункта управления с целью обоснования рационального варианта организации работ на нем и необходимого состава сил и средств.

Программа модели приведена ниже.

; Пример 7.5

; Задание исходных данных

; *Присвоение именам МКУ – отделам номеров*

Pun1	EQU	1
Pun2	EQU	2
Pun3	EQU	3
Pun4	EQU	4
Pun5	EQU	5
Pun6	EQU	6
Pun7	EQU	7
Pun8	EQU	8
Pun9	EQU	9
Pun10	EQU	10
Pun11	EQU	11
Pun12	EQU	12
Pun13	EQU	13

; *Задание числа АРМ в отделах*

Pun1	STORAGE	2
Pun2	STORAGE	3
Pun3	STORAGE	4
Pun4	STORAGE	3
Pun5	STORAGE	3
Pun6	STORAGE	5
Pun7	STORAGE	4
Pun8	STORAGE	5
Pun9	STORAGE	3
Pun10	STORAGE	6
Pun11	STORAGE	4
Pun12	STORAGE	3
Pun13	STORAGE	5

; *Математическое ожидание и стандартное отклонение*

MatV	EQU	8	; Количества видов сообщений в группе
S0tkV	EQU	1	
MatS	EQU	15	; Количества сообщений в одной группе
S0tkS	EQU	2	

; *Интервалы поступления групп сообщений из соединений*

SInt1	EQU	360	; 1
SInt2	EQU	300	; 2
SInt3	EQU	210	; 3
SInt4	EQU	240	; 4
SInt5	EQU	150	; 5

; *Определение функций*

ObrS	FUNCTION	P3,D13	; Среднее время обработки

1,6/2,2/3,3/4,2.5/5,3.5/6,8/7,7/8,3/9,3/10,4/11,6/12,5/13,4.5

```

Otdel      FUNCTION    RN63,C2      ; Номер отдела
0,1/1,14
; Сегмент имитации потока 1 групп сообщений
      GENERATE    (Exponential(33,0,SInt1)) ; Источник потока 1
      ASSIGN      1,1 ; Код 1 – поток 1
; Сегмент распределения сообщений по отделам
Met1       ASSIGN   2,(INT(Normal(47,MatV,S0tkV)-1)); Число видов
      SPLIT        +2 ; Транзактов – по числу видов сообщений
      ASSIGN       2+,1 ; Запомнить число видов сообщений
      ASSIGN       3,(INT(FN$Otdel)) ; В какой отдел направить?
      ASSIGN       4, (INT(Normal(55,MatS,S0tkS)-1)) ; Сообщения
      SPLIT        +4 ; Транзактов – по числу сообщений
      ASSIGN       4+,1 ; Запомнить число сообщений
; Сегмент имитации обработки сообщений
Met2       QUEUE    P3 ; Встать в очередь
      ENTER        P3 ; Занять свободное АРМ
      DEPART       P3 ; Покинуть очередь
      ADVANCE      (Exponential(73,0,FN$ObrS)); Обработка
      LEAVE        P3 ; Освободить АРМ
Met3       ASSEMBLE +4 ; Собрать сообщения одного вида
      ASSEMBLE     +2 ; Собрать сообщения всех видов
Met4       TERMINATE ; Обработанные группы сообщений
; Сегмент имитации потока 2 групп сообщений
      GENERATE    (Exponential(81,0,SInt2)) ; Источник потока 2
      ASSIGN      1,2 ; Код 2 – поток 2
      TRANSFER     Met1 ; Направить в отдел
; Сегмент имитации потока 3 групп сообщений
      GENERATE    (Exponential(93,0,SInt3)) ; Источник потока 3
      ASSIGN      1,3 ; Код 3 – поток 3
      TRANSFER     Met1 ; Направить в отдел
; Сегмент имитации потока 4 групп сообщений
      GENERATE    (Exponential(52,0,SInt4)) ; Источник потока 4
      ASSIGN      1,4 ; Код 4 – поток 4
      TRANSFER     Met1 ; Направить в отдел
; Сегмент имитации потока 5 групп сообщений
      GENERATE    (Exponential(62,0,SInt4)) ; Источник потока 5
      ASSIGN      1,5 ; Код 5 – поток 5
      TRANSFER     Met1 ; Направить в отдел
; Сегмент задания времени моделирования
      GENERATE     VrMod
; Вероятности обработки
      SAVEVALUE    VObRV,(N$Met4/N$Met1) ; Групп сообщений
      SAVEVALUE    VObRS,(N$Met3/N$met2) ; Сообщений
      TERMINATE    1

```

Если не использовать косвенную адресацию и каждый отдел представить МКУ с именем, то модель увеличится более чем на 72 объекта.

В модели примера 7.5 приняты ряд условий и допущений, которые не всегда будут выполнены в реальной боевой обстановке. Считается, что сообщения поступают одновременно группами, что нет потерь ни сообщений, ни отделов, ни АРМ. Эти условия и допущения могут быть учтены. Например, для учета потерь сообщений можно использовать блок TRANSFER в режиме передачи транзакта случайным образом. Для учета потерь вида сообщений следует ввести соответствующую функцию потерь. Возможны и другие способы совершенствования рассмотренных моделей.

Применение косвенной адресации продемонстрировано также в примере 9.2.

7.4. Обработка одновременных событий

При моделировании возможно одновременное наступление двух и более событий. В этом случае говорят о существовании временного узла. *Временной узел* — это наличие более одного транзакта с одинаковым временем в СБС. В некоторых случаях временной узел может нарушить логику моделирования. Рассмотрим на примере.

Пример 7.6. В системе передачи данных осуществляется обмен пакетами сообщений между двумя узлами связи по дуплексному каналу. Пакеты поступают от абонентов с интервалами времени между ними $T1 \pm T2$ мс на 1-й и $T3 \pm T4$ мс на 2-й узлы связи. Время передачи пакета подчинено экспоненциальному закону со средним значением $T5$ мс. На узлах связи имеются накопители, которые могут хранить два пакета (включая передаваемый). При поступлении пакета в момент занятости накопителей узлам связи предоставляется выход на спутниковую полудуплексную линию связи, которая осуществляет передачу пакетов данных за время, распределенное по экспоненциальному закону, со средним значением $T6$ мс. При занятости спутниковой линии связи пакет получает отказ и теряется.

Построить модель системы передачи данных с целью определения в течение 1 мин ее функционирования частоты вызовов спутниковой линии связи и ее загрузки, а также вероятности потерь пакетов.

; Пример 7.6

; Сегмент имитации работы узла 1

	GENERATE	10,3	; Источник пакетов на узле 1
Met1	ASSIGN	2,1	; Код 1 – признак узла 1
	TEST L	Q\$UslQ1,1, Met3	; Есть место в накопителе?
	QUEUE	UslQ1	; Встать в очередь на узле 1
	SEIZE	Usl1	; Занять канал на узле 1
	DEPART	UslQ1	; Покинуть очередь на узле 1
	ADVANCE	(Exponential(313,0,10));	Передача с узла 1
	RELEASE	Usl1	; Освободить канал на узле 1
	TERMINATE		; Переданные пакеты с узла 1

; Сегмент имитации работы узла 2

	GENERATE	9,4	; Источник пакетов на узле 2
Met2	ASSIGN	2,2	; Код 2 – признак узла 2
	TEST L	Q\$Usl02,1,Met3	; Есть место в накопителе?
	QUEUE	Usl02	; Встать в очередь на узле 2
	SEIZE	Usl2	; Занять канал на узле 2
	DEPART	Usl02	; Покинуть очередь на узле 2
	ADVANCE	(Exponential(313,0,10));	Передача с узла 2
	RELEASE	Usl2	; Освободить канал на узле 2
	TERMINATE		; Переданные пакеты с узла 2

; Сегмент имитации спутниковой связи

Met3	GATE NU	Sputnik, Met4	; Свободна спутниковая связь?
	SEIZE	Sputnik	; Занять спутниковую связь
	ADVANCE	(Exponential(201,0,12));	Передача пакета
	RELEASE	Sputnik	; Освободить спутниковую связь
Met5	TRANSFER	,(Met5+P2)	
	TERMINATE		; Переданные пакеты с узла 1
	TERMINATE		; Переданные пакеты с узла 2
Met4	TERMINATE		; Потерянные пакеты

; Сегмент задания времени моделирования

	GENERATE	60000
	SAVEVALUE	VPot, ((N\$Met4/(N\$Met1+N\$Met2));
	TERMINATE	1

Предположим, что при поступлении очередного транзакта в модель накопитель на узле 1 оказался заполненным. Тогда временной узел может образоваться между одновременными событиями «поступление транзакта в модель от генератора узла 1» и «завершение обслуживания в ОКУ Sputnik».

Пусть первым в СБС расположен транзакт, освобождающий ОКУ Sputnik. Тогда вначале будет отработан этот транзакт, т. е. событие «завершение обслуживания в ОКУ Sputnik», причем условие «ОКУ Sputnik свободно», проверяемое в блоке GATE, станет истинным. Затем будет обработан транзакт, поступивший в модель из блока GENERATE узла 1. Пройдя через блоки TEST и GATE, транзакт будет впущен в блок SEIZE и ОКУ Sputnik в тот же момент модельного времени снова окажется занятым. Такая ситуация при обработке временного узла представляется естественной.

Пусть теперь первым в СБС расположен транзакт, поступающий в модель из блока GENERATE узла 1. Так как условие «ОКУ Sputnik свободно» ложно, то блок GATE направит этот транзакт в блок с именем Met4 для уничтожения. Таким образом, в модели будет зафиксирован отказ в обслуживании, хотя в этот момент модельного времени, после обработки транзакта, освобождающего спутниковую связь, ОКУ Sputnik станет свободным.

В системе GPSS World при обнаружении временного узла транзакты, попавшие в него, исключаются из СБС в случайном порядке. Для этого GPSS World

вырабатывает псевдослучайные числа с помощью генератора, заданного в поле Time Ties (Временные узлы) на странице Random Numbers (Случайные числа) журнала настроек модели. Все исключенные из СБС транзакты помещаются в СТС. Транзакт с наивысшим приоритетом становится активным транзактом.

Разрешение временных узлов случайным образом можно запретить, если указать 0 в поле Time Ties (Временные узлы) на странице Random Numbers (Случайные числа) журнала настроек модели. В этом случае в качестве критерия равенства вещественных значений модельного времени используется значение одной миллионной.

В рассматриваемом примере аналогично временной узел может образоваться, если в накопителе 2 не будет места при входе транзакта в модель от блока GENERATE узла 2 и завершении обслуживания ОКУ Sputnik. Во временном узле могут оказаться и три транзакта: два транзакта при одновременном входе в модель от обоих GENERATE узлов и транзакт, завершающий обслуживание на ОКУ Sputnik.

После 10 000 прогонов при разрешении временных узлов случайным образом и при отсутствии случайного разрешения коэффициенты загрузки каналов связи обоих узлов, спутниковой связи и вероятности потерь пакетов оказались одинаковыми: 0,781, 0,812, 0,348 и 0,081 соответственно.

Порядок расположения транзактов, соответствующих рассматриваемым событиям, в СБС случаен, и в среднем в половине случаев временной узел будет обрабатываться не так, как нужно. В результате статистика, связанная с отказами, может оказаться искаженной.

Для правильной обработки временного узла надо обеспечить такой порядок расположения транзактов в СТС, чтобы транзакт, освобождающий ОКУ Sputnik, всегда располагался первым. Этого можно добиться, управляя приоритетами транзактов.

Транзакты, поступающие в модель через блок GENERATE, имеют нулевой приоритет. Такой же приоритет имеют транзакты, получившие отказ в обслуживании и направленные в блок с меткой Met4 для уничтожения. Поэтому нужно тем транзактам, что поступают на обслуживание ОКУ Sputnik, повысить приоритет до 1 дополнительным блоком PRIORITY (после блока SEIZE Sputnik). Тогда после выхода из блока ADVANCE обслуженные транзакты возвращаются из СБС в СТС, располагаясь в начале списка. Таким образом, нужный порядок транзактов обеспечивается, и временной узел будет обработан правильно.

Неверная обработка временных узлов возможна и для моделей со списками пользователя. Рассмотрим это на примере.

Пример 7.7. На вычислительный комплекс коммутации сообщений (ВККС) поступают сообщения от четырех абонентов и далее передаются по двум каналам передачи данных со скоростью 1000 бит/с. Длительности интервалов между сообщениями от каждого абонента распределены по экспоненциальному закону с интенсивностью λ 1/с. Сообщения равновероятно могут прина-

длежать одной из двух категорий: команды (донесения) или неформализованная информация. Команды (донесения) обладают абсолютным приоритетом. Длины сообщений 1-й категории равномерно распределены в интервале [1400, 6000] байт. Длины сообщений 2-й категории распределены по нормальному закону с параметрами $m = 2 \cdot 10^3$ байт и $\sigma = 3 \cdot 10^2$ байт. Для хранения сообщений, ожидающих обработки в ВККС, предусмотрен накопитель емкостью L Мбайт.

Разработать имитационную модель с целью исследования в течение 1 ч функционирования ВККС зависимости емкости накопителя от интенсивности поступления сообщений, обеспечивающей вероятность передачи сообщений 1-й категории не менее 0,9, а сообщений 2-й категории — не менее 0,6.

Программа модели приведена ниже.

; Пример 7.7

; Задание исходных данных

VrMod	EQU	3600	; Время моделирования, 1 ед. мод. времени = 1 с
Ver	EQU	500	; Вероятность поступления команд в тысячных
долях			
T1	EQU	7	; Средний интервал поступления сообщений
ENak	EQU	100000	; Емкость накопителя, байт
Skor	EQU	1000	; Скорость передачи сообщений, бит/с
DlMin1	EQU	1400	; Минимальная длина сообщений 1-й категории
DlMax1	EQU	6000	; Максимальная длина сообщений 1-й категории
SrDl2	EQU	2000	; Средняя длина сообщений 2-й категории
SkDl2	EQU	300	; Среднеквадратическое отклонение
			длины сообщений 2-й категории

; Описание арифметических выражений вычисления

; длины сообщения 1-й категории

Dl1 VARIABLE DlMin1+((DlMax1-DlMin1)#(RN1/1000))

; длины сообщения 2-й категории

Dl2 VARIABLE Normal(144,SrDl2,SkDl2)

WrPer VARIABLE (8#P2)/Skor ; Время передачи сообщения

TEmk VARIABLE ENak-Q\$Nak ; Текущая емкость накопителя

; вероятностей передачи и потерь сообщений

VPeK VARIABLE N\$Ter1/N\$Met2 ; Вероятность передачи команд

VNfI VARIABLE N\$Ter2/N\$Met3 ; сообщений 2-й категории

VPoK VARIABLE N\$Ter3/N\$Met2 ; Вероятность потери команд

VPNfI VARIABLE N\$Ter4/N\$Met3 ; сообщений 2-й категории

; Сегмент имитации потока команд (донесений), т. е. сообщений 1-й категории

GENERATE (Exponential(11,0,T1))

TRANSFER Ver, Met3, Met2 ; Категория сообщения

Met2 ASSIGN 1,1 ; Код 1 — команда

ASSIGN 2,V\$Dl1 ; Длина сообщения 1-й категории

ASSIGN 3,V\$WrPer ; Расчет времени передачи

; сообщения 1-й категории

```

        PRIORITY    10          ; Команде – высокий приоритет
        TRANSFER    ,Met1       ; Отправить на ВККС
; Сегмент имитации потока сообщений 2-й категории
Met3      ASSIGN    1,2         ; Код 2 – сообщение 2-й категории
        ASSIGN      2,V$Dl2     ; Расчет длины сообщения 2-й категории
        ASSIGN      3,V$WrPer   ; Расчет времени передачи
;                                     сообщения 2-й категории
        PRIORITY    0          ; Сообщениям 2-й категории – низкий приоритет
; Сегмент имитации работы накопителя и 1-го канала
Met1      TEST LE    P2,V$TEmk, Met5 ; Проверка: есть ли место
;                                     в накопителе Nak? Если нет, то сообщение теряется
        QUEUE       Nak,P2      ; Помещение сообщения в накопитель
        GATE NI     Can1, Met7   ; Проверка: не захвачен ли 1-й канал?
Met6      PREEMPT   Can1, PR     ; Если нет, то захватить 1-й канал,
        TEST E      P1,1, Met8   ; Если сообщение 1-й категории,
        DEPART      Nak,P2      ; то освободить накопитель
Met8      ADVANCE    P3          ; Задержаться на время передачи сообщения
        RETURN      Can1        ; Освободить 1-й канал,
        TEST E      P1,2, Met9   ; Если сообщение 2-й категории,
        DEPART      Nak,P2      ; то освободить накопитель
Met9      UNLINK     Spis, Met6, 1 ; Вывести из накопителя сообщение
        TRANSFER    ,Met10      ; К счетчику переданных сообщений
; Сегмент имитации работы 2-го канала
Met7      GATE NI     Can2, Met12 ; Проверка: не захвачен ли 2-й канал?
Met14     PREEMPT   Can2, PR     ; Если да, то захватить 2-й канал,
        TEST E      P1,1, Met11  ; если сообщение 1-й категории,
        DEPART      Nak,P2      ; то освободить накопитель
Met11     ADVANCE    P3          ; Задержаться на время передачи сообщения
        RETURN      Can2        ; Освободить 2-й канал
        TEST E      P1,2, Met13  ; Если сообщение 2-й категории,
        DEPART      Nak,P2      ; то освободить накопитель
Met13     UNLINK     Spis, Met14, 1; Вывести из накопителя сообщение
        TRANSFER    ,Met10      ; К счетчику переданных сообщений
Met12     LINK       Spis, PR     ; Помещение сообщения в накопитель
; Сегмент счета переданных и потерянных сообщений
Met10     TEST E      P1,1, Ter2
Ter1      TERMINATE                                     ; Счет переданных сообщений 1-й категории
Ter2      TERMINATE                                     ; Счет переданных сообщений 2-й категории
Met5      TEST E      P1,1, Ter4
Ter3      TERMINATE                                     ; Счет потерянных сообщений 1-й категории
Ter4      TERMINATE                                     ; Счет потерянных сообщений 2-й категории
; Сегмент задания времени моделирования и расчета результатов
        GENERATE     VrMod
        SAVEVALUE    VPek, V$VPek ; Вероятность передачи команд

```

SAVEVALUE	VNfI, V\$VNfI	; сообщений 2-й категории
SAVEVALUE	VPoK, V\$VPoK	; Вероятность потери команд
SAVEVALUE	VPNfI, V\$VPNfI	; сообщений 2-й категории
TERMINATE	1	

В данном примере емкость накопителя определена в байтах, а не в единицах сообщений. Поэтому для организации накопителя использованы очередь с именем Nak и список пользователя с именем Spis. Сначала определяется: есть ли место в накопителе, т. е. в очереди Nak с ограниченной длиной Emk. И только если место есть, тогда транзакт помещается в список пользователя. Емкость списка пользователя неограниченна. Таким образом, вывод транзакта из списка пользователя еще не означает, что в накопителе появилось свободное место для размещения очередного транзакта. Все зависит от того, сколько «байтов» освободилось в очереди и достаточно ли свободной текущей емкости Темk накопителя для пытающегося войти в него транзакта.

Пусть в момент входа транзакта в модель накопитель оказался заполненным. Одновременно должно было завершиться обслуживание транзактов обоими ОКУ Can1 и Can2. То есть временной узел возможен между событиями «ввод транзакта в список пользователя», «завершение обслуживания ОКУ Can1» и «завершение обслуживания ОКУ Can2».

Пусть первым в СБС располагается транзакт, который должен войти в список пользователя. Так как в списке пользователя места нет, этот транзакт направляется к блоку TERMINATE с меткой Met3 и уничтожается. Затем будут последовательно обработаны транзакты, завершающие обслуживание на ОКУ Can1 и Can2. Из списка пользователя в порядке приоритета будут выведены в СТС транзакты и они займут ОКУ Can1 и Can2.

Таким образом, транзакт был потерян, хотя в этот момент модельного времени после обработки завершающих обслуживание транзактов в списке пользователя возможно оказались свободные места.

Если же первыми в СБС располагаются транзакты, освобождающие ОКУ Can1 и Can2, то они последовательно в порядке приоритетов выведут из списка пользователя два транзакта. Возможно, что освободится достаточно емкости для размещения поступившего транзакта и он не будет потерян.

Может быть и такая ситуация. В СБС первым и третьим расположены транзакты, освобождающие ОКУ Can1 и Can2. В этой ситуации возможны два исхода: после освобождения ОКУ первым транзактом в накопителе появится достаточно места и второй транзакт войдет в него; места окажется недостаточно и второй транзакт будет потерян.

Рассмотренные ситуации возможны тогда, когда все три транзакта, попавшие во временной узел, имеют одинаковые приоритеты (0 или 10) или приоритеты освобождающих ОКУ транзактов ниже, чем приоритет входящего транзакта. Вероятности возникновения этих ситуаций невелики, поэтому можно воспользоваться средствами, имеющимися в системе GPSS World. Пользователь тоже может предусмотреть разрешение подобных временных узлов. Нужно

после вывода из СБС обеспечить такой порядок расположения транзактов в СТС, чтобы первыми были транзакты, освобождающие ОКУ Can1 и Can2. Для этого, так как приоритеты транзактов 0 и 10, следует перед блоками RETURN обоих сегментов поставить блок PRIORITY 11. Перед освобождениями ОКУ транзакты проходят через эти блоки и переводятся в начало СТС.

Рассмотрим еще один пример со списком пользователя.

Пример 7.8. Одноканальная СМО с ожиданием. Необходимо организовать такую дисциплину, при которой приоритет отдается заявкам с наименьшим временем обслуживания.

; Пример 7.8

; Сегмент имитации поступления и обслуживания заявок

```
GENERATE (Exponential(43,0,200)) ; Источник заявок
ASSIGN Vrem,(Exponential(33,0,180)); Время обслуживания
GATE NU RRot,Otst ; Свободно ОКУ?
App SEIZE RRot ; Занять ОКУ
ADVANCE P$Vrem ; Обслуживание заявки
PRIORITY PR,BU ; Поставить последним в СТС
RELEASE RRot ; Освободить ОКУ
UNLINK POL,App,1 ; Вывести из списка пользователя
TERMINATE ; Обслуженные заявки
Otst LINK Pol,P$Vrem ; Список пользователя
```

; Сегмент задания времени моделирования

```
GENERATE 4320
TERMIATE 1
```

Пусть первым в СТС располагается вновь пришедший транзакт. Так как ОКУ с именем RRot занято, то блок GATE направит этот транзакт в блок LINK с меткой Otst, и он будет введен в список пользователя с именем Polk. Затем будет обработан транзакт, освобождающий ОКУ. Проходя через блок UNLINK, он выведет транзакт с начала списка пользователя и направит его в СТС, где тот продвинется в блок SEIZE, занимая устройство RRot.

Если же первым в СТС располагается транзакт, освобождающий ОКУ, он выведет первый из ожидающих транзактов из списка пользователя в СТС, где тот расположится после вновь пришедшего транзакта. Поэтому первым будет обработан вновь пришедший транзакт, который пройдет через блок GATE и займет ОКУ «без очереди». Очередной транзакт, который был выведен из списка пользователя, будет остановлен перед блоком SEIZE и после очередного освобождения ОКУ займет его, нарушая, в свою очередь, логику работы модели.

Проведенный анализ показывает, что для правильной обработки временного узла необходимо обеспечить такой порядок расположения транзактов в СТС, чтобы первым всегда располагался вновь пришедший транзакт. В рассматриваемом случае этого можно добиться, используя блок PRIORITY с операндом BU. Это и показано в приведенной выше модели.

Перед освобождением ОКУ обслуженный транзакт проходит через блок **PRIORITY**, который, оставляя неизменным приоритет транзакта **PR**, переводит его в конец **СТС**. При новом просмотре списка в случае наличия временного узла начинает обрабатываться вновь поступивший транзакт. Так как ОКУ еще занято, он направляется блоком **GATE** в список пользователя. При повторной обработке обслуженного транзакта тот освобождает ОКУ и выводит очередной транзакт из списка пользователя. Таким образом, правильная обработка временного узла обеспечивается и в этом случае.

Результаты моделирования при использовании средств **GPSS World** и рекомендуемого метода после 10 000 прогонов получились одинаковыми: коэффициент загрузки ОКУ составляет 0,9, а среднее время обслуживания одной заявки — 180,782.

7.5. Методы построения моделей с рациональными вариантами обслуживания

7.5.1. Блок **COUNT**

Блок **COUNT** определяет число объектов, удовлетворяющих заданному условию, и записывает в параметр активного транзакта.

Пусть, например, необходимо узнать число свободных ОКУ или число МКУ с коэффициентом использования меньше 700 (в долях от 1 000) и др.

Блок **COUNT** имеет следующий формат записи:

COUNT *X* *A, B, C, [D], [E]*

Блок **COUNT** работает в двух режимах: условном или логическом. Режим работы определяется оператором *X*:

○ *X* — логический или условный оператор.

Логический оператор задает логическое условие, например, подсчет числа свободных МКУ, занятых ОКУ и др. Для ОКУ используются следующие логические операторы: **NU**, **U**, **NI**, **I**, **FV**, **FNV**, для МКУ: **SE**, **SNE**, **SF**, **SNF**, **SV**, **SNV**, для логических ключей: **LR**, **LS**. Содержание указанных логических операторов изложено при рассмотрении блока **GATE**. При использовании логических операторов операнды *D* и *E* могут быть опущены.

В качестве условных операторов используются следующие: **L**, **NE**, **LE**, **E**, **G**, **GE**, **MAX**, **MIN**, **NE. MAX (MIN)** — наибольшее (наименьшее) значение из всех значений **СЧА** объектов, удовлетворяющих заданному условию. **NE** — значение проверяемого **СЧА** должно быть не равно заданному значению. Содержание остальных условных операторов изложено в п.1.5.2. Если используются условные операторы, то операнды *D* и *E* должны быть заданы.

Операнд *A* — номер параметра, вошедшего в блок **COUNT** транзакта, в котором будет организован счетчик числа объектов. Операнд *A* может быть именем,

положительным целым числом, выражением в скобках, СЧА или СЧА*параметр.

Операнды В и С определяют соответственно нижнюю и верхнюю границы диапазона изменения номеров или имен объектов, для которых проверяется заданное условие. Значение операнда С должно быть больше или равно значению операнда В. Операнды А и В могут быть такими же, как и операнд А.

Операндом D задается значение, которое сравнивается со значением операнда Е. Операнды D и Е используются только совместно с условными операторами L, LE, E, NE, G, GE. Если условный оператор не используется, то операнд D можно не задавать.

Операнд Е — какое-либо из СЧА проверяемых объектов. Необходимо записывать только мнемоническое обозначение атрибута, поскольку диапазон изменения номеров объектов задан операндами В и С.

Рассмотрим примеры использования блока COUNT.

```
COUNT L      3, 4, 8, X$Bb2, FT
COUNT SNF    2, Pogr1, Pogr3
COUNT E      KolQ_0, ObsQ1, ObsQ6, 0, 0
```

В первом примере блок COUNT работает в условном режиме. Подсчитывается число ОКУ в диапазоне 4 ... 8 (операнды В и С), среднее время использования (операнд Е) одним транзактом которых меньше времени, записанного в сохраняемой ячейке BB2 (операнд D), и заносится в параметр 3 (операнд А) вошедшего в блок COUNT транзакта.

Во втором примере блок COUNT работает в логическом режиме. Определяется и заносится в параметр 2 (операнд А) вошедшего в блок транзакта число незаполненных (оператор SNF) МКУ в диапазоне Pogr1 ... Pogr3 (операнды В и С). Операнды D и Е не используются. В данном примере МКУ указаны именами. По условиям блока COUNT необходимо этим именам предварительно присвоить командой EQU последовательные номера. В модели это будет так:

```
Pogr1 EQU      1
Pogr2 EQU      2
Pogr3 EQU      3
Pogr1 STORAGE  5
Pogr2 STORAGE  3
Pogr3 STORAGE  8
```

```
COUNT SNF    2, Pogr1, Pogr3
```

В третьем примере блок COUNT работает также в условном режиме. Определяется количество очередей (операнд Е) в диапазоне ObsQ1 ... ObsQ6 (операнды В и С), длины которых равны нулю (операнд D), и заносится в параметр активного транзакта с именем KolQ_0 (операнд А). Именам ObsQ1 ... ObsQ6 очередей также командами EQU должны быть присвоены последовательные номера.

7.5.2. Блок SELECT

Блок SELECT выбирает первый объект в заданном диапазоне, который удовлетворяет определенному условию. Номер этого объекта заносится в заданный параметр вошедшего в блок транзакта.

Блок SELECT имеет следующий формат записи:

```
SELECT  X      A, B, C, [D], [E], [F]
```

Логические и условные операторы X имеют тот же смысл, что и в блоке COUNT. Назначение операндов A, B, C, D и E также аналогично.

Операнд F задает номер альтернативного блока для входящего транзакта, если заданному условию не удовлетворяет ни один объект в указанном диапазоне. Если операнд F не используется, то транзакт будет направлен к следующему по номеру блоку. Например:

```
SELECT  MIN    2, 1, 6, , FR  
SELECT  SV     PPogr, Pogr1, Pogr3, , Met4  
SELECT  LE     4, Aa1, Aa7, X$Bb3, Q
```

В первом примере блок SELECT работает в условном режиме (условный оператор MIN). В диапазоне 1...6 (операнды B и C) определяется первое ОКУ с минимальным коэффициентом использования (операнд E). Номер выбранного ОКУ записывается в параметр 2 (операнд A) вошедшего в блок транзакта. Операнды D и F не используются. Транзакт из блока SELECT будет направлен к следующему блоку.

Во втором примере блок SELECT работает в логическом режиме. Из диапазона Pogr1...Pogr3 (операнды B и C) определяется первое МКУ, которое доступно (логический оператор SV). Номер найденного МКУ записывается в параметр транзакта с именем PPogr (операнд A). Операнды D и E не используются. Если доступный МКУ не будет найден, активный транзакт будет направлен к блоку с меткой Met4 (операнд F). В противном случае — к следующему блоку. Именам Pogr1...Pogr3 МКУ командами EQU должны быть присвоены последовательные номера.

В третьем примере блок SELECT также работает в условном режиме (условный оператор LE). Из очередей Aa1...Aa7 (операнды B и C) выбирается первая очередь (операнд E), длина которой меньше или равна значению ячейки Bb3 (операнд D). Номер очереди записывается в параметр 4 (операнд A) вошедшего в блок транзакта. Операнд F не используется, поэтому транзакт из блока SELECT направляется к следующему блоку. Именам Aa1...Aa7 очередей также командами EQU должны быть присвоены последовательные номера.

Пример 7.9. На склад, имеющий 10 пунктов выгрузки, прибывают транспорта с материальными средствами (МС). В транспорте могут быть автомобили различной грузоподъемности. Всего количество типов автомобилей, отличающихся грузоподъемностью, которое может быть в транспорте, распределено по нормальному закону с математическим ожиданием 8 автомобилей и стан-

дартным отклонением 1 автомобиль. Количество автомобилей одного типа в транспорте также распределено по нормальному закону с математическим ожиданием 15 автомобилей и стандартным отклонением 2 автомобиля. Интервалы времени прибытия транспортов распределены по экспоненциальному закону со средним значением 9 часов. Время разгрузки автомобиля зависит от его типа и подчиняется экспоненциальному закону. Среднее время выгрузки МС приведено в табл. 7.2. После выгрузки из тех же автомобилей вновь формируется транспорт.

Построить модель функционирования пунктов выгрузки в течение трех суток с целью определения среднего времени выгрузки транспортов и коэффициентов загрузки пунктов выгрузки при рациональной организации работ на складе.

Таблица 7.2

Среднее время выгрузки МС из автомобилей различных типов

Характеристики	Тип автомобиля									
	1	2	3	4	5	6	7	8	9	10
Среднее время выгрузки, мин	26	20	30	25	15	18	28	32	23	30

Для имитации транспорта в целом и одного автомобиля следует использовать транзакты. Пункты выгрузки в модели будут представлять ОКУ.

Приведем программу модели.

; Пример 7.9

; Задание исходных данных

```
IntTp      EQU      540    ; Средний интервал поступления транспортов
MOTip      EQU      8      ; Матожидание типов автомобилей в транспорте
SOTip      EQU      1      ; Стандартное отклонение типов автомобилей
MatA       EQU      15     ; Матожидание автомобилей одного типа
SotkA      EQU      2      ; Стандартное отклонение автомобилей одного типа
KolPun     EQU      10     ; Количество пунктов выгрузки
VrMod      EQU      1440   ; Время моделирования
```

; Определение функций и булевых переменных

```
SrVrA      FUNCTION  P3,D10 ; Среднее время выгрузки
1,26/2,20/3,30/4,25/5,15/6,18/7,28/8,32/9,23/10,30
TipAvt     FUNCTION  RN64,C2 ; Число типов автомобилей
0,1/1,9
```

; Сегмент имитации поступления и выгрузки транспортов

```
GENERATE    (Exponential(33,0,IntTp)); Источники транспортов
SAVEVALUE  1,0 ; Обнуление счетчика
ASSIGN     5,AC1 ; Время прибытия транспорта
ASSIGN     2,(INT(Normal(77, MOTip,SOTip)-1)); Число типов
```

```

SPLIT      *2          ; Транзактов - по числу типов автомобилей
ASSIGN     2+,1        ; Запомнить число типов автомобилей
ASSIGN     4,(INT(Normal(55,MatA,S0tkA)-1)); Автомобили
SPLIT      *4          ; Транзактов - по числу автомобилей
ASSIGN     4+,1        ; Запомнить число автомобилей одного типа
TEST L     X1,KolPun,Met1; Есть свободные пункты выгрузки?
SAVEVALUE  1+,1
Met2       SELECT MIN  3,1,KolPun,,FR    ; Выбор пункта с min загрузкой
TEST NE    P3,0,Met1    ; Есть пункты с min загрузкой?
QUEUE      P3           ; Встать в очередь
SEIZE      P3           ; Занять пункт выгрузки
DEPART     P3           ; Покинуть очередь
ADVANCE    (Exponential(73,0,FN$SrVrA)); Выгрузка
RELEASE    P3           ; Освободить АРМ
UNLINK     PunSb,Met2,1 ; Автомобиль на выгрузку
ASSEMBLE   *4           ; Собрать автомобили одного типа
ASSEMBLE   *2           ; Собрать автомобили всех типов
ASSIGN     6,AC1        ; Время убывтия транспорта
SAVEVALUE  VrVigS+,(P6-P5) ; Суммарное время выгрузки
VigTr      TERMINATE    ; Выгруженные транспорта
Met1       LINK         PunSb,FIFO      ; Ожидание выгрузки
; Сегмент задания времени моделирования
GENERATE    VrMod
TEST NE     N$VigTr,0,Met3
SAVEVALUE   VrVig,(X$VrVigS/N$VigTr) ; Среднее время выгрузки
Met3        TERMINATE   1

```

Замысел построения модели заключается в следующем. Блок GENERATE имитирует поступление транзактов. Один транзакт — один транспорт. Затем случайным образом определяется количество типов автомобилей в транспорте и заносится в параметр 2 транзакта. Исходный (порождающий) транзакт-транспорт копируется блоком SPLIT и результат копирования — число автомобилей различных типов в прибывшем транспорте копируется вторым блоком SPLIT. Результат копирования — число транзактов, равное числу автомобилей в транспорте. Число автомобилей одного типа заносится в параметр 4 всех транзактов, имитирующих автомобили данного типа.

Далее первые десять транзактов-автомобилей (по числу пунктов выгрузки) поступают на блок SELECT, остальные — в список пользователя с именем PunSb. По мере освобождения пунктов выгрузки из списка пользователя выводятся транзакты и направляются на блок SELECT с меткой Met2. Результат работы блока SELECT — номер пункта выгрузки с минимальным коэффициентом загрузки заносится в параметр 3 вошедшего транзакта. Если результатом поиска блока SELECT является P3=0, т. е. номер нужного блока не найден, то транзакт блоком TEST снова отправляется в список пользователя. При успешном поиске транзакт занимает соответствующий пункт выгрузки.

Разгруженные автомобили первым блоком ASSEMBLE сначала собираются по каждому типу отдельно, а вторым блоком ASSEMBLE — по всем типам.

При входе транзакта-транспорта в модель и выходе из нее в параметры 5 и 6 записывается соответствующее модельное время. Оно используется для расчета времени разгрузки транспорта. Суммарное время разгрузки всех транспортов накапливается в ячейке VrVig5.

Приведем результаты моделирования, полученные после 1 000 прогонов.

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND
1	26338	0.477	26.066	1	295377	0
2	34123	0.477	20.116	1	0	0
3	22869	0.477	30.017	1	0	0
4	27576	0.477	24.891	1	0	0
5	46137	0.477	14.883	1	0	0
6	38151	0.477	18.009	1	0	0
7	24685	0.479	27.931	1	294357	0
8	21632	0.478	31.801	1	294997	0
9	29835	0.477	23.010	1	295324	0
10	22783	0.476	30.087	1	295355	0

QUEUE	MAX	CONT.	ENTRY	ENTRY (0)	AVE. CONT.	AVE. TIME	AVE. (-0)
1	394	9	26347	2948	26.108	1426.958	1606.738
2	369	0	34123	4840	22.959	968.860	1128.997
3	413	0	22869	2350	24.795	1561.254	1740.062
4	360	0	27576	3438	22.265	1162.660	1328.258
5	496	0	46137	6493	21.508	671.290	781.236
6	409	0	38151	4185	27.478	1037.151	1164.940
7	432	154	24839	2349	27.512	1594.987	1761.578
8	583	50	21682	2233	27.053	1796.737	2003.026
9	349	24	29859	3353	22.728	1096.107	1234.764
10	373	19	22802	2472	24.485	1546.256	1734.271

Коэффициенты загрузки всех десяти пунктов выгрузки оказались практически одинаковыми (0,477...0,479), так как абсолютное равенство получить невозможно.

Пример 7.10. Соединение ведет боевые действия в течение восьми суток. В соединении имеются СС трех типов, из них первого типа — 36, второго — 54, третьего — 72 шт. СС может в любой момент времени выйти из строя. При этом оно может быть либо безвозвратно потеряно (БВП), либо требовать одного из трех видов ремонта: капитального (КР), среднего (СР) или текущего (ТР). Распределение времени выхода из строя любого СС подчиняется нормальному закону с математическим ожиданием и стандартным отклонением, приведенными в табл. 7.3 для каждого типа СС.

Таблица 7.3

Данные о времени выхода средств связи из строя

Характеристика закона распределения	Тип средств связи		
	1	2	3
Математическое ожидание времени выхода средств связи из строя, ч	480	410	360
Стандартное отклонение времени выхода средств связи из строя, ч	95	80	70

Если СС требует КР, то оно эвакуируется из части в ремонтный орган, который может выполнить этот вид ремонта. Время эвакуации распределено по нормальному закону с математическим ожиданием 240 ч и стандартным отклонением 10 ч. Время выполнения КР распределено по экспоненциальному закону со средним временем для каждого типа СС соответственно 480, 370, 320 ч.

Если СС требует СР, оно также эвакуируется из части в ремонтный орган, способный выполнить этот вид ремонта. При этом время эвакуации распределено по нормальному закону с математическим ожиданием 120 ч и стандартным отклонением 5 ч. Время СР распределено по экспоненциальному закону со средними значениями соответственно 220, 200, 180 ч.

Если СС требует ТР, оно остается в части и ремонтируется своими мастерами. При этом среди мастеров существуют три группы специалистов, каждая из которых ремонтирует одну из трех соответствующих частей СС, вышедшего из строя. Время ремонта распределено по экспоненциальному закону со средними значениями, приведенными в табл. 7.4.

Таблица 7.4

Данные о среднем времени ремонта средств связи

Группа мастеров	Среднее время ремонта типа средства связи, ч		
	1	2	3
1	24	18	32
2	36	28	16
3	48	15	28

После ремонта СС возвращаются для выполнения боевых задач. Распределение времени возвращения после КР или СР подчиняется нормальному закону с указанными в табл. 7.5 параметрами, соответственно. Вероятность возвращения в часть отремонтированных СС после КР, СР и ТР соответственно 0,1, 0,9, 1,0.

Таблица 7.5

Данные о времени возвращения средств связи в строй

Характеристика закона распределения	Вид ремонта		
	1	2	3
Математическое ожидание времени возвращения в строй, ч	120	60	6
Стандартное отклонение времени возвращения в строй, ч	5	2	1

Вероятность выхода в тот или иной вид ремонта распределена по равномерному закону и составляет: ТР — 0,5; СР — 0,25; КР — 0,125; БВП — 0,125.

Построить модель системы и исследовать ее с целью определения среднего числа исправных образцов средств связи в части.

В системе, модель которой необходимо построить, существуют следующие ограничения:

1. Число образцов средств связи 1-го, 2-го и 3-го типов.
2. Число групп мастеров в ремонтном подразделении.
3. Число средств эвакуации в каждый вид ремонта.
4. Число средств возвращения в строй после каждого вида ремонта.

Первое ограничение в модели удобно учесть транзактами, а второе, третье и четвертое — многоканальными устройствами.

За единицу модельного времени возьмем 1 ч. Тогда время моделирования будет равно $8 \times 24 = 192$.

Примем условие, что вначале все СС исправны. Поэтому три блока GENERATE должны иметь только операнды D, равные соответственно 36, 54 и 72 (операнды A, B, C отсутствуют). То есть генераторы генерируют одновременно 36, 54 и 72 транзакта и на этом их активность прекращается на весь период работы модели.

Для определения времени выхода в ремонт СС, времени его эвакуации в ремонтный орган и времени возвращения в строй, распределенных по нормальному закону, используем функцию SNorm.

Для определения времени ремонта в любом ремонтном органе, распределенного по экспоненциальному закону, воспользуемся нормированной функцией XpDis.

Для определения случайным образом вида ремонта вышедшего из строя СС используем дискретную функцию WidRem, в качестве аргумента которой возьмем значения RN121. Значения функции WidRem приведены в табл. 7.6. Код вида ремонта будем записывать в параметр 4 (P4) соответствующего транзакта.

Таблица 7.6

Данные для задания дискретной функции WidRem

Значение аргумента RN121	0,5	0,75	0,875	1,0
Значение функции WidRem	3	2	1	4
Соответствует виду потерь	ТР	СР	КР	БВП

Чтобы определить, какая группа специалистов-ремонтников должна ремонтировать неисправный образец СС, требующий ТР, воспользуемся дискретной функцией GrRem, в качестве аргумента которой опять возьмем значения RN111. Значения функции Grm представлены в табл. 7.7. Номер группы мастеров записывается в P5 соответствующего транзакта.

Выбор среднего времени ТР соответствующей группой специалистов производится с помощью матрицы VrRem, значения которой приведены в табл. 7.4. А выбор среднего времени КР или СР — с помощью матрицы VrKS. Аналогично из матриц MatEv и MatStr (см. табл. 7.5) выбираются математическое ожидание и стандартное отклонение для расчета времени соответственно эвакуации и возвращения в строй.

Таблица 7.7

Данные для задания дискретной функции GrRem

Значение аргумента RN111	0,35	0,75	1,0
Значение функции GrRem	1	2	3
Соответствует группе мастеров	1	2	3

Предположим, что восполнение потерь осуществляется не сразу после выхода из строя одного СС какого-либо типа, а нескольких. Скольких? Право решать это предоставим разработчику. Введем сохраняемые величины Pot1, Pot2, Pot3 — количество потерянных образцов СС соответственно 1-го, 2-го и 3-го типов и переменную VrDost для расчета времени доставки (задержки) СС для восполнения потерь, распределенного по нормальному закону с математическим ожиданием M0G1 и стандартным отклонением ST01 (M0G1 и ST01 — сохраняемые величины).

Восполняются СС, вышедшие в КР и безвозвратно потерянные. Программа модели приведена ниже.

```
; Пример 7.10
;      Определение функции экспоненциального распределения
Xpdis      FUNCTION      RN121,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
```

; *Определение функции нормального распределения*

SNorm FUNCTION RN123, C25

0, -5/.00003, -4/.00135, -3/.00621, -2.5/.02275, -2/.06681, -1.5/.11507, -1.2/.15866, -1.21186, -.8/.27245, -.6/.34458, -.4/.42074, -.2/.5, .0/.57926, .2/.65542, .4/.72575, .6.78814, .8/.84134, 1/.88493, 1.2/.93319, 1.5/.97725, 2/.99379, 2.5/.99865, 3/.99997, 4/1, 5

; *Задание исходных данных*

; *Определение функций*

WidRem FUNCTION RN121, D4 ; Вид ремонта
.5, 3/.75, 2/.875, 1/1, 4

GrRem FUNCTION RN111, D3 ; Номер группы ремонта
.35, 1/.75, 2/1, 3

; *Определение матриц*

VrVix MATRIX , 2, 3

VrRem MATRIX , 3, 3

VrKS MATRIX , 2, 3

MatEv MATRIX , 2, 3

MatStr MATRIX , 2, 3

; *Данные о времени выхода СС из строя*

INITIAL MX\$VrVix(1, 1), 480

INITIAL MX\$VrVix(1, 2), 410

INITIAL MX\$VrVix(1, 3), 360

INITIAL MX\$VrVix(2, 1), 95

INITIAL MX\$VrVix(2, 2), 80

INITIAL MX\$VrVix(2, 3), 70

; *Данные о времени ТР группами мастеров*

INITIAL MX\$VrRem(1, 1), 24

INITIAL MX\$VrRem(1, 2), 18

INITIAL MX\$VrRem(1, 3), 32

INITIAL MX\$VrRem(2, 1), 36

INITIAL MX\$VrRem(2, 2), 28

INITIAL MX\$VrRem(2, 3), 16

INITIAL MX\$VrRem(3, 1), 48

INITIAL MX\$VrRem(3, 2), 15

INITIAL MX\$VrRem(3, 3), 22

; *Данные о времени КР и СР*

INITIAL MX\$VrKS(1, 1), 480

INITIAL MX\$VrKS(1, 2), 370

INITIAL MX\$VrKS(1, 3), 320

INITIAL MX\$VrKS(2, 1), 220

INITIAL MX\$VrKS(2, 2), 200

INITIAL MX\$VrKS(2, 3), 180

; *Данные о времени эвакуации в ремонт*

INITIAL MX\$MatEv(1, 1), 240

INITIAL MX\$MatEv(1, 2), 120

```

INITIAL      MX$MatEv(1,3),6
INITIAL      MX$MatEv(2,1),10
INITIAL      MX$MatEv(2,2),5
INITIAL      MX$MatEv(2,3),1
;           Данные о времени возвращения в строй
INITIAL      MX$MatStr(1,1),120
INITIAL      MX$MatStr(1,2),60
INITIAL      MX$MatStr(1,3),6
INITIAL      MX$MatStr(2,1),5
INITIAL      MX$MatStr(2,2),2
INITIAL      MX$MatStr(2,3),1
;           Данные о нормах потерь
INITIAL      X$Pot1,6
INITIAL      X$Pot2,8
INITIAL      X$Pot3,10
INITIAL      X$Mog1,45
INITIAL      X$Sto1,3
; Определение арифметических выражений
VrDost       VARIABLE      X$Mog1+X$Sto1#FN$SNorm
VixStr       VARIABLE      MX$VrVix(1,P6)+FN$SNorm#MX$VrVix2,P6)
VrEvak       VARIABLE      MX$MatEv(1,P4)+FN$SNorm#MX$MatEv(2,P4)
VrVosvr      VARIABLE      MX$MatStr(1,P6)+FN$SNorm#MX$MatStr(2,P6)
Xp1          VARIABLE      MX$VrKS(P4,P6)#FN$XpDis
Xp2          VARIABLE      MX$VrRem(P5,P6)#FN$XpDis
; Определение многоканальных устройств
;           Количество СС по типам
Bat1         STORAGE       36
Bat2         STORAGE       54
Bat3         STORAGE       72
;           Количество средств эвакуации
EvKR         STORAGE       3
EvSR         STORAGE       3
EvTR         STORAGE       3
;           Количество каналов ремонта
KRem         STORAGE       4
SRem         STORAGE       2
TRem         STORAGE       3
;           Количество средств возвращения в строй
VosKR        STORAGE       5
VosSR        STORAGE       5
VosTR        STORAGE       5
; Сегмент имитации выхода из строя СС типа 1
Met6         GENERATE      ,,36           ; Генератор транзактов - СС типа 1
Met6         ASSIGN        6,1           ; Код 1 - СС типа 1

```



```

Met1      ENTER      Bat1      ; Занять канал МКУ
          ADVANCE     V$VixStr  ; Время выхода из строя
          LEAVE       Bat1      ; Освободить канал МКУ
; Сегмент имитации эвакуации, КР и возвращения в строй
Met4      ASSIGN     4, FN$widRem ; Вид ремонта
          TEST E      P4, 4, KRem ; БВП?
          TRANSFER    ,Bbb4      ; Тогда к счетчику БВП и КР
KRem      TEST E      P4, 1, SRem ; КР?
          SPLIT       1, Bbb4     ; Тогда к счетчику БВП и КР
          ENTER       EvKR        ; Занять средство эвакуации
          ADVANCE     V$VrEvak    ; Время эвакуации
          LEAVE       EvKR        ; Освободить средство эвакуации
          ENTER       KRem        ; Занять канал КР
          ADVANCE     V$Xp1       ; Время КР
          LEAVE       KRem        ; Освободить канал КР
          ENTER       VosKR       ; Занять МКУ для возвращения в строй
          ADVANCE     V$VrVosvr   ; Время возвращения в строй из КР
          LEAVE       VosKR       ; Освободить МКУ
          TRANSFER    ,Met5       ; Отправить для распределения
; Сегмент имитации эвакуации, СР и возвращения в строй
SRem      TEST E      P4, 2, TRem ; СР?
          ENTER       EvSR        ; Да, занять средство эвакуации
          ADVANCE     V$VrEvak    ; Время эвакуации
          LEAVE       EvSR        ; Освободить средство эвакуации
          ENTER       SRem        ; Занять канал СР
          ADVANCE     V$Xp1       ; Время СР
          LEAVE       SRem        ; Освободить канал СР
          ENTER       VosSR       ; Занять МКУ для возвращения в строй
          ADVANCE     V$VrVosvr   ; Время возвращения в строй из СР
          LEAVE       VosSR       ; Освободить МКУ
          TRANSFER    ,Met5       ; Отправить для распределения
; Сегмент имитации эвакуации, ТР и возвращения в строй
TRem      ENTER       EvTR        ; Занять средство эвакуации
          ADVANCE     V$VrEvak    ; Время эвакуации
          LEAVE       EvTR        ; Освободить средство эвакуации
          ASSIGN      5, FN$GrRem ; Определить ремонтную группу
          ENTER       TRem        ; Занять канал ТР
          ADVANCE     V$Xp2       ; Время ТР
          LEAVE       TRem        ; Освободить канал ТР
          ENTER       VosTR       ; Занять МКУ для возвращения в строй
          ADVANCE     V$VrVosvr   ; Время возвращения в строй из ТР
          LEAVE       VosTR       ; Освободить МКУ
          TRANSFER    ,Met5       ; Отправить для распределения

```

; Сегмент имитации выхода из строя СС типа 2

	GENERATE	,,54	; Генератор транзактов – СС типа 2
Met7	ASSIGN	6,2	; Код 2 – СС типа 2
Met2	ENTER	Bat2	; Занять канал МКУ
	ADVANCE	V\$VixStr	; Время выхода из строя
	LEAVE	Bat2	; Освободить канал МКУ
	TRANSFER	,Met4	; Отправить в ремонт

; Сегмент имитации выхода из строя СС типа 3

	GENERATE	,,72	; Генератор транзактов – СС типа 3
Met8	ASSIGN	6,3	; Код 3 – СС типа 3
Met3	ENTER	Bat3	; Занять канал МКУ
	ADVANCE	V\$VixStr	; Время выхода из строя
	LEAVE	Bat3	; Освободить канал МКУ
	TRANSFER	,Met4	; Отправить в ремонт
Ccc1	TERMINATE		; Потери СС

; Сегмент распределения СС по типам после ремонта

Met5	TEST NE	P6,1,Met1	; СС типа 1
	TEST NE	P6,2,Met2	; СС типа 2
	TRANSFER	,Met3	; СС типа 3

; Сегмент имитации восполнения БВП и КР

; *Учет и восполнение СС типа 3*

Bbb4	SAVEVALUE	Dost,V\$VrDost	; Время доставки при восполнении потерь
	TEST NE	P6,1,Bbb1	; СС типа 1
	TEST NE	P6,2,Bbb2	; СС типа 2
	SAVEVALUE	BWP3+,1	; Учет потерь СС типа 3
	SAVEVALUE	BWP31+,1	; Суммарные потери СС типа 3
	TEST GE	X\$BWP3,X\$Pot3,Bbb3	; Потери больше нормы?
	ADVANCE	X\$Dost	; Доставка при восполнении потерь
	SPLIT	X\$BWP3,Bbb8	; Транзактов – по числу СС типа 3
	SAVEVALUE	BWP3,0	; Сброс счетчика потерь СС типа 3

Bbb3 TERMINATE

; *Учет и восполнение СС типа 1*

Bbb1	SAVEVALUE	BWP1+,1	; Учет потерь СС типа 1
	SAVEVALUE	BWP11+,1	; Суммарные потери СС типа 1
	TEST GE	X\$BWP1,X\$Pot1,Bbb9	; Потери больше нормы?
	ADVANCE	X\$Dost	; Доставка при восполнении потерь
	SPLIT	X\$BWP1,Bbb6	; Транзактов – по числу СС типа 1
	SAVEVALUE	BWP1,0	; Сброс счетчика потерь СС типа 1

Bbb9 TERMINATE

; *Учет и восполнение СС типа 2*

Bbb2	SAVEVALUE	BWP2+,1	; Учет потерь СС типа 2
	SAVEVALUE	BWP21+,1	; Суммарные потери СС типа 2
	TEST GE	\$BWP2,X\$Pot2,Bbb5	; Потери больше нормы?
	ADVANCE	X\$Dost	; Доставка при восполнении потерь

	SPLIT	X\$BWP2, Bbb7	; Транзактов – по числу СС типа 2
	SAVEVALUE	BWP2, 0	; Сброс счетчика потерь СС типа 2
Bbb5	TERMINATE		

; Сегмент имитации потерь при доставке СС для восполнения

Bbb8	TRANSFER	.05, Met8, Ccc1	; Потери СС типа 3
Bbb6	TRANSFER	.1, Met6, Ccc1	; Потери СС типа 1
Bbb7	TRANSFER	.08, Met7, Ccc1	; Потери СС типа 2

; Сегмент задания времени моделирования

	GENERATE	192	; 1 ед. модельного времени = 1 час
	TERMINATE	1	

В модели функционирование СС имитируется МКУ Bat1, Bat2 и Bat3. В начале работы модели генераторы сразу вырабатывают количество транзактов, равное соответствующему количеству СС и перестают быть активными. Эти транзакты заполняют МКУ. В параметр 2 транзактов заносятся коды типов СС.

Время обслуживания МКУ транзактов соответствует возможному времени выхода СС из строя. При прохождении транзакта путем розыгрыша определяется вид требующегося ремонта. Код вида ремонта заносится в параметр 3 транзакта.

Если СС безвозвратно потеряно, транзакт направляется в сегмент имитации восполнения потерь и используется для организации восполнения потерь.

В случае требования КР, транзакт расщепляется на два транзакта. Один также направляется в сегмент имитации восполнения потерь, а второй проходит последовательность блоков, начиная с блока ENTER с меткой KRm, которыми имитируется эвакуация, КР и возвращение в строй.

Обработка транзактов — СС, требующих СР, производится аналогичной последовательностью блоков сегмента имитации эвакуации, СР и возвращения в строй.

При выполнении КР и СР не делается различия между мастерами.

Отличие обработки транзактов — СС, требующих ТР, состоит в том, что после имитации эвакуации путем розыгрыша определяется группа мастеров, которая будет выполнять ремонт. Код группы мастеров заносится в параметр 4 транзакта. Далее имитируется ТР, а затем — возвращение в строй.

Было принято условие, что восполняются только СС, требующие КР или безвозвратно потерянные. Хотя, конечно, разработчик может взять и другие условия. Транзакты распределяются по видам СС и блоками SAVEVALUE подсчитываются. Если потери больше установленных норм, блоками SPLIT производится размножение по числу потерь. Группой блоков TRANSFER в статистическом режиме имитируются потери при доставке СС для восполнения.

Время доставки СС для восполнения потерь взято одинаковым, хотя на практике может и не быть таким. Для учета различия модель легко совершенствуется. После имитации доставки определяется тип СС и в соответствии с ним транзакты направляются в соответствующие МКУ (Bat1, Bat2 или Bat3).

Ниже приведены результаты моделирования.

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.
BAT1	36	21	0	36	54	1	23.369	0.649
BAT2	54	33	0	54	97	1	36.949	0.684
BAT3	72	45	0	72	141	1	46.578	0.647
EVKR	3	0	0	3	10	1	2.153	0.718
EVSR	3	0	0	3	19	1	2.225	0.742
EVTR	3	3	0	3	121	1	0.750	0.250
KREM	4	1	0	4	7	1	1.536	0.384
SREM	2	0	0	2	6	1	1.271	0.636
TREM	3	0	0	3	96	1	2.176	0.725
VOSKR	5	5	0	2	4	1	0.255	0.051
VOSSR	5	5	0	1	4	1	0.027	0.005
VOSTR	5	0	0	5	77	1	3.082	0.616

По результатам моделирования можно судить о рациональном или нет варианте работы элементов и системы восстановления в целом. Например, обеспеченность исправными СС первого, второго и третьего типов в среднем составляет 64,9, 68,4 и 64,7 % соответственно. Этот вывод интерпретирован на основании коэффициентов использования МКУ: Bat1 — 0,649, Bat2 — 0,684 и Bat3 — 0,647.

7.6. Организация прогонов при изменении версий модели

В п. 3.3 рассмотрены команды CLEAR и RESET. Приведем примеры применения этих команд.

Пример 7.11. 50 средств связи (СС) находятся на дежурстве в течение 25 недель. Каждое СС может в любой момент времени выйти из строя. В этом случае его заменяют резервным, причем либо сразу, либо по мере его появления. Тем временем вышедшее из строя СС ремонтируют, после чего содержат в качестве резервного. В среднем на ремонт неисправного СС уходит 7 ч, распределение экспоненциальное. Когда СС находится на дежурстве, время наработки до отказа распределено по экспоненциальному закону и составляет в среднем 157 ч. Между мастерами в ремонтном подразделении не делают никаких различий. Иными словами, распределение времени ремонта не зависит от того, кто какое СС ремонтирует.

Построить модель такой системы и исследовать на ней ожидаемый доход при различном числе резервных СС и мастеров-ремонтников. Прибыль от СС, находящегося на дежурстве, составляет 5 денежных единиц в час. Почасовой убыток — 6 денежных единиц на средство связи. Оплата мастеров — 3,75 денежных единиц в час. Затраты на резервное СС составляют 10 денежных единиц в сутки. Время моделирования — 25 недель ($25 \times 42 = 1\,050$ часов).

Построение модели. Какие ограничения в системе?

1. Число мастеров в ремонтном подразделении.
2. Максимальное число СС, одновременно находящихся на дежурстве.
3. Общее число СС в системе.

Для моделирования двух первых ограничений удобно использовать МКУ, а третье ограничение моделировать транзактами.

Программа модели представлена ниже.

; Пример 7.11

; Задание исходных данных

KolSS	EQU	50	; Количество СС, находящихся на дежурстве
ResSS	EQU	3	; Количество резервных СС
VrMod	EQU	1050	; Время моделирования, 1 ед. мод. вр. = 1 час
NarOtk	EQU	157	; Среднее время наработки до отказа, час
TimeRem	EQU	7	; Среднее время ремонта, час
Stroka	EQU	1	; Номер строки матрицы
Stolbez	EQU	1	; Номер столбца матрицы
StoMast	EQU	3.75	; Оплата одного мастера в час
StoResSS	EQU	10	; Стоимость одного резервного СС
UbitSS	EQU	6	; Убыток при отсутствии на дежурстве одного СС
DoxSS	EQU	5	; Доход от одного СС, находящегося на дежурстве

; Определение матриц и МКУ

Pribil	MATRIX	,3,3	; Матрица ожидаемой прибыли
Koef	MATRIX	,3,3	; Матрица коэффициентов использования СС
Zentr	STORAGE	50	; Емкость МКУ по количеству СС
Rem	STORAGE	3	; Емкость МКУ по числу мастеров-ремонтников

; Сегмент имитации системы дежурства СС

	GENERATE	,,,1	
	SPLIT	(KolSS+ResSS-1)	; Число СС + число резервных СС
Met1	ENTER	Zentr	; Встать на дежурство
	ADVANCE	(Exponential(211,0,NarOtk));	Имитация отказа
	LEAVE	Zentr	; Снятие с дежурства из-за выхода из строя
	ASSIGN	2,(Exponential(311,0,TimeRem));	Расчет времени ремонта
	ENTER	Rem	; Занять одного мастера
	ADVANCE	P2	; Имитация ремонта
	LEAVE	Rem	; Конец ремонта
	TRANSFER	,Met1	; Направить СС на дежурство или в резерв

; Сегмент задания времени моделирования и расчета результатов

	GENERATE	VrMod	
	SAVEVALUE	Doxod,(ST\$Zentr#KolSS#DoxSS)	; Доход от дежурства СС
	SAVEVALUE	StoRem,(ST\$Rem#SM\$Rem#StoMast);	Затраты на ремонт
	SAVEVALUE	Ubitok,((NarOtk-ST\$Zentr)#UbitSS#KolSS);	Убыток
	SAVEVALUE	ZatrResSS,((StoResSS/24)#ResSS#VrMod);	Затраты на резерв

; Расчет ожидаемой прибыли

```

SAVEVALUE SumPrib, (X$Doxod-(X$StoRem+X$Ubitok+X$ZatrResSS))
MSAVEVALUE Pribil, Stroka, Stolbez, X$SumPrib ; Прибыль
MSAVEVALUE Koef, Stroka, Stolbez, (SR$Zentr/1000)
TERMINATE 1

ResSS START 10, NP ; Вариант 1: Резервных CC=3, мастеров=3
EQU 4
Stolbez EQU 2
CLEAR OFF

ResSS START 10, NP ; Вариант 2: Резервных CC=4, мастеров=3
EQU 5
Stolbez EQU 3
CLEAR OFF

ResSS START 10, NP ; Вариант 3: Резервных CC=5, мастеров=3
EQU 3
Rem STORAGE 4
Stroka EQU 2
Stolbez EQU 1
CLEAR OFF

ResSS START 10, NP ; Вариант 4: Резервных CC=3, мастеров=4
EQU 4
Stolbez EQU 2
CLEAR OFF

ResSS START 10, NP ; Вариант 5: Резервных CC=4, мастеров=4
EQU 5
Stolbez EQU 3
CLEAR OFF

ResSS START 10, NP ; Вариант 6: Резервных CC=5, мастеров=4
EQU 3
Rem STORAGE 5
Stroka EQU 3
Stolbez EQU 1
CLEAR OFF

ResSS START 10, NP ; Вариант 7: Резервных CC=3, мастеров=5
EQU 4
Stolbez EQU 2
CLEAR OFF

ResSS START 10, NP ; Вариант 8: Резервных CC=4, мастеров=5
EQU 5
Stolbez EQU 3
CLEAR OFF

START 10 ; Вариант 9: Резервных CC=5, мастеров=5

```

Программа позволяет вначале выполнить 10 прогонов при 53 CC и трех мастерах, затем последовательно при 54 и 55 CC. После этого количество мас-

теров-ремонтников увеличивается до четырех и опять выполняются 100 прогонов последовательно при 53, 54 и 55 СС. То же самое происходит и при пяти мастерах-ремонтниках.

Для исследования поведения системы при четырех и пяти мастерах-ремонтниках в программу добавляются переопределяемые по мере необходимости переменная пользователя ResSS и Rem STORAGE. Переменные пользователя Stroka и Stolbez добавляются для указания номеров строки и столбца соответственно матриц Pribil и Koef. В матрицу Pribil заносится ожидаемая прибыль, а в матрицу Koef — достигаемые при данной прибыли коэффициенты использования СС. Так как в матрицах должны сохраняться результаты, полученные при различных модификациях модели, то в команде CLEAR в качестве операнда A используется OFF. В команде START операнд B равен NP для всех вариантов модели, кроме варианта 9, поэтому стандартный отчет выдается только для этого варианта. Для включения матриц в стандартный отчет выберите File ▶ Settings (Файл ▶ Настройки) и на странице Reports (Отчеты) установите флажок Matrices (Матрицы). После этого нажмите Применить и Ok.

Приведем результаты моделирования.

MATRIX	RETRY	INDICES	VALUE
PRIBIL	0		
		1 1	37549.112
		1 2	34906.154
		1 3	33604.960
		2 1	34434.836
		2 2	36732.930
		2 3	37070.033
		3 1	36384.154
		3 2	35290.517
		3 3	36209.558
KOEF	0		
		1 1	.981
		1 2	.983
		1 3	.983
		2 1	.990
		2 2	.996
		2 3	.996
		3 1	.992
		3 2	.996
		3 3	.998

Количество прогонов при одной конфигурации модели, задаваемое оператором START, определяется разработчиком и может отличаться от указанного в примере.

Пример 7.12. В предыдущем примере результаты моделирования характеризовали работу системы в целом за 25 недель. В данной модели результаты первых 100 прогонов (по желанию пользователя число прогонов может быть другим) отбрасываются и далее выводятся за каждые пять недель моделирования. Число пар операторов RESET—START равно числу моделируемых недель, деленному на пять (в данном случае пять пар). Время моделирования равно продолжительности пяти недель ($42 \cdot 5 = 210$ часов). Программа также позволяет за один этап моделирования осуществить девять модификаций модели, идентичных модификациям предыдущей модели.

Программа модели представлена ниже.

; Пример 7.12

; Задание исходных данных

KolSS	EQU	50	; Количество СС, находящихся на дежурстве
ResSS	EQU	3	; Количество резервных СС
VrMod	EQU	210	; Время моделирования, 1 ед. мод. вр. = 1 час
NarOtk	EQU	157	; Среднее время наработки до отказа, час
TimeRem	EQU	7	; Среднее время ремонта, час
Stroka	EQU	1	; Номер строки матрицы
Stolbez	EQU	1	; Номер столбца матрицы
StoMast	EQU	3.75	; Оплата одного мастера в час
StoResSS	EQU	10	; Стоимость одного резервного СС
UbitSS	EQU	6	; Убыток при отсутствии на дежурстве одного СС
DoxSS	EQU	5	; Доход от одного СС, находящегося на дежурстве

; *Определение матриц и МКУ*

Pribil	MATRIX	,3,3	; Матрица ожидаемой прибыли
Koef	MATRIX	,3,3	; Матрица коэффициентов использования СС
Zentr	STORAGE	50	; Емкость МКУ по количеству СС
Rem	STORAGE	3	; Емкость МКУ по числу мастеров-ремонтников

; Сегмент имитации системы дежурства СС

	GENERATE	,,1	
	SPLIT	(KolSS+ResSS-1)	; Число СС + число резервных СС
Met1	ENTER	Zentr	; Встать на дежурство
	ADVANCE	(Exponential(211,0,NarOtk))	; Имитация отказа
	LEAVE	Zentr	; Снятие с дежурства из-за выхода из строя
	ASSIGN	2,(Exponential(311,0,TimeRem))	; Расчет времени ремонта
	ENTER	Rem	; Занять одного мастера
	ADVANCE	P2	; Имитация ремонта
	LEAVE	Rem	; Конец ремонта
	TRANSFER	,Met1	; Направить СС на дежурство или в резерв

; Сегмент задания времени моделирования и расчета результатов

	GENERATE	VrMod	
	SAVEVALUE	Doxod,(ST\$Zentr#KolSS#DoxSS)	; Доход от дежурства СС
	SAVEVALUE	StoRem,(ST\$Rem#SM\$Rem#StoMast)	; Затраты на ремонт
	SAVEVALUE	Ubitok,((NarOtk-ST\$Zentr)#UbitSS#KolSS)	; Убыток
	SAVEVALUE	ZatrResSS,((StoResSS/24)#ResSS#VrMod)	; Затраты на резерв


```

;      Расчет ожидаемой прибыли
SAVEVALUE SumPrib, (X$Doxod-(X$StoRem+X$Ubitok+X$ZatrResSS))
MSAVEVALUE Pribil, Stroka, Stolbez, X$SumPrib ; Прибыль
MSAVEVALUE Koef, Stroka, Stolbez, (SR$Zentr/1000)
TERMINATE 1
Stolbez START 10, NP ; Недели 1-5: Резервных CC=3, мастеров=3
EQU 2
RESET
Stolbez START 10, NP ; Недели 6-10: Резервных CC=3, мастеров=3
EQU 3
RESET
Stolbez START 100, NP ; Недели 11-15: Резервных CC=3, мастеров=3
EQU 4
RESET
Stolbez START 10, NP ; Недели 16-20: Резервных CC=3, мастеров=3
EQU 5
RESET
ResSS START 10, NP ; Недели 21-25: Резервных CC=3, мастеров=3
EQU 4
Stroka EQU 2
Stolbez EQU 1
Stolbez START 10, NP ; Недели 1-5: Резервных CC=4, мастеров=3
RESET
EQU 2
RESET
Stolbez START 10, NP ; Недели 6-10: Резервных CC=4, мастеров=3
EQU 3
RESET
Stolbez START 10, NP ; Недели 11-15: Резервных CC=4, мастеров=3
EQU 4
RESET
Stolbez START 10, NP ; Недели 16-20: Резервных CC=4, мастеров=3
EQU 5
RESET
ResSS START 10, NP ; Недели 21-25: Резервных CC=4, мастеров=3
EQU 5
Stroka EQU 3
Stolbez EQU 1
Stolbez START 10, NP ; Недели 1-5: Резервных CC=5, мастеров=3
RESET
EQU 2
RESET
Stolbez START 10, NP ; Недели 6-10: Резервных CC=5, мастеров=3
EQU 3
RESET
Stolbez START 10, NP ; Недели 11-15: Резервных CC=5, мастеров=3

```

Stolbez	EQU	4	
	RESET		
	START	10,NP	; Недели 16-20: Резервных СС=5, мастеров=3
Stolbez	EQU	5	
	RESET		
	START	10	; Недели 21-25: Резервных СС=5, мастеров=3

Программа приведена (в целях сокращения) только для моделирования случая три мастера-ремонтника, а резервных СС — три, четыре и пять. Результаты моделирования для этого случая приведены ниже.

MATRIX	RETRY	INDICES	VALUE
PRIBIL	0		
		1 1	25784.757
		1 2	28592.308
		1 3	37736.527
		1 4	30908.380
		1 5	32741.754
		2 1	35645.390
		2 2	34688.980
		2 3	38479.960
		2 4	34113.060
		2 5	31545.331
		3 1	35572.478
		3 2	31757.949
		3 3	37614.524
		3 4	30745.238
		3 5	37402.018
KOEF	0		
		1 1	.969
		1 2	.976
		1 3	.979
		1 4	.979
		1 5	.987
		2 1	.986
		2 2	.979
		2 3	.976
		2 4	.981
		2 5	.977
		3 1	.980
		3 2	.976
		3 3	.975
		3 4	.979
		3 5	.981

В заключение п. 7.6 еще раз обратим внимание на следующее. Оператор RESET используется обычно при моделировании нестационарных процессов, когда требуется собрать статистику по отдельным интервалам стационарности или исключить влияние переходного периода на собираемую статистическую информацию. Возможности современной вычислительной техники позволяют выполнить вначале моделирования большое число прогонов, отбросить полученную статистику и далее продолжить моделирование. Оператор CLEAR применяется для организации нескольких независимых прогонов модели на разных последовательностях случайных чисел. Для изменения последовательностей случайных чисел нужно использовать оператор RMULT.

7.7. Применение текстовых объектов и потоков данных

Модель представляет собой набор операторов, содержащихся в одном объекте «Модель» и в любом количестве необязательных текстовых объектов. Пример такой модели приведен в п. 1.4.

Текстовые объекты с наборами операторов модели подключаются к объекту «Модель» командой INCLUDE (см. 3.3.3.6). Операторы INCLUDE можно также закреплять за функциональными клавишами. Это позволяет одним нажатием клавиши объекту «Процесс моделирования» передать набор команд и (или) Plus-операторов, содержащихся в текстовом файле.

Текстовые объекты применяются также вместе с потоками данных. Потоки данных позволяют процессу моделирования считывать из файлов и записывать данные в текстовые файлы, а также создавать файлы результатов моделирования для последующего использования.

Поток данных — это последовательность текстовых строк, используемых процессом моделирования. Существуют два типа потоков данных:

- потоки ввода-вывода (I/O или «файловые» потоки) для доступа к файлам;
- потоки в памяти для тестирования или прямого доступа к внутренним данным.

Основной элемент потока данных — текстовая строка, которая является строкой печатных символов, включая пробелы.

Для обработки потоков данных существуют пять блоков GPSS World: OPEN, CLOSE, READ, WRITE, SEEK. Три из них — READ, WRITE, SEEK — выполняют операции только с одной отдельной строкой текста. Перейдем к рассмотрению блоков обработки потоков данных.

7.7.1. Блок OPEN

Блок OPEN предназначен для инициализации потока данных. Формат блока:

OPEN A, [B], [C]

Операнд A — дескриптор потока данных. Операнд определяет тип потока данных. Он обрабатывается как строка. Если это нулевая строка, создается поток

в памяти. Если это канальное имя, такое как «\pipe\турpipe», создается канальный поток. В противном случае создается поток ввода-вывода и предполагается, что операнд А является спецификацией файла.

Операнд В — номер потока данных, произвольное положительное число, задаваемое пользователем. Нумерация потоков введена с целью использования одновременно в одном процессе моделирования нескольких потоков данных. По умолчанию номер потока данных равен 1.

Операнд С — метка блока, в который направляется транзакт в случае ошибки инициализации потока данных. Коды ошибок:

- 0 — нет ошибки;
- 10 — длинное имя файла;
- 11 — ошибка чтения внешнего файла;
- 12 — во время попытки открыть файл был запрещен доступ к памяти.

Например:

```
OPEN      ("Plan.txt"), 3, Kon1
OPEN      ("G\Model\NorPogr.txt")
OPEN      ("")
```

В первом примере открывается поток ввода-вывода, потоку присваивается номер 3 и в случае ошибки открытия активный транзакт направляется к блоку с меткой Kon1. Операндом А указан неполный путь доступа к файлу, поэтому подразумевается, что используется папка объекта «Процесс моделирования». Если файл с указанным именем не найден, то предполагается, что файл создается и ошибка не возникает.

Во втором примере операндом А указан полный путь доступа к файлу, открываемому потоку ввода-вывода по умолчанию присваивается номер один. В случае ошибки открытия активный транзакт направляется к следующему по порядку блоку.

В третьем примере открывается поток в памяти.

ЗАМЕЧАНИЕ

Если указанный операндом А файл найден, во время обработки блока OPEN он полностью загружается в виртуальную память. Все находившиеся в файле данные хранятся как часть объекта «Процесс моделирования» до тех пор, пока поток данных не будет закрыт. Все изменения данных отражаются в файловой системе только после того, как поток данных закрывается блоком CLOSE или библиотечной Plus-процедурой Close().

7.7.2. Блок CLOSE

Блок CLOSE предназначен для закрытия потока данных. Формат блока:

```
CLOSE      A, [B], [C]
```

Операнд А — номер или имя параметра транзакта, в который записывается код ошибки закрытия потока данных. Если такой параметр не существует, он создается.

Операнд В — номер закрываемого потока данных, по умолчанию равен 1, т. е. если операнд В не используется, закрывается поток номер один.

Операнд С — метка блока, в который направляется транзакт в случае ошибки закрытия потока данных. Коды ошибок:

- 0 — нет ошибки;
- 41 — запись файла на диск не произведена из-за ошибки ввода-вывода;
- 42 — файл не был открыт.

Например:

```
CLOSE      Parm_Error, (P1+1)
```

В примере блок CLOSE закрывает поток данных, номер которого задан выражением в скобках. Это выражение вычисляется, округляется и используется в качестве номера потока данных. Результат должен быть положительным целым числом.

Блок CLOSE для потоков ввода-вывода записывает данные из виртуальной памяти в дисковый файл.

7.7.3. Блок READ

Блок READ предназначен для считывания из потока данных текстовой строки. Формат блока:

```
READ      A, [B], [C]
```

Операнд А — номер или имя параметра транзакта, в который записывается считанная из потока ввода или потока в памяти строка, находящаяся на позиции текущей строки. После считывания позиция текущей строки увеличивается на единицу. Если такой параметр активного транзакта не существует, он создается.

Операнд В — номер потока данных, из которого производится считывание. По умолчанию равен 1, т. е. если операнд В не используется, считывание производится из потока номер один.

Операнд С — метка блока, в который направляется транзакт в случае ошибки считывания. Если операнд С не используется, код ошибки все равно сохраняется. Для его получения нужно использовать блок CLOSE. Коды ошибок:

- 21 — во время попытки выполнить чтение был запрещен доступ к памяти;
- 22 — файл не был открыт.

Пример:

```
READ      Stroka_Text, 4, Kon5
```

В примере блок READ считывает текстовую строку из потока данных номер 4 и записывает в параметр транзакта с именем *Stroka_Text*. Если текстовая строка считывается без ошибки, а параметр активного транзакта не существует, он создается. В случае ошибки строка не считывается и активный транзакт направляется к блоку с меткой *Kop5*.

7.7.4. Блок WRITE

Блок WRITE предназначен для передачи текстовой строки потоку данных. Формат блока:

WRITE A, [B], [C], [D]

Операнд A — текстовая строка, которая должна быть передана потоку данных.

Операнд B — номер потока данных, по умолчанию равен 1.

Операнд C — метка блока, в который направляется транзакт в случае ошибки записи. Коды ошибок:

- 0 — нет ошибки;
- 31 — во время попытки выполнить запись был запрещен доступ к памяти;
- 32 — файл не был открыт.

Операнд D — задает режим работы блока WRITE. Если операнд D не используется (по умолчанию) или равен ON, блок WRITE работает в режиме вставки. Если операнд D равен OFF — в режиме замены.

Режим вставки:

- все текстовые строки, находящиеся на или за позицией текущей строки, сдвигаются на одну позицию;
- если позиция текущей строки находится далеко за последней текстовой строкой, она устанавливается сразу после последней строки потока данных;
- копия новой текстовой строки помещается на позицию текущей строки;
- позиция текущей строки увеличивается на единицу.

Режим замены:

- если позиция текущей строки находится далеко за последней текстовой строкой, все промежуточные позиции заполняются нулевыми текстовыми строками;
- текстовая строка, находящаяся на позиции текущей строки, удаляется;
- копия новой текстовой строки помещается на позицию текущей строки;
- позиция текущей строки увеличивается на единицу.

Пример:

WRITE "INITIAL MX\$TDon(1, 1), 420", 5, Met3

В этом примере блок WRITE передает текстовую строку потоку данных с номером 5. Если происходит ошибка, активный транзакт переходит к блоку

с меткой Met3. В противном случае он переходит к следующему по порядку блоку. Если в данном случае поток данных является потоком вывода или потоком в памяти, запись производится в режиме вставки, так как операнд D не используется.

ЗАМЕЧАНИЕ

Если операнды C блоков OPEN, CLOSE, READ и WRITE не используются, ошибка игнорируется. Обычно эти операнды используются для того, чтобы направить активный транзакт на блок CLOSE, который код ошибки помещает в параметр транзакта. Из параметра можно записать код ошибки в сохраняемую ячейку и завершить моделирование. Значение ячейки будет выведено в стандартном отчете.

ЗАМЕЧАНИЕ

При построчном считывании данных из файла может оказаться, что на позиции текущей строки нет строки данных. Это происходит тогда, когда из файла считаны все текстовые строки. В данном случае активный транзакт входит в блок READ и далее направляется к блоку, метка которого указана операндом C. Код ошибки при этом не сохраняется.

7.7.5. Блок SEEK

Блок SEEK устанавливает позицию текущей строки потока данных. Формат блока:

SEEK A, [B]

Операнд A — новая позиция текущей строки.

Операнд B — номер потока данных, по умолчанию равен 1.

Позиция текущей строки — это односвязный индекс, указывающий позицию следующей строки, которую необходимо считать или записать. Она не может быть меньше единицы. При попытке установить ее меньше единицы позиция текущей строки устанавливается равной единице.

51 — код ошибки при изменении текущей позиции.

Пример:

SEEK (Stroka+P1), (Potok+3)

В этом примере, когда транзакт входит в блок SEEK, операнд A вычисляется, округляется и используется как новый номер позиции текущей строки. Операнд B также вычисляется, округляется и используется как номер потока данных. Операнды A и B должны быть целыми положительными числами.

7.7.6. Пример применения текстовых объектов и потоков данных

Пример 7.13. Система авиаперевозок включает два аэропорта. Перевозки выполняются из первого аэропорта во второй и обратно. Время полета между аэропортами распределено по нормальному закону (табл. 7.8).

Таблица 7.8

Характеристики полета	Маршруты полета
	1-2 (2-1)
Математическое ожидание времени полета в прямом (обратном) направлении, час	1,2 (1,4)
Среднеквадратическое отклонение времени полета в прямом (обратном) направлении, час	0,1 (0,15)

Грузы в каждый аэропорт поступают партиями в контейнерах. Количество контейнеров в партии распределено по равномерному закону (табл. 7.9). Интервалы времени между поступлениями партий грузов распределены по экспоненциальному закону (табл. 7.9).

Таблица 7.9

Характеристики потоков грузов	Аэропорты	
	1	2
Минимальное количество контейнеров в партии для отправки, шт.	1	1
Максимальное количество контейнеров в партии для отправки, шт.	15	22
Среднее значение интервалов времени поступления партий грузов, час	0,5	1

Для авиaperевозок используют два типа самолетов в количестве N_1 и N_2 соответственно и грузоподъемностью q_1 и q_2 соответственно. В аэропорту нет фиксированного расписания. Каждый самолет отправляется в полет сразу после его полной загрузки. Время погрузки и время выгрузки одного контейнера распределены по экспоненциальному закону со средними значениями t_1 и t_2 соответственно. В первую очередь используются самолеты с грузоподъемностью q_1 , а при их отсутствии — с грузоподъемностью q_2 .

Построить модель системы авиaperевозок с целью определения среднего времени простоя каждого самолета и коэффициента его использования.

; Пример 7.13

; Определение матриц

DGrus	MATRIX	,3,2	; характеристик грузопотоков
DPol	MATRIX	,4,1	; характеристик полетов
DSam1	MATRIX	,3,4	; показателей самолетов типа 1
DSam2	MATRIX	,2,4	; показателей самолетов типа 2

; Задание исходных данных

Port	EQU	2	; Количество аэропортов
Sam1	EQU	1	; Количество самолетов типа 1


```

KSam2    EQU    1                ; Количество самолетов типа 2
GSam1    EQU    5000            ; Грузоподъемность самолета типа 1
GSam2    EQU    10000           ; Грузоподъемность самолета типа 2
VKont    EQU    100             ; Вес контейнера
TPogr    EQU    0.1             ; Среднее время погрузки контейнера
TVigr    EQU    0.1             ; Среднее время разгрузки контейнера
VrMod    EQU    720             ; Время моделирования
Prog     EQU    10              ; Количество прогонов
;      Инициализация матрицы - запись характеристик грузопотоков
        INCLUDE "DanGrus.txt"; Подключение файла
;      Считывание из файла характеристик полетов
GENERATE    ...,1
OPEN        ("DanPolet.txt"),1,Kon1; Открыть для чтения файл
Met1 READ   SPol,1,Met3          ; Считать одну строку за один раз
SAVEVALUE   Rownum+,1           ; Увеличить строку матрицы на 1
Met2 SAVEVALUE Colnum+,1        ; Увеличить столбец матрицы на 1
MSAVEVALUE  DPol,X$Rownum,X$Colnum,(Word(P$SPol,X$Colnum))
TEST G      X$Colnum,(Port-2),Met2; Есть ли еще столбцы?
SAVEVALUE   Colnum,0
TRANSFER    ,Met1               ; Перейти на следующую строку
Met3 CLOSE   Prob,1,Kon2         ; Закрыть файл данных
TERMINATE   ; Нормальное завершение считывания
Kon1 TERMINATE ; Завершение при ошибке открытия файла
Kon2 TERMINATE ; Завершение при ошибке считывания файла
;      Сегмент имитации прибытия самолетов в аэропорт 1
GENERATE    ...,1
SAVEVALUE   NSam11,Sam1         ; Самолетов типа 1 в аэропорту 1
SAVEVALUE   NSam12,0           ; Самолетов типа 1 в аэропорту 2
SPLIT       (Sam1-1),,NumSam    ; Расщепление по числу самолетов
ASSIGN      TipSam,1            ; Тип 1 самолета
ADOPT       P$NumSam1
Met26 LINK   Angar11,FIFO        ; Самолеты типа 1 на стоянку
GENERATE    ...,1
SAVEVALUE   NSam21,Sam2         ; Самолетов типа 2 в аэропорту 1
SAVEVALUE   NSam22,0           ; Самолетов типа 2 в аэропорту 2
SPLIT       (Sam2-1),,NumSam    ; Расщепление по числу самолетов
ASSIGN      TipSam,2            ; Тип 2 самолета
ADOPT       (P$NumSam1+2)
Met27 LINK   Angar12,FIFO        ; Самолеты типа 2 на стоянку
;      Сегмент имитации аэропорта 1
GENERATE    (Exponential(301,0,MX$DGrus(3,1))); Источник грузов
SAVEVALUE   KPort1+,(INT(DUniform(37,MX$DGrus(1,1),MX$DGrus(2,1))))
ASSIGN      MKont1,(X$KPort1#VKont); Вес отправляемых контейнеров
TEST GE     P$MKont1,GSam1,Met6 ; Достаточно ли контейнеров?
TEST E      X$NSam11,0,Met7     ; Да, а самолет типа 1 есть?

```

```

TEST GE      P$MKont1,GSam2,Met6      ; Достаточно ли контейнеров?
TEST NE      X$NSam21,0,Met6          ; Да, а самолет типа 2 есть?
ASSIGN       KKont,(INT(GSam2/VKont)); Отправляемые контейнеры
SAVEVALUE    KPort1-,P$KKont          ; Оставшиеся контейнеры
SAVEVALUE    NSam21-,1                ; Оставшиеся самолеты типа 2
UNLINK       Angar12,Met9,1           ; Отправка самолета типа 2 под погрузку
Met6 TERMINATE                                ; Уничтожение транзакта - партии контейнеров
Met9 ASSIGN   KKont,(INT(GSam2/VKont)); Отправляемые контейнеры
TRANSFER     ,Met10                    ; Направить на погрузку
Met7 ASSIGN   KKont,(INT(GSam1/VKont)); Отправляемые контейнеры
SAVEVALUE    KPort1,(X$KPort1-P$KKont); Оставшиеся контейнеры
SAVEVALUE    NSam11-,1                ; Оставшиеся самолеты типа 1
UNLINK       Angar11,Met8,1           ; Отправка самолета типа 1 под погрузку
TERMINATE                                ; Уничтожение транзакта - партии контейнеров
Met8 ASSIGN   KKont,(INT(GSam1/VKont)); Погружаемые контейнеры
; Погрузка контейнеров в аэропорту 1
Met10 ASSIGN  TPogrN,AC1                ; Время начала погрузки
SPLIT        (P$KKont-1)                ; Расщепление по количеству контейнеров
ENTER        Pogr1                      ; Занять пункты погрузки
ADVANCE      (Exponential(257,0,TPogr)); Задержаться на время погрузки
LEAVE        Pogr1                      ; Освободить пункты погрузки
ASSEMBLE     P$KKont                    ; Погрузка закончена
TEST E       P$TipSam,1,Met11           ; Самолет какого типа?
MSAVEVALUE   DSam1+,P$NumSam,1,((AC1-P$TPogrN)/(Prog#VrMod))
TRANSFER     ,Met12
Met11 MSAVEVALUE DSam2+,P$NumSam,1,((AC1-P$TPogrN)/(Prog#VrMod))
Met12 ASSIGN  TPol,(Normal(39,MX$DPol(1,1),MX$DPol(2,1))); Время полета
TEST E       P$TipSam,1,Met13           ; Самолет какого типа?
MSAVEVALUE   DSam1+,P$NumSam,2,(P$TPol/(Prog#VrMod))
TRANSFER     ,Met14
Met13 MSAVEVALUE DSam2+,P$NumSam,2,(P$TPol/(Prog#VrMod))
Met14 ADVANCE P$TPol                    ; Имитация полета из 1-2
TRANSFER     ,Met30                    ; Направить на выгрузку
; Выгрузка контейнеров в аэропорту 1
Met24 ASSIGN  TVigrN,AC1                ; Время начала выгрузки
SPLIT        (P$KKont-1)                ; Расщепление по количеству контейнеров
ENTER        Vigr1                      ; Занять пункты выгрузки
ADVANCE      (Exponential(711,0,TVigr)); Имитация выгрузки
LEAVE        Vigr1                      ; Освободить пункты выгрузки
ASSEMBLE     P$KKont                    ; Выгрузка закончена
TEST E       P$TipSam,1,Met25           ; Самолет какого типа?
MSAVEVALUE   DSam1+,P$NumSam,3,((AC1-P$TVigrN)/(Prog#VrMod))
SAVEVALUE    NSam11+,1                 ; Прибыл самолет типа 1
TRANSFER     ,Met26                    ; Отправить в ангар
Met25 MSAVEVALUE DSam2+,P$NumSam,3,((AC1-P$TVigrN)/(Prog#VrMod))

```

```

SAVEVALUE NSam21+,1 ; Прибыл самолет типа 2
TRANSFER ,Met27 ; Отправить в ангар
; Сегмент имитации аэропорта 2
GENERATE (Exponential(301,0,MX$DGrus(3,2))); Источник грузов
SAVEVALUE KPort2+,(INT(DUniForm(377,MX$DGrus(1,2),MX$DGrus(2,2))))
ASSIGN MKont2,(X$KPort2#VKont) ; Вес отправляемых контейнеров
TEST GE P$MKont2,GSam1,Met32 ; Достаточно ли контейнеров?
TEST E X$NSam12,0,Met33 ; Да, а самолет типа 1 есть?
TEST GE P$MKont2,GSam2,Met32 ; Достаточно ли контейнеров?
TEST NE X$NSam22,0,Met32 ; Да, а самолет типа 2 есть?
ASSIGN KKont,(INT(GSam2/VKont)); Отправляемые контейнеры
SAVEVALUE KPort2-,P$KKont ; Количество оставшихся контейнеров
SAVEVALUE NSam22-,1 ; Оставшиеся самолеты типа 2
UNLINK Angar22,Met35,1 ; Отправка самолета типа 2 под погрузку
Met32 TERMINATE ; Уничтожение транзакта - партии контейнеров
Met35 ASSIGN KKont,(INT(GSam2/VKont)); Отправляемые контейнеры
TRANSFER ,Met36
Met33 ASSIGN KKont,(INT(GSam1/VKont)); Отправляемые контейнеры
SAVEVALUE KPort2-,P$KKont ; Оставшиеся контейнеры
SAVEVALUE NSam12-,1 ; Оставшиеся самолеты типа 1
UNLINK Angar21,Met34,1 ; Отправка самолета типа 1 под погрузку
TERMINATE ; Уничтожение транзакта - партии контейнеров
; Погрузка контейнеров в аэропорту 2
Met34 ASSIGN KKont,(INT(GSam1/VKont)); Отправляемые контейнеры
Met36 ASSIGN TPogrN,AC1 ; Время начала погрузки
SPLIT (P$KKont-1) ; Расщепление по количеству контейнеров
ENTER Pogr2 ; Занять пункты погрузки
ADVANCE (Exponential(257,0,TPogr)); Задержаться на время погрузки
LEAVE Pogr2 ; Освободить пункты погрузки
ASSEMBLE P$KKont ; Погрузка закончена
TEST E P$TipSam,1,Met37 ; Самолет какого типа?
MSAVEVALUE DSam1+,P$NumSam,1,((AC1-P$TPogrN)/(Prog#VrMod))
TRANSFER ,Met38
Met37 MSAVEVALUE DSam2+,P$NumSam,1,((AC1-P$TPogrN)/(Prog#VrMod))
Met38 ASSIGN TPol,(Normal(39,MX$DPol(3,1),MX$DPol(4,1))); Время полета
TEST E P$TipSam,1,Met39 ; Самолет какого типа?
MSAVEVALUE DSam1+,P$NumSam,2,(P$TPol/(Prog#VrMod))
TRANSFER ,Met15
Met39 MSAVEVALUE DSam2+,P$NumSam,2,(P$TPol/(Prog#VrMod))
Met15 ADVANCE P$TPol ; Имитация полета из 2-1
TRANSFER ,Met24
; Выгрузка контейнеров в аэропорту 2
Met30 ASSIGN TVigrN,AC1 ; Время начала выгрузки
SPLIT (P$KKont-1) ; Расщепление по количеству контейнеров
ENTER Vigr2 ; Занять пункты выгрузки

```

```

ADVANCE      (Exponential(711,0,TVigr))      ; Имитация выгрузки
LEAVE        Vigr2                            ; Освободить пункты выгрузки
ASSEMBLE     P$KKont                          ; Выгрузка закончена
TEST E       P$TipSam,1,Met31                 ; Самолет какого типа?
MSAVEVALUE   DSam1+,P$NumSam,3,((AC1-P$TVigrN)/(Prog#VrMod))
SAVEVALUE    NSam12+,1                       ; Прибыл самолет типа 1
LINK         Angar21,FIFO                     ; Отправить на стоянку
Met31 MSAVEVALUE DSam2+,P$NumSam,3,((AC1-P$TVigrN)/(Prog#VrMod))
SAVEVALUE    NSam22+,1                       ; Прибыл самолет типа 2
LINK         Angar22,FIFO                     ; Отправить на стоянку
; Сегмент задания времени моделирования
; и расчета результатов моделирования
GENERATE      VrMod                          ; Время моделирования 1 ед. мод. вр.=1 час
TEST E       TG1,1,Met16                     ; Счетчик завершений равен 1? Если да, то
OPEN         ("DanSam1.txt"),1,Kon3           ; Открыть файл (поток 1)
OPEN         ("DanSam2.txt"),2,Kon4           ; Открыть файл (поток 2)
ASSIGN       Rownum,0                         ; Подготовка к выводу
ASSIGN       Colnum,0                         ; Подготовка к выводу
Met17 ASSIGN  Rownum+,1                       ; Увеличить строку матрицы на 1
MSAVEVALUE   DSam1,P$Rownum,4,(MX$DSam1(P$Rownum,1)+
MX$DSam1(P$Rownum,2)+MX$DSam1(P$Rownum,3))
MSAVEVALUE   DSam2,P$Rownum,4,(MX$DSam2(P$Rownum,1)+
MX$DSam2(P$Rownum,2)+MX$DSam2(P$Rownum,3))
Met18 ASSIGN  Colnum+,1                       ; Увеличить столбец матрицы на 1
TEST LE      P$Rownum,Sam1,Kon3              ; Обработана последняя строка ?
WRITE        MX$DSam1(P$Rownum,P$Colnum),1 ; Вывод столбца
WRITE        MX$DSam2(P$Rownum,P$Colnum),2 ; Вывод столбца
TEST G       P$Colnum,3,Met18                ; Есть еще столбцы для вывода?
ASSIGN       Colnum,0                        ; Нет, тогда номер столбца равен нулю
TRANSFER     ,Met17                          ; И перейти к очередной строке
Kon3 CLOSE   Err1,1                           ; Записать файл DanSam1.txt на диск
SAVEVALUE    Err1,P$Err1                     ; Сохранить код ошибки
Kon4 CLOSE   Err2,2                           ; Записать файл DanSam2.txt на диск
SAVEVALUE    Err2,P$Err2                     ; Сохранить код ошибки
Met16 TERMINATE 1

```

Модель имеет четыре текстовых объекта: DanGrus.txt, DanPolet.txt, DanSum1.txt и DanSum2.txt.

Текстовый файл DanGrus.txt подключается к модели командой INCLUDE, так как он содержит операторы GPSS:

```

Pogr1 STORAGE 3
Pogr2 STORAGE 2
Vigr1 STORAGE 3
Vigr2 STORAGE 2
INITIAL       MX$DGrus(1,1),1

```

```
INITIAL      MX$DGrus(2,1),15  
INITIAL      MX$DGrus(3,1),0.5  
INITIAL      MX$DGrus(1,2),1  
INITIAL      MX$DGrus(2,2),22  
INITIAL      MX$DGrus(3,2),0.4
```

Текстовый файл DanPolet.txt содержит числовые характеристики полетов самолетов:

```
2.4  
0.1  
3.2  
0.1
```

Для считывания данных из этого файла используются блоки OPEN, READ и CLOSE.

ЗАМЕЧАНИЕ

Данные из файла DanGrus.txt применяются в блоках GENERATE, имитирующих источники потоков грузов. Если организовать применение данных текстовых объектов наоборот, возникнет ошибка «Operand C. Аргумент масштаба должен быть больше 0.0. Ошибка при заполнении блока GENERATE первым транзактом».

Результаты моделирования выводятся в текстовые файлы DanSam1.txt и DanSam2.txt блоками OPEN, READ и CLOSE.

Текстовый файл DanSam1.txt:

```
.3468133603170526  
.249253224875837  
.3651381736263824  
.961204758819272  
.3481050975100845  
.2399765689687344  
.359935543568054  
.9480172100468729
```

Текстовый файл DanSam2.txt:

```
.3655045820079604  
.2327588740919099  
.3501839265288511  
.9484473826287214
```

Из полученных результатов моделирования видно, что коэффициенты использования двух самолетов типа 1 и самолета типа 2 равны 0,961, 0,948 и 0,948 соответственно, при этом на погрузку затрачивается 0,346, 0,348 и 0,365, на полеты — 0,249, 0,239 и 0,232, на выгрузку — 0,365, 0,359 и 0,350 от общего времени моделирования, равного 720 часам. Простой самолетов составляют 0,039, 0,052 и 0,052 соответственно.

ЗАМЕЧАНИЕ

Результаты моделирования, выведенные в файлы DanSum1.txt и DanSum2.txt, вычисляются в том числе с помощью блоков, определенных пользователем. Они накапливаются в течение нескольких прогонов, а вывести их нужно только в конце моделирования. С этой целью в сегменте задания времени моделирования и расчета результатов моделирования введен блок TEST, который пропустит транзакт, иницирующий вывод в файлы из матриц результатов, только тогда, когда содержимое счетчика завершения будет равно единице. Равенство содержимого счетчика завершения единице свидетельствует о том, что выполнен последний прогон.

ЗАМЕЧАНИЕ

В сегменте имитации прибытия самолетов в аэропорт 1 использованы два блока ADOPT. Блок ADOPT предназначен для изменения номера семейства активного транзакта. Необходимость применения этих блоков вызвана следующим. Блок SPLIT расщепляет вырабатываемый блоком GENERATE один транзакт по числу самолетов соответствующего типа. В последующем при имитации погрузки и выгрузки каждый из этих транзактов вновь расщепляется по числу погружаемых или выгружаемых контейнеров. После окончания погрузки (выгрузки) производится сбор транзактов-контейнеров блоком ASSEMBLE в транзакт-самолет. Но так как все транзакты-самолеты одного семейства, то при сборке контейнеров возможна их «путаница». Для предотвращения этого и изменяется номер семейства. Можно ли было сделать это же изменением операндов D блоков GENERATE? Можно, но удобнее изменять количество самолетов с помощью переменных пользователя Sam1 и Sam2, чем каждый раз отыскивать в модели блоки GENERATE. Тем более, эти переменные пользователя используются в различных блоках модели.

7.8. Непрерывное моделирование

GPSS World позволяет строить модели, в которых можно выразить изменение переменных в виде их производных по времени. Для включения в модель системы обыкновенных дифференциальных уравнений любого порядка в GPSS World имеются команда INTEGRATE и блок INTEGRATION. Интегрирование осуществляется модифицированным методом Рунге-Кутты-Фехлберга пятого порядка с переменной длиной шага (RKF4(5)).

7.8.1. Команда INTEGRATE

Команда INTEGRATE предназначена для задания дифференциального уравнения и пороговых значений переменной. Формат команды:

Name INTEGRATE A, [B], [C], [D], [E]

Name — имя переменной пользователя.

Операнд A — производная переменной пользователя по времени.

Операнд B — пороговое значение 1 переменной пользователя.

Операнд C — метка блока, в который направляется сгенерированный транзакт при преодолении переменной пользователя порогового значения 1 в любом направлении.

Операнд D — пороговое значение 2 переменной пользователя.

Операнд E — метка блока, в который направляется сгенерированный транзакт при преодолении переменной пользователя порогового значения 2 в любом направлении.

Операнды A, B и D могут быть именем, числом, строкой, выражением в скобках или СЧА, а также вызовами Plus-процедур.

В течение процесса моделирования, если задано пороговое значение, при пересечении его в любом направлении создается новый транзакт с нулевым приоритетом и направляется в блок, указанный операндом C или E команды INTEGRATE. Таким образом, непрерывно интегрируемые переменные могут использоваться и в дискретной части процесса моделирования, в том числе и для изменения пороговых значений.

Пример 1.

```
y_      INTEGRATE      (y_/x_+SQR(1+y_^2/x_^2))
x_      EQU             1
y_      EQU             0
```

Пример определяет однородное дифференциальное уравнение первого порядка $y' = \frac{y}{x} + \sqrt{1 + \frac{y^2}{x^2}}$, частное решение которого при начальных условиях

$x = 1$ и $y = 0$ имеет вид: $y + \sqrt{x^2 + y^2} = x^2$. Выражение в команде INTEGRATE используется как производная по времени. Начальные условия переменных задаются командами EQU.

Пример 2.

```
x_      INTEGRATE      (y_), 0.707, Met1
y_      INTEGRATE      (x_)
x_      EQU             1
y_      EQU             0
```

Пример определяет систему двух обыкновенных дифференциальных уравнений, имеющих решения $x = \cos y$ и $y = \sin x$. Для переменной пользователя $x_$ задается пороговое значение. Когда переменная пересекает значение 0,707 в любом направлении, создается новый транзакт и направляется в блок с меткой Met1.

7.8.2. Блок INTEGRATION

Блок INTEGRATION предназначен для включения или выключения процесса интегрирования переменной пользователя. Формат блока:

```
INTEGRATION      A, [B]
```

Операнд A — имя переменной пользователя, заданной командой INTEGRATE.

Операнд В — состояние процесса интегрирования, может принимать значения ON — включен или OFF — выключен. По умолчанию — ON.

Примеры:

INTEGRATION	Skorost
INTEGRATION	Skorost, OFF

В первом примере процесс интегрирования переменной пользователя Skorost активизируется при входе транзакта в блок INTEGRATION. Интегрирование производится по времени. Во втором примере вход активного транзакта в блок INTEGRATION выключает процесс интегрирования переменной пользователя Skorost. Процесс интегрирования будет прекращен до тех пор, пока вновь не будет включен.

7.8.3. Фазы моделирования и ошибки интегрирования

Моделирование протекает в чередующихся *непрерывной* и *дискретной* фазах. При планировании событий процесс моделирования протекает в дискретной фазе. Во время дискретной фазы модельное время не меняется. Между такими моментами процесс моделирования протекает в непрерывной фазе. В течение этой фазы через небольшие приращения модельного времени, называемые мини-шагами, происходит интегрирование. В конце мини-шагов зафиксированные значения интегрируемых переменных возвращают промежуточные значения.

Из непрерывной фазы в дискретную процесс моделирования переходит тогда, когда при пересечении порогового значения генерируется транзакт. Таким образом, непрерывная и дискретная фазы взаимосвязаны. В дискретной фазе переменным пользователя, даже если они интегрируются, могут быть заданы новые значения. Это можно сделать с помощью команды EQU или присваивающих Plus-операторов.

Параметр модели, называемый допустимым отклонением интегрирования, используется для ограничения локальной ошибки округления при интегрировании. Этот параметр применяется ко всем процессам интегрирования, выполняющимся в процессе моделирования. Допустимое отклонение устанавливается на странице Simulation (Моделирование) журнала настроек модели. Выберите пункт меню Edit ► Settings (Правка ► Настройка), затем выберите Simulation (Моделирование). Далее введите необходимое значение в поле ввода, помеченное Integration Tolerance (Допустимое отклонение интегрирования). По умолчанию — 10^{-6} .

7.8.4. Пример непрерывного моделирования

Пример 7.14. Мина вылетает из точки $O(0;0)$ под углом α к горизонту с начальной скоростью V_0 . Полет мины описывается следующей системой дифференциальных уравнений (ДУ) второго порядка:

$$\begin{cases} x'' = -h \cdot \sqrt{(x')^2 + (y')^2} \cdot x', \\ y'' = -q - h \cdot \sqrt{(x')^2 + (y')^2} \cdot y', \end{cases} \quad (7.1)$$

где $h = 0,5\pi C\rho \frac{R^2}{m} = 0,0001492589$;

$C = 0,15$ — коэффициент лобового сопротивления воздуха;

$\rho = 1,1928 \frac{\text{кг}}{\text{м}^3}$ — средняя плотность воздушной среды;

$R = 0,07 \text{ м}$ — радиус шара, заменяющего мину в модели;

$m = 10,057 \text{ кг}$ — масса мины;

$q = 9,81 \text{ м/с}^2$ — ускорение свободного падения;

x, y — текущие координаты летящей мины, м .

Время полета мины измеряется в секундах. Начальные условия:

$$t = 0; x(0) = y(0) = 0, \alpha = 45^\circ, V_0 = 90; x'(0) = V_0 \cos \alpha, y'(0) = V_0 \sin \alpha.$$

Для интегрирования в GPSS World дифференциальные уравнения высоких порядков сводятся к дифференциальным уравнениям первого порядка. Производится это введением новой переменной для каждой промежуточной производной. Сведем систему (7.1) ДУ второго порядка, введя новые переменные $u = x'$ и $w = y'$, к системе из четырех уравнений первого порядка:

$$\begin{cases} x' = u, \\ y' = w, \\ u' = -h \cdot u \cdot \sqrt{u^2 + w^2}, \\ w' = -q - h \cdot w \cdot \sqrt{u^2 + w^2} \end{cases} \quad (7.2)$$

при начальных условиях

$$\begin{pmatrix} u \\ w \\ x \\ y \end{pmatrix}_{t=0} = \begin{pmatrix} 90 \cdot \cos 45^\circ \\ 90 \cdot \sin 45^\circ \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 63,6396 \\ 63,6396 \\ 0 \\ 0 \end{pmatrix}. \quad (7.3)$$

Процесс моделирования следует остановить тогда, когда $y(t) \leq 0$, т. е. когда мина упадет на землю. Следовательно, для u нужно установить пороговое значение 0. При пересечении порогового значения генерируется транзакт. Его нужно использовать для завершения процесса моделирования.

Программа модели приведена ниже. В ней использованы те же обозначения, что и в (7.1) — (7.3), с добавлением знака подчеркивания.

; Пример 7.14

; *Непрерывная часть*

```
h_ EQU 0.0001492589 ; Коэффициент учета баллистических условий
x_ EQU 0 ; Начальное значение координаты x
y_ EQU 0 ; Начальное значение координаты y
u_ EQU 63.6396 ; Начальная по горизонтали скорость полета мины
w_ EQU 63.6396 ; Начальная по вертикали скорость полета мины
```

```
x_ INTEGRATE (u_)
```

```
y_ INTEGRATE (w_), 0, Met1
```

```
u_ INTEGRATE (-h_#u_#SQR(u_^2+w_^2))
```

```
w_ INTEGRATE (-9.81-h_#w_#SQR(u_^2+w_^2))
```

; *Дискретная часть*

```
GENERATE 20
```

```
Met1 TERMINATE 1
```

Программа состоит из непрерывной и дискретной частей. Дискретная часть всегда нужна для запуска и завершения процесса моделирования.

Для наблюдения за процессом моделирования создайте график. Выберите Command ► Create Simulation (Команда ► Создать процесс моделирования). Выберите Window ► Simulation Window ► Plot Window (Окно ► Окно процесса моделирования ► Окно «График»). В окне правки графика в поле Label (Метка) наберите $x_$, в поле Expression (Выражение) наберите также $x_$. В поле Title (Заголовок) наберите «Координаты траектории полета мины x/y ». В поле Time Range (Временной интервал) наберите 15. В поле Min Value (Минимальное значение) введите 0, в поле Max Value (Максимальное значение) — 800. Нажмите кнопки Plot (Построить) и Memorize (Запомнить). После этого добавьте в график другую переменную. Замените значения в окне Edit Plot (Редактор графика) в полях Label (Метка) и Expression (Выражение) на $y_$, а другие значения не меняйте. Нажмите кнопки Plot (Построить), Memorize (Запомнить) и Ok.

После этого проведите интегрирование. В окне главного меню выберите Command ► START (Команда ► START). График должен быть таким, как показано на рис. 7.1. Из стандартного отчета следует, что при данных условиях дальность полета мины равна 754,165 м, время полета — 12,655 сек, а моделирование завершится при $y = 0,075$.

Изменяя условия, можно получать различные характеристики траектории полета мины. Например, требуется определить максимальную высоту траектории полета и время, за которое будет достигнута эта высота. При достижении максимальной высоты вертикальная составляющая скорости полета мины будет равна нулю. Следовательно, нужно завершить моделирование при $y' = w = 0$. Зададим пороговое значение 0 для переменной интегрирования w и запустим модель. В стандартном отчете получим, что максимальная

высота траектории полета мины равна 196,466 м, а время ее достижения составляет 6,26 сек. При этом по горизонтали мина улетит на 384,553 м. Теперь исследуем, как изменятся характеристики траектории полета мины, если не учитывать сопротивление воздуха, т. е. при $h = 0$. Изменим значение h и запустим модель. Получим, что максимальная высота траектории и горизонтальная дальность увеличились и составляют 206,422 м и 412,844 м соответственно при времени полета 6,487 с.

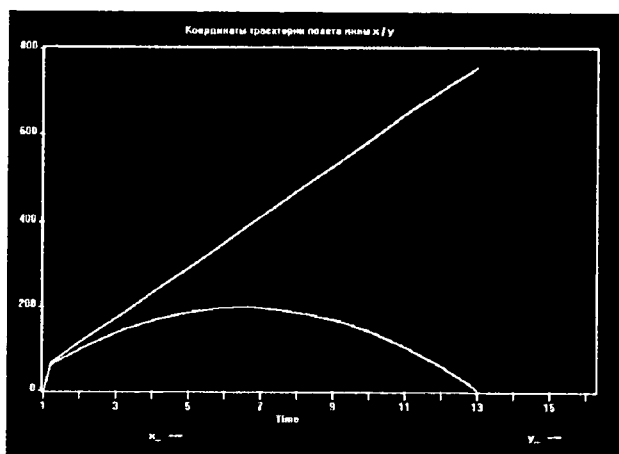


Рис. 7.1. График координат траектории полета мины

ЗАМЕЧАНИЕ

Интегрирование дифференциальных уравнений чувствительно к задаваемым условиям. Например, задайте в рассматриваемом примере $h = 0,1492589$ вместо $h = 0,0001492589$ и запустите процесс моделирования. Произойдет останов по ошибке «Допустимое отклонение интегрирования не может быть соблюдено для: w_». Попытки добиться результата интегрирования изменением точности Integration Tolerance (Допустимая ошибка интегрирования) на странице Simulation (Моделирование) в журнале настроек модели к успеху не приведут. Если все же нужно получить в полном объеме результаты моделирования именно при данных условиях, то рекомендуется создать собственный метод интегрирования с использованием Plus-процедуры (глава 8).

Изложенных методов вполне достаточно для построения моделей сложных систем с использованием GPSS World. Для исследования различных вариантов систем нужно в модели вносить необходимые изменения и получать соответствующие результаты. Однако GPSS World имеет средства, позволяющие эффективно организовывать и проводить машинные эксперименты с моделями. Перейдем к рассмотрению этих средств.

8. Язык PLUS

8.1. Алфавит

Алфавит включает алфавитно-цифровые и специальные символы.

Алфавитно-цифровые символы содержат прописные буквы A...Z, строчные a...z, цифры 0...9 и знак подчеркивания _. Алфавитно-цифровые символы используются для задания имен. В комментариях допускается использовать символы русского алфавита А...Я, а...я.

Специальные символы: «#», «*», «—», «/», «\», «,», «;», «^». Специальные символы используются для обозначения операторов и пунктуации.

Символ # используется для обозначения операции умножения, а символ * — в качестве оператора косвенной адресации СЧА GPSS. Однако их можно поменять. Для этого нужно выбрать в главном меню Edit ▶ Settings (Правка ▶ Настройки) и на странице Simulation (Моделирование) журнала настроек модели установить флажок Switch # and * (Переключить # и *).

8.2. Имена

Имена — это созданные пользователем последовательности алфавитно-цифровых символов для идентификации объектов переменных и программ.

Имя должно начинаться с буквы и может содержать от одного до 250 алфавитно-цифровых символов. Имена не должны совпадать с ключевыми словами и с СЧА GPSS. Для избежания таких совпадений рекомендуется в каждом имени употреблять знак подчеркивания. Например, Run_Progr.

GPSS World не различает регистр алфавита, поэтому использование прописных или строчных букв не имеет никакого значения. Только строковые константы и комментарии сохраняют запись строчными буквами. Все остальные строчные буквы, встречаемые в тексте, автоматически переводятся в прописные.

Поименованные величины — это имена, помещенные в поле метки или в присваивающий Plus-оператор. Они используются для обозначения объектов или для хранения величин.

Если поименованные величины используются для обозначения операторов GPSS, определяющих объекты, они называются *метками* объектов. Если поименованные величины применяются для присвоения значений, они называются *переменными пользователя*.

Переменные пользователя могут быть локальными и глобальными. *Локальные* переменные пользователя объявляются внутри Plus-процедуры с помощью оператора TEMPORARY. Один оператор TEMPORARY создает одну или несколько временных неинициализированных именованных величин. Их можно использовать только внутри той процедуры, в которой они были определены. Остальные переменные пользователя являются *глобальными*. Они могут использоваться во всей модели. Значение переменной пользователя может быть присвоено не только командой EQU и Plus-оператором, но и при помощи численного интегрирования (задается командой INTEGRATE).

Имена могут использоваться для идентификации операторов в Plus-процедуре. Для этого оператор нужно начать с имени, после которого следует двоеточие. Такие операторы применяются для перехода в Plus-операторе GOTO. *Метки операторов* имеют локальную область видимости, т. е. к ним можно обращаться только в той процедуре, в которой они определены.

Plus-процедуре и Plus-эксперименту обязательно присваиваются имена. *Имена процедур* имеют глобальную область видимости. Любая процедура может вызываться из любого места модели.

Эксперименты являются специальным видом процедур. Они идентичны процедурам в синтаксисе, за исключением того, что вместо ключевого слова PROCEDURE записывается слово EXPERIMENT. Эксперименты могут вызываться только командой CONDUCT.

Метки матриц, как и переменные пользователя, могут иметь локальную или глобальную область видимости.

Локальные матрицы объявляются с помощью оператора TEMPORARY MATRIX. Один оператор TEMPORARY MATRIX создает одну временную неинициализированную матрицу. Локальные матрицы доступны только в той процедуре, в которой были объявлены. В этом случае матрицы создаются при вызове процедуры и уничтожаются, как и локальные поименованные величины, при завершении выполнения процедуры.

Глобальные матрицы объявляются командой MATRIX. Глобальные матрицы являются в отличие от локальных матриц постоянными и к ним можно обращаться из любого места модели.

8.3. Выражения

Выражения строятся с использованием операторов и вызовов процедур, соединяющих факторы. Выражения применяются в Plus-процедурах и в операндах операторов GPSS.

В выражениях переменные пользователя, элементы матриц, ячейки, параметры транзактов могут иметь значения различных типов данных (целочисленный (integer), вещественный (real) и строковый (string)). Кроме того, каждый из них сначала имеет неопределенную форму, которая исключает их использование до того, как им будет присвоено первое значение.

Факторы — это основные элементы выражений. Факторами операторов GPSS являются:

- строковые константы;
- вещественные константы;
- целочисленные константы;
- имена;
- элементы Plus-матрицы;
- обращения к процедурам;
- стандартные числовые атрибуты.

ЗАМЕЧАНИЕ

Элементы Plus-матрицы записываются в следующем виде: Name [ExpressionList], например, DMatr1[P\$Stroka,X\$Stolbez]. Список ExpressionList может содержать от одного до шести выражений, соответствующих измерениям конкретной матрицы. Каждое из них может быть простым целым числом или сложным Plus-выражением.

Для объединения факторов в выражения используются операторы и вызовы процедур. Объединение производится согласно правилам, связанным с конкретным оператором или процедурой.

Операторы, используемые для объединения факторов в выражения, приведены в табл. 8.1.

Таблица 8.1

Оператор	Действие	Результат	Арность	Ассоциативность
—	Отрицание	Аддитивная инверсия	Унарный	Право
^	Возведение в степень	Арифметический показатель степени	Бинарный	Право
NOT или ~	Инверсия	1 (TRUE) или 0 (FALSE)	Унарный	Право
AND	Логическое И	1 (TRUE) или 0 (FALSE)	Бинарный	Левое
OR	Логическое ИЛИ	1 (TRUE) или 0 (FALSE)	Бинарный	Левое
G или >	Больше	1 (TRUE) или 0 (FALSE)	Бинарный	Левое
L или <	Меньше	1 (TRUE) или 0 (FALSE)	Бинарный	Левое
E или =	Равно	1 (TRUE) или 0 (FALSE)	Бинарный	Левое

Таблица 8.1 (продолжение)

NE или /=	Не равно	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
LE или <=	Меньше или равно	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
GE или >=	Больше или равно	1 (TRUE) или 0 (FALSE)	Бинарный	Лево
# (или *)	Умножение	Арифметическое произведение	Бинарный	Лево
/	Деление	Арифметическое частное	Бинарный	Лево
\	Целочисленное деление	Целочисленное деление	Бинарный	Лево
@	Деление по модулю	Целочисленный остаток	Бинарный	Лево
+	Сложение	Арифметическая сумма	Бинарный	Лево
-	Вычитание	Арифметическая разность	Бинарный	Лево

Арифметические операторы перечислены в порядке убывания их приоритетов. В соответствии с этими приоритетами и вычисляются выражения. При вычислении операнды автоматически преобразуются в необходимые типы данных. Например, строки, объединенные арифметическим знаком «+», автоматически преобразуются в числа, которые складываются. Полученный результат автоматически не округляется. Для округления необходимо вызвать соответствующую процедуру. Так же процедура, имеющая строковый аргумент, при необходимости преобразует число в строковый эквивалент, т. е. в свой ASCII эквивалент.

В отчетах можно использовать экспоненциальное представление больших чисел. Для этого выберите Edit ▶ Settings (Правка ▶ Настройки) и перейдите на страницу Reports (Отчеты). После этого установите или снимите флажок Scientific Notation (Экспоненциальное представление). Если флажок установлен, то, например, число 2200 будет представляться как 2.2E3.

8.4. Plus-операторы

В отличие от операторов GPSS, которые записываются одной текстовой строкой, Plus-операторы могут записываться в несколько строк. Plus-операторы могут быть использованы только внутри оператора PROCEDURE. Plus-процедуры могут записываться в любом месте модели, кроме тела другой процедуры. Plus-операторы могут быть простыми и составными. Каждый Plus-оператор должен заканчиваться разделителем «;» (точка с запятой).

Plus-операторы приведены в табл. 8.2.

Таблица 8.2

Оператор	Назначение	Синтаксис
Присваивания	Устанавливает значение именованной величины или элемента матрицы	LValue=Expression LValue — именованная величина или элемент матрицы; Expression — Plus-выражение
BEGIN/END	Создает блок Plus-операторов	BEGIN StatementList END; StatementList — последовательность Plus-операторов, каждый из которых должен заканчиваться «;»
EXPERIMENT	Определяет Plus-эксперимент	EXPERIMENT Name (ArgumentList) Statement Name — имя процедуры, данное пользователем; ArgumentList — список имен (формальных параметров), разделенных запятыми; Statement — блок Plus-операторов.
GOTO	Передаёт управление внутри процедуры	GOTO Label; Label — метка Plus-оператора.
IF/THEN	Проверяет выражение, если результат TRUE, выполняет блок Plus-операторов	IF (Expression) THEN Statement Expression — проверяемое Plus-выражение; Statement — блок Plus-операторов.
IF/THEN/ELSE	Проверяет выражение и в зависимости от результата выполняет один из двух блоков Plus-операторов	IF (Expression) THEN Statement1 ELSE Statement2 Expression — проверяемое Plus-выражение; Statement1 — блок Plus-операторов, выполняемый, если результат проверки TRUE; Statement2 — блок Plus-операторов, выполняемый, если результат проверки FALSE.
С меткой	Указывает место перехода для оператора GOTO	Label: Statement Label — метка Plus-оператора, задаваемая пользователем; Statement — Plus-оператор.
PROCEDURE	Определяет Plus-процедуру	PROCEDURE Name (ArgumentList) Statement Name — имя процедуры, заданное пользователем; ArgumentList — список имен (формальных параметров), разделенных запятыми; Statement — блок Plus-операторов.

Таблица 8.2 (продолжение)

Вызова процедуры	Вызывает библиотечную процедуру	ProcedureName (ExpressionList) ProcedureName — имя процедуры; ExpressionList — список аргументов процедуры, разделенных запятыми (Plus-выражения).
RETURN	Останавливает выполнение процедуры и может возвращать результат ее выполнения	RETURN Expression; Expression — Plus-выражение (необязательный операнд).
TEMPORARY	Определяет и ограничивает область действия локальных переменных пользователя и локальных матриц	TEMPORARY NameList; NameList — список имен, определенных пользователем и разделенных запятыми; TEMPORARY MATRIX Name[IntegerList]; Name — имя локальной матрицы, определенной пользователем; IntegerList — последовательность положительных чисел от 0 до 1, разделенных запятыми — количество элементов в каждом измерении матрицы.
WHILE/DO	Выполняет действие несколько раз	WHILE (Expression) DO Statement Expression — проверяемое Plus-выражение; Statement — блок Plus-операторов.

8.5. Библиотека процедур

Существует два типа библиотек: библиотека пользователя и встроенная библиотека GPSS World. Библиотека пользователя создается самим пользователем. Встроенная библиотека включает готовые к использованию в любом Plus-выражении следующие процедуры:

- обслуживающие;
- файловые;
- динамического вызова;
- математические;
- вероятностных распределений;
- строковые;
- запроса состояния транзакта.

Имена процедур встроенной библиотеки, их назначение и правила записи при обращении к ним приведены в табл. 8.3.

Таблица 8.3

Процедура	Назначение	Синтаксис
<i>Обслуживающие процедуры</i>		
DoCommand	Транслирует и выполняет оператор GPSS	DoCommand (CommandString) CommandString — строковое представление оператора GPSS. Процедура может вызываться только из экспериментов.
Anova	Производит многофакторный дисперсионный анализ	StandartError=Anova(ResultMatrixName, ReplicateDimension, InteractionLimit) ResultMatrixName — имя матрицы, содержащей данные для дисперсионного анализа; ReplicateDimension — измерение матрицы, используемое для реплик; InteractionLimit — ограничение учитываемых при дисперсионном анализе взаимодействий между факторами, может принимать значения 1, 2, 3.
Exit	Завершает сеанс работы с GPSS World	Exit(ExitCode) ExitCode — -1, 0, и 1 — «не сохранять ничего», «запросить сохранение измененных файлов» и «Сохранять все» соответственно (открытые объекты «Модель» и «Процесс моделирования»).
<i>Математические процедуры</i>		
ABS	Абсолютное значение	ABS(Expression) Expression — Plus-выражение.
ATN	Арктангенс	ATN(Expression) Expression — Plus-выражение.
COS	Косинус	COS(Expression) Expression — Plus-выражение.
EXP	Экспонента	EXP(Expression) Expression — Plus-выражение.
INT	Выделяет целую часть с отбрасыванием дробной части	INT(Expression) Expression — Plus-выражение.
LOG	Натуральный логарифм	LOG(Expression) Expression — Plus-выражение.
SIN	Синус	SIN(Expression) Expression — Plus-выражение.
SQR	Извлекает квадратный корень	SQR(Expression) Expression — Plus-выражение.
TAN	Тангенс	TAN(Expression) Expression — Plus-выражение.

Таблица 8.3 (продолжение)

<i>Процедуры запроса состояния транзакта</i>		
QueryXNExist	Возвращает 1, если транзакт есть в модели, иначе — 0	QueryXNExist(TrasactionNumber) TrasactionNumber — номер транзакта.
QueryXNParameter	Возвращает значение параметра транзакта	QueryXNParameter(TrasactionNumber, Parameter) TrasactionNumber — номер транзакта; Parameter — номер или имя параметра транзакта.
QueryXNAssemblySet	Возвращает номер семейства транзакта	QueryXNAssemblySet(TrasactionNumber) TrasactionNumber — номер транзакта.
QueryXNPtiority	Возвращает приоритет транзакта	QueryXNPtiority(TrasactionNumber) TrasactionNumber — номер транзакта.
QueryXNM1	Возвращает время входа транзакта в систему	QueryXNM1(TrasactionNumber) TrasactionNumber — номер транзакта.
<i>Строковые процедуры</i>		
Align	Вставляет одну строку в другую с выравниванием по правому краю	ReturnString=Align(InsertString, SourceString, Offset) InsertString — строка, которую нужно вставить; SourceString — строка, в которую будет вставлена первая строка; Offset — смещение в строке SourceString для самого правого символа строки InsertString, начиная с которого будет вставляться строка.
Catenate	Объединяет две строки	ReturnString=Catenate(String1, String2) String1, String2 — строки, которые должны быть в результирующей строке первой и второй соответственно.
Copies	Создает одну строку из нескольких копий исходной	ReturnString=Copies(SourceString, Count) SourceString — строка, которая будет размножена; Count — количество копий.
Find	Вычисляет смещение одной строки в другой строке	ReturnInteger=Find(TestString, SourceString) TestString — строка для поиска; SourceString — строка, в которой будет выполнен поиск строки TestString.
Left	Возвращает левую часть заданной строки	ReturnString=Left(SourceString, MaxCount) SourceString — исходная строка; MaxCount — максимальное количество символов в результирующей строке.

Таблица 8.3 (продолжение)

Length	Подсчитывает количество символов в строке	ReturnInteger=Length(SourceString) SourceString — исходная строка.
Lowercase	Переводит прописные буквы строки в строчные	ReturnString=Lowercase(SourceString) SourceString — исходная строка.
Place	Вставляет одну строку в другую с выравниванием по левому краю	ReturnString=Place(InsertString,SourceString,Offset) InsertString — строка, которую нужно вставить; SourceString — строка, в которую будет вставлена первая строка; Offset — смещение в строке SourceString для самого левого символа строки InsertString, начиная с которого будет вставляться строка.
PolyCatenate	Производит объединение двух или более строк в одну строку	ReturnString=PolyCatenate(String1,...) String1,... — строки, объединяемые по порядку.
Right	Возвращает правую часть заданной строки	ReturnString=Right(SourceString,MaxCount) SourceString — исходная строка; MaxCount — максимальное количество символов в результирующей строке.
String	Преобразует данные в их строковый эквивалент	ReturnString=String(Datum) Datum — исходные данные.
StringCompare	Сравнивает две строки	ReturnInteger=StringCompare(String1, String2) String1, String2 — сравниваемые строки.
Substring	Возвращает часть строки	ReturnString=Substring(SourceString, Offset, MaxCount) SourceString — исходная строка; Offset — смещение первого символа подстроки; MaxCount — максимальное количество символов в выделяемой подстроке.
Trim	Удаление пробелов с начала и конца строки	ReturnString=Trim(SourceString) SourceString — исходная строка.
Uppercase	Преобразует строчные буквы строки в прописные	ReturnString=Uppercase(SourceString) SourceString — исходная строка.

Таблица 8.3 (продолжение)

Value	Преобразует строку в числовой эквивалент	ReturnReal=Value(Datum) Datum — элемент данных, подлежащих преобразованию.
Word	Извлекает слово из строки	ReturnString=Word(SourceString,WordNumber) SourceString — исходная строка; WordNumber — порядковый номер слова в строке SourceString.

Процедуры потоков данных

Open	Открывает файл	ReturnCode=Open(DataStream,FileNameString) DataStream — номер нового открываемого файла (потока данных); FileNameString — имя открываемого файла; ReturnCode — код результата выполнения операции: 0 — нет ошибки; 10 — длинное имя файла; 11 — ошибка чтения внешнего файла; 12 — во время попытки открыть файл был запрещен доступ к памяти.
Close	Закрывает файл	ReturnCode=Close(DataStream) DataStream — номер закрываемого файла (потока данных); ReturnCode — результат выполнения операции: 0 — нет ошибки; 41 — запись файла на диск не произведена из-за ошибки ввода-вывода; 42 — файл не был открыт.
Read	Считывает текстовую строку из файла	ReturnString=Read(DataStream) DataStream — номер файла (потока данных), из которого производится считывание; коды результата выполнения операции: 0 — нет ошибки; 21 — во время попытки выполнить чтение был запрещен доступ к памяти; 22 — файл не был открыт.
Write	Записывает текстовую строку в файл	ReturnCode= Write(DataStream,SourceString) DataStream — номер файла, в который производится запись; SourceString — строка, которая записывается в файл; ReturnCode — код результата выполнения операции: 0 — нет ошибки; 31 — во время попытки выполнить запись был запрещен доступ к памяти; 32 — файл не был открыт.

Таблица 8.3 (продолжение)

Seek	Устанавливает позицию текущей строки файла	PreviousLinePosition=Seek(DataStream,NewLinePosition) DataStream — номер файла, в котором устанавливается новая позиция текущей строки; NewLinePosition — новое значение позиции текущей строки; PreviousLinePosition — предыдущее значение позиции текущей строки; 51 — код ошибки: файл не был открыт.
<i>Процедуры динамического вызова</i>		
Call	Вызывает внешнюю CDECL-функцию без аргументов, т. е. с пустым аргументом	ReturnCode=Call(ExecutableFileName, FunctionName) ExecutableFileName — имя модуля (.DLL или .EXE файла), содержащего вызываемую функцию; FunctionName — имя вызываемой функции.
Call_Integer	Вызывает внешнюю CDECL-функцию с одним целочисленным аргументом	ReturnCode=Call_Integer(ExecutableFileName, FunctionName,Argument) ExecutableFileName — имя модуля (.DLL или .EXE файла), содержащего вызываемую функцию; FunctionName — имя вызываемой функции.
Call_String	Вызывает внешнюю CDECL-функцию с одним символьным аргументом (char)	ReturnCode=Call_String(ExecutableFileName, FunctionName,Argument) ExecutableFileName — имя модуля (.DLL или .EXE файла), содержащего вызываемую функцию; FunctionName — имя вызываемой функции.
Call_Real	Вызывает внешнюю CDECL-функцию с одним вещественным аргументом (double)	ReturnCode=Call_Real(ExecutableFileName, FunctionName,Argument) ExecutableFileName — имя модуля (.DLL или .EXE файла), содержащего вызываемую функцию; FunctionName — имя вызываемой функции.

Рассмотрим примеры применения языка Plus, в которых обратим внимание на ряд особенностей. Эти особенности позволят избежать ошибок и повысить эффективность разработки и отладки моделей, содержащих Plus-процедуры.

8.6. Подготовка текстовых объектов

В некоторых моделях требуется использовать данные, представленные в виде матриц. Операторы GPSS для инициализации элементов матрицы можно набрать вручную при подготовке объекта «Модель». Язык Plus позволяет построить процедуры, с помощью которых создаются текстовые объекты, содер-

жащие необходимые операторы для инициализации элементов матриц. Созданные таким образом объекты подключаются к объекту «Модель» командой INCLUDE.

Пример 8.1. В модели будут использованы исходные данные, содержащиеся в двух матрицах Plan(2,5) и Pogr(2,5). Требуется создать два текстовых объекта с операторами GPSS для инициализации этих матриц.

Решение данного примера начните с создания двух текстовых объектов, содержащих только числовые значения элементов матриц. Для этого выберите в главном меню File ► New (Файл ► Новый документ), а затем Text File (Текстовый файл).

Введите построчно элементы матрицы Plan:

43	37	0	19	45
15	68	93	11	19

Сохраните эту матрицу в файле с именем Plan1.txt. Для этого выберите File ► Save As... (Файл ► Сохранить как...). В появившемся окне вместо Untitled укажите Plan1 и нажмите Сохранить.

Аналогичным образом создайте текстовый объект, содержащий матрицу Pogr:

3	12	16	8	5
8	10	9	11	7

Сохраните также эту матрицу в файле с именем Pogr1.txt.

Программа, которая вводит данные из текстовых объектов Plan1.txt и Pogr1.txt и выводит в текстовые объекты Plan.txt и Pogr.txt с операторами инициализации матриц Plan и Pogr соответственно, приведена ниже.

; Пример 8.1

```

Str EQU 2 ; Количество строк в матрице
Stol EQU 5 ; Количество столбцов в матрице
Plan MATRIX ,2,5 ; Определение матрицы Plan
Pogr MATRIX ,2,5 ; Определение матрицы Pogr
; Ввод данных
GENERATE ,,,1 ; Создать только один транзакт
OPEN ("Plan1.txt"),1,Kon8; Открыть для чтения файл Plan1
OPEN ("Pogr1.txt"),2,Flag10; Открыть для чтения файл Pogr1
Met1 READ SPlan,1,Kon3; Считать одну строку за один раз в параметр SPlan
; В конце или в случае ошибки перейти к метке Kon3
Kon3 READ SPogr,2,Kon4; Считать одну строку за один раз в параметр SPogr
; В конце или в случае ошибки перейти к метке Kon4
SAVEVALUE Rownum+,1 ; Увеличить строку матрицы на 1
Met2 SAVEVALUE Colnum+,1 ; Увеличить столбец матрицы на 1
MSAVEVALUE Plan,X$Rownum,X$Colnum,(Word(P$SPlan,X$Colnum))
MSAVEVALUE Pogr,X$Rownum,X$Colnum,(Word(P$SPogr,X$Colnum))
; Сохранить значения параметров в ячейках матриц
TEST G X$Colnum,(Stol-1),Met2;Есть ли еще столбцы для загрузки

```

```

SAVEVALUE Colnum,0 ; Нет столбцов? Перейти на следующую строку
TRANSFER ,Met1 ; Перейти на метку Met1
Kon4 CLOSE Prob,1,Kon9 ; Закрыть файл данных Plan1
CLOSE Prob,2,Flag11 ; Закрыть файл данных Pogr1
TERMINATE 1 ; Нормальное завершение считывания файлов Plan1, Pogr1
Kon8 TERMINATE 1 ; Завершение при ошибке открытия файла Plan1
Kon9 TERMINATE 1 ; Завершение при ошибке закрытия файла Plan1
Flag10TERMINATE 1 ; Завершение при ошибке открытия файла Pogr1
Flag11TERMINATE 1 ; Завершение при ошибке закрытия файла Pogr1

```

; Вывод операторов инициализации матриц

```

OPEN ("Plan.txt"),3,Kon1; Открыть для вывода файл Plan
OPEN ("Pogr.txt"),4,Kon2; Открыть для вывода файл Pogr
ASSIGN Rownum,0 ; Подготовка к выводу в цикле из матриц
ASSIGN Colnum,0 ; Подготовка к выводу в цикле из матриц
Met3 ASSIGN Rownum+,1 ; Увеличить строку матрицы на 1
Met4 ASSIGN Colnum+,1 ; Увеличить столбец матрицы на 1
TEST LE P$Rownum,Str,Kon5 ; Была ли обработана последняя строка?
WRITE (VivOper1()),3,Kon6,OFF
WRITE (VivOper2()),4,Kon7,OFF
TEST G P$Colnum,(Stol-1),Met4
ASSIGN Colnum,0
TRANSFER ,Met3
Kon5 CLOSE Prob3,3,Kon10
CLOSE Prob4,4,Kon11
TERMINATE 1 ; Нормальное завершение вывода в файлы Plan, Pog
Kon1 TERMINATE 1 ; Завершение при ошибке открытия файла вывода Plan
Kon2 TERMINATE 1 ; Завершение при ошибке открытия файла вывода Pogr
Kon6 TERMINATE 1 ; Завершение при ошибке записи в файл вывода Plan
Kon7 TERMINATE 1 ; Завершение при ошибке записи в файл вывода Pogr
Kon10 TERMINATE 1 ; Завершение при ошибке закрытия файла вывода Plan
Kon11 TERMINATE 1 ; Завершение при ошибке закрытия файла вывода Pogr

```

; Plus-процедуры

```

PROCEDURE VivOper1() BEGIN
  TEMPORARY Arg1,Arg3,Arg4,Arg5,Arg6;
  Arg1="Initial MX$Plan(";
  Arg3=",";
  Arg4=")";
  Arg5=MX$Plan(P$Rownum,P$Colnum);
  Arg6=" ";
  RETURN (Polycatenate
    (Arg1,
    P$Rownum,
    Arg3,
    (Polycatenate
    (P$Colnum,
    Arg4,

```



```

        Arg5,
        Arg6))));
END;
;
PROCEDURE VivOper2() BEGIN
    TEMPORARY Arg1,Arg3,Arg4,Arg5,Arg6;
    Arg1="Initial MX$Pogr(";
    Arg3=",";
    Arg4="),"";
    Arg5=MX$Pogr(P$Rownum,P$Colnum);
    Arg6=" ";
    RETURN  (Polycatenate
        (Arg1,
        P$Rownum,
        Arg3,
        (Polycatenate
        (P$Colnum,
        Arg4,
        Arg5,
        Arg6))));
END;

```

Программа содержит два сегмента: ввода данных и вывода операторов инициализации матриц, а также две процедуры VivOper1 и VivOper2 для вывода в файлы Plan и Pogr соответственно. Блок GENERATE генерирует только один транзакт, поэтому при запуске модели примените команду START 1. В результате работы программы будут созданы два файла.

Файл вывода — текстовый объект Plan.txt:

```

Initial MX$Plan(1,1),43
Initial MX$Plan(1,2),37
Initial MX$Plan(1,3),0
Initial MX$Plan(1,4),19
Initial MX$Plan(1,5),45
Initial MX$Plan(2,1),15
Initial MX$Plan(2,2),68
Initial MX$Plan(2,3),93
Initial MX$Plan(2,4),11
Initial MX$Plan(2,5),19

```

Файл вывода — текстовый объект Pogr.txt:

```

Initial MX$Pogr(1,1),3
Initial MX$Pogr(1,2),12
Initial MX$Pogr(1,3),16
Initial MX$Pogr(1,4),8
Initial MX$Pogr(1,5),5
Initial MX$Pogr(2,1),8

```

```
Initial MX$Pogr(2,2),10
Initial MX$Pogr(2,3),9
Initial MX$Pogr(2,4),11
Initial MX$Pogr(2,5),7
```

Plus-процедурам VivOper1 и VivOper2 аргументы не передаются. Все необходимые значения, существующие в процессе моделирования, глобальны и поэтому доступны Plus.

В каждой процедуре оператор RETURN обрабатывает составную строку и возвращает ее в блок WRITE для записи в соответствующий файл. Задание файла в блоке WRITE производится операндом потока данных. При записи в файл Plan.txt номер потока данных равен трем, а при записи в файл Pogr.txt — четверем. Эти же номера потоков были использованы и при открытии файлов.

8.7. Реализация собственного метода интегрирования

В п. 7.8 отмечалось, что при необходимости пользователь с помощью языка Plus может реализовать собственный метод интегрирования. Продемонстрируем такую возможность на примере. Условия примера возьмем те же, что и в примере 7.14. В этом случае интегрирование можно выполнить по следующей рекурсивной процедуре:

$$\begin{cases} u_{k+1} = u_k \left(1 - h \cdot \Delta t \cdot \sqrt{u^2 + w^2} \right), \\ w_{k+1} = -9,81 + w_k \left(1 - h \cdot \Delta t \cdot \sqrt{u^2 + w^2} \right), \\ x_{k+1} = x_k + 0,5 \cdot \Delta t \cdot (u_{k+1} + u_k), \\ y_{k+1} = y_k + 0,5 \cdot \Delta t \cdot (w_{k+1} + w_k). \end{cases}$$

Для интегрирования текущих координат x и y используется формула трапеций. Шаг интегрирования Δt выбирается согласно принципу: чем меньше, тем точнее конечный результат. Начальные условия интегрирования см. (7.3).

; Пример 8.2

; Задание начальных условий интегрирования

```
u_ EQU 63.6396; Начальное значение горизонтальной скорости
w_ EQU 63.6396; Начальное значение вертикальной скорости
x_ EQU 0 ; Начальное значение координаты x
y_ EQU 0 ; Начальное значение координаты y
h_ EQU 0.0001492589; Баллистический коэффициент
dt_ EQU 1 ; Шаг интегрирования
```

; Сегмент интегрирования

; Дискретная часть

```
GENERATE ...,1
SAVEVALUE Dal,(Poisk())
TERMINATE 1
```

```

;      Непрерывная часть
PROCEDURE Poisk() begin
    TEMPORARY uk,wk,xk,yk,t_,Dal;
    t_=0;
Met1: t_=t_+dt_;
    uk=u_#(1-h_#dt_#SQR(u_^2+w_^2));
    wk=-9.81#dt_+w_#(1-h_#dt_#SQR(u_^2+w_^2));
    xk=x_+0.5#dt_#(u_+uk);
    yk=y_+0.5#dt_#(w_+wk);
    u_=uk;
    w_=wk;
    x_=xk;
    y_=yk;
    if (y_>0) then GOTO Met1;
    Dal=x_;
    RETURN Dal;
END;

```

ЗАМЕЧАНИЕ

Фактические аргументы вызова процедуры Poisk, определяющие начальные условия, являются простыми. Однако каждый из этих аргументов может быть сложным выражением, вычисляющимся одновременно с вызовом процедуры. Результат вычисления каждого аргумента используется вместо аргумента, объявленного в определении процедуры. Окончательное значение каждого аргумента определяется в соответствии с его позицией.

ЗАМЕЧАНИЕ

Локальные переменные в Plus-процедуре автоматически не инициализируются, поэтому они должны быть определены до выполнения действий с ними. Если в примере 8.3 не инициализировать локальную переменную $t_$, т. е. не присвоить ей значение, равное нулю, произойдет останов по ошибке «Operand B. Недопустимое выражение. Элемент данных не был инициализирован».

В табл. 8.4 приведены результаты интегрирования в зависимости от величины шага интегрирования.

Таблица 8.4

Шаг интегрирования	Координата x , м	Время полета t , с
1	771,967	13
0,1	756,564	12,7
0,01	754,453	12,66

При $\Delta t = 0,01$ полученные значения дальности и времени полета практически не отличаются от значений, вычисленных встроенными средствами GPSS World, т. е. методом Рунге-Кутты-Фихлберга ($x = 754,165$ м и $t = 12,655$ с).

8.8. Отладка Plus-процедур

При отладке Plus-процедур нужно применять режим трассировки. Режим трассировки устанавливается пользователем. Для этого выберите Edit ▶ Settings (Правка ▶ Настройки), перейдите на страницу Reports (Отчеты) журнала настроек модели. Установите флажок Plus Trace (Трассировка Plus), нажмите Применить и Ok. Сохраните модель. В режиме трассировки каждый вызов Plus-процедуры сопровождается записью в окне Journal (Журнал). Трассировочные сообщения показывают глубину обработки и составляют список вычисленных аргументов процедуры вызова.

Трассировку можно проводить и внутри Plus-процедур. Для этого создайте одну или несколько фиктивных Plus-процедур, которые никаких действий производить не будут. Тогда можно будет многократно их вызывать в исходной Plus-процедуре. При установленном флажке Plus Trace (Трассировка Plus) в окне Journal (Журнал) будут выдаваться трассировочные сообщения, показывающие состояние переменных.

Ниже приведена модель, рассмотренная в примере 8.2, с пятью фиктивными процедурами Poisk1, Poisk2, Poisk3, Poisk4 и Poisk5.

; Пример 8.3

; *Задание начальных условий интегрирования*

```
u_   EQU   63.6396      ; Начальное значение горизонтальной скорости
w_   EQU   63.6396      ; Начальное значение вертикальной скорости
x_   EQU   0             ; Начальное значение координаты x
y_   EQU   0             ; Начальное значение координаты y
h_   EQU   0.0001492589; Баллистический коэффициент
dt_  EQU   1             ; Шаг интегрирования
```

; *Сегмент интегрирования*

; *Дискретная часть*

```
GENERATE    ...,1
SAVEVALUE   Dal,(Poisk())
TERMINATE   1
```

; *Непрерывная часть*

```
PROCEDURE   Poisk() begin
    TEMPORARY uk,wk,xk,yk,t_,Dal;
    t_=0;
Met1: t_=t_+dt_;
    Poisk1(t_);
    uk=u_#(1-h_#dt_#SQR(u_^2+w_^2));
    Poisk2(uk);
    wk=-9.81#dt_+w_#(1-h_#dt_#SQR(u_^2+w_^2));
    Poisk3(wk);
    xk=x_+0.5#dt_#(u_+uk);
    Poisk4(xk);
    yk=y_+0.5#dt_#(w_+wk);
```

```

    Poisk5(yk);
    Poisk();
    u_=uk;
    w_=wk;
    x_=xk;
    y_=yk;
    if (y_>0) then GOTO Met1;
    Dal=x_;
    RETURN Dal;
END;
PROCEDURE Poisk1(Arg1) RETURN;
PROCEDURE Poisk2(Arg2) RETURN;
PROCEDURE Poisk3(Arg3) RETURN;
PROCEDURE Poisk4(Arg4) RETURN;
PROCEDURE Poisk5(Arg5) RETURN;

```

В примере 8.3 исходная (отлаживаемая) процедура Poisk вызывается только один раз, а фиктивные процедуры столько раз, сколько будет сделано шагов до выполнения условия окончания интегрирования. Фрагмент выдаваемых трассировочных сообщений приведен ниже.

```

10/03/03 12:16:10 Model Translation Begun.
10/03/03 12:16:10 Ready.
10/03/03 12:16:10 POISK Procedure registered.
10/03/03 12:16:10 POISK1 Procedure registered.
10/03/03 12:16:10 POISK2 Procedure registered.
10/03/03 12:16:10 POISK3 Procedure registered.
10/03/03 12:16:10 POISK4 Procedure registered.
10/03/03 12:16:10 POISK5 Procedure registered.
10/03/03 12:16:15 START 1
10/03/03 12:16:15 Simulation in Progress.
10/03/03 12:16:15 PLUS Trace POISK()
10/03/03 12:16:15 PLUS Trace POISK1( 1.0000000)
10/03/03 12:16:15 PLUS Trace POISK1 Returns: 0
10/03/03 12:16:15 PLUS Trace POISK2( 62.7847102)
10/03/03 12:16:15 PLUS Trace POISK2 Returns: 0
10/03/03 12:16:15 PLUS Trace POISK3( 52.9747102)
10/03/03 12:16:15 PLUS Trace POISK3 Returns: 0
10/03/03 12:16:15 PLUS Trace POISK4( 63.2121551)
10/03/03 12:16:15 PLUS Trace POISK4 Returns: 0
10/03/03 12:16:15 PLUS Trace POISK5( 58.3071551)
10/03/03 12:16:15 PLUS Trace POISK5 Returns: 0
10/03/03 12:16:15 PLUS Trace POISK1( 2.0000000)
10/03/03 12:16:15 PLUS Trace POISK1 Returns: 0

```

10/03/03 12:16:15	PLUS Trace	POISK2(62.0148899)	
10/03/03 12:16:15	PLUS Trace	POISK2 Returns:	0
10/03/03 12:16:15	PLUS Trace	POISK3(42.5151730)	
10/03/03 12:16:15	PLUS Trace	POISK3 Returns:	0
10/03/03 12:16:15	PLUS Trace	POISK4(125.6119552)	
10/03/03 12:16:15	PLUS Trace	POISK4 Returns:	0
10/03/03 12:16:15	PLUS Trace	POISK5(106.0520967)	
10/03/03 12:16:15	PLUS Trace	POISK5 Returns:	0
10/03/03 12:16:15	PLUS Trace	POISK1(3.0000000)	
10/03/03 12:16:15	PLUS Trace	POISK1 Returns:	0
10/03/03 12:16:15	PLUS Trace	POISK2(61.3189198)	
10/03/03 12:16:15	PLUS Trace	POISK2 Returns:	0
10/03/03 12:16:15	PLUS Trace	POISK3(32.2280410)	
10/03/03 12:16:15	PLUS Trace	POISK3 Returns:	0
10/03/03 12:16:15	PLUS Trace	POISK4(187.2788601)	
10/03/03 12:16:15	PLUS Trace	POISK4 Returns:	0
10/03/03 12:16:15	PLUS Trace	POISK5(143.4237037)	
10/03/03 12:16:15	PLUS Trace	POISK5 Returns:	0

По трассировочным сообщениям можно проследить изменение пяти переменных на каждом шаге интегрирования и при необходимости внести соответствующие правки. Очередной шаг интегрирования начинается значением времени t_* , выдаваемым в ходе трассировки фиктивной процедуры Poisk1. В приведенном фрагменте показаны три шага интегрирования. Начало каждого шага выделено жирным шрифтом.

При отладке Plus-процедур следует также применять команду SHOW.

9. Эксперименты с имитационными моделями

Необходимым условием успешного моделирования является обеспечение адекватности модели, т. е. функционального или структурного тождества модели исследуемому процессу или объекту относительно цели исследования. Однако этого недостаточно. Для успешного исследования необходимо разработать план эксперимента над моделью, правильно обработать и интерпретировать результаты моделирования. GPSS World имеет средства, которые позволяют провести:

- дисперсионный анализ;
- регрессионный анализ;
- собственный эксперимент пользователя.

ЗАМЕЧАНИЕ

Согласно [22] дисперсионный анализ называется *отсеивающим экспериментом*, а регрессионный анализ — *оптимизирующим экспериментом*.

Прежде чем рассматривать эти средства, остановимся на основных понятиях теории планирования экспериментов.

9.1. Основные понятия теории планирования экспериментов

9.1.1. Необходимость планирования экспериментов

Теория исходит из абстрактной схемы сложной системы, называемой «черным ящиком» (рис. 9.1). Считается, что исследователь может наблюдать входы и выходы «черного ящика» (имитационной модели) и по результатам наблюдений определять зависимость между входами и выходами.

Эксперимент на имитационной модели будем рассматривать состоящим из *наблюдений*, а каждое наблюдение — из *прогонов модели*.

Входные переменные $x_1, x_2, \dots, x_m$ называются *факторами*. Выходная переменная y называется *наблюдаемой переменной (реакцией, откликом)*.

Факторное пространство — это множество факторов, значения которых исследователь может контролировать в ходе подготовки и проведения модельного эксперимента.

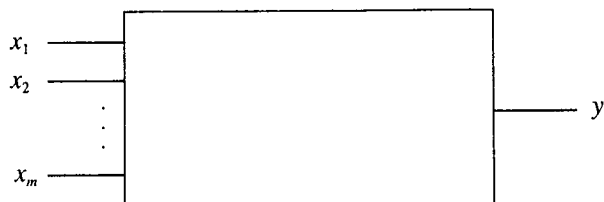


Рис. 9.1. Абстрактная схема системы

Каждый фактор имеет уровни. *Уровни* — это значения, которые устанавливаются для каждого фактора при определении условий прогона модели в наблюдении.

Целью эксперимента является нахождение функции y , при этом предполагается, что значение отклика складывается из двух составляющих:

$$y = f(x_1, x_2, \dots, x_m) + e(x_1, x_2, \dots, x_m),$$

где $f(x_1, x_2, \dots, x_m)$ — функция отклика (неслучайная функция факторов);

$e(x_1, x_2, \dots, x_m)$ — ошибка эксперимента (случайная величина);

$x_1, x_2, \dots, x_m$ — определенное сочетание уровней факторов из факторного пространства.

Очевидно, что y является случайной переменной, так как зависит от случайной величины $e(x_1, x_2, \dots, x_m)$. Дисперсия $D[y]$, которая характеризует точность измерения, равна дисперсии ошибки опыта: $D[y] = D[e]$.

Дисперсионный анализ — это статистический метод анализа результатов наблюдений, зависящих от различных, одновременно действующих факторов, выбор наиболее важных факторов и оценка их влияния.

В условиях эксперимента факторы могут варьировать, благодаря чему исследовать влияние фактора на наблюдаемую переменную. Если влияние некоторого фактора на наблюдаемую переменную изменяется при изменении уровня некоторого другого фактора, говорят, что между факторами существует *взаимодействие*.

Суть анализа заключается в разложении общей вариации случайной величины на независимые слагаемые — *эффекты*, каждый из которых характеризует влияние того или иного фактора (*главный эффект*) или их взаимодействия (*эффект взаимодействия*).

Эксперимент, в котором реализуются все возможные сочетания уровней факторов, называется *полным факторным экспериментом* (ПФЭ).

Общее число различных сочетаний уровней в ПФЭ для m факторов можно вычислить по формуле:

$$S = k_1 \cdot k_2 \cdot k_3 \cdot \dots \cdot k_i \cdot \dots \cdot k_m,$$

где k_i — число уровней i -го фактора.

Если число уровней для всех факторов одинаково, то $S = k^m$. Каждому сочетанию уровней факторов соответствует одно наблюдение.

Недостаток ПФЭ — большие затраты на подготовку и проведение, так как с увеличением числа факторов и их уровней число наблюдений в эксперименте растет. Например, если имеется шесть факторов с двумя уровнями каждый, то даже при одном прогоне модели в каждом наблюдении нужно $S = 2^6 = 64$ наблюдения. Очевидно, что каждый прогон удваивает это число, следовательно, увеличивает затраты машинного времени.

Возрастает также число взаимодействий различных порядков. В табл. 9.1 приведено число главных эффектов и эффектов взаимодействий различных порядков, которые могут быть выявлены при полном исследовании 2^m сочетаний уровней.

Таблица 9.1

m	2^m	Число главных эффектов	Число эффектов взаимодействий				
			1-го порядка	2-го порядка	3-го порядка	4-го порядка	5-го порядка
5	32	5	10	10	5	1	
6	64	6	15	20	15	6	1
7	128	7	21	35	35	21	7
8	256	8	28	56	70	56	28

На практике бывает необходимо исследовать влияние на наблюдаемую переменную, например, десяти факторов ($m = 10$), каждый из которых имеет четыре уровня ($k = 4$). В этом случае $S = k^m = 4^{10} = 1\,048\,576$.

Такого рода задачи и явились одной из причин возникновения теории планирования экспериментов.

Планирование экспериментов — один из разделов математической статистики, изучающий рациональную организацию измерений, подверженных случайным ошибкам.

Планом эксперимента называется совокупность значений факторов, при которых находятся значения оценок функции отклика, удовлетворяющих некоторому критерию оптимальности, например, точности.

Различают стратегическое планирование эксперимента и тактическое планирование эксперимента.

9.1.2. Стратегическое планирование эксперимента

Целью стратегического планирования эксперимента является определение количества наблюдений и сочетаний уровней факторов в них для получения наиболее полной и достоверной информации о поведении системы.

При стратегическом планировании эксперимента должны быть решены две основные задачи:

1. Идентификация факторов.
2. Выбор уровней факторов.

Под *идентификацией факторов* понимается их ранжирование по степени влияния на значение наблюдаемой переменной.

По итогам идентификации целесообразно разделить все факторы на две группы — первичные и вторичные.

Первичные — это факторы, исследование которых необходимо провести.

Вторичные — факторы, которые не являются предметом исследования, но влиянием которых нельзя пренебречь.

Выбор уровней факторов производится с учетом двух противоречивых требований:

- уровни фактора должны перекрывать весь возможный диапазон его изменения;
- общее количество уровней по всем факторам не должно приводить к большому количеству наблюдений.

Отыскание компромиссного решения, удовлетворяющего этим требованиям, и является задачей стратегического планирования эксперимента.

Так как ПФЭ может потребовать много машинного времени, необходимо располагать методами отбора факторов, оказывающих существенное влияние на отклик. Оказывается, что если исследователя не интересуют взаимодействия высокого порядка, можно получить большое количество информации с помощью исследования лишь некоторой части ($1/2$, $1/4$, $1/8$ и т. д.) всех возможных сочетаний уровней факторов.

Если в эксперименте производится лишь часть возможных наблюдений, т. е. уменьшается выборка, эксперимент называется *частным факторным экспериментом* (ЧФЭ).

Когда используется выборка меньшая, чем того требует ПФЭ, плата за это осуществляется риском смешивания эффектов. Под *смешиванием* понимается то, что исследователь, измеряя один эффект, в то же время измеряет, возможно, и некоторый другой эффект. Например, если главный эффект смешивается с взаимодействием более высокого порядка, то эти два эффекта уже невозможно отделить друг от друга. Таким образом, если анализ показывает наличие некоторого эффекта, то нельзя с уверенностью сказать, главный ли это эффект, или эффект взаимодействия, или некоторая аддитивная комбинация этих эффектов.

При построении плана ЧФЭ исследователь должен определить эффекты, смешивание которых он может допустить. Успех ЧФЭ достигается в случае, если его план позволяет не смешивать ни один главный эффект с другим.

Если число факторов невелико (обычно меньше пяти), то ЧФЭ нецелесообразен вследствие смешивания эффектов, не позволяющего различить главные эффекты и важные взаимодействия.

В качестве примера рассмотрим план *дробного факторного эксперимента* (ДФЭ) — одного из видов ЧФЭ, с полным числом возможных сочетаний 2^5 .

В ДФЭ каждый фактор имеет два уровня — *нижний* и *верхний*, поэтому общее число наблюдений $S = 2^m$. Планы, построенные по такому способу, обладают определенными свойствами (симметричности, нормированности, ортогональности и ротатабельности), обеспечивающими повышение качества проводимых экспериментов.

Этот план строится следующим образом. Смешиваем взаимодействия высокого порядка и разделим 32 возможных сочетания на четыре блока одинакового размера (по восемь сочетаний в каждом блоке). Затем выберем один из этих блоков для проведения эксперимента. Это означает, что мы должны смешать три эффекта в процессе объединения наблюдений в блоки. Мы сделаем это, выбирая двух- и трехфакторные взаимодействия с общим фактором. В данном случае мы выбрали *BCE* и *ADE* (*порождающие функции* [19]), имеющие общий фактор *E*. Смешав эти взаимодействия, мы автоматически смешали *ABCD*. Поэтому мы получили так называемое *тождество* или *определяющее разбиение*, $I = BCE = ADE = ABCD$, которое было использовано для деления 32 наблюдений на четыре блока по восемь наблюдений в каждом. Это означает, что мы утратили возможность распознавания таких эффектов. Используя это тождество как основу для смешивания в нашем плане, мы должны сделать то же и со следующими эффектами:

$$\begin{aligned} A &= DE = BCD = ABCE, \\ B &= CE = ACD = ABDE, \\ C &= BE = ABD = ACDE, \\ D &= AE = ABC = BCDE, \\ E &= BC = AD = ABCDE, \\ AB &= CD = ACE = BDE, \\ AC &= BD = ABE = CDE. \end{aligned}$$

Получили семь групп смешивания. Из этого следует, что если взаимодействий нет или ими можно пренебречь, то мы можем по восьми наблюдениям получить информацию о каждом из пяти главных эффектов (*A*, *B*, *C*, *D* и *E*) и использовать данные эффектов взаимодействий *AB* и *BC* для оценки ошибки дисперсионного анализа.

9.1.3. *Тактическое планирование эксперимента*

Целью *тактического планирования эксперимента* является определение необходимого количества прогонов модели в каждом наблюдении.

Так как имитационное моделирование представляет собой статистический эксперимент, то при его проведении необходимо получить достоверный результат с заданной точностью.

В общем случае количество прогонов модели (объем выборки), необходимое для получения оценок наблюдаемой переменной с заданной точностью, зависит от следующих факторов:

- вида распределения наблюдаемой переменной;
- коррелированности между собой элементов выборки;
- наличия и длительности переходного режима функционирования моделируемой системы.

При отсутствии сведений о перечисленных факторах для повышения точности оценок истинного значения наблюдаемой переменной увеличивается количество прогонов модели (объем выборки) для каждого наблюдения, т. е. для каждого сочетания уровней факторов, выбранного на этапе стратегического планирования эксперимента.

Если случайные значения наблюдаемой переменной не коррелированы и их распределение не меняется от прогона к прогону, то выборочное среднее можно считать нормально распределенным. Для случая, когда целью моделирования является определение среднего значения некоторого случайного параметра, требуемое число прогонов N_T модели в каждом наблюдении определяется по формуле:

$$N_T = \frac{\sigma_\alpha^2}{\epsilon^2} \cdot t_\alpha^2, \quad (9.1)$$

где ϵ — точность оценки;

σ_α — среднеквадратическое отклонение;

t_α — аргумент функции Лапласа, при заданном уровне значимости $\alpha/2$ находится по таблице 9.2, в которой даны наиболее актуальные пары α и t_α .

Таблица 9.2

α	0,8	0,85	0,9	0,95	0,99	0,995	0,999
t_α	1,28	1,44	1,65	1,96	2,58	2,81	3,30

Если требуемое значение среднеквадратического отклонения σ_α до начала эксперимента не известно хотя бы ориентировочно, целесообразно выполнить пробное количество прогонов N_0 и вычислить на их основе выборочное среднеквадратическое отклонение или дисперсию, значение которой подставить в (9.1) и получить предварительную оценку числа прогонов N_T . Затем выполнить $N_T - N_0$ оставшиеся прогонов, периодически уточняя оценку и число прогонов N_T .

Для случая, когда целью моделирования является нахождение вероятности P исхода какого-либо события, требуемое число прогонов N_T модели в каждом наблюдении определяется по формуле:

$$N_T = \frac{P \cdot (1 - P)}{\epsilon^2} \cdot t_\alpha^2. \quad (9.2)$$

Параметр t_α находится также по таблице значений функции Лапласа в соответствии с заданной величиной достоверности (табл. 9.2). При вычислении N_T по формуле (9.2) значение вероятности P до эксперимента также неизвестно. Поэтому для определения N_T нужно поступить так, как и при нахождении N_T по формуле (9.1).

В ряде случаев для определения необходимого количества N_T прогонов можно использовать упрощенную формулу:

$$N_T = \frac{t_\alpha^2}{4\epsilon^2}. \quad (9.3)$$

Формула (9.3) ориентирована на «худший случай», т. е. для $P = 0,5$. Следовательно, она, как правило, будет давать завышенные значения для N_T . Но при не очень высоких требованиях к точности применение этой формулы вполне оправдано, так как некоторая избыточность N_T будет скомпенсирована исключением процедуры определения ориентировочного значения P .

Теперь перейдем к средствам GPSS World. Для проведения дисперсионного и регрессионного анализов GPSS World имеет автоматические генераторы экспериментов.

Сначала рассмотрим, как проводятся эксперименты с применением автоматических генераторов, а потом разработаем и проведем собственный эксперимент.

9.2. Дисперсионный анализ (отсеивающий эксперимент)

Пример 9.1. Двухпроцессорный вычислительный комплекс (ВК) обслуживает три автоматизированных рабочих места (АРМ). Первое АРМ имеет абсолютный приоритет по отношению к остальным, а второе — более высокий приоритет по отношению к третьему АРМ. От первого АРМ поступают запросы на обслуживание, интервалы между которыми распределены равномерно $[t_1, t_2]$ с. Интервалы времени поступления запросов от второго и третьего АРМ распределены по экспоненциальному закону с параметрами t_3 и t_4 соответственно. На обслуживание одного запроса 1-й категории любой процессор затрачивает время, экспоненциально распределенное с параметром t_5 с, 2-й категории — t_6 с, 3-й категории — t_7 с. В оперативной памяти ВК организован накопитель емкостью L запросов.

Каждый процессор имеет конечную надежность. Длительность интервалов между отказами распределена по экспоненциальному закону с параметром t_8 с. Длительность восстановления распределена по нормальному закону с параметрами t_9 с и t_{10} с. При отказе любого процессора обслуживаемые и прерванные запросы теряются.

Разработать имитационную модель с целью исследования в течение 2 ч функционирования двухпроцессорного ВК зависимости вероятности потерь запросов от емкости накопителя, времени обработки запросов, интервалов времени выхода из строя и времени восстановления процессоров. Провести дисперсионный анализ результатов моделирования.

Программа модели приведена ниже.

; Пример 9.1

; Задание исходных данных

```
VrMod EQU 3600 ; Время моделирования, 1 ед. модельного времени = 1с
L_ EQU 1 ; Емкость накопителя, запросов
T1_ EQU 11 ; Средний интервал времени поступления запросов от АРМ1
T2_ EQU 3 ; Среднеквадратическое отклонение интервала времени
; поступления запросов от АРМ1
T3_ EQU 10 ; Средний интервал времени поступления запросов от АРМ2
T4_ EQU 12 ; Средний интервал времени поступления запросов от АРМ3
T5_ EQU 5 ; Среднее время обработки запросов от АРМ1
T6_ EQU 7 ; Среднее время обработки запросов от АРМ2
T7_ EQU 9 ; Среднее время обработки запросов от АРМ3
T8_ EQU 10000 ; Средний интервал времени возникновения отказов
T9_ EQU 15 ; Среднее время восстановления
T10_ EQU 2 ; Среднеквадратическое отклонение времени восстановления
Koef EQU 1 ; Коэффициент изменения T9_ и T10_
```

; Описание арифметических выражений

```
Ver VARIABLE N$Term1/N$Met0
```

; Сегмент имитации потока запросов с АРМ1

```
GENERATE T1_,T2_,,2 ; Назначение приоритета 2
ASSIGN 1,1 ; Признак принадлежности к потоку 1
ASSIGN 2,(Exponential(11,0,T5_))
TRANSFER ,Met0 ; Отправить на Сру1 или Сру2
```

; Сегмент имитации потока запросов с АРМ2

```
GENERATE (Exponential(2,0,T3_)),,1 ; Назначение приоритета 1
ASSIGN 1,2 ; Признак принадлежности к потоку 2
ASSIGN 2,(Exponential(13,0,T6_))
TRANSFER ,Met0 ; Отправить на Сру1 или Сру2
```

; Сегмент имитации потока запросов с АРМ3

```
GENERATE (Exponential(2,0,T4_))
ASSIGN 1,3 ; Признак принадлежности к потоку 3
ASSIGN 2,(Exponential(15,0,T7_))
```

```
Met0 TEST E P1,1,Met5
```

; Сегмент имитации обработки запросов с АРМ1 на Сру1

```
GATE FV Cру1,Met1 ; Проверка: исправен ли Сру1? Если да, то
GATE NU Cру1,Met1 ; Проверка: свободен ли Сру1? Если да, то
Met3 PREEMPT Cру1,PR ; Захватить Сру1
ADVANCE P2 ; Имитация обработки запроса
RETURN Cру1 ; Освободить Сру1
UNLINK Nak,Met3,1 ; Вывести из накопителя очередной транзакт
TERMINATE ; Обработанные запросы
```

; Сегмент имитации обработки запросов с АРМ1 на Сру2

```

Met1  GATE FV      Cpu2,TERM1 ; Проверка: исправен ли Сру2? Если да, то
      GATE NU      Cpu2,Met2  ; Проверка: свободен ли Сру2? Если да, то
Met4  PREEMPT     Cpu2,PR     ; Захватить Сру2
      ADVANCE      P2         ; Имитация обработки запроса
      RETURN       Cpu2       ; Освободить Сру2
      UNLINK       Nak,Met4,1 ; Вывести из накопителя очередной транзакт
      TERMINATE    ; Обработанные запросы

```

; Сегмент имитации обработки запросов с АРМ2 и АРМ3 на Сру1

```

Met5  GATE FV      Cpu1,Met6  ; Проверка: исправен ли Сру1? Если да, то
      GATE NU      Cpu1,Met6  ; Проверка: свободен ли Сру1? Если да, то
Met7  SEIZE        Cpu1       ; Занять Сру1
      ADVANCE      P2         ; Имитация обработки запроса
      RELEASE      Cpu1       ; Освободить Сру1
      UNLINK       Nak,Met7,1 ; Вывести из накопителя очередной транзакт
      TERMINATE    ; Обработанные запросы

```

; Сегмент имитации обработки запросов с АРМ2 и АРМ3 на Сру2

```

Met6  GATE FV      Cpu2,TERM1 ; Проверка: исправен ли Сру2? Если да, то
      GATE NU      Cpu2,Met2  ; Проверка: свободен ли Сру2? Если да, то
Met8  SEIZE        Cpu2       ; Занять Сру2
      ADVANCE      P2         ; Имитация обработки запроса
      RELEASE      Cpu2       ; Освободить Сру2
      UNLINK       Nak,Met8,1 ; Вывести из накопителя очередной транзакт
      TERMINATE    ; Обработанные запросы

```

; Сегмент имитации работы накопителя

```

Met2  TEST L       CH$Nak,L_,Term1; Имеется ли свободное место?
      LINK         Nak,P1      ; Помещение запросов в накопитель по P1
Term1 TERMINATE    ; Потерянные запросы

```

; Сегмент имитации сбоя Сру1

```

      GENERATE     ,,,1
Met10 ADVANCE      (Exponential(13,0,T8_)); Розыгрыш интервала между отказами
      FUNAVAIL     Cpu1,RE,Term1,,RE,Term1,RE,Term1; Имитация отказа
      ADVANCE      (Normal(4,T9_#Kcoef,T10_#Kcoef)); Имитация восстановления
      FAVAIL       Cpu1 ; Возвращение в работоспособное состояние
      TRANSFER     ,Met10

```

; Сегмент имитации сбоя Сру2

```

      GENERATE     ,,,1
Met9  ADVANCE      (Exponential(13,0,T8_)); Розыгрыш интервала между отказами
      FUNAVAIL     Cpu2,RE,Term1,,RE,Term1,RE,Term1; Имитация отказа
      ADVANCE      (Normal(4, T9_#Kcoef,T10_#Kcoef)) ; Имитация восстановления
      FAVAIL       Cpu2 ; Возвращение в работоспособное состояние
      TRANSFER     ,Met9

```

; Сегмент задания времени моделирования и расчета результата моделирования

```

      GENERATE     VrMod
      SAVEVALUE    Ver,V$Ver ; Расчет вероятности потерь запросов
      TERMINATE    1

```

Откройте модель Пример 9.1. Выберите Edit ▶ Insert Experiment ▶ Screening... (Правка ▶ Вставить эксперимент ▶ Отсеивающий...). Откроется диалоговое окно Screening Experiment Generator (Генератор отсеивающего эксперимента) (рис.9.2), поля которого нужно заполнить необходимой информацией.

Screening Experiment Generator			
Experiment Name		BK_DisExp	
'Run Procedure' Name		BK_DisExpRun	
Factors			
	Name (User Variable)	Value 1	Value 2
A	L_	1	5
B	T5_	3	15
C	T6_	3	20
D	T7_	4	18
E	Koef	0.5	2
F			
Fraction: ☐ Full ☒ Half ☐ Quarter ☐ Eighth ☐ Sixteenth			
			Run Count: 16
Result: Expression N\$Term1/N\$Met0			
☒ Generate Run Procedure ☒ Load F11 with CONDUCT Command			
Insert Experiment		Cancel	Help
Alias Groups			

Рис. 9.2. Диалоговое окно Screening Experiment Generator (Генератор отсеивающего эксперимента)

В поля Experiment Name (Имя эксперимента) и Run Procedure Name (Имя процедуры запуска) введите BK_DisExp и BK_DisExpRun соответственно.

Дальше идет группа полей Factors (Факторы). В примере 9.1 определяется вероятность потерь запросов, поступающих с АРМ. В качестве факторов, влияние которых на вероятность потерь запросов необходимо исследовать, рекомендуются следующие:

- $L_$ — емкость накопителя;
- $T5_$ — среднее время обработки запросов от АРМ1;
- $T6_$ — среднее время обработки запросов от АРМ2;
- $T7_$ — среднее время обработки запросов от АРМ3;
- Koef — коэффициент изменения среднего времени восстановления $T9_$ и средноквадратического отклонения $T10_$ времени восстановления.

Каждое имя фактора должно быть именем *переменной пользователя*. Оно должно начинаться с буквы и не совпадать с ключевым словом, СЧА или классом СЧА. Это было учтено при разработке модели Пример 9.1. Для изменения среднего времени восстановления T9_ и среднеквадратического отклонения T10_ времени восстановления введена переменная пользователя Koef. За счет этого вместо двух факторов используется один (Koef).

В GPSS World максимальное количество факторов, влияние которых на функцию отклика можно исследовать посредством дисперсионного анализа, равно шести. Для каждого фактора необходимо выбрать два уровня. Рекомендуется выбирать уровни, значительно отстоящие друг от друга. Это необходимо для получения также значительно отличающихся откликов.

В рассматриваемом примере выбраны пять факторов и для каждого фактора — по два уровня (см. рис. 9.2). Введите их последовательно, начиная с фактора А.

Следующей идет группа Fraction (Часть дробного эксперимента). Эксперимент, проводимый в GPSS World, может быть ПФЭ или ДФЭ. Группа Fraction позволяет это задавать. Задайте пока Eighth (1/8). Справа под Run Count появится число 16. Это количество наблюдений, которые необходимо сделать. Количество прогонов модели в каждом наблюдении будет указано позже.

В поле Expression (Выражение) группы Result (Результат) введите выражение

$$N\$Term1/N\$Met0$$

по которому вычисляется вероятность потерь запросов с АРМ.

После группы Result (Результат) идут два флажка, позволяющие выбирать опции. При выборе опции Generate Run Procedure GPSS World создает вместе с экспериментом стандартную процедуру запуска, которую пользователь может откорректировать согласно своим требованиям. Выбор опции Load F11 with CONDUCT Command закрепляет соответствующую команду CONDUCT за функциональной клавишей [F11]. Тогда после создания объекта «Процесс моделирования» для запуска эксперимента нужно только нажать функциональную клавишу [F11]. Выберите обе опции.

Перед созданием эксперимента необходимо изучить группы смешивания с целью осуществления стратегического планирования эксперимента (см. п. 9.1.2). Для этого нужно нажать кнопку Alias Groups (Группы смешивания). Появится диалоговое окно Alias Groups (Группы смешивания) (рис. 9.3). При изучении групп смешивания необходимо вначале найти отсутствующие факторы, а затем факторы, которые неразличимы, так как находятся в одной группе смешивания. Факторы обозначаются одной буквой. Например, пятый фактор обозначается буквой Е, а взаимодействие первого и пятого факторов — АЕ.

Из рис. 9.3 видно, что фактор Е отсутствует, о нем не будет никакой информации. Факторы С и D находятся в одной группе и, следовательно, будут неразличимы.

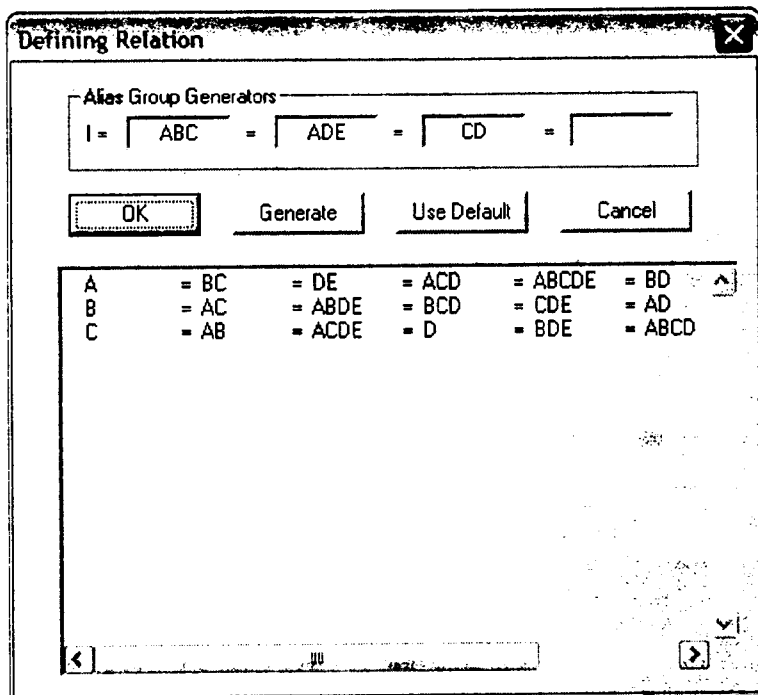


Рис. 9.3. Группы смешивания

Распределение факторов по группам смешивания, а следовательно, и получаемых эффектов, можно изменить, используя поле Alias Generators (Порождающие функции). Для этого нужно изменить порождающие функции и нажать кнопку Generate (Создать). В качестве порождающих функций следует использовать наборы взаимодействий высших порядков. Когда GPSS World создает группы смешивания, сначала он расширяет множество порождающих функций, формируя все возможные комбинации при помощи деления по модулю два и добавляя их к расширенному множеству порождающих функций. Например, если ABD и ADE — порождающие функции, их результат — BE будет занесен в расширенное множество ($ABD \cdot ADE = AABDDE = (AA)B(DD)E = BE$). Эффекты расширенного множества порождающих функций не включаются в отчет. Это значит, что нельзя выбирать порождающие функции, которые приведут к появлению фактора в расширенном множестве порождающих функций. Нельзя также использовать порождающую функцию тождества I. При невыполнении какого-либо из этих условий GPSS World известит об ошибке.

Для возврата к исходным порождающим функциям, предложенным GPSS World, нужно нажать кнопку Use Default (По умолчанию).

В рассматриваемом примере имеются только три группы смешивания, поэтому невозможно различить эффекты каждого из пяти факторов. Закройте диалоговое окно Alias Groups (Группы смешивания), нажав кнопку Cancel (Отмена).

В диалоговом окне Screening Experiment Generator (Генератор отсеивающего эксперимента) в группе Fraction (Часть дробного эксперимента) установите Quarter (1/4). Снова откройте диалоговое окно Alias Groups (Группы смешивания) (рис. 9.4).

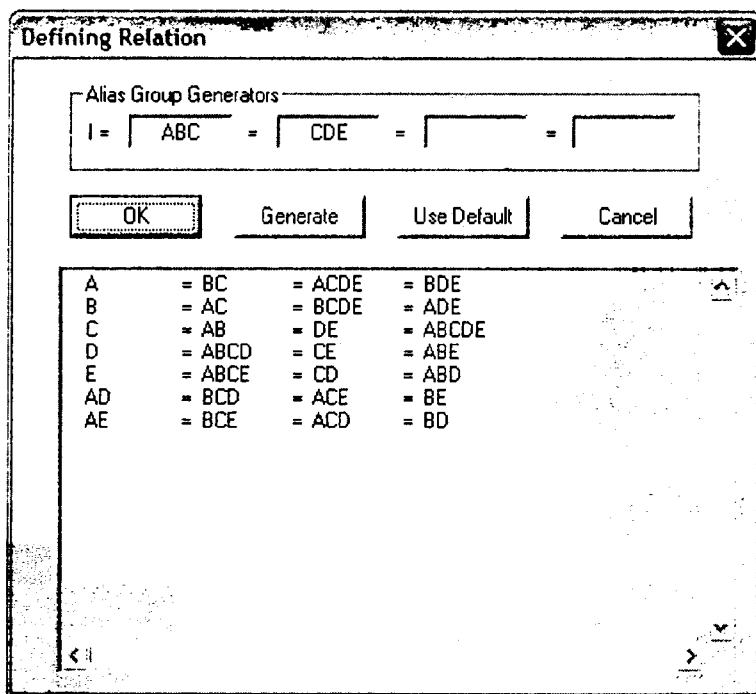


Рис. 9.4. Группы смешивания 1/4

Теперь факторы находятся в отдельных группах. Однако невозможно будет судить об эффектах, т. е. о влиянии на отклик взаимодействий двух факторов. В некоторых случаях этого будет достаточно. Иначе можно попробовать различные порождающие функции для выделения одного или двух двухфакторных взаимодействий. Не делайте этого и закройте диалоговое окно Alias Groups (Группы смешивания), нажав кнопку Cancel (Отмена).

В диалоговом окне Screening Experiment Generator (Генератор отсеивающего эксперимента) в группе Fraction (Часть дробного эксперимента) установите Half (1/2). Снова откройте диалоговое окно Alias Groups (Группы смешивания) (рис. 9.5).

Предложенный GPSS World план эксперимента лучше предыдущих. Вместо 32 нужно будет выполнить 16 наблюдений. В плане выделены все факторы и двухфакторные взаимодействия нескольких факторов. Создайте эксперимент с этим планом, нажав кнопку Ok.

Теперь необходимо создать Plus-операторы и вставить их в нижнюю часть модели Пример 9.1. В левой нижней части диалогового окна Screening Experiment

Generator (Генератор отсеивающего эксперимента) нажмите кнопку Insert Experiment (Вставить эксперимент).

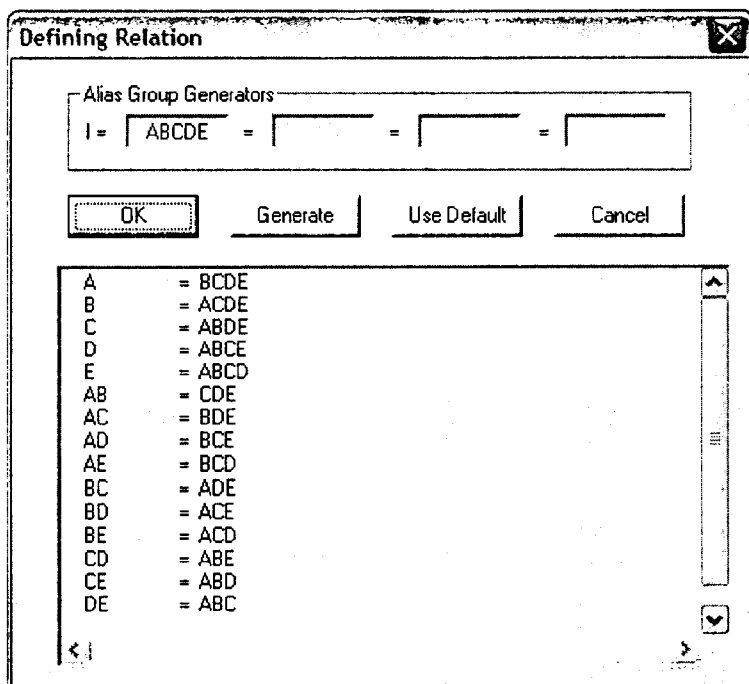


Рис. 9.5. Группы смешивания 1/2

Так как была выбрана опция Generate Run Procedure, то создана стандартная процедура запуска. Появится диалоговое окно (рис. 9.6), дающее возможность пользователю изменить ее согласно своим требованиям.

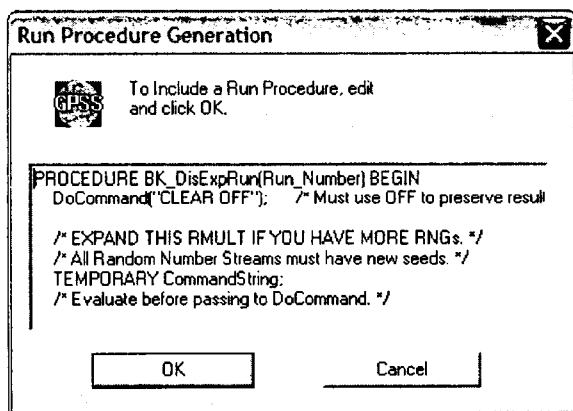


Рис. 9.6. Диалоговое окно стандартной процедуры запуска

Перелистайте стандартную процедуру запуска. В конце ее найдите условия моделирования в каждом наблюдении по умолчанию (рис. 9.7). Откорректируйте их так, как показано на рис. 9.8 (проведите тактическое планирование эксперимента и вы получите требуемое количество прогонов, значительно отличающееся от предложенного).

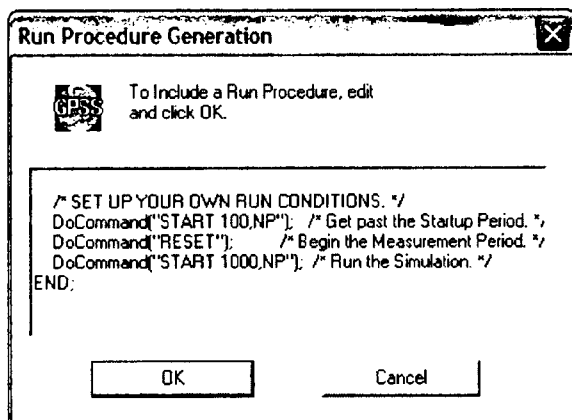


Рис. 9.7. Условия стандартной процедуры запуска по умолчанию

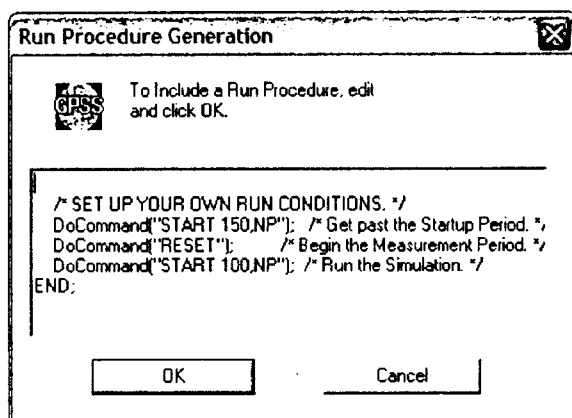


Рис. 9.8. Условия стандартной процедуры запуска после корректировки

Корректировка процедуры запуска возможна до и после того, как она будет добавлена к объекту «Модель».

После корректировки нажмите OK. Сгенерированный Plus-эксперимент представлен ниже. Изучите его. Это необходимо для создания собственных экспериментов. В начале автоматически сгенерированного эксперимента определяется и инициализируется в неопределенное состояние матрица результатов. Далее имеются Plus-операторы, которые для каждого из наблюдений

определяют сочетания уровней факторов. В рассматриваемом примере таких сочетаний шестнадцать ($1/2$ часть ПФЭ).

Plus-эксперимент содержит также вызов Plus-процедуры запуска. Процедура запуска осуществляет связь между генерируемым экспериментом и процессом моделирования. Она вызывается столько раз, сколько требуется сделать наблюдений. Так как процедура запуска вызывается Plus-экспериментом, ей разрешается вызывать библиотечную процедуру DoCommand и, следовательно, выполнять RMULT, CLEAR, RESET и многие другие команды GPSS. Поэтому все команды, необходимые для определения условий наблюдения, следует помещать в процедуру запуска. Для сохранения матрицы результатов при обнулении переменных перед очередным наблюдением используется команда CLEAR OFF. Для изменения начального числа генератора случайных чисел в каждом наблюдении процедуре запуска передается номер запуска.

```
*****
*
*
*                               BK_DisExp
*
*           Дробный факторный отсеивающий эксперимент           *
*
*
*****
BK_DisExp_Results      MATRIX      ,2,2,2,2,2
                        INITIAL      BK_DisExp_Results,UNSPECIFIED
BK_DisExp_NextRunNumber EQU          0

EXPERIMENT  BK_DisExp() BEGIN
            /* Наблюдение 1 */
            L_ = 1;
            T5_ = 3;
            T6_ = 3;
            T7_ = 4;
            Koef = 0.5;
            IF (StringCompare(DataType(BK_DisExp_Results[1,1,1,1,1]),
                                "UNSPECIFIED")'E'0)
            THEN BEGIN
/* Установить начальное значение переменной количества наблюдений */
            BK_DisExp_NextRunNumber = 1;
/* Записать данные наблюдения и запустить процесс моделирования */
            BK_DisExp_GetResult();
            BK_DisExp_Results[1,1,1,1,1] = N$Term1/N$Met0;
            END;
```

```

/* Наблюдение 2 */
L_ = 1;
T5_ = 3;
T6_ = 3;
T7_ = 18;
Kcoef = 2;
IF (StringCompare(DataType(BK_DisExp_Results[1,1,1,2,2]),
"UNSPECIFIED")='E'0)
    THEN BEGIN
/* Записать данные наблюдения и запустить процесс моделирования */
BK_DisExp_GetResult();
BK_DisExp_Results[1,1,1,2,2] = N$Term1/N$Met0;
END;

/* Наблюдения 3 -15 для краткости пропущены*/

/* Наблюдение 16 */
L_ = 5;
T5_ = 15;
T6_ = 20;
T7_ = 18;
Kcoef = 0.5;
IF (StringCompare(DataType(BK_DisExp_Results[2,2,2,2,1]),
"UNSPECIFIED")='E'0)
    THEN BEGIN
/* Записать данные опыта и запустить процесс моделирования */
BK_DisExp_GetResult();
BK_DisExp_Results[2,2,2,2,1] = N$Term1/N$Met0;
END;

/* Эффекты смешивания в дробном факторном эксперименте */
SE_Effects(BK_DisExp_Results,"I=ABCDE");
END;

*****
*                               Процедура запуска наблюдения                               *
*****
PROCEDURE BK_DisExp_GetResult() BEGIN
/* Выполнить указанное в наблюдении число прогонов и записать результаты */
/* Факторы для этого наблюдения уже были определены */
TEMPORARY CurrentYield,ShowString,CommandString;
/* Вызов процедуры запуска */
BK_DisExpRun(BK_DisExp_NextRunNumber);
CurrentYield = N$Term1/N$Met0;
ShowString = PolyCatenate("Run ",String(BK_DisExp_NextRunNumber),". ", "" );

```

```

ShowString = PolyCatenate(ShowString," Yield=",String(CurrentYield),". ");
ShowString = PolyCatenate(ShowString," L_=",String(L_), "; " );
ShowString = PolyCatenate(ShowString," T5_=",String(T5_), "; " );
ShowString = PolyCatenate(ShowString," T6_=",String(T6_), "; " );
ShowString = PolyCatenate(ShowString," T7_=",String(T7_), ";");
ShowString = PolyCatenate(ShowString," Koef=",String(Koef), "; " );
CommandString = PolyCatenate("SHOW """,ShowString,"""", "" );
DoCommand(CommandString);
BK_DisExp_NextRunNumber = BK_DisExp_NextRunNumber + 1;
RETURN CurrentYield;
END;

```

```

*****
*                                     Процедура запуска
*
*****
PROCEDURE BK_DisExpRun(Run_Number) BEGIN
    DoCommand("CLEAR OFF");          /* Использовать OFF для сохранения
результата */
/* Увеличьте количество команд RMULT, если у вас большее количество ГСЧ */
/* Задать новые случайные числа всем потокам случайных чисел */
    TEMPORARY CommandString;
/* Вычислить, прежде чем перейти к DoCommand */
    CommandString = Catenate("RMULT ",Run_Number#111);
/* DoCommand компилирует строку в глобальном контексте */
    DoCommand(CommandString);
/* Установить собственные условия наблюдения */
    DoCommand("START 150,NP");        /* Пройти неустоявшийся режим */
    DoCommand("RESET");               /* Начать период измерений */
    DoCommand("START 100,NP");        /* Провести моделирование */
END;

```

Проведем эксперимент. Для вызова эксперимента предназначена команда CONDUCT (см. п. 3.3.3.2). Однако за функциональной клавишей [F11] была закреплена соответствующая команда CONDUCT (Edit ► Settings ► Function Keys (Правка ► Настройки ► Функциональные клавиши)).

Проведите трансляцию, т. е. создайте объект «Процесс моделирования», для чего нажмите [Ctrl]+[Alt]+[S]. После этого нажмите [F11]. Эксперимент начнет работать. В ходе работы создается отчет, который по готовности записывается в окно Journal (Журнал) созданного объекта «Процесс моделирования» (рис. 9.9).

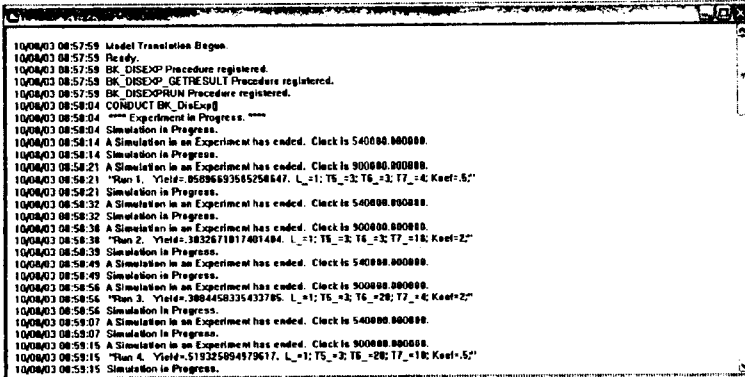


Рис. 9.9. Окно Journal (Журнал) с отчетами по каждому опыту

Так как эксперимент включает 16 наблюдений по 100 прогонов в каждом из них, будет выдано 16 отчетов. После этого окончательные результаты будут выведены в виде таблицы (рис. 9.10).

The screenshot shows the 'JOURNAL' window displaying a table of dispersion analysis results. The table has the following structure:

Alias Group	Effect	Sum of Squares	Degrees of Freedom	F-for Only Main Effects	Critical Value of F (p=.05)
A = BCDE	-0.002	0.027	1	18.424	4.96
B = ACDE	0.162	0.184	1	78.898	4.96
AD = CDE	0.010				
C = ABDE	0.256	0.262	1	178.050	4.96
AC = BDE	0.023				
BC = ADE	-0.028				
DE = ABC	0.012				
D = ABCE	0.167	0.111	1	75.544	4.96
AD = BCE	0.003				
BD = ACE	-0.007				
CE = ABD	0.016				
CD = ABE	-0.038				
BE = ACD	0.013				
AE = BCD	-0.016				
E = ABCD	-0.022	0.002	1	1.201	4.96
Error		0.015	16		
Total		0.522	16		
Grand Mean	0.352				

 Below the table, it says 'Experiment ended.'

Рис. 9.10. Результаты дисперсионного анализа

В таблице каждый фактор и взаимодействие факторов представлены отдельной строкой. В каждой строке для данного эффекта (Effect) имеются результаты основных вычислений и F-статистика. Чем больше значение F-статистики (F-for Only Main Effects), тем сильнее эффект. Эффект, а, следовательно, и фактор, считается значимым, если превышает критическое значение (Critical Value of F (p=.05)), которое также вычисляется GPSS World. В данном примере факторы A, B, C и D являются значимыми, так как их F-статистики больше критического значения, равного 4,96. Наибольший эффект на вероятность потери запросов оказывает фактор C, так как он имеет самую большую статистику (178,050).

Таким образом, по результатам моделирования можно сделать вывод, что при данных потоках и характеристиках процессоров вероятность потерь запросов

в среднем составит 0,352. Для уменьшения потерь запросов нужно продолжить исследование каждого значимого фактора и в первую очередь фактора C .

9.3. Эксперименты пользователя

Эксперименты пользователя являются более гибкими по сравнению с экспериментами, генерируемыми автоматически. Однако их необходимо разрабатывать и запускать самостоятельно. Но и в этом случае GPSS World обеспечивает средства поддержки на каждом шаге их создания. При разработке собственного эксперимента рекомендуется применять стиль программирования, который используют автоматические генераторы. Этот стиль был рассмотрен в примере 9.1. Покажем на следующем примере его применение.

Пример 9.2. На вычислительный комплекс коммутации сообщений (ВККС) поступают сообщения от четырех абонентов и далее передаются по n каналам передачи данных со скоростью 1000 бит/с. Длительности интервалов между сообщениями от каждого абонента распределены по экспоненциальному закону с интенсивностью λ_1 1/с. Сообщения с вероятностью 0,75 и 0,25 могут принадлежать одной из двух категорий: команды (донесения) или неформализованная информация соответственно. Команды (донесения) обладают относительным приоритетом. Длины сообщений 1-й категории равномерно распределены в интервале [1400, 6000] байт. Длины сообщений 2-й категории распределены по нормальному закону с параметрами $m = 2 \cdot 10^3$ байт и $\sigma = 3 \cdot 10^2$. Для хранения сообщений, ожидающих обработки в ВККС, предусмотрен накопитель емкостью L Мбайт.

Каждый канал связи имеет конечную надежность. Выход каналов из строя равновероятен. Интервалы времени между отказами распределены по экспоненциальному закону с интенсивностью $\lambda_2 = 0,5 \cdot 10^{-3}$ 1/с. Длительность восстановления распределена равномерно на интервале [20, 30] с. При отказе канала связи сообщение 1-й категории, которое в нем обслуживалось, возвращается в буфер, если в нем есть место, иначе теряется. Обслуживаемое сообщение 2-й категории в момент отказа канала теряется. В процессе восстановления канала сообщения по нему не передаются.

Разработать имитационную модель с целью исследования в течение 1 ч функционирования ВККС зависимости вероятности передачи сообщений 1-й категории от количества каналов связи и емкости накопителя. Провести двухфакторный дисперсионный анализ результатов моделирования.

Plus-эксперимент пользователя представлен ниже.

; Пример 9.2. ВККС_Exp

```
ВККС_Itoq MATRIX ,4,2,3 ; Определение матрицы результатов
INITIAL ВККС_Itoq,UNSPECIFIED ; Инициализация матрицы результатов
EXPERIMENT ВККС_Exp() BEGIN
TEMPORARY Can0,UrCan,DCan,Emk0,UrEmk,DEmk;
Can0=1; /*Начальное количество каналов связи (фактор 1)*/
```

```

UrCan=4;          /*Количество уровней фактора 1*/
DCan=1;           /*Уровень варьирования фактора 1*/
Emk0=100000;      /*Начальная емкость накопителя (фактор 2)*/
UrEmk=2;          /*Количество уровней фактора 2*/
DEmk=100000;      /*Уровень варьирования фактора 2*/
/* Записать в журнал наименование эксперимента */
DoCommand("SHOW "" """);
DoCommand("SHOW (""*Вычислительный комплекс коммутации сообщений*""");
DoCommand("SHOW (""*****Двухфакторный дисперсионный анализ*****"");
DoCommand("SHOW "" """);
TCan = Can0;      /*Текущему количеству каналов связи -
начальное значение*/
WHILE (TCan<=UrCan) DO BEGIN/*Цикл варьирования фактора 1*/
  Emk=Emk0; /*Текущей емкости накопителя - начальное значение*/
  TUREmk=1;    /*Текущему номеру уровня фактора 2 - начальное значение*/
  WHILE (TUREmk<=UrEmk) DO BEGIN/*Цикл варьирования фактора 2*/
    Replicate=1; /*Текущему номеру наблюдения - начальное значение*/
    WHILE (Replicate<=3) DO BEGIN/*Цикл по числу наблюдений*/
/*Вызов процедуры запуска - выполнение одного наблюдения*/
      BKKCRun(TCan,Emk,Replicate);
/*Запись результата моделирования (наблюдения) в матрицу*/
      BKKC_Itoq[TCan,TUREmk,Replicate]=X$VPerK;
      Replicate=Replicate+1; /*Перейти к следующему наблюдению*/
    END;
    TUREmk=TUREmk+1; /*Перейти к следующему уровню фактора 2*/
    Emk=Emk+DEmk; /*Установить следующее значение фактора 2*/
  END;
  TCan=TCan+DCan; /*Перейти к следующему уровню фактора 1*/
  END;
  ANOVA(BKKC_Itoq,3,2); /*Провести двухфакторный дисперсионный анализ*/
END;
PROCEDURE BKKCRun(NTCan,NEmk,NReplicate) BEGIN
  TEMPORARY RandomN; /*Определение локальной переменной*/
/* Изменить начальное число генератора случайных чисел */
  RandomN = 137 # NReplicate;
  DoCommand("CLEAR OFF"); /* OFF используется для сохранения результата */
/* Установить начальное число генератора случайных чисел*/
  DoCommand(Catenate("RMULT ",RandomN));
  DoCommand(Catenate("NCan EQU ",NTCan)); /*Установить значение фактора 1*/
  DoCommand(Catenate("ENak EQU ",NEmk)); /*Установить значение фактора 2*/
  DoCommand("START 100,NP"); /* Пропустить неустановившийся режим */
  DoCommand("RESET"); /* Начать собирать статистику */
  DoCommand("START 1000,NP"); /* Запустить модель */
END;

```

; Пример 9.2. Модель вычислительного комплекса коммутации сообщений

; Задание исходных данных

```

NCan EQU 5 ; Максимальное количество каналов связи
T1 EQU 7 ; Средний интервал поступления сообщений
T0tk EQU 2000 ; Средний интервал между отказами
T2 EQU 20 ; Среднее время восстановления
T3 EQU 10 ; Среднеквадратическое отклонение времени восстановления
ENak EQU 100000 ; Емкость накопителя, байт
Skor EQU 1000 ; Скорость передачи сообщений, бит/с
DlMin1 EQU 1400 ; Минимальная длина сообщений 1-й категории
DlMax1 EQU 6000 ; Максимальная длина сообщений 1-й категории
SrDl2 EQU 2000 ; Средняя длина сообщений 2-й категории
SkDl2 EQU 300 ; Среднеквадратическое отклонение
; длины сообщений 2-й категории
VrMod EQU 3600 ; Время моделирования, 1 ед. мод. времени = 1 с
Transit TABLE M1,.5,1,20

```

; Описание арифметических выражений вычисления

; *длины сообщения 1-й категории*

```
Dl1 VARIABLE DlMin1+((DlMax1-DlMin1)#(RN1/1000))
```

; *длины сообщения 2-й категории*

```
Dl2 VARIABLE Normal(144,SrDl2,SkDl2)
```

```
WrPer VARIABLE (8#P2)/Skor ; Время передачи сообщения
```

; *вероятностей передачи и потерь сообщений*

```
VPeK VARIABLE N$Ter1/N$Met2; Вероятность передачи команд
```

```
VNfI VARIABLE N$Ter2/N$Met3; сообщений 2-й категории
```

```
VPoK VARIABLE N$Ter3/N$Met2; Вероятность потери команд
```

```
VPNfI VARIABLE N$Ter4/N$Met3; сообщений 2-й категории
```

```
GENERATE (Exponential(11,0,T1))
```

; Сегмент имитации потока команд (донесений), т. е. сообщений 1-й категории

```
TRANSFER .75, Met2, Met3; Категория сообщения
```

```
Met2 ASSIGN 1,1 ; Код 1 - сообщение 1-й категории
```

```
ASSIGN 2,V$Dl1 ; Расчет длины сообщения 1-й категории
```

```
ASSIGN 3,V$WrPer ; Расчет времени передачи
```

; сообщения 1-й категории

```
PRIORITY 10 ; Команде - высокий приоритет
```

```
TRANSFER ,Met1 ; Отправить на ВКК
```

; Сегмент имитации потока сообщений 2-й категории

```
Met3 ASSIGN 1,2 ; Код 2 - сообщение 2-й категории
```

```
ASSIGN 2,V$Dl2 ; Расчет длины сообщения 2-й категории
```

```
ASSIGN 3,V$WrPer ; Расчет времени передачи
```

; сообщения 2-й категории

```
PRIORITY 0 ; Сообщениям 2-й категории - низкий приоритет
```

; Сегмент имитации работы накопителя и каналов связи

```
Met1 TEST LE P2,(ENak-X$TEmk),Met5; Проверка: есть ли место
```

; в накопителе Nak? Если нет, то сообщение теряется

```

    SAVEVALUE   TEmk+,P2    ; Увеличить текущую емкость, байт
Met18 ASSIGN4,0 ; Подготовка к циклу
Met16 ASSIGN    4+,1    ; Начало цикла по определению исправного и
;                    свободного канала
    GATE FV      P4,Met21 ; Исправен ли канал?
    GATE U       P4,Met21 ; Занят ли канал?
    TEST GE      P4,NCan,Met16; Все каналы просмотрены?
    LINK         Spis,PR   ; Помещение сообщения в накопитель
Met21 SEIZE      P4        ; Занять канал связи
    SAVEVALUE   TEmk-,P2    ; Уменьшить текущую емкость, байт
    ADVANCE     P3         ; Задержаться на время передачи сообщения
    RELEASE     P4         ; Освободить канал связи
    UNLINK      Spis,Met18,1; Очередное сообщение на передачу
    TRANSFER    ,Met10
; Сегмент имитации отказов каналов связи
    GENERATE     ...,1
Met19 ADVANCE    (Exponential(237,0,T0tk))
; Розыгрыш номера отказавшего канала связи
    SAVEVALUE    1,(RN317/1000)
    ASSIGN       5,0      ; Начало цикла
Met22 ASSIGN     5+,1     ; Увеличить параметр цикла на 1
    TEST LE      X1,(P5#(1/NCan)),Met22; Если условие выполнено, то
    ASSIGN       4,P5     ; Номер отказавшего канала связи
    GATE FV      P4,Met19 ; Проверка на повторный отказ
; Имитация отказов
    FUNAVAIL     P4,RE,Met20 ; Перевод в недоступность
    ADVANCE      T2,T3     ; Имитация восстановления
    FAVAIL       P4        ; Перевод в доступность
    TRANSFER     ,Met19    ; Отправить для розыгрыша отказа
Met20 RELEASE    P4
    TEST NE      P1,1,Met1 ; Сообщение 1-й категории? Если да, то
    TRANSFER     ,Met5     ; отправить в накопитель
; Сегмент счета переданных и потерянных сообщений
Met10 TEST E     P1,1,Ter2
Ter1 TABULATE    Transit
    TERMINATE    ; Счет переданных сообщений 1-й категории
Ter2 TERMINATE   ; Счет переданных сообщений 2-й категории
Met5 TEST E      P1,1,Ter4
Ter3 TERMINATE   ; Счет потерянных сообщений 1-й категории
Ter4 TERMINATE   ; Счет потерянных сообщений 2-й категории
; Сегмент задания времени моделирования и расчета результатов
    GENERATE     VrMod
    SAVEVALUE    VPek,V$VPek ; Вероятность передачи команд
    SAVEVALUE    VNfI,V$VNfI ; сообщений 2-й категории

```

SAVEVALUE	V\$PoK, V\$VPoK ; Вероятность потери команд
SAVEVALUE	VPnFI, V\$VPnFI; сообщений 2-й категории
TERMINATE	1

Эксперимент ВККС_Exp разработан для заполнения результатами моделирования глобальной матрицы ВККС_Itoq и передаче ее библиотечной процедуре ANOVA (см. п. 8.5). Процедура ANOVA предназначена для выполнения многофакторного дисперсионного анализа, рассматривающего до шести факторов и трехфакторные взаимодействия.

Процедура ANOVA имеет три аргумента. Первый — имя матрицы результатов. Второй аргумент (необязательный) — измерение матрицы результатов для реплик. Каждый уровень этого измерения (*реплика*) представляет наблюдение с отличающимися начальными числами генератора случайных чисел. Информацию с измерениями для реплик ANOVA использует как часть стандартной ошибки и предполагает, что взаимодействие с другими факторами отсутствует. Это обеспечивает оценке большую информативность и большее количество степеней свободы, чем без использования измерений для реплик.

Некоторые планы экспериментов не используют измерение для реплик (например, латинский квадрат). В этом случае второй аргумент процедуры ANOVA указывается равным нулю.

Третий аргумент процедуры ANOVA *управляет добавлением взаимодействий факторов в анализ*. Если аргумент равен единице, взаимодействия в анализ не будут включены. Если аргумент равен двум, в анализ будут включены только двухфакторные взаимодействия. Если аргумент равен трем, в анализ будут включены двух- и трехфакторные взаимодействия. Четырехфакторные взаимодействия GPSS World не поддерживаются.

Элементы матрицы результатов ВККС_Itoq инициализируются в неопределенное состояние (UNSPECIFIED) для того, чтобы процедура ANOVA могла определить, какие наблюдения не были выполнены (наблюдений не было, следовательно, элементы матрицы остались в неопределенном состоянии).

Начальные значения факторов, количество уровней варьирования, значения интервалов варьирования определяются в процедуре эксперимента.

Так как проводится двухфакторный дисперсионный анализ (количество каналов связи — первый (А) фактор, емкость накопителя — второй (В) фактор) и для каждого сочетания уровней факторов выполняется три реплики (Replicate) при различных начальных значениях (RandomN=137#Replicate;) генератора случайных чисел, то процедура Plus-эксперимента имеет три цикла WHILE/DO. Эти циклы выполняют все необходимое для эксперимента. Внутренний цикл организован по количеству реплик, следующий внутренний цикл — по количеству уровней фактора 2, а внешний цикл — по количеству уровней фактора 1.

Во внутреннем цикле по количеству реплик Plus-эксперимента ВККС_Exp первый оператор вызывает процедуру запуска ВККСRun. Процедура запуска со-

держит набор команд для инициализации и выполнения одного наблюдения, состоящего из прогонов, количество которых указано в процедуре

```
DoCommand("START 1000,NP"); /*Запустить процесс моделирования*/
```

Так как процедура BKKCRun вызывается экспериментом, она может вызывать библиотечную процедуру DoCommand. Так она передает процессу моделирования команды CLEAR OFF, RESET и START. Команда CLEAR с OFF используется для того, чтобы глобальная матрица результатов не обнулялась при каждом выполнении этой команды. С помощью процедуры DoCommand процессу моделирования также передаются команды, которые устанавливают соответствующие значения уровней факторов.

Каждый раз при завершении работы процедуры BKKCRun эксперимент BKKC_Exp сохраняет результат (значение X\$VPerK) в соответствующем элементе глобальной матрицы BKKC_Itoq.

Процедура дисперсионного анализа ANOVA рассматривает каждое измерение матрицы результатов как фактор. Размер измерения матрицы результатов должен быть не меньше количества уровней фактора или максимального количества реплик (реплики указываются последним индексом). Количество уровней каждого фактора ограничивается только объемом виртуальной памяти компьютера.

Например:

```
Server_Itoq      MATRIX      ,6,4,8,3
```

В этом примере создается матрица, которая может содержать результаты эксперимента с четырьмя факторами или эксперимента с тремя факторами и репликами. Фактор А может иметь шесть, фактор В — четыре и фактор С — восемь уровней.

Количество матриц результатов может быть любым. Например, если в данной модели нужно было бы исследовать также вероятность передачи сообщений второй категории, то следовало бы ввести и вторую матрицу результатов. Эту матрицу также передавать процедуре ANOVA и получать второй отчет о двухфакторном дисперсионном анализе. Так как в примере проводится двухфакторный дисперсионный анализ, то первый индекс матрицы результатов предназначен для уровней первого фактора, второй индекс — для уровней второго фактора, третий индекс — для индексации реплик. Таким образом, матрица результатов в примере 9.2 имеет три измерения.

Прежде чем проводить эксперимент, закрепите за функциональной клавишей [F10] команду CONDUCT для запуска эксперимента. Откройте объект «Модель» BKKC_Exp. Выберите Edit ► Settings ► Function Keys (Правка ► Настройки ► Функциональные клавиши). В поле F10 введите

```
CONDUCT BKKC_Exp()
```

Нажмите кнопки Применить и Ok. Никакие данные для проведения эксперимента при запуске не передаются. Они указываются в процедуре экспери-

мента, хотя можно было бы их передавать и при запуске эксперимента. Для этого в заголовке процедуры нужно было бы указать соответствующий список.

Теперь создайте процесс моделирования, нажав [Ctrl]+[Alt]+[S]. После этого для запуска эксперимента нажмите функциональную клавишу [F10]. Эксперимент выполняется автоматически, что видно по журналу процесса моделирования.

По завершении эксперимента в журнале процесса моделирования появляется таблица ANOVA (Дисперсионный анализ) (рис. 9.11).

ANOVA						
Source of Variance	Sum of Squares	Degrees of Freedom	Mean Square	F	Critical Value of F ($\alpha=0.05$)	
A	1.841	3	0.614	10498.983	3.24	
B	0.019	1	0.019	328.254	4.48	
AB	0.019	3	0.006	107.718	3.24	
Error	0.001	16	0.000			
Total	1.860	23				

Treatment Level	Count	Mean	Minimum	Maximum	95% C.I. [SE]	
A 1	3	0.261	0.251	0.274	[0.252, 0.270]	
A 2	3	0.302	0.377	0.391	[0.373, 0.391]	
B 1	3	0.724	0.728	0.726	[0.716, 0.733]	
B 2	3	0.826	0.813	0.843	[0.817, 0.835]	
AB 1	3	1.808	0.998	1.083	[0.991, 1.083]	
AB 2	3	1.801	1.004	1.082	[0.992, 1.018]	
	4	1.808	0.999	1.080	[0.990, 1.081]	
	4	0.999	0.998	1.000	[0.990, 1.009]	

Рис. 9.11. Результаты дисперсионного анализа эксперимента ВККС_Exp

В таблице ANOVA:

- Source of Variance — составляющая компонента разброса значений;
- Treatment Level — разброс значений за счет уровней факторов;
- Error — разброс значений внутри уровня факторов;
- Total — общая ошибка;
- Sum of Squares — сумма квадратов ошибок;
- Degrees of Freedom — число степеней свободы;
- Mean Square — средний квадрат;
- F — вычисленное значение критерия.

Нижняя часть таблицы отображает результаты расчета доверительных интервалов вероятности передачи сообщений 1-й категории.

Каждый фактор и взаимодействие в таблице ANOVA представлен отдельной строкой. В каждой строке сначала идет сумма квадратов (Sum of Squares) и количество степеней свободы (Degrees of Freedom), связанные с данной оценкой.

Они являются основными, по которым рассчитываются остальные данные строки. Делением суммы квадратов на количество степеней свободы получается среднеквадратическое (например, для фактора A: $\text{Mean Square} = 1.841/3 = 0.614$). Частное от деления этого среднеквадратического на среднеквадратическую ошибку дает F-статистику для данного эффекта (например, для фактора A: $F = 0.614/0.000... = 10498.983$).

Из результатов дисперсионного анализа видно, что оба фактора являются существенными (значимыми). Существенным является и взаимодействие этих факторов, так как значение F-статистики превышает критическое значение (Critical Value of F). Но все-таки наибольшее влияние на вероятность передачи сообщений 1-й категории оказывает количество каналов связи, так как значение F-статистики здесь наибольшее (10498.983). Видно также, что увеличение каналов связи более трех нецелесообразно. При трех каналах связи увеличение емкости накопителя в два раза также нецелесообразно, так как вероятность передачи сообщений 1-й категории и так ожидается равной 1.

Ранее было отмечено, что матриц результатов может быть столько, сколько необходимо для исследования показателей процесса моделирования. Исследуйте влияние тех же факторов на вероятность передачи сообщений 2-й категории. Добавьте в эксперимент пользователя следующие строки:

```
BKKC_Itoq1 MATRIX      ,4,2,3; Определение матрицы результатов
/*Инициализация матрицы результатов*/
INITIAL      BKKC_Itoq1, UNSPECIFIED
/*Запись результата моделирования (наблюдения) в матрицу*/
BKKC_Itoq1[TCan,TUrEmk,Replicate] = X$VPnfI;
ANOVA(BKKC_Itoq1,3,2); /*Провести двухфакторный дисперсионный анализ*/
```

Проведите эксперимент, создав процесс моделирования и нажав функциональную клавишу [F10]. По окончании эксперимента в журнал процесса моделирования будут выведены две таблицы ANOVA. Первая из них уже была приведена на рис. 9.11, а вторая показана на рис. 9.12.

По результатам второго отчета о дисперсионном анализе можно сделать вывод, что при трех каналах связи и емкости накопителя 100 000 байт вероятность передачи сообщений 2-й категории также ожидается равной 1. Наибольшее влияние на вероятность передачи сообщений 2-й категории оказывает также фактор A (количество каналов связи), но не столь сильное, как на вероятность передачи сообщений 1-й категории. Это следует из сравнения соответствующих F-статистик ($10498.983 > 2173.905$).

ЗАМЕЧАНИЕ

Команды SHOW можно применять более широко, чем показано в примере, для вывода сообщений в журнал процесса моделирования. Это удобный способ наблюдения за ходом выполнения эксперимента. Результаты моделирования можно также выводить в виде текстовых объектов с использованием Plus-процедур Open(), Seek(), Write() и Close(). Эти файлы можно считать другими программами для анализа статистики или просто использовать для записи и накопления экспериментальных данных.

ANOVA						
Source of Variance	Sum of Squares	Degrees of Freedom	Mean Square	F	Critical Value of F ($\alpha=0.05$)	
A	1.987	3	0.662	2173.905	3.24	
B	0.085	1	0.085	15.115	4.49	
AB	0.087	3	0.029	7.728	3.24	
Error	0.085	16	0.005			
Total	2.083	23				

Treatment Level	Count	Mean	Minimum	Maximum	95% C.I. (SE)
A B					
1 1	3	0.313	0.209	0.357	(0.292, 0.334)
1 2	3	0.279	0.248	0.305	(0.258, 0.300)
2 1	3	0.736	0.731	0.741	(0.715, 0.758)
2 2	3	0.655	0.646	0.668	(0.634, 0.676)
3 1	3	0.954	0.931	0.997	(0.973, 1.015)
3 2	3	0.998	0.996	1.001	(0.976, 1.019)
4 1	3	0.997	0.996	0.999	(0.976, 1.019)
4 2	3	0.999	0.998	1.000	(0.977, 1.020)

For the press F1 Experiment ended.

Рис. 9.12. Результаты дисперсионного анализа

9.4. Регрессионный анализ (оптимизирующий эксперимент)

Дисперсионный анализ (отсеивающий эксперимент) показывает силу влияния каждого фактора на наблюдаемую переменную (отклик). Однако оптимизация и количественный прогноз поведения систем часто являются основными задачами моделирования. В GPSS World решение таких задач возможно посредством проведения регрессионного анализа (оптимизирующего эксперимента).

Методика поверхности отклика — это набор статистических приемов для эмпирического построения математической модели и ее исследования. Пользователь задает исходные условия, а GPSS World автоматически создает план и проводит с поверхностью отклика эксперимент, который отыскивает оптимальное значение. В ходе эксперимента GPSS World пытается подобрать либо линейную модель, либо модель второго порядка (включая двухфакторные взаимодействия).

Рассмотрим на примере проведение регрессионного анализа (оптимизирующего эксперимента).

Пример 9.3. Провести регрессионный анализ результатов моделирования вычислительного комплекса коммутации сообщений (Пример 9.2).

Откройте модель Пример 9.2. Сохраните как Пример 9.3 и удалите Plus-операторы эксперимента пользователя. Выберите Edit ▶ Insert Experiment ▶ Optimizing...

(Правка ► Вставить эксперимент ► Оптимизирующий...). Откроется диалоговое окно Optimizing Experiment Generator (Генератор оптимизирующего эксперимента) (рис. 9.13).

Experiment Name: BKKC_OptExp

Run Procedure Name: BKKC_OptExpRun

Initial Local Experimental Region		
Factor Name (User Variable)	Value 1	Value 2
A NCan	1	3
B ENak	100000	150000
C		
D		
E		

Movement Limits	
Low Limit	High Limit
1	4
100000	200000
0	100
0	100
0	100

Result:

Expression: X\$VPerK

☒ Maximize ☐ Minimize

Redirection Limit: 3

☒ Generate Run Procedure ☒ Load F12 with CONDUCT Command

Buttons: Insert Experiment, Cancel, Help

Рис. 9.13. Диалоговое окно Optimizing Experiment Generator (Генератор оптимизирующего эксперимента)

В поля Experiment Name (Имя эксперимента) и Run Procedure Name (Имя процедуры запуска) введите BKKC_OptExp и BKKC_OptExpRun соответственно.

Дальше идет группа полей Initial Local Experimental Region (Задание локальной экспериментальной области). Нужно указать факторы (не более пяти) и для каждого из них два уровня. В примере 9.2 определяется вероятность передачи сообщений 1-й категории. В качестве факторов, влияние которых на вероятность передачи сообщений необходимо исследовать, рекомендуются те же, что и в эксперименте пользователя:

- NCan — количество каналов связи;
- ENak — емкость накопителя.

Введите имена этих факторов и значения уровней, показанных на рис. 9.13. В поле Expression (Выражение) группы Result (Результат) введите имя ячейки, в которой сохраняется результат наблюдения:

X\$VPerK

Оптимизирующий эксперимент нужно ограничить пределами. Группа полей Movement Limits (Пределы перемещения) позволяет ограничить перемещение локальной экспериментальной области. Если это возможно, то необходимо

ввести пределы для каждого фактора. Другой способ ограничить поиск — задание величины Redirection Limit (Предел изменения направления). Это значение устанавливает предел на количество изменений направления движения эксперимента по факторному пространству. Введите в эти поля данные, приведенные на рис. 9.13.

Далее, так как вероятность передачи сообщений должна быть максимальной, оставьте Maximize. Также установите флажки, позволяющие выбрать опции Generate Run Procedure и Load F12 with CONDUCT Command. Назначение этих опций такое же, как и в отсеивающем эксперименте. Только после создания объекта «Процесс моделирования» для запуска эксперимента нужно нажать функциональную клавишу [F12], а не [F11].

Вся необходимая информация введена в соответствующие поля. Нажмите кнопку Insert Experiment (Вставить эксперимент). Так как GPSS World было указано сгенерировать процедуру запуска, то появится диалоговое окно, позволяющее отредактировать ее так же, как и в отсеивающем эксперименте. После корректировки нажмите ОК. Автоматически сгенерированные Plus-операторы появятся в нижней части модели.

Для запуска эксперимента сначала создайте, нажав [Ctrl]+[Alt]+[S], объект «Процесс моделирования», а затем нажмите функциональную клавишу [F12].

Эксперимент начнет выполняться. В окно Journal (Журнал) процесса моделирования будут выводиться отчеты о состоянии и результатах каждого наблюдения. По ним можно узнать о перемещении локальной экспериментальной области. Plus-эксперимент пытается переместить локальную экспериментальную область так, чтобы она включала в себя оптимальные условия. Если это происходит, или если достигнут предел изменения направления или предел перемещения, эксперимент заканчивается.

Когда перемещение останавливается, в окно Journal (Журнал) также выводятся результаты проверки критерия адекватности и уравнение подобранной (найденной) поверхности отклика. Затем, если это еще не сделано, Plus-эксперимент пытается проверить прогнозируемые оптимальные условия, запустив процесс моделирования для них.

Результаты регрессионного анализа (оптимизирующего эксперимента), полученные после выполнения двадцати наблюдений, приведены на рис. 9.14. Заметим, что это на шестнадцать наблюдений больше, чем в эксперименте пользователя.

Полученное уравнение поверхности отклика имеет вид:

$$Y = -1.69374 + 0.200442A + 4.36356e - 005B - \\ - 2.54797e - 007AB - 0.133394A^2 - 1.35857e - 011B^2.$$

Количество каналов связи — 3,5 и емкость накопителя — 102 013 байт включены в оптимальные условия. При этом оптимальное значение вероятности передачи сообщений 1-й категории составит 0.9977. По результатам дисперсионного (п. 9.3) и регрессионного анализов видно, что для обеспечения мак-

симальной вероятности передачи сообщений 1-й и 2-й категорий необходимо иметь три канала связи (3,5 канала не может быть). Емкость накопителя можно увеличить до 105 000 байт или оставить прежним 100 000 байт.

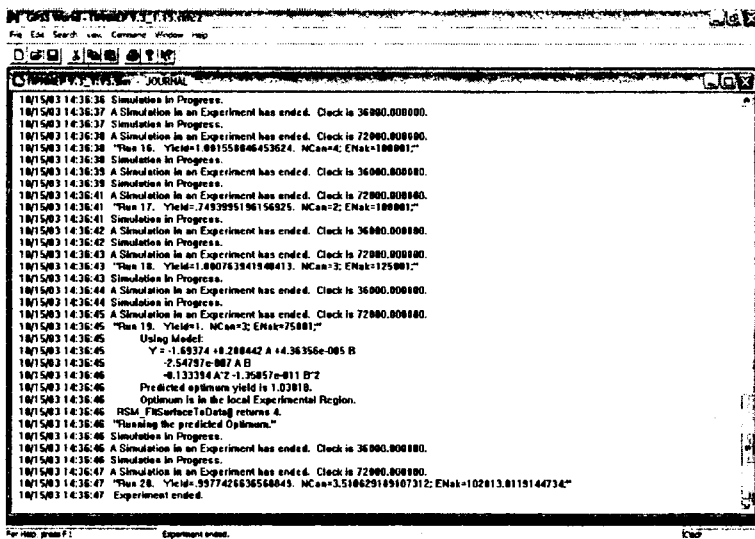


Рис. 9.14. Результаты регрессионного анализа

При проведении регрессионного анализа (оптимизирующего эксперимента) важным является задание локальной экспериментальной области. Значения факторов, указанные исследователем, формируют начальную точку эксперимента. От этого зависит достижение его цели, заключающейся в получении оптимального значения в пределах локальной области, выполнении подтверждающего прогона и выводе математической модели, которая была использована. Следует иметь в виду, что эта цель достигается не всегда.

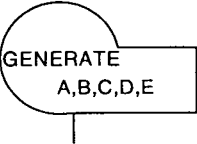
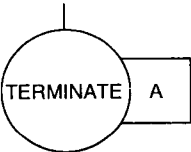
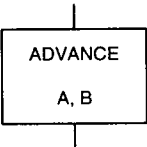
Список литературы

1. Асеев А. А., Боев В. Д., Кулешов И. А., Сеченев Д. М. Основы моделирования систем связи и автоматизации на GPSS/PC: Учеб. пособие. — СПб.: ВУС, 2000. — 230 с.
2. Асеев А. А., Боев В. Д., Кулешов И. А. и др. Основы моделирования систем связи и автоматизации в Delphi 4: Учеб. пособие. — СПб.: ВУС, 2003. — 232 с.
3. Боев В. Д. Имитационное моделирование на GPSS/PC систем обеспечения войск: Учеб. пособие. — СПб.: ВАУ, 1999. — 186 с.
4. Боев В. Д., Сылченко Р. П. Компьютерное моделирование: Руководство по курсовому проектированию. — СПб.: ВУС, 2002. — 96 с.
5. Боев В. Д., Таран С. П. Основы моделирования систем военного назначения в Delphi 4: Учеб. пособие. — СПб.: ВАУ, 2000. — 218 с.
6. Боев В. Д., Филюстин А. Е. Датчик равномерно распределенных случайных чисел: Авт. св. № 1381499, бюл. № 10, 1988.
7. Боев В. Д., Филюстин А. Е. Устройство для определения прогнозных оценок случайного процесса: Авт. св. № 1383406, бюл. № 11, 1988.
8. Боев В. Д., Филюстин А. Е. Датчик равномерно распределенных случайных чисел: Авт. св. № 1594530, бюл. № 35, 1990.
9. Боев В. Д., Филюстин А. Е. Датчик случайных чисел, распределенных по треугольному закону: Авт. св. № 1674116, бюл. № 32, 1992.
10. Боев В. Д., Мартыщенко Л. А., Филюстин А. Е. Генератор случайных чисел: Авт. св. № 1781681, бюл. № 46, 1992.
11. Боев В. Д., Лысенков А. И., Филюстин А. Е. Генератор случайных чисел: Авт. св. № 1798780, бюл. № 8, 1993.
12. Боев В. Д., Филюстин А. Е. Генератор случайных чисел: Авт. св. № 1833868, бюл. № 30, 1993.
13. Бочков А. П., Гасюк Д. П., Филюстин А. Е. Модели и методы управления развитием технических систем: Учеб. пособие. — СПб.: «Союз», 2003. — 288 с. (Высшая школа)
14. Варжапетян А. Г., Глушенко В. В. Системы управления: исследование и компьютерное проектирование: Учеб. пособие. — М.: Вузовская книга, 2000. — 328 с.
15. Гульяев А. К. MATLAB 5.3. Имитационное моделирование в среде Windows: Практическое пособие. — СПб.: КОРОНА принт, 2001. — 400 с.

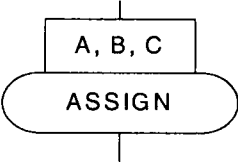
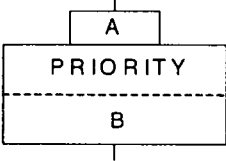
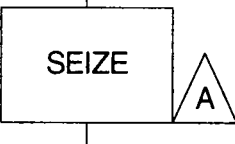
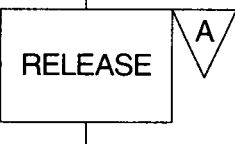
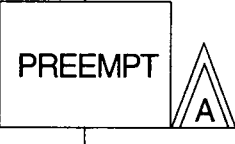
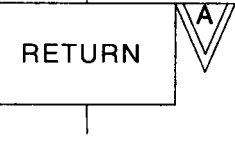
16. Иващенко А. В., Сыпченко Р. П. Основы моделирования сложных систем на ЭВМ: Учебник. — Л.: ЛВВИУС, 1988. — 272 с.
17. Кудрявцев Е. М. GPSS World. Основы имитационного моделирования различных систем. — М.: ДМК Пресс, 2003. — 320 с.
18. Максимей И. В. Имитационное моделирование на ЭВМ. — М.: Радио и связь, 1988.—232 с.
19. Марков А. А., Пирумов Н. Р., Гудзенко Д. Ю. СИМПАС — общецелевая система имитационного моделирования на Паскале. — М.: Наука, 1988. — 52 с.
20. Марков А. А. Моделирование информационно-вычислительных процессов: учебное пособие для вузов. — М.: Изд-во МГТУ им. Н. Э. Баумана, 1999.
21. Методические указания для слушателей ФПКП по моделированию систем и сетей связи на GPSS/PC. Ч. 1. Основы моделирования на GPSS/PC / Сост.: Воробейчиков Л. А., Сосновиков Г. К. — М.: МИС, 1993.
22. Руководство пользователя по GPSS World. /Перевод с английского/. — Казань: Изд-во «Мастер Лайн», 2002. — 384 с.
23. Рыжиков Ю. И. Имитационное моделирование: Теория и технологии. — СПб.: Корона принт; М.: Альтекс — А, 2004. — 384 с.
24. Томашевский В. Н., Жданова Е. Г. Имитационное моделирование в среде GPSS. — М.: Бестселлер, 2003. — 416 с.
25. Советов Б. Я., Яковлев С. А. Моделирование систем: учебник для вузов. — М.: Высшая школа, 2001.
26. Учебное пособие по GPSS World. /Перевод с английского/. — Казань: Изд-во «Мастер Лайн», 2002. — 270 с.
27. Шрайбер Т. Дж. Моделирование на GPSS. — М.: Машиностроение, 1980. —592 с.

Приложение 1

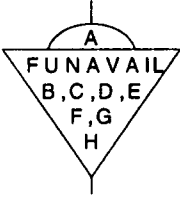
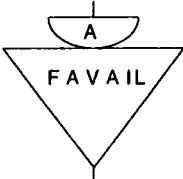
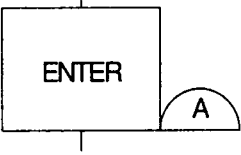
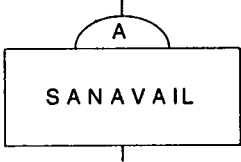
Форматы и графические изображения блоков

Название и формат	Операнды			Графическое изображение
	имя	значение	тип*	
1. Источник потока транзактов Формат: GENERATE A, B, C, D, E	[A]	Среднее время	1...5, 10	
	[B]	Полуинтервал или модификатор-функция	1...5, 10	
	[C]	Время начальной задержки	1...5, 10	
	[D]	Количество генерируемых транзактов	1...5, 8, 10	
	[E]	Приоритет	1, 2, 4, 5, 8, 9	
2. Удаление из модели активного Формат: TERMINATE A	[A]	Величина уменьшения счетчика завершения	1, 2, 5...8	
3. Задержка транзакта на заданный отрезок модельного времени Формат: ADVANCE A, B	A	Среднее время задержки	2...7	
	[B]	Полуинтервал или модификатор-функция	1...7	

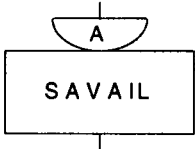
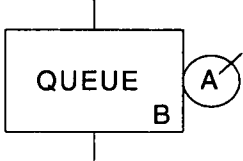
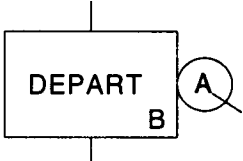
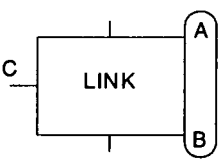
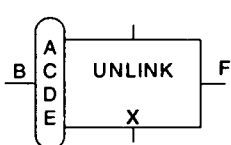
Продолжение прил. 1

Название и формат	Операнды			Графическое изображение
	имя	значение	тип*	
4. Присвоение или изменение значения параметра Формат: ASSIGN A[±], B, C	A	Параметр активного транзакта	2...7	
	B	Значение	1...7	
	[C]	Номер функции	1, 2, 5...8	
5. Установка приоритета активному транзакту Формат: PRIORITY A, B	A	Новое значение приоритета	2, 4...7, 9	
	[B]	Помещение транзакта в CTC	1, BU	
6. Занятие устройства Формат: SEIZE A	A	Имя или номер устройства	2, 5...8	
7. Освобождение занятого устройства Формат: RELEASE A	A	Имя или номер устройства	2, 5...8	
8. Захват устройства Формат: PREEMPT A, B, C, D, E	A	Имя или номер устройства	2, 5...8	
	[B]	Режим	1, PR	
	[C]	Метка блока	1, 2, 5...8	
	[D]	Имя или номер параметра	1, 2, 5...8	
	[E]	Режим удаления	RE	
9. Освобождение устройства от захвата Формат: RETURN A	A		2, 5...8	

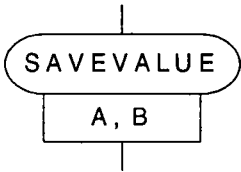
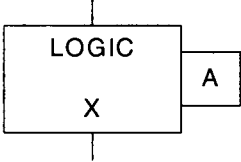
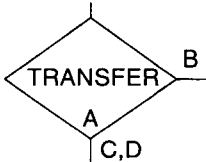
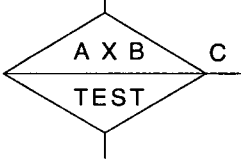
Продолжение прил. 1

Название и формат	Операнды			Графическое изображение
	имя	значение	тип *	
10. Перевод устройства в недоступное состояние Формат: FUNAVAIL A, B, C, D, E, F, G, H	A	Имя или номер устройства	2, 5...8	
	[B]	Режим удаления или продолжения	1, RE, CO	
	[C]	Метка блока выхода	1, 2, 5...8	
	[D]	Имя или параметр устройства	1, 2, 5...8	
	[E]	Режим удаления или продолжения	1, RE, CO	
	[F]	Метка блока выхода	1, 2, 5...8	
	[G]	Режим удаления или продолжения	1, RE, CO	
	[H]	Метка блока выхода	1, 2, 5...8	
11. Перевод устройства в доступное состояние Формат: FAVAIL A	A	Имя или номер устройства	2, 5...8	
12. Занятие памяти Формат: ENTER A, B	A	Имя или номер памяти	2, 5...8	
	[B]	Число занимаемых элементов памяти	1, 2, 5...8	
13. Освобождение памяти Формат: LEAVE A, B	A	Имя или номер памяти	2, 5...8	
	[B]	Число освобождаемых элементов памяти	1, 2, 5...8	
14. Перевод памяти в недоступное состояние Формат: SUNAVAIL A	A	Имя или номер памяти	2, 5...8	

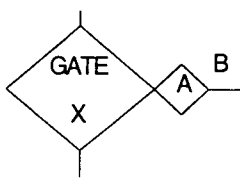
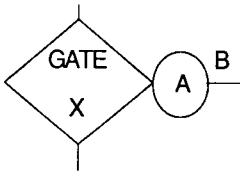
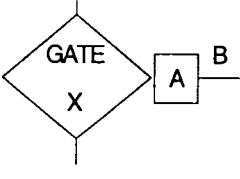
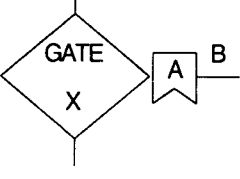
Продолжение прил. 1

Название и формат	Операнды			Графическое изображение
	имя	значение	тип*	
15. Перевод памяти в доступное состояние Формат: SAVAIL A	A	Имя или номер памяти	2, 5...8	
16. Увеличение длины очереди Формат: QUEUE A, B	A	Имя или номер очереди	2, 5...8	
	[B]	Число, на которое увеличивается длина очереди	1, 2, 5...8	
17. Уменьшение длины очереди Формат: DEPART A, B	A	Имя или номер очереди	2, 5...8	
	[B]	Число, на которое уменьшается длина очереди	1, 2, 5...8	
18. Присоединение транзакта к списку пользователя Формат: LINK A, B, C	A	Имя или номер списка пользователя	2, 5...8	
	B	Задание дисциплины обслуживания	5...7, LIFO, FIFO	
	[C]	Метка альтернативного блока	1, 2, 5...8	
19. Удаление транзакта из списка пользователя Формат: UNLINK X A, B, C, D, E	[X]	Условный оператор	1, G, GE, L, LE, E, NE	
	A	Имя или номер списка пользователя	2, 5...8	
	B	Метка блока выхода	2, 5...8	
	[C]	Число удаляемых транзактов	1, 2, 5...8 ALL	
	[D]	Имя или номер параметра	1, 2, 5...8 BACK	
	[E]	Аргумент для сравнения	1...7	
	[F]	Метка альтернативного блока	1, 2, 5...8	

Продолжение прил. 1

Название и формат	Операнды			Графическое изображение
	имя	значение	тип *	
20. Модификация значения ячейки Формат: SAVEVALUEA[±], B	A	Имя или номер ячейки	2, 5...8	
	B	Модификатор	2...7	
21. Изменение состояния логического ключа Формат: LOGIC X A	X	Логический оператор	S, R, I	
	A	Имя или номер логического ключа	2, 5...8	
22. Переход активного транзакта в указанный блок Формат: TRANSFER A, B, C, D	[A]	Режим перехода: безусловный; статистический; параметрический; функциональный; оба; все; выборочный; одновременный; подпрограммный.	1, 2, 5...8 [,] .xxx P FN BOTH ALL PICK SIM SBR	
	[B]	Имя или номер блока (параметра, функции)	1, 2, 5...8	
	[C]	Имя или номер блока (аргумент для режимов P и FN)	1, 2, 5...8	
	[D]	Шаг нумерации блоков для режима ALL	1, 2, 5...8	
23. Управление потоком транзактов в зависимости от результата сравнения операндов Формат: TEST X A, B, C	X	Условный оператор	G, GE, L, LE, N, NE	
	A	Первая сравниваемая величина	2...7	
	B	Вторая сравниваемая величина	2...7	
	[C]	Метка блока	1, 2, 5...8	

Продолжение прил. 1

Название и формат	Операнды			Графическое изображение
	имя	значение	тип *	
24. Управление потоком транзактов в зависимости от состояния устройства Формат: GATE X A, B	X	Проверяемое условие: занято или нет; захвачено или нет; доступно или нет	U, NU, I, NI, FV, FNV	
	A	Имя или номер устройства	2, 5...8	
	[B]	Метка блока	1, 2, 5...8	
25. Управление потоком транзактов в зависимости от состояния памяти Формат: GATE X A, B	X	Проверяемое условие: пусто или нет; заполнено или нет; доступно или нет	SE, SNE, SF, SNF, SV, SNV	
	A	Имя или номер памяти	2, 5...8	
	[B]	Метка блока	1, 2, 5...8	
26. Управление потоком транзактов в зависимости от состояния ключа Формат: GATE X A, B	X	Проверяемое условие: ключ установлен или нет	LS, LR	
	A	Имя или номер логического ключа	2, 5...8	
	[B]	Метка блока	1, 2, 5...8	
27. Управление потоком транзактов в зависимости от состояния синхронизации проверяемого транзакта Формат: GATE X A, B	X	Проверяемое условие: есть транзакт, ожидающий условия синхронизации, в указанном блоке MATCH или нет	M, NM	
	A	Метка блока MATCH	2, 5...8	
	[B]	Метка блока	1, 2, 5...8	

Продолжение прил. 1

Название и формат	Операнды			Графическое изображение
	имя	значение	тип*	
28. Организация циклического прохождения транзактов в модели Формат: LOOP A, B	A	Параметр, содержащий число циклов	2, 5...8	
	[B]	Метка блока — начало цикла	1, 2, 5...8	
29. Создание необходимого числа копий (семейства) активного транзакта Формат: SPLIT A, B, C	A	Количество создаваемых копий	2, 5...8	
	[B]	Метка блока для входа копий	1, 2, 5...8	
	[C]	Имя или номер параметра, в который помещаются порядковые номера копий	1, 2, 5...8	
30. Объединение транзактов одного семейства Формат: ASSEMBLE A	A	Количество собираемых транзактов	2, 5...8	
31. Накопление заданного числа транзактов одного семейства Формат: GATHER A	A	Количество транзактов, которые необходимо накопить	2, 5...8	
32. Синхронизация движения двух транзактов одного семейства Формат: MATCH A	A	Метка сопряженного блока MATCH	2, 5...8	
33. Разрешение транзакту на выполнение операции любого другого блока Формат: EXECUTE A	A	Метка блока, операция которого выполняется активным транзактом	2, 5...8	

Окончание прил. 1

Примечания

В таблице допустимые типы операндов указаны цифрами, которые соответствуют:

- 1 — *Null* (нет ввода);
- 2 — *Name* (имя);
- 3 — *Number* (неотрицательное число);
- 4 — *String* (строковый);
- 5 — *ParenthesizedExpression* (выражение в скобках);
- 6 — *SNA* (СЧА);
- 7 — *SNA*Parameter* (косвенная адресация);
- 8 — *PosInteger* (положительное целое число);
- 9 — *Integer* (целое число со знаком (необязательно));
- 10 — *DirectSNA* (прямой СЧА, т. е. не использующий косвенную адресацию).

Приложение 2

Системные числовые атрибуты GPSS

Код	Значение	Примечание
<i>Системные числовые атрибуты</i>		
RNj	Случайное число. Целочисленное значение.	Датчик генерирует последовательность равномерно распределенных случайных целых чисел от 0 до 999.
CI	Относительное модельное время. Вещественное значение.	Модельное время с момента последней команды RESET. Автоматически изменяется системой.
ACI	Абсолютное модельное время. Вещественное значение.	Модельное время с момента последней команды CLEAR. Автоматически изменяется системой.
TG1	Число, равное текущему значению счетчика завершения. Целочисленное значение.	Начальное значение TG1 задается оператором START и указывает на завершение моделирования, когда становится равным нулю. Вошедшие в блоки TERMINATE с ненулевым положительным операндом A транзакты уменьшают это значение на число, равное значению операнда A.
ZI	Текущая свободная системная память. Целочисленное значение.	
<i>Системные числовые атрибуты транзактов</i>		
Pj	Значение j-го параметра активного транзакта. Целочисленное, вещественное или строковое значение.	Для косвенной адресации используются: СЧА*j ; СЧА*имя параметра; СЧА*\$имя параметра.
XN1	Номер активного транзакта. Целочисленное значение.	
PR	Приоритет активного транзакта. Целочисленное значение.	Может изменяться блоком PRIORITY. По умолчанию приоритет равен 0.
MBj	Соответствие в блоке MATCH. Целочисленное значение.	1, если в сопряженном блоке MATCH находится транзакт того же семейства, что и активный транзакт; 0 — иначе.

Системные числовые атрибуты GPSS (продолжение)

Код	Значение	Примечание
MPj	Время прохождения транзактом некоторого участка модели. Вещественное значение.	Вычисляется как разность текущего абсолютного модельного времени и значения j-го параметра активного транзакта. Разность — транзитное время — сохраняется в j-м параметре.
M1	Время пребывания в модели активного транзакта. Вещественное значение.	Вычисляется как разность абсолютного модельного времени и отметки времени активного транзакта, в которой хранится либо время поступления транзакта в модель, либо время последнего прохождения транзактом блока MARK без операнда A.
<i>Системные числовые атрибуты блоков</i>		
Nj	Общее число входов транзактов в j-й блок. Целочисленное значение.	Увеличивается при каждом входе транзакта в j-й блок.
Wj	Текущее число транзактов, находящихся в j-м блоке. Целочисленное значение.	Увеличивается при каждом входе транзакта в j-й блок и уменьшается при каждом выходе транзакта из j-го блока.
<i>Системные числовые атрибуты MKY (памяти)</i>		
Sj	Текущее содержимое памяти j. Целочисленное значение.	Изменяется блоками ENTER и LEAVE.
Rj	Число свободных единиц памяти j. Целочисленное значение.	Изменяется блоками ENTER и LEAVE.
SRj	Коэффициент использования памяти j. Вещественное значение.	Выражается в тысячных долях, но значение СЧА — целое, т. е. если коэффициент равен 0,65, то SRj равно 650. Может быть нецелочисленным.
SAj	Среднее значение занятой памяти j. Вещественное значение.	Взвешенное по времени среднее количество занятых элементов памяти.
SMj	Максимальное количество элементов занятой памяти j. Целочисленное значение.	Максимальное количество элементов памяти, которое было занято одновременно в процессе моделирования.
SCj	Счетчик использования памяти j. Целочисленное значение.	Общее количество использовавшихся элементов памяти j. Увеличивается при каждом входе транзакта в блок ENTER.
STj	Среднее время использования одного элемента памяти j.	Определяется системой автоматически.
SEj	Занятость памяти j.	1 — не занята; 0 — занята.
SFj	Заполненность памяти j.	1 — заполнена; 0 — не заполнена.
SVj	Доступность памяти j.	1 — доступна; 0 — недоступна.

Системные числовые атрибуты GPSS (продолжение)

Код	Значение	Примечание
<i>Системные числовые атрибуты ОКУ</i>		
Fj	Занятость устройства j. Целочисленное значение.	1 — занято; 0 — не занято. Может быть изменен блоками SEIZE, RELEASE, PREEMPT и RETURN.
Flj	Прерванность устройства j. Целочисленное значение.	1 — прервано; 0 — не прервано. Может быть изменен блоками PREEMPT и RETURN.
FVj	Доступность устройства j. Целочисленное значение.	1 — доступно; 0 — не доступно. Может быть изменен блоками FAVAIL и FUNAVAIL.
FRj	Коэффициент использования устройства j.	Выражается в тысячных долях, но значение СЧА — целое, т. е. если коэффициент равен 0,88, то FRj равен 880.
FCj	Количество транзактов, занимавших устройство j.	Увеличивается при входе транзакта в блок SEIZE или PREEMPT.
FTj	Среднее время использования устройства j одним транзактом.	Определяется системой автоматически.
<i>Системные числовые атрибуты очередей</i>		
Qj	Текущая длина очереди j. Целочисленное значение.	Увеличивается при входе транзакта в блок QUEUE и уменьшается при входе в блок DEPART.
QAj	Средняя длина очереди j. Вещественное значение.	Взвешенное по времени количество транзактов в очереди j.
QMj	Максимальная длина очереди j. Целочисленное значение.	Максимальное количество транзактов, находившихся одновременно в очереди j.
QCj	Общее число входов в очередь j. Целочисленное значение.	Автоматически увеличивается системой при входе транзакта в очередь j.
QZj	Число входов с нулевым временем пребывания в очереди j.	Число входов транзактов, время нахождения которых в очереди равно нулю.
QTj	Среднее время пребывания транзакта в очереди j.	Рассчитывается с учетом нулевых входов.
QXj	Среднее время пребывания транзакта в очереди j.	Рассчитывается без учета нулевых входов.
<i>Системные числовые атрибуты ячеек и матриц ячеек</i>		
Xj	Содержимое ячейки j. Целочисленное, вещественное или строковое значение.	Изменяется блоком SAVEVALUE.
MXj (a, b)	Содержимое ячейки матрицы j. Целочисленное, вещественное или строковое значение.	Значение ячейки в строке a и столбце b. Изменяется блоком SAVEVALUE. Для матриц, имеющих более двух измерений, все остальные индексы считаются равными 1. В отличие от СЧА класса MX, PLUS-выражения могут обращаться к любому элементу многомерных матриц.

Системные числовые атрибуты GPSS (продолжение)

Код	Значение	Примечание
<i>Системные числовые атрибуты вычислительных объектов</i>		
FNj	Результат вычисления значения функции j. Вещественное значение.	
Vj	Результат вычисления целочисленной переменной j или переменной j с плавающей точкой. Вещественное значение.	
BVj	Результат вычисления булевой переменной j. Вещественное значение.	Принимает значения: 1 (true); 0 (false).
<i>Системные числовые атрибуты групп, списков и ключей</i>		
GNj	Текущее число членов в j-й числовой группе.	
GTj	Текущее число членов в j-й группе транзактов.	
CHj	Текущее число транзактов в j-м списке пользователя.	Список пользователя создается и ведется самим программистом в отличие от остальных списков модели. Используются блоки LINK и UNLINK. Значения всех СЧА определяются аналогично значениям СЧА очередей.
CAj	Среднее число транзактов в j-м списке пользователя.	
CMj	Максимальное число транзактов в j-м списке пользователя.	
CCj	Общее число транзактов в j-м списке пользователя.	
CTj	Среднее время пребывания транзакта в j-м списке пользователя.	
LSj	Состояние j-го логического ключа.	Принимает значения: 1 — установлен; 0 — не установлен
<i>Системные числовые атрибуты таблиц</i>		
TBj	Среднее значение невзвешенных аргументов таблицы j.	Для занесения в таблицу используется блок TABULATE.
TCj	Число включений в таблицу j.	Целочисленное значение.
TDj	Среднеквадратическое отклонение для таблицы j.	Вещественное значение.

Приложение 4

Элементы стандартного отчета

Строка заголовка стандартного отчета содержит имя файла модели, который создал отчет. Также в нее включена дата и время прогона модели.

GPSS World Simulation Report - Пример_2.8.24.2
Tuesday, September 02, 2003 11:45:55

1. Общая информация о результатах работы модели

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	43200.000	23	1	0

START TIME — начальное время. Абсолютное модельное время в момент начала моделирования. Устанавливается равным абсолютному модельному времени с помощью оператора RESET или CLEAR;

END TIME — конечное время. Абсолютное модельное время, когда счетчик завершения принимает значение 0;

BLOCKS — количество блоков, использованных в текущей модели, к моменту завершения моделирования;

FACILITIES — количество устройств, использованных в модели, к моменту завершения моделирования;

STORAGES — количество многоканальных устройств, использованных в текущей модели к моменту завершения моделирования.

2. Информация об именах

Файл статистики содержит информацию об именах, которые просматривает GPSS в ходе моделирования.

NAME	VALUE
REM1	10005.000
REMQ	10003.000
REMQ1	10006.000
REMQ2	10004.000
VRREM	10002.000
VRREM1	10000.000
VRREM2	10001.000

NAME — перечень заданных пользователем имен, содержащихся в программе модели;

VALUE — числовое значение, присваиваемое имени. Система начинает отсчет с 10 000;

3. Информация о блоках

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
	1	GENERATE	728	0	0
	2	QUEUE	728	0	0
	3	QUEUE	728	153	0
	4	SEIZE	575	0	0
	5	DEPART	575	0	0
	6	DEPART	575	0	0
	7	ADVANCE	575	0	0
	8	RELEASE	575	0	0
	9	TABULATE	575	0	0
	10	TRANSFER	575	0	0
	11	GENERATE	595	0	0
	12	QUEUE	595	0	0
	13	QUEUE	595	124	0
	14	SEIZE	471	0	0
	15	DEPART	471	0	0
	16	DEPART	471	0	0
	17	ADVANCE	471	1	0
	18	RELEASE	470	0	0
	19	TABULATE	470	0	0
MET1	20	TABULATE	1045	0	0
	21	TERMINATE	1045	0	0
	22	GENERATE	100	0	0
	23	TERMINATE	100	0	0

LABEL — метка, алфавитно-цифровое имя данного блока (если оно задано);

LOC — числовой номер позиции данного блока в модели;

BLOCK TYPE — тип блока GPSS;

ENTRY COUNT — количество транзактов, вошедших в данный блок, с начала работы программы или после последнего выполнения оператора RESET или CLEAR;

CURRENT COUNT — количество транзактов, находящихся в данном блоке к моменту завершения моделирования;

RETRY — количество транзактов, ожидающих специального условия, зависящего от состояния данного блока.

4. Информация об объектах типа «устройство»

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL.	OWNER	PEND	INTER	RETRY	DELAY
REM1	82	0.828	435.987	1	92	0	0	0	0

Элементы статистики, представленные в данном разделе, имеют следующее содержание:

FACILITY — имя или номер устройства;

ENTRIES — количество раз, когда устройство было занято или занято с прерыванием с начала моделирования или после последнего выполнения оператора RESET или CLEAR;

UTIL. — коэффициент использования, доля времени моделирования, в течение которого устройство было занято;

AVE.TIME — среднее время занятия устройства одним транзактом в течение времени моделирования с начала моделирования или после выполнения оператора RESET или CLEAR;

AVAIL. — состояние устройства в конце моделирования (равно 1, если устройство доступно; 0 — если недоступно);

OWNER — номер транзакта, который занимает устройство (0 — устройство не занято);

PEND — количество транзактов, ожидающих выполнения с прерыванием других транзактов (т. е. вошедших в блоки PREEMPT в режиме прерывания);

INTER — количество транзактов, прерванных на данный момент (количество транзактов в списке прерываний);

RETRY — количество транзактов, ожидающих выполнения специального условия, зависящего от состояния данного устройства;

DELAY — количество транзактов, ожидающих занятия устройства (входят также транзакты, ожидающие занятия устройства в режиме прерывания с помощью блоков PREEMPT).

5. Информация об объектах типа «очередь»

QUEUE	MAX	CONT.	ENTRY	ENTRY(0)	AVE.CONT.	AVE.TIME	AVE.(-0)	RETRY
REMQ	2	0	82	28	0.461	242.755	368.628	0
REMQ2	2	0	47	19	0.237	218.158	366.195	0
REMQ1	1	0	35	9	0.223	275.785	371.249	0

QUEUE — имя или номер очереди;

MAX — максимальное содержимое очереди в течение периода моделирования, который начинается с момента трансляции модели или применения оператора RESET или CLEAR;

CONT. — текущее содержимое очереди в конце процесса моделирования;

ENTRY — общее количество входов транзактов в очередь в течение времени моделирования;

ENTRY(0) — общее количество входов транзактов в очередь с нулевым временем ожидания;

AVE.CONT. — среднее значение содержимого очереди в течение времени моделирования;

AVE.TIME — среднее время пребывания одного транзакта в очереди с учетом всех входов в очередь;

AVE.(-0) — среднее время пребывания одного транзакта в очереди без учета «нулевых» входов в очередь;

RETRY — количество транзактов, ожидающих выполнения специального условия, зависящего от состояния очереди.

6. Информация об объектах типа «многоканальное устройство»

STORAGE	CAP.	REM.	MIN.	MAX.	ENTRIES	AVL.	AVE.C.	UTIL.	RETRY	DELAY
PUN1	5	5	0	5	54	1	0.368	0.074	0	0
PUN3	4	4	0	4	74	1	0.103	0.026	0	0

STORAGE — имя или номер памяти;

CAP. — емкость памяти, заданная оператором STORAGE;

REM. — число единиц памяти, свободных в конце моделирования;

MIN. — минимальное число единиц памяти, использовавшихся за период моделирования;

MAX. — максимальное число единиц памяти, использовавшихся за период моделирования;

ENTRIES — количество входов в память за период моделирования;

AVL. — состояние памяти в конце моделирования (1 — доступно; 0 — недоступно);

AVE.C — среднее значение занятой емкости за период моделирования;

UTIL. — коэффициент использования памяти;

RETRY — количество транзактов, ожидающих выполнения специального условия, зависящего от состояния данной памяти;

DELAY — количество транзактов, ожидающих в блоках ENTER, связанных с данной памятью.

7. Информация о таблицах

TABLE	MEAN	STD.DEV.	RANGE	RETRY	FREQUENCY	CUM. %
VRREM1	591.899	271.392		0		
			- 420.000		11369	31.54
			420.000 - 600.000		9972	59.21
			600.000 - 780.000		6765	77.98
			780.000 - 960.000		4418	90.23
			960.000 -		3520	100.00

TABLE — имя или номер таблицы или Q-таблицы;

MEAN — средневзвешенное значение табулируемого аргумента;

STD.DEV. — взвешенное среднеквадратическое отклонение:

$$\text{STD. DEV.} = \text{SQR}((\text{SOS}/(\text{COUNT}-1)) - (\text{SUM2}/(\text{COUNT})(\text{COUNT}-1))),$$

где SOS — накопленная сумма квадратов;

RANGE — нижний и верхний пределы частотного класса:

- при попадании табулируемого аргумента в интервал, который имеет значение больше или меньше нижней границы частотного класса или равное верхней границе, изменяется значение частоты (FREQUENCY);
- операнд В (весовой коэффициент) блока TABULATE может быть использован для определения величины, которая добавляется в частотный класс при попадании табулируемого значения в этот частотный класс;
- частотные классы, суммарное значение которых равно 0, в файл статистики не выводятся;
- RETRY — количество транзактов, ожидающих выполнения специального условия, зависящего от состояния данной таблицы;
- FREQUENCY — суммарная величина, которая формируется при попадании табулируемого аргумента в указанные границы (значения операнда В суммируются блоком TABULATE);
- CUM.% — величина частоты в процентах к общему количеству значений табулируемого аргумента.

8. Информация о списках пользователя

USER CHAIN	SIZE	RETRY	AVE.CONT	ENTRIES	MAX	AVE.TIME
NAK	0	0	0.260	2709	1	3.458

USER CHAIN — имя или номер списка пользователя;

SIZE — количество транзактов в списке пользователя в конце времени моделирования;

RETRY — количество транзактов, ожидающих наступления специального условия, зависящего от состояния данного списка пользователя;

AVE.CONT. — среднее содержимое списка пользователя в течение времени моделирования;

ENTRIES — общее число транзактов, входивших в список пользователя в течение времени моделирования;

MAX — максимальное количество транзактов в списке пользователя за период моделирования;

AVE.TIME — среднее время пребывания транзакта в списке пользователя.

9. Информация о группах транзактов

XACT GROUP	GROUP SIZE	RETRY
USERGR	10	0

XACT GROUP — имя или номер объекта группы транзактов;

GROUP SIZE — число транзактов в группе в конце моделирования;

RETRY — число транзактов, ожидающих выполнения специального условия, зависящего от состояния данной группы транзактов.

10. Информация о числовых группах

NUMERIC GROUP	GROUP SIZE	RETRY
DEPOSITE	1	0

NUMERIC GROUP — имя или номер числовой группы;

GROUP SIZE — количество транзактов, содержащихся в числовой группе в конце моделирования;

RETRY — число транзактов, ожидающих выполнения специального условия, зависящего от состояния данной числовой группы.

11. Информация о логических переключателях

LOGICSWITCH	VALUE	RETRY
SWITCH1	1	0

LOGICSWITCH — имя или номер логического ключа;

VALUE — значение ключа в конце моделирования (1 (true) — «установлен»; 0 (false) — «сброшен»);

RETRY — количество транзактов, ожидающих выполнения специального условия, зависящего от состояния данного логического ключа.

12. Информация о сохраняемых величинах (ячейках)

SAVEVALUE	RETRY	VALUE
KOLVIDMS	0	13.000
VSETRANS	0	8.000

SAVEVALUE — имя или номер ячейки;

VALUE — значение сохраняемой величины в конце моделирования;

RETRY — количество транзактов, ожидающих выполнения специального условия, зависящего от состояния данной ячейки.

13. Информация о матрицах

MATRIX	RETRY	INDICES	VALUE
PLAN	0		
		1 1	52
		1 2	0
		1 3	64

MATRIX — имя или номер матрицы;

RETRY — количество транзактов, ожидающих выполнения специального условия, зависящего от состояния данной матрицы;

INDICES — до шести целых чисел, определяющих элемент матрицы;

VALUE — значение элемента матрицы в конце моделирования (элементы, равные 0, выводятся в отчете группами).

14. Информация о списках текущих и будущих событий

CEC	XN	PRI	M1	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
32		0	3600138.403	1713	0	29		

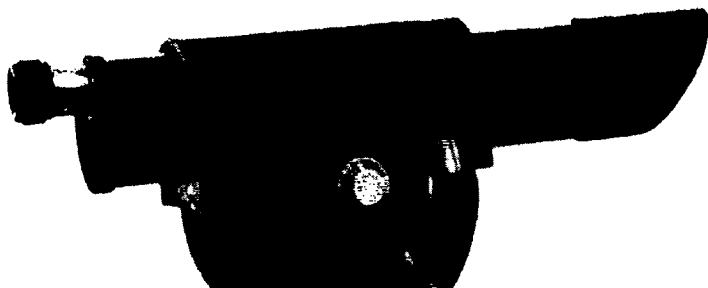
Списки текущих (CEC) и будущих (FEC) событий выводятся в файл статистики, если в команде START значение операнда D равно 1.

Элементы статистики для CEC, представленные в данном разделе, имеют следующее содержание:

- XN — номер каждого транзакта, находящегося в списке текущих событий;
- PRI — приоритет транзакта;
- M1 — время входа транзакта в модель или время его самого раннего предка (породившего данный транзакт);
- ASSEM — номер семейства данного транзакта;
- CURRENT — номер блока, в котором находится транзакт в конце моделирования;
- NEXT — номер следующего блока, в который должен был войти транзакт;
- PARAMETER — имя или номер параметра транзакта;
- VALUE — значение параметра.

В содержании статистики для FEC имеется только одно отличие. Вместо поля M1 присутствует поле BDT, которое определяет момент абсолютного модельного времени, когда транзакт покинет список будущих событий.

FEC	XN	PRI	BDT	ASSEM	CURRENT	NEXT	PARAMETER	VALUE
53		0	3600142.809	1719	0	29		
32		0	3600251.545	1717	0	15		
16		1	3604171.277	1714	52	53	1	1.000



не ищи информацию — подпишись!

что объединяет более 80 000
it-специалистов ведущих компаний?
они подписаны на каталог
softline®-direct!

Как регулярно получать свежую информацию из мира информационных технологий? Как добиться максимальной эффективности инвестиций в ИТ? Как выбрать правильную стратегию в развитии ИТ-инфраструктуры, как выбрать правильное с технологической точки зрения решение той или иной проблемы? Эти вопросы постоянно возникают как перед руководителями бизнеса, так и перед руководителями ИТ-подразделений. Где можно получить объективные ответы на все эти вопросы и узнать, как ИТ используются другими компаниями? Каталог SoftLine®-direct — источник такой информации.

Информация, которая необходима. Правильно построенная ИТ-инфраструктура компании — залог успешного и конкурентноспособного бизнеса. В каталоге представлена детальная информация о последних новинках в мире программного обеспечения, приведены обзорные статьи, примеры успешного внедрения и разнообразные типовые решения. Эта информация поможет Вам сделать правильный выбор.

Выберите каталог, который Вам нужен. В настоящее время выпускается несколько версий каталогов SoftLine®-direct и ряд спецвыпусков (Microsoft, Linux, Novell, Apple). Основной каталог SoftLine®-direct представлен в трех редакциях:

SoftLine®-direct Standard Edition
SoftLine®-direct Professional Edition
SoftLine®-direct Enterprise Edition

Подпишись на каталог сегодня!

Для бесплатной подписки на каталог SoftLine®-Direct позвоните по тел.: +7(095)232-00-23 или посетите сайт www.softline.ru

softline®

программное обеспечение — лицензирование, обучение, консалтинг

+ 7 (095) 232-00-23

www.softline.ru

©2005 SoftLine Inc. Все права защищены. SoftLine, журнал SoftLine являются торговыми марками SoftLine Inc. в ряде стран, включая Россию и другие страны. Другим компаниям и их продуктам даны названия по собственному усмотрению и не являются объектами авторских прав.



Уважаемые господа!

Издательство "БХВ-Петербург" приглашает специалистов в области компьютерных систем и информационных технологий для сотрудничества в качестве авторов книг по компьютерной тематике.

Если Вы знаете и умеете то, что не знают другие,
если у Вас много идей и творческих планов,
если Вам не нравится то, что уже написано...

напишите книгу вместе с "БХВ-Петербург"

Ждем в нашем издательстве как опытных, так и начинающих авторов
и надеемся на плодотворную совместную работу.

С предложениями обращайтесь к главному редактору
Екатерине Кондуковой
Тел.: (812) 251-4244, 251-6501
E-mail: kat@bhv.ru

Россия, 199397, Санкт-Петербург, а/я 194,
www.bhv.ru



www.bhv.ru

Книги издательства "БХВ-Петербург" в продаже:

Магазин "Новая техническая книга": СПб., Измайловский пр., д. 29, тел. (812) 251-41-10

Отдел оптовых поставок: e-mail: opt@bhv.spb.su

Серия "В подлиннике"

Адаменко А., Кучуков А. Логическое программирование и Visual Prolog (+CD-ROM)	992 с.
Андреев А и др. Microsoft Windows XP. Home Edition и Professional. Русские версии	640 с.
Андреев А. и др. Microsoft Windows 2000 Professional. Русская версия	752 с.
Андреев А. и др. Microsoft Windows 2000 Server. Русская версия	960 с.
Андреев А. и др. Microsoft Windows 2000 Server и Professional. Русские версии	1056 с.
Андреев А. и др. Microsoft Windows XP. Руководство администратора	848 с.
Ахаян Р. Macromedia ColdFusion	672 с.
Бурлаков М. Создание видеоклипов	1216 с.
Бурлаков М. CorelDRAW 11	720 с.
Долженков В., Колесников Ю. Microsoft Excel 2002	1072 с.
Дронов В. Macromedia Dreamweaver MX	736 с.
Дронов В. Macromedia Dreamweaver MX 2004	736 с.
Дронов В. Macromedia Flash MX	848 с.
Закер К. Компьютерные сети. Модернизация и поиск неисправностей	1088 с.
Колесниченко О., Шишигин И. Аппаратные средства PC, 4-е изд.	1024 с.
Макдональд М. ASP.NET	992 с.
Матросов А., Сергеев А., Чаунин М. HTML 4.0	672 с.
Михеева В., Харитонов И. Microsoft Access 2002	1040 с.
Нортон Р., Мюллер Дж. Windows 98	592 с.
Ноултон П., Шилдт Г. Java 2	1072 с.
Пауэлл Т. Web-дизайн, 2-е изд.	1072 с.
Персон Р., Роуз К. Microsoft Word 97 в подлиннике	1120 с.
Питц-Моултис Н., Кирк Ч. XML	736 с.
Полещук Н. AutoCAD 2002 (+дискета)	1200 с.
Полещук Н. AutoCAD 2004	976 с.
Полещук Н. AutoCAD 2004. Разработка приложений и адаптация	624 с.
Пономаренко С. Adobe Illustrator 10	688 с.
Пономаренко С. Adobe Photoshop 7	928 с.

Русеев С. WAP: технология и приложения	432 с.
Стахнов А. Linux	912 с.
Стивенс Р. Протоколы TCP/IP. Практическое руководство	672 с.
Сузи Р. Python (+CD-ROM)	768 с.
Тайц А. М., Тайц А. А. Adobe InDesign	704 с.
Тайц А. М., Тайц А. А. Adobe PageMaker 7.0	784 с.
Тайц А. М., Тайц А. А. CorelDRAW Graphics Suite 11: все программы пакета	1200 с.
Тихомиров Ю. Microsoft SQL Server 7.0	720 с.
Фаронов В. Turbo Pascal	1056 с.
Фаронов В. Система программирования Delphi	912 с.
Хомоненко А. и др. Delphi 7	1216 с.
Чекмарев А., Вишневский А., Кокорева О. Microsoft Windows Server 2003. Русская версия	1184 с.

Серия "Мастер"

Microsoft. Visual Basic 6.0	992 с.
Microsoft. Ресурсы Microsoft Windows NT Server 4.0. Книга 1 (+CD-ROM)	408 с.
Microsoft. Электронная коммерция. B2B-программирование (+CD-ROM)	368 с.
Айзекс С. Dynamic HTML (+CD-ROM)	496 с.
Андерсен В., Нортон П. Разработка приложений в Access 97 (+CD-ROM)	656 с.
Анин Б. Защита компьютерной информации	384 с.
Асбари С. Корпоративные решения на базе Linux (+CD-ROM)	496 с.
Березин С. Факсимильная связь в Windows	304 с.
Березин С. Факс-модемы: выбор, подключение, выход в Интернет	256 с.
Борн Г. Реестр Windows 98 (+дискета)	496 с.
Бухвалов А., Бухвалова В., Идельсон А. Финансовые вычисления для профессионалов (+CD-ROM)	320 с.
Валиков А. Технология XSLT	544 с.
Габбасов Ю. Internet 2000	448 с.
Гарбар П. Novell GroupWise 5.5: система электронной почты и коллективной работы	480 с.
Гарнаев А. Excel, VBA, Internet в экономике и финансах	816 с.
Гарнаев А. Visual Basic .NET: разработка приложений	624 с.
Гарригус С. Sound Forge. Музыкальные композиции и эффекты	384 с.
Гордеев О. Программирование звука в Windows (+дискета)	384 с.
Гофман В., Хомоненко А. Работа с базами данных в Delphi, 2-е изд.	624 с.
Дронов В. JavaScript в Web-дизайне	880 с.
Дубина А., Орлова С., Шубина И. MS Excel в электротехнике и электронике (+дискета)	304 с.

Дунаев С. Технологии Интернет-программирования	480 с.
Жарков С. Shareware: профессиональная разработка и продвижение программ	320 с.
Зима В., Молдовян А., Молдовян Н. Безопасность глобальных сетевых технологий	320 с.
Кокорева О. Реестр Windows 2000	640 с.
Кокорева О. Реестр Windows XP	560 с.
Костельцев А. GTK+. Разработка переносимых графических интерфейсов (+CD-ROM)	368 с.
Краснов М. DirectX. Графика в проектах Delphi (+CD-ROM)	416 с.
Краснов М. OpenGL. Графика в проектах Delphi (+дискета)	352 с.
Кубенский А. Создание и обработка структур данных в примерах на Java	336 с.
Куприянов М., Мартынов О., Панфилов Д. Коммуникационные контроллеры фирмы Motorola	560 с.
Лавров С. Программирование. Математические основы, средства, теория	320 с.
Матросов А. Maple 6. Решение задач высшей математики и механики	528 с.
Мешков А., Тихомиров Ю. Visual C++ и MFC, 2-е изд. (+дискета)	1040 с.
Мещеряков М., Хомоненко А. Публикация баз данных в Интернете	560 с.
Миронов Д. Создание Web-страниц в Microsoft Office 2000	320 с.
Новиков Ф. Microsoft Office 2000: разработка приложений	680 с.
Одинцов И. Профессиональное программирование. Системный подход	512 с.
Олифер В., Олифер Н. Новые технологии и оборудование IP-сетей	512 с.
Подольский С., Скиба С., Кожедуб О. Разработка интернет-приложений в Delphi (+дискета)	432 с.
Полещук Н. Visual LISP и секреты адаптации AutoCAD	576 с.
Понамарев В. COM и ActiveX в Delphi	320 с.
Пономаренко С. Adobe InDesign: дизайн и верстка	544 с.
Приписнов Д. Моделирование в 3D Studio MAX 3.0 (+CD-ROM)	352 с.
Роббинс Д. Отладка приложений (+CD-ROM)	512 с.
Русеев Д. Технологии беспроводного доступа. Справочник.	352 с.
Рязанцева Н., Рязанцев Д. 1С:Предприятие. Комплексная конфигурация. Секреты работы	624 с.
Рязанцева Н., Рязанцев Д. 1С:Предприятие. Бухгалтерский учет. Секреты работы	320 с.
Рязанцева Н., Рязанцев Д. 1С:Предприятие. Зарплата и кадры. Секреты работы	304 с.
Рязанцева Н., Рязанцев Д. 1С:Предприятие. Торговля и склад. Секреты работы	368 с.
Соколенко П. Программирование SVGA-графики для IBM PC	432 с.
Сысойкина М. Borland Kylix: разработка приложений (+CD-ROM)	544 с.

Тихомиров Ю. SQL Server 7.0: разработка приложений	370 с.
Трасковский А. Секреты BIOS	400 с.
Тяпичев Г. Компьютер на любительской радиостанции (+дискета)	448 с.
Хабибуллин И. Создание распределенных приложений на Java 2	704 с.
Шапошников И. Web-сервисы Microsoft .NET	336 с.
Шапошников И. Интернет-программирование	224 с.
Шапошников И. Интернет-программирование, 2-е изд.	224 с.
Шапошников И. Справочник Web-мастера. XML	304 с.
Шилдт Г. Теория и практика C++	416 с.
Шинкарев О. SOFTIMAGE XSI: от моделирования до анимации (+CD-ROM)	432 с.
Яцук О. Компьютерные технологии в дизайне. Логотипы, упаковка, буклеты (+CD-ROM)	464 с.
Яцук О., Романычева Э. Компьютерные технологии в дизайне. Эффективная реклама (+CD-ROM)	480 с.

Серия Мастер Медиа

Делла-Росса Р., Делла-Росса А. Adobe Photoshop для художников	592 с.
Кулагин Б. 3ds max 5: от фантазии к реальности	480 с.
Медведев Е., Трусова В. Реальность виртуального звука (+CD-ROM)	496 с.
Медведев Е., Трусова В. Cubase SX для музыкантов (+CD-ROM)	640 с.
Медведев Е., Трусова В. Steinberg Nuendo 2. Секреты виртуального звука (+ прил. на CD)	432 с.
Петелин Р., Петелин Ю. Cubase SX. Секреты мастерства (+CD-ROM)	640 с.
Петелин Р., Петелин Ю. Профессиональные плагины для Cubase и Sonar (+CD-ROM)	592 с.
Погорелов В. AutoCAD: трехмерное моделирование и дизайн	288 с.
Пономаренко С. Пиксел и вектор. Принципы цифровой графики	496 с.
Файола Э. Шрифты для печати и WEB-дизайна	288 с.
Цоллер С. Создание музыки на ПК: от простого к сложному	320 с.

Серия "Мастер программ"

Боровский А. C++ и Pascal в Kylix 3. Разработка Интернет-приложений и СУБД	544 с.
Будилев В. Интернет-программирование на Java	704 с.
Вершинин М., Иванова Е. Java 2, Enterprise Edition. Технологии проектирования и разработки	1088 с.
Гарнаев А. Microsoft Excel 2002: разработка приложений	768 с.
Дарахвелидзе П., Марков Е. Программирование в Delphi 7 (+ дискета)	784 с.
Дарахвелидзе П., Марков Е. Разработка Web-служб средствами Delphi (+дискета)	672 с.

Касперски К. Техника оптимизации программ. Эффективное использование памяти	464 с.
Магда Ю. Ассемблер. Разработка и оптимизация Windows-приложений (+CD-ROM)	544 с.
Матросов А и др. MS Office XP: разработка приложений	944 с.
Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем	400 с.
Несвижский В. Программирование устройств SCSI и IDE	592 с.
Пирогов В. Программирование на Visual C++ .NET	800 с.
Пирогов В. Ассемблер для Windows, 2-е изд.	656 с.
Поляков А., Брусенцев В. Методы и алгоритмы компьютерной графики в примерах на Visual C++, 2-е изд. (+CD-ROM)	560 с.
Постолит А. Visual Studio. NET: разработка приложений баз данных	544 с.
Рафалович В., Карнаухов С. Программирование для карманных компьютеров Palm и Pocket PC	352 с.
Рохилла С. и др. Microsoft ADO.NET: разработка профессиональных проектов	768 с.
Секунов Н. Программирование на C++ в Linux	368 с.
Старыгин А. XML: разработка Web-приложений (+CD-ROM)	592 с.
Тихомиров Ю. OpenGL. Программирование трехмерной графики, 2-е изд. (+ дискета)	304 с.
Хабибуллин И. Разработка Web-служб средствами Java	400 с.
Чакраборти А. и др. Microsoft.NET Framework : разработка профессиональных проектов	896 с.
Чекалов А. Базы данных: от проектирования до разработки приложений (+дискета)	384 с.
Шорт С. Разработка XML Web-сервисов средствами Microsoft .NET (+CD-ROM)	480 с.

Серия "Мастер решений"

Агапонов С. и др Средства дистанционного обучения. Методика, технология, инструментарий (+CD-ROM)	336 с.
Елисеев Д. Аппаратно-программные средства карманных компьютеров (+CD-ROM)	368 с.
Касперски К. ПК: решение проблем	560 с.
Кетков Ю. Matlab 6.x: программирование численных методов	672 с.
Куперштейн В. Microsoft Project в делопроизводстве и управлении	480 с.
Леоненков А. Нечеткое моделирование в среде MATLAB и fuzzyTECH	736 с.
Маклаков С., Матвеев Д. Анализ данных. Генератор отчетов Crystal Reports	496 с.
Масленников М. Практическая криптография (+CD-ROM)	464 с.

Рудометов В., Рудометов Е. PC: настройка, оптимизация и разгон, 3-е изд.	496 с.
Рудометов В., Рудометов Е. Форсирование аппаратных средств PC	542 с.
Салманов О. Математическая экономика с применением Mathcad и Excel	464 с.

Серия "Мастер систем"

Галичский К. Компьютерные системы в телефонии	400 с.
Гаскин Дж. Администрирование Novell Netware 6/6.5 (+CD-ROM)	1056 с.
Зима В., Молдовян А., Молдовян Н. Безопасность глобальных сетевых технологий, 2-е изд.	368 с.
Кокорева О. Реестр Microsoft Windows Server 2003	640 с.
Конеев И., Беляев А. Информационная безопасность предприятия	752 с.
Лукацкий А. Обнаружение атак, 2-е изд.	608 с.
Попов А. Windows ScriptHost для Windows 2000/XP (+дискета)	640 с.
Федорчук А., Торн А. FreeBSD: установка, настройка, использование	640 с.
Филимонов А. Протоколы Интернета	528 с.

Серия "Самоучитель"

Ананьев А., Федоров А. Самоучитель Visual Basic 6.0	624 с.
Ануфриев И. Самоучитель MatLab 5.3/6.x (+дискета)	736 с.
Бекаревич Ю., Пушкина Н. Самоучитель Microsoft Access 2002	720 с.
Будилов В. Основы программирования для Интернета	736 с.
Бурлаков М. Самоучитель Macromedia Flash MX	656 с.
Гаевский А. Основы работы в Интернете	464 с.
Гарнаев А. Самоучитель Visual Studio .NET 2003	688 с.
Гарнаев А. Самоучитель VBA	512 с.
Деревских В. Синтез и обработка звука на PC	352 с.
Дмитриева М. Самоучитель JavaScript	512 с.
Долженков В., Колесников Ю. Самоучитель Excel 2000 (+дискета)	368 с.
Дунаев В., Дунаев В. Графика для Web	640 с.
Жаринов К. Основы веб-мастеринга	352 с.
Жуков А., Авдюхин А. Ассемблер. Самоучитель (+дискета)	448 с.
Исагулиев К. Самоучитель Macromedia Flash 5	368 с.
Кетков Ю., Кетков А. Практика программирования: Visual Basic, C++ Builder, Delphi (+дискета)	464 с.
Кетков Ю., Кетков А. Практика программирования: Бейсик, Си, Паскаль (+дискета)	480 с.
Кириянов Д. Самоучитель Mathcad 11	560 с.
Кириянов Д., Кириянова Е. Самоучитель Adobe Premiere 6.5	480 с.
Клюквин А. Краткий самоучитель работы на ПК	432 с.

Комолова Н. Компьютерная верстка и дизайн	512 с.
Коркин И. Самоучитель Microsoft Internet Explorer 6.0	288 с.
Костромин В. Самоучитель Linux для пользователя	672 с.
Котеров Д. Самоучитель PHP 4	576 с.
Кузютина А., Шапошников И. Самоучитель Adobe GoLive 6	352 с.
Культин Н. Основы программирования в Delphi 7 (+дискета)	608 с.
Культин Н. Программирование в TurboPascal 7 и Delphi, 2-е изд. (+дискета)	416 с.
Культин Н. C++ Builder (+прил. на CD-ROM)	320 с.
Леоненков А. Самоучитель UML	304 с.
Матросов А., Чаунин М. Самоучитель Perl	432 с.
Медников В. Основы компьютерной музыки	336 с.
Мур М. и др. Телекоммуникации. Руководство для начинающих	624 с.
Надеждин Н. Цифровая фотография. Практическое руководство	368 с.
Немнюгин С. Современный Фортран	496 с.
Омельченко Л., Федоров А. Самоучитель Microsoft Windows XP	560 с.
Омельченко Л., Федоров А. Самоучитель Windows 2000 Professional	528 с.
Омельченко Л. Самоучитель Visual FoxPro 8	688 с.
Пекарев Л. Самоучитель 3ds max 5	336 с.
Полещук Н. Самоучитель AutoCAD 2002	608 с.
Полещук Н., Савельева В. Самоучитель AutoCAD 2004	640 с.
Поляк-Брагинский А. Сеть своими руками	320 с.
Понамарев В. Самоучитель Delphi 7 Studio	512 с.
Понамарев В. Самоучитель JBuilder 6/7	304 с.
Понамарев В. Самоучитель KYLIX	416 с.
Секунов Н. Самоучитель C#	576 с.
Секунов Н. Самоучитель Visual C++ .NET (+дискета)	738 с.
Секунов Н. Самоучитель Visual C++ 6 (+дискета)	960 с.
Сироткин С., Чалышев И., Воробьев С. Самоучитель WML и WMLScript	240 с.
Тайц А. М., Тайц А. А. Самоучитель Adobe Photoshop 7 (+дискета)	688 с.
Тайц А. М., Тайц А. А. Самоучитель CorelDRAW 11	704 с.
Тихомиров Ю. Самоучитель MFC (+дискета)	640 с.
Токарев С. Самоучитель Macromedia Dreamweaver MX	544 с.
Токарев С. Самоучитель Macromedia Fireworks	448 с.
Трасковский А. Устройство, модернизация, ремонт IBM PC	608 с.
Трусова В., Медведев Е. Музыкальная азбука на PC (+дискета)	496 с.
Федорова А. Самоучитель Adobe PageMaker 7	736 с.
Хабибуллин И. Самоучитель Java	464 с.
Хабибуллин И. Самоучитель XML	336 с.

Хомоненко А. Самоучитель Microsoft Word 2000	688 с.
Хомоненко А. Самоучитель Microsoft Word 2002	624 с.
Хомоненко А., Гофман В. Самоучитель Delphi	576 с.
Шапошников И. Самоучитель HTML 4	288 с.
Шапошников И. Интернет. Быстрый старт	272 с.
Шапошников И. Самоучитель ASP.NET	368 с.
Шилдт Г. Самоучитель C++, 3-е изд. (+дискета)	688 с.

Серия "Учебное пособие"

Бенькович Е., Колесов Ю., Сениченков Ю. Практическое моделирование динамических систем (+CD-ROM)	464 с.
Вуль В. Электронные издания (+дискета)	560 с.
Гомоюнов К. Транзисторные цепи	240 с.
Грушвицкий Р., Мурсаев А., Угрюмов Е. Проектирование систем на микросхемах программируемой логики	608 с.
Дорот В., Новиков Ф. Толковый словарь современной компьютерной лексики, 2-е изд.	512 с.
Ирвин Дж., Харль Д. Передача данных в сетях: инженерный подход	448 с.
Иртегов Д. Введение в операционные системы	624 с.
Культин Н. C/C++ в задачах и примерах	288 с.
Культин Н. Turbo Pascal в задачах и примерах	256 с.
Никулин Е. Компьютерная геометрия и алгоритмы машинной графики	560 с.
Порев В. Компьютерная графика	432 с.
Поршев С. Вычислительная математика. Курс лекций	320 с.
Рапаков Г., Ржеуцкая С. Программирование на языке Pascal	480 с.
Сафронов И. Бейсик в задачах и примерах	224 с.
Сеннов А. Курс практической работы на ПК	576 с.
Солонина А., Улахович Д., Яковлев Л. Алгоритмы и процессоры цифровой обработки сигналов	464 с.
Солонина А. Основы цифровой обработки сигналов. Курс лекций	576 с.
Солонина А., Улахович Д., Яковлев Л. Цифровые процессоры обработки сигналов фирмы MOTOROLA	512 с.
Сорокина С., Тихонов А., Щербаков А. Программирование драйверов и систем безопасности	256 с.
Стешенко В. P-CAD. Технология проектирования печатных плат	720 с.
Суворова Е., Шейнин Ю. Проектирование цифровых систем на VHDL	576 с.
Угрюмов Е. Цифровая схемотехника	528 с.
Ускова О. и др. Программирование алгоритмов обработки данных	192 с.

Частиков А., Гаврилова Т., Белов Д. Разработка экспертных систем. Среда CLIPS	608 с.
Чемоданова Т. Pro/Engineer: деталь, сборка, чертеж	560 с.
Черняк А. и др. Математика для экономистов на базе Mathcad	496 с.
Хрящев В., Шипова Г. Моделирование и создание чертежей в системе AutoCAD	224 с.
Шелест В. Программирование	592 с.

Серия "Быстрый старт"

Васильева В. Персональный компьютер. Быстрый старт	480 с.
Гофман В., Хомоненко А. Delphi. Быстрый старт	288 с.
Дмитриева М. JavaScript. Быстрый старт	334 с.
Культин Н. Microsoft Excel. Быстрый старт	208 с.
Культин Н. Microsoft Word. Быстрый старт	176 с.
Хомоненко А., Гридин В. Microsoft Access. Быстрый старт	304 с.

Серия "Экспресс-курс"

Васильева В. Обслуживание ПК своими руками. Экспресс-курс	320 с.
Дронов В. Macromedia Flash MX. Экспресс-курс	352 с.
Омельченко Л., Федоров А. Microsoft Windows 98/ME/XP. Экспресс-курс	352 с.
Петюшкин А. HTML. Экспресс-курс	258 с.
Погорелов В. AutoCAD. Экспресс-курс	352 с.
Поляк-Брагинский А. Сеть под Windows. Экспресс-курс	336 с.
Понамарев В. Visual Basic. Net. Экспресс-курс	304 с.
Розенталь М. Как собрать свой компьютер, 3-е изд.	256 с.

Серия "Научное издание"

Воеводин В., Воеводин В. Параллельные вычисления	608 с.
Касьянов В., Евстигнеев В. Графы в программировании: обработка, визуализация и применение	1104 с.
Молдовян А. и др. Криптография: скоростные шифры	496 с.

Магазин-салон “НОВАЯ ТЕХНИЧЕСКАЯ КНИГА”

190005, Санкт-Петербург, Измайловский пр., 29

В магазине представлена литература по
компьютерным технологиям
радиотехнике и электронике
физике и математике
экономике
медицине
и др.

Низкие цены
Прямые поставки от издательств
Ежедневное пополнение ассортимента
Подарки и скидки покупателям

Магазин работает с 10.00 до 20.00
без обеденного перерыва
выходной день – воскресенье

Тел.: (812)251-41-10, e-mail: trade@techkniga.com