O‘ZBEKISTON RESPUBLIKASI
OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

NIZOMIY NOMIDAGI TOSHKENT DAVLAT
PEDAGOGIKA UNIVERSITETI

M.O‘. ASHUROV, SH.A.SATTAROVA,
SH.U.USMONQULOV

ALGORITMLAR

O‘zbekiston Respublikasi Oliy va o‘rta maxsus ta’lim vazirligi tomonidan 5110700 – Informatika o‘qitish metodikasi ta’limi yo‘nalishi uchun o‘quv qo‘llanma sifatida tavsiya etilgan

TOSHKENT – 2018

UO‘K: 004.421(075.8)
KBK 32.965
A 94

A 94 **M.O‘.Ashurov, Sh.A.Sattarova**, **Sh.U.Usmonqulov**.
Algoritmlar. –T.: «Fan va texnologiya», 2018, 244 bet.

ISBN 978–9943–11–852–2

Ushbu o‘quv qo‘llanmada algoritm tushunchasi va uning xossalari, algoritm ijrochilari, algoritmlarni tasvirlash usullari, algoritmlarni ishlab chiqish metodlari, algoritmlar tahlili, saralash, qidiruv usullari, graflar bilan ishlovchi algoritmlar imkoniyatlari yoritib berilgan. O‘quv qo‘llanma bakalavriat yo‘nalishi: 5110700 – Informatika o‘qitish metodikasi ta’lim yo‘nalishida tahsil olayotgan talabalarning o‘zlashtirishi lozim bo‘lgan bilimlarni o‘z ichiga olgan.

UO‘K: 004.421(075.8)
KBK 32.965

Mas’ul muharrir:
A.A.Abduqodirov – p.f.d., professor

Taqrizchilar:
F.M.Zakirova **–** Muhammad Al-Xorazmiy nomidagi TATU huzuridagi PXQT va UMO tarmoq markazi direktori, p.f.d., professor;
M.E.Mamarajabov **–** Nizomiy nomidagi TDPUning “Informatika o‘qitish metodikasi” kafedrasi dotsenti, pedagogika fanlari nomzodi.

ISBN 978–9943–11–852–2

KIRISH

Mazkur o‘quv qo‘llanma bakalavriat yo‘nalishi: 5110700 – Informatika o‘qitish metodikasi ta’lim yo‘nalishida tahsil olayotgan talabalarning o‘zlashtirishi lozim bo‘lgan bilimlari asosida tuzilgan bo‘lib, bo‘lajak fan o‘qituvchisi egallashi kerak bo‘lgan bilimlar va ko‘nikmalar mazmunini o‘z ichiga oladi: algoritm va uning xossalari, algoritm ijrochilari, algoritmlarni tasvirlash usullari, rekursiya, algoritmning murakkabligi tushunchasi, algoritm turlari, samarali algoritmlar ishlab chiqishning asosiy usullari, algoritmik tillar; saralash algoritmlari, qo‘shib saralash, almashish usulida saralash, saralashning Sheyker, Shella usullari, piramida usulida saralash, turnir usulida saralash va ulardan foydalanish usullari haqida tasavvurlar hosil qilish; qidiruv usullari: binar qidiruv, Fibonachchi qidiruv, binar daraxt bo‘yicha qidiruv, muvozanatlash-tirilgan daraxt bo‘yicha qidiruv; Rabin-Karp algoritmi, rekursiv algoritmlar bilan tanishish; paskal dasturlash tili, dasturlash tilining alifbosi, buyruqlar tizimi va operatorlari, kattaliklar va ularning tiplari, massivlar, chiziqli, tarmoqlanuvchi va takrorlanuvchi operatorlar, funksiya va protseduralar, fayllar bilan ishlash, tilning grafik imkoniyatlari buyruqlar tizimi va operatorlarini, chiziqli, tarmoqlanuvchi va takrorlanuvchi dasturlar tuzish kabilarni.

Algoritmlar fanini o‘qitishdan maqsad – informatika o‘qituv-chisining kasbiy sohasida egallashi lozim bo‘lgan bilimlar va amal-da qo‘llash uchun ko‘nikma va malakalarni shakllantirish va rivoj-lantirishdan iborat. Ushbu dasturda har bir kasb egasi uning faoliyat ko‘rsatish turidan qat’i nazar egallashi kerak bo‘lgan tayanch nazariy va amaliy ma’lumotlarni o‘z ichiga oladi.

Algoritmlar fanining vazifasi: algoritm tushunchasi va uning xossalari, algoritm ijrochilari, algoritmlarni tasvirlash usullari, rekursiya va iteratsiya, algoritmning murakkabligi tushunchasi, algoritm turlari, samarali algoritmlar ishlab chiqishning asosiy usul-lari, algoritmik tillar bilimlari bilan tanishtirish; algoritmik tillarning asosiy tushunchalari: steklar, navbatlar, daraxtlar, algoritmlar tahlili

kabilar haqida ma’lumotlar berish; saralash algoritmlari, qo‘shib saralash, almashish usulida saralash, saralashning Sheyker, Shella usullari, piramida usulida saralash, turnir usulida saralash va ulardan foydalanish usullari haqida tasavvurlar hosil qilish; qidiruv usullari: binar qidiruv, Fibonachchi qidiruv, binar daraxt bo‘yicha qidiruv, muvozanatlashtirilgan daraxt bo‘yicha qidiruv; Rabin-Karp algoritmi, rekursiv algoritmlar bilan tanishish; paskal dasturlash tili, dasturlash tilining alifbosi, buyruqlar tizimi va operatorlari, kattaliklar va ularning tiplari, massivlar, chiziqli, tarmoqlanuvchi va takrorlanuvchi operatorlar, funksiya va protseduralar, fayllar bilan ishlash, tilning grafik imkoniyatlari haqidagi ma’lumotlarga ega bo‘lishdan iborat.

“Algoritmlar” o‘quv fanini o‘zlashtirish jarayonida amalga oshiriladigan masalalar doirasida talaba:

– algoritm va uning xossalari, algoritmik tillar, qidiruv usullari: binar qidiruv, Fibonachchi qidiruv, binar daraxt bo‘yicha qidiruv, muvozanatlashtirilgan daraxt bo‘yicha qidiruv; Rabin-Karp algoritmi, rekursiv algoritmlar, saralash algoritmlari, qo‘shib saralash, almashish usulida saralash, saralashning Sheyker, Shella usullari, piramida usulida saralash, turnir usulida saralashlar to‘g‘risida tasavvurga ega bo‘lishi;

– algoritmlar, samarali algoritmlar ishlab chiqishning asosiy usullari, algoritmik tillar, dasturlash tillari, chiziqli, tarmoqlanuvchi va takrorlanuvchi dasturlar, modulli dasturlar, dastur tuzishni bilishi va ulardan foydalana olishi;

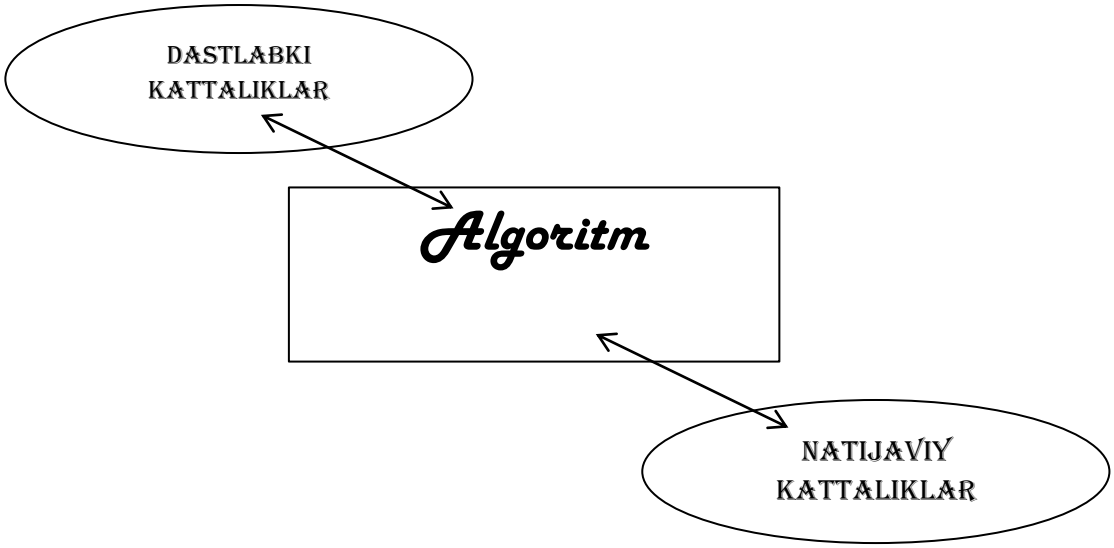
– algoritmlarni tasvirlash usullari, samarali algoritmlar ishlab chiqishning asosiy usullari, algoritmik tillar, massivlar, grafik operatorlar, satriy kattaliklar bilan ishlash, modulli dasturlar, dasturlash tillarida dastur tuzish ko‘nikmalariga ega bo‘lishi lozim.

Ushbu o‘quv qo‘llanma orqali “Algoritmlar” o‘quv fanini o‘zlashtirishda talaba yuqorida ta’kidlab o‘tilgan bilim va ko‘nikmalarga ega bo‘ladi.

I BOB. ALGORITMLAR HAQIDA DASTLABKI TUSHUNCHALAR

1-§. Algoritm tushunchasi va ulardan foydalanish

Algoritm- bu aniq hisoblashlarni bajaruvchi protsedura bo‘lib unga kirish qismida kattalik yoki kattaliklar berilib chiqishda natijaviy kattalik yoki kattaliklar olinadi. Demak algoritm hisoblovchi qadamlardan tashkil topgan bo‘lib, dastlabki qiymatlarga ko‘ra natijaviy kattaliklar qiymatini beradi. Bu holatni sxematik tarzda quyidagicha tasvirlash mumkin.



Algoritmni qo‘yilgan hisoblash masalani (**computational problem**) aniq bajaruvchi uskuna sifatida ham qaralishi mumkin. Algoritmlarda keltirilgan protseduralar yordamida kattaliklar bilan amallar bajarilib natijalar olinadi. Masalan, biror sonlar ketma-ketligini orta borish tartibida saralash. Saralash masalasi (**sorting problem**) ga misol keltiramiz:

Kirish: n-ta sondan iborat sonlar ketma-ketligi $(a_1,a_2,\ldots,a_n)$.

Chiqish: n-ta sondan iborat sonlar ketma-ketligi $(b_1 \le b_2 \ldots \le b_n)$.

Misol. (31, 41, 59, 26, 41,56) kiruvchi ketma-ketlik bo‘lsa, chiquvchi ketma-ketlik (26, 31, 41, 41, 56, 59) bo‘lishi lozim.

Bunga o‘xshash kiruvchi ketma-ketlik saralash namunasi **(instanse)** deb yuritiladi. Agar algoritm har qanday kiruvchi qiymatlar uchun aniq va mos chiquvchi qiymatlarni bera olsa, u aniq (**correct**) deb yuritiladi. [1]

Algorimlardan amaliyotda foydalanishga ayrim misollarni keltiramiz:

- Odam DNK si tarkibidagi 100 ming gen identifikatsiyasi, DNK-ni tashkil etuvchi 3 milliard asosiy juftlikni saralash va tahlili masalasi;
- Internetda ma’lumotlar olish masalasi: katta hajmdagi ma’lumotlarni olish, jo‘natish, qidiruv va optimal marshrut tanlash;
- Elektron tijorat masalalarida (kredit karta nomerlari, parollar, bank hisob-kitob raqamlari himoyasi, raqamli imzo va boshqalar);

Algoritmlarni ishlab chiqishda masalani yechimi uchun zarur bo‘lgan vaqt va xotira hajmi muhim ko‘rsatgichlar hisoblanib algoritmlarni yaratishda ularni samarali foydalanishni hisobga olish zarur. Aynan bir masalani yechish uchun turli algoritmlar tuzilishi mumkin. Ular bir-biridan samardorlik darajasi bilan farqlanadilar. Bu farq turli texnik va dasturiy ta’minotlarda har xil bo‘lishi mumkin.

Misol uchun ikkita saralash algoritmlari farqini ko‘rib chiqamiz:

Saralash algoritmi	**Sarflanadigan vaqt**	**Izoh**
Joylashtirish usuli	C_1n^2 bu n^2 ga proporsional	C_1-n ga bog‘liq bo‘lmagan doimiylik n-saralanadigan elementlar soni
Qo‘shish usuli	C_2nlgn	lgn=$\log_2$n, C_2-n ga bog‘liq bo‘lmagan doimiylik

Qo‘shish usuli joylashtirish usulidan samaraliroq ekanligini quyida keltirilgan jadval ma’lumotlarini tahlili orqali keltiramiz.

[1] Thomas H. Cormen va b. Intruduction to algorithms. Massachusetts Institute of Technology. London 2009.(5-10pp)

Kompu-terlar	Saralana-digan sonlar soni	Saralovchi algoritm	Talab qilinadigan vaqt
A(tez ishlovchi 1sekundda 10mlrd amal bajaradi)	10 mln ta(taqriban80 mb)	Joylashtirish usuli (tajri-bali dastur-chi tomoni-dan yaratil-gan algo-ritm sara-lash uchun $2n^2$ amal bajariladi)	$\frac{2 * (10^{7})^2 \ \ buyruqlar}{10^{10}\ buyruq/sec}$ $=20000\ sec$ *(5,5 soatdan ko‘proq)*
B(sekin ishlovch-1sekundda 10 mln amal bajaradi)		Qo‘shish usuli (o‘rta dara-jali dastur-chi tomoni-dan yaratil-gan algo-ritm sara-lash uchun 50nlgnamal bajariladi))	$\frac{50 * 10^7 \ \ lg10^7}{10^7} \approx 1163\ sekund$ $(20\ \text{min}\ dan\ kam)$

Umuman olganda algoritm - bu qo‘yilgan masalaning yechimiga olib keladigan, ma’lum qoidaga binoan bajariladigan amallarning chekli qadamlar ketma-ketligidir. Boshqacha qilib aytganda, algoritm boshlang‘ish ma’lumotlardan natijagasha olib keluvshi jarayonning aniq yozilishidir.

Algoritm tushunshasining turli ta’riflari bir qator talablarga javob berishi kerak:

- algoritm chekli sondagi elementar bajariluvshi ko‘rsatmalardan iborat bo‘lishi kerak;
- algoritm chekli sondagi qadamlardan iborat bo‘lishi kerak;

- algoritm barcha boshlang‘ich berilganlar uchun umumiy bo‘lishi kerak;

- algoritm to‘g‘ri yechimga olib kelishi kerak.

Har qanday algoritm ma’lum ko‘rsatmalarga binoan bajariladi va bu ko‘rsatmalarga buyruq deyiladi. Yuqoridagi fikrga ko‘ra algoritm asosan masalani yechimini topish uchun tuziladi.

Bitta masalani yechishning bir necha algoritmi mavjud bo‘lishi mumkin. Ular orasida eng samaralisini, bajarilishi uchun eng kam amallar, mashina vaqti, xotira va h.k.ni talab qiluvchi algoritmni tanlash lozim. Samarali algoritmlar mavjud bo‘lish shartlari va ularni qurish (ishlab chiqich)ni o‘rganish algoritmlar nazariyasi asosini tashkil etadi.

Algoritm kibernetika va matematikaning asosiy tushunchalaridan biri bo‘lib, bu atama o‘rta asrlarda yashab ijod etgan buyuk o‘zbek matematigi Al-Xorazmiy nomidan kelib chiqqan. U IX asrning 825 yilidayoq o‘zi kashf etgan o‘nli sanoq tizimida to‘rt arifmetika amallarini bajarish qoidalarini bergan. Arifmetika amallarini bajarish jarayoni esa al-xorazm deb atalgan. Bu atama 1747 yildan boshlab algorismus, 1950 yilga kelib algorifm deb ham ataldi. Fanda "Yevklid algoritmi", "G‘iyosiddin Koshiy algoritmi", "Laure algoritmi", "Markov algoritmi" deb ataluvchi algoritmlar ma’lum algoritm tushunchasi tobora kengayib borib, kibernetikaning nazariy va mantiqiy asosi hisoblangan algoritmlar nazariyasi paydo bo‘lgan. Kompyuterlar paydo bo‘lishi bilan algoritm atamasi hozirgi ma’nosi bilan axborot texnologiyalari sohasida eng asosiy atamalardan biri bo‘lib qoldi. Odatda algoritmlar u yoki bu hisoblashga doir masalalarni (**computational problems**) yechish uchun tuziladi.

Qo‘yilgan masala ushun yaratiladigan algoritmda kiruvchi va chiquvchi ma’lumotlar muhim ahamiyatga ega, agar algoritm to‘g‘ri tuzilgan bo‘lsa, ijrosi (**kompyuter**) aniq natijalar beradi.

Algoritm quyidagi xossalarga ega: aniqlik, tushunarlilik, ommaviylik, natijaviylik va diskretlik.

Aniqlik va tushunarlilik – deganda, algoritmda ijrochiga berilayotgan ko‘rsatmalar aniq mazmunda bo‘lishi tushuniladi. Chunki ko‘rsatmalardagi noaniqliklar mo‘ljallangan maqsadga erishishga olib kelmaydi. Ijrochiga tavsiya etiladigan ko‘rsatmalar

tushunarli mazmunda bo‘lishi shart, aks holda ijrochi uni bajara olmaydi.

Ommaviylik – deganda, har bir algoritm mazmuniga ko‘ra bir turdagi masalalarning barchasi uchun ham o‘rinli bo‘lishi, ya’ni umumiy bo‘lishi tushuniladi.

Natijaviylik – deganda, algoritmda chekli qadamlardan so‘ng albatta natija bo‘lishi tushuniladi. Shuni ta’kidlash joizki, algoritm avvaldan ko‘zlangan maqsadga erishishga olib kelmasligi ham mumkin. Bunga ba’zan algoritmning noto‘g‘ri tuzilgani yoki boshqa xatolik sabab bo‘lishi mumkin, ikkinchi tomondan, qo‘yilgan masala ijodiy yeshimga ega bo‘lmasligi ham mumkin. Lekin salbiy natija ham deb qabul qilinadi.

Diskretlik – deganda, algoritmlarni chekli qadamlardan tashkil qilib bo‘laklash imkoniyati tushuniladi.

Algoritmlarga doir quyidagi masalalarni misol sifatida keltirish mumkin:

- Talabani kundalik ishlarni tashkil etish;
- To‘rtburchak perimetri va yuzasini hisoblash;
- R radiusli doira yuzasini va aylana uzunligini topish;
- A_1, A_2, A_3,…, A_n sonlarni toq elementlarini yig‘indisini topish;
- Berilgan ketma-ketlik sonlarni o‘sish (kamayish) tartibda joylashtirish va h.k.

Algoritmning uchta turi mavjud: chiziqli, tarmoqlanuvchi va takrorlanuvchi(siklik).

Chiziqli algoritmlar - hech qanday shartsiz faqat ketma-ket bajariladigan jarayonlardir.

Tarmoqlanuvchi algoritmlar - ma’lum shartlarga muvofiq bajariladigan jarayonlardir.

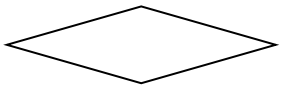
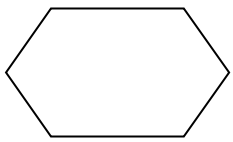
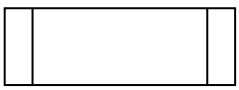
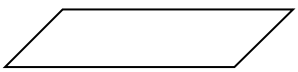
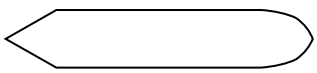
Takrorlanuvchi algoritmlar – biron-bir shart tekshirilishi yoki biron parametrning har xil qiymatlari asosida chekli ravishda takrorlanish yuz beradigan jarayonlardir.

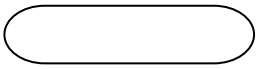
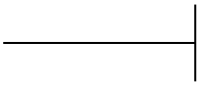
Algoritmlarni turli usullarda tasvirlash mumkin.

- so‘z bilan ifodalash;
- formulalarda berish;
- blok-sxemalarda tasvirlash;

- dastur shaklida ifodalash va boshqalar.

Algoritmlarni blok-sxema ko‘rinishda tasvirlash qulay va tushunarli bo‘lgani uchun eng ko‘p ishlatiladi. Bunda algoritmdagi har bir ko‘rsatma o‘z shakliga ega. Masalan: parallelogramm ko‘rinishdagi belgi ma’lumotlarni kiritish va chiqarish; to‘g‘ri to‘rtburchak belgisi hisoblash jarayonini; romb belgisi shartlarning tekshirilishini bildiradi. Algoritimni blok-sxema shaklida tasvirlashda quyidagi geometrik figuralardan foydalaniladi:

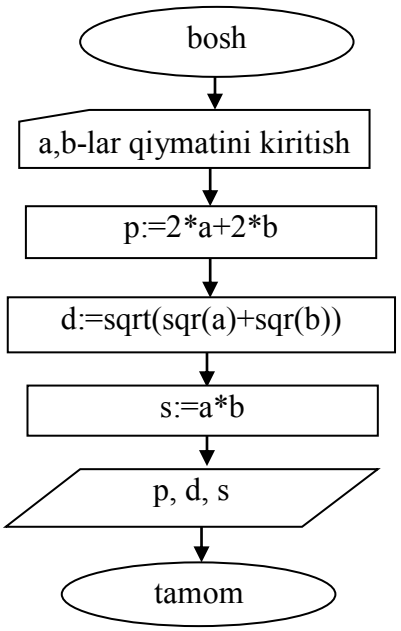
Nomi	**Belgilanishi**	**Bajaradigan vazifasi**
Jarayon		Bir yoki bir nechta amallarni bajarilishi natijasida ma’lumotlarning o‘zgarishi
Qaror		Biror shartga bog‘liq ravishda algoritmning bajarilish yo‘nalishini tanlash
Shakl O‘zgartirish		Dasturni o‘zgartiruvchi buyruq yoki buyruqlar turkumini o‘zgartirish amalini bajarish
Avval aniqlangan jarayon		Oldindan ishlab chiqilgan dastur yoki algoritmdan foydalanish
Kiritish Chiqarish		Axborotlarni qayta ishlash mumkin bo‘lgan shaklga o‘tkazish yoki olingan natijani tasvirlash
Displey		EHMga ulangan displeydan axborotlarni kiritish yoki chiqarish
Hujjat		Axborotlarni qog‘ozga chiqarish yoki qog‘ozdan kiritish
Axborotlar oqimi chizigi		Bloklar orasidagi bog‘lanishlarni tasvirlash
Bog‘lagich		Uzilib qolgan axborot oqimlarini ulash belgisi

Boshlash Tugatish		Axborotni qayta ishlashni boshlash, vaqtincha yoki butunlay to‘xtatish
Izoh		Bloklarga tegishli turli xildagi tushuntirishlar

Algoritmlarni tasvirlash usullariga misollar keltirib o‘tamiz:

Masala: to‘g‘ri to‘rtburchakning tomonlariga ko‘ra uning perimetri, diagonali va yuzasini hisoblash.

1. So‘z bilan ifodalash:
1.1. boshlash;
1.2. tomonlar qiymatini kiritish (a, b);
1.3. perimetr qiymatini hisoblash (p);
1.4. diagonal qiymatini hisoblash (d);
1.5. yuzasini hisoblash (s);
1.6. perimetr, diagonal va yuzasini qiymatini chop etish.
2. Formulalarda berish:
2.1. A va B to‘rtburchak tomonlari qiymatlari;
2.2. *P=2*a+2*b;*
2.3. $D=\sqrt{a^2+b^2}$;
2.4. S=a*b;
2.5. P, D va S qiymatlarini chop etish
3. Blok-sxemalarda tasvirlash:



4. Dastur shaklida ifodalash: (Paskal dasturlash tili misolida)

```
Program to‘rtburshak yuzi;
Var a, b: Integer;
P, d, s: real;
Begin
Write (‘a,b tomonlarni qiymatlari kiritilsin’);
ReadLn(a,b);
P:=2*a+2*b;
D:=sqrt(sqr(a)+sqr(b));
S:=a*b;
WriteLn(‘to‘rtburshak perimetri=’,p);
WriteLn(‘to‘rtburshak dioganperli=’,d);
WriteLn(‘to‘rtburshak yuzasi=’,S);
End.
```

Hozirgi kunda juda ko‘p algoritmik tillar mavjud bo‘lib, ularni *dasturlash tillari* deb ataymiz. Algoritmik til- algoritmlarni bir xil va aniq yozish uchun ishlatiladigan belgilashlar va qoidalar tizimidir. Algoritmik til oddiy tilga yaqin bo‘lib, u matematik belgilarni (yuqorida aytilganidek) o‘z ichiga oladi. Qo‘yilgan masalalarni yechishga tuzilgan algoritmlarni to‘g‘ridan-to‘g‘ri mashinaga berib, yechib bo‘lmaydi, shu sababli yozilgan algoritmni biror-bir algoritmik tilga o‘tkazish zarur. Har qanday algoritmik til o‘z qo‘llanilish sohasiga ega. Masalan, o‘quv jarayonlari uchun Pascal, Delphi, VBA, java, C++dasturlash tillari va boshqalar.

1-misol: kiritilgan n-natural sonni tub ko‘paytuvchilarga ajratuvchi algoritmni Pascal dasturlash tilida ifodalanishini ko‘rib chiqamiz:

```
var i,k:integer;  n:integer;
a:array[byte] of integer; label qq;
procedure opr(nn:integer);
begin i:=2;
while(nn>0) do
begin
if nn mod i=0 then write(i,' ');
i:=i+1;
nn:=nn div i;
end;
```

```
end;
begin
readln(n);
k:=1;
for i:=2 to n do
while (n mod i=0) do
begin   a[k]:=i; write(a[k],' ');  n:=n div i ; k:=k+1; end;
writeln;
readln;
end.
```

2-misol: $\int_0^5 x^2 dx$ – qiymatini hisoblovchi dastur tuzing.

```
procedure TForm1.Button1Click(Sender: TObject);
var
h,a,x,d,b,s:real;
n:integer;
begin
a:=0;  s:=0;
b:=5;
n:=10000;
h:=(b-a)/n;
x:=a;
while (x<b) do
begin
d:=sqr(x);
s:=s+d*h;
x:=x+h;
end;
end;
end.
```

2-§. Algoritmlar samaradorligini baholash

Algoritmlar texnologiya sifatida.

Kompyuteringizning tezligi va xotira miqdorini abadiy oshirish mumkin, deylik. Bu holatda algoritmlar o‘rganish kerakmi? Bor bo‘lishi mumkin, lekin faqat namoyish etish uchun, yechim usulini

cheklangan vaqti bor va u to‘g‘ri javob beradi. Kompyuterlar juda tez bo‘lganda, masalani yechishga har qanday konkret usul mos kelarmidi.

Albatta, bugungi kunda juda samarali kompyuterlar, lekin ularning ishlashi juda katta bo‘lishi mumkin emas. Xotira ham arzon, lekin bepul bo‘lishi mumkin emas. Shunday qilib, hisob-vaqti - cheklangan resurs, shuningdek xotira miqdori ham. Siz donolik bilan bu resurslarni boshqarishingiz kerak, bunga algoritmlardan, vaqt va xotira xarajatlaridan samarali foydalanish kerak.

Samaradorligi

Har xil masalalarni yechish uchun mo‘ljallangan, turli xil algoritmlar, samaradorligi bo‘yicha sezilarli darajada farq qiladi. Bu farqlar juda katta bo‘lishi mumkin ekan. Masalan, ikki saralash algoritmlar olsak, birinchisini bajarish uchun, saralashni joylashtirish, bunga vaqt kerak bo‘ladi, shunday baholanmoqda c_1n^2, n- bu saralash elementlarning soni, c_1 bo‘lsa bu – doimiy, n ga bog‘liq emas. Shunday qilib, bu algoritmni vaqti taxminan n^2 ga proporsional.

Ikkinchi algoritmni amalga oshirish uchun, saralash birlash-tirishi, vaqt talab etadi, taxminan c_2n lg n ga teng, lgn- bu $\log_2 n$ qisqa yozuvi, c_2 bu - boshqa doimiy n ga bog‘liq emas. Odatda doimiy usul qo‘shimchalar doimiy birlashish usulidan kichikroq, $c_1 < c_2$. Doimiy omillar algoritmni ish vaqtiga juda katta ta’sir qiladi, n ga bog‘liq omillardan ko‘ra, shunga ishonch hosil qilaylik. Saralashni joylashtirish algoritmni ish vaqtini shunday yozaylik c_1 n n, birlashtirish saralashini esa c_2n lgn.

Joylashtirish saralashi n omilga ega, birlashtirish saralashi esa lgn ga ega bu esa sezilarli darajada kamligini ko‘rishimiz mumkin. Kiritish hajmi n yetarlicha katta bo‘lganda qo‘shish saralashi odatda tezroq bo‘ladi, saralash obyektlar kichik hajmdagi birlashtirishda, katta n uchun ahamiyatsiz qiymati lgn nisbatan n to‘liq doimiy farqi qadriyatlar o‘rnini qoplash, aslida birlashish yanada sezilarli namoyon bo‘ladi, saralash afzalligi ziyoda. Bu doimiy c_1, c_2 dan necha marta kam muhim emas.

Misol tarzida ikkita A va B kompyuterlarni ko‘rib chiqamiz. A kompyuteri ancha tezroq va unda joylashtirish saralash algoritmi ishlaydi, B kompyuter esa sekin va unda saralash algoritmi birlashtirish usuli bilan ishlaydi. Har ikkita kompyuterlar bir nechta

saralashni bajarishi kerak. Kompyuter A sekundiga o‘n milliard ko‘rsatmalar bajaradi, B kompyuter sekundiga faqat o‘n million ko‘rsatmalar bajaradi, shunday qilib A kompyuteri ming marta B kompyuterdan tez. Saralash birlashishi yuqori darajadagi til yordamida bir dasturchi tomonidan amalga oshirilgan. Bu kompilyator juda samarali emas edi, va natija 50 nlgn ko‘rsatmalarga bajaradigan kod paydo bo‘ldi.

O‘n million raqamlarini tartiblashtirish uchun A kompyuterga kerak bo‘ladi:

$$\frac{2 * (10^7)^2}{10^{10}} = 20000$$

B kompyuterga kerak bo‘ladi

$$\frac{50 * 10^7 \ \lg 10^7}{10^7} \approx 1163$$

Ko‘rib turganingizdek, kod bilan foydalanish, ish vaqti sekin ko‘tarilganda, yomon kompilyator bilan ham eng sekin kompyuterda ham 17 marta kam vaqt talab qiladi.

Qo‘shish usuli joylashtirish usulidan samaraliroq ekanligini quyida keltirilgan jadlal ma’lumotlarini tahlili orqali keltiramiz.

Kompu-terlar	**Saralana-digan sonlar soni**	**Saralovchi algoritm**	**Talab qilinadigan vaqt**
A (tez ishlovchi 1sekund-da 10mlrd amal bajaradi)	10 mln ta (taqriban 80 mb)	Joylashtirish usuli (tajribali dasturchi tomonidan yaratilgan algoritm saralash uchun $2n^2$ amal bajariladi)	$\frac{2 * (10^7)^2 \ \ buyruqlar}{10^{10}\ buyruq/sec}$ *=20000 sec (5,5 soatdan ko‘proq)*
B (sekin ishlovchi-		Qo‘shish usuli (o‘rta	$\frac{50 * 10^7 \ \ lg 10^7}{10^7} \approx 1163\ sekund$

1 sekundda 10 mln amal bajaradi)		darajali dasturchi tomonidan yaratilgan algoritm saralash uchun 50 nlgn amal bajariladi))	(20 min *dan kam*)

Algoritmlar va boshqa texnologiyalar

Yuqoridagi misol shuni ko‘rsatadiki, kompyuter apparat kabi algoritmlarni ham, texnologiya sifatida hisobga olinishimiz kerak.

Umumiy tizim ish faoliyatini algoritm samaradorligiga ham bog‘liq va apparat kuchiga ham. Algoritm rivojlantirish sohasida jadal rivojlantirish bo‘lyapti, boshqa kompyuter texnologiyalaridek.

Savol tug‘iladi, algoritmlar shunchalik muhimi, zamonaviy kompyuterlarda ishlaydigan bo‘lsin, agar shunday kabi yuqori texnologiyalar boshqa sohalarda ulkan yutuqlarga erishilgan bo‘lsa:

• zamonaviy kompyuter mimarileri va ularning ishlab chiqarish texnologiyalari;

• osonlik bilan erishish, intuitiv grafik foydalanuvchi interfeysi (GUI);

• obyektga yo‘naltirilgan tizimlar;

• integratsiyalashgan web texnologiyasi;

• tezroq tarmoqlari, simli va simsiz. [2]

Misol uchun, bir joydan boshqasiga olish uchun qanday belgilaydigan Web xizmat. Uni amalga oshirish bir yuqori samarali apparat, grafik foydalanuvchi interfeysi, bir global tarmoq ehtimol, bir obyekt yo‘naltirilgan yondashuv yotadi.

Bundan tashqari, bunday yo‘nalishlarini topish kabi bir berilgan web-xizmati tomonidan amalga muayyan operatsiyalar uchun zarur algoritmlarni foydalanish, ko‘rish va enterpolatsyon manzilini, xaritalar bilan foydalaniladi. Bundan tashqari, dastur,

[2] Thomas H. Cormen va b. Intruduction to algorithms. Massachusetts Institute of Technology. London 2009. (11-13pp)

yuqori saviyada algoritmik mazmunini talab qilmaydi, kuchli algoritmlarga bog‘liq. Bu dastur ishlash apparat ishiga bog‘liq ekanligi ma'lum va amaliy rivojlanishida turli algoritmlardan foydalaniladi.

Biz hammamiz bilamizki, ilova yaqindan grafik foydalanuvchi interfeysi bilan bog‘liq va har qanday grafik foydalanuvchi interfeysini ishlab chiqish uchun talab algoritmlari kerak bo‘ladi. Tarmoq ustida ishlaydigan ilovalarni eslatib o‘tamiz.

Ular faoliyat olib borishlari uchun, algoritmlarga asoslangan yo‘nalishni olib borishlari kerak bo‘ladi. Eng keng tarqalgan dasturlar tilda tuziladi, mashinadan farqli. Ularning kodi turli kompilyator va interpretatorlar bilan ishlov beriladi, turli algoritmlardan keng foydalanadi. Bundan tashqari, kompyuterlar kuchini doimiy o‘sishi, ular tobora murakkab vazifalarni hal qilish uchun qo‘llaniladi. Biz muammoni murakkabligini ortishimiz bilan, ikki saralash usullari qiyosiy tahlili misolida ko‘rib turganimizdek eng muhim farqlar algoritmlari samaradorligi oshirilmoqda. Asosiy algoritmlar va ularni rivojlantirish usullari-asosiy xususiyatlatdan biri. Zamonaviy kompyuter texnologiyalari bilan, ayrim vazifalarni algoritmlarni bilmagan holda ham qilinishi mumkin, lekin bu sohada ko‘p narsaga erishish mumkin.

Mashqlar

1.2.1 Dastur darajasida zarur bo‘lgan algoritmik content dasturini misol qilib keltiring va bu algoritmlarni funksiyasini muhokama qiling.

1.2.2 Deylik, bitta mashinada ikkita saralash algoritmni qiyosiy tahlil amalga oshirilmoqda. N elementlarni joylashtirish saralashi uchun $8n^2$ kerak bo‘ladi, birlashtirish saralashi uchun 64n lgn qadamlar kerak bo‘ldi. Joylashtirish saralashi birlashtirish saralashidan qiymati oshsa, n ni qiymati qancha bo‘lishi kerak?

1.1. Algoritmlarni ish vaqtini solishtirish

Quyida bir jadval bo‘lib, satrlari turli vazifalarga mos f(n), ustunlaru esa- vaqt qiymatiga t. N ni maksimal qiymatlari bilan jadvalni to‘ldiring, masala t vaqti bilan yechilishi mumkin, agar

masalani yechish uchun algoritmni ish vaqti f(n)mikro sekundga teng bo‘lsa.

	1 sekund	1 minut	1 soat	1 kun	1 oy	1 yil	1 asr
$\lg n$							
$\sqrt{n}$							
n							
$nlgn$							
n^2							
n^2							
n^2							
$n!$							

3-§. Tanlash va joylashtirish turkumidagi murrakkablikga ega saralash algoritmlari

Ushbu mavzuda quyidagi saralash masalasini hal qilish mumkin bo‘lgan bir qancha algoritmlar keltirilgan.

Kirish. n sonlardan iborat $\{a_1, a_2, \dots, a_n\}$ ketma-ketlik.

Chiqish. Tartiblangan $\{a_1, a_2, \dots, a_n\}$ kiritish ketma-ketligi $\{a_1{}' \le a_2{}' \le \cdots \le a_n{}^n\}$ dan iborat.

[3]Odatda kiritish ketma-ketligi ***n-chi*** massiv elementi ko‘rinishida bo‘ladi, shu bilan birga yana bog‘langan ro‘yhat ko‘rinishida ham bo‘lishi mumkin.

Amaliyotda saralanayotdan sonlar kamdan-kam hollarda yakkalangan qiymatlar hisoblanadi. Odatlar ularning har biri yozuv **(record)** deb nomlanuvchi berilganlarning tarkibiga kiradi. Har bir yozuv**(record)**da qiymatlarni saralovchi kalitlar bo‘ladi. Saralash algoritmi amaliyotda shunday amalga tatbiq qilinishi kerakki, u kalitlari bilan birgalikda tegishli bo‘lishi va qayta ishlanishi kerak.

Saralash algoritmi saralash tartibini aniqlab beruvchi saralanishiga bog‘liq bo‘lmagan alohida sonlar yoki ko‘pbaytli katta yozuvlar bilan berilganlardan iborat usullarni tasvirlaydi. Shu tariqa,

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 16-p.

agar saralash masalasi haqida gap ketsa, berilganlar faqat sonlardan iborat bo‘ladi.

Nima uchun saralashdan foydalanamiz?

Hisoblash texnikasi sohasidagi ko‘plab olimlar saralashni algoritmlarni o‘rganishda eng fundamental masala deb qarashadi. Buning bir qancha sabablari bor:

1) Ba’zan ilovalarda axborotlarni saralamaslikning aslo iloji bo‘lmaydi. Masalan, bankda mijozlarning hisob holatlari to‘g‘risidagi hisobotni tayyorlash uchun ularning chek nomerlari bo‘yicha saralashni bajarish kerak bo‘ladi.

2) Odatda saralash algoritmlarda kalit qismdasturi sifatida ishlatiladi. Masalan, dasturda, turli xil darajalarda bo‘lgan grafik obyektlarni vizual berkitishni bajarishda, dastlab ushbu obyektlarni chiqish tartibini o‘rnatish uchun obyektlarni “pastdan tepaga” darajalari bo‘yicha saralash kerak bo‘ladi.

3) Saralash algoritmlarning ko‘plab turli xil texnologiyalar qo‘llaniladigan turlari mavjud. Algoritmlarni saralashda turli xil algoritm sinflariga qo‘llaniladigan juda ko‘p muhim usullardan foydalaniladi.

4) Algoritmlarni saralash jarayonini amalga oshirishda oldingi qatorga juda ko‘plab amaliy muammolar chiqadi. Ko‘plab saralash dasturi ishlab chiquvchilarni tanlash shu yoki boshqa vaziyatlarda juda ko‘p faktlarga bo‘g‘liq bo‘lishi mumkin. Juda ko‘p shunga o‘xshash masalalar hal qilishda “kodlarni sozlash” dan ko‘ra “algoritmlar” darajasidan foydalanish maqsadga muvofiqdir.

Saralash algoritmlari

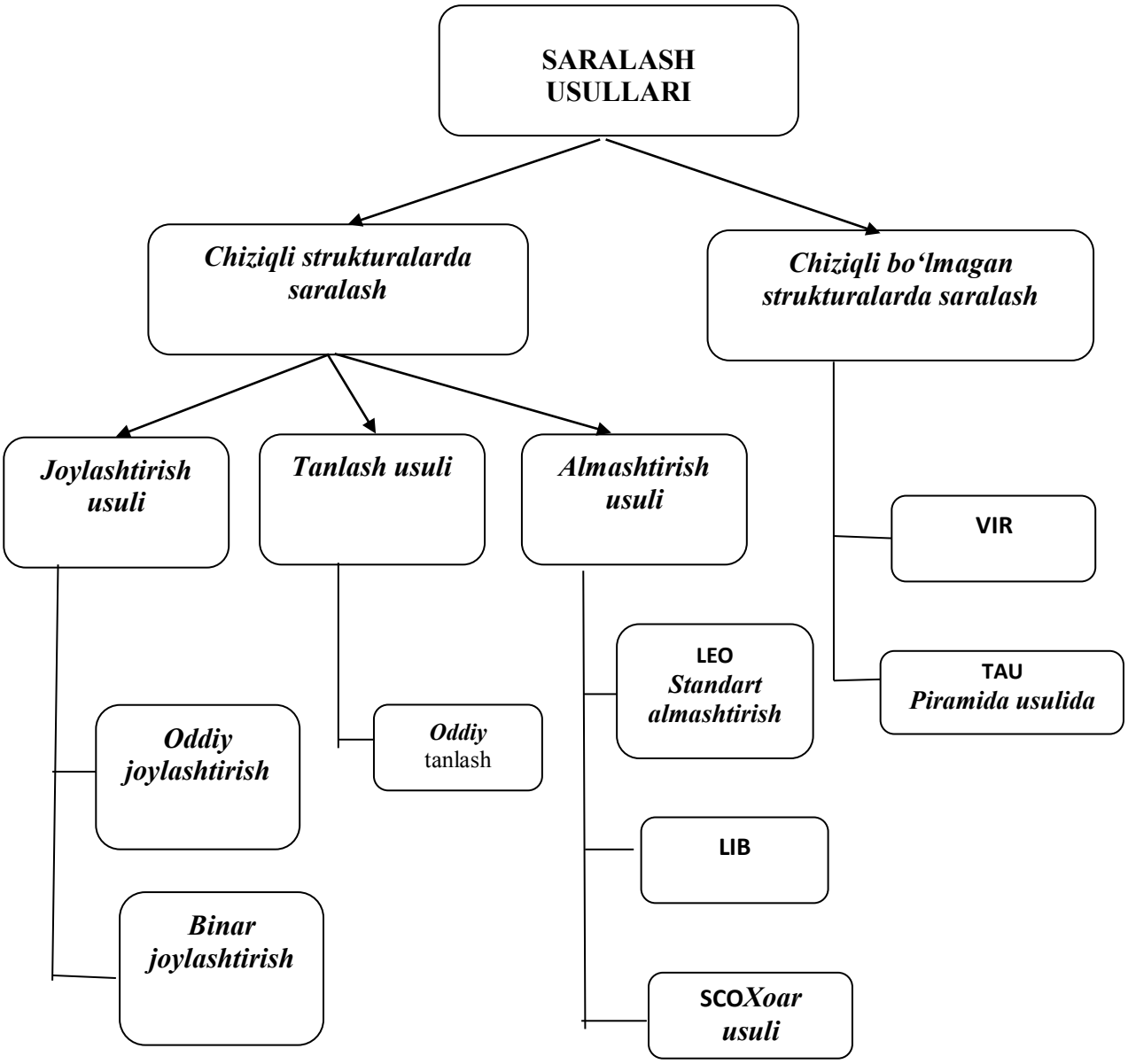
O‘sish yoki kamayish tartibida to‘plam elementlarini *tartiblangan saralash* deyiladi.

Tartiblangan elementlar bilan ishlash tartibsiz joylashgan elementlardan ko‘ra qulayroq: kerakli elementlarni yengil topish, olib tashlash, yangilarini qo‘yish mumkin.

Saralash algoritmlarini quyidagi guruhlarga ajratish mumkin (1-chizma):

Odatda saralanayotgan to‘plam elementlari yozuvlar deyiladi va $k_1, k_2, \dots, k_n$ ko‘rinishida yoziladi.[4]

Saralashlar turlicha algoritmlar bajarilsa-da, yagona natijalarga olib keladi. Ammo saralashlar jarayonida sarflanadigan vaqt miqdori ularning o‘zaro taqqoslashdagi me’zonlardan biri hisoblanadi. Dastur bajarilish vaqti amallar bajarilish soniga va protsessor tezligiga proporsional.



1-chizma. Saralash algoritmlari

[4] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 66-с.

Algoritmning vaqt bo‘yicha murakkabligi $T_\alpha(V)$- (α algoritm ushun) bilan belgilanadi. Bu yerda V- α algoritm bajarilishi uchun zarur bo‘lgan dastlabki kattaliklar miqdori.

Function fast(x:integer):integer;
Var m,i:integer;
Begin
m:=1;
for i:=2 to x do
m:=m*I;
fast:=m
end;
	1 marotaba bajariladi
	
	2 ta amal (ko‘paytirish va qiymat berish) 2*(x-1)
	1 marotaba bajariladi

Agar bajariladigan har bir amalni murakkablikning 1 birligi sifatida qabul qilsak, uning qiymati *1+2(x-1)+1=2x* ga teng bo‘ladi. Demak, vaqt bo‘yicha murakkablik $T_\alpha(V)=2V$ ekanligi kelib chiqadi. o‘z navbatida bu murakkablik qiymati chiziqli holda fast funksiyasining parametriga bog‘liqligi aniqlanadi.

Joylashtirish usulida saralash eng yomon holda $\theta(n^2)$ vaqtida bajariladi. Qo‘shish saralash usulining asimptotik eng yaxshi ishlash vaqti $\theta(n\,lg\,n)$ ga teng.

Joylashtirish, qo‘shish, piramida va tezkor saralash usullarining bitta umumiy tomoni shundan iboratki, ular berilgan massiv elementlarini juftliklarda taqqoslash prinsipi bo‘yicha ishlaydi.

Quyidagi jadvalda saralash algoritmlarining ish vaqti haqida ma’lumotlar keltirilgan. Odatdagidek, n saralanishi kerak bo‘lgan elementlar sonini bildiradi. Saralash jarayonida saralanadigan elementlar $\{0, 1, \dots, k\}$ to‘plamdagi butun sonlar bo‘ladi. Saralash holida har bir element ***d-*** qiymatli sonni tasvirlaydi, har bir uning soni turli xil k qiymatlarni qabul qiladi. Cho‘ntak usulida saralashda kalitlar bir xil tartibda yarim ochiq $[0.1)$ haqiqiy sonlar oraliqda joylashgan bo‘ladi. O‘ng tomondagi chetki ustunda algoritmlarning o‘rtacha holda ish vaqtlari yoki kutilayotgan ish vaqtlari keltirilgan.

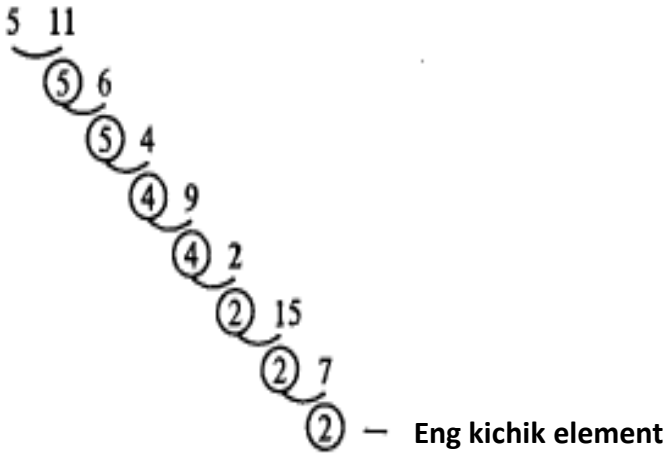
ALGORITMLAR	ENG YOMON HOLDA ISH VAQTI	O‘RTACHA (KUTILAYOTGAN) HOLDA ISH VAQTI
Joylashtirish usulida saralash	$\theta(n^2)$	$\theta(n^2)$
Qo‘shish usulida saralash	$\theta(n\,lg\,n)$	$\theta(n\,lg\,n)$
Piramida usulida saralash	$\theta(n\,lg\,n)$	-
Tezkor saralash	$\theta(n^2)$	$\theta(n\,lg\,n)$ **(kutilayotgan)**
Sanash usulida saralash	$\theta(k+n)$	$\theta(k+n)$
Tashlash usulida saralash	$\theta(d(n+k))$	$\theta(d(n+k))$
Cho‘ntak saralash	$\theta(n^2)$	$\theta(n)$ **(o‘rtacha holda)**

Tanlash saralash

Tanlash saralash boshida tartibsiz ro‘yxatdan eng kichik elementni tanlanashdan iborat. Shundan so‘ng dastlabki ro‘yxat o‘zgaradi. O‘zgartirilgan ro‘yxat boshlang‘ich ro‘yxat sifatida qabul qilinadi va jarayon barcha elementlar tanlangungacha davom etadi. Tanlangan elementlar tartiblangan ro‘yxatni hosil qiladi. Masalan, ro‘yxatdan eng kichik elementni topish talab etilsin:

$$\{5, 11, 6, 4, 9, 2, 15, 7\}.$$

Tanlash jarayoni chizmada keltirilgan. Tanlanayogan elementlar kichik hajmda aylanaga olingan. Taqqoslanayotgan elementlar soni rasmdagi satrlar soniga mos kelishini, shuningdek, elementlarni ko‘chirish soni tanlangan elementlarni o‘zgartirish soniga mos kelishini Ko‘rish qiyin emas. $\{5, 11, 6, 4, 9, 2, 15, 7\}$.



2-chizma. Tanlash usulida saralash

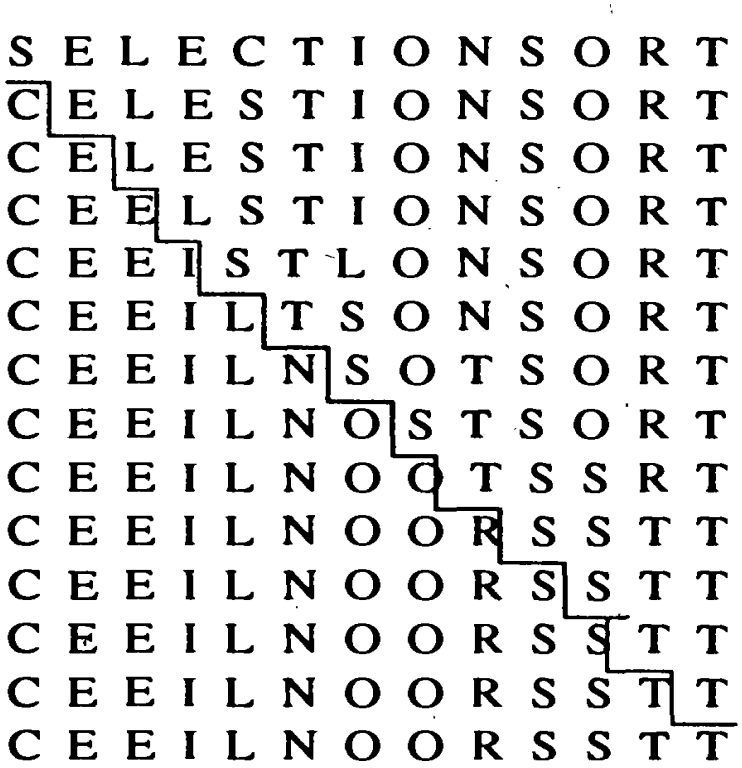
Boshlang‘ich ro‘yxatdan tanlangan eng kichik element unga berilgan joyga bir qancha usullar bilan joylashadi:

- Eng kichik element *i chi* marta *i chi marta (i=1, 2, ..., n)* yangi ro‘yxat joyiga ko‘chirilganda, boshlang‘ich ro‘yxatning tanlangan element joyiga boshqa katta element joylashadi, bu bilan berilgan ro‘yxat uzunligi o‘zgarmaydi. Ushbu usulda o‘zgartirilgan ro‘yxatni boshlang‘ich ro‘yxat sifatida qabul qilish mumkin.
- Eng kichik element boshlang‘ich ro‘yxatning *(i=1, 2, ..., n) i* chi joyiga joylashadi, *i chi* joyning elementi esa tanlangan element joyiga joylashadi.
- Shu bilan birga ko‘rinib turibdiki, tartiblangan elementlar keyingi saralashdan chiqariladi, shuning uchun ham har bir keyingi ro‘yxatning uzunligi oldingi ro‘yxatdan bitta elementga kam bo‘lishi kerak.
- Tanlangan eng kichik element oldingi holdagi kabi berilgan ro‘yxatning *i chi* joyiga, *i chi* joy keyingi eng kichik elementni yozish uchun bo‘shashi uchun tanlangan elementdan ro‘yxatning chap tomonida turgan qismi o‘ng tomonga bitta pozitsiya bilan tanlangan element joyni to‘ldirish uchun siljiydi. (Ro‘yxat elementlari sikl bo‘yicha siljiydi). [5]

Tanlash usulida saralash murakkabligi $O(n^2)$ tartibda tashkil qiladi.

[5] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 67-68 сс.

Tanlash saralash usulida eng kichik saralanmagan element aniqlanib massivning saralangan qismining oxiriga joylashtiriladi. Bu hol massivning barcha elementlari saralanib bo‘lguncha davom etadi. Bu holatni quyidagicha tasvirlash mumkin.



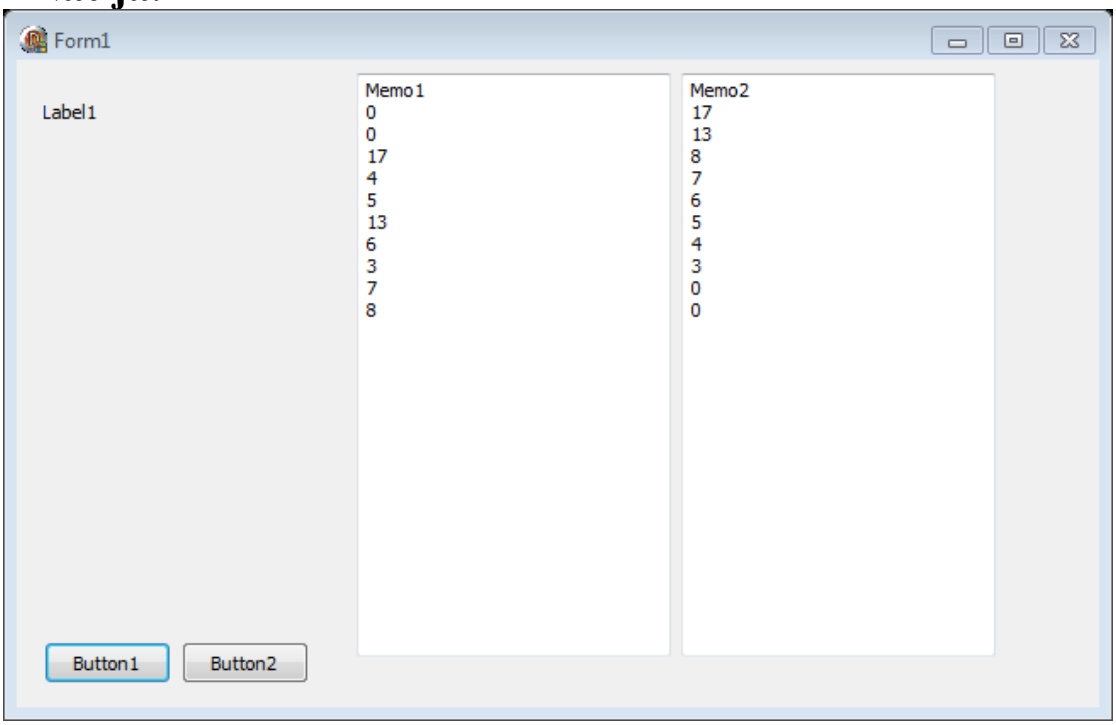
Ushbu saralashning kodini keltirib o‘tamiz:

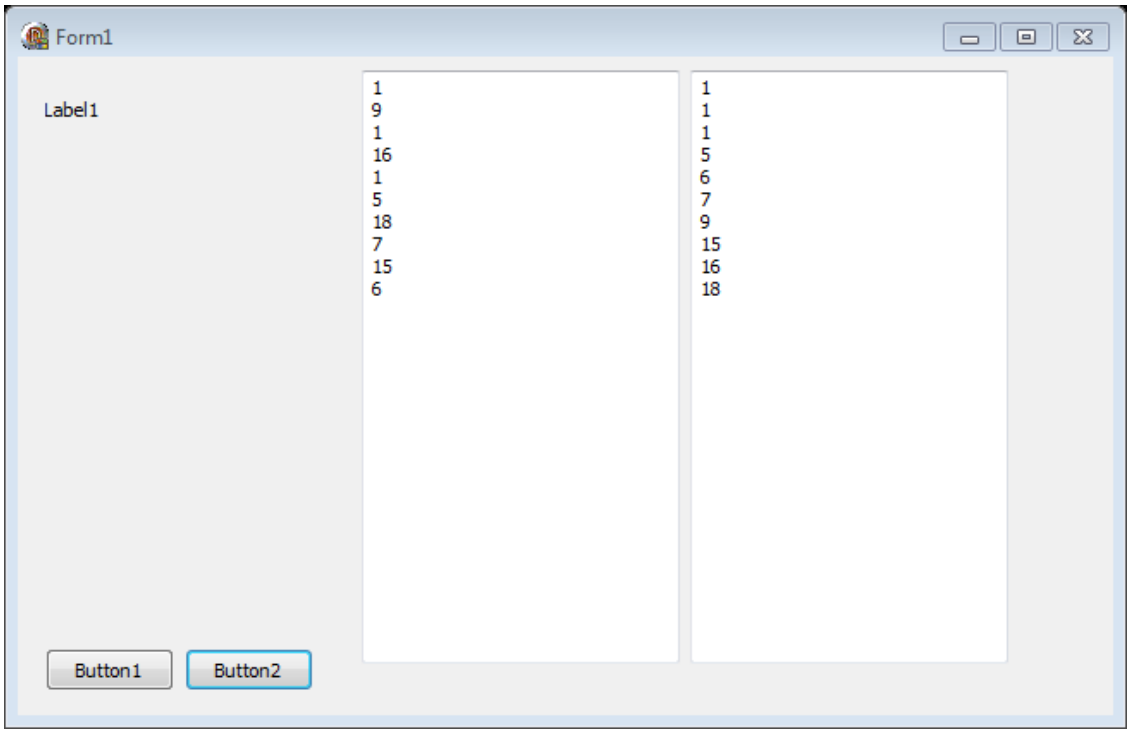
```
procedure TForm1.Button1Click(Sender: TObjest);
var a:array [byte] of integer;
i,j,k,m,z,d:integer;
begin k:=10;
for i := 1 to k do
begin
a[i]:=random(20);
memo1.Lines.Add(inttostr(a[i]));
end;
for j :=1 to k do
begin
m:=a[j];
for i := j to k do begin
```

```
if m<a[i] then
begin
m:=a[i];
d:=i;
z:=a[j];
a[j]:=m;
a[d]:=z;
end ;
end;
end;
for i := 1 to k do
memo2.Lines.Add(inttostr(a[i]));
end;

procedure TForm1.Button2Click(Sender: TObjest);
var a:array [byte] of integer;
i,j,k,m,z,d:integer;
begin k:=10;
memo1.Slear;
memo2.Slear;
for i := 1 to k do
begin
a[i]:=random(20);
memo1.Lines.Add(inttostr(a[i]));
end;
for j :=1 to k do
begin
m:=a[j];
for i := j to k do begin
if m>a[i] then
begin
m:=a[i];
d:=i;
z:=a[j];
a[j]:=m;
a[d]:=z;
end ;
```

```
end;
end;
for i := 1 to k do
memo2.Lines.Add(inttostr(a[i]));
end;end.
```

Natija.





Dastur tarkibidagi tashqi sikl n (10) marotaba. Ichki sikl n-i-1 bajariladi, if operatori esa

$$S(n) = (n-1) + (n-2) + (n-3) + ... + 2 + 1$$

marotaba bajariladi. Demak, algoritmning murakkablik darajasi uchun S(n)<=n(n-1)=O(n^2) tenglik o‘rinli.

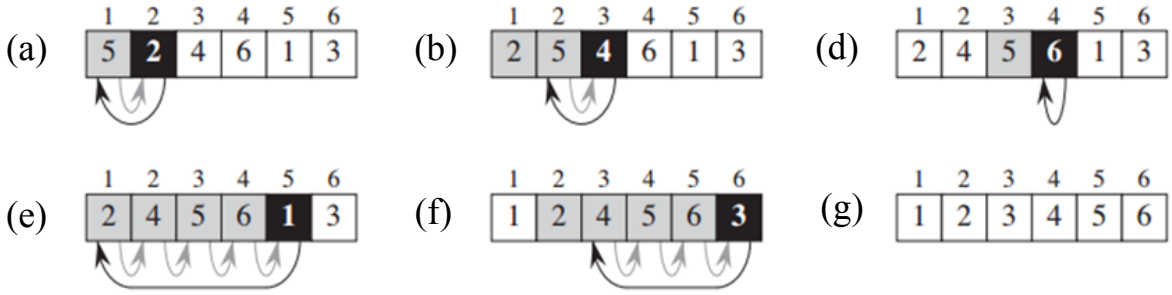
JOYLASHTIRISH SARALASH USULI

Biz joylashtirish usulida saralash **(Insertion-sort)**ni ko‘rib chiqaylik. Ushbu saralash algoritmi katta bo‘lmagan elementlarni saralashda qo‘llaniladi. Saralash jarayonini sonlarni tartiblashdan boshlaymiz. Dastlab chap tomonda bo‘sh bo‘lsin. So‘ng sonlarni chap tomonga joylashtirib boramiz.

[6]Joylashtirish usulida saralash psevdokodi saralanishi lozim bo‘lgan n uzunlikdagi ketma-ketlikdan iborat $A[1..n]$ massiv parametri sifatida Insertion-sort deb nomlangan protsedura ko‘rinishida quyida keltirilgan. Algoritm kiritilgan sonlarni qo‘shimcha xotira talab qilmasdan o‘z o‘rnida saralaydi.

Faqat bitta pozitsiyaga o‘ng tomonga siljiydigan massivlar qiymati kulrang ko‘rsatgich bilan ko‘rsatilgan (6-satr), qora rangli ko‘rsatgich orqali – kalitning ko‘chishi ko‘rsatilgan (8-satr). **(e)** qismida saralangan massivning yakuniy holati ko‘rsatilgan. Insertion-sort prosedurasini yakunlanishida A kiritish massivi saralangan chiqarish ketma-ketligidan iborat bo‘ladi. **3-chizma.** $A = \{5, 2, 4, 6, 1, 3\}$ massivni joylashtirish usulida saralash (**Insertion-sort**). Kvadratlar bilan belgilangan massiv elementlarining yuqori qismida elementlar indekslari va kvadratlar ichida mos elementlar qiymatlari berilgan. Bu rasmning **a-d** qismi **for** sikliga mos keladi. 1-8 qatorida **for** sikl iteratsiya psevdokodlari **(a)-(d)** rasm qismiga mos keladi. Har bir iteratsiyada qora kvadratlarda **A[j]** kaliti qiymatlari tashkil topgan bo‘lib, undan chapda joylashgan (psevdakodning 5-satri) kulrang kvadratlar bilan taqqoslanadi.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 17-p.



2.2-rasm A = {5,2,4,6,1,3} qatorida INSERTION-SORTning ishlashi. Array indeces to‘rtburchaklar ustida paydo bo‘ladi va qator pozitsiyalarida saqlangan qiymatlar to'rtburchaklar ichida paydo bo‘ladi. (a) - (e) 1-8 bosqichlari uchun aylananing yinelemalari. Har bir iteratsiyada qora to‘rtburchak A [j] dan olingan tugmachani ushlab turadi, bu esa soyali to‘rtburchaklardagi qadriyatlar bilan 5-qatorli testda chap tomonga taqqoslanadi. Shaded arrowlar ketmaketliklarni bir qatorni 6-satrda o‘ngga ko‘chiradi, va qora o‘qlar kalit 8-qatorga qayerda ekanligini ko‘rsatadi. (f) Oxirgi tartiblangan qator.

Chizmada ushbu algoritm $A = \{5, 2, 4, 6, 1, 3\}$ massivda qanday ishlashi ko‘rsatilgan. ***j*** saralanayotgan sonlar indeksini bildiradi. Dastlab ***j*** indeksli ***A*** massivning har bir **for** sikl iteratsiyasi ikki qismdan tashkil topadi. $A[1..j-1]$ elementlari saralangan sonlarga mos keladi, $A[j+1..n]$ elementlari esa hali saralanmagan sonlardir. $A[1..j-1]$ elementlari dastavval ***1*** dan ***j-1*** gacha bo‘lgan pozitsiyada edi,

```
INSERATION-SORT (A)
1 for j=2 to A.Length
2 key= A[j]
3 //[j] ni tartiblashtirilgan tartibda [1..j-1] ga joylashtiring.
4 i=j-1
5 while i>0 and A[i] >key
6 A[i+j] =A[i]
7 i=i-1
8 A[i+1]=key
```

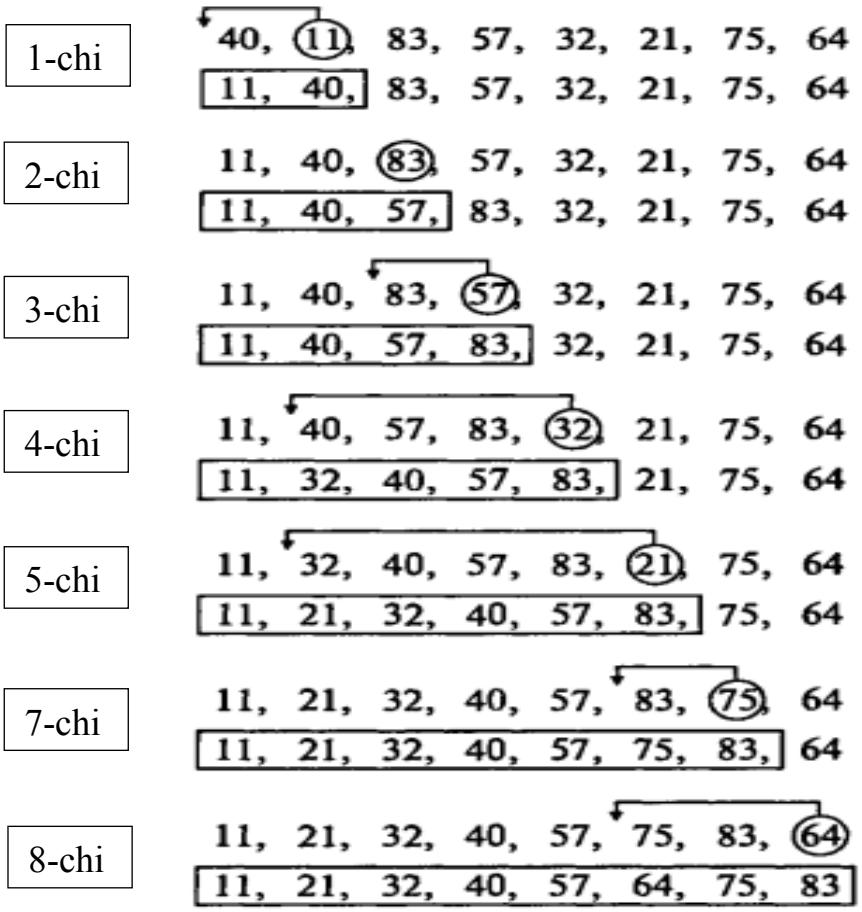
3-chizma

Ushbu usulda tartiblanmagan elementlarning ketma-ketligidan navbatma-navbat har bir oldingi tartiblangan element bilan taqqoslanib tanlanadi va mos o‘ringa joylashtiriladi.

Joylashlashtirib saralashni berilgan tartiblanmagan elementlar ketma-ketligi misolida ko‘ramiz:

$$\{40, 11, 83, 57, 32, 21, 75, 64\}$$

Saralash jarayoni 4-chizmada har bir bosqichda tahlil qilinayotgan element aylanaga olib tasvirlangan. Yuqoriga strelka orqali tahlil qilinayotgan element kelib joylashgan joy belgilangan, ramkaga ketma-ketlikning tartiblangan qismi olingan.[7]



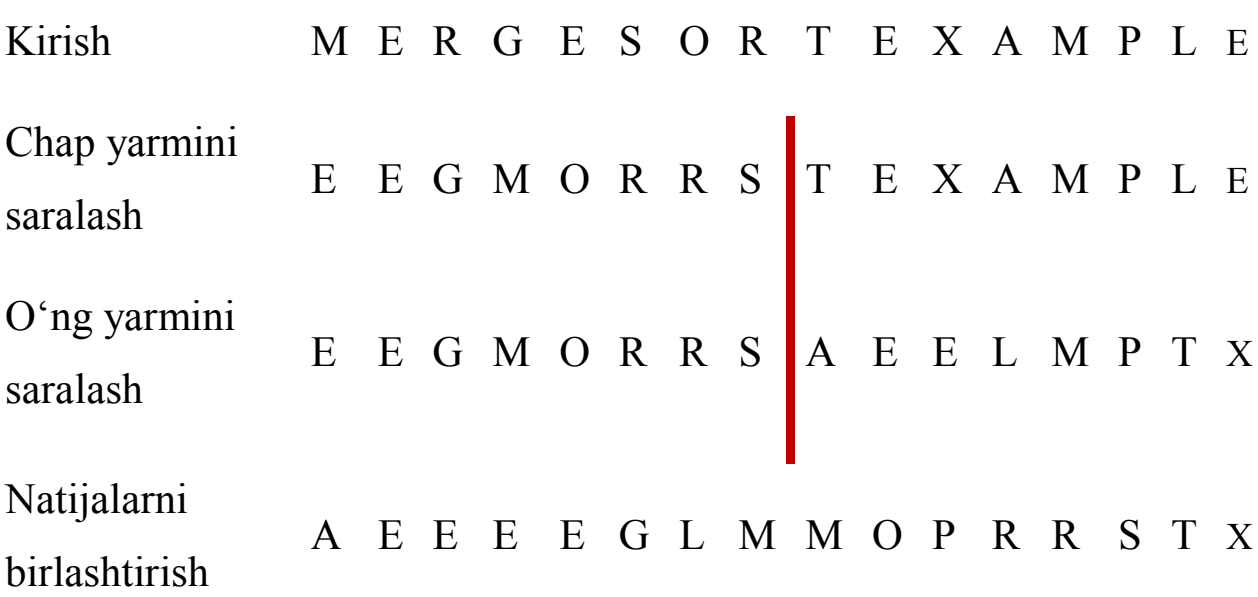
4-chizma. Joylashtirib saralash

[7] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 68-69 сс.

Birinchi bosqichda ikkita boshlang‘ich element taqqoslanadi. Ikkinchi element birinchi elementdan kichik bo‘lganligi tufayli, u bitta pozitsiya o‘ngga siljiydigan birinchi element joyiga kelib joylashadi. Ketma-ketlikning qolgan qismi esa o‘zgarmasdan qoladi.

Ikkinchi bosqichda tartiblanmagan ketma-ketlikdan element tanlanadi va avvalgi ikkita tariblangan elementlar bilan taqqoslanadi. U element avvalgi elementdan kattaligi tufayli, o‘z joyida qoladi. So‘ngra, to‘rtinchi, beshinchi va keyingi elementlar butun ro‘yxat (yettinchi bosqichda joyga ega bo‘lgunicha) tartiblanmaguncha tahlil qilinadi.

QO‘SHISH USULIDA SARALASH



Mergesort haqida

[8]Ushbu usul juda oddiy qo‘shish amaliga asoslanib, ikkita tartiblangan massivlarni bitta yagona tartiblangan massiv ko‘rinishida tasvirlaydi. Ushbu amal qo‘shish usulida saralash nomli saralashning oddiy rekursiv metodiga olib kelib, saralash uchun massivni teng ikkiga bo‘lib, hosil bo‘lgan yarimtaliklarni (rekursiv) saralash

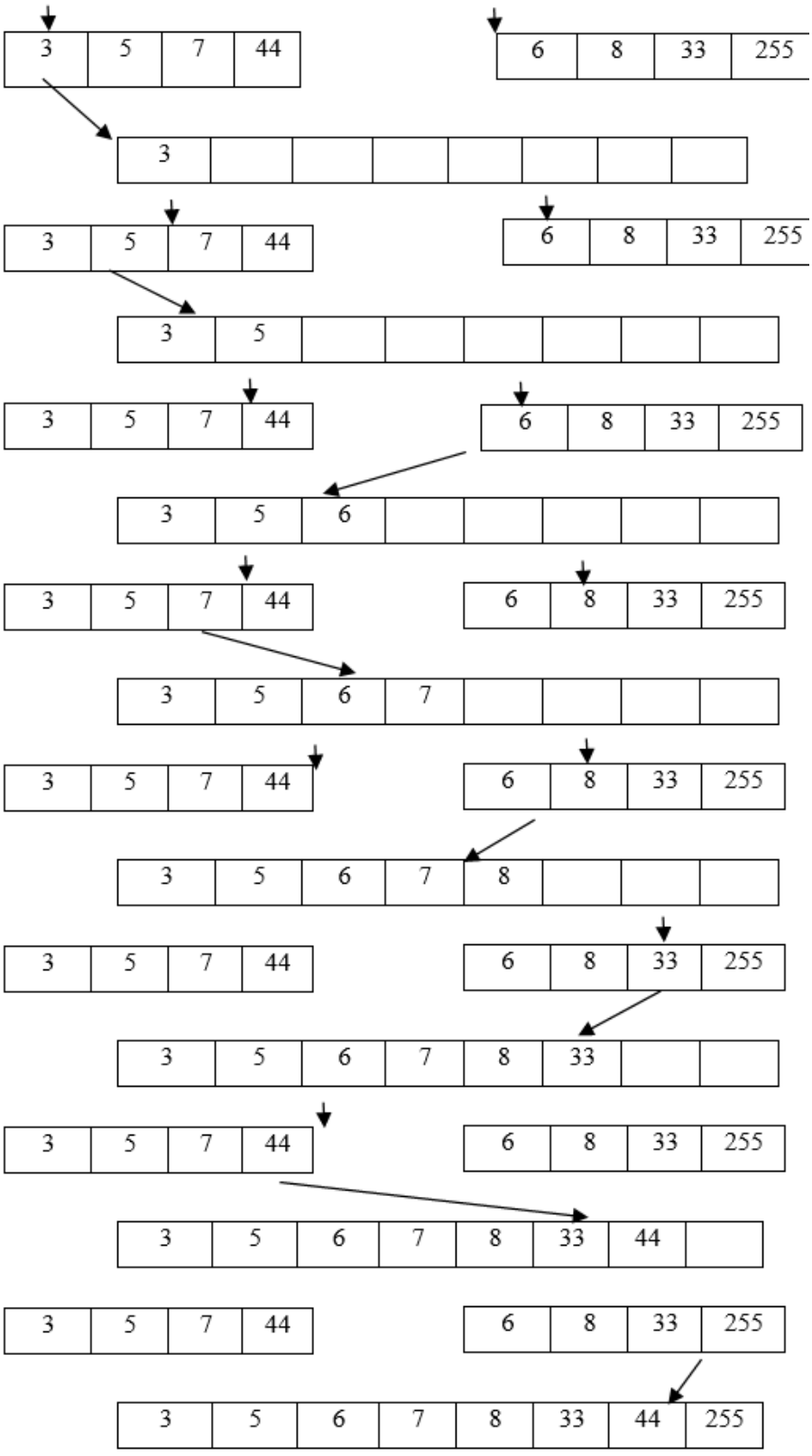
[8]Robert Sedgewick and Kevin Wayne. Algorithms. FOURTH EDITION. Princeton University. First printing, March 2011. Pp 270 .

va natijalarni qo‘shish kerak bo‘ladi. Qo‘shish usulida saralashning eng qiziq tomoni shundan iboratki, N elementdan iborat ixtiyoriy massivni NlogN prorotsional garantirlangan vaqtda saralashidir.

Joylashtirish usulida saralashning turlaridan biri fon Neyman usuli hisoblanadi. Ushbu masalani yechish algoritmi “fon Neyman saralash usuli” nomi bilan tanilgan yoki qo‘shib saralash usuli deb nomlanadi. Ushbu usul quyidagidan iborat: avval ikkala massivning birinchi elementlari tahlil qilinadi. Eng kichik element yangi massivga yozilib boriladi. Ketma-ketlikning qolgan elementlari boshqa massiv elementlari bilan taqqoslanadi. Har bir taqqoslashdan keyin yangi massivga eng kichik element borib tushadi. Jarayon massivlarning birida elementlarning kamayishigacha davom etadi. Shundan so‘ng, boshqa massivning qoldig‘i yangi massivga yoziladi. Hosil qilingan yangi massiv dastlabki massiv singari shu usulda tartiblangan bo‘ladi.

Ikkita o‘sish tartibida tartiblangan $p[1], p[2], \dots, p[n]$ va $q[1], q[2], \dots, q[n]$ massivlar bo‘lsin va $\boldsymbol{p}$ va $\boldsymbol{q}$ massivlar qiymatlari bilan o‘sish tartibida to‘ldirilishi kerak bo‘lgan $r[1], r[2], \dots, r[2n]$ ko‘rinishidagi bo‘sh massiv mavjud bo‘lsin. Qo‘shish uchun quyidagi amallar bajariladi: $p[1]$ va $q[1]$ lar taqqoslanadi va eng kichik qiymat $r[1]$ ga yoziladi. Ushbu qiymat $p[1]$ deb faraz qilamiz. U holda $p[2]$ bilan $q[1]$ taqqoslanadi va eng kichik qiymat $r[2]$ ga yoziladi. Ushbu qiymat $q[1]$ deb faraz qilamiz. U holda keyingi qadamda $p[2]$ va $q[2]$ lar taqqoslanadi, jarayon biron-bir massiv chegarasiga yetib borilgunigacha davom ettiriladi. U holda boshqa massivning qoldig‘i $\boldsymbol{r}$ massiv oxiriga yozib qo‘yiladi.

4-chizmada massivlarni qo‘shish usuli keltirilgan. Joylashtirish tartibida saralash usulining murakkabligi $O(n^2)$ ga teng bo‘ladi.



5-chizma. Qo‘shib saralash usuli[9]

[9] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 69-70 сс.

Bu usulda massiv ikkita massivlargra bo‘linib, har bir massiv alohida saralanib bir massivga qo‘shiladi.

```
Var A,b : array[1..1000] of integer;
N, i, j, p : integer;
Min, Max : integer;
Procedure Sliv(p,q : integer);
Var r,i,j,k : integer;
Begin
r:=(p+q) div 2; i:=p; j:=r+1;
for k:=p to q do
if (i<=r) and ((j>q) or (a[i]<a[j])) then
begin
b[k]:=a[i]; i:=i+1; end else
begin
b[k]:=a[j]; j:=j+1;
end ;
for k:=p to q do
a[k]:=b[k];
End;
Procedure Sort(p,q : integer);
Begin
if p<q then
begin
Sort(p,(p+q) div 2); Sort((p+q) div 2 + 1,q); Sliv(p,q);
end;
End;
Begin
readln(n); randomize;
for i:=1 to n do
begin a[i]:=random(120); write(a[i],' '); end;
writeln; Sort(1,N);
for i:=1 to n do
write(a[i],' ');
readln;
End.
```


```
procedure TForm1.Button1Click(Sender: TObjest);
var i1,i2,i3,i4,i5, n,s,k:integer;
var wordapp, dos:Variant;
begin
wordapp:=CreateOleObjest('Word.Applisation');
wordApp.Visible:=true;
dos:=WordApp.Dosuments.ADD;
Wordapp.SELESTION.FONT.NAME:='sourier' ;
WordApp.Selestion.Font.Bold:=true;
n:=10; s:=0;
for i1 := 1 to n do
for i2 := i1 + 1 to n do
for i3 := i2 + 1 to n do
for i4 := i3 + 1 to n do
for i5 := i4 + 1 to n do
begin
WordApp.Selestion.TypeText(inttostr(i1)+' '+inttostr(i2)+'
'+inttostr(i3)+' '+IntToStr(i4)+' '+IntToStr(i5)+#13);
s:=s+1;
end;
WordApp.Selestion.TypeText(inttostr(s));
ShowMessage('tamom');
end;
end.
```

Dastur natijasi:

1 2 3 4 5
1 2 3 4 6
….
5 7 8 9 10
6 7 8 9 10
252

Joyida abstrakt qo‘shib saralash

Ma’lumki, qo‘shish usulida saralash - ikkita alohida tartiblangan massiv **Comparable** obyektlarini uchinchi massivga qo‘shadi. Bu usulni amalga oshirish qiyinchilik tug‘dirmaydi: kerakli o‘lchamli chiquvchi massivni yarating, so‘ngra ketma-ket tarzda kiruvchi ikkita massivlardan eng kichik qolgan elementni tanlang va uni chiquvchi massivga qo‘shing.

Ammo katta massivni saralashda ko‘p marta qo‘shish amalini bajarishga to‘g‘ri keladi. Saralashni joyida saralash usuli ancha qulayroqdir, sababi dastlab birinchi yarimtalik joyida saralanadi, keyin ikkinchi yarimtalik joyida saralanadi, so‘ng elementlarni massivga joylashtirib ushbu yarimtaliklarni qo‘shiladi.

Joyida abstrakt qo‘shib saralash usuli ancha samaralidir. Shuning uchun ham usulni ifodalashda a[lo..m id] va a[m id+1..hi] qism massivlarning natijalarini yagona tartiblangan a[lo..hi] massivga joylashtiruvchi **merge**(a, lo, mid, hi) signaturasidan foydalanamiz. [10]

Joyida mavhum birlashish

```
Public static void  merge (Comparable [] a, int lo, int mid, int hi)
{
// [mid .. o'rta] bilan [o‘rta +1 .. xi] bilan birlashtirish.
Int i=lo, j=mid+1;
for (int k=lo; k<=hi; k++) // A [lo] ni aux [lo .. hi] ga nusxalash.
aux[k]=a[k];
for (int k=lo; k<=hi; k++) // [lo ..hi] ga teskaridan qo‘shish.
if (i>mid)                          a[k]=aux[j++];
else if (j>hi)                       a[k]=aux[i++];
else if (less(aux[j], aux[i]))    a[k]=aux[j++];
else
}
```

Qo‘shish usulini bajarish uchun dastlab berilganlardan aux[ ] yordamchi massivga nusxa oladi, so‘ng ularni qayta a[ ] massivga qo‘shadi. Qo‘shish jarayonida (ikkinchi for sikli) to‘rtta variant bo‘lishi mumkin: chap yarimtalik tugasa (berilganlarni o‘ngdan olamiz), o‘ng yarimtalik tugasa (berilganlarni chapdan olamiz), o‘ng

[10] Robert Sedgewick and Kevin Wayne. Algorithms. FOURTH EDITION. Princeton University. First printing, March 2011. Pp 270-273.

yarimtalikning joriy kaliti chap yarimtaliknikidan kichik (o‘ngdan olamiz), o‘ng yarimtalikning joriy kaliti chap yarimtaliknikidan katta yoki teng bo‘lsa (chapdan olamiz).

	k	0	1	2	3	4	5	6	7	8	9
Kirish		E	E	G	M	R	A	C	E	R	T
Nusxa-lash		E	E	G	M	R	A	C	E	R	T
	0	A									
	1	A	C								
	2	A	C	E							
	3	A	C	E	E						
	4	A	C	E	E	E					
	5	A	C	E	E	E	G				
	6	A	C	E	E	E	G	M			
	7	A	C	E	E	E	G	M	R		
	8	A	C	E	E	E	G	M	R	R	
	9	A	C	E	E	E	G	M	R	R	T
Birlash-tirilgan natija		A	C	E	E	E	G	M	R	R	T

i	j
0	5
0	6
0	7
1	7
2	7
2	8
3	8
4	8
5	8
5	9
6	10

0	1	2	3	4	5	6	7	8	9
-	-	-	-	-	-	-	-	-	-
E	E	G	M	R	A	C	E	R	T
E	E	G	M	R	A	C	E	R	T
E	E	G	M	R		C	E	R	T
E	E	G	M	R			E	R	T
	E	G	M	R			E	R	T
		G	M	R			E	R	T
		G	M	R				R	T
			M	R				R	T
				R				R	T
									T

Abstraktnaya trassirovka sliyaniya na meste.

```
                                  sort(a, 0, 15)
Chap yarmini saralash                sort (a, 0, 7)
                                       sort(a, 0, 3)
                                         sort(a, 0, 1)
                                           merge(a, 0, 0, 1)
                                         sort(a, 2, 3)
                                           merge(a, 2, 2, 3)
                                         merge(a, 0, 1, 3)
                                       sort(a, 4, 7)
                                         sort(a, 4, 5)
                                           merge(a, 4, 4, 5)
                                         sort(a, 6, 7)
                                           merge(a, 6, 6, 7)
                                       merge(a, 4, 5, 7)
                                     merge(a, 0, 3, 7)
O’ng yarmini saralash              sort(a, 8, 15)
                                     sort(a, 8, 11)
                                       sort(a, 8, 9)
                                         merge(a, 8, 8, 9)
                                       sort(a, 10, 11)
                                         merge(a, 10, 10, 11)
                                       merge(a, 8, 9, 11)
                                   sort(a, 12, 15)
                                     sort(a, 12, 13)
                                       merge(a, 12, 12, 13)
                                     sort(a, 14, 15)
                                       merge(a, 14, 14, 15)
                                     merge(a, 12, 13, 15)
                                   merge(a, 8, 11, 15)
Birlashtirilgan natija             merge(a, 0, 7, 15)
```

Top-down mergesort call trace

Сверхвысокая слитная трассировка вызовов.

Yuqoridan pastga qo‘shib saralash

Berilgan quyidagi rekursiv kod shuni ko‘rsatadiki, ikkita qism massivlarni saralab, ularni qo‘shish butun massivni saralaydi. Biz ikkita tartiblangan a[lo..hi] qism massivni ikkita a[lo..m id] va a[m id+1..hi] qismga bo‘lamiz va saralashdan so‘ng, natija olish uchun ularni qo‘shamiz.

Algorithm 2.4 Top-down mergesort

```
public class Merge
{
    private static Comparable [] aux;
    public static void sort (Comparable [] a)
    {
        aux=new Comparable [a.length];
        sort (a, 0, a. length-1);
}
  private static void sort (Comparable [] a, int lo, int hi)
{   // [lo .. hi] ni saralash
   if (hi<=lo0 return;
    int mid=lo+ (hi-lo)/2;
    sort(a, lo, mid);      // Chap yarmini saralash.
   sort (a, mid+1, hi);     // O‘ng yarmini saralash.
   merge (a, lo, mid, hi); // Birlashtirilgan natija.
     }
}
```

	a[]															
lo hi	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a,0,0,1)	E	M	R	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a,2,2,3)	E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a,0,1,3)	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a,4,4,5)	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a,6,6,7)	E	G	G	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a,4,5,7)	E	G	M	R	E	O	R	R	T	E	X	A	M	P	L	E
merge(a,0,3,7)	E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E
merge(a,8,8,9)	E	E	G	M	O	R	R	S	E	T	X	A	M	P	L	E
merge(a,10,10,11)	E	E	G	M	O	R	R	S	E	T	A	X	M	P	L	E
merge(a,8,9,11)	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E
merge(a,12,12,13)	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E
merge(a,14,14,15)	E	E	G	G	O	R	R	S	A	E	T	X	M	P	E	L
merge(a,12,13,15)	E	E	G	M	O	R	R	S	A	E	T	X	E	L	M	P
merge(a,8,11,15)	E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X
merge(a,0,7,15)	A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

Yuqoridan pastga birlashtirish uchun birlashtirish natijalarini kuzatish

1-misol.

$A = \{8, 12, 3, 7, 19, 11, 4, 16\}$ elementlardan iborat massiv berilgan. To‘siq element sifatida **(7)** mazkaziy elementni olamiz. Kerakli joylashtirishlarni amalga oshirb, $(4, 3)\ 7\ (12, 19, 11, 8, 16)$ hosil qilamiz; endi **7** elementi o‘z o‘rnida turibdi. Saralashni davom ettiramiz:

Chap tomondagi qism:	O‘ng tomondagi qism
(3) 4 7 (12 19 11 8 16)	3 4 7 (8) 11 (19 12 16)
3 4 7 (12 19 11 8 16)	3 4 7 8 11 (19 12 16)
3 4 7 8 11 12 (19 16)	
3 4 7 8 11 12 (16) 19	
3 4 7 8 11 12 16 19	

Dastur kodi:

```
Procedure Quicksort (m, t: Integer);  {Tezkor saralash
Quicksort(1,N)}
Var i , j , x , w: Integer;
Begin
i:=m; j:=t; x:=A[ (m+t) Div 2]; {To‘siq elementni aniqlash}
Repeat
While A[i]<x Do Inc(i);
While A[j]>x Do Dec(j);
If i<=j Then Begin w:=A[i]; A[i]:=A[j]; A[jl:=w; Inc(i);
Dec(j) End
{Agar katta bo‘lsa joylarni almashtiramiz}
Until i>j;
If m<j Then Quicksort(m, j); {Rekkursiya}
If i<t Then Quicksort(i,t),
End;
```

4-§. Almashish usulida saralash, saralashning sheyker usuli

Almashish usulida saralash

Almashish usulida saralash- bu ro‘yxat elementlari ketma-ket o‘zaro taqqoslanadi va avvalgi element keyingi elementdan katta bo‘lgan holda joyini almashtiradi. Masalan, ro‘yxatni standart almashish usulida saralash talab qilinsin:

{40, 11, 83, 57, 32, 21, 75, 64}.

Almashtiriladigan elementlarni ↑ ↑ strelkali kvadrat qavslar orqali, taqqoslanayotgan elementlarni esa ⌊ ⌋ kvadrat qavslar orqali belgilaymiz. Saralashning birinchi bosqichi 6-chizmada, ikkinchi bosqichi esa 7-chizmada ko‘rsatilgan.

Har bir ro‘yxatni ko‘rishdan so‘ng, oxiridan boshlab barcha elementlar o‘zlarining oxirgi pozitsiyalarini egallashini ko‘rish qiyin emas. [11]

Dastlabki ro‘yhat	40	11	83	57	32	21	75	64
Birinchi ko‘rinish	11	40 40	83 57	83 32	83 21	83 75	83 64	83
Hosil qilingan ro‘yhat	11	40	57	32	21	75	64	83

6-chizma. Almashish usulida saralash (birinchi ko‘rinish)

Dastlabki ro‘yhat	11	40	57	32	21	75	64
Ikkinchi ko‘rinish	11	40 40	57 32	57 21	57 57	75 64	75
Hosil qilingan ro‘yhat	11	40	32	21	57	64	7

7-chizma. Almashish usulida saralash (birinchi ko‘rinish)

[11] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 71-с.

Har bir keyingi ko‘rib chiqish eng katta element topgan pozitsiyani yo‘qotib borib, ro‘yhatni qisqartirib boradi. Birinchi ko‘rib chiqishdan so‘ng oxirgi pozitsiyada 83 ga teng eng katta element qoldi.

Ikkinchi ko‘rib chiqishdan eng katta element 75 ga teng ekenligini kelib chiqadi (7-chizmaga qarang).

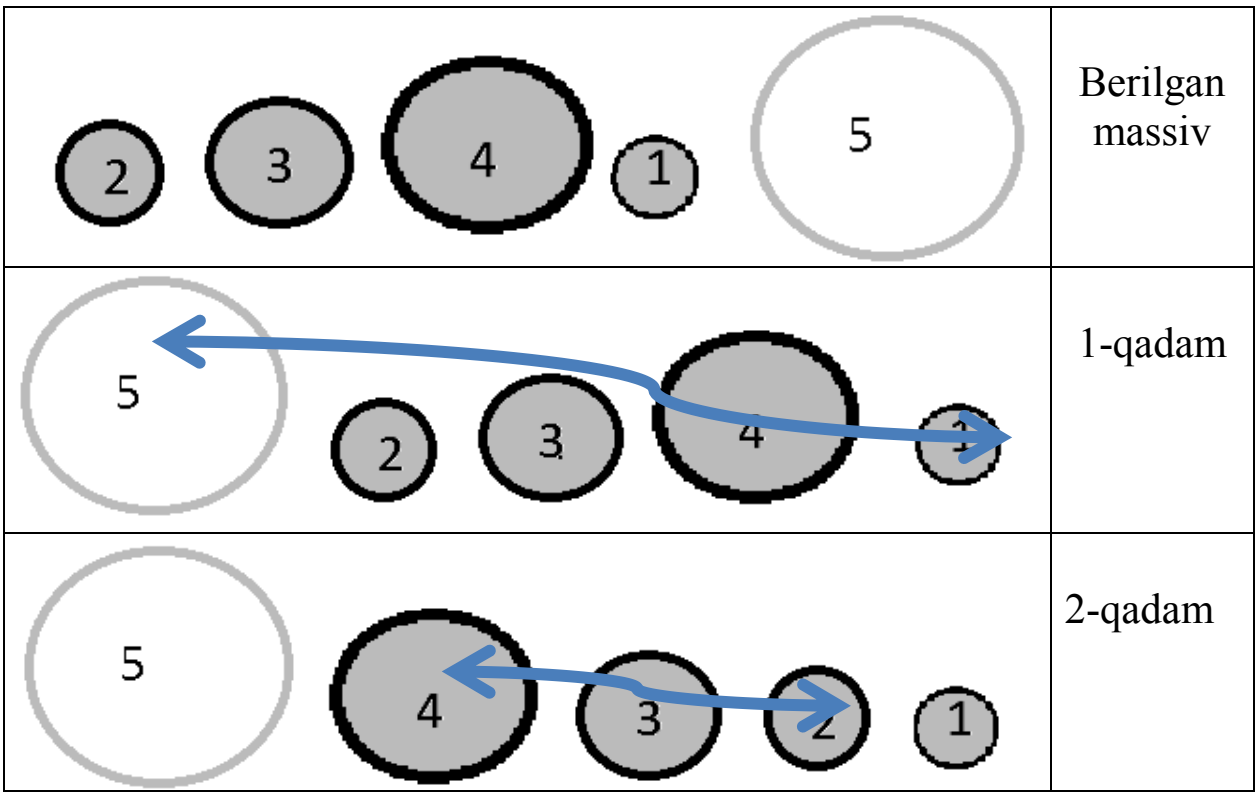
Saralash jarayoni oxirgi ro‘yxatning barcha elementlari shakllangunicha davom etadi, aks holda Ayverson sharti bajarilmaydi.

Ayverson sharti: agar saralash jarayonida elementlarni taqqoslashda bir marotaba bo‘lsa ham o‘zgartirish bo‘lmasa, u holda to‘plam tartiblangan hisoblanadi (Ayverson sharti qadam *d=1* bo‘lganda bajariladi).

Sheker usulida saralash

Standart almashtirish saralash usulidan biri sheker yoki chelnochnaya saralash hisoblanadi. Bu yerda elementlar o‘zaro saralanib boriladi. Bu bilan birinchi o‘tish chapdan o‘ngga, ikkinchisi esa o‘ngdan chapga bo‘ladi va hokazo. Bir so‘z bilan aytganda, ro‘yxat elementlarning yo‘nalishi o‘zgaradi.

Standart almashish usulining murakkabligi- $O(n^2)$.[12]



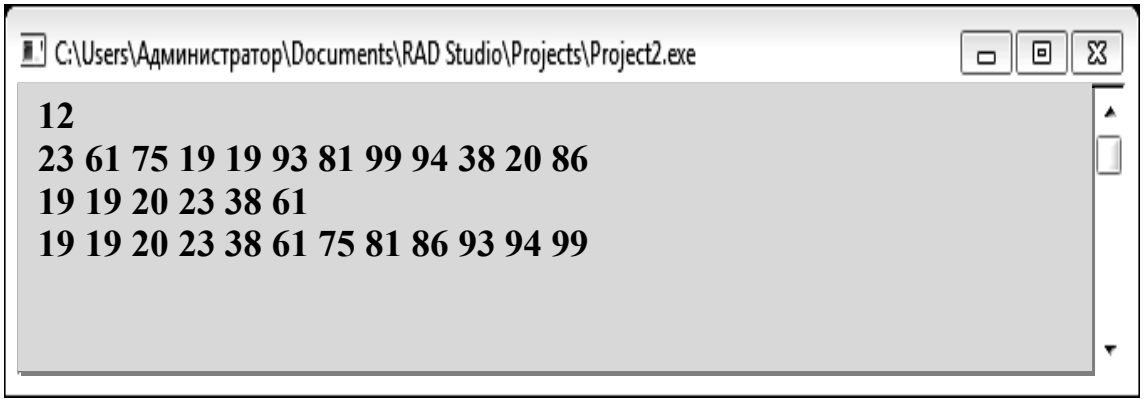
[12] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 71-72 сс.

Dastur kodi:

```
Var A : array[1..1000] of integer;
N,i,j,p : integer;
Min, Max : integer;
Begin
readln(n); randomize;
for i:=1 to n do
begin
a[i]:=random(120);
write(a[i],' ');
end;
writeln;
for i:=1 to n div 2 do
begin
if A[i]>A[i+1] then
begin
Min:=i+1;
Max:=i;
end
else
begin
Min:=i;
Max:=i+1;
end;
for j:=i+2 to n-i+1 do
if A[j]>A[Max] then
Max:=j
else
if A[j]<A[Min] then Min:=j;
P:=A[i];
A[i]:=A[min];
A[min]:=P;
if max=i then
max:=min;
P:=A[N-i+1];
A[N-i+1]:=A[max];
```

```
A[max]:=P; write(a[i],' ');
end;
writeln;
for i:=1 to n do
write(a[i],' ');
readln;
End.
```

Olinadigan natija:



Pufakcha (qalqib chiqish) usuli.

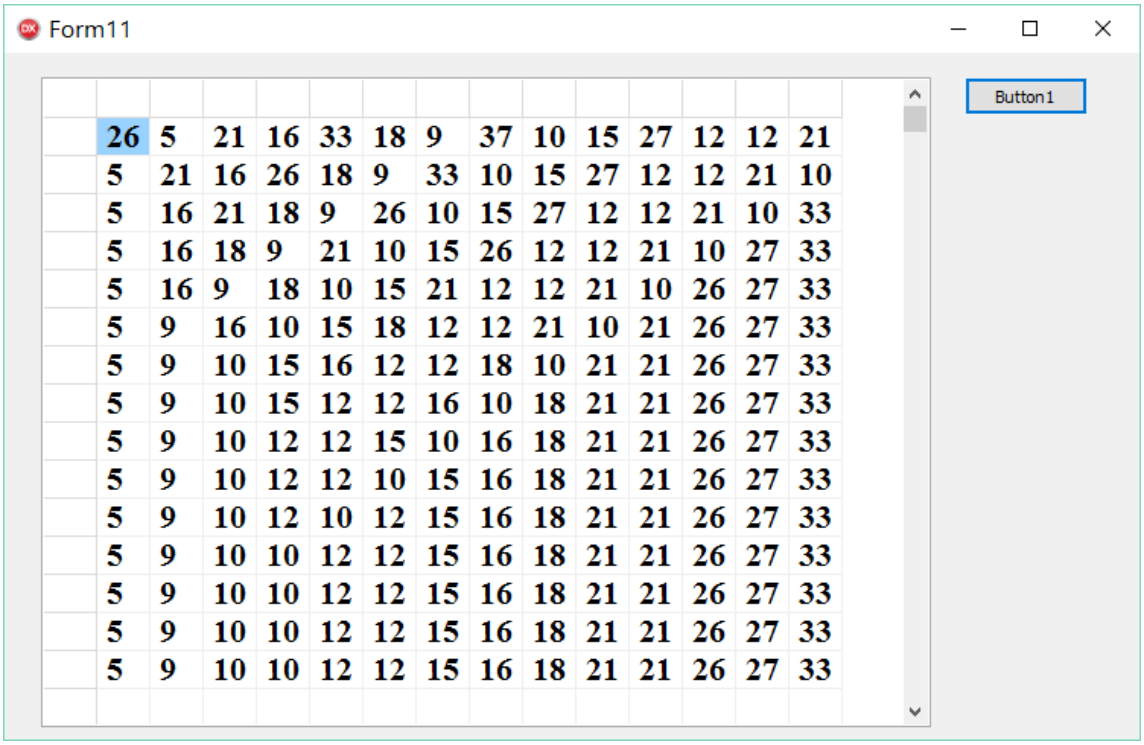
A[0], A[1],.., A[N] massivning elementlari berilgan bo‘lsin. Ketma-ket ravishda A[0], va A[1], A[1], va A[2] elementlar o‘zaro taqqoslanib agar A[i]>a[i+1] bo‘lsa, ular o‘zaro o‘rin almashadilar. Ikkinchi qadamda shu holat A[N-1] gacha davom ettiriladi va hokazo. Bu usul hubobcha(qalqib chiqish) usuli deyilishiga sabab har safar hajmi katta «sharcha» element qolganlarini ortda qoldirib yuzaga “qalqib” chiqadi.

Ushbu algoritmni Delphi dasturlash tilida keltirib o‘tamiz.

```
procedure TForm11.Button1Click(Sender: TObject);
var A:ARRAY[1..15] OF INTEGER; I,D,K,Z,QAT:INTEGER;
begin
RANDOMIZE; QAT:=2;
StringGrid1.DefaultColWidth:=34; StringGrid1.RowCount:=15;
StringGrid1.ColCount:=15; StringGrid1.Font.Size:=16;
for I := 1 to 15 do
BEGIN
A[I]:=RANDOM(39);
StringGrid1.Font.Name:='Times new Roman';
```

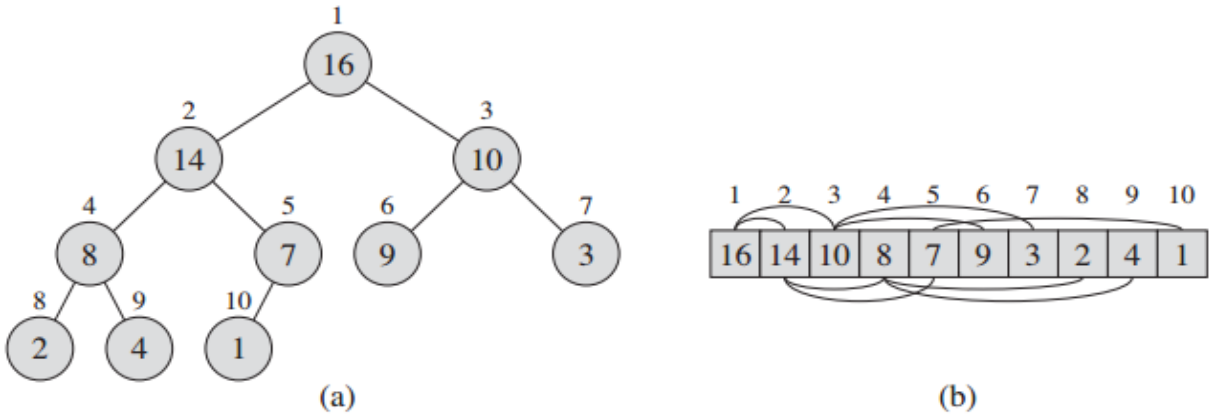
```
StringGrid1.Cells[I,1]:=FloatToStr(A[I]);
END;
K:=14;
while (K>=1) do
BEGIN
I:=1;
while (I<=K) do
BEGIN
if A[I]>A[I+1] then
BEGIN D:=A[I]; A[I]:=A[I+1]; A[I+1]:=D; END;
I:= I+1;
END;
K:=K-1;
for Z := 1 to 15 do
BEGIN StringGrid1.Cells[Z,QAT]:=FloatToStr(A[Z]); END;
QAT:=QAT+1;
StringGrid1.RowCount:=StringGrid1.RowCount+1;
END;
end;
end.
```

Dastur natijasi:



Piramida usulida saralash

[13](Binar) piramidaning tuzilishi butun binar daraxt sifatida qaralishi mumkin bo‘lgan obyekt-massivni ifodalaydi (8-chizmada keltirilgan).



Shakl 6.1. (A) shishirilgan (b) massiv sifatida ko'rib chiqilgan maksimal birikma. Daraxtning har bir tugunidagi doira ichidagi raqam - bu tugunda saqlangan bu qiymat qatorda mos keladigan indeksdir. Ota-bola va ota-ona o‘rtasidagi munosabatni ko‘rsatuvchi chiziqlar. ota-onalar doimo o‘z farzandlari va farzandlari. Daraxt balandligi bor; Indeks 4 (8 qiymati bilan) bo‘lgan tugunning birining balandligi bor.

8-chizma.

```
PARENT(i)
1  return ⌊i/2⌋
LEFT(i)
1  return 2i
RIGHT(i)
1  return 2i + 1
```

Ushbu daraxtning har bir tuguni massiv elementlariga mos keladi. Daraxt chapdan o‘ng-ga qarab to‘lib boradigan mavjud bo‘lishi mumkin bo‘lgan eng pastkidan tashqari barcha sathlar bo‘yicha to‘ldirilgan. piramidani tasvir-lovchi ***A*** massiv ikkita atributlarga ega obyekt hisoblanadi: odatda massiv elementlar sonini be-radigan ***A.length*** va piramidaning nechta elementlari ***A*** massivda joylashganini ko‘rsatuvchi ***A.heap-size.*** Shuningdek, $A[1..A.length]$ ega faqat $0 \le A.heap-size \le A.length$ oraliqagi $A[1..A.heap-size]$ qism massiv elementlari piramidaning to‘g‘ri elementlari

[13] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 152-bet

hisoblanadigan ba'zi bir sonlarga ega bo'lishi mumkin. $A[1]$ daraxtning ildizi bo'ladi, berilgan ***i*** indeks tuguni uchun uning ota-ona indekslarini chap va o'ng tugunlar orqali oson hisoblash mumkin.

Binar piramidalarni ikki turga ajratishadi: kamaymaydigan va o'smaydigan. Piramidalarning asosiy xususiyatlaridan biri hisoblanadigan piramidalar xossalari (heap property) qoniqtiradigan ikkala ko'rinishdagi qiymatlar tugunlarda joylashadi. **O'smovchi piramidalar xossalari (max-heap property)** dan biri har bir ***i*** indeksli ildiz tugun uchun quyidagi tengsizlik bajariladi:

$$A[PARENT(i)] \geq A[i].$$

Shu tarzda, o'smovchi piramidaning eng katta elementi daraxt ildizida qiymatlari esa qism daraxt tugunlarida joylashgan bo'ladi. Kamayvovchi piramida prinsipi esa mutlaqo teskaridir. **Kamayvovchi piramida xossasi (min-heap property)** da har bir ***i*** indeksli ildiz tugun uchun quyidagi tengsizlik bajariladi:

$$A[PARENT(i)] \leq A[i].$$

Shuning uchun ham piramidaning eng kichik elementi ildizda joylashgan bo'ladi.

Piramida xossasining saqlanishi

O'smovchi piramida xossalarini saqlab turish uchun **MAX-HEAPIFY** protsedurasini chaqiramiz. Uning kiritilganlari ***A*** massiv va shu massivdagi ***i*** indeks hisoblanadi. **MAX-HEAPIFY** prosedurasini chaqirishda ***LEFT(i) va RIGHT(i)*** ildizlardan iborat binar daraxt o'smovchi piramidani ifodalashi taxmin qilinadi, ammo $A[i]$ farzand tugunlardan kichik bo'lishi ham mumkin.

```
                    Max-Heapify (A,i)
1   l=Left(i)
2   r=Right(i)
3   if l <= A.heap-size and A [l] > A[i]
4   largest =l
5   else largest=i
6   if r <= A.heap-size and a[r]> A{largest]
7   largest =r
8   if largest ≠i
9   exchange A[i] with A[largest]
10  Max-Heapify (A,largest)
```

Chizmada **MAX-HEAPIFY** protsedurasi ko‘rsatilgan. Har bir qadamda $A[i]$, $A[LEFT(i)]$ va $A[RIGHT(i)]$ elementlaridan eng katta element aniqlanadi va uning indeksi ***largest*** o‘zgaruvchida saqlanadi. Agar $A[i]$ eng katta bo‘lsa, u holda ***i*** ildizdan iborat qismdaraxt o‘smovchi piramidani tasvirlaydi va protsedura to‘xtatiladi. Agar $A[LEFT(i)]$, $A[RIGHT(i)]$ lardan biri eng katta bo‘lsa, u holda prosedura $A[i]$ **ni** $A[largest]$ bilan almashtirilib, ***i*** tugun va uning farzand tugunlari uchun o‘smovchi piramida xossasi bajariladi. Biroq $A[i]$ ning dastlabki qiymati tugunning ***largest*** indeksida ekanligi kelib chiqdi, bu esa ***largest*** ildizdan iborat qism daraxt o‘zining o‘smovchi piramida xossalarini buzishiga olib keladi. Shuning uchun ham ushbu daraxt uchun **MAX-HEAPIFY** protsedurasini rekursiv chaqirish kerak bo‘ladi.

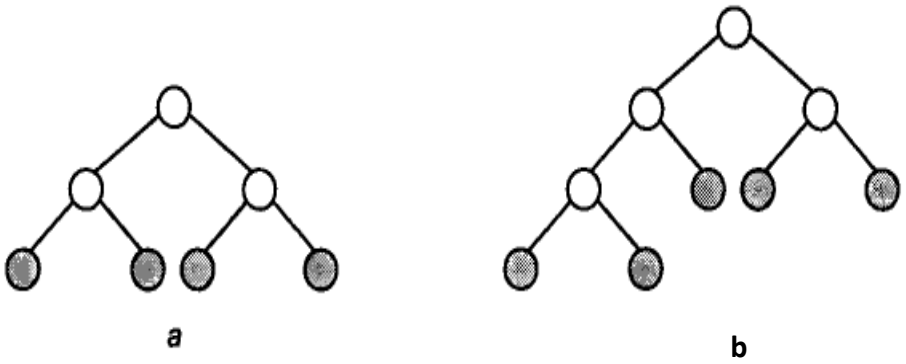
Piramida saralash usuli piramidali daraxtni qurish bilan ifodalanadi.

Piramidali daraxt – bu binar daraxt bo‘lib, uchta xossaga egadir:

1) Har bir triad uchida eng katta element joylashadi.

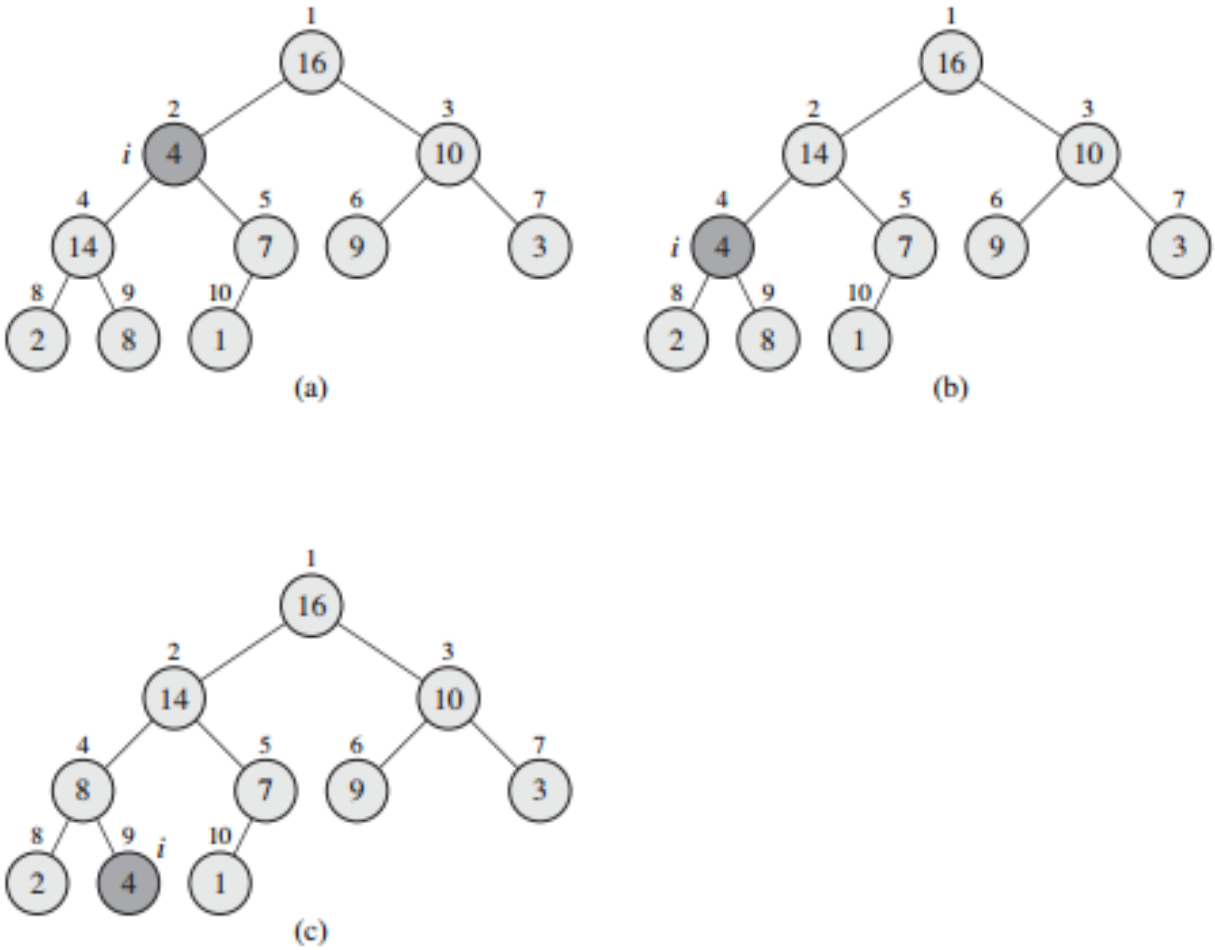
2) Binar daraxt barglari bir sathda yoki ikkalasi qo‘shni joylashadi.

3) Pastki sath barglari baland barglar sathidan chaproqda joylashadi. [14]



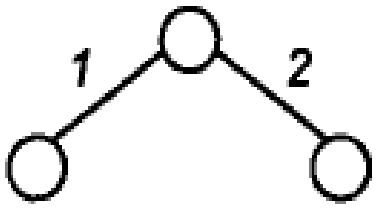
9-chizma. Binar daraxt:
a-bir sathdagi barglar; b-qo‘shni sathlardagi barglar

[14] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 76-сс.



6.2-shakl. Maks-heapify (A, 2) ta’sir qiladi, bu erda A.heap-size = 10. (a) Boshlang‘ich konfiguratsiya, A [2] tugmachasida i = 2, har ikkala boladan kattaroq bo‘lgani uchun maksimal birikma xususiyatini buzadi. Maks-birikma xususiyati 4-tugma uchun tiklanadi. Max-heapify (A, 4) o‘z-o‘zidan yineluvchi chaqiruvi i = 4 ga ega. A [4] ni [9] bilan o‘zgartirib bo‘lgach, (s) da ko'rsatilgandek, tugun 4 ga o‘rnatildi va Max-Heapify (A, 9) takrorlangan chaqiruvi ma’lumotlar tuzilmasiga boshqa hech qanday o‘zgartirish bermaydi.

Aylantirish jarayonida triad elementlari ikki marotaba taqqos–lanadi (10-chizma), shu bilan birga eng katta element yuqoriga, eng kichik element esa pastga o‘tadi.



10-chizma. Triad elementlarini taqqoslash:
1- birinchi taqqoslash; 2- ikkinchi taqqoslash;[15]

$n = A.length$ bo‘lgan $A[1..n]$ kirish massivida o‘smovchi piramidani qurish uchun piramida usulida saralash algoritmi **BUILD-MAX-HEAP** protsedurasini chaqirishdan boshlanadi. Massivning eng katta elementi $A[1]$ bo‘lganligi sababli, uni $A[n]$ elementi o‘rni bilan almashtirib, saralangan massivning aniq va oxirgi pozitsiyaga qo‘yish mumkin.

Piramidali saralash (A)

1 Build-Max-Heap (A)

2 **for** *i A.length* **downto** *2*

3 exchange *A[1]* with *A[i]*

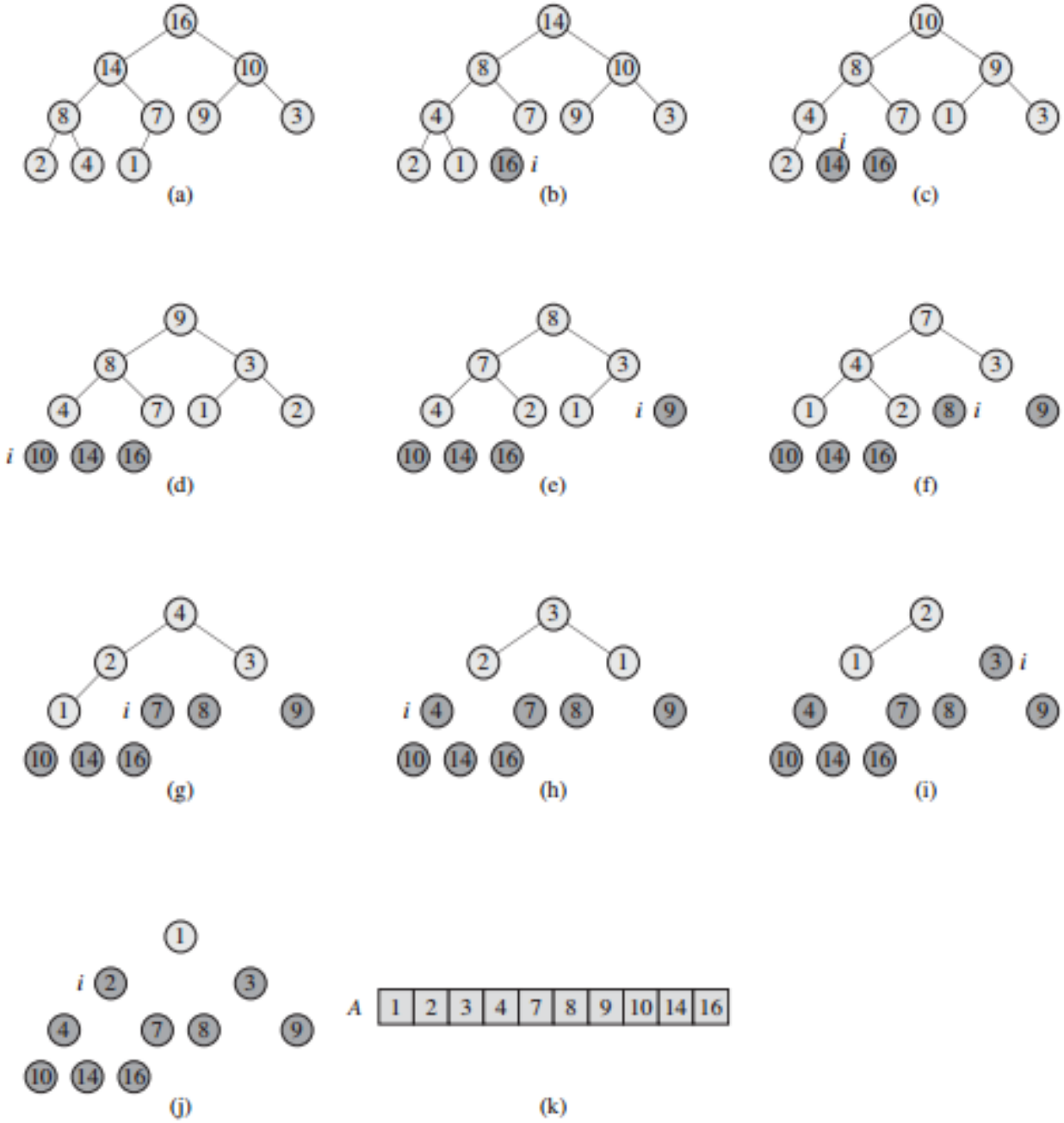
4 *A.heap-size* = *A.heap-size*-1

5 Max-Heapify (A,1)

1-qatorda boshlang‘ich o‘smovchi piramida qurilgandan so‘ng **HEAPSORT** protsedurasini ishlashi keltirilgan. Chizmada 2-5 qatorlarda birinchi va keyingi har bir **for** silk interatsiyasidan o‘smovchi piramida ko‘rsatilgan.

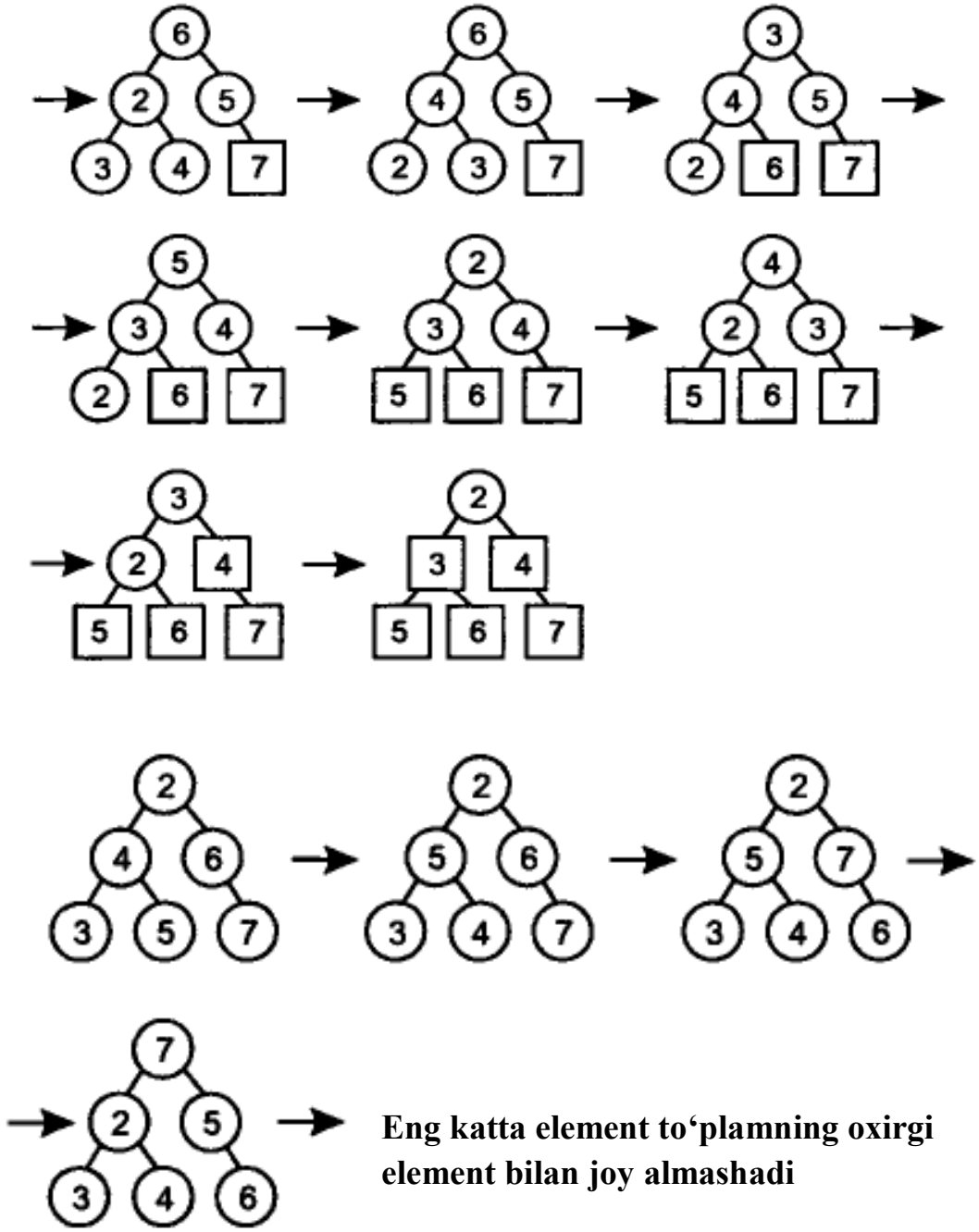
BUILD-MAX-HEAP protsedurasini chaqirish $O(n)$ vaqt talab etishi hamda **MAX-HEAPIFY** protsedurasining har bir ***n-1*** chaqiruv vaqti $O(lgn)$ ga teng ekanligidan, **HEAPSORT** protsedurasining ish vaqti $O(nlgn)$ ga teng bo‘ladi.

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 159-160 pp



6.4-shakl. Heapsort ishi (a). 5-sathdagi Maks-Heapiy chaqiruvidan so‘ng darhol ma’lumotlarning tuzilishi, shu vaqtning o‘zida I qiymatini ko‘rsatib beradi. Uyingizda (k), faqat biroz soyali tugunlar qoladi. Olingan tartiblangan qator A

1-misol. Dastlabki $\{2, 4, 6, 3, 5, 7\}$ to‘plam berilgan. Piramida usulida saralash 6-chizmada ko‘rsatilgan.



11-chizma. Piramida usulida saralash[16]

Natijada tartiblangan $\{2, 3, 4, 5, 6, 7\}$ to‘plam hosil bo‘ladi.

[16] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 75-76 сс

5-§. Algoritmlar tahlili

Tahlil nima?

Algoritm tahlilini, qo‘yilgan masalani ushbu algoritm bilan yechish qancha vaqt talab qilishi deb tasavvur qilish mumkin.

Har bir qaralayotgan algorimtni N o‘lchovli boshlang‘ich ma’lumotlar massividagi masalalarning qanchalik tez yechilishi bilan baholaymiz.

Masalan, saralash algoritmi N ta qiymatdan iborat ro‘yxatni o‘sish tartibida joylashtirish uchun qancha taqqoslash talab qiladi yoki N*N o‘lchamli ikkita matritsani ko‘paytirishda qancha arifmetik amallar zarurligini hisoblash.

Bitta masalani turli algoritmlar bilan yechish mumkin. Algoritmlar tahlili bizga algoritmni tanlash uchun qurol bo‘ladi. To‘rtta qiymatdan eng kattasini tanlaydigan ikkita algoritmni qaraymiz:

largest = a
if b > largest then
largest = b
end if
return a
if s > largest then
largest = s end if if d > largest then
largest = d end if return largest
if a > b then if a > s then I f a > d then
return a else
return d end if else
if s > d then
return s else
return d end if end if else
if b > s then if b > d then
return b else
return d end if else
if s > d then
return s else
return d end if end if end if

Ko‘rinib turibdiki, qaralayotgan algoritmlarning har birida uchta taqqoslash bajariladi. Birinchi algoritmni o‘qish va tushunish oson, ammo kompyuterda bajarilish nuqtayi nazaridan ularning murakkablik darajalari teng. Bu ikki algoritm vaqt nuqtayi nazaridan teng, lekin birinchi algoritm largest nomli qo‘shimcha o‘zgaruvchi hisobiga ko‘proq xotira talab qiladi. Agarda son yoki belgilar taqqoslansa, ushbu qo‘shimcha o‘zgaruvchi katta ahamiyatga ega bo‘lmaydi, lekin boshqa turdagi ma’lumotlar bilan ishlaganda bu muhim ahamiyatga ega. Ko‘plab zamonaviy dasturlash tillari katta va murakkab obyektlarni yoki yozuvlarni taqqoslash operatorlarini aniqlash imkonini beradi. Bunday hollarda qo‘shimcha o‘zgaruvchilarni joylashtirish katta joy talab qiladi. Algoritmlarning effektivligini tahlil qilishda bizni birinchi navbatda vaqt masalasi qiziqtiradi, ammo xotira muhim rol o‘ynaydigan vaziyatda uni ham muhokama qilamiz.

Algoritmlaring turli xossalari bitta masalani yechuvchi ikki turdagi algoritmlarning effektivligini taqqoslash uchun xizmat qiladi.

Biz shuning uchun hech qachon matritsalarni ko‘paytirish algoritmi bilan saralash algoritmini emas, balki ikkita turli saralash algoritmlarini bir-biri bilan taqqoslaymiz.

Algoritm tahlilining natijasi – belgilangan algoritmning kompyuterdan qancha vaqt yoki takrorlash talab qilishini aniq hisoblovchi formula emas.

Bunday ma’lumot muhim emas, bu holatda kompyuter turi, u bitta yoki undan ortiq foydalanuvchi tomonidan ishlatilyaptimi, uning protsessori va chastotasi qanaqa, protsessor chipida komandalar to‘liqmi va kompilyator bajarilayotgan kodni qay darajada amalga oshirmoqda kabi tomonlarni nazarda tutishni kerak.

Bu shartlar algoritm bajarilish natijasida dasturning ishlash tezligiga ta’sir qiladi. Yuqoridagi shartlar hisobiga dasturni boshqa tez ishlaydigan kompyuterga o‘tkazilganda algoritm yaxshi ishlaganday bajarilishi tezroq amalga oshadi. Aslida esa unday emas, biz shuning uchun tahlilimizda kompyuterning imkoniyatlarini inobatga olmaymiz.

Oddiy va katta bo‘lmagan dasturlarda bajariladigan amallar sonini N ning funksiyasi ko‘rinishida aniq hisoblash mumkin.

Aksariyat holatlarda bunga zaruriyat qolmaydi.

Algoritm tomonidan bajariladigan jarayonlar borki, biz ularning hammasini hisoblab o‘tirmaymiz, buning sababi shundaki, hatto uning eng kichik sozlashi ham samaradorlikning sezilmas yaxshilanishiga olib keladi. Fayldagi turli belgilar sonini hisob-lovchi algoritmni qaraymiz. Bu masala yechimi uchun algoritmning taxminiy ko‘rinishi quyidagicha bo‘ladi:

for all 256 belgilarni do hisoblagichni nolga tenglash end for while agar faylda belgi qolsa do navbatdagi belgini ko‘rsat va hisoblagichni bittaga oshir end while do hisoblagichni nolga tenglashtirish end for while faylda belgi mavjud bo‘lsa do navbatdagi belgini ko‘rsat va hisoblagichni bittaga oshir end while

Ushbu algoritmni ko‘rib chiqamiz. U takrorlanish bajarilishida 256 ta o‘tish qiladi. Agar berilgan faylda N ta belgi bo‘lsa unda ikkinchi takrorlanishda *N* ta o‘tish qilinadi. «Bu qanday hisoblash?» degan savol tug‘iladi. **For** siklida avval sikl o‘zgaruvchisi bajariladi, keyin har bir o‘tishda uning sikl chegarasidan chiqmayotganligi tekshiriladi va o‘zgaruvchi qiymatini oshiradi. Bu esa sikl bajarilishida 257 ta yuklash bajariladi (biri sikl o‘zgaruvchisi, 256 tasi hisoblagich uchun), ya’ni 256 ta oshirish va 257 ta sikl chegarasidan chiqmaganligini tekshirish (bitta amal siklni to‘xtatish uchun qo‘shilgan). Ikkinchi siklda *N* + 1 marta shart tekshiriladi (+1 fayl bo‘sh bo‘lgandagi oxirgi tekshiruv), va *N* *h*isoblagichni oshirish. Jami amallar:

Oshirish	*N + 256*
Yuklash	*257*
Shartlarni tekshirish	*N + 258*

Shunday qilib, 500 belgidan iborat fayl berilsa algoritmda 1771 ta amal bajariladi, ulardan 770 tasi natija beradi (43%). Endi N ning qiymati oshganda nima bo‘lishini ko‘ramiz. Agar fayl 50 000 belgidan iborat bo‘lsa, unda algoritm 100 771 amal bajaradi, ularning 770 tasi natija uchun (jami amallar sonining 1% ini tashkil etadi). Yechimga qaratilgan amallar soni oshmayapti, lekin *N* katta bo‘lganda ularning foizi juda kam.

Endi boshqa tomoniga e’tibor qaratamiz. Kompyuterda ma’lumotlar bilan shunday ishlashga mo‘ljallanganki, katta hajmdagi ma’lumotlar blokini ko‘chirish va yuklash bir xil tezlikda amalga oshiriladi. Shuning uchun biz avval 16 ta hisoblagichga boshlang‘ich qiymat 0 ni yuklaymiz, keyin qolgan hisoblagichlarni to‘ldirish uchun shu blokdan 15 ta nusxa olamiz. Bu esa sikl bajarilish davomida tekshirishlar sonini 33 ga, yuklashlar sonini 33 va oshirishlar sonini 31 ga kamayishiga olib keladi. Demak amal bajarilishlar soni 770 dan 97 gacha kamaydi, ya’ni 87%. Agar erishilgan natijani 50000 belgidan iborat fayl ustida bajarsak, tejamkorlik 0,7% ni tashkil qiladi (100771 ta amal o‘rniga 100098 amal bajaramiz).

Agarda barcha amallarni sikldan foydalanmay 31 ta yuklashlar orqali bajarganimizda, vaqtni yanada tejagan bo‘lardik, ammo bu usul 0,07 foyda keltiradi. Ishimiz unumli bo‘lmaydi.

Ko‘rib turganimizdek, algoritmning bajarilish vaqti bilan bog‘liq barcha amallar befoyda. Tahlil tili bilan aytganda, boshlang‘ich ma’lumotlar hajmining ortishiga alohida e’tibor qaratish kerak.

Avvalgi ishlarda algoritmlarni tahlil qilishda algoritmlarni Tyuring mashinasida hisoblash aniqlangan. Tahlilda masalani yechish uchun zarur bo‘lgan o‘tishlar soni hisoblangan. Bu turdagi tahlil to‘g‘ri bo‘lib, ikki algoritmning nisbiy tezliklarini aniqlash imkonini beradi, ammo uning amaliyotda qo‘llanilishi kam, chunki ko‘p vaqt talab qiladi. Avval bajariladigan algoritmning Tyuring mashinasidagi o‘tish funksiyalarini yozish, keyin esa bajarilish vaqti hisoblanadi.

Algoritmlarni tahlil qilishning boshqa yaxshiroq usuli - uni biror yuqori bosqichli til Pascal, C, C++, JAVA da yozish yoki oddiy psevdokodlarda yozishdir. Barcha algoritmlarning asosiy boshqaruv strukturasini ifodalaganda psevdokodlarning xossalari ahamiyatga ega emas. Ixtiyoriy til bizning talabimizga javob beradi, chunki **for yoki while** shaklidagi sikllar, **if, case yoki switch** ko‘rinishidagi tarmoqlanish mehanizmlari barcha dasturlash tillarida mavjud. Har gal biz bitta aniq algoritmni ko‘rib chiqishimizga to‘g‘ri keladi – unda birdan ortiq funksiya yoki programma fragmenti kiritilgan bo‘ladi, shuning uchun yuqorida keltirilgan

tillarning tezligi umuman muhim emas. Psevdokodlardan foydalanishimizning sababi shunda.

Ko‘plab dasturlash tillarida mantiqiy ifodaning qiymatlari qisqartirilgan shaklda hisoblanadi. Bu A and V ifodadagi V hadning qiymati qachonki A rost bo‘lsagina hisoblanadi, aks holda natija V ga bog‘liq bo‘lmagan tarzda yolg‘on bo‘ladi. Xuddi shunday A or V ifodada A ning qiymati rost bo‘lsa, V hadning qiymati hisoblanmaydi. Ko‘rinib turibdiki, murakkab shartlarning 1 yoki 2 ga tengligidagi taqqoslashlarining sonini hisoblash shart emas.

Kiruvchi berilganlar sinfi

Algoritmlarning tahlilida kiruvchi ma’lumotlarning roli yuqori, chunki algoritm harakatlarining ketma-ketligi kiruvchi ma’lumotlar bilan belgilanadi. Masalan, *N* ta elementdan tashkil topgan ro‘yxatning eng katta elementini topish uchun quyidagi algoritmdan foydalanish mumkin:

```
largest = list  [l]
for i = 2 to N do
if  (list  [i]  > largest) then
 largest = list[i]
end if
end for
```

Agar ro‘yxat kamayish tartibida bo‘lsa, u holda sikl boshlanishidan avval bitta o‘zlashtirish bajariladi, sikl tanasida esa o‘zlashtirish bo‘lmaydi. Agar ro‘yxat o‘sish tartibida bo‘lsa, u holda *N* ta o‘zlashtirish bajariladi (sikl boshlanishidan avval bitta va *N*-1 ta siklda). Biz tahlil qilish davomida kiruvchi qiymatlar to‘plamining turli imkoniyatlarini ko‘rib chiqishimiz kerak, agar bitta to‘plam bilan chegaralansak, bu yechim eng tez (yoki eng sekin) bo‘lgan to‘plam bo‘lib chiqishi mumkin. Natijada biz algoritm haqidagi yolg‘on tasavvurga ega bo‘lamiz. Buning o‘rniga kiruvchi to‘plamlar turining barchasini ko‘rib chiqamiz.

Biz kiruvchi to‘plamlarni har bir to‘plamdagi algoritm holatiga bog‘liq holda sinflarga bo‘lib chiqamiz. Bunday bo‘linish ko‘rib chiqilayotgan to‘plamlar miqdorini kamaytirish imkonini beradi. Masalan, 10 ta sondan iborat ro‘yxat uchun eng katta elementni

topish algoritmini qo‘llaymiz. Birinchi soni eng katta bo‘lgan 362 880 kiruvchi to‘plamlar mavjud, ularni bitta sinfga joylashtirish mumkin. Agar qiymati bo‘yicha eng katta son ikkinchi o‘rinda turgan bo‘lsa, u holda algoritm ikkita o‘zlashtirishni amalga oshiradi. Eng katta son ikkinchi o‘rinda turgan to‘plam 362 880. Ularni boshqa sinfga kiritish mumkin. 1 dan *N* gacha bo‘lgan sonlar orasida eng katta sonning o‘zgarish holida o‘zlashtirishlar sonining qanday o‘zgarishini ko‘rishimiz mumkin.

Shunday qilib, barcha kiruvchi to‘plamlarni bajarilgan o‘zlashtirishlar soni bo‘yicha *N* ta turli sinfga bo‘lish kerak. Ko‘rib turganingizdek, har bir sinfda joylashgan to‘plamlarni birma-bir yozish yoki yozib olish shart emas. Faqatgina har bir to‘plamdagi sinflar miqdori va ish hajmini bilish yetarli.

Kiruvchi berilganlarning mumkin bo‘lgan to‘plami *N* kattalashganda judayam katta bo‘lishi mumkin. Masalan, 10 ta turli soni ro‘yxatda 3 628 800 usulda joylashtirish mumkin. Bu usullarning barchasini ko‘rib chiqishning imkoni yo‘q. Biz buning o‘rniga algoritm bajarilishiga ko‘ra ro‘yxatni sinflarga bo‘lamiz. Yuqorida ko‘rsatilgan algoritm uchun bo‘linish eng katta qiymatning joylashishi o‘rniga asoslanadi. Natijada 10 ta turli sinf hosil bo‘ladi. Boshqa algoritm uchun, masalan, eng katta va eng kichik sonni topish algoritmida bo‘linish eng katta va eng kichik sonning joylashuviga asoslanadi. Bunday bo‘linishda 90 ta sinf bo‘ladi. Sinflarni ajratib bo‘lgach, har bir sinfdan bitta to‘plamda algoritm holatini ko‘rish mumkin. Agar sinflar to‘g‘ri tanlangan bo‘lsa, u holda bir sinfdagi kiruvchi berilganlar to‘plamida algoritm bir xil miqdordagi amallarni bajaradi, boshqa sinfning to‘plamlari uchun esa amallar miqdori boshqacha bo‘ladi.

Xotira bo‘yicha murakkablik

Biz asosan algoritmlarning vaqt bo‘yicha murakkabligini muhokama qilamiz, ammo ish bajarish uchun u yoki bu algoritmga qancha xotira kerakligi haqida ham aytish mumkin. Kompyuter xotirasi (ham ichki, ham tashqi) hajmi chegaralangan. Kompyuterlar rivojlanishining dastlabki bosqichlarida bu tahlil uslubiy xarakterga ega edi. Barcha algoritmlar chegaralangan xotira yetarli yoki

qo‘shimcha maydonni talab qiluvchi algoritmlarga bo‘linadi. Ko‘pincha dasturlovchilar xotirasiga ega va tashqi qurilmalar talab qilmaydigan sekin ishlovchi algoritmni tanlashar edi.

Kompyuter xotirasiga bo‘lgan talab juda katta edi, shuning uchun qaysi ma’lumotlar saqlanib qoladi, bunday saqlashning samarali usullari qanday kabi savollar o‘rganilar edi. Faraz qilaylik, masalan, biz -10 dan +10 gacha intervaldagi verguldan keyin bitta o‘nli belgiga ega bo‘lgan moddiy son yozyapmiz. Moddiy sonni yozishda ko‘pchilik kompyuterlar 4 dan 8 baytgacha xotira sarflaydi, lekin agar bu sonni avvaldan 10 ga ko‘paytirsak, -100 dan +100 gacha intervaldagi butun son hosil qilamiz va uni saqlash uchun bor yo‘g‘i bir bayt sarflanadi. Birinchi variant bilan solishtirsak, 3-7 bayt tejashga erishildi. 1000 ta shunday son saqlaydigan dastur 3000 dan 7000 baytgacha tejaydi. Agar o‘tgan asrning 80 yillarida kompyuterlarning xotirasi 65536 bayt bo‘lganligini e’tiborga olsak, jiddiy tejash ko‘zga tashlanadi. Aynan shu kompyuter dasturlarining uzoq yil ishlashi xotirani tejash zaruriyati bilan bir qatorda 2000 yil muammo tug‘dirdi. Agar sizning dasturingiz turli sanalardan foydalansa, yilni yozish uchun 1999 o‘rniga 99 ifodasini saqlagan holda joyning yarmini tejasa bo‘ladi. 80-yillardagi dastur mualliflari mahsulotlari 2000-yilgacha yashashini taxmin ham qilishmagan edi.

Hozirgi kunda bozorlarda taklif qilinayotgan dasturiy ta’minotga nazar tashlasak, xotiraning bunday tahlili o‘tkazilmaganligi ayon bo‘ladi. Oddiy dasturlar uchun zarur xotira hajmi megabaytlarda o‘lchanadi. Dastur tuzuvchilar joyni tejash ehtiyojini his qilmayotganga o‘xshaydilar, ularning fikricha, agar foydalanuvchida yetarli xotira bo‘lmasa, u dastur bajarilishi uchun yetmayotgan 32 yoki undan ortiq megabayt xotira yoki uni saqlash uchun yangi qattiq disk sotib oladi. Natijada kompyuterlar o‘zining belgilangan muddatidan avval yaroqsiz holga kelib qoladi.

Yaqinda tarqalgan cho‘ntak kompyuterlari (PDA – personal digital assistant) yangi ohang olib kirdi. Bunday qurilmaning xotirasi ham ma’lumotlar, ham dasturlar uchun 2 dan 8 megabaytgacha. Shuning uchun ham ma’lumotlarni ixcham saqlashni ta’minlovchi kichik dasturlarni yaratish qiyin bo‘lib qolmoqda.

Nimani hisoblash va nimani inobatga olish lozim

Nimani hisoblash masalasi ikkita qadamdan iborat. Birinchi qadamda ahamiyatli jarayon yoki jarayonlar guruhi tanlanadi, ikkinchi qadamda shu jarayonlardan qay biri algoritmda joylashgan, qaysilari esa qo‘shimcha xarajatlarni yoki ma’lumotlarni qayd etish va hisobga olishga ketishi tashkil etadi. Ikki xil ahamiyatli jarayon turi mavjud: taqqoslash va arifmetik amallar. Barcha taqqoslash operatorlari ekvivalent hisoblanadi va ularni izlash hamda ajratish algoritmlarida inobatga olinadi. Taqqoslash miqdori bunday algoritmlarning muhim elementi hisoblanadi, izlashda ushbu miqdor izlangan kattalik bilan mos tushadimi, saralashda esa berilgan oraliqdan chiqishi aniqlanadi. Solishtiruvchi operatorlar bir kattalikning ikkinchisi bilan teng yoki teng emasligi, katta yoki kichikligi, kichik yoki tengligi, katta yoki tengligini tekshiradi.

Biz arifmetik amallarni ikki guruhga bo‘lamiz: additiv va multiplikativ. Additiv operatorlar (qisqa qilib aytganda qo‘shuv) qo‘shish, ayirish, orttirish va qisqartirishni o‘z ichiga oladi. Multiplikativ operatorlar (yoki qisqacha ko‘paytirishlar) ko‘paytirish, bo‘lish va modul bo‘yicha qoldiq olishni o‘z ichiga oladi. Ikki guruhga ajratish ko‘paytirishning qo‘shishdan ko‘proq ishlatilishiga bog‘liq. Amaliyotda ba’zi algoritmlar ularda ko‘paytirish kam bo‘lsa, qo‘shishlar soni proporsional darajada o‘ssa ham afzalroq hisoblanadi. Butun sonli ikki darajaga ko‘paytirish yoki bo‘lish alohida holatni tashkil qiladi. Bu jarayon siljishga olib keladi, keyingisi esa tezlik jihatdan qo‘shishning ekvivalenti hisoblanadi. Biroq, bu tafovut sezilarli bo‘lgan hollar juda kam, chunki 2 ga ko‘paytirish va bo‘lish birinchi navbatda taqqoslash operatorlari ahamiyatga ega bo‘lgan «taqsimla va boshqar» singari algoritmlarda uchraydi.

Kiruvchi ma’lumotlarning sinflari

Algoritmni tahlil qilishda kiruvchi ma’lumotlarni tanlash uning bajarilishiga ta’sir qilishi mumkin. Aytaylik, ba’zi saralash algoritmlari, agar kirish ro‘yxati saralangan bo‘lsa, juda tez ishlashi mumkin, boshqa algoritmlar shunday ro‘yxatda uncha katta

bo‘lmagan natijani ko‘rsatadi. Tasodifiy ro‘yxatda esa natija buning teskarisi bo‘lishi mumkin. Shuning uchun biz ma’lumotlarning bir kirish ro‘yxatidagi algoritmlar harakatini tahlil qilish bilan chegaralanmaymiz. Biz algoritmni ham eng tez, ham eng sekin ishlashini ta’minlovchi ma’lumotlarni qidiramiz. Bundan tashqari, biz barcha mavjud ma’lumotlar to‘plamidagi algoritmlarning o‘rtacha samarasini ham baholaymiz.

Eng yaxshi holat

Bo‘limning nomlanishidan ham ko‘rinib turibdiki, algoritmlar uchun eng yaxshi holat bu qisqa vaqt ichida amalga oshiriladigan algoritmning ma’lumotlar jamlanmasi. Bunday jamlanma algoritm eng yaxshi amal bajaradigan qiymatlar kombinatsiyasini ifodalaydi. Agar biz izlash algoritmini tekshirsak, izlangan qiymat birinchi algoritm tekshirayotgan katakka yozilgan bo‘lsa (odatda maqsadli qiymat yoki kalit deb ataladi), ma’lumotlar to‘plami eng yaxshi hisoblanadi. Bunday algoritmga uning murakkabligidan qat’i nazar, bitta taqqoslash kerak bo‘ladi. Shuni eslatish kerakki, ro‘yxatdan izlashda, uning qanchalik uzun bo‘lishidan qat’i nazar, eng yaxshi holat doimiy vaqtni talab qiladi. Umuman, eng yaxshi holatda algoritmni bajarish vaqti kichik yoki doimiy bo‘ladi, shuning uchun biz bunday tahlilni kam o‘tkazamiz.

Eng yomon holat

Eng yomon holatni tahlil qilish juda muhim, chunki u algoritm ishining maksimal vaqtini tasavvur qilishga yordam beradi. Eng yomon holatni tahlil qilganda algoritm eng ko‘p ish bajaradigan kirish ma’lumotlarini topish zarur. Izlovchi algoritm uchun bu kabi kiruvchi ma’lumotlar – bu shunday ro‘yxatki, unda izlangan kalit oxirida keladi yoki umuman bo‘lmaydi. Natijada N taqqoslash kerak bo‘ladi. Eng yomon holatning tahlili tanlangan algoritmga qarab dasturning ishlash vaqti uchun yuqori bahoni beradi.

O‘rtacha holat

O‘rta holatning tahlili eng murakkab hisoblanadi, chunki u ko‘pgina detallarni hisobga olishni talab qiladi. Tahlil asosini mavjud bo‘lgan kiruvchi ma’lumotlar to‘plamini bo‘lib chiqish lozim bo‘lgan turli guruhlarni aniqlash tashkil qiladi. Ikkinchi qadamda kiruvchi ma’lumotlar to‘plami qaysi guruhga tegishli bo‘lish ehtimoli aniqlanadi. Uchinchi qadamda har bir guruhdagi ma’lumotlarga algoritmning ish vaqti hisoblanadi. Algoritmning bir guruhdagi hamma kiruvchi ma’lumotlar uchun ishlash vaqti bir xil bo‘lishi kerak, aks holda guruhni bo‘lish lozim. O‘rtacha ish vaqti

$$A(n)\sum_{i=1}^{m} p_i \square. \qquad (1)$$

formula orqali hisoblanadi. Bu yerda n - kiruvchi ma’lumotlar o‘lchami, m - guruhlar soni, p_i - kiruvchi ma’lumotlarning i sonli guruhga tegishlilik ehtimoli, $t_i - i$ sonli guruhdagi ma’lumotni qayta ishlash uchun algoritmga kerak bo‘ladigan vaqt deb belgilangan.

Ba’zi hollarda biz kiruvchi ma’lumotlarning har bir guruhga tushish ehtimolini bir hil deb taxmin qilamiz. Boshqacha aytganda, agar guruh 5 ta bo‘lsa, birinchi guruhga tushish ehtimoli ikkinchi yoki boshqa guruhga tushish ehtimolidek, ya’ni har bir guruhga tushish ehtimoli 0,2 ga teng. Bu holda ishning o‘rtacha vaqtini avvalgi formula bilan yoki unga ekvivalent soddalashtirilgan barcha guruhlarning teng ehtimolligida haqiqiy bo‘lgan

$$A(n)\sum_{i=1}^{m} p_i \square. \qquad (2)$$

formuladan foydalanishimiz mumkin.

	INSERTION-SORT(A)		**Takrorlanish**
1	For j=2 to A.Length	c_1	n
2	Key=A[j]	C_2	n-1
3	A[1..j-1]	0	n-1

4	I=j-1	c_4	n-1
5	While i>0 va A[i]>key	c_5	$\sum_{j=2}^{n} t_j$
6	A[1..j-1]=A[i]	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	I=i-1	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8	A[i+1]=key	C_8	n-1

$$T(n) = c_1 n + c_1 (n-1) + c_4 (n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1)$$

$$+ c_7 \sum_{j=2}^{n} (t_j - 1) + c_8 (n-1).$$

$$T(n) = c_1 n + c_2 (n-1) + c_4 (n-1) + c_5 (n-1) +$$

$$c_8 (n-1) = (c_1 + c_2 + c_4 + c_5 + c_8) n - (c_1 + c_2 + c_4 + c_5 + c_8).$$

$$\sum_{j=2}^{n} j = \frac{n(n-1)}{2} - 1 \qquad \sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

$$T(n) = c_1 n + c_2 (n-1) + c_4 (n-1) + c_5 \left(\frac{n(n+1)}{2} - 1 \right) +$$

$$+ c_6 \left(\frac{n(n+1)}{2} \right) + c_7 \left(\frac{n(n+1)}{2} \right) + c_8 (n-1) = \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2$$

$$= (c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8) n - (c_2 + c_4 + c_5 + c_8).$$

Ko‘plab algoritmlar ishlash vaqtiga sezilarli ta’sir ko‘rsatadigan asosiy -N parametrga ega bo‘ladi. N parametr polinom darajasi, saralash yoki qidiruvda fayl o‘lchami, qatordagi simvollar soni yoki boshqa abstract o‘lchov bo‘lishi mumkin. Bizning maqsadimiz N ga bog‘liq bo‘lgan holda matematik formulalar yordamida maksimal darajada oddiy va ko‘pgina parametrlar qiymatiga mos keluvchi dasturning resurs talablarini ifodalashdan iboratdir. algoritmlar ishlashi asosan quyidagi funksiyalarga prorortsional bo‘ladi:

1 ko‘p hollarda ko‘plab dasturlar bir yoki bir necha marta ishga tushiriladi, bunday dasturlar bajarish vaqti muntazam bo‘ladi.

LogN -agar bajarilish vaqti logarifmik bo‘lsa, dastur N uzunlikda sekinlashadi. Bizni qiziqtirayotgan sohani bajarilish vaqti katta bo‘lmagan konstantaga teng. Logarifm asosi konstantaga o‘zgaradi, N-1000 ga teng bo‘lganda logarifm asosi 10 ga teng bo‘lsa, logN 3 ga teng bo‘ladi. Agar logarifm asosi 2 ga bo‘lsa N millionga teng bo‘lib, logN qiymati 2 marta ortadi.

NlogN algoritm masalani qism masalalarga ajratib, ularni alohida yechib bo‘lgach yechimlarni birlashtirganda bajarish vaqti NlogN ga proportsional bo‘ladi. N 1 millionga teng bo‘lsa, NlogN 20 millionga yaqin bo‘ladi. N 2 marta ortsa, bajarish vaqti 2 martadan ortiqroq bo‘ladi.

N^2 agar algoritm bajarish vaqti kvadratik bo‘lsa, u katta bo‘lmagan vazifalarni yechish uchun amaliy foydalanishda samarali bo‘ladi. Odatda algoritmlarda kvadratik bajarish vaqti ma’lumotlarning barcha juftlik elementlarini qayta ishlab chiqishda yuzaga keladi. Agar *N* 1000 ga teng bo‘lsa, bajarilish vaqti 1 millionga teng bo‘ladi. Agar *N* 2 ga ko‘paysa, bajarish vaqti 4 martaga ko‘payadi.

N^3 ma’lumotlarni qayta ishlovchi uchlik elementlar algoritmi bo‘lib, kublik bajarish vaqtiga ega va kichik masalalarga amaliy qo‘llaniladi. N 100 ga teng bo‘lganda, bajarish vaqti 1 millionga teng bo‘ladi. Agar N ikki marta oshsa, bajarish vaqti 8 marta ortadi.

2^N faqatgina eksponensial bajarish vaqtli algoritmlargina amaliy qo‘llanilishga ega.

Agar *N 20* ga teng bo‘lsa, bajarish vaqti 1 millionga teng bo‘ladi. Agar N ikki marta oshsa, bajarish vaqti 4 marta ortadi.

Ko‘p hollarda algoritm bararish vaqti “chiziqli”, “*NlogN*”, “kubik” va hokazo deb ataladi. Dasturning bajarish vaqtini kamaytirish uchun ichki sikldagi ketma-ketliklarni minimallashtiramiz.

Kichik hajmdagi dasturlarni qaysi metod orqali bajarish muhim emas, chunki zamonaviy kompyuter dasturni yuqori tezlikda bajarib beradi. Ammo katta hajmdagi dasturlarni bajarish esa hattoki zamonaviy kompyuterlar uchun murakkablik tug‘dirishi mumkin. Quyidagi jadvalda katta hajmdagi soniyalarni kun, oy, yillarga ko‘rinishi keltirilgan:

soniyalar	
10^2	1.7daqiqalar
10^4	2.8 soatlar
10^5	1.1 kunlar
10^6	1.6 haftalar
10^7	3.8 oylar
10^8	3.1 yillar
10^9	3.1 o‘n yilliklar
10^{10}	3.1 yuz yilliklar
10^{11}	hech qachon

1-jadval. Funksiyalarda tez-tez uchrab turadigan qiymatlar

Ushbu jadvalda algoritmlar tahlilida tez-tez uchrab turadigan ba’zi bir funksiyalarning nisbiy kattaliklari keltirilgan. Ko‘rinib turibdiki, kvadrat funksiya ayniqsa N ning katta qiymatlari uchun dominantlik qiladi. Masalan, N ning katta qiymatlarida $N^{3/2}$ chisi Nlg^2 N dan katta bo‘ladi, ammo N ning kichik qiymatlarida teskari hol kuzatiladi. Biz tezkor algoritmlardan sekin bajariladigan algoritmlarni ular orasidagi katta farq orqali oson farqlay olamiz, masalan, lgn va N yoki N va N^2 .

$\lg N$	$\sqrt{N}$	N	$N\lg N$	$N(\lg N)^2$	$N^{3/2}$	N^2
3	3	10	33	110	32	100
7	10	100	664	4414	1000	10000
10	32	1000	9966	99317	31623	1000000
13	100	10000	132877	1765633	1000000	100000000
17	316	100000	1660964	27588016	31622777	10000000000
20	1000	1000000	19931569	397267426	1000000000	1000000000000

Katta hajmdagi masalalarni yechish algoritmini bajarish vaqti

Katta hajmdagi masalalarni yechishda samarali algoritmdan foydalanish zarur.

Quyidagi jadvalda 1 million yoki 1 milliard o‘lchamdagi masalani yechishda chiziqli, NlogN va kvadrat algoritmlardan foydalanib, 1 sekundda 1 million, 1 milliard va 1 trillion instruksiyani tezlik bilan bajaruvchi kompyuterlarda algoritm minimal bajarish vaqti keltirilgan. Tezkor algoritm sekin ishlovchi mashinada ham masalani yechish imkonini beradi, ammo sekin algoritmdan tezkor mashinada foydalanish hech qanday natijaga olib kelmaydi.

Operatsiyalar sekundda	1 mln o‘lchamdagi masala			1 mlrd o‘lchamdagi masala		
	N	$N \lg N$	N^2	N	$N \lg N$	N^2
10^6	Sekund	sekund	hafta	soat	soat	hech qachon
10^9	bir zumga	bir zumga	soatda	sekund	sekund	o‘n yillik
10^{12}	bir zumga	bir zumga	sekund	bir zumda	bir zumda	hafta

Algoritm tahlilida bir qancha funksiyadan foydalanish mumkin. Masalan, bajarish vaqti N^3 bo‘lgan N^2 li kiritilgan algoritmni $N^{3/2}$ algoritm sifatida qarash ham mumkin. Bundan tashqari ba’zi bir 2 qism masalaga bo‘linuvchi algoritmlar $Nlog^2 N$ bajarish vaqtiga proportsional bo‘ladi.

1-misol. Misol tariqasida faylda N butun sonlar uchliklar sonini hisoblovchi ThreeSum dasturini olamiz. Matnli kiruvchi ma’lumot sifatida 1Mints.txt faylini olamiz. [17]

1Kints.txt, 2Kints.txt, 4Kints.txt va 8Kints.txt faylli mos ravishda 1Mints.txt dagi 1000, 2000, 4000 va 8000 sondan iborat ThreeSum dasturini bajarishni birinchi mashqi sifatida shaxsiy kompyuteringizda bajarib ko‘ring.

[17] Robert Sedgewick and Kevin Wayne. Algorithms. FOURTH EDITION. Princeton University. First printing, March 2011. Pp 173-175.

```
public class ThreeSum
{
   public static int count(int[] a)
   {  // Count triples that sum to 0.
      int N = a.length;
      int cnt = 0;
      for (int i = 0; i < N; i++)
         for (int j = i+1; j < N; j++)
            for (int k = j+1; k < N; k++)
               if (a[i] + a[j] + a[k] == 0)
                  cnt++;
      return cnt;
   }

   public static void main(String[] args)
   {
      int[] a = In.readInts(args[0]);
      StdOut.println(count(a));
   }
}
```

Siz 1Kints.txt faylda nolinchi yig‘indi bilan 70 ta uchlik, 2Kints.txt faylda esa 528 ta shunday uchliklar mavjud ekanligini bilib olasiz. Dasturda 4Kints.txt fayldagi nolinchi yig‘indi bilan 4039 uchlikni hisoblashga ko‘p vaqt talab etiladi. Tabiiyki, sizda dasturda 8Kints.txt fayldagi uchliklarni hisoblashga qancha vaqt ketadi degan savol tug‘iladi? B savolga javob topish qiyin emas. Dastur bajarilishiga ketadigan aniq vaqtni ko‘rsatish qiyin emas.

```
% more 1Mints.txt
 324110
-442472
 626686
-157678
 508681
 123414
 -77867
 155091
 129801
 287381
 604242
 686904
-247109
  77867
 982455
-210707
-922943
-738817
  85168
 855430
 ...
```

% java ThreeSum 1000 1Kints.txt

tick tick tick

70

% java ThreeSum 2000 2Kints.txt

tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick

528

% java ThreeSum 4000 4Kints.txt

**tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick tick

API

```
public class Stopwatch
```

	`Stopwatch()`	Sekundomer yaratilishi
`double`	`elapsedTime()`	Bajarilgan vaqtni qaytarib beradi

typical client

```
public static void main(String[] args)
{
   int N = Integer.parseInt(args[0]);
   int[] a = new int[N];
   for (int i = 0; i < N; i++)
      a[i] = StdRandom.uniform(-1000000, 1000000);
   Stopwatch timer = new Stopwatch();
   int cnt = ThreeSum.count(a);
   double time = timer.elapsedTime();
   StdOut.println(cnt + " triples " + time);
}
```

application

```
% java Stopwatch 1000
51 triples 0.488 seconds

% java Stopwatch 2000
516 triples 3.855 seconds
```

implementation

```
public class Stopwatch
{
   private final long start;

   public Stopwatch()
   {  start = System.currentTimeMillis();  }

   public double elapsedTime()
   {
      long now = System.currentTimeMillis();
      return (now - start) / 1000.0;
   }

}
```

Sekundomer uchun konstruksiya

Ko‘p hollarda uchraydigan funksiyalar[18]

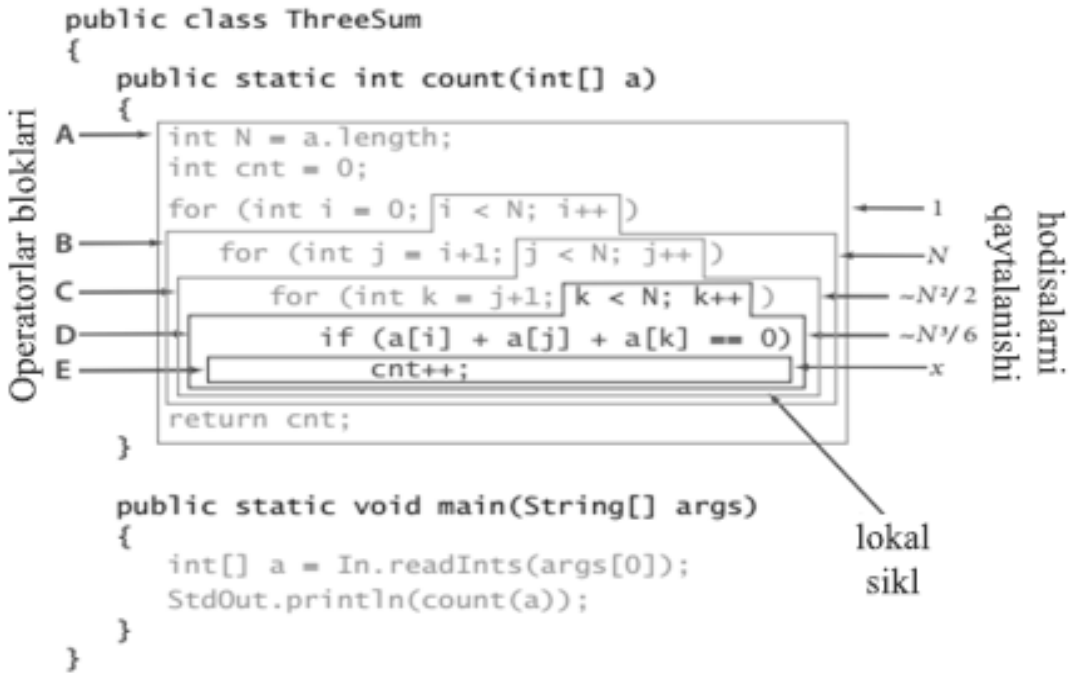
O’sish tartibida saralash

Ko’rinishi	Funksiya
konstanta	1
logarifm	$\log N$
chiziqli	N
chiziqli logarifm	$N \log N$
kvadratik	N^2
kubik	N^3
eksponenta	2^N

Odatda o’sish tartibida saralash funksiya tarkibiga kiritilgan

Dastur operatorlarining bajarilish chastotasi

A Xossa. ThreeSum dasturini bajarilish vaqti N^3 ga teng.



[18] Robert Sedgewick and Kevin Wayne. Algorithms. FOURTH EDITION. Princeton University. First printing, March 2011. Pp 179-185.

Operator bloki	Vaqt (sekund)	Takrorlanish soni	Bajariladigan vaqti
E	t_0	x (kirituvchi songa bog'liq)	$t_0 x$
D	t_1	$N^3/6 - N^2/2 + N/3$	$t_1 (N^3/6 - N^2/2 + N/3)$
C	t_2	$N^2/2 - N/2$	$t_2 (N^2/2 - N/2)$
B	t_3	N	$t_3 N$
A	t_4	1	t_4
		Umumiy yeg'indi	$(t_1/6)\, N^3 + (t_2/2 - t_1/2)\, N^2 + (t_1/3 - t_2/2 + t_3)\, N + t_4 + t_0 x$
		taxminiy	$\sim (t_1/6)\, N^3$ (x- kichik son)
		o'sish tartibida saralash	N^3

Dastur bajarilish vaqtining tahlili

Algoritmlar tahlilida ko‘p hollarda uchraydigan funksiyalar jadvali

Ko'rinishi	Belgilanishi	Ma'nosi
Katta butun son	$\lfloor x \rfloor$	x dan katta bo'lmagan butun son
Kichik butun son	$\lceil x \rceil$	x dan kichik bo'lmagan butun son
Natural logarifmik	$\ln N$	$\log_e N$ (x uchun $e^x = N$)
Binar logarifmik	$\lg N$	$\log_2 N$ (x uchun $2^x = N$)
Butun sonli binar logarifmik	$\lfloor \lg N \rfloor$	lgN dan katta bo'lmagan butun son (# N ning binar ko'rinishi (ikkilik)) – 1
Garmonik sonlar	H_N	$1 + 1/2 + 1/3 + 1/4 + \ldots + 1/N$
Faktorial	$N!$	$1 \times 2 \times 3 \times 4 \times \ldots \times N$

Algoritmlar tahlilida foydali bo‘lgan approksimatsiyalar

Ko’rinishi	Formulasi
garmonik yeg'indi	$H_N = 1 + 1/2 + 1/3 + 1/4 + \ldots + 1/N \sim \ln N$
uchburchakli yeg'indi	$1 + 2 + 3 + 4 + \ldots + N \sim N^2/2$
geometrik yeg'indi	$1 + 2 + 4 + 8 + \ldots + N = 2N - 1 \sim 2N$ agar $N = 2^n$
Stirlingning taxminiy yeg'indisi	$\lg N! = \lg 1 + \lg 2 + \lg 3 + \lg 4 + \ldots + \lg N \sim N \lg N$
binom koeffitsient	$\binom{N}{k} \sim N^k/k!$ k eng kichik konstanta bo’lsa
eksponent	$(1 - 1/x)^x \sim 1/e$

II BOB. ALGORITMLARNING TATBIQI

6-§. Algoritmlarni ishlab chiqish uslublari

Algoitmlarni yaratish jarayoni birinchi navbatda bevosita qo‘yilgan masalaga va uni samarali, ixcham hamda qulay bo‘lishi esa dasturchining bilimi va mahoratiga bog‘liq. Samarali algoritm yaratilishi uchun quyida keltirilgan omillar ham muhim o‘rin tutadi:

- Algoritm yaratuvchining bilimlari hajmi;
- Intuitsiya, tasavvur va fikrlash darajasi;
- Yechimlarni mustaqil aniqlash tajribasi;
- Asosiy matematik amallardan foydalanish ko‘nikmalari;
- Tahlil, sintez, taqqoslash, moslik, umumlashtirish, analogiya kabi amallardan unumli foydalanish;
- Standart va nostandart masalalarni yechimini aniqlay olish;
- Mantiqiy ko‘nikmalarni (gipotezalarni ilgari surish, strukturalarni tezlik bilan tuzish, xulosalar chiqarish) doimiy ravishda takomillashtirib borishi;
- Boshqalarning aniq masalaga keltirgan yechimini aniqlay olishi va uni baholay bilishi;
- Dasturlash tilining imkoniyatlaridan foydalana olish darajasi;

Strukturaviy algoritmlar- faqat standart boshqaruvchi strukturalardan foydalaniladigan yetarli darajada oddiy, tushunarli va oson o‘qiladigan masalalarni yaratish metodi.

Algoritmlashning strukturali tamoyillari –bu ularning bazaviy algoritmik birliklar asosida (chiziqli, tarmoqlanish, takrorlanish) yaratilishidir. Strukturali algoritm statik va dinamik holatlarda bo‘lishi mumkin. Bunday algoritmlarning to‘g‘riligini ularning bajarilish va yaratilish jarayonining barcha bosqichlarida kuzatish mumkin. Har qanday algoritmni strukturalashtirilgan –ekvivalent algoritm bilan ifodalanishi mumkin.

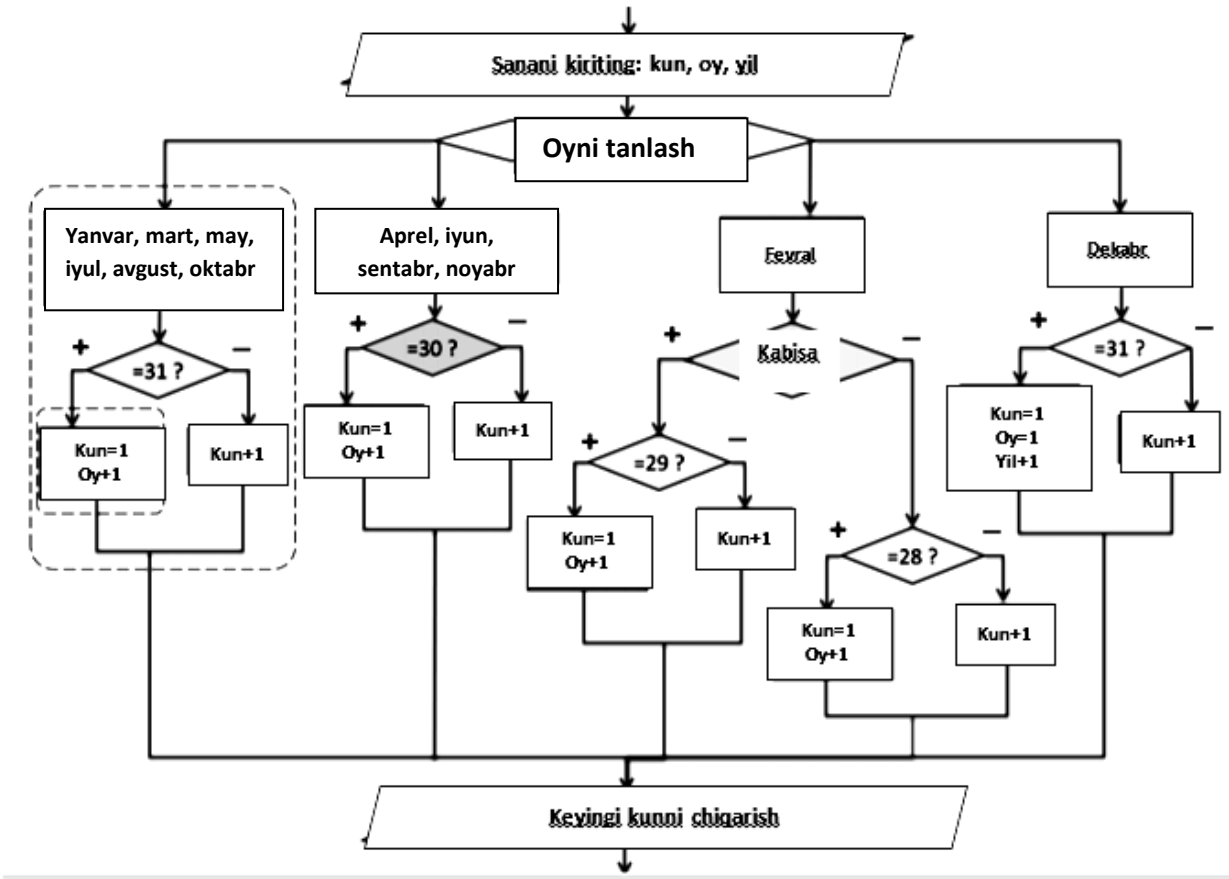
Bu metodga quyidagi masalalarga algoritm tuzishni misol sifatida keltiramiz:

1-masala: Kiritilgan sanaga ko‘ra navbatdagi sanani chop etuvchi algoritm tuzilsin.

Asosiy holatlar:

- oy tarkibidagi kunlar sonini aniqlash;
- yilning kabisa yili yoki kabisa yili emasligini aniqlah;
- oyning oxirgi kunu bo‘lsa, keyingi oyning 1 chi kunini belgilash;
- fevral oyi kunlarini hisobga olish;

Asosiy shart: ((Y mod 4=0) and (Y mod 100<>0)) or (Y mod 400=0)



Juda ko‘p qo‘llaniladigan metod bu **“yuqoridan pastga”** deb yuritiladi. Bunda qo‘yilgan asosiy masala kichik masalalarga bo‘linadi, demak masala uchun yaratiladigam algoritm ham bir necha yordamchi algoritmlarga bo‘linadi. Bunday algoritmlarni funksiya yoki protseduralar deb ham yuritish mumkin.

2-masala. Koordinatalari berilgan nuqtani klaviaturadan mos tugmalar bosilganda “harakatlanishini” ta’minlovchi dastur tuzing. ('u'-yuqoriga, 'd'-pastga, ‘l’-chapga, ‘r’-o‘ngga)

Dasturni qisqacha izohlar bilan keltiramiz:

uses graph,crt; var gd,gm,x,y:integer; sim: char; label 10,20;	Dasturning tavsiflash qismi: foydalanadigan modullar, o‘zgaruvchilar va ularning tiplari, belgilar.

begin
initgraph(gd,gm,'');
gd:=detect; x:=345; y:=120;
	Grafik rejimni belgilash, grafik drayverni avtomatik aniqlash va nuqtaning dastlabki koordinatalari qiymatlari

10:sim:=readkey;	Manzil va klaviaturadan simvolni kiritish

if sim='u' then
begin
putpixel(y,x,black) ;
x:=x-1; if x=0 then x:=getmaxy;
putpixel(y,x,red);
END else
	sim='u' bo‘lganda nuqtani ekranning yuqori qismiga qarab harakatlantirish va u eng yuqori nuqtaga yetib borsa harakatni quyi chegaradan boshlash. (ranglar effekti orqali)

if sim='r' then
begin
putpixel(y,x,black) ;
y:=y+1; if y=getmaxy then y:=0;
putpixel(y,x,red);
end else
	sim='r' bo‘lganda nuqtani ekranning o‘ng qismiga harakatlantirish va u maksimal nuqtaga yetib borsa harakatni chap chegaradan boshlash. (ranglar effekti orqali)

if sim='l' then
begin
putpixel(y,x,black) ;
y:=y-1; if y=0 then y:=getmaxy;
putpixel(y,x,red);
end else
	sim='l' bo‘lganda nuqtani ekranning chap qismiga harakatlantirish va u eng minimal nuqtaga yetib borsa harakatni o‘ng chegaradan boshlash. (ranglar effekti orqali)

if sim='d' then
begin
putpixel(y,x,black) ;
x:=x+1; if x=getmaxy then x:=0;
putpixel(y,x,red);
end;
	sim='d' bo‘lganda nuqtani ekranning quyi qismiga qarab harakatlantirish va u eng quyi nuqtaga yetib borsa harakatni yuqori chegaradan boshlash. (ranglar effekti orqali)

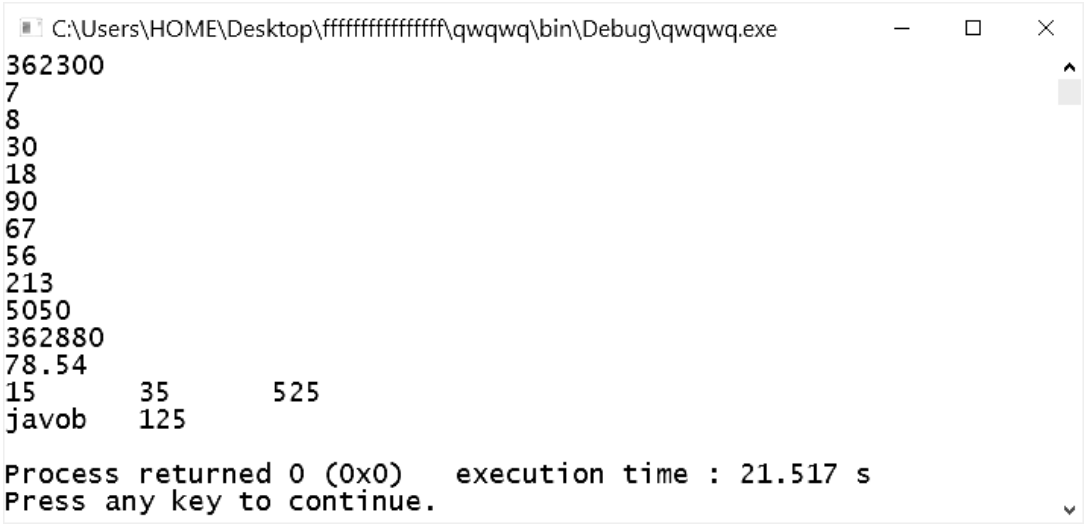
goto 10; end.	Dastur tugashi.

Protsedurali algoritmlar metodi. Bu metod algoritmlar tarkibida aniq natijalar beruvchi funksiya yoki proseduralardan unumli foydalanishdan iborat. Algoritmlarni bu metoddan foydalanib yaratish uni ixcham, hamda yaqqol ko‘zga tashlanishini ta’minlab, algoritm tarkibida ulardan bir necha marotaba foydalanish imkoniyatini beradi. Quyida bu metodni C++ tilida ifodalangan bir necha sodda protsedura va funksiyalar qatnashgan algoritm misolida kuzatishimiz mumkin.

```
#include <iostream>
int sum (int x,int y,int z) { return(x+y+z);} // uchta sonning
yig‘indisini beruvchi funksiya

int sum1 (int n)
{int s; s=0; for (int i=1; i<n;i++)   s=s+i; return(s);} // n
sonigacha bo‘lgan natural sonlar yig‘indisini hisoblovchi funksiya

int fakn (int n)
{int p; p=1; for (int i=1; i<n;i++)  p=p*i; return(p);} // n faktoril
qiymatini hisoblovchi funksiya

void Area(double Radius, double &Result)
{const double Pi=3.1416; Result = Pi * Radius * Radius; } // R
radiusli doira yuzasini hisoblovchi prosedura

void gggg(double d1, double &Result1,double &result2,double
&result3)
```

```
{ Result1 = 3 * d1; result2=7*d1; result3=Result1*result2;} //3 ta natija beruvchi prosedura

int dol_summ(int n) { int s; s = n*3623;return(s); } // dollar qiymatini so‘mga o‘tkazuvchi funksiya

int per(int a, int b) { int p;  p=2*(a+b);   return(p); } // to‘rtburchak perimetrini hisoblovchi funksiya

void powerAA(double d1, double &Result1)  { Result1 = d1* d1*d1;} // sonning kubini hisoblovchi prosedura

using namespace std;

int main()
{int t1,t2; cout<<dol_summ(100)<<endl; cin >>t1; cin >>t2;
cout<<per(t1,t2)<<'\n';

int xx,yy,zz;  cout<< sum(5,8,5)<< endl;
cin>>xx;  cin>>yy;  cin>>zz;  cout<< sum(xx,yy,zz)<< endl;
cout<< sum1(101)<< endl;
cout<< fakn(10)<< endl;
double uza; Area(5,uza); cout<< uza<< endl;

double dd1,dd2,dd3;
gggg(5,dd1,dd2,dd3); cout<<dd1<<'\t'<<dd2<<'\t'<<dd3<<endl;

powerAA(5,dd3); cout<<"javob"<<'\t'<<dd3<<endl;
 return 0;
}
```

Keltirilgan dastur natijasiga ko‘ra dastur tahlilini o‘tkazishni tavsiya qilamiz:



Algoritmlarni loyihalash va yaratishning yana bir usuli modulli usul hisoblanadi. Modul –bu nomga ega biror algoritm yoki uning bir qismi bo‘lib, bu nom orqali uni faollashtirih mumkin. Ko‘p hollarda modulni ham ixtiyoriy algoritmga xos bo‘lgan yordamchi algoritm nomi bilan ham yuritishadi.

Modul xossalari:

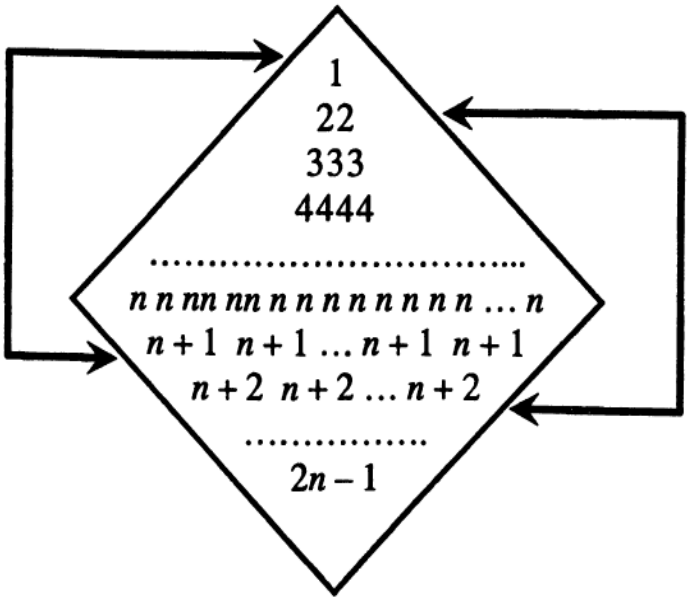
- Funksional butunlik va tugatuvchanlik;
- Avtonom va boshqa modulga bog‘liq bo‘lmaslik;
- Rivojlanuvchanlik;
- Yaratuvchi va foydalanuvchiga ochiqligi;
- Aniqlik va ishonarlilik;
- Moduldan foydalanishda faqat uning nomiga murojaat qilish orqali foydalanish;

Algoritmlarni modulli loyihalashning afzalliklari:

- Katta hajmdgi algoritmlarni loyihalashda qulayligi;
- Ko‘p foydalanadigan algoritmlar kutubxonasini yaratish;
- Loyihalash va moslashtirish jarayonini soddalashtirish;
- Ulardan foydalanish jarayonini kuzatish imkoniyati;

3-masala. Kiritilgan n soni uchun “sonli romb” quyidagicha yaratiladi: 1-qatorda 1 ta 1, 2-qatorda 2 ta 2, 3-qatorda 3 ta 3 va n-chi qator uchun n ta n. So‘ng keyingi qatorlarda sonlar miqdori bittadan kamayib boradi. Bu jarayon bitta son qolguniga qadar davom ettiriladi. Shu sonlar yig‘indisini topish algoritmini tuzing.

Masala shartiga ko‘ra rombdagi sonlarni quyidagicha tas-virlash mumkin:



Masala uchun tuzilgan ikki xil algoritmni ko‘rib chiqamiz.

Birinchi algoritm.

```
...
var i,j,n,k,s:integer;
begin
   readln(n); s:=0;
 for I := 1 to n do
  begin   for j := 1 to i do  begin write(i,#9); s:=s+i;end; writeln;
end;
  k:=n-1;
  while (i<=2*(n-1)+1) do
  begin
  for j := 1 to k do
  begin write(i,#9); s:=s+i;end;
   dec(k,1); i:=i+1; writeln;  end;  writeln(s);  readln;...
```

Bu algoritmda rombda qatnashgan barcha sonlar aniqlanib, chop etiladi. Lekin masala shartiga ko‘ra faqat yig‘indi natija sifatida chop etilishi kerak. Matematik faktlarga asoslangan holda algoritm tarkibidagi buyruqlarni kamaytirib, samarali algoritmni ishlab chiqish mumkin.

Buning uchun sonlar orasidagi quyidagi munosabatlardan foydalanish mumkin.

$$1+(n+1)(n-1)=n^2$$
$$2+2+(n+2)(n-2)=n^2$$
.....
$$(n-1)^2+2n-1=n^2$$
$$S_n = n^2\ n = n^3.$$

Ikkinchi algoritm. Maqsadga erishish yo‘lida eng qulay algoritm hisoblanib, uning tuzilishi quyidagicha:

```
var n,s:integer;
begin
readln(n); s:=n*sqr(n);
writeln(s);  readln;
end.
```

Algoritmlarni yaratish ijobiy ish, shuning uchun ixtiyoriy zarur algoritmlarni tuzish imkonini beradigan bir umumiy usul mavjud emas. Lekin algoritmlarni ishlab chiqishni asoslangan oddiy sxemalarini beradigan ko‘pgina algoritmlashtirish nazariyalari bor. Bunday sxemalar va yangi algoritmlarni paydo qilishning o‘rtasida qattiq bog‘liqlik kuzatiladi. Tez uchraydigan va ko‘p foydalaniladigan usullarni quyidagicha ajratib olish mumkin:

1. **Algoritmlarni konstruksiyalash.** Bu usulda yangi algoritm mavjud algoritmlardan tarkibiy qismlar sifatida foydalanib, bir-biriga moslab bir butunlik hosil qilish yo‘li bilan ishlab chiqiladi.
2. **Algoritmlarni ekvivalent qayta ishlash.** Ikki algoritm ekvivalent hisoblanishi uchun quyidagi shartlar bajarilish kerak:

- Bittasi uchun mumkin bo‘lgan dastlabki berilganlar varianti, ikkinchisi uchun ham mumkin bo‘lishi kerak.
- Bir algoritmni qandaydir dastlabki ma’lumotga qo‘llanilishi, ikkinchi algoritmni ham shu berilganga qo‘llanilishiga kafolat beradi.
- Bir xil dastlabki berilgan ma’lumot uchun ikkala algoritm ham bir xil natija berishi. Lekin bu algoritmni ikki xil shakllarini ekvivalent deb nomlash noto‘g‘ridir.

Shunday qilib, algoritmni ekvivalent qayta ishlash deb, natijada dastlabki algoritmga ekvivalent algoritmni paydo qiladigan o‘zgartirilishlarga aytiladi.

Misol tariqasida, algoritmni bir tildan boshqa tilga o‘tkazishni keltirish mumkin. Shu bilan birgalikda algoritmni ekvivalent qayta ishlash usuli bilan keskin o‘zgartirish mumkin, lekin bu holda asosiy e’tiborni dastlabki algoritmga nisbatan yahshi algoritmni yaratishga berish kerak.

3. Toraytiruvchi o‘zgartirishlar. Bunday o‘zgartirishlar natijasida dastlabki algoritmlar yechish kerak bo‘lgan masalalarning xususiy holati yechimi algoritmlari ishlab chiqiladi. Odatda, bu usulda ekvivalent qayta ishlash jarayonida algoritmni ixchamlashtirish maqsadida foydalaniladi.

4. Formal usulni matematikaga bog‘liq bo‘lmagan muammoga qo‘llash. Bu yerda matematik muammo matematik ko‘rinishga o‘tkazilib, uning algoritmini ishlab chiqishga uriniladi. Agar o‘xshash matematik masala yechimining algoritmi ma’lum bo‘lsa, undan foydalaniladi.

Takrorlash uchun savollar

1. Har bir usul bo‘yicha algoritm tuzishga misol ko‘rsating.
2. Algoritmni ishlab chiqish uchun yana qanday usullarni bilasiz?

7-§. Rekursiya va rekursiv funksiyalar

Boshlang‘ich va oraliq ma’lumotlarni masalani yechish natijasiga aylantiradigan jarayonni bir qiymatli qilib, aniqlab beradigan qoidalarning biror bir chekli ketma-ketligi algoritm ekanligi yuqoridagi mavzulardan bizga ma’lum.

Buning mohiyati shundan iboratki, agar algoritm ishlab chiqilgan bo‘lsa, uni yechilayotgan masala bilan tanish bo‘lmagan biron bir ijrochiga, shu jumladan kompyuterga ham bajarish uchun topshirsa bo‘ladi va u algoritmning qoidalariga aniq rioya qilib masalani yechadi. Quyida pekursiv algoritmga to‘xtalamiz. Avvalo, rekursiv o‘zi nima degan savolga javob beramiz. Agar obyektning tarkibiy qismlari obyektning o‘zi orqali aniqlansa, u holda bu obyekt rekursiv deyiladi. Rekursiya nafaqat matematikada, balki kundalik hayotimizda ham ushrab turadi.

Rekursiya o‘z kuchini ayniqsa, matematik ta’riflarda namoyon etadi. Eng tanish misollar sifatida natural sonlar, daraxtsimon tuzilmalar va ba’zi funksiyalarni keltirishimiz mumkin:

1. Natural sonlar:

a) 0 natural son hisoblanadi.

b) Natural sondan keyin keluvchi son natural hisoblanadi.

2. Daraxtsimon tuzilmalar:

a) $\varnothing$ daraxt hisoblanadi (va bo‘sh daraxt deyiladi).

b) Agar t1 va t2 –daraxtlar bo‘lsa, t1 va t2ning avlodlaridan iborat tugundan tashkil topgan tuzilma ham daraxt deyiladi (ikkilik yoki binar daraxt).

3. Faktorial funksiya f(n):

f(0)=1

n>0 bo‘lganda f(n)///

Ko‘rinib turibdiki, rekursiyaning kuchi cheksiz obyektlar to‘plamini chekli mulohaza orqali aniqlash imkoniyatida namoyon bo‘ladi. Xuddi shunday chekli sonli hisoblashlarni chekli dastur orqali tavsiflash mumkin. Bunda dastur oshkor sikllarni o‘z ichiga olmasligi ham mumkin. Ammo rekursiv algoritmlar, avvalo, yechilayotgan masala, hisoblanayotgan funksiya yoki qayta ishlanayotgan ma’lumotlar tuzilmasi rekursiv ravishda berilgan holda o‘rinlidir.

Umumiy holda, R rekursiv dastur (R ni o‘z ishiga olmagan) S ko‘rsatmalarning va R ni o‘zining ketma-ketligidan iborat R birlashma (kompozitsiya) sifatida ifodalanishi mumkin:

R=...

Dasturlarning rekursiv ifodasi uchun zaruriy va yetarli vosita bo‘lib protseduralar xizmat qiladi. Bu protseduralar ko‘rsatmalar to‘plamiga nom berish imkonini yaratib, bu nom orqali dastur bajarilish jarayonida ko‘rsatmalar chaqirilishi mumkin.

Agar R protsedura oshkor holda o‘ziga murojaatni o‘z ichiga olsa, bunday protsedura oshkor rekursiv deyiladi. Agar R protsedura R ga to‘g‘ridan-to‘g‘ri yoki bilvosita murojaatni o‘z ichiga olgan

boshqa bir Q prosedurani tarkibiga olsa, u holda R bilvosita rekursiv deyiladi.

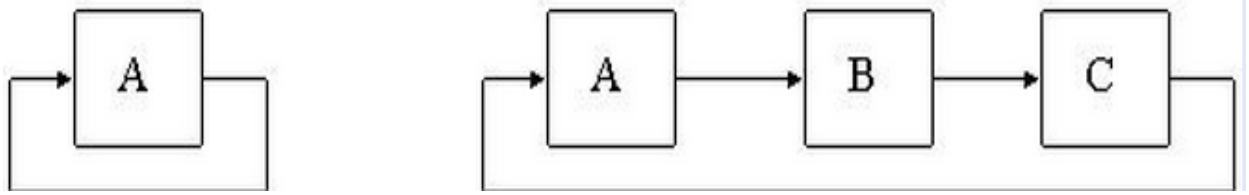
Rekursiv protseduralar cheksiz hisoblashlarga yo‘l ochganligini hisobga olsak, uni to‘xtatish muammosini ham hal etishimiz zarur. Fundamental talablar shundan iboratki, qaysidir vaqt momentiga kelib, bajarilmay qoladigan shunday V shart bajarilgandagina R rekursiv protseduraga murojaatlar amalga oshirilishi kerak.

Shuning uchun ham rekursiv algoritmlarning sxemasini quyidagi ko‘rinishlardan biri bilan ifodalash mumkin:

P=IF B THEN P[S,P] END
P=…

Ma’lumki, har qanday prosedura yoki funksiya boshqa protsedura yoki funksiyalarga murojaatni o‘z ichiga olishi mumkin. Xususiy holda, agar bu murojaat protsedura yoki funksiyaning o‘ziga murojaatdan iborat bo‘lsa, bu protsedura yoki funksiya rekursiv deyiladi.[19]

Umumiy holda, rekursiv prosedura va funksiyalarni quyidagi sxema ko‘rinishida ifodalashimiz mumkin:



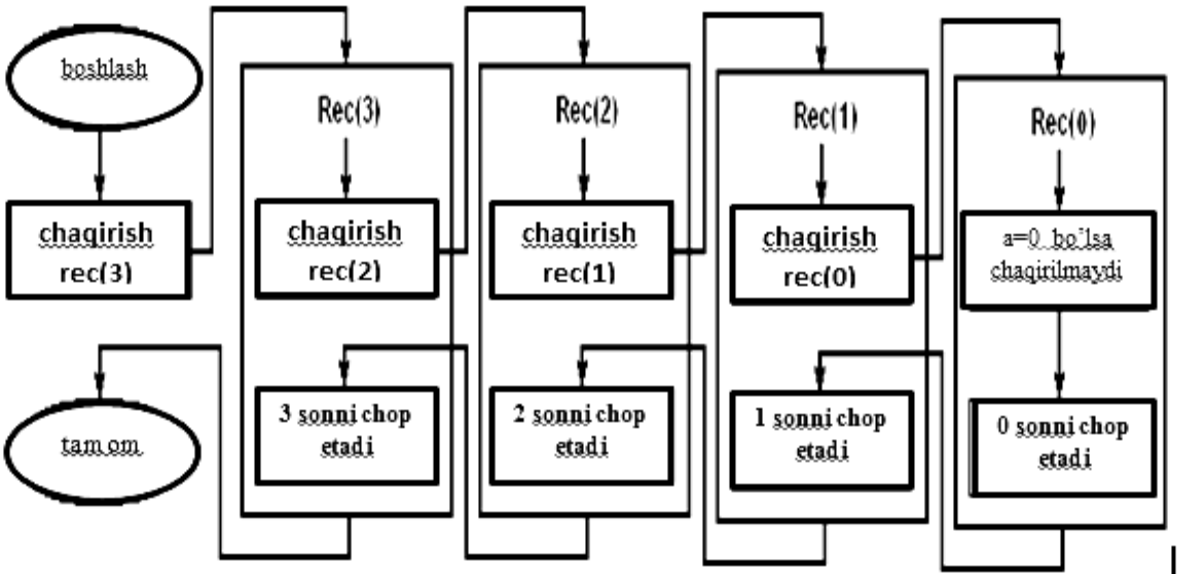
Rekursiv protseduraga misol sifatida quyidagi qism dasturni tahlil etaylik:

```
prosedure Res(a: integer);
begin
 if a>0 then
 Res(a-1);
 writeln(a);
end;
```

[19] Н.Вирт. Алгоритмы и структуры данных. Москва – 2010 (132-136с)

Tushunib olish uchun asosiy dasturdan Rec(3) ko‘rinishdagi murojaatda protsedura qanday ishlashini ko‘rib chiqaylik.

Quyidagi blok-sxemada operatorlarning bajarilish ketma-ketligi tasvirlangan:



Boshlanish Rec(3)ga murojaat Tugash

1-misol: Rekursiv protsedura ishini tasvirlovchi blok-sxema.

Rec protsedurasi dastlab a=3 parametr bilan chaqiriladi, ya’ni Rec(3) tarzidagi murojaat amalga oshiriladi. Uning tarkibida esa a=2 parametrli Rec(2) tarzidagi murojaat mavjud. Hali dastlabki murojaat tugamasdan keyingi protseduraga murojaat tashkil etiladi, u tugamasa dastlabki Rec(3) protsedura ishini tugatmaydi. Protseduraga murojaat a=0 ga davom etadi. Demak, bir vaqtning o‘zida protsedura 4 marta chaqirilib ishlaydi. Bir vaqtda bajariladigan protseduralar soni rekursiya chuqurligi deyiladi.

To‘rtinchi marta chaqirilgan Res(0) protsedura 0 sonini chop etadi va o‘z ishini yakunlaydi. Shundan so‘ng boshqaruv Res(1)ni chaqirgan protseduraga o‘tadi va 1 ni chop etadi. Xuddi shu tariqa barcha protseduralar tugaguncha jarayon davom etadi. Natijada to‘rtta 0, 1,2,3 sonlarini chop etiladi.

2-misol. Sonni ikkilik sanoq sistemasiga o‘tkazish masalasini sikl operatori va rekursiya orqali hal etishni qarab chiqamiz.

Ma’lumki, ikkilik sanoq sistemadagi sonlarni hosil qilish uchun berilgan sonni sanoq sistema asosi bo‘lan 2 ga bo‘lish orqali amalga oshiriladi.

Agar berilgan son x bo‘lsa, uning ikkilik sistemadagi ifodasida oxirgi raqam *S1=x mod 2*

Ikkiga bo‘lishda bo‘linmaning butun qismini olamiz:

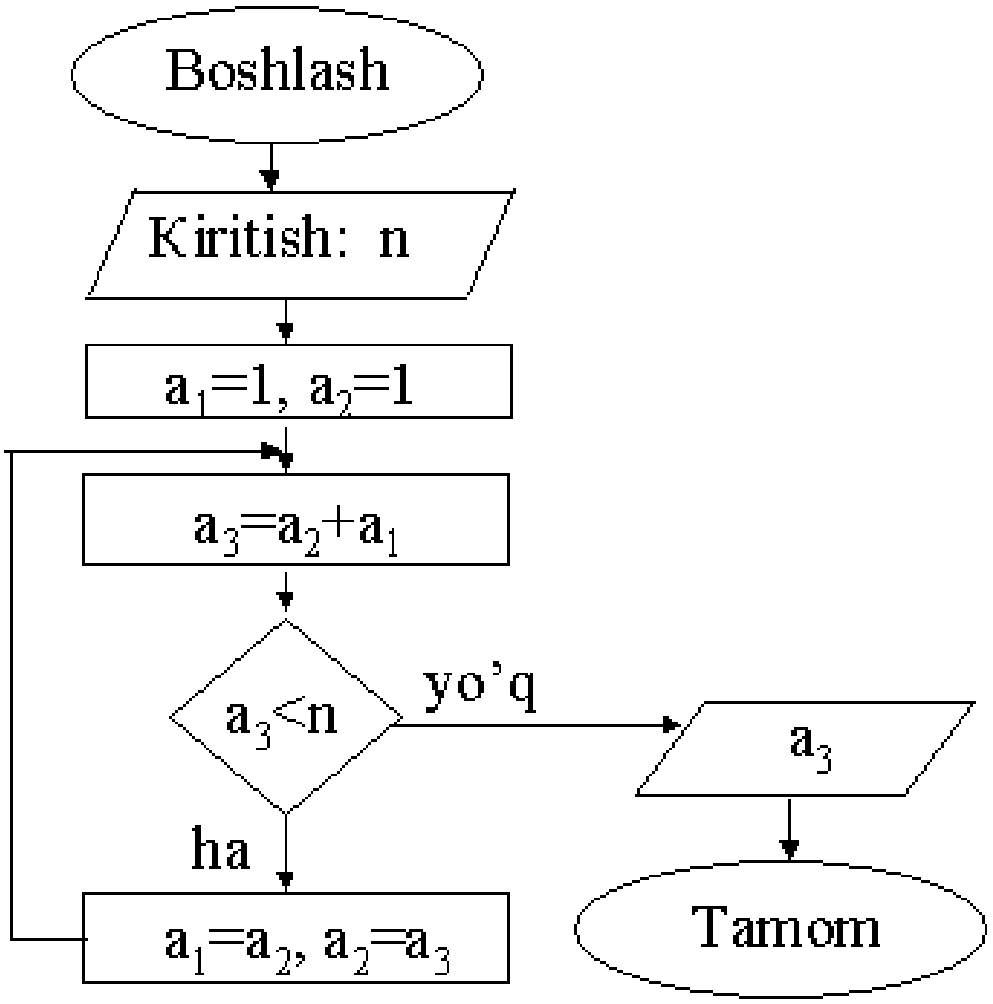
X2=x div 2

Bu jarayon butun qism nolga teng bo‘lguncha davom etadi. Bu amalda qoldiq 0 ga teng bo‘lguncha davom etadi. Dastur rekursiyadan foydalanilgan holda quyidagi ko‘rinishda bo‘ladi:

```
prosedure BinarRepresentation(x: integer);
var
s, x: integer;
begin
{Birinchi chaqiruv.Prosedura murojaat qilinish tartibida amalga oshiriladi}
s := x mod 2;
x := x div 2;
{Rekursiv murojaat}
if x>0 then
BinarRepresentation(x);
{Ikkinchi blok. Teskari tartibda bajariladi}
write(s);
end;
```

Bunday algoritmga yana *misol* sifatida Fibonashchi sonlarini keltirish mumkin. Ma’lumki, Fibonashchi sonlari quyidagicha aniqlangan.

$a_0 q a_1 q 1$, $a_i q a_{i-1} + a_{i-2}$ $iq2,3,4,....$ Bu rekkurent ifoda algoritmiga mos keluvchi blok-sxema keltirilgan. Eslatib o‘tamiz formuladagi *i*-indeksga hojat yo‘q, agar Fibonachchi sonining nomerini ham aniqlash zarur bo‘lsa, birorta parametr-kalit kiritish kerak bo‘ladi.



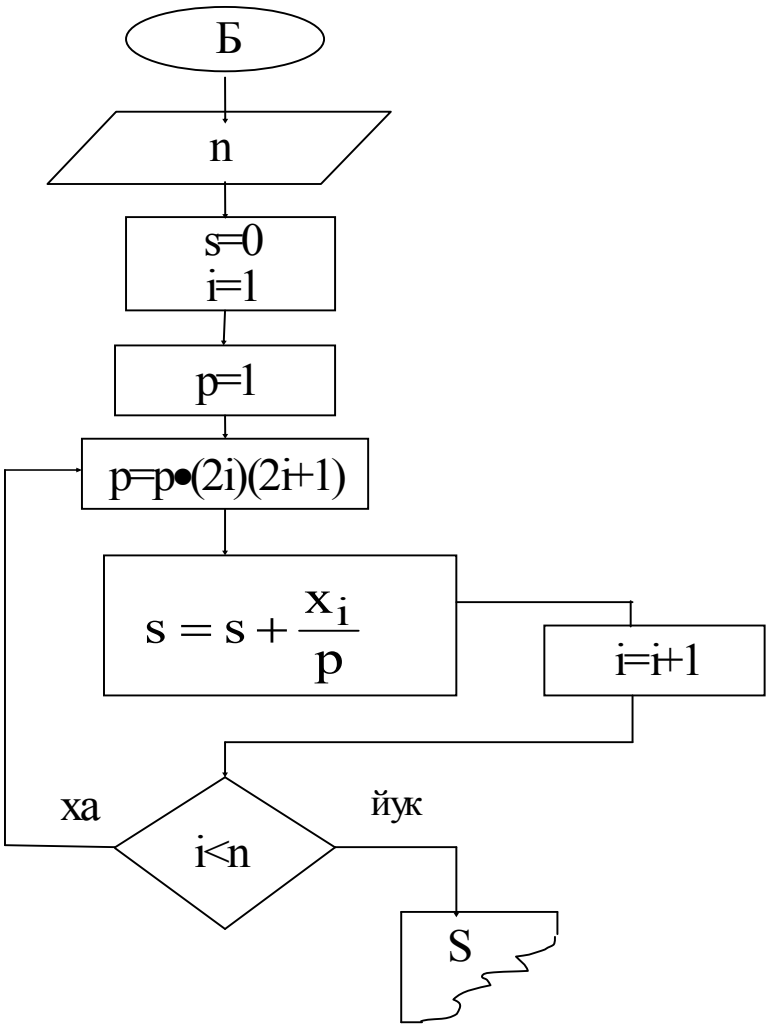
Fibonachchi sonlarining *n*- hadini hisoblash algoritmi.

3-misol. $S = \sum_{i=1}^{n} \frac{x_i}{(2i+i)!}$

Bu ifoda i ning har bir qiymatida faktorialni va yig‘indini hisoblashni taqozo etadi. Shuning uchun avval faktorialni hisoblashni alohida ko‘rib chiqamiz. Quyidagi rekkurent ifoda faktorialni kam amal sarflab qulay usulda hisoblash imkonini beradi:

R=1
R=R*2i*(2i+1)

Haqiqatan ham, i=1 da 3! ni, i=2 da R=3!*4*5=5! ni va hokazo tarzda (2i+1)! ni yuqoridagi rekkurent formula yordamida hisoblash mumkin bo‘ladi. Bu misolga mos keluvchi blok-sxema quyida keltirilgan.



Nazorat savollari va topshiriqlar

1. $S = \sum_{i=1}^{n} \frac{x_i}{(2i+i)!}$ ifodani hisobdash uchun rekkurent ifodasini keltiring.
2. Quyidagi misolga algoritm tuzing: $S = \sum_{i=0}^{3} (2i+3)!$
3. Quyidagi misolga algoritm tuzing: $Y = \sum_{i=1}^{3} \frac{(2i+2)!}{(3i+4)!}$
4. $S = \sum_{i=0}^{3} (2i+3)!$ hisoblansin.
5. $Y = \sum_{i=1}^{3} \frac{(2i+2)!}{(3i+4)!}$ hisoblansin.

8-§. Qidiruv usullari: binar qidiruv, fibonachchi qidiruv, binar daraxt bo‘yicha qidiruv

To‘plamni tashkil etuvchi predmet(obyekt)lar to‘plam elementlari deyiladi. To‘plam elementlari kalit deb nomlanadi va indeksida element tartib raqami ko‘rsatilib turuvchi lotin alifbosining ***“k”*** harfi bilan belgilanadi.

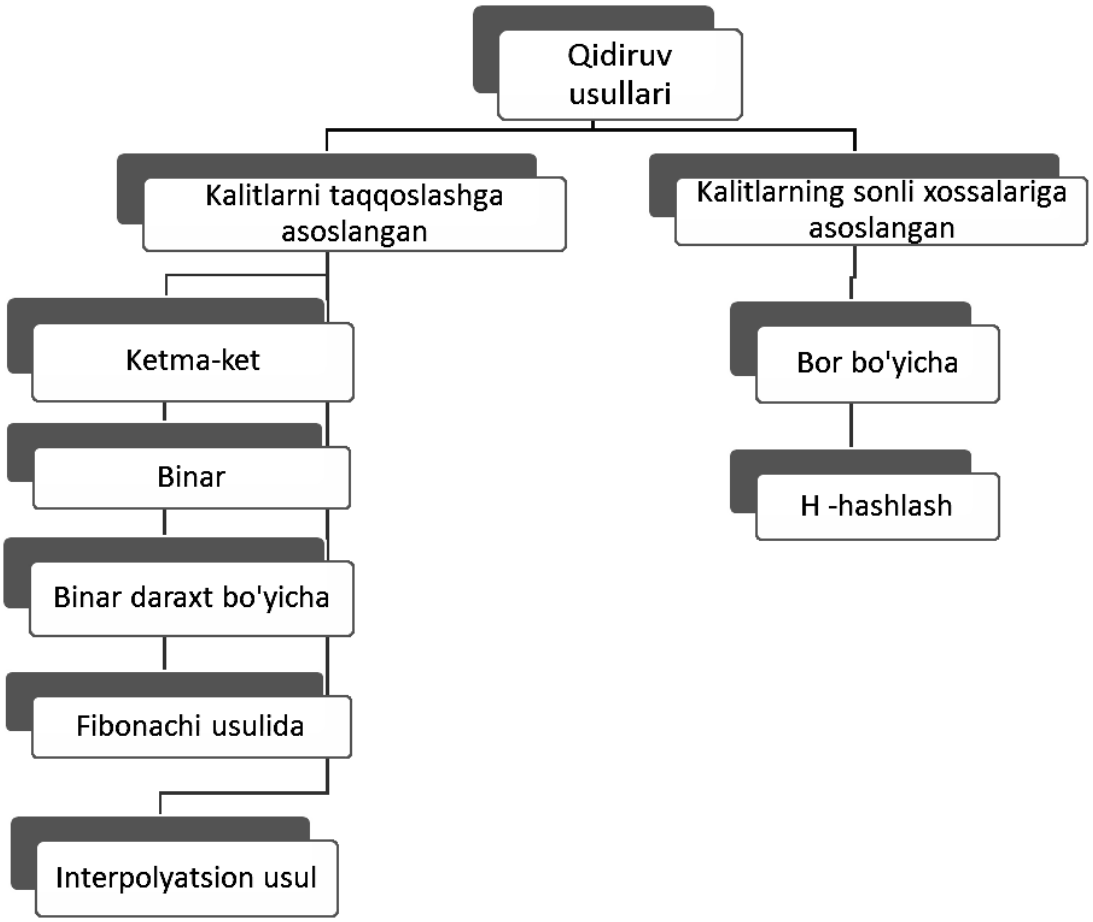
Qidiruv algoritmlarini quyidagi guruhlarga ajratish mumkin (10-rasm).

Qidiruv masalasi.

$\{k_1, k_2, k_3, \dots, k_n\}$ kalit to‘plam berilgan bo‘lsin. To‘plamdan k_i kalitni topish kerak. Qidiruv jarayoni 2 xil holda tugatilishi mumkin:

1)Kalit to‘plamda mavjud bo‘lmasa;

2)Kalit to‘plamda topilsa;[20]



2-chizma. Qidiruv usullari

[20] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 78-79 сс.

Yozuvlarni oddiy ko‘rib chiqish usuli. Bu usulni quyidagi algoritm yordamida realizatsiya qilish mumkin:

```
function search(x: integer): integer;
var
i: integer;
begin
for i:=1 to n do
begin
if x = a[i] then
begin
search := i;
exit;
end;
end;
search:=0;
end;
```

Ketma-ketlik usulida qidiruv

Ketma-ketlik qidiruv usulida dastlabki to‘plam tartiblanmagan, ya’ni $\{k_1, k_2, k_3, \dots, k_n\}$ ixtiyoriy kalitlar jamlanmasi mavjuddir. Qidiruv usuli shundan iboratki, qidirilayotgan k_i kalit navbatma-navbat to‘plamning barcha elementlari bilan taqqoslanadi. Kalit topilishi bilanoq qidiruv jarayoni to‘xtatiladi.[21]

Binar qidiruv.

Binar qidiruv usulida berilgan to‘plam o‘sish tartibida tartiblangan bo‘lishi kerak. Boshqacha aytganda, har bir keyingi kalit o‘zidan oldingisidan katta bo‘ladi.

$$\{k_1 \le k_2 \le k_3 \le k_4 \dots \le k_{n-1} \le k_n\}.$$

Qidirilayotgan kalit to‘plamning markazidagi markaziy element bilan taqqoslanadi, agar u markaziy elementdan kichik bo‘lsa, u holda qidiruv jarayoni chap tomondagi qism to‘plamda davom

[21] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 79-с.

etadi, aks holda binar qidiruv usulida markaziy element tahlil qilinadi.

Markaziy element tartib raqami quyidagi formula orqali topiladi:

$$\textbf{\textit{Element tartib raqami}}\ № = \left[n/2\right] + 1,$$

Bu yerda kvadrat qavslar bo‘linmadan faqat butun qism olinishini bildiradi. Binar qidiruv usulida faqatgina markaziy element tahlil qilinadi.

1-misol.

$$\{7, 8, 12, 16, 18, 20, 30, 38, 49, 50, 54, 60, 61, 69, 75, 79, 80, 81, 95, 101, 123, 198\}$$

to‘plam berilgan. To‘plamdan ***K=61*** kalit topilsin.

Birinchi qadam.

$$\textbf{\textit{Element tartib raqami}}\ № = \left[n/2\right] + 1 = [22/2] + 1 = 12.$$

$K \sim k_{12}$. **61>60.** Keyingi qidiruv o‘ng qism to‘plamda olib boriladi:
$\{61, 69, 75, 79, 80, 81, 95, 101, 123, 198\}$.

“$\sim$” belgisi elementlarni (son, qiymatlarini) taqqoslanmasini bildiradi.

Ikkinchi qadam.

$$\textbf{\textit{Element tartib raqami}}\ № = \left[n/2\right] + 1 = [10/2] + 1 = 6.$$

$K \sim k_{18}$. **61<81.** Keyingi qidiruv chap qism to‘plamda (oldingi qism to‘plamga nisbatan) olib boriladi:
$\{61, 69, 75, 79, 80\}$.

Uchinchi qadam.

$$\textbf{\textit{Element tartib raqami}}\ № = \left[n/2\right] + 1 = [5/2] + 1 = 3.$$

$K \sim k_{15}$. **61<75.** Keyingi qidiruv chap qism to‘plamda olib boriladi:
$\{61, 69\}$.

To‘rtinchi qadam.

$$\textbf{\textit{Element tartib raqami}}\ № = \left[n/2\right] + 1 = [2/2] + 1 = 2.$$

$K \sim k_{14}$. **61<69.** Keyingi qidiruv chap qism to‘plamda olib boriladi:
$\{61\}$.

Beshinchi qadam.

$$\textbf{\textit{Element tartib raqami}}\ № = \left[n/2\right] + 1 = [1/2] + 1 = 1.$$

$K \sim k_{13}$. **61=61.**

Xulosa shundan iboratki, qidirilayotgan kalit tartib raqami 13 ga teng ekan.

Ushbu algoritmning mohiyati, saralangan massivda massiv o‘rtasi qidirilishidan iborat. Agar izlangan element massiv o‘rtasidagi elementdan kichik bo‘lsa, chap tomondan izlaymiz, katta bo‘lganda esa o‘ng tomondan izlanadi. Topilgan intervalda yana o‘rtacha element izlanadi va taqqoslash bajariladi va h.k.z.

```
Type fun= function (j:word):Boolean;
Function fa(j:word):Boolean; far; Begin fa:= A[j] < x1 End;
Function fb(j:word):Boolean; far; Begin fb:= A[j] > x2 End;
Procedure Search(f1,f2:fun; L:integer; var m,k,usp:word);
Var i,j,kk:word;
Begin usp:=1; i:=m+1; kk:=k;
If L<>1 Then
Begin {f2 funksiyasi bo‘yicha binar izlash sikli: }
Repeat j:=(i+kk) div 2; If f2(j) Then kk:=j Else i:=j+1
Until i=kk; If i=m+1 Then usp:=0
Else If (L > 1) And f1(i-1) Then usp:=0
End; {i – argumentdga eng yaqin katta kalitli topilgan yozuv
nomeri} If usp = 0 Then Exit;
If (L > 2) Or (L = 1) Then
Begin i:=kk-1; {izlanayotgan yozuvlarni pastdan chegaralaydigan
yozuv topiladi;f1 funksiyasi bo‘yicha binar izlash sikli}
Repeat j:=(m+i+1) div 2; If f1(j) Then m:=j Else i:=j-1
Until m=i;
If L=1 Then Begin If m=k-1 Then usp:=0; i:= m+1 End
End
Else If (L=-1) Or Not f1(i-1) Then i:=i-1;
m:=i; k:=kk
End;
```

L = -1 (L = 1) bo‘ladi, pastdan(yuqoridan) yaqinlashish orqali izlash holida; L = 2 bo‘ladi, mos tushish bo‘yicha izlash holida(bitta yozuv); L = 3 bo‘ladi, barcha yozuvlar mos tushishi bo‘yicha izlash holida.

L = 3 va usp = 1 (izlash jarayoni muvaffaqiyatli) bo‘lganda topilgan m (eng kichik), k (eng katta) lar izlanayotgan yozuvlar guruhlariga qo‘shni pozitsiyalarni belgilaydi, qolgan hollarda, ya’ni usp = 1 ("muvaffaqiyat") bo‘lganda, topilgan yozuv nomeri m dan iborat bo‘ladi.

Izlash argumenti oshkor ko‘rsatilmagan. Bu argument foydalanuvchi tomonidan yoziladigan mantiqiy f1, f2 bitta j yozuv nomeridan iborat bo‘lgan parametrli funksiyalarda berkitilgan. fl (f2) da yozuv kaliti argumentdan kichik (katta) bo‘lganda f1 (f2) rost deb yoziladi. m, k – yozuvlarning natijaviy nomerlarigina bo‘lmasdan, izlash natijaviy sohasining tashqi chegaralari hamdir.

Fibonachchi usulida qidiruv

Ushbu qidiruv usulida Fibonachchi sonlariga teng positsiyalarda joylashgan elementlar tahlil qilinadi. Fibonachchi sonlari quyidagi qoidaga ko‘ra hosil bo‘ladi:
har bir keyingi son oldingi ikkita sonlarning yig‘indisiga teng, masalan

$$\{1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots\}.$$

Qidiruv jarayoni ikkita kalitlar orasidagi qidirilayotgan kalit joylashgan oraliq topilgunigacha davom etadi.

2-misol.

$$\{3, 5, 8, 9, 11, 14, 15, 19, 21, 22, 28, 33, 35, 37, 42, 45, 48, 52\}$$

dastlabki kalit to‘plam berilgan. Qidirilayotgan kalit 42 ga teng bo‘lsin. Qidirilayotgan kalitning taqqoslash ketma-ketligi Fibonachchi sonlariga teng $\{1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots\}$ pozitsiyalarda olib boriladi.

Birinchi qadam. $K \sim k_1$. **42>3=>** Qidirilayotgan kalit Fibonachchi sonlariga teng pozitsiyalarda joylashgan kalit bilan taqqoslanadi.

Ikkinchi qadam. $K \sim k_2$. **42>5=>** Qidiruv jarayoni Fibonachchining keyingi sonining pozitsiyasiga teng kalit bilan taqqoslanadi.

Uchinchi qadam. $K \sim k_3$. **42>8=>** Taqqoslash davom ettiriladi.

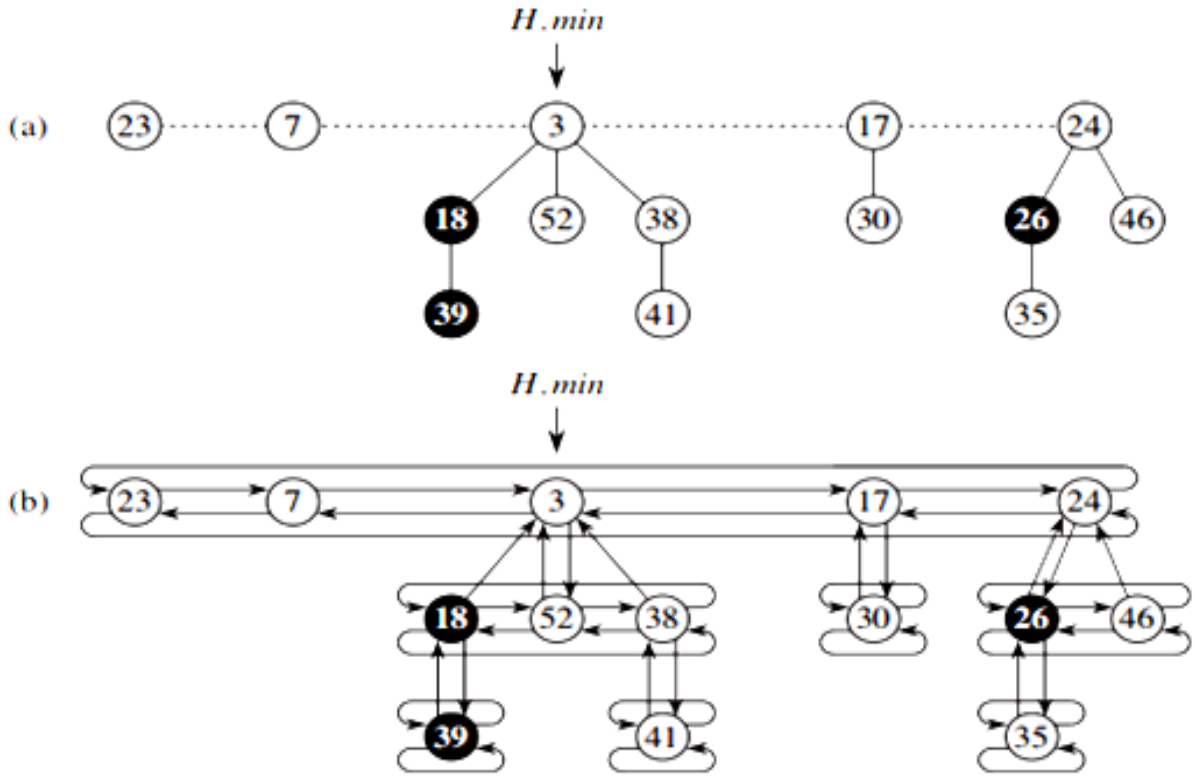
To‘rtinchi qadam. $K \sim k_5$. **42>11=>** Taqqoslash davom ettiriladi.

Beshinchi qadam. $K \sim k_8$. **42>19=>** Taqqoslash davom ettiriladi.

Oltinchi qadam. $K \sim k_{13}$. **42>35=>** Taqqoslash davom ettiriladi.

Yettinchi qadam. $K \sim k_{18}$. 42>52=> 13 dan 18 gacha pozitsiyadan iborat qidirilayotgan kalit joylashgan oraliq topildi: {35,37,42,45,48,52}.

Topilgan oraliqda qidiruv jarayoni yana Fibonachchi sonlariga teng pozitsiyalarda davom ettiriladi. Fibonachchi piramidasi (Fibonacci heap) kamaymovchi piramida xossasiga tartiblangan ravishda mos bo‘lib, ildiz daraxtlar jamlanmasini ifodallaydi. Har bir daraxt ushbu xossalarga bo‘ysunadi. Quyidagi rasmda Fibonachchi piramidasi tasvirlangan.



Interpolyatsiya usulida qidiruv

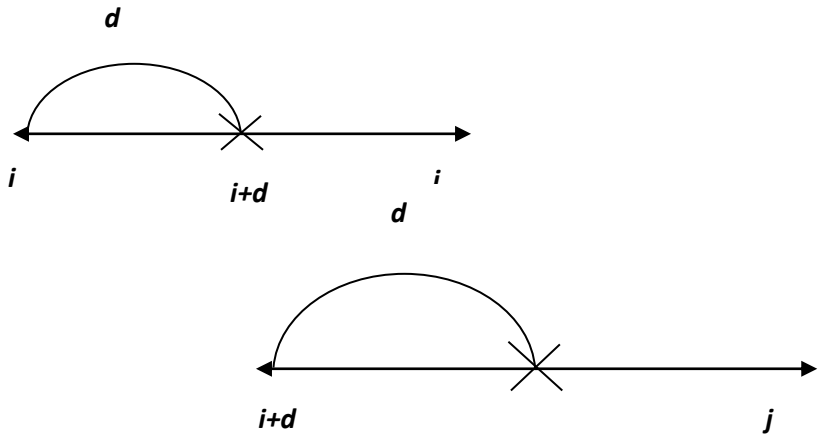
Dastlabki to‘plam o‘sish tartibida tartiblangan bo‘lishi kerak. Boshlang‘ich taqqoslash quyidagi formula orqali aniqlanuvchi ***d*** qadamga teng masofada amalga oshiriladi:

$$d = \left[\frac{(j-i)(K-K_i}{K_j-K_i}\right],$$[22]

bu yerda ***i***- birinchi qaraladigan element tartib raqami; ***j***- oxirgi qaraladigan element tartib raqami; ***k***- qidirilayotgan kalit, K_i, K_j –*i* va *j* pozitsiyalardagi kalit qiymatlari, [ ] - sonning butun qismi.

Qidiruv usulining g‘oyasi shundan iboratki, ***d***- qadam yuqorida keltirilgan formulaning har bir bosqichlarida o‘zgaradi. (13-chizma).

Algoritm ***d=0*** bo‘lganda to‘xtaydi, shu bilan birga qo‘shni elementlar tahlil qilinadi. Shundan so‘ng qidiruv natijalari bo‘yicha xulosa qilinadi.



13-chizma. Qidiruv algoritmi

Agar to‘plam arifmetik progressiyani tashkil qilsa, ushbu usul samarali natija beradi.

3-misol.

{2, 9, 10, 12, 20, 24, 28, 30, 37, 40, 45, 50, 51, 60, 65, 70, 74, 76}

kalit to‘plam berilgan bo‘lsin. Qidirilayotgan kalit 70 ga (***K=70***) teng bo‘lsin.

Birinchi qadam. Dastlabki kalit to‘plam uchun ***d*** qadamni topamiz:

[22] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 81-82 сс.

$$d = \left[\frac{(18-1)(70-2)}{76-2}\right] = 5$$

Berilgan to‘plamda qidirilayotgan kalit bilan joylashgan 16-tartib raqamda turgan kalitni taqqoslaymiz:

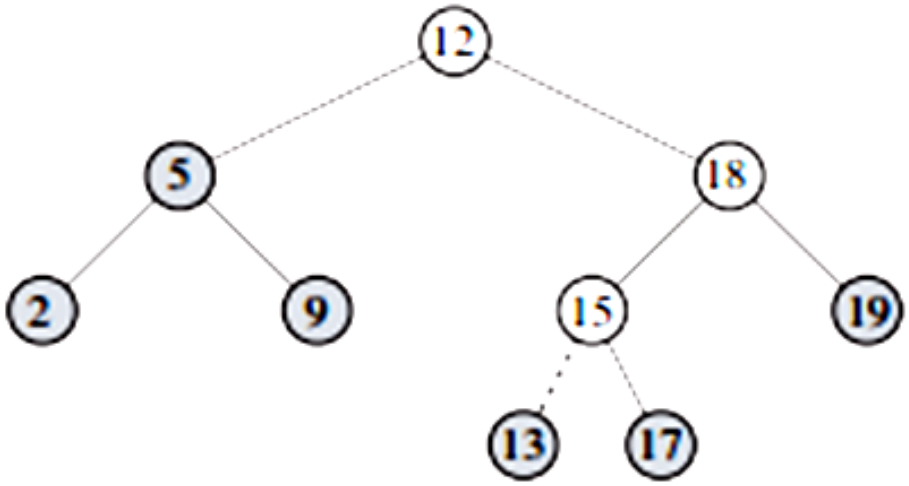
$K_{16} \sim K,\ 70 = 70$ kalit topildi.

Binar daraxt bo‘yicha qidiruv usuli

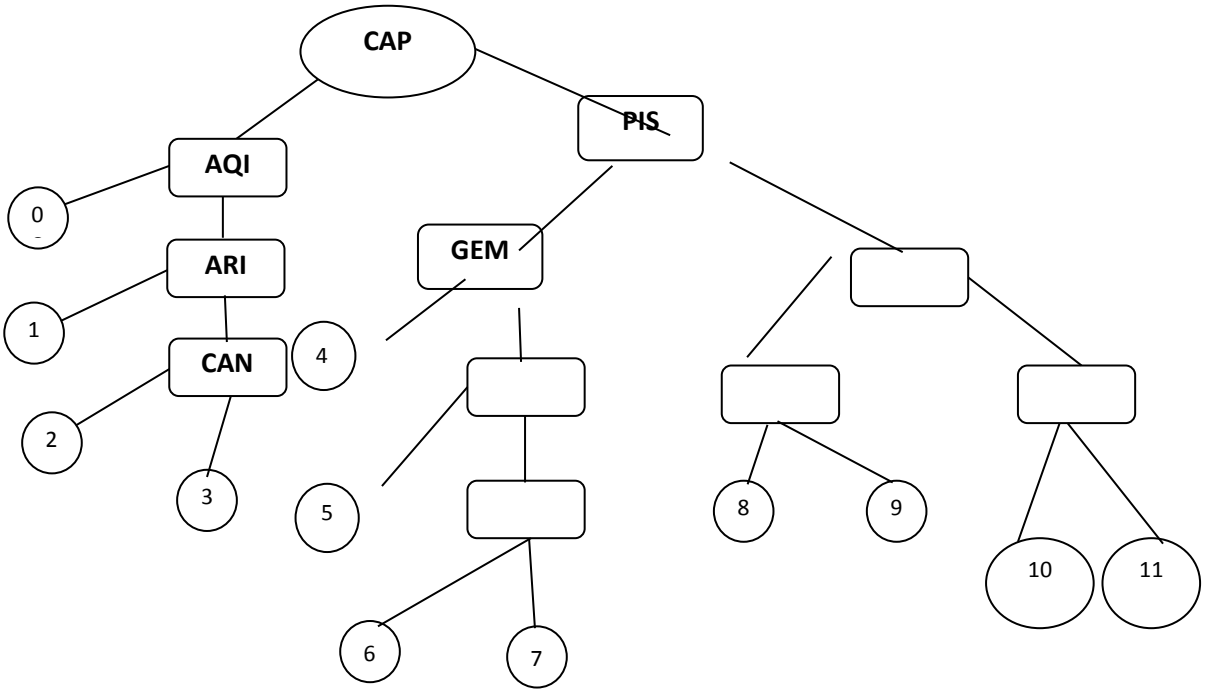
Binar daraxt strukturasini qo‘llash yozuvlarni tez qo‘yish va o‘chirish imkoniyatini beradi va jadvalda samarali qidiruvni amalga oshiradi. Biz yangi v value T binar daraxtga joylashtirishimiz uchun **TREE-Insert** prosedurasini qo‘llaymiz. [23]

```
TREE-INSERT(T, z)
 1  y = NIL
 2  x = T.root
 3  while x ≠ NIL
 4        y = x
 5        if z.key < x.key
 6              x = x.left
 7        else x = x.right
 8  z.p = y
 9  if y == NIL
10        T.root = z          // tree T was empty
11  elseif z.key < y.key
12        y.left = z
13  else y.right = z
```

[23] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 294-295-pp.



Qidiruvning binar daraxti berilgan bo‘lsin. (14-chizma)



14 a)-chizma. Binar daraxt

Binar daraxtdan SAG kalitni topish talab etilsin. Daraxt ildizidan ko‘rinib turibdiki, lotin alifbosining 1 chi harfidan SAG nomi CAP dan ko‘ra katta. Albatta, keyingi qidiruv o‘ng shoxdan amalga oshiriladi. Ushbu so‘z PIS dan ko‘ra katta, demak, yana o‘ng tomonga yuramiz; u TAU dan ko‘ra kichik, endi chap tomonga yuramiz; u SCD dan kichik va 8 chi tugunga kelamiz. Shunday qilib, SAG nomi 8 chi tugunda joylashgan bo‘lishi kerak.

Shu bilan birga daraxt tugunlari quyidagi strukturaga ega: (1-jadval).

Daraxt tugunlari strukturasi

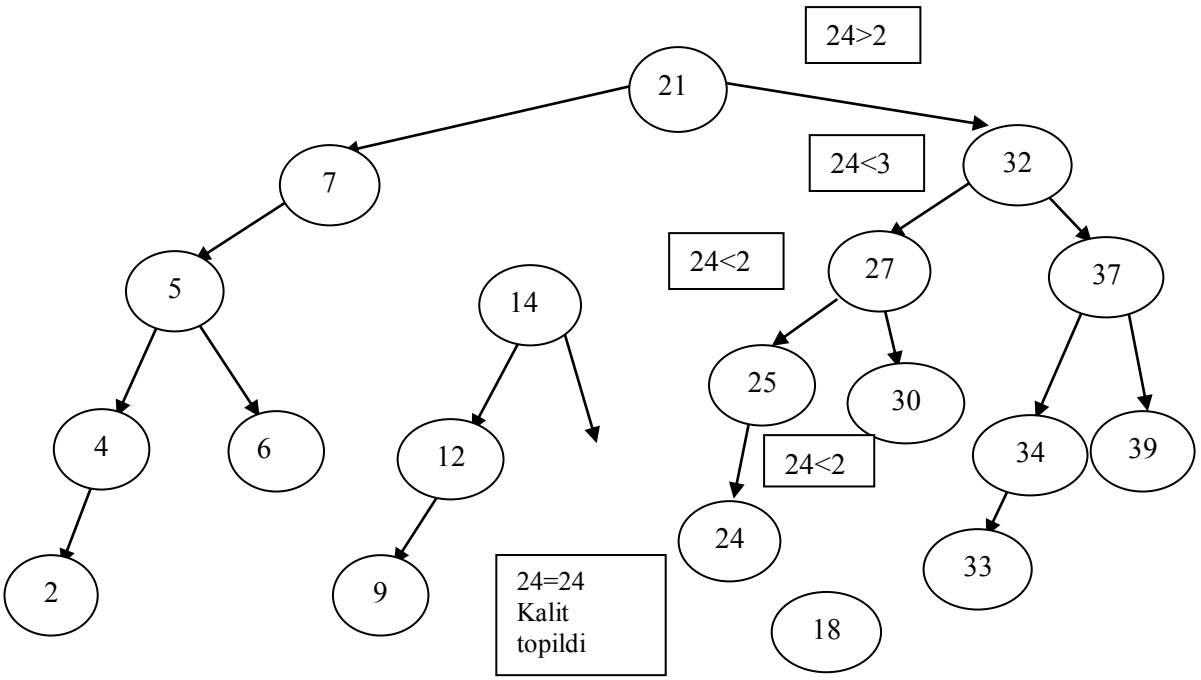
1-jadval.

Kalit	**Axborot soati**	**Chap qism daraxt ko‘rsatkichi**	**O‘ng qism daraxt ko‘rsatkichi**
	Mavjud bo‘lmas-ligi mumkin	LLINK	RLINK
KEY			

4-misol. Dastlabki to‘plam kaliti o‘sish tartibida tartiblangan bo‘lishi kerak. Chiziqli ro‘yxatdan to‘plamning markaziy elementi daraxtning ildizi bo‘lgan binar daraxt bo‘yicha qidiruviga o‘tamiz. (15-chizma).

$$N_{markaz} = [N/2] + 1,$$

bu yerda ***N***- to‘plam elementlar soni. [24]



15-chizma. Binar daraxt bo‘yicha qidiruv

[24] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 83-с.

Chap tomondagi shoxning uchi chap qism to‘plamning markaziy elementi hisoblanadi; o‘ng tomondagi o‘ng qism to‘plamniki bo‘ladi.

$$\{2, 4, 5, 6, 7, 9, 12, 14, 18, 21, 24, 25, 27, 30, 32, 33, 34, 37, 39\}$$

to‘plam berilgan bo‘lsin.

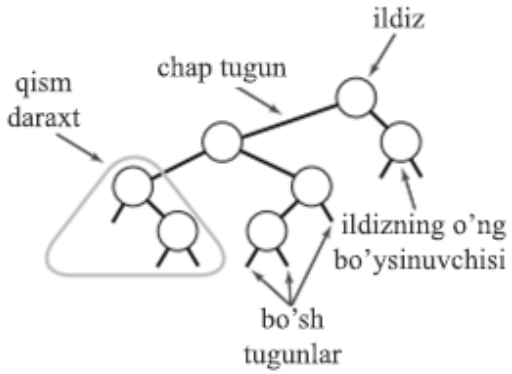
Qidirilayotgan kalit ***K=24***;

Barcha elementlar $N = 19$; $N_{markaz} = [19/2] + 1 = 10$.

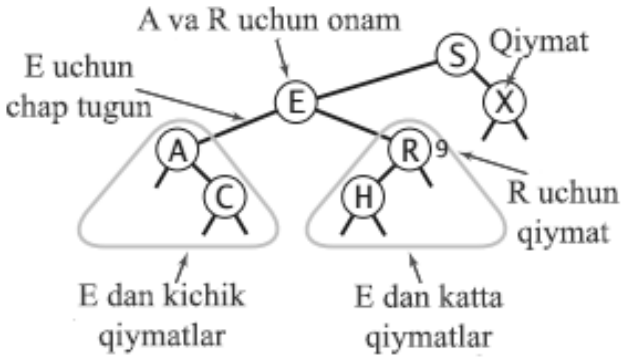
Binar daraxti bo‘yicha ***K=24*** qidiruv kaliti ildizdan barglargacha 16-chizmada ko‘rsatilgan.

Binar daraxtda chegaralar mavjud: har bir tugunga faqat bitta uning ota-onasi bo‘lgan boshqa tugun ko‘rsatilishi mumkin, har bir tugunda esa ikkita teng silkalar mavjud bo‘lib, chap va o‘ng silkalar deb nomlanadi. Ular esa mos ravishda chap va o‘ng tugunlarni ko‘rsatadi. Shu tariqa binar daraxtni nolinchi silka yoki har biri qism binar daraxt bo‘lgan daraxtni ko‘rsatuvchi chap va o‘ng silkali tugun sifatida qarashimiz mumkin.

Ta’rif. Binar daraxt bo‘yicha qidiruv- bu binar daraxt har bir Comparable tipli kalitdan iborat tugun (va u bilan qiymatlar bog‘langan) va ixtiyoriy tugundagi kalit chap qism daraxtdagi barcha kalitlardan katta va o‘ng qism daraxtdagi barcha kalitlardan kichik shartni qanoatlantiradi. Quyidagi rasmda biz kalitlarni tugunga joylashtiramiz. Har bir tugunning kalta kesmalar orqali ifodalangan nolinchi silkadan tashqari barcha i silkalari uni tugunlar bilan bog‘laydi. Odatdagidek, bizning misolimizda bir simvolli kalitlardan foydalanamiz.



Binar daraxt strukturasi



Binar daraxt bo‘yicha qidiruv strukturasi

Dastlab biz berilganlar strukturasini klassik ta’rifini va get( ) va put( ) (joylashtirish) metodlarini ko‘rib chiqamiz.

Binar daraxt bo‘yicha qidiruv asosida nom jadvali[25]

ALGORITHM 3.3 Binary search tree symbol table

```
public class BST<Key extends Comparable<Key>, Value>
{
   private Node root;               // root of BST

   private class Node
   {
      private Key key;              // key
      private Value val;            // associated value
      private Node left, right;     // links to subtrees
      private int N;                // # nodes in subtree rooted here

      public Node(Key key, Value val, int N)
      {  this.key = key; this.val = val; this.N = N; }
   }

   public int size()
   {  return size(root);  }

   private int size(Node x)
   {
      if (x == null) return 0;
      else           return x.N;
   }

   public Value get(Key key)
   // See page 399.

   public void put(Key key, Value val)
   // See page 399.

   // See page 407 for min(), max(), floor(), and ceiling().
   // See page 409 for select() and rank().
   // See page 411 for delete(), deleteMin(), and deleteMax().
   // See page 413 for keys().

}
```

[25] Robert Sedgewick and Kevin Wayne. Algorithms. FOURTH EDITION. Princeton University. First printing, March 2011. Pp 396-400.

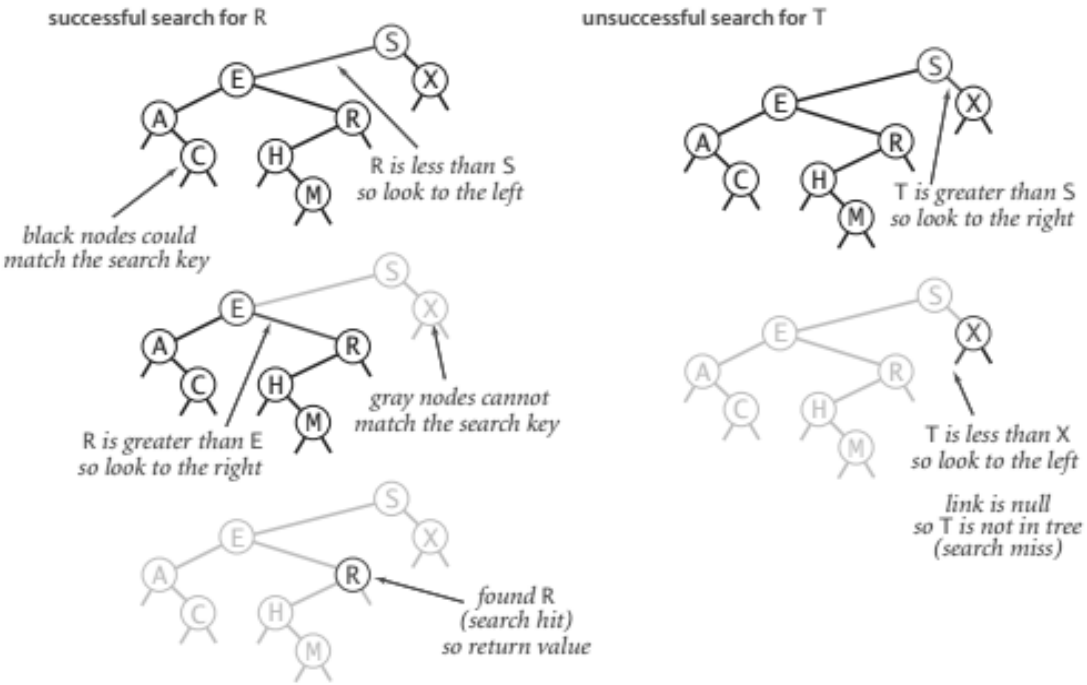
ALGORITHM 3.3 (continued) Search and insert for BSTs

```
public Value get(Key key)
{  return get(root, key);  }

private Value get(Node x, Key key)
{  // Return value associated with key in the subtree rooted at x;
   // return null if key not present in subtree rooted at x.
   if (x == null) return null;
   int cmp = key.compareTo(x.key);
   if      (cmp < 0) return get(x.left, key);
   else if (cmp > 0) return get(x.right, key);
   else return x.val;
}

public void put(Key key, Value val)
{  // Search for key. Update value if found; grow table if new.
   root = put(root, key, val);
}

private Node put(Node x, Key key, Value val)
{
   // Change key's value to val if key in subtree rooted at x.
   // Otherwise, add new node to subtree associating key with val.
   if (x == null) return new Node(key, val, 1);
   int cmp = key.compareTo(x.key);
   if      (cmp < 0) x.left  = put(x.left,  key, val);
   else if (cmp > 0) x.right = put(x.right, key, val);
   else x.val = val;
   x.N = size(x.left) + size(x.right) + 1;
   return x;
}
```

Qidiruv

Odatda, nom jadvalida kalitni qidirishda ikkita variant bo‘lishi mumkin. Agar qidirilayotgan kalit tuguni jadvalda mavjud bo‘lsa, u holda nishonga tegadi va kalit bilan bog‘langan qiymat qaytadi. Agar aksincha bo‘lsa, u holda nishonga tegmaydi va nolga qaytadi. Binar daraxt bo‘yicha qidiruvda kalit qidiruv rekkursiv algoritmi rekursiv struktura bo‘yicha yo‘naladi: agar daraxt bo‘sh bo‘lsa, u holda nishonga tegmaydi; agar qidirilayotgan kalit ildiz kalitga teng bo‘lsa. U holda nishonga tegishini bildiradi. Aks holda qidiruv (rekursiv) mos ravishda qidirilayotgan kalit kichik bo‘lsa chap qism daraxtda, katta bo‘lsa o‘ng qism daraxtda davom ettiriladi. Jarayon qidirilayotgan kalit mavjud bo‘lgan tugun topilganga yoki joriy

qism daraxt bo‘sh bo‘lganda tugatiladi. Qidiruv yuqoridan boshlanadi va qism tugunlarga rekursiv o‘tadi, shuning uchun ham qidiruv daraxt yo‘nalishini aniqlab beradi. Nishonga tegadigan bo‘lganda qidirilayotgan kalit mavjud bo‘lgan tugun yo‘nalishi tugatiladi, nishonga tegmaydida yo‘nalish nolinchi silka bilan tugatiladi.



Search hit (left) and search miss (right) in a BST

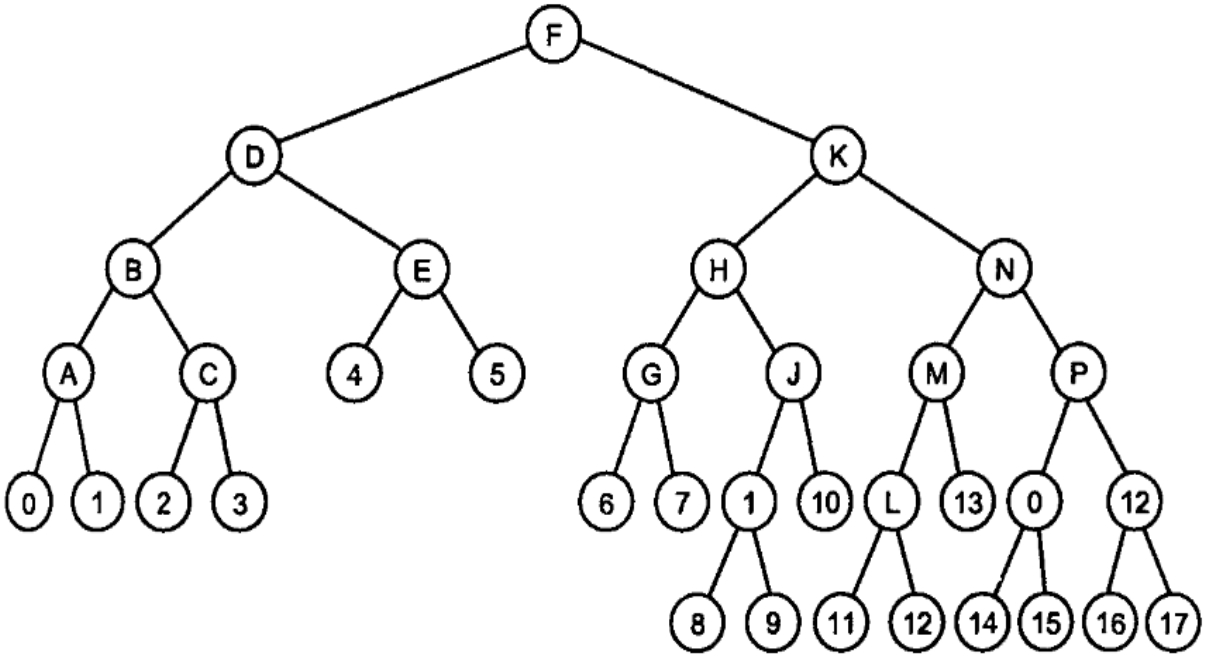
Binar daraxt bo‘yicha qidiruvda (chapda) nishonga tegish va (o‘ngda) nishonga tegmaslik.

Muvozanatlashgan (Balansirlangan) daraxt bo‘yicha qidiruv

Agar har bir tugunning chap qism daraxti o‘ng qism daraxtnikidan ± 1 dan ortiq bo‘lmagan holda farq qilsa, u holda Binar daraxt muvozanatlashgan (Balansirlangan) deyiladi. (16-chizma).

[26]Muvozanatlashgan (Balansirlangan) daraxt bo‘yicha oraliq joylashuvni eng samarali binar daraxtlar egallaydi (barcha tashqi tugunlar ikkita qo‘shni darajalarda joylashgan bo‘ladi).

[26] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 84-85 сс.



16-chizma. Muvozanatlashgan daraxtga misol

Quyidagi muvozanatlashgan (Balansirlangan) daraxt tugunlari strukturasini ko‘rib chiqamiz. (2-jadval).

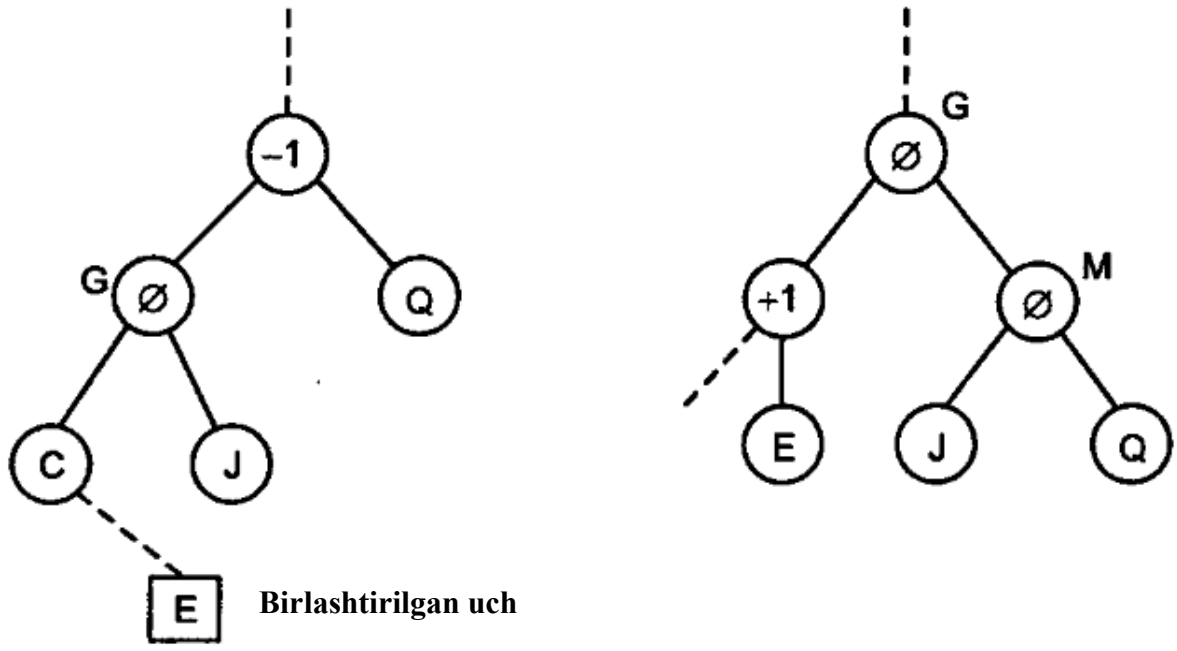
2-jadval

Kalit	**Chap qism daraxt ko‘rsatkichi**	**O‘ng qism daraxt ko‘rsatkichi**	**Muvozanatlashgan tugun ko‘rsatkichi**
KEY			

2-jadvalda – B o‘ng va chap qism daraxtlarning $(B = +1; 0; -1)$ farqidan iborat muvozanatlashgan tugun ko‘rsatkichidir. Balandlik bo‘yicha daraxt muvozanatini qayta tiklashda B ko‘rsatkich alohida e’tiborga olinadi. (16-chizma). [27]

17-chizmadagi $+1, \varnothing, -1$ belgilar chap qism daraxt o‘ng qism daraxtdan baland, qism daraxtlar balandliklari teng, o‘ng qism daraxt chap qism daraxtdan baland ekanliklarini ko‘rsatadi.

[27] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 85-86 сс.



17-chizma. Muvozanatlashgan tugunga ega ko‘rsatkichli daraxt

Bor usulida qidiruv

Asosiy qidiruv usullari guruhlarini son va harflar ketma-ketligi tashkil etadi. Masalan, ko‘pgina lug‘atlarda mavjud bo‘lgan harfi turkumlarni ko‘rib chiqamiz. U holda berilgan so‘zning birinchi harfi bo‘yicha boshlanadigan barcha so‘zlarni qamrab olgan sahifalarni topish mumkin. Harflar bo‘yicha turkumlanish g‘oyasini rivojlantirish natijasida Bor strukturasiga asoslangan indekslardan iborat qidiruv sxemasini hosil qilamiz.

Bor o‘zi bilan ***m-chi*** daraxtni tasvirlaydi. ***h*** litrdan iborat aniq ketma-ketlikdan boshlanuvchi ***h*** darajali har bir tugun barcha kalitlar to‘plamini tasvirlaydi. Tugun ***(h+1)*** chi litrga bog‘liq bo‘lgan ***m-*** tarmoqlanishni aniqlaydi. Bor quyidagi ko‘rinishdagi jadvalni namoyish etadi (3-jadval).

(-) probel- jadvalning asosiy belgisi hisoblanadi.

Birinchi tugunda kalitning 1 harfi yoki 1 soni yoziladi. Ikkinchi tugunda unga yana bitta belgi qo‘shiladi va hokazo. Agar aniq bir harf (son) dan boshlanuvchi so‘z tabiiy bo‘lsa, u holda ushbu so‘z darhol 1 chi tugunga yoziladi.

Bor strukturasi

3-jadval

Belgilar	Tugunlar				
(_) probel	**1**	**2**	**3**	**…**	***N***
					28

5-misol.

$\{A, AA, AB, ABC, ABCD, ABCA, ABCC, BOR, C, CC, CCC, CCCD, CCCB, CCCA\}$

to‘plam berilgan.

Dastlabki berilgan to‘plamdan bor hosil qiladiganiga o‘tamiz. Dastlabki alifbo $= \{A, B, C, D\}$

Bor – ***B*** harfdan boshlanuvchi tabiiy so‘z va u harflar bo‘yicha ajralmaydi.

Bor tugunlari har bir komponenti kalit yoki silkadan (bo‘sh bo‘lishi mumkin) iborat bo‘lgan vektorlarni tashkil qilinadi. (4-jadval).

Bor usulida qidiruvga misol

4-jadval

Belgilar	**Tugunlar**						
	1	2	3	4	5	6	7
_		A_	AB_	ABC_	C_	CC_	CCC_
A	2	AA		ABCA			CCCA
B	BOR	3					CCCB
C	5		4	ABCC	6	7	
D				ABCD			CCCD

[28] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 87-с.

1 tugun ildiz, shuning uchun ham birinchi harfni shu yerda qidirish kerak. Masalan, agar 1 chi ***B*** harfi bo‘lib chiqsa, u holda jadvaldan ko‘rinib turibdiki unga **BOR** so‘zi mos keladi. Agar 1 chi harf ***A*** bo‘lsa, u holda 1 tugun 2 chi harf qidiriladigan 2 tugunga boshqaruvni topshiradi. 2 chi tugun ikkinchi harf_,***A, B*** va hoko-zolar bo‘lishi mumkinligini ko‘rsatadi.

h-Xashlash usulida qidiruv

Qidiruv asosida dastlabki to‘plamdan ***h***-funksiya $(h(k))$ to‘plamiga o‘tishni yotadi.

h-funksiya quyidagi ko‘rinishga ega:

$$\big(h(k)\big) = kmod(m),$$

bu yerda ***k***-kalit; ***m***-butun son; ***mod***- bo‘lishning butun sonli qoldig‘i. [29]

H-funksiya quyidagi ko‘rinishga ega bo‘lganligi uchun bo‘lish amali yordamida *k* kalitni birorta ***m*** yacheykaga ***k*** ni m ga qoldiqli bo‘lish orqali tasvirlaydi:

$$\big(h(k)\big) = kmod(m),$$

Masalan, agar h-jadval **m=12** o‘lchovga, k kalit qiymati **k=100** ga ega bo‘lsa, u holda ***h(k)=4*** bo‘ladi. ***H***-funksiyani hisoblashda bitta bo‘lish amali yetarli bo‘lganli sababli, bo‘lish usuli orqali xashlash ancha tezroq amalga oshadi.

Masalan, $\{9, 1, 4, 10, 8, 5\}$ to‘plam berilgan bo‘lsin.

Uning uchun $\big(h(k)\big) = kmod(m)$ ***h***-funksiyani aniqlaymiz:

1) $m = 1$ bo′lganda

$$\big(h(k)\big) = \{0, 0, 0, 0, 0, 0\};$$

2) $m = 20$ bo′lganda

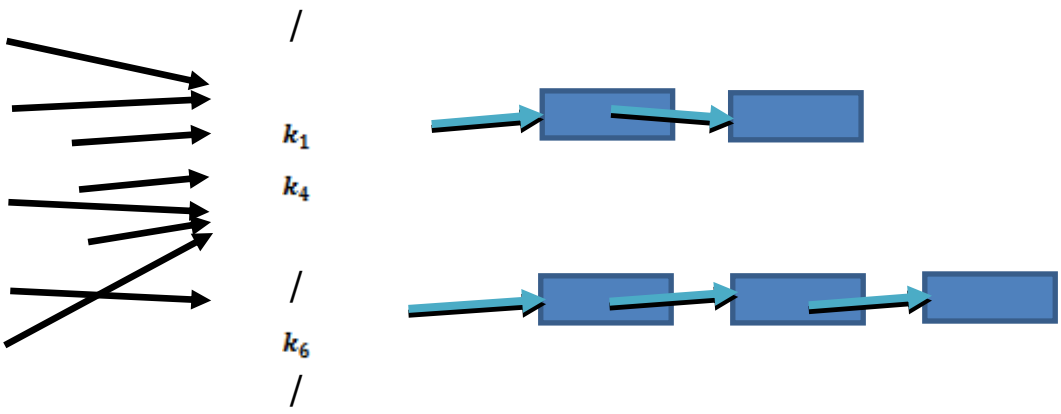
[29] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 262-263 pp.

$$(h(k)) = \{9, 1, 4, 10, 8, 5\};$$

3) m –eng katta kalitning yarmiga bo‘lganda **m**= [**10/2**]=**5**

$$(h(k)) = \{4, 1, 4, 0, 3, 0\}$$ bo‘ladi.

h-funksiya[30] qidirish kerak bo‘lgam manzilni ko‘rsatadi. Turli xil kalitlar uchun h-funksiya bir xil qiymatni qabul qilishi mumkin, bunday holat to‘qnashuv holati deb ataladi. Shu tariqa h-li qidiruv zanjir usulida to‘qnashuv holati bartaraf etish bilan yakunlanadi.(18-chizma).



18-chizma. Zanjir usulida to‘qnashuv holatini bartaraf etish

6-misol.

$$\{7, 13, 6, 3, 9, 4, 8, 5\}$$

to‘plam berilgan bo‘lsin. $K = 27$ kalit topilsin.
h-funksiya $(h(k)) = Kmod(m)$ ga teng;
$\mathbf{m} = [13/2] = 6$ (13-eng katta kalit bo‘lganligi uchun)

$$(h(k)) = \{1, 1, 0, 3, 3, 4, 2, 5\}$$ bo‘ladi.

To‘qnashuv holatini bartaraf etish 5-jadvalni tuzamiz.

[30] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 87-88 сс.

h-funksiyani dastlabki kalit to‘plamlarini juftliklari bilan taqqoslab, jadvalni to‘ldiramiz. ***h***-funksiya kalit qidirilishi kerak bo‘lgan manzilini ko‘rsatishini eslatib o‘tamiz.

To‘qnashuv holatining bartaraf etilishi

5-jadval

h (k)	**Kalit zanjirlari**
0	**6**
1	**7,13**
2	**8**
3	**3,9**
4	**4**
5	**5**

Masalan, agar qidirilayotgan kalit $K = 27$ bo‘lsa, u holda $h(k) = 27 mod 6 = 3$ bo‘ladi, bu esa $K = 27$ kalit faqat 3 chi qatorda bo‘lishini bildiradi. U yerda kalit mavjud bo‘lmasa, dastlabki to‘plamda ham mavjud emasligini bildiradi.

Interval bo‘yicha izlash. Bunda chap yoki unga eng maksimal yaqinlashgan chegara topiladi. So‘ngra stekni teskari tartibda ko‘rib chiqish yo‘li bilan o‘ng chegarani qidiramiz, yaqin chap chegaradan katta yoki teng va o‘ng chegaradan kichik tugunlarni qidiramiz.

```
Procedure Obrab(ss:pt);
Begin Write(ss^.kl:5) End; {imitiruyushaya obrabotku uzlov}
Prosedure Poisk(sor:pt; ar1,ar2:tipkl; l:integer; var pp:pt);
Label 1,2; {sl - BDU tuguniga ko‘rsatkich sohasi, ssl – stekdagi
bog‘lanish}
Type pnt=^z;
z=Record
sl:pt; ssl:pnt End;
Var rr,ss,tt: pt; q,t: pnt;
Procedure StekIn; { BDU tuguniga t ko‘rsatkichni stekka qo‘shish}
Begin New(q); q^.sl:= ss; q^.ssl:=t; t:=q End;
```

```
Begin pp:= Nil; rr:= Nil; t:= Nil; tt:= sor;
If l = 3 Then ar2:=ar1;
While tt <> Nil do {Izlash jarayoni sikli}
Begin ss:= tt;
If ss^.kl = ar1 Then {kalitning mos tushish holati}
Begin pp:= ss; StekIn; Goto 1 End;
If ss^.kl > ar1 Then Begin tt:= tt^.lson; StekIn End
Else Begin tt:= tt^.rson; rr:= ss End;
End;
If (l <> 2) And (t <> Nil) Then pp:= t^.sl;
If l=-1 Then pp:= rr;
1:If pp <> Nil Then If l < 3 Then Obrab(pp)
Else If t^.sl^.kl > ar2 Then pp:= Nil
Else { l=3,4: uchun ishning 2-ETAPI}
While t <> Nil do {BDUni chapdan qismiy ko‘rib chiqish sikli}
Begin q:= t; ss:qt^.sl; t:= t^.ssl; Dispose(q);
If ss^.kl > ar2 Then Goto 2;{ Ko‘rib chiqishni tugallash ehtimoli }
Obrab(ss); ss:= ss^.rson;
While ss <> Nil do
Begin While ss^.lson <> Nil do {“O‘yinchi”gacha chapga o‘tish}
Begin StekIn; ss:= ss^.lson End;
If ss^.kl > ar2 Then Goto 2;{Ko‘rib chiqishni tugallash ehtimoli}
Obrab(ss); ss:= ss^.rson
End
End; {BDU ni chapdan qismiy ko‘rib chiqish blokining oxiri}
2:While t <> Nil do Begin q:=t; t:=t^.ssl; Dispose(q) End;
End{Binar daraxtda izlash bloki oxiri( BDU)};
```

Binar daraxt B-daraxtning xususiy holi bo‘lib hisoblanadi.m-darajali V – daraxt quyidagi shartlarni qanoatlantiruvchi umumiy daraxt sifatida aniqlanadi:

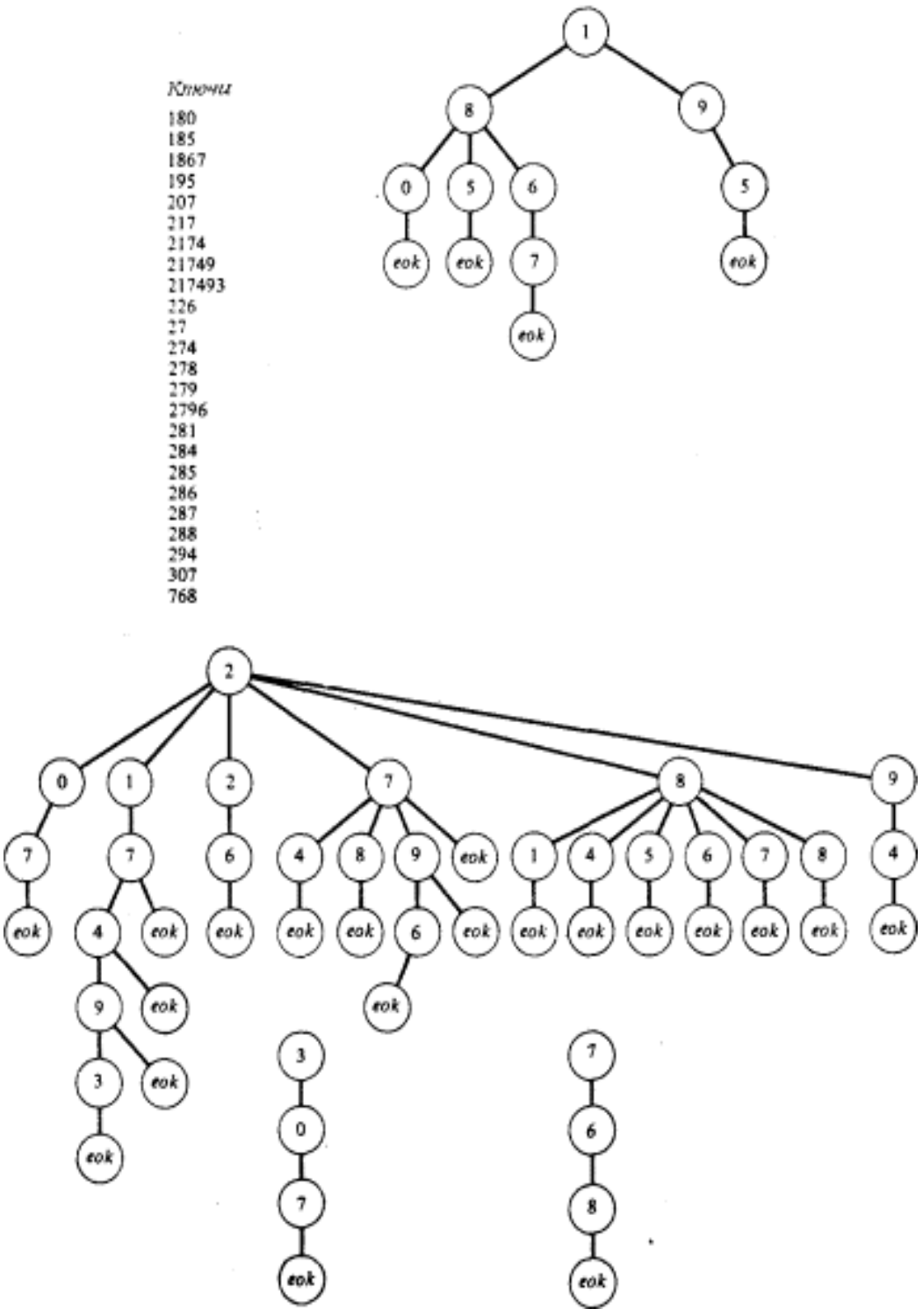
1. Har bir tugun maksimal m-1 ta kalit saqlaydi;
2. Bosh(ildiz) tugundan boshqa har bir tugun eng kamida int((m-l)/2) ta kalit saqlaydi;
3. Ildiz tugun avlod bo‘lmasa, eng kamida 2 ta avlod tugunga ega;
4. Barcha avlodlar bitta darajada joylashgan.

5. Avlod bo‘lmagan n kalitli tugun p+1 ta avlodga ega.

Quyidagi rasmda 5-darajali V-daraxt ifodalangan.Bu yerda har bir tugun qandaydir tartiblangan (p_1, k_1, p_2, k_2, ..., kn-1, rn) guruh orqali ifodalanishi mumkin.Bu yerda pi ko‘rsatkich (bo‘sh ko‘rsatkich gar berilgan tugun avlod bo‘lsa) esa qandaydir kalit. Pi ko‘rsatayotgan tugundagi kalitlar ki-1 va ki orasida joylashadi. Har bir tugun ishida esa $k_1<k_2<.. .<$kn-1 tengsizlik bajariladi.

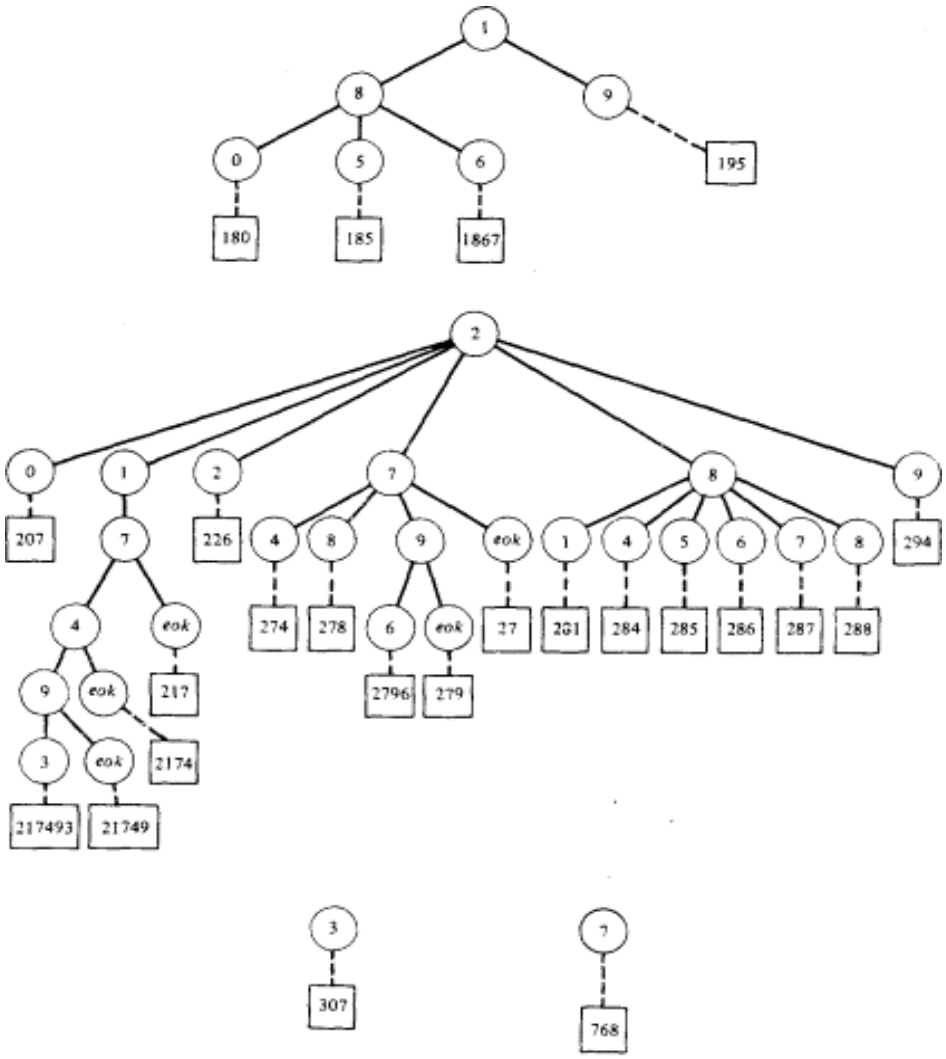
19-chizma

Raqamli izlash daraxtlari. Izlash jarayonini tezlashtirish ushun daraxtlardan foydalanishning boshqa usuli kalitlar tarkib torgan simvollarga asoslanadigan qandaydir umumiy daraxt shakllantirishdan iborat. Masalan, agar kalitlar sonli bo‘lsa, har bir raqam pozisityasi berilgan tugunning 10 ta mumkin bo‘lgan avlodlaridan birini aniqlaydi.



20-chizma

Daraxtning har bir tuguni maxsus eok simvoliga ega. Bu simvol qaysidir kalit oxirini bildiradi. Bunday tugun saqlab qolinuvchi yozuvni ko‘rsatuvchi ko‘rsatkichni ham o‘zida saqlaydi.



21-chizma

Shtrixlangan chiziq daraxt tugunidan kalitga ko‘rsatkichni ifodalaydi.

9-§. Rabin–karp algoritmi

Rabin-Karp algoritmi satriy kattalik tarkibida ikkinichi satriy kattalik mavjudligini aniqlovchi bo‘lib, 1987 yili Maykl Rabin va Richard Karp tomonidan ishlab chiqilgan. Undan kam foydalanishsa ham muhim amaliy ahamiyatga ega. Uzunligi n–ga teng bo‘lgan satr va uzunligi m bo‘lgan shablon uchun algoritmning o‘rtacha

bajarilish vaqti O(n) ga teng, ammo ba’zi hollarda bu vaqt O(mn) ga ham teng bo‘lishi mumkin. Bu algoritm “ko‘chirmakashlik” (flagiatni) aniqlashda qo‘llaniladi.

Umuman olganda satriy kattalik tarkibidagi ikkinchi satriy kattalik borligi har bir simvol mos kelisi bilan tekshirish bilan aniqlanadi

	1	2	3	4	5	…				n
S[1..n]	S[1]	S[2]	S[3]	S[4]	S[5]					
Sub[1..5]	Sub[1]	Sub[2]	Sub[3]							

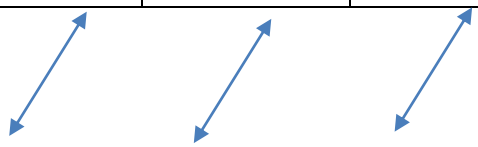
I=1, j=1 s[1+1-1]<>sub[1]=>s[1]<>sub[1]
J=2 s[1+2-1]<>sub[2]->s[2]<>sub[2]
J=3 s[1+3-1]<>sub[3]->s[2]<>sub[3]
…

Bunday taqqoslash m-simvol uchun o‘ngga surish bilan bajariladi va barcha m-simvollar teng bo‘lgandagina ikkinchi satriy kattalik birinchi satriy kattalik tarkibida mavjud deb xulosa chiqariladi. Bu holatni xususiy holda quyidagicha ifodalash mumkin:

1-qadam:

Birinchi satriy kattalik

1-simvol	2-simvol	3-simvol	4-simvol		n-simvol



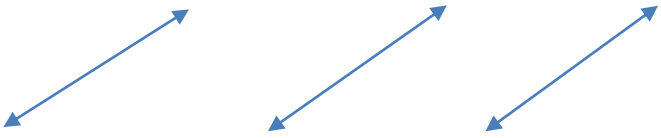
Ikkinchi satriy kattalik

1-simvol	2-simvol	3-simvol

2-qadam:

Birinchi satriy kattalik

1-simvol	2-simvol	3-simvol	4-simvol		n-simvol



Ikkinchi satriy kattalik

1-simvol	2-simvol	3-simvol

Qolgan qadamlar ham shu tartibda bajariladi. Endi shu holatga, agar birinchi satriy kattalik nihoyat uzun bo‘lib ikkinchi satriy kattalik ham kattaroq uzunlikka ega bo‘lib qolsa, taqqoslaslar soni va ish hajmiga e’tibor bering. Albatta bu oson emasligi darhol ko‘zga tashlanadi.

Quyida biz o‘zimiz yaratgan qism dastur va dasturni keltirib o‘tamiz.

```
program Project1;
{$APPTYPE CONSOLE}
{$R *.res}
function qidiruv(s:string;sub:string): boolean;
 var i,j,n,m,k:integer;
 begin
   n:=length(s);m:=length(sub);
   for i:=1 to ( n-m+1) do
   begin
   k:=0;
   for j := 1 to m do
   if s[i+j-1] = sub[j] then k:=k+1;
   if k=m then
   begin
   result:=true;writeln(i);
   end;
   end;
 end;
 function qidiruv1(s:string;sub:string):string;
 var i,j,n,m,k:integer; var aniqss:string;
 begin
   n:=length(s);m:=length(sub);
   for i:=1 to ( n-m+1) do
   begin
   k:=0;  aniqss:='';
    for j := 1 to m do
   if s[i+j-1] = sub[j] then
   begin
   k:=k+1; aniqss:=aniqss+sub[j];
   if k=m then
```

```
  begin
  result:=aniqss;//writeln(aniqss);
  end;
  end;
  end;
end;

 var ss,subsub:string; d:boolean; kk:char;
 l,g:integer; ssxash:real; sssub:string;
begin
  ss:='informatika';
  subsub:='for';
  d:=qidiruv(ss,subsub);
  sssub:= qidiruv1(ss,subsub);
  Writeln(d);
  writeln('------------------');
  Writeln (sssub,'----');
    readln;
end.
```

Dasturda ss:='informatika' satriy kattalik tarkibida subsub:='for' satriy kattalik mavjudligi tekshirilishi yo‘lga qo‘yilgan va unung to‘g‘ri bajarilishi dastur natijasiga ko‘ra kuzatiladi. Tushunarli bo‘lishi uchun qidiruv1 va qidiruv funksiyalaridan foydalanishni lozim topdik, aslida ularning biri yetarli edi.

Dastur natijasi:

Yuqorida keltirilganlardan shuni xulosa qilish mumkin: uzunligi katta bo‘lgan satriy kattaliklar uchun taqqoslashlar soni nihoyatda ko‘payadi va algoritmning samaradorligi kamayashi kuzatiladi. Masalan 10 000 000 “a” simvoldan iborat satriy kattalik tarkibida 10 000 ta “a” simvol va unga oxirgi simvol sifatida “b” qo‘shilgan bo‘lsa algoritm bajarilish vaqti O(mn)ga teng bo‘ladi.

Rabin-Karp algoritmi xesh-funksiya qiymatini hisoblash orqali takomillashtiriladi. Bunda dastlab solishtiriladigan satriy kattaliklar uchun maxsus sonli qiymat hisoblanib bu qiymatlar teng bo‘lganda har bir satriy kattalik simvollari solishtiriladi va bu bajaraladigan taqqoslashlar sonini keskin kamaytiradi.

Agar qidirilayotgan satr “on” bo‘lib 37 tub soni sanoq sistemasining asosi bo‘lsa u holda bu qiymatni hisoblash funksiyasi quyidagi ko‘rinishda bo‘ladi:

```
function xesh(xeshsub:string; ddd:byte): real;
 var i,j,n,m,g:integer; kk:char; k:real;
 begin
   m:=length(xeshsub); g:=m;
    for i:=1 to m do
   begin
    kk:= xeshsub[i];g:=g-1;
    k:=k+ord(kk)*exp(g*ln(37));
    end;
   result:=k;

   end;
]
```

Rabin va Karp amalda yaxshi natija beruvchi, shuningdek boshqa o‘xshash masalalarni, misol uchun ikkilik nusxasini o‘rnatish masalasini umumlashtiruvchi qatorlardan izlash algoritmini taqdim etishdi. Rabin - Karp algoritmida vaqt $\Theta(m)$ dastlabki ishlov uchun sarf qilinadi, uning ishlash vaqti esa eng yomon holatda $\Theta((n - m + 1)m)$ ga teng. Lekin aniq ko‘rsatmalar yordamida uning o‘rtacha ishlash tezligi sezirarli darajada yaxshi ekanini ko‘rish mumkin.

Algoritmda elementar sonlar nazariyasidagi belgilashlardan foydalaniladi:

Uchinchi sonning moduli asosida ikki sonning ekvivalentligi.

Sodda holatda $\sum = \{ 0 ,1 ,2 ,...,9 \}$ ga, ya’ni har bir belgi o‘nlik s.s dagi sonni beradi. (Umumiy holatda har bir belgi – bu d asosli s.s. dagi songa teng: d — $|\sum|$.

Endi ketma-ket joylashgan k belgilardan iborat qatorni k raqamlardan iborat o‘nlik son ko‘rinishida tasvirlash mumkin.

Shunday qilib 31415 belgili qatorni 31415 ko‘rinishidagi o‘nlik son ko‘rinishida tasvirlash mumkin.

Berilgan P[1..*m*] namunani unga mos keluvchi o‘nlik qiymati bilan belgilaymiz. Xuddi shu kabi berilgan *T*[1..*n*] matnni, *s=0, 1, ... , n-m* bo‘lganda uzunligi *m* o‘nlik belgili qator qiymati *T* [*s + 1 .. s + m*] bo‘lgan t_s ko‘rinishida belgilaymiz. Ko‘rinib turibdiki, faqat va faqat T[s + 1.. s+ m] = P[1.. m] bo‘lgandagina t_s=p bo‘ladi, shunday qilib t_s=p bo‘lganda s-surish qiymati bo‘ladi. Agar p ning qiymatini vaqt Θ(*m*) ga, t_s ning barcha qiymatlarini umumiy vaqt Θ(*n* — *m* + 1) qiymatiga nisbatan topish mumkin bo‘lsa, u holda i *p* ning har bir qiymatini t_s ning har bir qiymatiga solishtirish yo‘li bilan mumkin bo‘lgan surilishlarning qiymati vaqtga nisbatan topish mumkin: Θ(*m*)+ Θ(*n* — *m* + 1) = Θ (*n*).

Gorner qoidasiga asosan p ning qiymatini vaqt Θ(*m*)ga nisbatan quyidagi formula yordamida aniqlash mumkin:

$$p = P[m] + 10 (P[m - 1] + 10(P[m - 2] + ... + 10(P[2] + 10P[1])...)).$$

Xuddi shu kabi t_0 qiymatini shu vaqt Θ(*m*) ga nisbatan *T[1..m]* dan aniqlash mumkin.

$t_1, t_2, \dots , t_{n-m}$ larning qiymatini vaqt Θ(*n* — *m)* ga nisbatan qiymatini aniqlash uchun t_{s+1} kattalikni t_s kattalikni o‘zgarmas vaqtga nisbatan aniqlash mumkin:

$$t_{s+1} = 10 (t_s - 10^{m-1}T[s + 1]) + T [s + m + 1] . \quad (32.1)$$

10^{m-1}T[s + 1]ni ayirish t_s dan katta bo‘lgan sonni o‘chiradi, 10 ga ko‘paytirish natijasi sonni chapga 1 o‘nlik razryadga suradi *T[s + m + 1]* qo‘shish unga mos keluvchi kichik sonni qo‘shadi. Masalan, agar *m=5* va t_s*=31415* bo‘lsa, u holda *T[s+1] = 3* katta sonni o‘chirish va kichik sonni (*T[s+5+1]=2*) ni qo‘shish kerak:

$$t_{s+1} = 10(31415 - 10000 * 3) + 2 = 14152 .$$

Agar o‘zgarmas 10^{m-1} ni avvaldan hisoblab olinsa, u holda (32.1) ifodadaning har bir hisoblanishiga aniq miqdordagi arifmetik amallar talab qilinadi. Shunday qilib *p* sonni Θ(*m*) vaqtga nisbatan

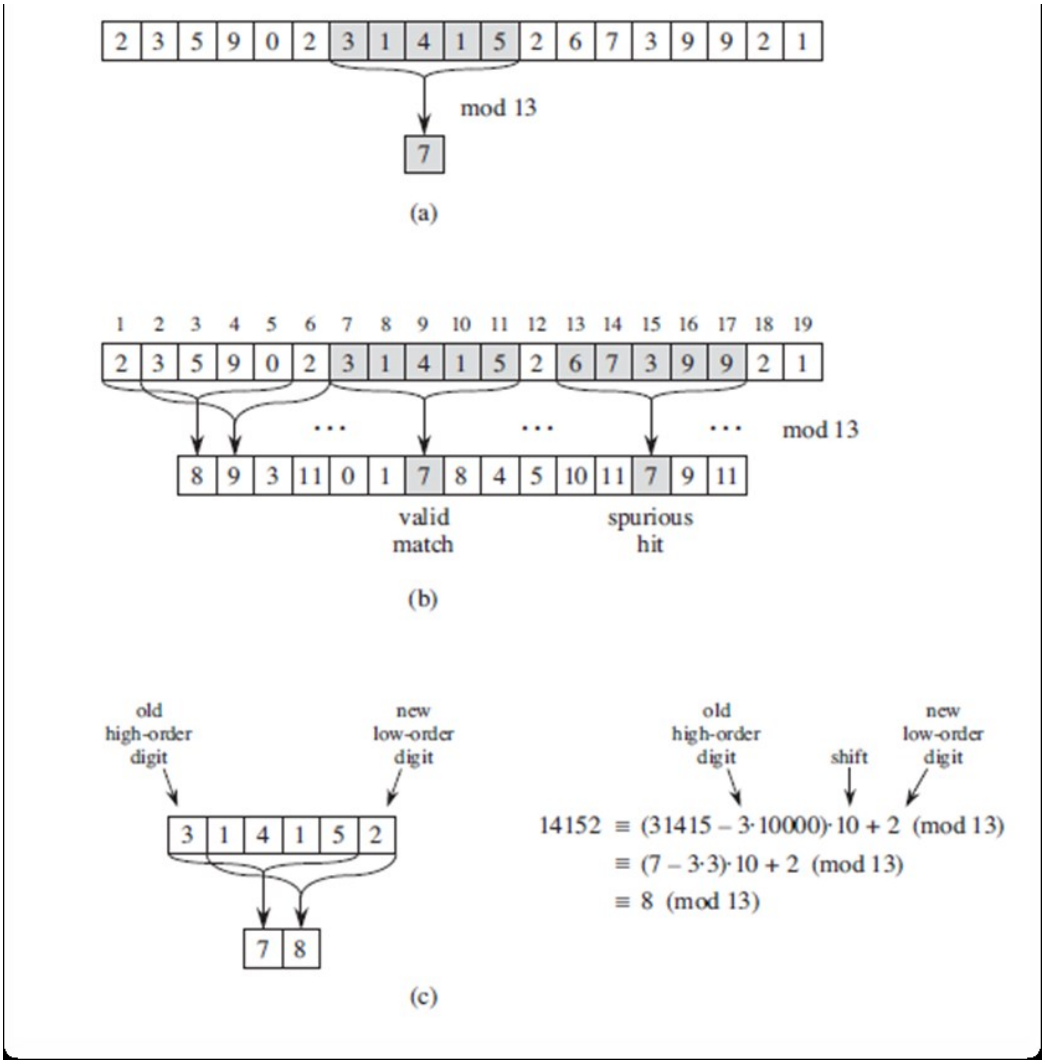
toppish mumkin, $t_1, t_2, \ldots, t_{n-m}$ –vaqtga nisbatan $\Theta(n-m+1)$, T[1..*n*] matndagi *P*[1..*m*] namunani tashkil etuvchilarni avvaldan olingan vaqt $\Theta(m)$ fazasini sarflab aniqlash mumkin. Taqqoslash fazasini esa – vaqtga nisbatan $\Theta(n-m+1)$

Bu jarayonda qiyinchilik tug‘dirishi mumkin bo‘lgan yanona vaziyat bu *p va* t_s sonlarning juda ham katta qiymatga ega bo‘lishidir. Agar namuna *P m* belgilarni qabul qilsa, u holda *p* son bilan bajariladigan arifmetik amal “aniq miqdordagi vaqt” ni oladi, bu esa o‘z navbatida haqiqatga javob bermaydi. Bu muammoni oson yechimi bor: *p* va t_s qiymatlarini bir qancha *q* sonlarning moduli asosida hisoblash. *p* ning qiymatini *q* ning moduli bo‘yicha vaqtga nisbatan $\Theta(m)$, t_s ning barcha qiymatlarini vaqtga nisbatan $\Theta(n-m+1)$ orqali topish mumkin. Agar *q* ning qiymati uchun oddiy sonni olinsa va 10*q* kompyuter so‘ziga mos kelsa, u holda barcha hisoblashlarni bir xil aniqlikdagi arifmetika yordamida bajarish mumkin. Umumiy holatda, *d*-belgili alifbo mavjud, bunda *q* ning qiymati shunday tanlanadiki, *dq* kompyuter so‘ziga mos kelishi kerak va rekurrent munosabat (32.1) *q* modul bo‘yicha ishlaydigan qilib moslashtiriladi:

$$t_{s+1} = (d(t_s - T[s+1]h) + T[s+m+1]) \bmod q , \quad (32.2)$$

bu yerda $h \equiv d^{m-1} (\text{mod } q)$ — “1” raqamining katta razryadli oyna o‘lchamidagi *m* raqam qiymati.

Lekin *q* moduli bilan ishlash kamchiliklardan holi emas, chunki $t_s \equiv p (\text{mod } q)$ ifodadan $t_s = p$ ekani kelib chiqmaydi, boshqa tarafdan, agar $t_s \neq p (\text{mod } q)$ bo‘lsa, u holda so‘zsiz $t_s \neq p$ ifoda bajariladi, bunda esa s surilish mumkin bo‘lmaydi. Shunday qilib, $t_s \equiv p (\text{mod } q)$ tekshirishdan mumkin bo‘lmagan s surilishni aniqlash uchun tezkor evristik test sifatida foydalanish mumkin. $t_s \equiv p (\text{mod } q)$ munosabat o‘rinli bo‘lgan har qanday s surilish uning haqiqatdan ham mumkin ekanligini (yolg‘on moslik (spurious hit) emasligi) aniqlash uchun qo‘shimcha tekshirilishi kerak. Bunday tekshirishni P[1.. *m*] = T[s+ 1 .. s+m] shartni tekshirish orqali o‘tkazish mumkin. Agar *q* ning qiymati yetarlicha katta bo‘lsa, u holda yolg‘on moslik uchrash ehtimoli kam bo‘ladi va qo‘shimcha tekshirish o‘tkazish talabi kam bo‘ladi.



Quyida keltirilgan protsedura yuqorida berilgan g‘oyani o‘zida aks ettiradi. Kiruvchi ma’lumot -T matn, namuna-P, sanoq sistemasining asosi d va oddiy son q:

R a b in - K a r p - M a t c h e r (T , P, d, q)
1 n = T.length
2 m = P. length
3 h = d^{m-1} m o d q
4 p = 0
5 t_0 = 0
6 for i = 1 to m // dastlabki ishlov
7 p = (dp+ P[i]) mod q
8 to = (dto + T[i]) mod q
9 for s = 0 to n-m // taqqoslash
10 if p==ts
11 if P[1 . . m] = = T[s + 1 . . S + m]
12 print s “surilish bilan namuna topildi ”
13 if S < n — m

14 $t_{s+1} = (d(t_s — T[s + 1]h) + T[s + m + 1]) \bmod q$

R a b in - K a r p - M a t c h e r protsedurasi quyidagicha ishlaydi: barcha belgilar *d* asosli sanoq sistemasidagi raqamlar sifatida ko‘riladi. O‘zgaruchi *t* ning indekslari aniqlik uchun ko‘rsatilgan; dastur ularsiz ham to‘g‘ri ishlaydi. 3-qatorda o‘zgaruvchi *h* *m*-raqamli oynaning katta razryadi raqami sifatida aks etadi. 4-8 qatorlarda *P*[1..*m*] mod *q* ga teng bo‘lgan *p* va *T*[1.. *m*] mod *q* ga teng bo‘lgan t_0 ning qiymati topiladi. 9-14 qatorlarda **for** sikli *s* ningbarcha mumkin bo‘lgan qiymatlarida iteratsiyani amalga oshiradi.

Satrning bajarilishida 10-satr o‘zaro nisbati to‘g‘ri T_s=T[s+1..s+m] mod q

Agarda 10 satrda p=t_s sharti bajarilsa, noto‘g‘ri o‘xshashlikni inkor etish uchun, 11-satrda P[1..m]=T[s+1..s+m] tenglik haqqoniyligi tekshiriladi. Barcha aniqlangan mumkin bo‘lgan siljishlar 12-satrga yuboriladi. Agar s<n-m (ushbu tengsizlik 13-satrda tekshiriladi), for davrini hech bo‘lmasa bir marta bajarish kerak bo‘ladi, shuning uchun avval 14-satr bajariladi, invariant davrini amal qilinishi ishonchli bo‘lishi uchun, biz yana 10-satrga o‘tamiz. 14-satrda t_s mod q qiymati asosida (32.2) tenglamasi yordamida, o‘zgarmas vaqt mobaynida t_s+1 mod q o‘lchami hisoblanadi.

Rabin-Karp-Matcher ish tartibida dastlabki ishlov berish uchun Θ(m) vaqt ketkaziladi, solishtirish mobaynida unda eng yomon holda Θ((n-m+1)m) ga teng, modomiki Rabin-Karp algoritmida (qatorosti qidirish algoritmidagi)mumkin bo‘lgan har bir siljishlar yaqqol tekshiriladi. Agar $P=a^m$, $aT=a^n$ bo‘lsa, tekshirish vaqti Θ((n-m+1)m) bo‘ladi, modomiki barcha n-m+1 ehtimolli siljishlar mumkin hisoblanadi.

Ko‘pincha ilovalarda kam miqdordagi yo‘l qo‘yilgan siljishlar kuzatiladi (bir qancha c konstantasi ifodalangan bo‘lishi mumkin); bunday ilovalarda matematik kutilma algoritm ish vaqti o‘lcham yig‘indisi O((n-m+1)+cm)=O(n+m) va tenglik haqqoniylikni tekshirish vaqti.

Mustaqil bajarish uchun savollar?

1. Agar q modul sifatida 11 qiymat tanlangan bo‘lsa, u holda T=31415926535897 93 matnida P=26 ni qidirish Rabin-Karp usulida qancha tenglik haqqoniyligi bajariladi?

2. Berilgan k namunadan birini matnli satrda qidirish masalasi uchun Karpa-Rabin usulini qanday umumlashtirish mumkin? Barcha k namuna bir xil uzunlikda deb hisoblang. So‘ng shunday umumlashtiringki, barcha namunalar har xil uzunlikda bo‘lishi mumkinligi nazarda tutilgan bo‘lsin.

10-§. Graflar bilan ishlovchi sodda algoritmlar

Graflar nazariyasi – bu diskret matematikaning sohasi bo‘lib, ayniqsa geometrik yondashuv asosida obyektlarni o‘rganishni bildiradi. Odatda, graflarni topologiyaga kiritish mumkin, lekin ular juda ko‘p fanlarda ham uchraydi. Graflar nazariyasining birinchi masalalari turli xil mantiqga oid masalalar bo‘lgan.

Graflarni tipik qo‘llanilishi[31]

application	item	connection
map	intersection	road
web content	page	link
circuit	device	wire
schedule	job	constraint
commerce	customer	transaction
matching	student	application
computer network	site	connection
software	method	call
social network	person	friendship

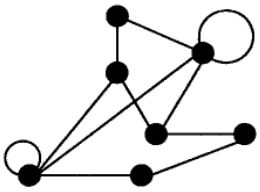
Ta’rif. Graf -bu uchlar to‘plami va har bir uchlar juftliklarini bog‘lovchi qirralar kolleksiyasidir. Ta’rifda uchlarni nomlash shart emas, ammo foydalanishda ularni nomlash kerak bo‘ladi.

$\boldsymbol{G = (V, E)}$ graf (tarmoq) –**G** grafning qirralari deb ataluvchi bo‘sh bo‘lmagan $\boldsymbol{(m \geq 1)}$ m uchlar to‘plami va tartiblanmagan $\boldsymbol{[u, v](n \geq 0)}$ juft elementlar to‘plamidan tuzilgan.

[31] Robert Sedgewick and Kevin Wayne. Algorithms. FOURTH EDITION. Princeton University. First printing, March 2011. Pp 517-518.

Uchlarni o‘zi bilan bog‘lab turuvchi qirraga halqa deyiladi.

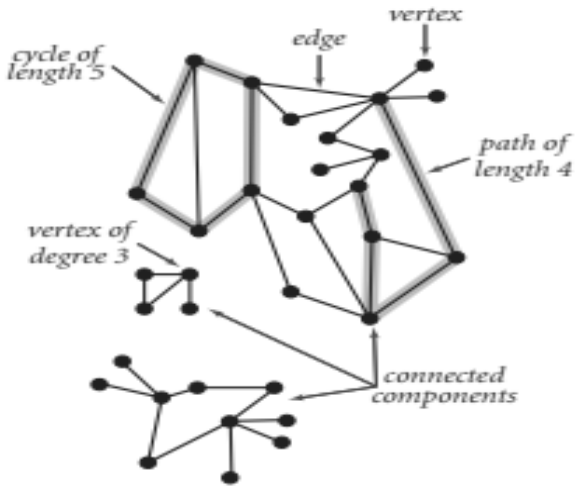
[32]Agar **G** da (u, v) qirra mavjud bo‘lsa, u holda Ikkita ***u*** va ***v*** uchlar qo‘shni bo‘ladi,



22-chizma. Grafli tuzilish(tarmoq)

Graflar terminologiyasi

Graflar bilan juda ko‘plab terminlar bog‘liq. Bu terminlar orasida ko‘pchiligi aniq ta’rifga ega va ularni quyidagi rasmda keltiramiz.



Anatomy of a graph

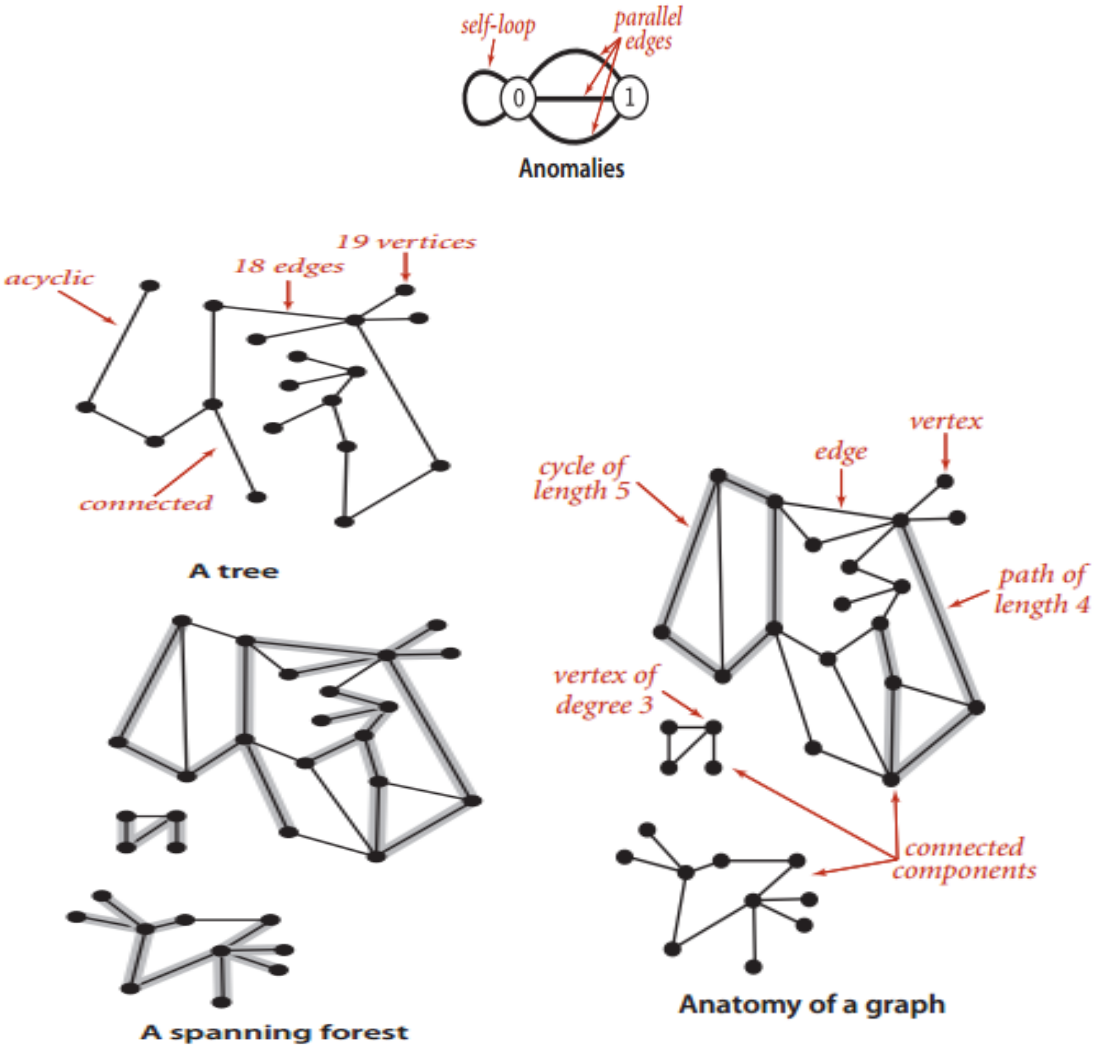
Agar ikkita uchni bog‘lovchi qirra mavjud bo‘lsa, uni qo‘shni uchlar, qirrani esa ikkita uchga teng kuchli deb ataymiz. Uchlar darajasi- bu u bilan teng kuchli qirralar sonidir. Qism graf bu grafni tashkil etuvchi qirralarining qism to‘plamidir.

Ta’rif. Grafda yo‘nalish- bu qirralar bilan bog‘langan uchlarning ketma-ketligi. Oddiy yo‘nalish bu takrorlanmaydigan uchlardan iborat yo‘nalishdir. Sikl bu, boshi va oxiri mos tushuvchi kamida bitta qirradan iborat bo‘lgan yo‘nalish.

[32] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 108-с.

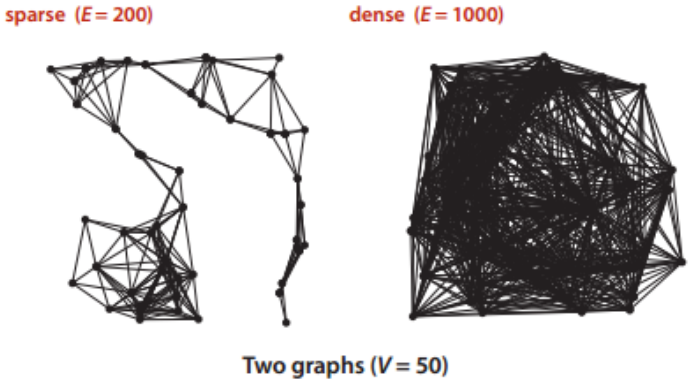
Oddiy sikl bu takrorlanmaydigan qirra va uchga ega sikldir. Yo‘nalish uzunligi yoki sikli uning qirralar soniga teng bo‘ladi.

Ta’rif. Agar grafda har bir uchdan boshqa uchga yo‘l mavjud bo‘lsa, Graf bog‘langan deyiladi. Bog‘lanmagan graf maksimal bog‘langan qism graflarni tasvirlovchi *bog‘langan komponentlar* to‘plamidan tashkil topgan bo‘ladi.



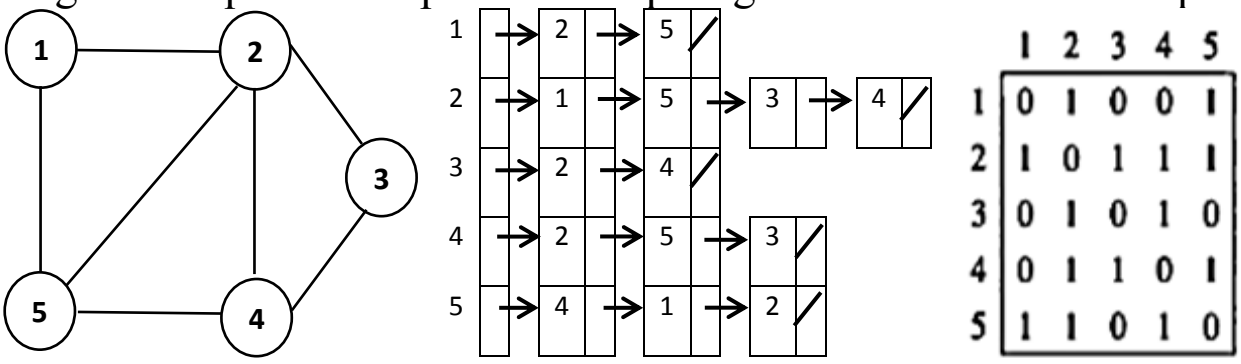
Ta’rif. Daraxt bu atsiklik bog‘langan grafdir. O‘zaro bo‘g-lanmagan daraxtlar o‘rmon deb ataladi. Bog‘langan grafning asos daraxti - bu berilgan grafning barcha uchlaridan iborat qism graf bo‘lib, yagona daraxt hisoblanadi. Grafning asos o‘r-moni bu asos daraxtlarining grafning komponentlarini bo‘g-lanishi. [33]

[33] Robert Sedgewick and Kevin Wayne. Algorithms. FOURTH EDITION. Princeton University. First printing, March 2011. Pp 519-521.



Grafning tasvirlanishi

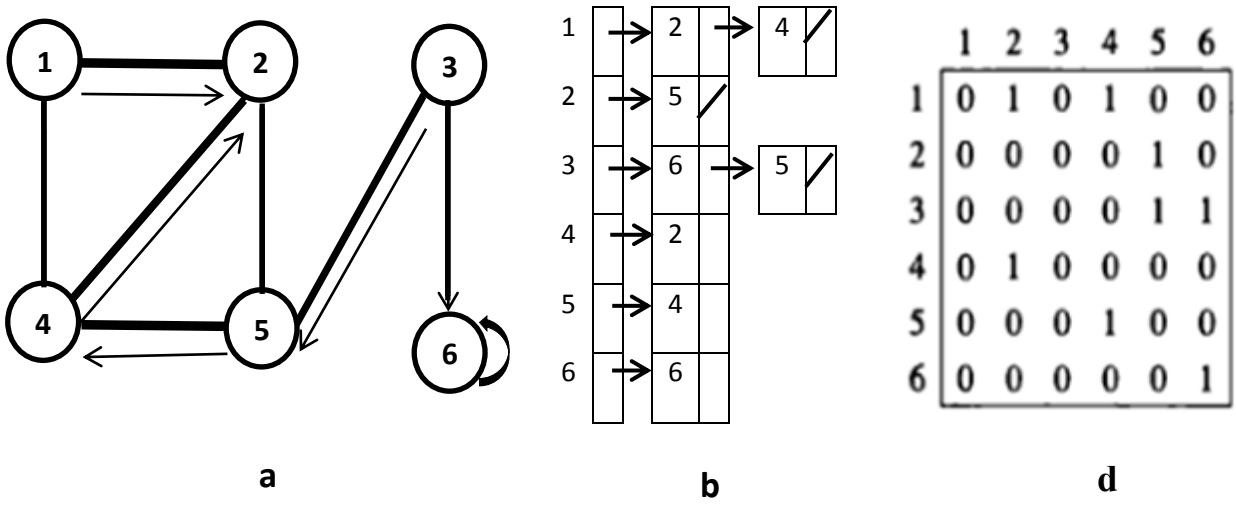
$G = (V, E)$ graflarni ikki xil usulda tasvirlash mumkin: uchlar ro‘yxatlari jamlanmasi va qo‘shni matritsalar. Ikki usul ham orentirlangan va orentirlanmagan graflarda qo‘llaniladi. Ko‘p hollarda uchlar ro‘yhatlari jamlanmasi foydalaniladi, chunki bu usul ruxsat etilgan graflarni $|E|$ $|V|^2$ dan kichik bo‘lgan hollarda ixchamroq tasvirlaydi. Oriyentirlangan graf- bu qirralarida harakat yo‘nalishlari bo‘yicha graf o‘qlari joylashgan, boshqacha qilib aytganda graf o‘qlariga strelkalar qo‘yilgan bo‘ladi. Biz odatdagi graflarda fundamental tushunchalar- uch darajalari, sikl va yo‘nalishlar kabi komponentlarni bo‘qliq deb qaraymiz. Qo‘shni matritsalar orqali tasvirlash ikkita uchlarini bog‘lovchi qirralarni topish hamda qalin graflar uchun samaraliroqdir.



a **b** **d**

23-chizma. Ikkita oriyentirlanmagan graf tasviri
a) G beshta uch va sakkiz qirrali oriyentirlangan graf. b) G qo‘shni ro‘yxatlar yordamida tasvirlangan d) G qo‘shni matritsalar yordamida tasvirlangan. [34]

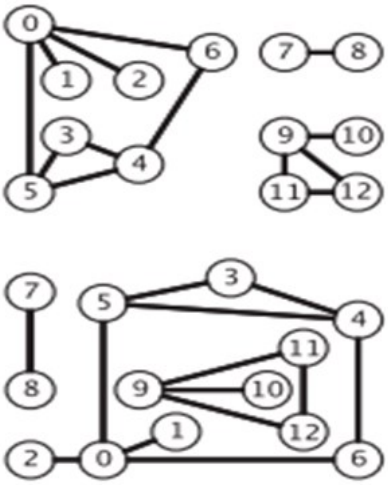
[34] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 590-p.



24-chizma. Ikkita oriyentirlangan graf tasviri

a) G olti uch va sakkiz qirrali oriyentirlangan graf. b) G qo‘shni ro‘yxatlar yordamida tasvirlangan d) G qo‘shni matritsalar yordamida tasvirlangan.

Quyidagi rasmda bitta grafning ikkita tasviri tasvirlangan. [35]



```
public class Graph

                 Graph(int V)                create a V-vertex graph with no edges
                 Graph(In in)                read a graph from input stream in
          int    V()                         number of vertices
          int    E()                         number of edges
         void    addEdge(int v, int w)       add edge v-w to this graph
Iterable<Integer> adj(int v)                 vertices adjacent to v
       String    toString()                  string representation
```

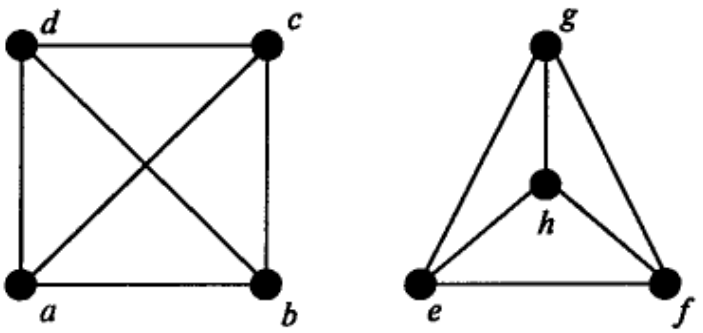
[35] Robert Sedgewick and Kevin Wayne. Algorithms. FOURTH EDITION. Princeton University. First printing, March 2011. Pp 522.

Oriyentirlanmagan graf API

Graflar izomorfizmi

Izomorfizm tushunchasi – graflar uchun alohida o‘ringa ega, graf uchlari uchlar tugunlarini bog‘lab turuvchi elastik iplar bilan bog‘langan deb faraz qilamiz.

Quyidagi 2 ta graflar izomorf ekanligini ko‘rsatamiz.(25-chizma). [36]



25-chizma. Izomorf graflar

Haqiqatan ham, $a \to e, b \to f, c \to g, d \to h$ izomorf akslantirishda birinchi graf d-uchidan rasm markaziga siljishini ko‘rishimiz mumkin.

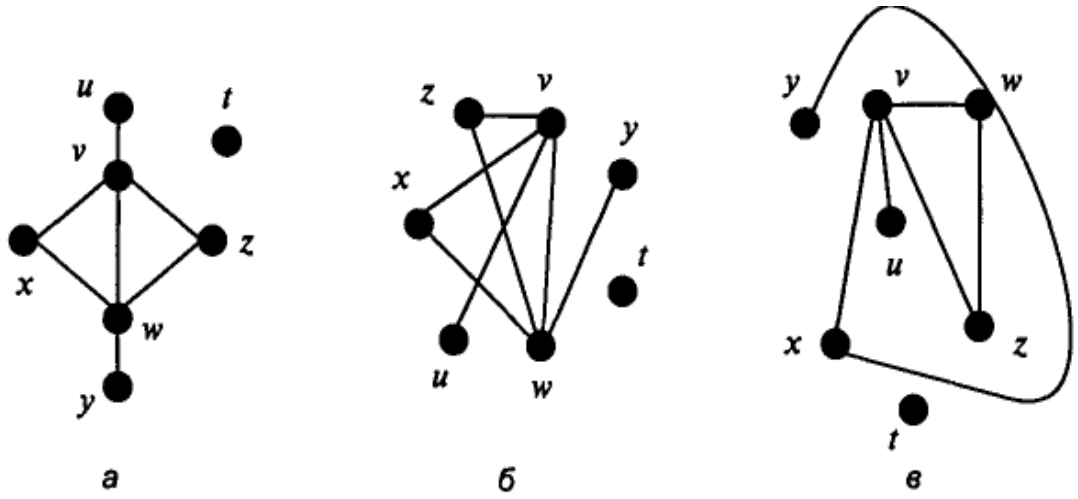
Graf uchlari darajasi

V graflar uchlari darajasini bildiruvchi d_v yoki $\deg(v)$ uning qirralar soniga teng bo‘ladi, ya’ni

$$d_v = \deg(v) = |N(v)|$$

Shuningdek, agar G_1 ning uchlari w,v,x,z,u,y,t tartibda joylashgan bo‘lsa, u holda ularga mos. 4,4,2,2,1,1,0 ya’ni d_w=4, d_v=4, d_x=2 va hokazo darajalar mavjud.

[36] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 109-с.



26-chizma. Tarmoq izomorfizmi

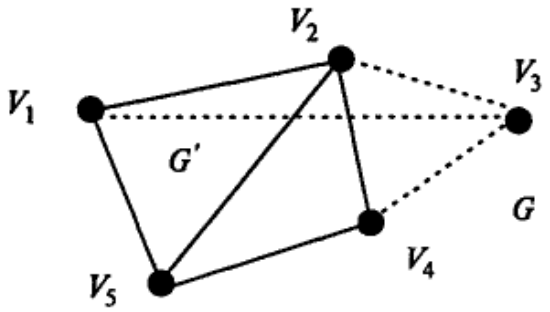
Graf uchi darajasini aniqlashda halqa ikki marotaba hisobga olinadi.

Agar **deg(v)=1** ga teng bo‘lsa, u holda V uch qulflangan deyiladi.

Agar **deg(v) = 0** ga teng bo‘lsa, u holda v-izolyatsiyalangan uch deyiladi.

Qism graf tushunchasi

Agar $V' \subseteq V$, va $E' \subseteq E$ bo‘lsa, u holda $G' = (V', E')$ tarmoq $G = (V, E)$ tarmoqning qism tarmog‘i bo‘ladi. Agar $v \in V$ bo‘lsa, u holda uchlari $V - \{v\}$, qirralari E dagi barcha qirralardan iborat G_{-v} qismtarmoqni bildiradi. [37]

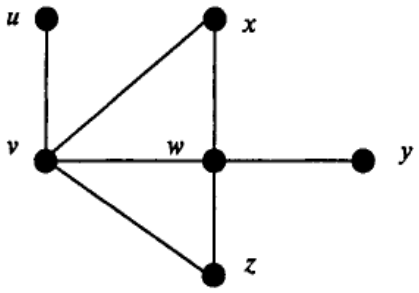


[37] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 109-110 сс.

$G = (V, E)$ grafning u_1 uchidan u_n yo‘nalish- bu har bir $i, 1 \le i \le n-1$ uchun $(u_i, u_{i+2}) \in E(G)$ bo‘lgan $W = u_1, u_2, \ldots, u_n$ uchlar ketma-ketligidir.

Agar $u_1 = u_n$ bo‘lsa, u holda W yo‘nalish yopiq, boshqa hollarda esa ochiq bo‘ladi.

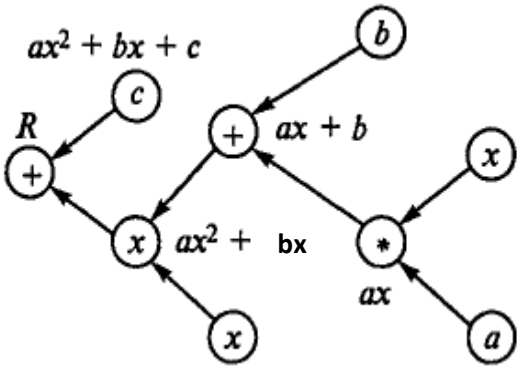
7-rasmda **uvxwywzvwvu** yo‘nalish bo‘lib; **uvxwz – yozuv yo‘l, vxwzv – sikl.** Har bir qirra juftliklari turli xil bo‘lgan yo‘nalishlar zanjir deb ataladi. Yopiq yo‘nalish sikl deb ataluvchi zanjir hisoblanadi.



27-chizma.Graflarda yo‘nalish va sikl[38]

Har bir uchlar juftliklari turli xil bo‘lgan yo‘nalish oddiy zanjir deb ataladi. Birinchi va oxirgidan tashqari, barcha uchlar juftliklari turli xil bo‘lgan sikl, *oddiy sikl* deyiladi.

Graflar hisoblash algoritmlarini tasvirlashda qulay vosita hisoblanadi. Misol tariqasida, Gorner sxemasida $ax^2 + bx + c$ kvadrat ko‘p hadni hisoblashni ko‘rib chiqamiz.(28-chizma)

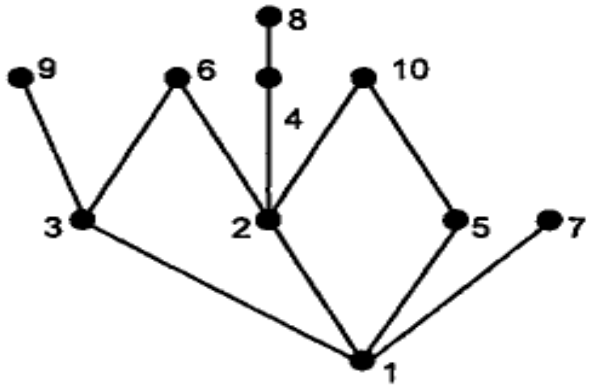


28-chizma. Gorner sxemasida kvadrat ko‘phadni hisoblash

[38] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 112- с.

$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ to‘plamdan bo‘lish munosabatini ifodalovchi grafni quramiz.

Ishlash prinsipi shundan iboratki, bor sondan ikkinchi songa yuqoriga olib boruvchi zanjir bo‘lsa, u holda ikkinchi son birinchi songa bo‘linadi. (29-chizma)



29-chizma. To‘plamda bo‘lish munosabatini tasvirlovchi graf

$G = (V, E)$ grafning qo‘shni ro‘yhat (adjacency-list representation)

[39]ko‘rinishida tasvirlanishda V dagi har bir uchdan $|V|$ ro‘yxatning Adj massiv ishlatiladi. Har bir $u \in V$ uch uchun $Adj[u]$ qo‘shni ro‘yxat $(u, v) \in E$ bo‘lgan barcha v uchlardan tashkil topadi. Qo‘shni ro‘yxat graf qirrasini tasvirlaganligi sababli, Adj massiv grafning atributi sifatida qaraladi.

Agar G oriyentirlangan graf bo‘lsa, u holda barcha qo‘shni ro‘yxatlarning yig‘indisining uzunligi. (u, v) qirraga $Adj[u]$ ro‘yxat-dagi v element mos kelganligi uchun $|E|$ ga teng bo‘ladi. Agar G oriyentirlanmagan graf bo‘lsa, u holda barcha qo‘shni ro‘yxatlarning yig‘indisining uzunligi (u, v) qirra u hamda v qo‘shni ro‘yxatlarda paydo bo‘lganligi uchun $2|E|$ ga teng bo‘ladi. Oriyentirlangan va oriyentirlanmagan graflarning ro‘yxatlar ko‘rinishiga tasvirlanishi uchun $\theta(V + E)$ hajmdagi xotira talab etiladi.

$G = (V, E)$ grafni qo‘shni matritsalar **(adjacency-matrix representation)** yordamida tasvirlashda graf uchlari ba’zi bir $1, 2, \ldots, |V|$ sonli tartibda nomerlangan deb taxmin qilinadi.

[39] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 590-p.

Bu vaziyatda qo‘shni matritsalar yordamida **G** graf tasvirlash $|V|\times V$ o‘lchamli $A=(a_{ij})$ ko‘rinishidagi matritsani o‘zida aks ettiradi.

$$a_{ij}=\begin{cases}1, & agar\ (i,j)\epsilon E,\\ 0 & aks\ holda\end{cases}$$

30-chizmada, a, b oriyentirlangan va oriyentirlanmagan graflar qo‘shni matritsalari ko‘rsatilgan. Graflar qo‘shni matritsalari grafning qirrasiga bog‘liq bo‘lmagan holda $\theta(V^2)$ xotirani talab qiladi.

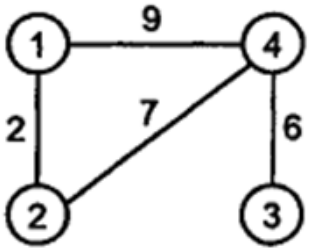
30-(b) chizmadagi qo‘shni matritsalarning asosiy diagonalga nisbatan simmetrikligiga e’tibor bering graf oriyentirlanmaganligidan $(u,v)\ va\ (v,u)$ bir xil yoqga ega ekanliklarini bildiradi orientirlanmagan grafning qo‘shni matritsalari $A=A^T$ transponirlangan qo‘shni matritsalarga mos keladi. Ilovalar qatorida ushbu xossa asosiy diagonal va undan yuqorida joylashgan faqat matritsa elementlarini saqlash imkonini beradi, bu esa ikki marta zarur xotira sig‘imini kamaytirish imkonini beradi.

Uchlari nishonlar bilan belgilangan M uchga ega ixtiyoriy $G=(V,E)$ graf $M\times M$ o‘lchamli matritsa ko‘rinishida tasvirlanadi.

Agar G grafning uchlari $v_1,v_2,\dots,v_m$ nishonlar bilan belgilangan bo‘lsa, u holda A qo‘shni matritsa quyidagicha aniqlanadi:

$$A=\begin{cases}K, agar\ V_i,V_j\ qo'shni\ bo'lsa\\ 0,\ agar\ V_i,V_j\ qo'shni\ bo'lmasa\end{cases}$$

Bu yerda K – uchlarni bog‘lab turuvchi qirraning o‘lchami. Chizmada graf va qo‘shni matritsalar tasvirlangan.



a

A=

	V_1	V_2	V_3	V_4
V_1	-	2	-	9
V_2		-	-	7
V_3	-	-	-	6
V_4	9	7	6	-

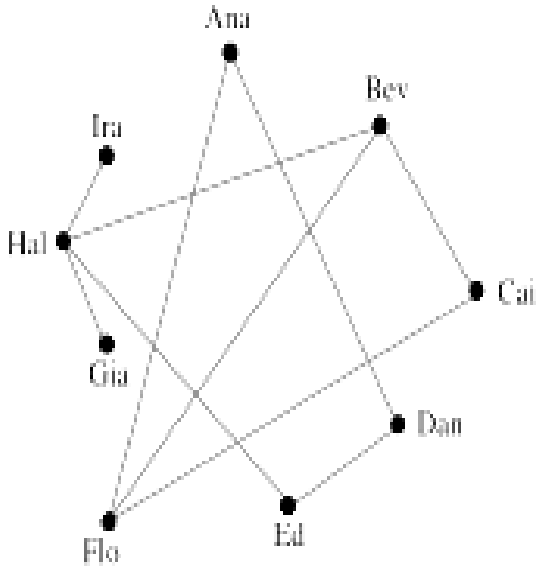
b

30-chizma. (a) Graf, (b) qo‘shni martitsa

Murakkab bo‘lmagan graflarni grafik sxemalar orqali ifodalash maqsadga muvofiqdir, u yerda uchlari nuqtalardan, qirralari esa ularni birlashtiruvchi chiziqlardan iboratdir.

Graflarga misollar

Name	Past Partners
Ana	Dan, Flo
Bev	Cai, Flo, Hal
Cai	Bev, Flo
Dan	Ana, Ed
Ed	Dan, Hal
Flo	Cai, Bev, Ana
Gia	Hal
Hal	Gia, Ed, Bev, Ira
Ira	Hal

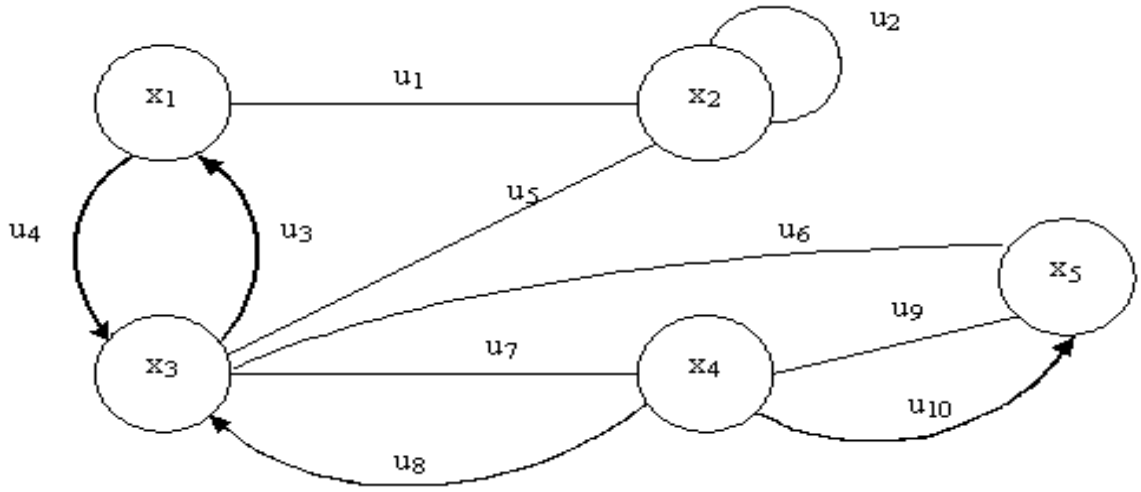


Shunday qilib, graf erkin konstruksiyalardir. Bunda ikki uchlari orasidagi bog‘lanishning bo‘lishi muhimdir, bir xilda ushbu bog‘lanishning xarakteri muhimdir.

Agar x_1 va x_2 lar qandaydir qirraga (x_i , x_j) ga tegishli bo‘lsa, u holda ushbu qirra x_i va x_j “insindent” deyiladi, x_i va x_j lar esa qo‘shni nuqtalar deyiladi. Agar qirra bir nuqtaga “insindent” bo‘lsa, u sirtmoq deyiladi.

Hech qanday qirraga “insindent” bo‘lmagan uch ajratilgan uch deyiladi. Agar grafda shunday uchlar bo‘lsaki ular ikki va undan ko‘p uchlar bilan birlashtirilgan bo‘lsa, bunday graf multigraf deyiladi. Ushbu uchga tegishli bo‘lgan qirralar soni uchning darajasini belgilaydi. 31-chizmada ko‘rsatilgan x_2 uch 6 darajaga ega, chunki unga $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7$, qirralar “insindent”dir, x_1 uchning darajasi 3, x_4 ning darajasi esa 1.

Ko‘pincha masalalarda uchlar orasidagi munosabat muhim rol o‘ynamaydi. Bunga misol tariqasida tartib munosabat bo‘lishi mumkin. Masalan, x_i x_j dan katta bu holda x_j x_i dan katta bo‘lishi mumkin emas. Demak, uchlar orasidagi munosabat ma'lum mo‘ljalga egadir. Bunday graflarni mo‘ljalga ega graflar yoki orientirli graflar deymiz.



31-chizma

In 3 and 4, draw pictures of the specified graphs.

3. Graph G has vertex set $\{v_1, v_2, v_3, v_4, v_5\}$ and edge set $\{e_1, e_2, e_3, e_4\}$, with edge-endpoint function as follows:

Edge	Endpoints
e_1	$\{v_1, v_2\}$
e_2	$\{v_1, v_2\}$
e_3	$\{v_2, v_3\}$
e_4	$\{v_2\}$

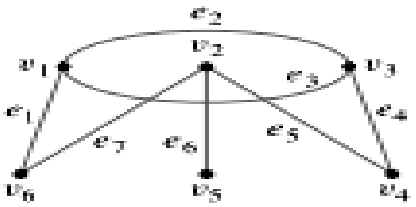
4. Graph H has vertex set $\{v_1, v_2, v_3, v_4, v_5\}$ and edge set $\{e_1, e_2, e_3, e_4\}$ with edge-endpoint function as follows:

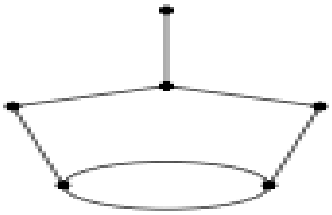
Edge	Endpoints
e_1	$\{v_1\}$
e_2	$\{v_2, v_3\}$
e_3	$\{v_2, v_3\}$
e_4	$\{v_1, v_5\}$

Quyidagi chizmalarning bir xil ekanligini ko‘rsating

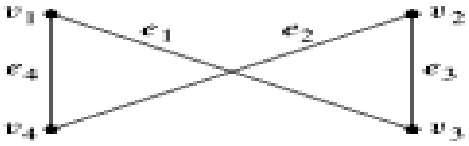
In 5–7, show that the two drawings represent the same graph by labeling the vertices and edges of the right-hand drawing to correspond to those of the left-hand drawing.

5.

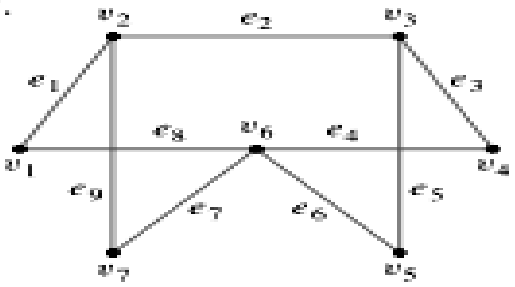


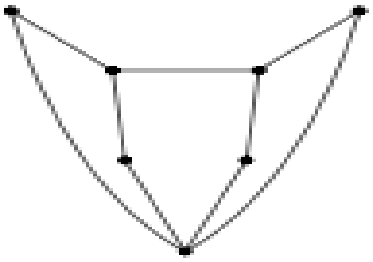


6.



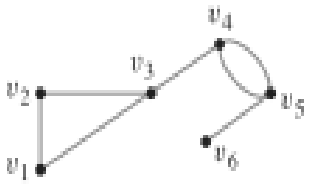
7.



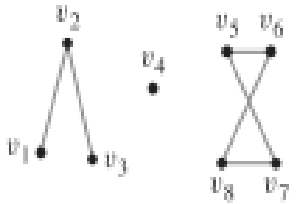


Quyidagi graflarning qaysi biri bog‘langan.

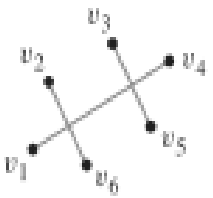
Which of the following graphs are connected?



(a)



(b)



(c)

III BOB. PASCAL DASTURLASH TILI VA UNING IMKONIYATLARI

11-§. Pascal tili, dasturlash tilining alifbosi, buruqlar tizimi va operatorlari

Muhitning birinchi satri – sarlavha satri deb yuritiladi

va odatda shu satrdan so‘ng menyular qatori joylashadi. Menyular qatorida joylashgan

har bir ichki menyuda qator vazifalarni bajaruvchi buyruqlar joylashgan. Undan pastda asosiy ish maydoni va holat satri joylashgan. Shuningdek, muhitning asosiy elemenlaridan biri bu oynalar hisoblanadi.

Turbo Pascalning professional programmalash - 7 versiyasi (Turbo Pascal Professional) Borland International Ins firmasining bir nechta disketalarida beriladi. Shu disketalarning biri «INSTALL COMPILER» deb nomlanadi va bu disketada INSTALL.EXE programmasi mavjud. Turbo Pascalni muhitini o‘rnatish uchun yuqorida ta’kidlangan disketani diskovodga qo‘yib, programmani

ishga tushiriladi. Programma sizdan bir necha savollarga javob berishni taklif qiladi:

• qaysi diskovoddan Turbo Pascalni ishga tushirmoqchisiz?

• Turbo Pascalni vinchesterga o‘rnatasizmi?

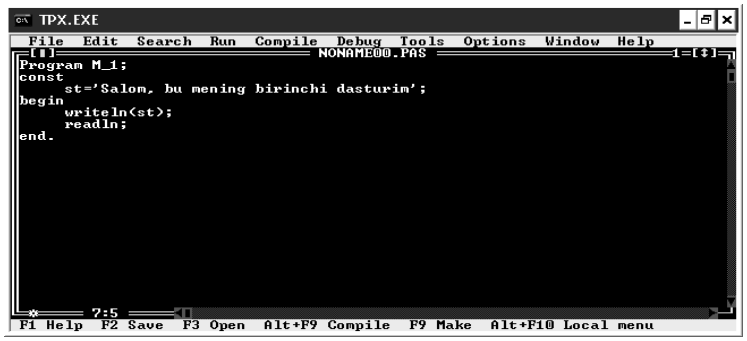
• qaysi kataloglarda Turbo Pascalning ishchi fayllarini joylashtirish lozim? Agar sizda yetarli asos bo‘lmasa, taklif qilingan katalog ismlari bilan chegaralangan ma’qul. Bu holda sizdan faqatgina, disk ismini o‘zgartirish talab etiladi. Shundan so‘ng, INSTALL.EXE programmasi o‘zining ishini davom ettiradi va uning so‘rovi bo‘yicha navbat bilan, ko‘rsatilgan nomli disketalarni diskovodga qo‘yiladi. Natijada Turbo Pascal ishga tayyor holga keladi. Faraz qilaylik, Turbo-Pascalni o‘rnatishda odatdagi «S» diskning o‘rniga «D» diskni tanladingiz, bu holda sistema quyidagi kataloglarda joylashdi: D TP TURBO D TP TVDEMOS D TP TVIZION D TP UTILS Turbo Pascal dasturi qattiq diskda joylash-gan TP\Bin\TPX.exe (Turbo.exe) fayli orqali ishga tushiriladi.

Turbo Pascal oynasi ikki xil holatda ishlaydi:

1. Turbo Pascal muhiti oynasining umumiy ko‘rinishi.

Sarlavha satri

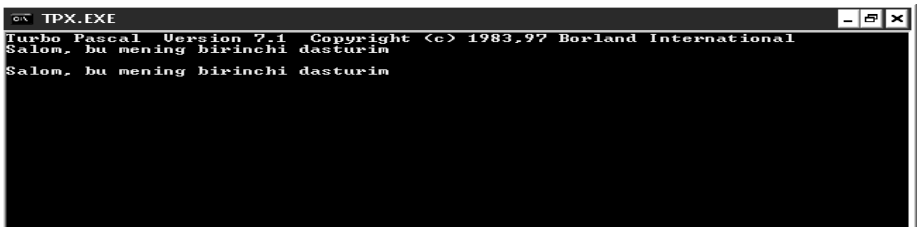
Ishchi soha(dastur matnini kiritish va tahrirlash oynasi)



Holat satri

Matn muhariri oynasi dastur matnini kiritish va matnni tekshirish uchun bir qancha qulayliklar yaratadi. Turbo.exe fayli ishga tushirilish bilan ekranda Turbo Pascal matn muharriri oynasi paydo bo‘ladi.

Dastur natijasini ko‘rish oynasi. Dastur natijasini olgandan keyin <Enter> tugmasi orqali matn muharriri oynasiga qaytib o‘tish mumkin.



Turbo Pascal dasturida asosiy fayllar va funksional tugmalar

Turbo Pascalning asosiy fayllari *D* : *TP* \ katalogida joylashgan bo‘lib, ular quyidagilardir:

Turbo.exe - programmani yaratish uchun lozim bo‘lgan integrallashgan muhit (Turbo Pascal Integrated Development Environment) fayli;

Turbo.hlp - tezkor yordam ma’lumotlarini saqlovchi fayl;

Turbo.tp - Turbo.exe dasturi foydalanadigan sistema konfiguratsiyasining fayli;

Turbo.tpl - Turbo Pascalning rezident modullari (Resident units);

Tptour.exe - integrallashgan muhitda ishlashni tanishtiruvchi dastur. Bulardan tashqari, *D* : *TP* \ *BGI* katalogida Pascalning grafik rejim ishini ta’minlovchi fayllar mavjud:

Graph.tpu - Turbo Pascalning barcha grafik dasturlarini ishlashi uchun zarur bo‘lgan modul;

BGI kengaytmali bir nechta fayl-videosistemalarning turli tiplari bilan ishlashni ta’minlovchi drayverlar;

CHR kengaytmali bir nechta fayl - vektorli shriftlarni o‘z ichiga oluvchi fayllar.

Turbo Pascal muhitini boshqarish uchun funksional tugma-lardan (<F1>, <F2>, …, <F10>) foydlaniladi. Quyida funksional tugmalar keltirilgan:

<F1> - Turbo Pascal dasturi yordam xizmatidan foydalanish. (Help -Yordam);

<F2> - Dastur matnini faylga saqlash;

<F3> - Faylni ochish;

<F4> - Dasturni ishlashini teksirish jarayonida qo‘llaniladi. Dastur matnidagi kursor o‘rnatildan satrgacha bo‘lgan satrlarni bajarilashini satrma-satr tekshiradi;

<F5> - Faol oynani butun ekran bo‘ylab yoyish;

<F6> - Keyingi tahrirlanuvchi oynaga o‘tish;

<F7> - Dasturni ishlashini teksirish jarayonida qo‘llaniladi. Agar dasturda protsedura (funksiya) ga murojaat bo‘lsa, protsedura (funksiya) ga kirib, birinchi operatordan oldin to‘xtaydi.

<F8> - Dasturni ishlashini teksirish jarayonida qo‘llaniladi. Dastur matnini satrma-satr bajarilishini tekshiradi. Agar dasturda protsedura (funksiya) ga murojaat bo‘lsa, protsedura (funksiya) ga kirmasdan dastur ishlashini tekshiradi;

<F9> - Dasturni Compilatsiya qiladi, lekin ishlashini ta’minlamaydi.

<F10> - Bosh menyu bo‘limlari bilan ishlash;

<CTRL>+<F9> - dasturni Compilatsiya qiladi, ishlashini ta’minlaydi va Turbo Pascal muhitiga qaytaradi;

<ALT>+<F5> - dastur matni oynasidan dastur natijalarni ko‘rish oynasiga o‘tish;

<ALT>+<X> - Turbo Pascal muhitidan chiqish;

Dastur matnini kiritishda boshqarish tugmalaridan foydalanish.

Dastur matnini kiritish jarayonida bir qancha boshqarish tugmalaridan foydalaniladi:

<Enter> - yangi (keyingi) satrga kursorni o‘tkazish;

<PageUp> - bir sahifa yuqoriga;

<PageDown> - bir sahifa pastga;

<Home> - satr boshiga kursorni o‘tkazish;

<End> - satr oxiriga kursorni o‘tkazish;

<CTRL>+<PageUp> - matn boshiga o‘tish;

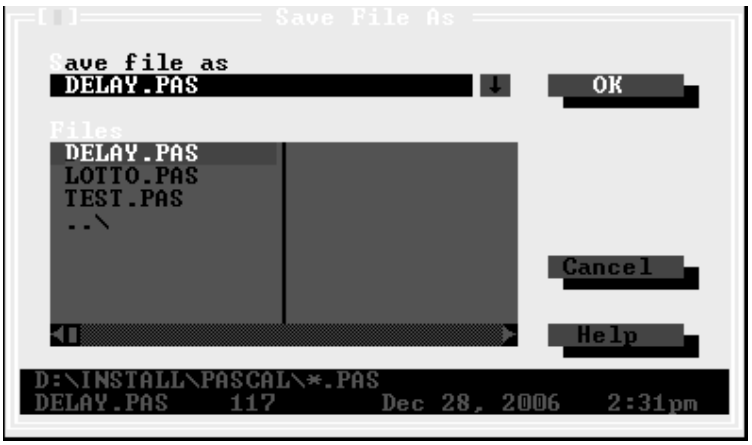
<CTRL>+<PageDown> - matn oxiriga o‘tish;

<Delete> - kursor joylashgan belgini o‘chiradi;

<Backspace> - kursordan chapda joylashdan belgini o‘chiradi;

Turbo Pascal muhitida dastur mantini kiritib, natijalar olingandan so‘ng, ushbu dasturdan keyinchlik foydalanish uchun dasturni xotirada faylga saqlash zarur bo‘ladi. <F2> tugmasi orqali yangi faylni joylashish manzili, nomi ko‘rsatiladi va <Enter> tugmasi bosiladi. Diskda saqlangan fayl *.pas kengaytmasini oladi. <F3> tugmasi orqali diskdagi avvaldan mavjud *.pas fayllarni ochish (yuklash) mumkin. Dasturni saqlashda TP katalogidagi tizimli fayllar bilan sizning fayllaringiz aralashib ketmasligi zarur, aks holda

siz uchun kerakmas deb hisoblandan fayl Turbo Pascal dasturining tizimli fayli bo‘lishi ehtimoli bor. Shuning uchun o‘zingiz yaratgan fayllarni TP katalogidan boshqa katalogda saqlash taklif etiladi.



Turbo Pascal muhiti menyulari.

Fayl menyusi. Fayllar bilan ishlash imkonini beradi: New – Yangi fayl yaratish; Open – diskda saqlangan faylni ochish muloqotli oynasini ochadi. Ushbu amalni <F3> tugmasi orqali bajarish mumkin; Save – Joriy tahrirlanayotgan dastur matnini faylga saqlash. Ushbu amalni <F2> tugmasi orqali bajarish mumkin; Save As– Joriy tahrirlanayotgan dastur matnini yangi nom va yangi manzili bilan saqlash muloqotli oynasini chaqiradi; Print – Dastur matnini chop etish; Exit – Turbo Pascal muhitidan chiqish; Fayl menyusidagi boshqa (Save All, Change dir, Print Setup, Dos shell) buyruqlardan juda kam foydalanilishi uchun ushbu buyruqlarni o‘tkazib yubordik;

Edit menyusi. Dastur matnini tahrirlash uchun bir qancha buyruqlarni taklif qiladi: Undo – Oxirgi amalni bekor qilish (ALT+Backspace); Redo – Bekor qilingan amalni qayta tiklash; Cut - beldilangan dastur qismini “Qirqish” (Shift + Del); Copy - beldilangan dastur qismini nusxalash (CTRL+Ins); Paste – Cut yoki Copy buyruqlari yordamida xotiraga olingan dastur qismini kursor turgan joyga o‘rnatish; (Shift+Ins); Clear – Belgilangan matn qismini o‘chirish (Shift+Ins); Show clipboard – Bufer xotiraga olingan dastur matni bo‘laklarini ko‘rish;

Search menyusi. Dastur matnidagi matn bo‘laklarini qidirish, xatoliklar mavjud bo‘lgan satrga o‘tish imkoniyatini beradi: Find – Qidirilayotgan matn qismi va qidirish usulini kiritish uchun

muloqotli oyna taklif qiladi; Replace – Biror matn bo‘lagini boshqa matn bo‘lagi bilan almashtirish uchun muloqotli oyna taklif qiladi; Search again – matn bo‘lagini qaytadan qidirish; Goto line number– berilgan tartib raqamli satrda o‘tish; Show last compile error– dastur matnidagi xato satrni va xatolik haqidagi ma’lumotni ko‘rsatadi; Find error – Dastur matnidagi xato satrni aniqlaydi. Find procedure – dastur matnidagi protsedurani qidirish muloqotli oynasini taklif qiladi. Muloqotli oynada dastur matnida ishtirok etuvchi protsedura nomi kitiladi.

Run menyusi. Dasturni ishga tushirishni amalga oshiradi. Run – Agar dasturning boshlag‘ich matnida o‘zgarish kiritilmagan bo‘lsa, dasturni ishlashini ta’minlaydi. Agarda dastur matniga o‘zgartirish kiritilgan bo‘lsa, avval dasturni qayta kompilaytsiya qilish zarurligi haqida so‘raladi. Dasturni kompilatsiya qilinishini bekor qilish uchun <Ctrl+break> tugmalari bosiladi. Step over - Qism dasturga kirmasdan dasturning asosiy qismini satrma-satr bajailishini tekshiradi (F8); Trace into– Dasturning asosiy qismida qism dasturga murojaat qilinganda qism dasturni satrma-satr bajarilishini tekshiradi (F7); Goto kursor – Kursor o‘rnatilgan satrgacha dasturni bajarilishini ta’minlaydi; Program reset – yuqoridagi tekshirish jarayolarini bekor qiladi. (CTRL+F12);

Compile menyusi. Dastur matni kiritilgandan so‘ng, uni kompilyatsiyaga yuborish zarur. Ushbu amalni ikki xil usulda amalga oshiriladi: Compile menyusidan Compile buyrug‘ini tanlash; <ALT>+<F9> tugmalarini bosish; Avval kompilyator dastur matnining sintaktik xatolarini tekshiradi. Agar xatolik aniqlansa, kompilyator xatolikka yo‘l qo‘yigan satrda o‘z ishini to‘xtatadi va xatolikka mos keluvchi xabarni beradi. Aniqlangan xatoliklarni dastur tuzuvchi to‘g‘rilaydi va qaytadan Compilyatsiyaga yuboradi. Dastur matni kompilyatsiyadan muvaffaqiyatli o‘tgandan so‘ng dastur natijasini olib, natijalarni tahlil qilish imkoniyati paydo bo‘ladi.

Debug menyusi. Dastur ishini teksirishda to‘xtash nuqtalarini o‘rnatish, dasturda ishlatiluvchi o‘zgaruvchilarni kiritish va mos qiymatlarini ko‘rish imkoniyatlarini beradi;

Tools menyusi: Turbo Pascal muhitidan chiqmasdan qo‘shimcha uskunalardan foydalanish imkoniyatini beradi;

Options menyusi: Turbo Pascal dasturlash muhitini kerakli parametrlarini o‘rnatish uchun buyruqlar taklif qiladi.

Window menyusi: Oynalarni ochish, yopish, faollashtirish va ekran maydoniga joylashtirishga yordam beradi.

Help menyusi. Turbo Pascal dasturi muhiti, operatorlari, funksiya-proseduralari, kutubxonalari haqida ma’lumot olish (<ALT>+<H>); Index – Yordamchi tizimni tematik ko‘rsatkich orqali ko‘rsatadi; Contents - Yordamchi tizimni mundarijasini ochadi.

Bosh menyu quyidagi bo‘limlarni o‘z ichiga oladi:

File *.pas kengaytmali fayllari va Turbo Pascalga tegishli bo‘lgan kataloglar ustida turli amallar bajaradi;

Edit yuqori menyudan matnlarni tahrirlash oynasiga qaytish;

Run programmani turli ko‘rinishda ishga yuklaydi va natijalar oynasiga o‘tishni ta’minlaydi;

Compile programmani kompilyatsiya qilish va uni tekshirishni ta’minlaydi;

Options Turbo Pascal muhitini va undagi programmalarga tegishli xususiyatlarni tahrirlaydi;

Debug programmadagi protseduralarni izlash, hisoblash jarayonlari va o‘zgaruvchilarni tekshirish, chaqirilgan stekni ko‘rish va shunga o‘xshash amallarni bajaradi;

Break + Watsh agarda *Debug+ Integrated Debugging* ustanovkasi *On* holatda bo‘lsa, u holda *Break + Watsh* menyusi *Watsh window* (tekshirish oynasi) punktini oynaga o‘rnatadi yoki olib tashlaydi; Quyida bosh menyuning mos ostki menyulariga qisqacha ta’riflar berilgan:

***FILES* BO‘LIMI:**

Load (F3) fayl yoki dasturni chaqirish uchun xizmat qiladi; *Pisk (ALT + F3)* tanlashni belgilaydi; *New* yangi dastur uchun oynani tozalaydi;

Save (F2) ekrandagi dasturni doimiy xotirada saqlaydi. *Write to* ishchi faylni yangi nom bilan saqlaydi; *Change dir* faol katalogni almashtiradi; *Os shell* vaqtincha OS ga chiqishni tashkil qiladi; *Quit (Alt + X)* TURBO PASCAL muhitidan chiqish.

RUN BO‘LIMI:

Run (Ctrl + F9) yuklangan dasturni ishga tushiradi; *Program reset (Ctrl+ F2)*oldindagi parametrlarni tahrirlash; *Goto sursor (F4)*

kursor turgan joyga qaytishni amalga oshiradi; *Trase into (F7)* dasturni qadamlab bajarish imkoniyatini hosil qiladi; *Setup over (F8)* protseduraga kirishni ta'minlaydi; *User ssreen(Alt-F5)*natijalar oynasiga o'tish.

COMPILE BO'LIMI:

Compile (Alt + F9) dasturni kompilyatsiya qiladi; *Make (F9)* dasturni kompilyatsiya qilishga tayyorlaydi; *Build* dasturni kompilyatsiya qilishga tayyorlaydi (*Make* dan farqi, dasturga tegishli barcha fayllarni tekshiradi); *Destination* dasturni doimiy yoki tashqi xotiraga kompilyatsiya bo'lishini ta'minlaydi; *Find error* dastur xatoligini izlaydi; *Primary files* qaysi **.pas* fayl kompilyatsiya qilinishini aniqlaydi; *Get info* ishchi oynadagi **.pas* fayli haqidagi ma'lumotlarni beradi.

OPTIONS BO'LIMI:

Comptler dastur xususiyatlarini o'zgartiradi; *Linker *.exe* fayldagi ishlamaydigan kodlarni o'chiradi; *Environment* dasturchi uchun ishchi maydon hosil qiladi; *Direstories* turli tipdagi fayllarni qaysi katalogda ishlatish lozimligini o'zida saqlaydi; *Parameters* buyruqlar satrida kiritiladigan parametrlarni o'zida saqlaydi; *Save options* muhit xususiyatini tashqi xotiraga saqlab qo'yish; *Retrieve options* muhit xususiyatini tiklash.

DEBUG BO'LIMI:

Evaluate har qanday o'zgaruvchining qiymatini ko'rish imkoniyatini yaratadi; *Call stask* dastur bajarilish vaqtida joriy chaqiruvchi stekni ko'rsatadi; *Find prosedure* protsedurani izlash; *Integrated debugging* dasturni qadamlab bajarilishi uchun yo'l ochib beradi; *Standalone debugging* dasturni **.exe* faylga aylantirish uchun yo'l ochib beradi; *Display swapping* ekran xususiyatini o'rnatadi; *Refresh display* ekran xususiyatini aktivlashtiradi.

BREAK + WATSH BO'LIMI:

Add watsh (Ctrl + F7) ko'rish oynasiga zarur ma'lumotlarni qo'shish imkoniyatini hosil qiladi; *Delete watch* ko'rish oynasidagi belgilangan punktni o'chiradi; *Edit watch* ko'rish oynasini tahrirlaydi; *Remove all watchs* ko'rish oynasidagi barcha punktlarni o'chiradi; *Toggle breaskpoint* tekshiriluvchi nuqtalarni o'rnatish; *Clear all breaskpoint* barcha tekshiriluvchi nuqtalarni o'chirish; *View next breaskpoint* keyingi tekshiriluvchi nuqtaga o'tish.

Yuqoridagilarga qo‘shimcha ravishda dastur matnini tahrirlash uchun funksional tugmachalardan ham foydalanish imkoniyati mavjud bo‘lib, ular quyidagilardan iboratdir:

Ctrl+K+B	blok boshini belgilash;
Ctrl+K+K	blok oxirini belgilash;
Ctrl+K+C	kursor turgan joyga blokdan nusxa olish;
Ctrl+K+V	kursor turgan joyga blokni ko‘chirish;
Ctrl+K+Y	Bloka olingan matnni o‘chirish;
Ctrl+Y	kursor turgan satrni o‘chirish;
Ctrl+K+H	Blokni olib tashlash yoki qo‘yish.

Menyular tarkibidagi amallarni ma’lum klavishlarni bosish orqali ham bajarish mumkin. Quyida asosiy amallarni bajarishga mo‘ljallangan klavishalar keltirilgan:

Klaviatura tugmalari:	Vazifalari:
F3	Ochish (tashqi xotiradagi faylni yuklash)
F2	Saqlash
Alt+F3	Aktiv oynani yopish
F6	Bir oynadan ikkinchi oynaga o‘tish
Ctrl+F9	Dasturni ishga tushirish
Alt+F5	Dastur natijasini ekranda ko‘rish
Alt+F9	Dasturni kompilyasiya qilish
Alt+x	chiqish

Yuqoridagilarga qo‘shimcha qilgan holda dastur matnini tahrirlash uchun ishlatiladigan klavishlar quyidagilardan iboratdir:

Yo‘nalish klavishal-ari (→ ← ↑ ↓)	Yurgichni kerakli yo‘nalishda siljitish;
Shift+(→ ← ↑ ↓)	Yurgich turgan joydan boshlab tanlangan yo‘nalishda dastur matn qismini belgilash;
Ctrl+Insert	dastur matnini belgilangan qismi nusxasini xotiraga olish;
Shift + Insert	Xotirada olingan qismni dastur mantining yurgich turgan joyga o‘rnatish;
Shift +Delete	dastur matnining belgilangan qismini o‘chirish;

Dasturchilar deyarli menyular satriga murojaat etmasdan, yuqorida keltirilgan klavishlardan foydalanadi. Bu esa vaqtni tejash bilan birga dastur tuzuvchini e'tibori faqat dastur matnida bo'lishini ta'minlaydi.

Turbo-Pascal dasturlash tili va uning alifbosi

Pascal tili 1969 yili N.Virt tomonidan yaratilib mashhur olim Blez Pascal nomi bilan ataldi. Bu til N.Virtning o'ylashi bo'yicha programmalashning zamonaviy texnologiyasiga va uslubiga, strukturali programmalash nazariyasiga asoslangan va boshqa programmalash tillaridan muayyan yutuqqa ega til bo'lishi lozim edi. Mazkur til:

- Programmalashtirish konsepsiyasini va strukturasini sistemali va aniq ifodalaydi;
- Programma tuzishni sistemali olib borish imkonini beradi;
- Programma tuzish uchun boy termin va struktura sxemalariga ega;
- Yo'l qo'yilgan xatoliklarni tahlil qilishning yuqori darajadagi sistemasiga ega.

1981 yili Pascal tilining xalqaro standarti taklif etildi va IBM PC tipidagi shaxsiy kompyuterlarda Pascal tilining Borland firmasi tomonidan ishlab chiqilgan Turbo-Pascal oiladosh tili keng qo'llanila boshlandi. Hozirda Turbo-Pascalning bir qancha versiyalari yaratilib, yuqori darajadagi programmalar yaratish imkoniyatlari borgan sari kengaytirilib borilmoqda:

- 4.0 versiyasidan boshlab programma yozishni, tahrirlashni va natijalar olishni osonlashtirish uchun yangi integrallashgan muhit hosil qilindi;
- 5.5 versiyasining paydo bo'lishi bilan Turbo-Pascalda obyektli programmalash imkoniyati paydo bo'ldi;
- 6.0 versiyasidan boshlab esa Pascal programmasi ichiga quyi programmalash tili bo'lmish Assembler tilida yozilgan programmalarni qo'shish holati hosil qilindi. Shu bilan bir qatorda *tilning integrallashgan muhiti* ham bir qator o'zgarishlarga ega bo'ldi.

Integrallashgan muhit-dasturlashga yordamlashuvchi dastur bo'lib, u quyidagi asosiy vazifalarni bajarishi lozim:

-dastur matnini kiritish imkonini beradi;

-vaqt-vaqti bilan kiritayotgan dastur matnini diskda saqlab turishi;

- dasturni ishga tushirish uchun translyatorga ega bo‘lishi;

-sintaktik xatoliklarni aniqlash vositasiga ega bo‘lishi kerak.

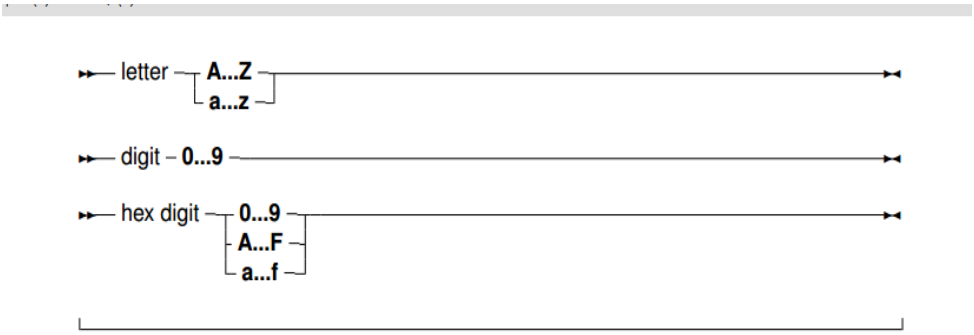
- 7.0 integrallashgan muhiti sanab o‘tilgan vazifalardan tashqari yana bir qancha vazifalarni ham amalga oshirish imkonini beradi.

Turbo- Pascal dasturlash tilining alifbosi

Ma’lumki, har qanday tilni o‘rganish uning alfavitini o‘rganishdan boshlanadi. Tilning alfaviti - shu tilgagina tegishli bo‘lgan asosiy belgilari va tushunchalar to‘plamidan iborat bo‘ladi. Pascal tilining alfavitini tashkil etuvchi asosiy belgilar jamlamasini 3 guruhga ajratish mumkin: harflar, raqamlar va maxsus belgilar.

Til alfavitining metalingvistik (Bekus - Naur) formulasi quyidagicha bo‘ladi:

<asosiy belgi>::=<harf>|<raqam>|<maxsus belgi>



Harf sifatida katta va kichik lotin harflari ishlatiladi. Ya’ni 26 ta **lotin alifbosi harflari**: A, a, B, b, Cc, Dd, Ee, Ff, Gg, Hh, Ii, Jj, Kk, Ll, Mm, Nn, Oo, Pp, Qq, Rr, Tt, Uu, Vv, Ww, Xx, Yy, Zz. Lekin, matnlar va programmaga izohlar yozish uchun kirill alifbosining bosh va kichik harflarini ham alfavitga kiritilgan.

Raqamlar sifatida oddiy **arab raqamlari** olingan: <raqam>::=0|1|2|3|4|...|9

Maxsus belgilar ko‘p sonli va bir jinssiz bo‘lganligi uchun ularni o‘z navbatida 4 ta guruhga ajratamiz:

<maxsus belgi>::=<arifmetik amal belgisi>|<solishtirish amali belgisi>|<ajratgich>|<xizmatchi so‘z>.

<arifmetik amal belgisi>::= * | / | + | -[40]

Bu amallar mos ravishda ko‘paytirish, bo‘lish, qo‘shish va ayirish belgilari hisoblanadi.

Solishtirish amallarining belgilari, ularning matematik ifodasi va amallarning ma’nosi 6-jadvalda o‘z ifodasini topgan. Bu yerda shu narsaga ahamiyat berish kerakki, ba’zi bir amallar ikkita belgi orqali ifodalangan.

6-jadval

Solishtirish amali belgisining Pascaldagi yozilishi	Amalning matematik ifodasi	Amalning ma’nosi
=	=	Teng
<>	≠	Tengmas
<	<	Kichik
<=	≤	Kichik yoki teng
>	>	Katta
>=	≥	Katta yoki teng

Ajratgichlar guruhini quyidagi belgilar tashkil qiladi:
<ajratgich>::= . | , | : | ; | (|) | [|] | { | } | ' | @| “”| ! | ? |%| $| &| |

Ajratgichlarning vazifalarini tilni o‘rganish davomida aniqlab boramiz.

Xizmatchi so‘zlar guruhi juda keng, shuning uchun bu so‘zlarni hammasini birdaniga yodlab, eslab qolish shart emas, balki ulardan foydalanish davomida ketma-ket eslab qolinaveradi:

<xizmatchi so‘zlar>::=and | array | begin | case | const | div | do | downto | else | end | for | function | goto | if | in | label | mod | nil | not | of | or | packed | program | procedure | record | repeat | set | then | to | type | until | var | while | with

Turbo pascal dasturlash tilining xizmatchi so‘zlari: [41]

[40] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 11-12

[41] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp-13.

absolute	file	object	string
and	for	of	then
array	function	operator	to
asm	goto	or	type
begin	if	packed	unit
case	implementation	procedure	until
const	in	program	uses
constructor	inherited	record	var
destructor	inline	reintroduce	while
div	interface	repeat	with
do	label	self	xor
downto	mod	set	
else	nil	shl	
end	not	shr	

Mantiqiy amallar: AND («VA» - mantiqiy ko‘paytirish amali);

OR (“YoKI”- mantiqiy qo‘shish amali);

NOT («EMAS» - mantiqiy inkor amali);

Barcha dasturlash dillari singari Pascal dasturlash tili ham o‘zining imlosi, qonun va qoidalariga ega bo‘lib, ular asosida yuqorida keltirilgan harflar, belgilar va amallar yordamida ko‘rsatma va buyruqlar tuziladi. Har bir ko‘rsatma yoki byuruq “;” (nuqtali vergul) belgisi bilan yakunlanadi.

Dastur yozishda quyidagilar qo‘llaniladi:

Konstantalar (o‘zgarmaslar)-dastur ishlashi davomida qiymati o‘zgarmaydigan miqdorlar;

O‘zgaruvchilar- dastur ishlashi davomida qiymati o‘zgaradigan miqdorlar;

Algebraik ifodalar-arifmetik amallar bilan bog‘langan o‘zgarmaslar, o‘zgaruvchilar va funksiyalar;

Operatorlar-dasturlash tilining biror tugallangan amalini berish uchun mo‘ljallangan buyrug‘i;

Funksiyalar va proseduralar-o‘z nomiga ega bo‘lgan alohida dastur qismlari;

Operatorlar

Operator tushunchasi tilning eng asosiy tushunchalaridan biri bo‘lib, har bir operator tilning yakunlangan jumlasi hisoblanadi va ma’lumotlar tahlilining tugallangan bosqichini ifodalaydi.

Operatorlarni ikki guruhga ajratish mumkin. 1-guruh operatorlarining tarkibida boshqa operatorlar qatnashmaydi va bu operatorlarni asosiy operatorlar deb ataladi. Asosiy operatorlar jumlasiga quyidagi operatorlar kiradi: o‘zlashtirish operatori, prosedura operatori, o‘tish operatori, bo‘sh operator. 2-guruh operatorlarining tarkibida esa boshqa operatorlar ham qatnashib, ular *tarkibiy operatorlar* deb ataladi. Ular jumlasiga quyidagi operatorlar kiradi: tashkiliy operator, tanlov operatori, takrorlash operatori, ulash operatori.

Masalani yechish algoritmida yuqoridagi ikki guruh operatorlarining ketma-ketligi cheklanmagan miqdorda qatnashishi mumkin. Bu ketma-ketlikdagi operatorlar o‘zaro ";" ajratish belgisi orqali ajratiladi, ya’ni programma matnining yozuvi alohida operatorlarga bo‘linadi. Shunday qilib, S orqali ixtiyoriy yozish mumkin bo‘lgan operatorni belgilasak, masala yechilishining algoritmi quyidagi ketma-ketlik bo‘yicha ifodalanishi mumkin:

S; S; ...;S.

Operatorlarning bu ketma-ketligi ularning programmada yozilish tartibi bo‘yicha bajariladi. Shunday qilib, operatorning izdoshi undan keyin yozilgan operator hisoblanadi. Operatorlar bajarilishining bu tabiiy ketma-ketligini faqat o‘tish operatori yordamida buzish mumkin. Tarkibiy operatorlarda esa operatorlarning bajarilish tartibi o‘ziga xos qoidalar bilan aniqlanadi.

Nomlar va identifikatorlar

Ma’lumki, ma’lumotlarning tahlili jarayonini ifodalovchi algoritm turli xil obyektlar (o‘zgarmaslar, o‘zgaruvchi miqdorlar, funksiyalar va hokazo) ustida ish olib boradi. Bu obyektlarga ularning vazifasi va qabul qiladigan qiymatlariga qarab maxsus ismlar beriladi. Shu ismlarni odatda, identifikatorlar deb ataladi. Identifikator deb harf yoki "_" belgisidan boshlanuvchi, harf, raqam va "_" belgisining ixtiyoriy ketma-ketligiga aytiladi:

<identifikator>::=<harf>|<identifikator><harf>|<identifikator><raqam>

Agar quyidagi oraliq tushunchani kiritsak:

<harf yoki raqam>::= <harf>|<raqam>

Yuqoridagi aniqlashni quyidagicha ham yozish mumkin:

<identifikator>::= <harf> {<harf yoki raqam>}.

Xizmatchi so‘zlardan identifikator sifatida foydalanish mumkin emas. Odatda identifikator so‘zining o‘rniga qulayroq va qisqaroq qilib ism deyish mumkin. Programmada qatnashuvchi obyektlarga ismlarni programma tuzuvchi o‘z ixtiyoriga ko‘ra tanlab olishi mumkin. Bir xil ism bilan bir necha xil obyektlarni nomlash mutlaqo mumkin emas. Turbo Pascal muhitida ismda qatnashuvchi belgilar soni (ism uzunligi) 63 ta belgidan oshmasligi kerak.

Ismlarga misollar:

_Burchak, _A1, Ahmad_Berdiyev, C, Summa, Time, A, S1, …

E’lonlar

Pascal tilining asosiy tushunchalaridan biri e’lon qilish hisoblanadi. Programmada qatnashuvchi barcha ob’yektlarning ismlari mos ravishda programmaning bosh qismida, ularning qanday tipdagi qiymatlar qabul qilishi mumkinligiga qarab, e’lon qilinib qo‘yilishi kerak. Pascal tilida e’lon qilishning 5 xil turi mavjud:

- nishonlar e’loni;
- o‘zgarmaslar e’loni;
- tip aniqlash uchun e’lon;
- o‘zgaruvchilar e’loni;
- prosedura va funksiyalar e’loni.

Umuman olganda, yuqorida sanab o‘tilgan e’lonlarning vazifalari ularning nomlaridan ham sezilib turibdi, e’lonning vazifalari esa keyinroq to‘la ochib beriladi.

O‘zgaruvchilar

O‘zgaruvchi, programma obyekti bo‘lib, turli xil qiymatlarni xotirada ma’lum nom bilan saqlab turish uchun ishlatiladi. O‘zgaruvchi o‘z qiymatini programmani bajarilish davomida, o‘zlashtirish operatori yordamida qabul qiladi. qabul qilingan qiymat, o‘zgaruvchiga boshqa yangi qiymat berilmaguncha saqlanib turiladi va yangi qiymat berilishi bilan eski qiymat butunlay o‘chib, yo‘q bo‘lib ketadi. Har bir o‘zgaruvchiga ma’lum bir tipga tegishli qiymatlarnigina qabul qilish huquqi beriladi. Boshqa tipdagi qiymatlarni o‘zlashtirishga urinish programmaning xatoligini ta’minlaydi. Quyida o‘zgaruvchilarni e’lon qilish keltirib o‘tamiz:

```
var
Value: Integer;
IsCorrect: Boolean;
```

A, B: Char;

O‘zgaruvchi - bu identifikatordir. Uning ismi o‘zgaruvchining qiymatiga murojaat qilishda ishlatiladi. Boshqacha aytganda, programma matnidagi ism, shu o‘zgaruvchining qiymatini ifodalaydi.

Pascalda o‘zgaruvchilarni e’lon qilayotganda boshlang‘ich qiymatni global o‘zgaruvchiga o‘zlashtirish mumkin:

```
var
Value: Integer = 10;
Correct: Boolean = True;
```
[42]

Dasturning umumiy ko‘rinishi

Pascalda *dastur quyidagi ikki qismdan tashkil topadi: tasvirlash qismi; asosiy qismi.*

```
program dastur ismi;
Uses (modullar ro‘yxati);
Label(nishonlar ro‘yxati);
Const (o‘zgarmas miqdorlar);
var
  (o‘zgaruvchi miqdorilar)
 protseduralar va funksiyalarni e’lon qilish;
begin
  (asosiy qismi)
end.
```

O‘zgarmas va o‘zgaruvchi miqdorlar

Turbo Pascal dasturlash tilida o‘zgarmas kattaliklar sifatida butun, haqiqiy va o‘n oltilik sanoq sistemasi sonlari, mantiqiy doimiyliklar, simvollar, satriy kattaliklar, to‘plam konstruktorlari va noma’lum ko‘rsatgich belgisidan foydalanish yo‘lga qo‘yilgan.

Butun sonlar sifatida -2 147 483 648 sonidan +2 147 483 647 gacha bo‘lgan butun sonlardan foydalaniladi va agar hisoblashlar jaryonida bu chegaralarlan «chiqish»ga yo‘l qo‘yilsa, dastur kompilyatori xatolik haqida xabar beradi.

[42] Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. Pp 30-31

Type	Min value	Max Value	Size in Bytes
Byte	0	255	1
ShortInt	-128	127	1
SmallInt	-32768	32767	2
Word	0	65535	2
Integer	-2147483648	2147483647	4
LongInt	-2147483648	2147483647	4
Cardinal	0	4294967295	4
Int64	-9223372036854780000	9223372036854775807	8

Table 3.2: Predefined integer types

Type	Range	Size in bytes
Byte	0 .. 255	1
Shortint	-128 .. 127	1
Smallint	-32768 .. 32767	2
Word	0 .. 65535	2
Integer	either smallint or longint	size 2 or 4
Cardinal	longword	4
Longint	-2147483648 .. 2147483647	4
Longword	0 .. 4294967295	4
Int64	-9223372036854775808 .. 9223372036854775807	8
QWord	0 .. 18446744073709551615	8

Butun tiplar jadvalini keltiramiz: [43] [44]

Tip nomi	**Uzun-ligi (bayt-larda)**	**Qiymatlar diapazoni**	**Tipning quvvati**
Byte	1	0 dan 255 gacha	256
ShortInt	1	-128 dan +127 gacha	256
Word	2	0 dan 65535 gacha	65536
integer	2	-32768 dan +32767 gacha	63536
LongInt	4	-2 147 483 648 dan +2 147 483 647 gacha	4294497296

[43] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp-25.
[44] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 18.

Butun tiplardan tashqari haqiqiy tiplardan foydalanish tavsiya etiladi.

Haqiqiy sonlar juda ko‘p hollarda qo‘zg‘aluvchi nuqtali-eksponensial ko‘rinishda ifodalanadilar. Juda ham kichik yoki juda ham katta sonlarni bunday ko‘rinishda tasvirlash qulay hisoblanadi.

Masalan: 0,0000035000 sonini 3,50E-06 ko‘rinishda ifodalash foydalanuvchi va dastur uchun qulay hisoblanadi.

Haqiqiy tiplar jadvalini keltiramiz: [45]

Tip nomi	**Uzun-ligi (bayt-larda)**	**Qiymatlar diapazoni**	**Razryadlari miqdori**
Real	6	2.e-39…1.7e38	11-12
Single	4	1.5e-45…3.4e38	7-8
Double	8	5.0e-324…1.7e308	15-16
Extended	10	3.4e-4932…1.1e4932	19-20
Comp	8	--9.2e18…9.2e18	19-20

Table 3.4: Supported Real types

Type	Range	Significant digits	Size
Real	platform dependant	???	4 or 8
Single	1.5E-45 .. 3.4E38	7-8	4
Double	5.0E-324 .. 1.7E308	15-16	8
Extended	1.9E-4932 .. 1.1E4932	19-20	10
Comp	-2E64+1 .. 2E63-1	19-20	8
Currency	-922337203685477.5808 .. 922337203685477.5807	19-20	8

O‘n oltilik sanoq sistemasi sonlari bu sanoq sistemasining sonlari yordamida ifodalanadilar va faqat sonlarning oldiga dollar belgisi qo‘yib yoziladi. Bu o‘zgarmaslar $00000000 … $FFFFFFFF diapazonda amal qiladilar.

Mantiqiy doimiyliklar. Pascal tilida mantiqiy tip boolean standart nomi bilan aniqlanadi. Mantiqiy tipli o‘zgaruvchilar faqat ikki xil qiymat: True(rost) va False (yolg‘on) larnigina qabul qilishi

[45] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp-28.

mumkin. Mantiqiy tipli qiymatlar ham tartiblangan, ya'ni False<True.

Pascal tilida asosan quyidagi uchta mantiqiy amaldan ko‘proq foydalaniladi: not - rad etmoq, and - mantiqiy ko‘paytirish, or - mantiqiy qo‘shish.

Bu amallarni faqat mantiqiy o‘zgarmaslar ustidagina ishlatish mumkin va natijada yana mantiqiy o‘zgarmas hosil bo‘ladi. quyida mantiqiy o‘zgarmaslar ustida amallar jadvali ko‘rsatilgan: [46]

Mantiqiy ko‘paytirish	**Mantiqiy qo‘shish**	**Mantiqiy rad etmoq**
True and true = true	true or true = true	not true = false
True and false= false	true or false= true	not false= true
False and true = false	False or true = true	
False and false = false	False or false = false	

Table 12.5: Boolean operators

Operator	Operation
not	logical negation (unary)
and	logical and
or	logical or
xor	logical xor

Ixtiyoriy, qiymatlarni solishtirish amali ham mantiqiy qiymatni beradi:

Misol: 3>2 natijasi true

0<-1 natijasi false.

Simvolli doimiyliklarga ‘(apostrof) belgisi ichiga olingan klaviaturaning ixtiyoriy simvolini misol keltirish mumkin(‘a’ yoki ‘f’- bunga misol bo‘la oladi). Agar apostrof simvolini o‘zidan foydalanmoqchi bo‘lsak uni dastur tarkibida quyidagcha yozish shart: ’’’’

Shu holatda birgina apostrof belgisini chop etishga erishish mumkin.

Simvolni uning kodi yordamida ham yozish mumkin va bu holda simvol kodi oldidan # belgisi yoziladi.

Misol: #97-a simvoli;

#90-Z simvoli;

[46] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp-159.

O‘zgarmaslar (const) - bu programmani ishlashi davomida o‘zgarmay qoladigan miqdordir. Agar miqdor programmada ko‘p marta ishlatilsa, uni programma matnida qayta - qayta yozgandan ko‘ra, bu miqdorni o‘zgarmas deb aniqlab olib, programmadagi miqdorni o‘rniga o‘zgarmasni ismini yozish qulay bo‘ladi. Masalan, hammaga ma’lum p (p=3,1415926535…) soni. Bu sonni bir necha marta takroran programmada yozish noqulay, shuning uchun, uni o‘zgarmas sifatida aniqlab olish maqsadga muvofiqdir.

O‘zgarmas ***const*** xizmatchi so‘zidan keyin e’lon qilinadi (aniqlanadi):

<o‘zgarmasni aniqlash>::=<o‘zgarmas ismi>=<o‘zgarmas>;

bu yerda <o‘zgarmas>::=<skalyar miqdor>|<belgili qator>|<o‘zgarmas ismi>|+<o‘zgarmas ismi>|-<o‘zgarmas ismi>;

<o‘zgarmaslar bo‘limi>::=<bo‘sh>| ***const*** <o‘zgarmasni aniqlash>;

Faqatgina quyidagi tipdagi o‘zgarmaslar e’lon qilinishi mumkin:

- Tartib tipi
- Berilgan tip
- Nuqtali tip (faqat ruxsat beriladigan yagona qiymat nolga teng bo‘ladi)
- Haqiqiy tip
- Char tipi
- String tipi

Quyida barcha o‘zgarmaslar deklaratsiyasi keltirilgan:[47]

```
Const
  e = 2.7182818;  { Real type constant. }
  a = 2;          { Ordinal (Integer) type constant. }
  c = '4';        { Character type constant. }
  s = 'This is a constant string'; {String type constant.}
  sc = chr(32)
  ls = SizeOf(Longint);
  P = Nil;
  Ss = [1,2];
```

[47] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 20-21, 30.

```
const
identifier = constant_value;
```

The following table provides examples of some valid constant declarations:

Constant Type	Examples
Ordinal(Integer)type constant	valid_age = 21;
Set type constant	Vowels = set of (A,E,I,O,U);
Pointer type constant	P = NIL;
Real type constant	e=2.7182818; velocity_light = 3.0E+10;
Character type constant	Operator = '+';

Misol: *Program L1;*

```
const Pi=3.1415926535;
var A,B: real;
Begin
   A:=Pi+Sin(Pi*(Pi-1));
   B:=sqrt(a);
end.
```

1-misol:[48]

The following example illustrates the concept:

```
program const_circle (input,output);
const
PI = 3.141592654;
var
r, d, c : real;   {variable declaration: radius, dia, circumference}
begin
  writeln('Enter the radius of the circle');
  readln(r);
  d := 2 * r;
  c :=  PI * d;
  writeln('The circumference of the circle is ',c:7:2);
end.
```

When the above code is compiled and executed, it produces the following result:

```
Enter the radius of the circle
23
The circumference of the circle is 144.51
```

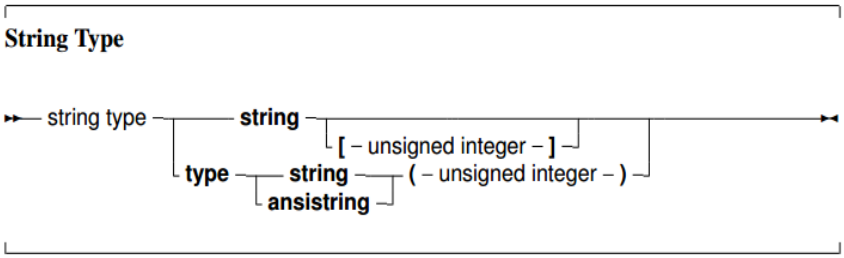
Quyida o‘zgarmaslarni e’lon qilishga misol keltiramiz:

```
const
Thousand = 1000;
Pi = 3.14;
AuthorName = 'Marco Cantu';
```
[49]

Satriy doimiyliklar – belilarning ixtiyoriy ketma-ketligi bo‘lib ular apostroflar ichida beriladi. Lekin shuni aytish joizki CR kodli

[48] Pascal tutorial. Tutorilaspoint.cm. Pp 29-30.
[49] Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. Pp 31-35.

belgi bu tarkibda qatnasha olmaydi va u maxsus vazifani bajaradi, satriy doimiylik bo‘sh ham bo‘lishi mumkin.



Masalan:
‘satriy doimiyliklar’;
‘G‘afur G‘ulom bekati’;
‘ ‘;

Turbo Pascal dasturlash tilida ayrim kattaliklarning qiymatlari o‘zgaruvchilar yordamida saqlanishlari mumkin. O‘zgaruvchi –bu amalda axborotni saqlash uchun mo‘ljallangan xotiraning bir qismi hisoblanadi. O‘zgaruvchilardan foydalanishda uning nomi va tipini ko‘rsatish talab etiladi. Dasturda qatnashuvchi o‘zgaruvchilarni tavsiflash o‘zgaruvchilar bo‘limi-var da amalga oshiriladi. Masalan, ikki butun sonning yig‘indisini hisoblovchi dastur uchun o‘zgaruvchilar bo‘limini quyidagicha yozish mumkin:

var a,b:integer;{a va b-butun sonlarni ifodalovchi nom-kattaliklar}

s:string; {S-satriy kattalik}

S a t r i y t i p	
Tip nomi	Qiymatlar diapazoni
Char	ANSI kodlar jadvalidagi 0 dan 255 gacha bo‘lgan kodlarga mos keluvchi belgilardan biri
String	Uzunligi 0 dan 255 gacha bo‘lgan belgilar ketma – ketligi
String[n]	Uzunligi 0 dan n (n<255) gacha bo‘lgan belgilar ketma – ketligi

Quyida o‘zgaruvchilar va uzgarmaslarga nom berishdagi qoidalar bilan tanishamiz:

1. Nom berishda faqat ‘A’.’Z’ va ‘a’…’z’ oraliqdagi ingliz tili alfaviti belgilaridan, 0 dan 9 gacha bo‘lgan raqamlardan hamda (ostki chiziq) belgisidan foydalanish mumkin.

2. Nom faqat ingliz tili alfaviti belgilaridan hamda va (ostki chiziq) belgisidan boshlanishi mumkin.

3. Nom bir necha so‘zlardan tashkil topgan bo‘lsa ular ‘_’ belgisi bilan tutashtiriladi.

4. Nom berishda foydalangan katta yoki kichik simvollar «farqlanmaydi».

5. Dasturda har bir nom faqat bir marotaba o‘zgaruvchilar bo‘limida qatnashish huquqiga ega.

6. Nom sifatida xizmatchi so‘zlardan foydalanish mumkin emas.

Masalan:

Tomonlari a,b qiymatlargi ko‘ra to‘g‘ri to‘rtburchak perimetri, diagonali va yuzasini aniqlovchi dasturda qatnashuvchi o‘zgaruvchilarni aniqlang.

```
var   a,b,p:integer;
          D,uza:REAL;
```

Standart funksiyalar va algebraik ifodalar[50]

Kattaliklarning tiplari ular foydalanadigan qiymat va ular bilan bajaraladigan amallar bilan aniqlanadilar. Butun va haqiqiy tiplarda kattaliklar uchun quyidagi amallar qo‘llaniladi:

mod	Modulus (the remainder of integer division)
as	Allows a type-checked conversion at runtime
and	Boolean or bitwise and
shl	Bitwise left shift
shr	Bitwise right shift
Additive Operators	
+	Arithmetic addition, set union, string concatenation, pointer offset addition
-	Arithmetic subtraction, set difference, pointer offset subtraction
or	Boolean or bitwise or
xor	Boolean or bitwise exclusive or

[50]Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. Pp 26.

Relational and Comparison Operators (Lowest Precedence)

=	Test whether equal
<>	Test whether not equal
<	Test whether less than
>	Test whether greater than
<=	Test whether less than or equal to, or a subset of a set
>=	Test whether greater than or equal to, or a superset of a set
in	Test whether the item is a member of the set
is	Test whether object is compatible with a specified type definition (which is only of use in object oriented programming.)

Amallar	Amalning vazifasi	Butun va haqiqiy tiplarda
+	qo‘shish	Qo‘shish amali bajariladi, amalning natijasi qo‘shiluvchilar tipiga mos
-	ayirish	Ayirish amali bajariladi, amalning natijasi qo‘shiluvchilar tipiga mos
*	ko‘paytirish	Ko‘paytirish amali bajariladi, amalning natijasi qo‘shiluvchilar tipiga mos
/	bo‘lish	Bo‘lish amali bajariladi, amalning natijasi faqat haqiqiy tip

Bundan tashqari faqat butun tiplar bilan bajariluvchi quyidagi amallar mavjud:

div	**Butunga bo‘lish**	**amalning natijasi faqat butun tip**
mod	Bo‘lish amalidagi qoldiq	

Div amalida bo‘lish paytida qoldiq tashlab yuboriladi;

Mod amalida bo‘lish paytidagi qoldiqni aniqlash;

Quyidagi jadvada Turbo Pascal dasturlash tilida foydalanadigan standart (matematik) funksiyalar keltirilgan.

Funksiyalar	Funksiya qiymati	Izohlar
Abs(n)	n-kattalikning absolyut qiymati	Abs(n) va Sqr(n) funksiya qiymatlari argument tipii bilan mos keladi
Sqr(n)	n-kattalikning kvadrati	

Sqrt(n)	n-kattalikdan kvadrat ildiz hisob-lash	Argumenti ixtiyoriy sonli tip, natijasi esa haqiqiy tip
Sin(n)	n-kattalikning sinusi (n-radianda)	
Cos(n)	n-kattalikning kosinusi (n-radianda)	
Arctan(n)	n-kattalikning arktangensi (n-radianda)	
exp(n)	e^n-qiymati	(e=2,7182818285…)
Ln(n)	Ln (n) qiymati	
Random(n)	0 dan n-1 gacha bo‘lgan tasodifiy butun son	
Frac(x)	X-soninng kasr qismi	X va natijaning tipi haqiqiy
Int(x)	**X-soninng butun qismi**	

Ifodalar

Ifodalar asosan operator va operandlardan tashkil topadi. Operand sifatida o‘zgaruvchi, doimiylik, funksiya yoki boshqa biror ifodadan foydalanish mumkin. Asosiy operatorlarni keltirib o‘tamiz. Bular +,_-,*, /, div va mod .

div operatori bir soni ikkinchisiga bo‘lishda natijaning butun qismini beradi

Masalan, 23 div 5 ifodaning qiymati 4 ga teng

mod operatori bir soni ikkinchisiga bo‘lishda qoldiq qiymatini beradi.

Masalan, 23 mod 5 ifodaning qiymati 3 ga teng

*, /, div va mod operatorlari + va – larga ko‘ra, *, /, div va mod operandlari yuqori darajali hisoblanishadi va ular ifoda qiymatini hisoblashda birinchilardan bo‘lib bajariladi.

O‘zlashtirish va ma’lumotlarni ekranga chiqarish operatorlari

Dasturning bajariluvchi qismi odatda bir yoki bir necha *operatorlardan* iborat bo‘ladi. Operator dastur bajarilishi lozim bo‘lgan biror amalni ifodalaydi. Shunday operatorlardan biri juda ko‘p

foydalanadigan o‘zlashtirish operatori hisoblanadi. Agar dasturda hisoblashlar bajarilsa u holda albatta o‘zlashtirish (qiymat berish) foydalaniladi. Umumiy holda o‘zlashtirish quyidagi ko‘rinishga ega:

o‘zgaruvchi *:= ifoda;*

Bu yerda:

- o‘zgaruvchi dastur bajarilishi natijasida unga o‘zlashtirgan qiymatni ifodalaydi.

*:=*qiymat berish belgisi

ifoda- bu ifodaning qiymati nomi ko‘rsatilgan kattalikga (o‘zgaruvchiga) o‘zlashtiriladi.

ifoda va o‘zgaruvchi tiplari mos kelishi shart:

Misollar:

*D:=5*g-12.5;*

Ff:=f1+(a-t);

*Perimetr:=2*a+2*b;*

Z:= False;

Dfg:=’diametr’;

O‘zlashtirish operatoridan foydalanishga qator misollar keltiramiz:

Tomonlari 5 sm, 7 sm va 8 sm teng bo‘lgan uchburchak perimetrini hisoblash uchun quyidagi o‘zlashtirish operatorlaridan dastur tarkibida foydalanish mumkin.

A:=5; B:=7; C:=8; P:=A+B+C;

Katetlari 3,5 sm,7,8 sm bo‘lgan to‘g‘ri burchakli uchburchak perimetrini hisoblash uchun quyidagi o‘zlashtirish operatorlaridan dastur tarkibida foydalanish mumkin.

A:= 3,5 ; B:= 7,8 ; C:=sqrt(sqr(a)+sqr(b)); P:=A+B+C;

$y=ax^2+bx+c$ funksiya qiymatini a=2, b=4, c=5, x=45 larda hisoblash uchun quyidagi o‘zlashtirish operatorlaridan foydalanishimiz mumkin.

a:=2; b:=4; c:=5; x:=45;

y:=a*sqr(x)+b*x+c;

Ma’lumotlarni ekranga chiqarish operatorlari

Ma’lumotlarni ekranga chiqarish Write va Writeln protseduralari yordamida amalga oshiriladi. Write so‘zidan keyin qavslarda

apostrof belgisi bilan chegaralangan matnli ma’lumotlar va o‘zgaruvchilar ro‘yxatini vergul orqali ajratib berish mumkin. Misollar:

Write(‘Tenglama ildizlari’);
Write(Ko‘paytma);
Write(rr);
Write(‘to‘rtburchak tomonlari. A=’, A,’B=’,B);

Writeln instruksiyasining Write instruksiyasidan farqi shundaki har safar bu instruksiya bajarilgach, kursor keyingi satr boshiga o‘tkaziladi. Bu holatlarni quyidagi dasturlar yordamida kuzatishimiz mumkin.

Dasturlardan namunalar	**Natija:**
var a,b,c:integer; **begin** **a:=4; b:=6;write ('a+b=');write(a+b);readln;** **end**	a+b=10
var a,b,c:integer; **begin** **a:=4;b:=6;c:=a+b;write(a,b,c);readln;** **end.**	4610
var a,b,c:integer; **begin** **a:=4;b:=6;c:=a+b;write(a:5,b:5,c:5);readln;** **end.**	4 6 10
var a,b,c:integer; **begin** **a:=4;b:=6;c:=a+b;** **writeln (a);writeln(b);writeln(c);readln;** **end.**	4 6 10

Ekran bilan ishlash operatorlari

Ekran bilan ishlash uchun Turbo Pascal tilida turli protseduralar, funksiyalar va operatorlarni qamrab olgan SRT modulidan foydalaniladi. SRT moduli sodda ammo samarali imkoniyatlarga ega. Kompyuter tarkibida foydalanadigan axborotni vizual akslantiruvchi vosita-

displey ikkita asosiy qismdan iborat: monitor va boshqarish bloki (adapter yoki displey adapteri). Bu ikki qurilma bir-biri bilan hamkorlikda (ayrim hollarni hisobga olmaganda, masalan rangli monitor va monoxrom adapter bilan ishlay olishi) “faoliyat” ko‘rsatishadi.

Birinchi adapter 1981 yili monoxrom adapter (MDA) nomi bilan IBM PC kompyuterlarida qo‘llanilgan bo‘lib, matnli ma’lumotlarni faqat 20 qatorda 40 tadan belgi yoki 25 qatorda 80 tadan baeli bilan chiqaradigan formatlarga ega edi. Bunda yorqin belgilar qora fonda akslanar edi va shriftlar kengligi har ikki holda ham bir xil bo‘lib, ular “oddiy” (обычной), hamda “tagiga chizilgan” (podcherknuto‘y) ko‘rinishga ega bo‘lishlari mumkin edi.

1982 yili Hercules firmasi HGC adapteri yaratdi va u MDA adapteri xossalarini takrorlash bilan birga 700x350(piksellarda) hajmdagi grafik tasvirlarni ham ekranga chiqara olar edi. Aynan shu vaqtda IBM kompaniyasi CGA(Color Graphics Adapter-rangli grafik adapter) nomli adapterni yaratdi. Bu adapter matnli(40x25 yoki 80x25) va grafik ma’lumotlarni (320x200) ekranga chiqarish imkoniyatiga ega edi. MDA adapteri kabi simvollarni “tagiga chizilgan” rejimda chiqara olmasada “negativ” (qora rangli simvollar yorqin fonda) tasvirda chiqarish imkoniyatiga ega bo‘lgan. Rangli rejimda simvollar uchun 16 rangdan fon uchun esa 8 rang qo‘llanilgan. CGA adapterining matnli imkoniyatlari IBM kompaniyasining keyinchalik yaratgan EGA, MGCA, VGA va SVGA adapterlarida ham saqlangan. SRT moduli imkoniyatlari bu adapterlarga qo‘llanilishi nuqtayi nazaridan qarab chiqiladi.

TextMode prosedurasi. Bu protsedura adapterning matn bilan ishlash rejmilaridan birini aniqlash imkoniyatini beradi. Uning sarlavhasi quyidagicha:

procedure TextMode (mode:word);

bu yerda mode-matnli rejim kodi.

SRT modulida mode uchun quyidagi doimiyliklardan foydalaniladi.

```
const
BW0 =0;        {40x25 hajmdagi oq-qora rejim}
Co40    =1;        {40x25 hajmdagi rangli rejim }
BW80    =2;        {40x25 hajmdagi oq-qora rejim }
CO80    =3;        {80x25 hajmdagi rangli rejim }
```

Mono =7; {MDA dan foydalaniladi}

Font8x8 =256; { EGA yoki VGA adapterlarida 80x43 yoki 80x50 rejimlariga xos, yuklanuvchi shrift uchun tanlanadi }

TextMode protsedurasi yordamida o‘rnatilgan rejim kodi modulning LastMode global o‘zgaruvchisida saqlanish orqali dastlabki holatga qaytarilishi mumkin.

TextColor prosedurasi Kiritiladigan matn belgilari rangini aniqlaydi. Undan foydalanish tartibi quyidagicha: TextColor(Color: Byte);

Misol, textcolor(4) yoki textcolor(red) operatorlari matnning rangi qizil bo‘lishini ta’minlaydi.

TextBackground protsedurasi. Fon rangigi belgilash uchun ishlatiladi. Undan foydalanish tartibi quyidagicha :TextBackground (Color: Byte);

Misol, TextBackground (1) yoki textcolor(blue) operatorlari fon rangini to‘q ko‘k bo‘lishini ta’minlaydi.

TextColor va TextBackground protsedurasi parametrlarining tipi Byte, tipi bilan aniqlanadilar va ular tegishli rangni ta’min-laydilar.

Rang	*Kod*	
Black	*0*	*Qora (Чёрный)*
Blue	*1*	*To‘q ko‘k (Тёмно-синий)*
Green	*2*	*To‘q yashil (Тёмно-зелёный)*
Cyan	*3*	*Moviy (Бирюзовый)*
Red	*4*	*Qizil (Красный)*
Magenta	*5*	*Binafsha rang (Фиалетовый)*
Brown	*6*	*Jigarrang rang (Коричневый)*
LightGray	*7*	*Och ko‘k (Светло-серый)*
DarkGray	*8*	*To‘q kul rang (Тёмно-серый)*
LightBlue	*9*	*Ko‘k (Синий)*
LightGreen	*10*	*Och yashil (Светло-зелёный)*
LightCyan	*11*	*Och moviy (Светло-бирюзовый)*
LightRed	*12*	*Och qizil(atirgul) (Розовый)*
LightMagenta	*13*	*Malina (och malina rang) (Малиновый)*
Yellow	*14*	*Sariq (Желтый)*
White	*15*	*Oq (Белый)*
Blink	*128*	*belgi yonib-o‘chishi*

ClrScr protsedurasi. Bu protsedura ekran va Window protsedurasi yordamida yaratilgan oynani tozalaydi, berilgan fon rangi bilan to‘ldirib, kursorni ekranning yuqori chap nuqtasiga joylashtiradi. Undan foydalanish tartibi quyidagicha: clrscr;

Window prosedurasi. Bu protsedura matn uchun mo‘ljallangan oyna («kichik ekran» ni yaratish uchun mo‘ljallangan. Bu protse-duradan foydalanganda kursor oynaning yuqori chap nuqtasiga joylashadi va oyna fon rangi bilan to‘ldiriladi. Matn chop etilishida matn shu oynaning o‘ng chegarasiga etgach, yangi qatorga tushishi va oyna to‘lgach matnni «tepaga qarab siljitish» ta’minlanadi. Undan foydalanish tartibi quyidagicha:

Window(XI,Y1,X2,Y2: Byte); Bu yerda (XI,Y1)-oynaning yuqori chap, (X2,Y2) –o‘ng past nuqtalari koordinatalari.

GotoXY protsedurasi. Bu protsedura kursorni ekran yoki oynada yangi koordinatali nuqtaga o‘tkazish uchun ishlatiladi. Undan foydalanish tartibi quyidagicha: GotoXY(X,Y: Byte); Misol, GotoXY(4,8); (X,Y-qiymatlari ekran yoki oyna chegarasidan chiqadigan bo‘lsa, buyruq bekor qilinadi)

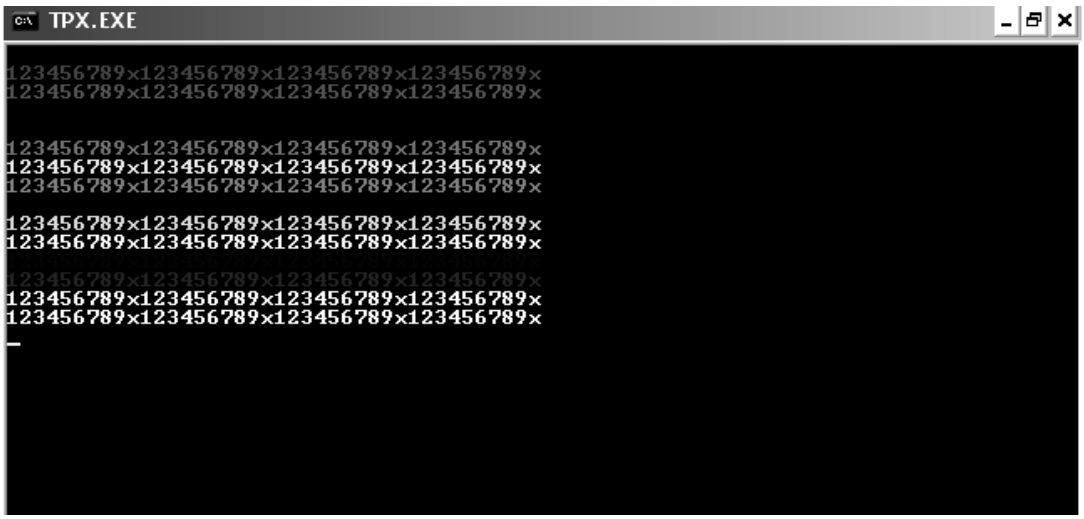
WhereX va WhereY funksiyalari. Bu funksiyalar yordamida kursorning joriy koordinatalari aniqlanadi: WhereX –uning gorizontal, WhereY - vertikal koordinatalari qiymatini aniqlashda qo‘llaniladi.

Misol: writeln(whereX,WhereY);

1-masala. LowVideo, NormVideo, HighVideo protseduralari yordamida matn yorqinligini turli holatlarda ta’minlovchi dastur yarating.

```
Uses CRT;
begin
clrscr;
LowVideo;WriteLn('Past yorqinlik');
NormVideo;WriteLn('nirmal yorqinlik');
HighVideo;WriteLn('yuqori yorqinlik');
readln;
end.
  uses crt;
var
lm,t:word;
```

```
const
col80=3;
begin
 lm:=lastmode;  textmode(col80);
 for t:=1 to 15 do
 begin
 textcolor(t);
 writeln('123456789x123456789x123456789x123456789x');
 end;
 {print('rejim 40*25')' }
 end.
```



Mustahkamlash uchun savollar

1. Dasturlashning integrallashgan muhiti nima?
2. Til alfaviti nima?
3. Asosiy belgilarni sanab bering.
4. Til operatori qanday tushuncha?
5. Asosiy va tarkibiy operatorlarni ajratib sanab bering.
6. Operatorlar qanday tartibda bajariladi?
7. Programma operatorlariga ismlar qanday qo‘yiladi va nima uchun qo‘yiladi?
8. Ism(identifikator) ning BNF formulasini yozing.
9. E’lonlarning vazifasini tushuntiring.
10. Miqdorlar qanday turdagi qiymatlarni qabul qilishi mumkin?

11. Simvolli doimiyliklar deganda nimani tushunasiz? Misollar keltiring.

12. Satriy doimiyliklar nima, unga misollar keltiring.

13. Mantiqiy tipni nomi va qanday qiymatlar qabul qiladi?

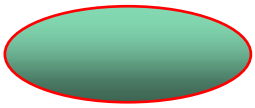
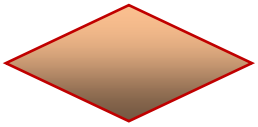
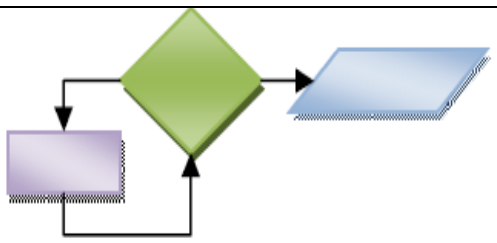
14. O‘zgaruvchilarni o‘zgarmaslardan farqi nimada?

15. Dastur tuzish jarayonida o‘zgaruvchi va o‘zgarmaslar qaysi xizmatchi so‘zlar bilan tavsiflanadi?

12-§. Chiziqli, tarmoqlanuvchi va takrorlanuvchi dasturlar

Chiziqli dasturlar tuzilish jihatidan aniq ketma-ketlikdagi amallardan iborat. Bu amallarning bajarilishi dasturda keltirilish tartibi bilan bog‘liq, ya’ni yozilgan ketma-ketlikda bajariladi. Har bir masalani yechishdan oldin algoritmi tuzib olinadi. Algoritim tuzishda quyidagi geometrik figuralardan foydalaniladi.

Blok – sxema jadvali

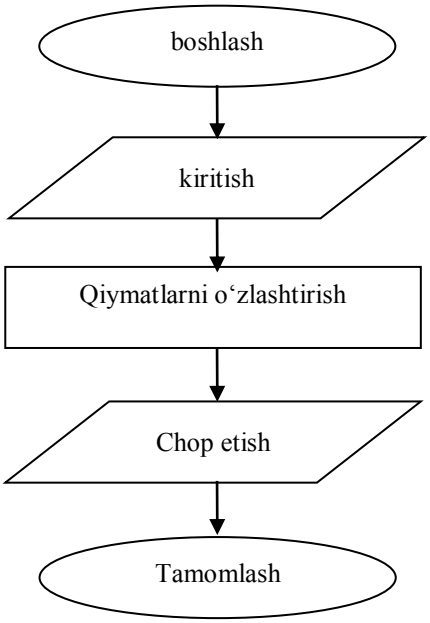
Shakl nomi	Geometrik figura	Vazifasi
Ishga tushirish, to‘xtatish		Algoritmning boshlash va tugatish;
Jarayon		Arifmetik ifodalarni hisoblash;
Kiritish		Boshlang‘ich ma’lumotlarni kiritish;
Yechim		Boshqarishni shart asosida o‘zgarishi;
Takrorlanish jarayoni		Takrorlanish jarayoni tasvirlash;

Avvaldan ma’lum jarayon		Qism dasturlarga murojaat qilish;
Chiqarish		Natijalarni tashqi qurilmalarga chiqarish;
Yurish		Amallarni bajarish yo‘nalishi;

Chiziqli algoritmlarni blok-sxemasi

Hech qanday shart tekshirilmaydigan va tartib bilan faqat ketma – ket bajariladigan algoritmlar

ch i z i q l i a l g o r i t m l a r deb yuritiladi .



Chiziqli algoritmlar va dasturlar odatda juda sodda masalalarni yechishda qo‘llaniladi. Bu masalalar yechimi biror shartga yoki siklik amallar bajarilishiga bog‘liq emas.

Quyida 1-misolni ko‘rib chiqamiz:

Project/New Project/Program

Source Editor oynasiga ushbu kodni kiritamiz:

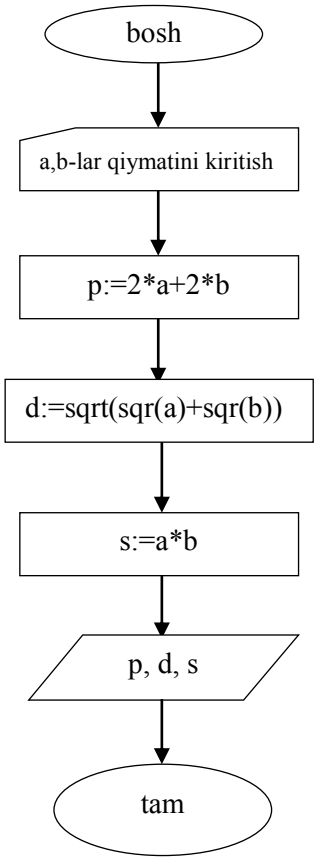
```
programProject1;
{$mode objfpc}{$H+}
uses
{$IFDEF UNIX}{$IFDEF UseCThreads}
cthreads,
```

{$ENDIF}{$ENDIF}
Classes
{ shungan keyin unit lardan foydalanishingiz mumkin };
{$IFDEF WINDOWS}{$R project1.rc}{$ENDIF}
begin
end.

Ushbu misolni File/Save buyruqlari orqali saqlashimiz mumkin. So‘ngra begin va end satrlari orasida quyidagi kodni yozishimiz mumkin:

Writeln('This is Free Pascal and Lazarus');
Writeln('Press enter key to close');
Readln;

Misol kodining umumiy ko‘rinishi:



programfirst;
{$mode objfpc}{$H+}
uses
{$IFDEF UNIX}{$IFDEF UseCThreads}
cthreads,
{$ENDIF}{$ENDIF}
Classes
{ you can add units after this };
{$IFDEF WINDOWS}{$R first.rc}{$ENDIF}
begin
Writeln('This is Free Pascal and Lazarus');
Writeln('Press enter key to close');
Readln;
end.

2-misol. To‘g‘ri to‘rtburchakning tomonlariga ko‘ra uning perimetri, diagonali va yuzasini hisoblashni (a, b – tomonlar qiymatiga ko‘ra) quyidagicha tashkillashtirish mumkin.

Yechish:

Program to‘rtburchak yuzi;
Var a, b: Integer;
P, d, s: real;
Begin

```
    Write('a,b tomonlarni qiymatlari kiritilsin');
    ReadLn(a,b);
    P:=2*a+2*b;
    D:=sqrt(sqr(a)+sqr(b));
    S:=a*b;
    WriteLn('to'rtburchak perimetri=',p);
    WriteLn('to'rtburchak diagonali=',d);
    WriteLn('to'rtburchak yuzasi=',S);
End.
```

Dastur tuzishda qiymatlardan foydalanishga misol ko'rib chiqaylik:

```
program FirstVar;
{$mode objfpc}{$H+}
uses
{$IFDEF UNIX}{$IFDEF UseCThreads}
cthreads,
{$ENDIF}{$ENDIF}
Classes
{ shundan keyin unit lardan foydalanish mumkin };
var
x: Integer;
begin
x:= 5;
Writeln(x* 2);
Writeln('Press enter key to close');
Readln;
end.
```

[51] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 10-14.

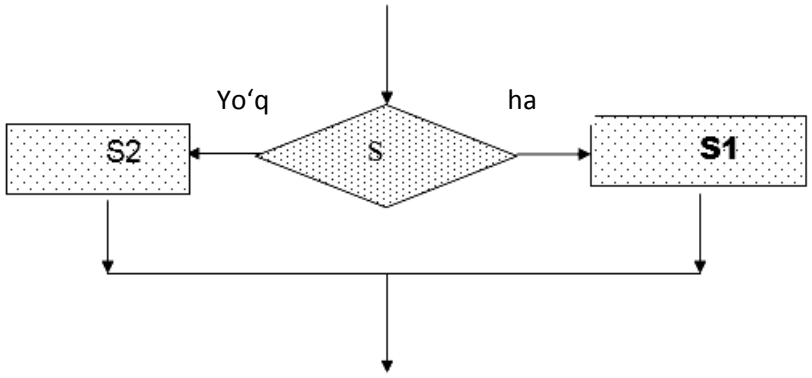
Tarmoqlanish va o‘tish operatorlari

Turli masalalarni yechganda ko‘rsatmalarni bajarish tartibi biror bir shartning bajarilishiga bog‘liq holda bajariladi. Ya’ni algoritm tarmoqlanadi. Tarmoqlanish «yechim» bloki orqali ifodalanadi.

Ma’lum bir shartni bajarilishi yoki bajarilmasligiga qarab, tarmoqlanuvchi jarayon holatlari aniqlanadi. Tarmoqlanuvchi jarayonlarni hisoblash uchun shartli operatordan foydalaniladi. Shartli operator ikki xil ko‘rinishda bo‘ladi:

- to‘liq shartli operator;
- chala shartli operator.

To‘la shartli operatorning algoritmik sxemasini quyidagi ko‘rinishga ega:



To‘liq shartli operator quyidagi formada yoziladi:

if <mantiqiy ifoda> *then* <operator> *else* <operator>

bu yerda *if* (agar), *then* (u holda), *else* (aks holda) xizmatchi so‘zlar.

Shunday qilib, to‘liq shartli operatorni quyidagicha yozish mumkin:

if S then S1 else S2;

bu yerda S - mantiqiy ifoda;

S1 – S mantiqiy ifoda rost qiymat qabul qilganda bajariluvchi operator;

S2 -S mantiqiy ifoda yolg‘on qiymat qabul qilganda bajariluvchi operator.

Shartli operatorning bajarilishi unda yozilgan S_1 yoki S_2 operatorlaridan birini bajarilishiga olib keladi, ya’ni agar S mantiqiy

ifoda bajarilishidan so‘ng *true* (rost) qiymati hosil bo‘lsa S_1 operatori, aks holda esa S_2 operatori bajariladi.

To‘liq shartli operatorga doir misollar:

if a=2 *then* d: = x+2 *else* d: = x-2;

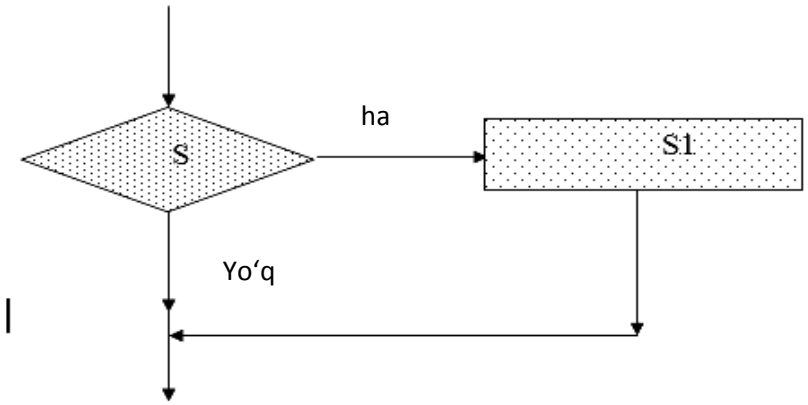
if (x<y) and (z>5) *then begin* y: = x * sin(x);

t: =x * cos(x) *end else begin* y: = 0; t: =1 *end*;

if x<0 or x =3 *then* y: = x*x+1 *else* if x<2

then y: = sqr(abs(x-1)) *else* y: = x*x;

Qisqa (to‘liqmas) shartli operatorning algoritmik sxemasini quyidagi ko‘rinishga ega:



Chala (to‘liqmas) shartli operatorning yozilishini quyidagicha ifodalanadi:

if S then S_1;

bu yerda S - mantiqiy ifoda, S_1 - operator.

Agar S ifoda qiymati *true* (rost) bo‘lsa S_1 operatori bajariladi, aks holda esa boshqarish shartli operatordan keyin yozilgan operatorga uzatiladi.

Shartli operatordan foydalanishga misollar keltiramiz.

3-misol. Kiritilgan ixtiyoriy butun sonni juft yoki toqligini aniqlovchi dastur yarating.

```
program r1;
var x:integer;   javob:string;
  begin
  readln(x);
  if x mod 2 =0 then javob:='juft' else javob:='to‘q' ;
  writeln('kiritilgan son ',javob);
```

```
readln;
end.
```

4-misol. Kiritiladigan ixtiyoriy a,b,d sonlar uchun a+b>d, a+d>b, b+d>a tengsizliklarning barchasi bajarilganda «shartlar qanoatlantirilgan» deb javob beruvchi dastur yarating.

```
program r2;
uses crt;
var a,b,c,d:integer;
 javob:string;
 begin  clrscr;
 readln(a,b,c);
 if (a+b>c) and (a+c>b) and (b+c>a) then
  writeln('shartlar qanoatlantirilgan');
   readln;
 end.
```

Yuqorida keltirilgan ikki misoda masala shartiga ko‘ra shartli operatorning to‘liq va to‘liq bo‘lmagan holatlaridan foydalanildi.

Tarmoqlanish operatoridan foydalanishda quyidagi qoidalarga amal qilish shart:

IF operatoridan foydalanganda ELSE dan oldin «;» (nuqta-vergul) qo‘yilmaydi.

Shartli operator Then va ELSE xizmatchi so‘zlaridan keyin bir necha operator (amal yoki buyruq) ishlatilishi zarurati bo‘lsa, u holda bu buyruqlar begin va end qavslari ichiga joylashtirishi shart.

```
var
Temp: Single;
begin
Write('Please enter Temperature of this room :');
Readln(Temp);
If Temp > 22 then
Writeln('Please turn on air-condition')
else
Writeln('Please turn off air-condition');
Write('Press enter key to close');
```

Readln;
end.[52]

(Shartsiz o‘tish operatorini o‘rganishda $ax^2+bx+c=0$ tenglamaning yechimlarini aniqlovchi dastur keltirilgan, shu holatga e’tibor bering).

O‘tish operatori (Shartsiz o‘tish operatori).

Shartsiz o‘tish operatori goto quyidagicha yoziladi:

goto belgi;

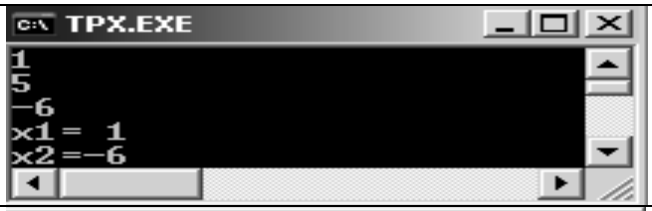
Bu yerda goto xizmatchi so‘z bo‘lib, belgi operator boshqarishni uzatishi zarur bo‘lgan (belgilangan) «manzili» hisoblanadi. Belgi sifatida Turbo Paskal dasturlash tilida 0 dan 9999 gacha bo‘lgan butun sonlardan va simvollar birikmasidan(xizmatchi so‘zlardan tashqari) foydalanish mumkin. Belgilar dasturning tavsiflash qismining Label (nishonlar ro‘yxati) bo‘limida beriladi, masalan: Label 12, bel, r1;

Yuqoridagi operatorlardan foydalanib, $ax^2+bx+c=0$ tenglamaning yechimlarini aniqlovchi dastur yaratamiz:

```
uses crt;
label 12,13,aa,2;
var a,b,c:integer; x, d:real;
begin
 clrscr;
readln(a,b,c);
d:=sqr(b)-4*a*c;
    if d=0 then goto aa else goto 12;
aa:x:=-b/(2*a);
  writeln('x=',x:2:0);  goto 13;
  12: if d>0 then
  begin
  writeln('x1=',(-b+sqrt(d))/2*a:2:0);
  writeln('x2=',(-b-sqrt(d))/2*a:2:0);
  end
  else
  writeln('tenglama haqiqiy yechimlarga ega emas');
13:end.
```

[52] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 19.

Keltirilgan dasturni turli holatlarda tenglamaning ildizlarini aniqlashini ko‘rib chiqamiz.

Tenglamaning ko‘rinishi	Dastur natijasi
x^2-8x+16=0(A=1,b=-8,c=16) qiymatning readln(a,b,c) buyrug‘i orqali kiritilishiga e’tibor qarating.	
x^2+5x-6=0(A=1,b=5,c=-6) qiymatning readln(a,b,c) buyrug‘i orqali kiritilishiga e’tibor qarating.	
$5x^2$+x+6=0(A=5,b=1,c=6)	

Goto operatoridan foydalanishda quyidagi qoidalarga amal qilish shart: Goto operatori boshqarishni uzatuvchi belgi nomi albatta tavsiflash bo‘limida ko‘rsatilishi va u dasturning kerakli joyida «:» bilan ajratilgan holda aniqlanishi shart. (ko‘rsatilgan misoldagi 12: if d>0 then… kabi)

Goto operatori boshqarishni uzatuvchi belgi nomi tavsiflash bo‘limida ko‘rsatilishi va u dasturning asoiy qismida foydalanmaslik mumkin. (ko‘rsatilgan misolda, label 12,13,aa,2; da aniqlangan «2» belgisidan dasturda foydalanilmagan)

Tanlash (Case) operatori

Bu operator bir necha yo‘nalish bo‘yicha tarmoqlanishni ta’minlab beruvchi(tanlashni amalga oshiruvchi) operator hisoblanadi. Uning umumiy ko‘rinishi quyidagicha:

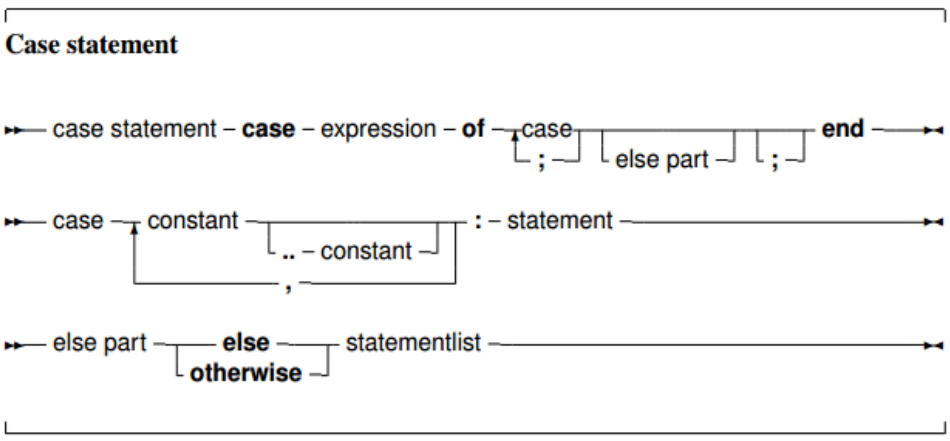
Case <tanlash indeksi-kalit> of <tanlash holatlari ro‘yxati-elementlari> else <operatorlar >end;

Bu yerda Case, of, else va end paskalning xizmatchi so‘zlari; <tanlash indeksi-kalit> - sonli, belgili yoki mantiqiy o‘zgaruvchi

yoki ifoda; <tanlash holatlari ro‘yxati-elementlari> - tanlash indeksi-kalitiga mos qiymatlar. [53]

13.2.2 The Case statement

Free Pascal supports the case statement. Its syntax diagram is



Tanlash indeksi sifatida haqiqiy tipdan foydalanish mumkin emas va bu indeks tanlash holatlari ro‘yxatidagi mos buyruqlni bajarilishini ta’minlaydi. [54]

```
case Number of
  1: Text := 'One';
  2: Text := 'Two';
  3: Text := 'Three';
end;

case aChar of
  '+' : Text := 'Plus sign';
  '-' : Text := 'Minus sign';
  '*', '/': Text := 'Multiplication or division';
  '0'..'9': Text := 'Number';
  'a'..'z': Text := 'Lowercase character';
  'A'..'Z': Text := 'Uppercase character';
else
  Text := 'Unknown character: ' + aChar;
end;
```

Case operatoridan foydalanishni quyidagi misollarda ko‘rib chiqamiz:

5-misol. «Sadaf» kichik tadbirkorlik firmasi bir kecha kunduzda W kVt/soat elektr energiyasini sarflaydi. Bu firmaning 2011

[53] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 169.
[54] Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. Pp 62-63.

yilning kerakli oylari uchun elektr energiyasini sarflash miqdorini aniqlang(Tanlash indeksi sifatida butun tipdan foydalanish).

```
const yil=2011;
var W,R:real; j:word;
begin
  writeln('Oyning tartib raqami ba bir kecha-kunduzdagi');
  writeln('sarflanadigan energiya miqdorini kiriting!');

readln(j,W);
case j of
   1,3,5,7,8,10,12: r:=31*W;
      4,69,11: r:=30*w;
         2: if (yil mod 4=0) then r:=29*W else r:=28*W;
  else writeln('oy tartib raqami xato kiritilgan')
 end;
 if (j>0) and (j<13) then
  begin
    writeln (j, '-nchi oyda ', r:6:2,'kvt/s miqdorda');
    writeln(' elektr energiyasi sarflangan');
  end;
end.
```

Keltirilgan dasturning bir qismiga izoh keltiramiz:

case j of **1,3,5,7,8,10,12: r:=31*W;** **4,69,11: r:=30*w;** **2: if (yil mod 4=0) then r:=29*W else r:=28*W;** **else writeln('oy tartib raqami xato kiritilgan')** **end;**	**Kiritilgan oy tartib raqami j-ga ko‘ra tanlash boshlanadi (oydagi kunlar miqdoriga nisbatan) va shunga mos R-ning qiymati hisoblanadi**

6-misol. Uchburchak tomonlarini ifodalovchi uchta son kiritilib bu sonlar uchburchak tomonlari bo‘lganda uning perimetri aks holda ularning uchburchak tomonlari bo‘la olmasligi haqidagi ma’lumotni

beruvchi dastur yarating (Tanlash indeksi sifatida mantiqiy tipdan foydalanish).

```
var t:boolean;
  a,b,c:real;
  begin
  writeln ('uchburchak tomonari uzunligini kiriting‘);
readln(a,b,c);
t:=(a+b>c) and (a+c>b) and (b+c>a);
case t of
  true: writeln('bunday uchburchak mavjud va uning perimetri ',(a+b+c):4:0,' ga teng‘);
  false :writeln('bunday uchburchak mavjud emas');
end;
readln;
end.
```

Dastur natijasi:

```
uchburchak tomonari uzunligini kiriting
67
45
60
bunday uchburchak mavjud va uning perimetri  172 ga teng
```

7-misol. Kiritilgan a va b haqiqiy sonlar uchun qo‘shish, ayirish, ko‘paytirish va bo‘lish belgisini kiritilganda bu sonlar ustida mos arifmetik amallarni bajaruvchi dastur yarating(Tanlash indeksi sifatida char tipidan foydalanish).

```
uses crt;
var
  t:char;
  a,b:real;
  begin
  writeln ('ikkita sonni kiriting‘);
 readln(a,b);
 writeln( 'arifmetik amallarga mos belgini kiriting‘) ;
 readln(t);
```

```
case t of
'+': writeln(a+b);
'-': writeln(a-b);
'*': writeln(a*b);
':': writeln(a/b);
'/': writeln(a/b)
 else  writeln ('arifmetik amallarga moc belgini kiriting‘);
end;
end.
```

Dastur natijasi:

```
ikkita sonni kiriting
20
20
arifmetik amallarga mos belgini kiriting
*
 4.00000000000000E+0002
```

8-misol. [55]Talabalarning baholash dasturi.

```
var
  Mark: Integer;
begin
  Write('Please enter student mark: ');
  Readln(Mark);
  Writeln;

  case Mark of
    0 .. 39 : Writeln('Student grade is: F');
    40 .. 49: Writeln('Student grade is: E');
    50 .. 59: Writeln('Student grade is: D');
    60 .. 69: Writeln('Student grade is: C');
    70 .. 84: Writeln('Student grade is: B');
    85 .. 100: Writeln('Student grade is: A');
  else
    Writeln('Wrong mark');
  end;

  Write('Press enter key to close');
  Readln;
end.
```

Parametrik takrorlash operatori

[55] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 26.

Yechilayotgan masalaning mohiyatiga qarab, dasturchi tuzuvchi o‘zi uchun qulay bo‘lgan takrorlash operatorini tanlab olishi mumkin.

Takrorlash operatorlarining 3 xil turi mavjud:

· parametrli takrorlash operatori;

· repeat takrorlash operatori;

· while takrorlash operatori.

Dasturlash jarayonida ayrim hollarda bir yoki bir necha amallarni bir necha marotaba takrorlab bajarish zarurati tug‘iladi. Masalan, 1+2+…+2011 yig‘indini hisoblashimiz hisoblash uchun quyidagi dastur tuzadigan bo‘lsak,

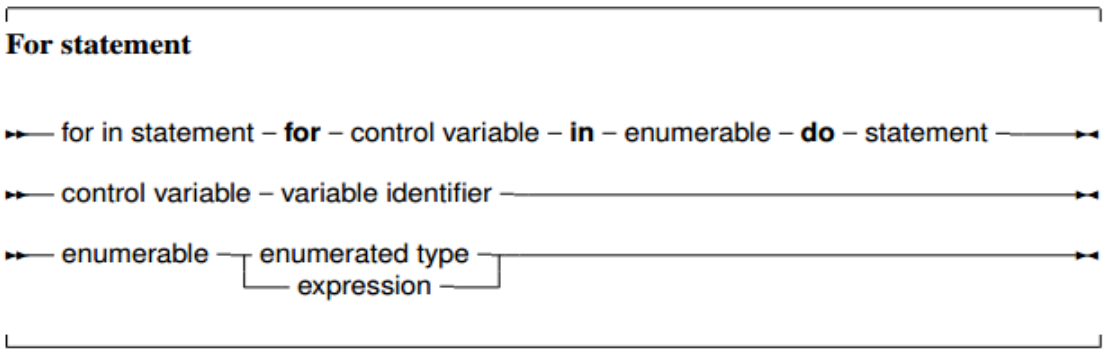
..

a:=1; S:=s+a;

a:=2; S:=s+a;

a:=3; S:=s+a;

a:=4; S:=s+a; va hokazo, ya’ni dasturimiz «uzundan-uzun» ko‘rinishga ega bo‘lar edi. E’tibor bilan qaraydigan bo‘lsak a-o‘zgaruvchi har safar 1-ga ortib borib, ega bo‘lgan qiymati S-ga qo‘shilmoqda. Aynan shunday hollar uchun parametrli sikllardan foydalanish dasturchining ishini yengillashtiradi. [56]



Parametrik sikllarning umumiy ko‘rinishi quyidagicha:

For <Sikl parametri>:=<a> to <b> do

[56] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 173.

< operator yoki operatorlar>

Bu yerda a-parametr bosh qiymatiga, b-parametr oxirgi qiymatiga teng. [57]

```
var
  total, I: Integer;

begin
  total := 0;
  for I := 1 to 10 do
    total := total + I;
```

This same for statement could have been written using a reverse counter:

```
var
  total, I: Integer;

begin
  total := 0;
  for I := 10 downto 1 do
    total := total + I;
```

9-misol:

For i:=1 to 23 do
s:=s+1/I;

Siklning bu holatida parametr i-ning qiymati dastlab 1-ga teng bo‘lib, sungra siklning har bir qadamida ‘+1’-ga orta boradi va 2,3,…,23 ga teng bo‘ladi. Zarur hollarda parametrning qiymatini ‘-1’ orttirish mumkin bo‘lib, bunda «to» o‘rniga «downto» ishlatiladi.

10-misol:

For k:=30 downto 1 do
begin W:=W+sqr(k); R:=r+sqrt(k); end;

<operator> ko‘rinishidagi sikl bo‘lib, ayrim hollarda ham ishlatiladi. Sikl parametrining qiymati faqat butun sonlardan iborat va sikl qadami doimo birga teng.

Parametrik sikllarning o‘ziga xos xususiyatlari quyidagilardan iborat:

[57] Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. . Pp 64-65.

For siklidan takrorlanishlar soni aniq bo‘lgan hollarda foydalanish maqsadga muvofiqdir. [58]

```
var
  i: Integer;
  Count: Integer;
begin
  Write('How many times? ');
  Readln(Count);
  for i:= 1 to Count do
    Writeln('Hello there');

  Write('Press enter key to close');
  Readln;
end.
```

Sikl parametri qiymati +1 yoki -1 ga avtomatik tarzda oshiriladi(«to» yoki «downto» ishlatilishiga ko‘ra).

Sikl parametri sifatida butun, belgili, mantiqiy yoki sanoq tiplaridan foydalanish mumkin.

Sikl bir necha amalni bajarishga mo‘ljallangan bo‘lsa, sikl tanasida bu amallar «begin» va «end» qavslari ichida berilishi shart(2-misolga qarang).

Parametrik takrorlanishlar «ichma-ich» joylashishlari ham mumkin va bu holat juda ko‘p masalalarni yechishda qo‘llaniladi.

Masalan:

```
for t:=1 to 3 do
for k:=1 to 5 do
writeln(t,k);
```

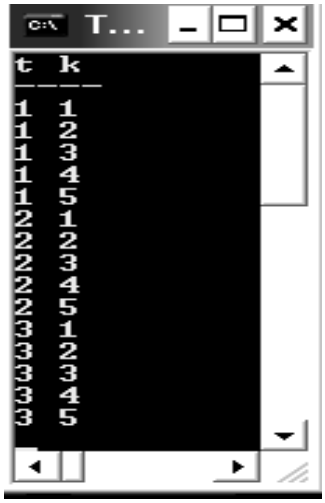
Bu sikllarni aniqroq tasavvur etish uchun quyidagi dastur va uning natijasini taqqoslaymiz:

t-parametrning qiymati 1-ga teng bo‘lganda, k-parametr 1,2,3,4,5 qiymatlarni qabul qiladi.

t-parametrning qiymati 2-ga teng bo‘lganda, k-parametr yana 1,2,3,4,5 qiymatlarni qabul qiladi va hokazo.

[58] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 28.

```
uses crt;
var t,k:byte;
begin  clrscr;
writeln('t',' ','k');
writeln('----');
for t:=1 to 3 do
for k:=1 to 5 do
writeln(t,' ',k);
end.
```



11-misol. 'A' dan 'Z'-gacha va 'z' dan 'a'-gacha bo‘lgan barcha simvollarni chop etuvchi dastur tuzing.

Dastur ko‘rinishi:

```
var i:char;
 begin
 for i:='A' to 'Z' do
 write(i,' ');
 writeln;
 for i:='z' downto 'a' do
 write(i,' ');
 readln;
 end.
```

Dastur natijasi:

12-misol. Raqamlari yig‘indisi 8-ga teng bo‘lgan barcha ikki xonali sonlarni aniqlab, chop etuvchi dastur yarating.

```
uses crt;
 var i:10 ..99 ;
 a,b:0..9;
```

```
begin
clrscr;
for i:=10 to 99 do
begin
a:=i div 10;
b:=i mod 10;
if a+b=8 then write(i,' ');
end;
end.
```

Shart bo‘yicha takrorlash operatorlari

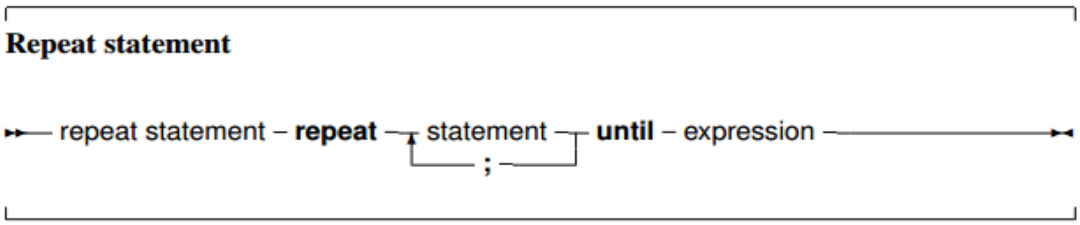
Shart bo‘yicha takrorlash operatorlari ikki xil ko‘rinishda bo‘lib ular quyidalardan iborat:

a) repeat sikli (takrorlanadigan amallar kamida bir marotaba bajarilib so‘ngra shart tekshiriladi).

bu takrorlashning tuzilishi quyidagicha:

Repeat <sikl tanasi-operatorlar ketma-ketligi>

Until <shart (mantiqiy ifoda)> [59]



Bu yerda <operatorlar ketma-ketligi> bajarilishi lozim bo‘lgan amallar yoki sikl tanasida joylashgan operatorlar majmui, <shart> takrorlanishi, bajarilishi yoki to‘xtatilishini boshqaruvchi shartdan iborat. Bu xil ko‘rinishdagi sikl hech bo‘lmaganda bir marotaba

[59] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 180.

bajariladi, negaki operatorlar ketma-ketligi shartni tekshirishdan oldin yozilgan. [60]

The following are valid `repeat` statements

```
repeat
  WriteLn ('I =',i);
  I := I+2;
until I>100;

repeat
 X := X/2
until x<10e-3;
```

Repeat takrorlash operatorini bajarilishini quyidagi masala yordamida ko‘rib chiqamiz:

Masala. $y=ax^2$ funksiya qiymatlarini x=0 dan x=5 gacha 0,5 qadam bilan hisoblovchi dastur yarating.

Masala shartiga ko‘ra, foydalanuvchi faqat a-ning qiymatini kiritishi dastur esa $y=ax^2$ funksiya qiytmatini 0,5 qadam bilan hisoblashi zarur. a=2 qiymat uchun natija quyidagicha bo‘lishi zarur, ya’ni dastavval x=0 da funksiya qiymati hisoblanishi (chop etilishi), so‘ngra x-ning qiymati 0,5 ga ottirilishi va hosil bo‘lgan qiymat 5-dan katta bo‘lmasligi tekshirilishi zarur (quyidagi jadvalga e’tibor bering).

X	0	0,5	1	1,5	2	2,5	3	3,5	4	4,5	5
Y	0	0,5	2	4,5	8	12,5	18	24,5	32	40,5	50

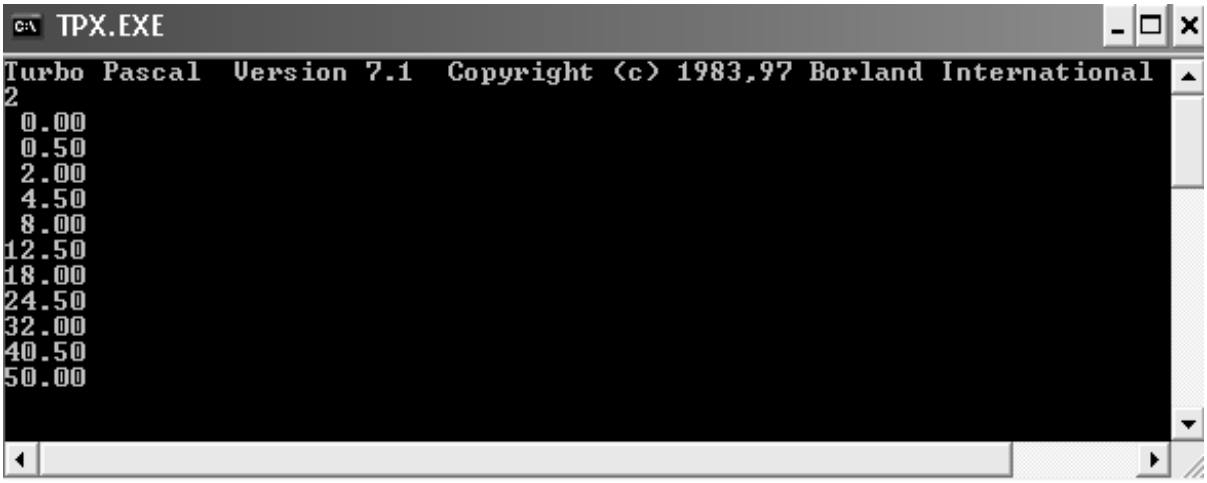
Masala shartiga mos dastur quyidagicha: var x,a,y:real;

```
begin
readln (a);
x:=0;
repeat
y:=a*sqr(x);
```

[60] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 180.

```
x:=x+0.5;
writeln(y:5:2);
 until (x>5)
end.
```

Dastur natijasi:



b) while sikli(takrorlanadigan amallar bajarilishi uchun avval shart tekshiriladi). Bu takrorlashning tuzilishi quyidagicha:

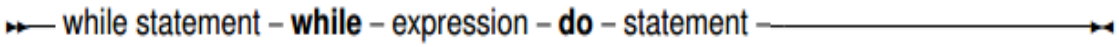
While <mantiqiy ifoda (shart)> do

<operator – sikl tanasi>

Bu yerda mantiqiy ifoda (shart) qiymati True bo‘lguncha sikl tanasidagi operatorlar bajariladi, aks hollarda sikl tanasidagi amallar bajarilmaydi. [61]

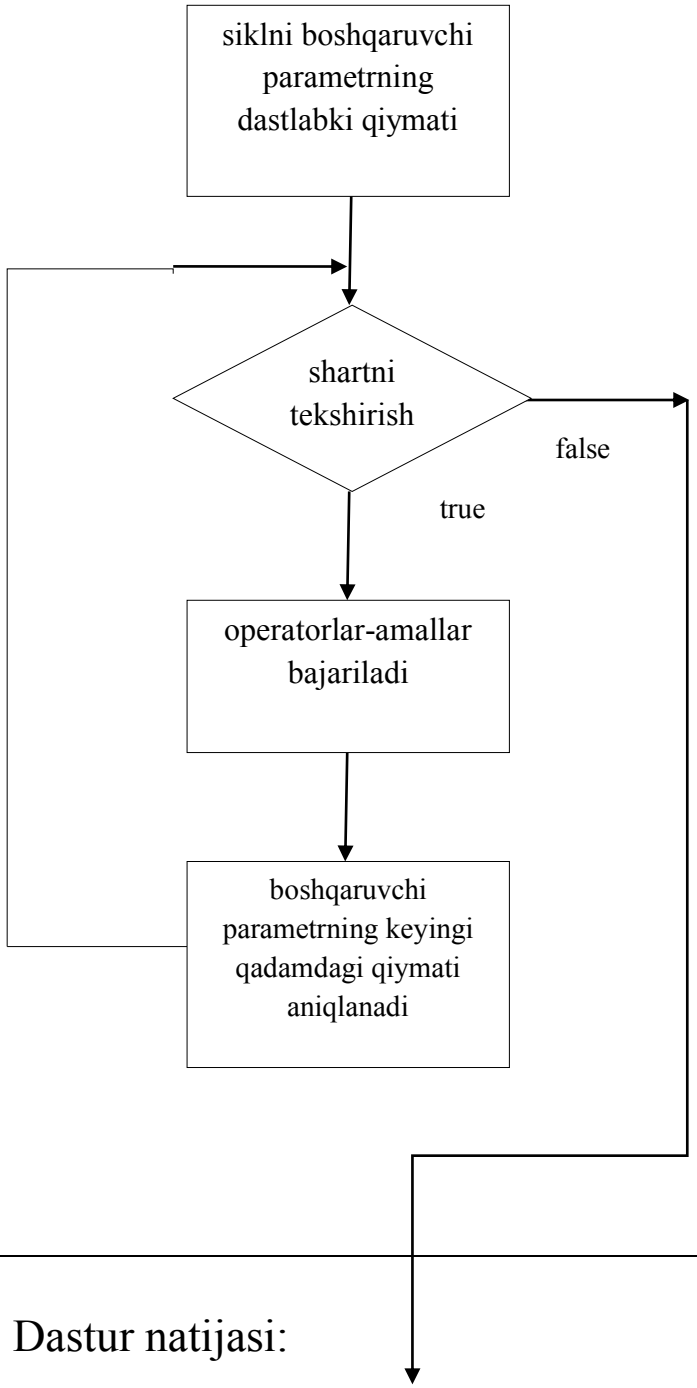
The prototype syntax of the `While..do` statement is

While statements

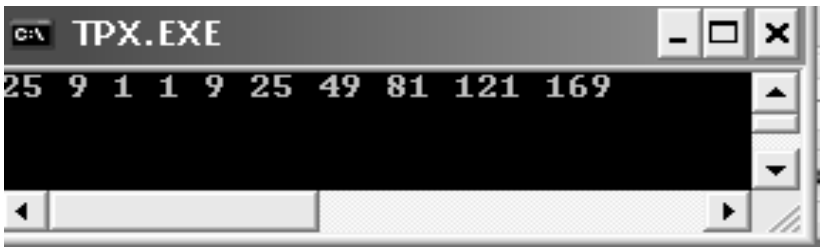


[61] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp-180.

While shartli tarorlash blok-sxemasi	Dastur-misol

	var x:integer;
	begin
	x:=-5;
	while (x<15) do
	begin
	write(sqr(x), ' ');
	x:=x+2;
	end;
	end.

Dastur natijasi:



```
while (I <= 100) and (J <= 100) do
begin
  // use I and J to compute something...
  I := I + 1;
  J := J + 1;
end;

repeat
  // use I and J to compute something...
  I := I + 1;
  J := J + 1;
until (I > 100) or (J > 100);
```

62

The following are valid `while` statements:

```
I := I+2;
while i<=100 do
  begin
  WriteLn ('I =',i);
  I := I+2;
  end;
X := X/2;
while x>=10e-3 do
  X := X/2;
```

63

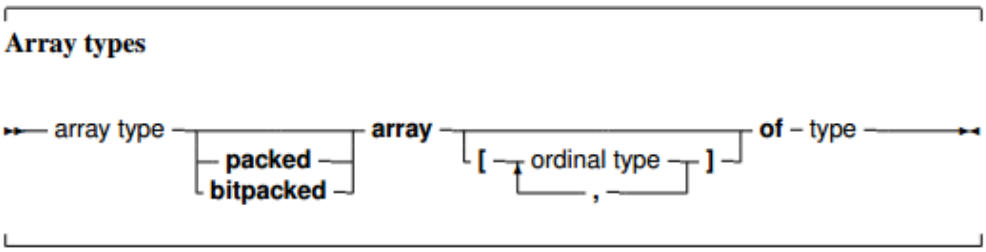
Mustahkamlash uchun savollar

1. Tarmoqlanuvchi algoritm deb nimaga aytiladi?
2. O‘tish operatori nima uchun qo‘llaniladi?
3. Dasturda nishonlar nima uchun ishlatiladi?
4. Nishonlar ishlatilgan dasturda o‘tish operatori ishlatilmasligi mumkinmi?
5. Tarmoqlanuvchi operatorida operatorlar ketma-ketligi ishtirok etsa, ular qanday so‘zlar orasida yoziladi?
6. Tarmoqlanuvchi operatorning qisqa va to‘liq ko‘rinishlari haqida nimalar bilasiz?
7. Shart bo‘yicha takrorlash operatorlarining parametrik takrorlash operatoridan farqi nimada?
8. Repeat operatorining ishlashini tushuntiring.

[62] Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. Pp 65.
[63] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 181.

13-§. Paskalda massivlar

Inson o‘zining kundalik hayotida axborotlarni o‘ziga qulay tarzda tasvirlashga odatlanib qolgan. Masalan, o‘quvchining kundalik darslarini dars jadvali yordamida tasvirlash, futbol o‘yinlari natijalarini qayd qilish kerakli ma’lumotlarni qidirib topish va ular ustida amallar bajarish jadval yordamida qulay. Siz yana karra jadvali, biror funksiyani qiymatlari jadvali, kimyoviy elementlar jadvali, sinf o‘quvchilarining navbatchilik jadvali kabi jadvallar bilan tanishasiz. Odatda jadvalni tashkil etuvchilar uning elementlari deb yuritiladi. Jadvallar bir o‘lchovli, ikki o‘lchovli, uch o‘lchovli va ko‘p o‘lchovli bo‘lishlari mumkin. Hayotda ko‘proq bir o‘lchovli (chiziqli) va ikki o‘lchovli(to‘g‘ri to‘rtburchakli) jadvallar ko‘proq qo‘llaniladi. [64]



Massivlarning umumiy ko‘rinishi: [65]

The general form of type declaration of one-dimensional array is:

```
type
  array-identifier = array[index-type] of element-type;
```

Masalan, 24 ta butun sondan iborat guruhni aniqlash mumkin: [66]

```
type
  TDayTemperatures = array [1..24] of Integer;
```

[64] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp-39.
[65] Pascal tutorial. Tutorilaspoint.cm. Pp 81-82.
[66] Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. Pp 53.

For example,

```
type
  vector = array [ 1..25] of real;
var
  velocity: vector;
```

Now, velocity is a variable array of vector type, which is sufficient to hold up to 25 real numbers.

To start the array from 0 index, the declaration would be:

```
type
  vector = array [ 0..24] of real;
var
  velocity: vector;
```

Avtomobil soatiga 60 km/soat tezlik bilan harakatlansa, uning dastlabki 8 soatda bosib o‘tgan yo‘lini jadval yordamida tasvirlaydigan b o‘lsak, uning k o‘rinishi quyidagicha b o‘ladi:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
60	120	180	240	300	360	420	480

Jadval nomini V deb yuritadigan bo‘lsak, V[1]=60, V[2]=120,…, V[8]=180 larga teng ekanligini kuzatamiz.

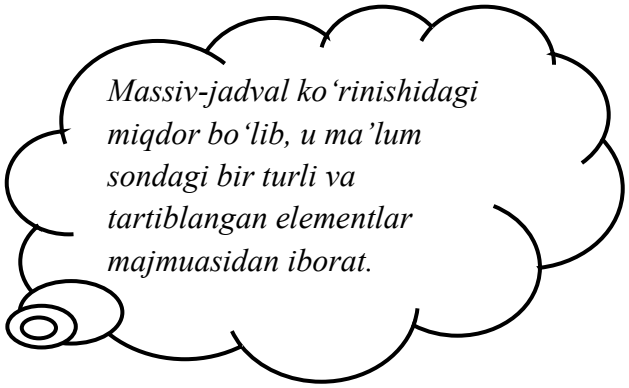
Ko‘paytirish jadvalida birinchi o‘lchov satr qiymatlaridan ikkinchi o‘lchov ustun qiymatlaridan tashkil topgan.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Jadval nomini K deb yuritadigan bo‘lsak, K[1,1]=1, K[1,2]=2,…, K[9,9]=81 larga teng ekanligini kuzatamiz.

Turbo Pascal dasturlash tilida jadval k o‘rinishidagi miqdorlarni tasvirlash va ulardan foydalanish uchun massiv tushunchasi kiritilgan.

Massiv nomga ega-bir xil tipga tegishli bo‘lgan va tartiblangan elementlar to‘plamidir. Odatda massivni bir necha usullarda berish mumkin va biz hozir ularning ayrimlari bilan tanishtirib o‘tamiz:



1. Tiplar yordamida. Bu holda tip-massiv quyidagicha aniqlanadi:

2.

<tipning nomi>= array [<indeksli tiplar r o‘yxati>] of <tip>

Bu yerda: -<tipning nomi>- t o‘g‘ri nomli kattalik;

- Array, of- xizmatchi s o‘zlar;

-<indeksli tiplar ro‘yxati>-ixtiyoriy tartibli tip(longint dan boshqa)

Misol keltiramiz:

```
type
digit = array [0..9] of Char;
matrix = array [Byte] of Single;
var
m: matrix;
d: digit;
```

3. Massivni o‘zgaruvchilar safida tip-massivdan foydalanmasdan ham berish mumkin.

Masalan:

```
Var
    A,b:array [1..10] of real;
```

Bu yerda birdaniga ikkita bir o‘lchamli massiv (har biri 10 – elementdan iborat bo‘lgan a va b massivlar) aniqlangan bo‘lib, elementlarining tipi haqiqiy sonlardan iborat.

Yuqoridagi hollarda bir o‘lchamli massivlar misol sifatida keltirilgan bo‘lib, kvadrat qavslarda massiv elementlarining indekslari, ya’ni tartib raqamlari ro‘yxati keltirilgan.

A[1]-birinchi element

A [2]-ikkinchi element

…

A [10]- o‘ninchi element

Agar massiv ko‘p o‘lchamli b o‘lsa, uning indekslari ro‘yxati bir-biridan vergul bilan ajratilib yoziladi. Misollar keltiramiz:

D: array [1..5, 1..15] of real; (ikki o‘lchamli)

kimel: array [0..25, 0..11] of string;(ikki o‘lchamli)

yok: array [2001..2011,1..12,1..6] of byte;(uch o‘lchamli)

1-misol: Meteorolgik stansiyada har bir soatda havo temperaturasi o‘lchanib alohida jadvalga to‘ldirilib boriladi:

O‘lchash vaqti, s	1	2	3	4	5	6	……	23	24
Temperatura, C	12	13	11	15	10	12	……	11	11

Bu chiziqli jadval bo‘lib, 24 elementdan tashkil topgan bo‘lib, uning elementlari 1 dan 24 gacha bo‘lgan tartib raqamlariga ega. Mazkur jadvaldan 2 tartib raqamli element 13 ga, 23 tartib raqamli element 11 ga teng.

To‘rtburchak jadval uchun gorizontal va veritikal yo‘nalishlar bo‘yicha chegaralar ko‘rsatilishi shart (qatorlar va ustunlar uchun). Bu turdagi jadval elementi uchun uning joylashgan joyiga ko‘ra qator va ustun tartib raqamiga ko‘ra aniq indekslardan foydalaniladi.

Har qanday jadval ixtiyoriy tipdagi elementlardan iborat bo‘lishi mumkin.

2-misol: Yo‘lovchilarning 6 ta vagonda 15 chi 19 chi o‘rinlarda joylashishlari jadvalini yaratamiz	

Vagon-jadvali

	1 chi vagon	**2 chi vagon**	**3 chi vagon**	**4 chi vagon**	**5 chi vagon**	**6 chi vagon**
15	Mavjuda	Salima	Kamola	Menura	Aziza	Zulfiya
16	Sherzod	Salim	Ikrom	Qudrat	Aziz	Akram
17	Tolib	Sarvar	Coli	Karim	Olim	No‘mon
18	Sojida	Gavhar	Lola	Gulruh	Zuhra	Fotima
19	Rustam	Mansur	Botir	Farruh	Azlar	Mahkam

Jadval ko‘rinishdagi miqdorlarni, ya’ni massivlarni uning elementlari tashkil etadi. Oddiyroq qilib aytganda jadval kataklarida joylashgan qiymatlar jadval elementlari b o‘lib, bu kataklar nomeri massivning indeksi deb yuritiladi. Dasturchi nuqtayi nazaridan massivni jadval k o‘rinishida tasavvur qilish nihoyat qulay. Masalan, M massiv 6 elemendan iborat b o‘lib, quyidagi jadval yordamida elementlari berilgan b o‘lsin:

1	*2*	*3*	*4*	*5*	*6*	***- elementlar indeksi***
12	**2**	**0**	**-2**	**12**	**-18**	***- elementlar qiymati***

U holda M[1]=12, M[2]=2, M[3]=0, M[4]=-2, M[5]=12, M[6]=-18 shu massiv elementlarini tashkil qiladi.

Quyidagi 3-misolda biz foydalanuvchi 10 ta talabaning bahosini kiritishi va ularni massivga joylashtirishini ko‘rib chiqamiz: [67]

[67] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 43-44.

```
var
  Marks: array [1 .. 10] of Integer;
  i: Integer;
begin
  for i:= 1 to 10 do
  begin
    Write('Input student number ', i, ' mark: ');
    Readln(Marks[i]);
  end;

  for i:= 1 to 10 do
  begin
    Write('Student number ', i, ' mark is : ', Marks[i]);
    if Marks[i] >= 40 then
      Writeln(' Pass')
    else
      Writeln(' Fail');
  end;

  Writeln('Press enter key to close');
  Readln;
end.
```

O‘zbekiston futbol jamolarining turnir natijalari jadval ko‘rinishida keltirilgan. Agar uni FF massivi deb yuritadigan bo‘lsak:

FF[1] =’Bunyodkor’
FF[2] =’Paxtakor’
FF[3] =’Nasaf’
FF[4] =’Sho‘rtan’
FF[5] =’Mash’al’
FF[6] =’Metallurg‘
FF[7] =’Andijon’
FF[8] =’Qizilqum’
FF[9] =’Navbahor’
FF[10]=’Neftchi’
FF[11]=’Olmaliq’
FF[12]=’Dinamo‘
FF[13]=’lokomotiv’
FF[14]=’Xorazm’

O‘ZBEKISTON PROFESSIONAL FUTBOL LIGASI

№	Jamoalar
1	Bunyodkor
2	Paxtakor
3	Nasaf
4	Sho’rtan
5	Masha’l
6	Metalurg
7	Andijon
8	Qizilqum
9	Navbahor
10	Neftchi
11	Olmaliq
12	Dinamo
13	Lokomativ
14	Xorazm

Massiv elementlarini kiritishning bir necha yo‘li mavjud va hozir shulardan o‘zlashtirish operatori yordamida va klaviatura orqali elementni kiritish usuliga misollar ko‘rib chiqamiz:

Massiv elementlarini o‘zlashtirish operatori yordamida kiritish uchun massiv dasturning tavsiflash qismida e’lon qilinadi. So‘ngra dasturning asosiy qismida elementlar ketma-ket quyidagicha kiritiladi:

Massiv nomi[element indeksi]:=<elementning qiymati>;
Misol. «Hafta kunlari» massivini yaratish.

```
var Hafta_kuni:array[1..7] of string; i:byte;
begin

  Hafta_kuni[1]:='Yakshanba';
  Hafta_kuni[2]:='Dushanba';
  Hafta_kuni[3]:='Seshanba';
  Hafta_kuni[4]:='Chorshanba';
  Hafta_kuni[5]:='Payshanba';
  Hafta_kuni[6]:='Juma';
  Hafta_kuni[7]:='Shanba';
  end.
```

Dasturning kerakli joyida elementni kiritish uchun quyidagi tartibda buyruqni kiritamiz:

Readln(Massiv nomi[element indeksi]);
Misol: Ishqoriy metallar massivini yarating.

```
var Ishqormet: array [1..5] of string;
begin
readln(Ishqormet [1]);
readln(Ishqormet [2]);
readln(Ishqormet [3]);
readln(Ishqormet [4]);
readln(Ishqormet [5]);
end.
```

Foydalanuvchi ishqoriy metallar nomlarini(litiy, natriy, kaliy, rubidiy, seziy) massiv elementlari sifatida kiritib chiqadi va ular

mos ravishda quyidagi o‘zlashtirish operatori bilan teng kuchli bo‘lishadi:

```
Ishqormet [1]:=’Litiy’;
Ishqormet [2]:=’Natiry’;
Ishqormet [3]:=’Kaliy’;
Ishqormet [4]:=’Rubidiy’;
Ishqormet [5]:=’Seziy’;
```

Ko‘rib o‘tganimizdek agar massiv elemenlari soni ko‘proq bo‘lsa, ularni kiritish har ikki usulda ham qiyinchilik tug‘diradi, ya’ni dastur hajmi «kattalashib» boradi. Bu holatni sikl operatorlari yordamida bartaraf etish mumkin. Shunday operatorlardan biri ***for*** operatori hisoblanadi. Bu operator bir buyruqdan takrorlanish yordamida bir necha marotaba foydalanish imkoniyatini beradi:

```
for i:=1 to 10 do
readln(a[i]);
```

Keltirilgan buyruqlar A-massivning 10 ta elementini kiritishga qulay bo‘lib, readln(a[1]), readln(a[2]),…, readln(a[10]) buyruqlari bilan teng kuchli hisoblanadilar.

Dastur yordamida massiv elementlari yoki ular bilan bajarilgan amallar natijasini ekranda ko‘rish uchun writeln operatoridan foydalanish mumkin. Masalan, writeln (a[5])-buyrug‘i a-massivning beshinchi elementini ekranda chop etadi.

4-misol. 10 ta butun sonlardan A-massiv elementlarini klaviaturadan kiritish va ularni chop etish dasturini ko‘rib chiqamiz:

```
var a:array [1..10] of integer;
i:1..10;
begin
for i:=1 to 10 do readln(a[i]);
for i:=1 to 10 do writeln('a[',i,']=',a[i],' ');
readln ;
end.
```

Misol. Elementlari tasodufiy yaratilgan butun sonlardan iborat bo‘lgan 15 elementli A va B massivlardan quyidagi shartni qanoatlantiruvchi uchinchi C massivni yarating.

C[i]:=A[i]+B[i], bu yerda i:=1,2,3,..,15;

Javob:

```
var a,b,c:array [1..15] of integer; i:byte;
begin
for i:=1 to 15 do a[i]:=random(30);
for i:=1 to 15 do b[i]:=random(30);
for i:=1 to 15 do c[i]:=a[i]+b[i];
for i:=1 to 15 do writeln(c[i]);
end.
```

5-misol. 5 sondan iborat bir o‘lchamli D massiv berilgan. Bu massivning har bir mos elementining 10 foizdan iborat bo‘lgan yangi massiv yarating.

```
var d,d1:array [1..5] of real;
i:byte;
begin
for i:=1 to 5 do
readln( d[i]);
for i:=1 to 5 do
d1[i]:=(d[i]/100)*10;
for i:=1 to 5 do
writeln(d1[i]);
readln;
end.
```

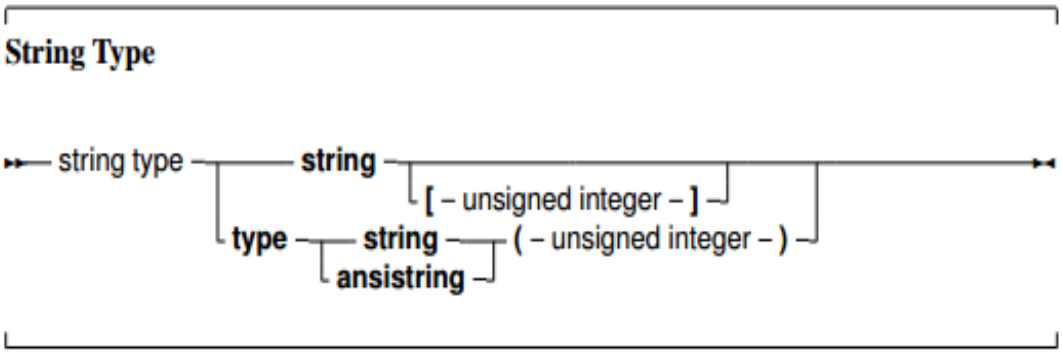
Mustahkamlash uchun savollar

1. Hayotda uchraydigan jadval ko‘rinishdagi miqdorlarga misol keltiring.
2. Chiziqli massiv qanday o‘lchovlarda bo‘ladi?
3. Ikki o‘lchovli massiv qanday ko‘rinishda bo‘ladi?
4. Massivda indeks nima uchun kerak?
5. Massiv elementlarining indekslari qanday qiymatlar qabul qilishi mumkin?

6. Jadval ko‘rinishidagi miqdorlarning turlarini qanday ajiratish mumkin?

14-§. Pascalda satriy kattaliklar bilan ishlash

Turbo Pascal dasturlash tilida String tipidan matnli ma’lumotlar bilan ishlashda foydalaniladi.



Bu tipning har bir elementi xotirada 1 bayt joy egallovchi belgilardan iborat bo‘ladi. Satriy o‘zgaruvchida belgilar miqdori 0 dan 255 tagacha bo‘lishi mumkin. Satriy kattalik o‘zgaruvchilar (Var) bo‘limida quyidagicha e’lon qilinishi mumkin.

```
var d:string;
 satr:string [15];
var
A : String;
```

Bu yerda d –kattalikning uzunligi 0 dan 255 tagacha, satr-kattalikning uzunligi esa 0 dan 15 gacha bo‘lishi nazarda tutilgan. Har qanday satriy tipidagi kattalikda birinchi baytning indeksi 0 ga teng bo‘lib, unda satriy kattalik uzunligi haqidagi ma’lumot joylashadi. Satriy kattalikning birinchi belgisi ikkinchi baytni egallaydi va uning indeksi birga teng.

Satrli miqdorlaning elementlari kattalik nomi va kvadrat qavslarga olingan elementning tartib nomeri yordamida aniqla-nadilar. Quyida foydalanuvchining ismini kiritish dasturini tuzamiz:

```
var
  Name: string;
begin
  Write('Please enter your name : ');
  Readln(Name);
  Writeln('Hello ', Name);

  Writeln('Press enter key to close');
  Readln;
end.
```

Ushbu misolda esa foydalanuvchi haqidagi ma'lumotlarni kiritish dasturini tuzamiz: [68]

```
var
  Name: string;
  Address: string;
  ID: string;
  DOB: string;
begin
  Write('Please enter your name : ');
  Readln(Name);
  Write('Please enter your address : ');
  Readln(Address);
  Write('Please enter your ID number : ');
  Readln(ID);
  Write('Please enter your date of birth : ');
  Readln(DOB);
  Writeln;
  Writeln('Card:');
  Writeln('----------------------------------------');
  Writeln('| Name    : ', Name);
  Writeln('| Address : ', Address);
  Writeln('| ID      : ', ID);
  Writeln('| DOB     : ', DOB);
  Writeln('----------------------------------------');

  Writeln('Press enter key to close');
  Readln;
end.
```

Yana bir misol:

```
program EssHello;
{$APPTYPE CONSOLE}

var
  strMessage: string;
begin
  strMessage := 'Hello, Small World';
  writeln (strMessage);
  // wait until Enter is pressed
  readln;
end.
```

[68] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 35.

Misol:

Satr:=’Geologiya’ bo‘lsa, satr[1]=’G‘-ga, satr[2]=’e’-ga va h.

ShortString tipining uzunligi string tipi kabi 255 ga teng:

ShortString = String[255];

Quyidagi dastur yordamida satriy kattalikning uzunligini uning birinchi baytiga ko‘ra, ya’ni 0-indeks orqali aniqlaymiz.

```
var t:string;i:byte;
begin
t:='informatika';i:=ord(t[0]);writeln(i);end.
```

Dasturdagi ord(t[0]) protsedurasi t-kattalikning uzunligini aniqlashda yordam beradi. Bu qiymatni Length(t) funksiyasi yordamida ham aniqlash mumkin. Uzunlik qiymati mazkur kattalik uchun 11 ga tengligi dastur natijasidan aniqlanadi. Quyidagi dastur yordamida satriy kattalikning har bir elementini alohida «ko‘rishimiz» mumkin:.

```
var t:string; i,k:byte;
begin
t:='informatika';i:=ord(t[0]);
for k:=1 to i do
writeln(k,t[k]);
end.
```

O‘zgarmas satriy kattalik:

```
S := ’This is a string.’;
S := ’One’+’, Two‘+’, Three’;
S := ’This isn’’t difficult !’;
S := ’This is a weird character : ’#145’ !’;
```
[69]

String tipining xotira o‘lchami

Table 3.6: String memory sizes

String type	Stack size	heap size
Shortstring	Declared length + 1	0
Ansistring	Pointer size	L + 1 + HS
Widestring	Pointer size	2*(L + 1) + WHS
UnicodeString	Pointer size	2*(L + 1) + UHS
Pchar	Pointer size	L+1

[69] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 35-37.

Endi satriy kattaliklar ustida ayrim funksiya va protseduralar yordamida bajaraladigan amallar bilan tanishamiz:

Concat(s1,[s2,…,sN]) – funksiyasi (tipi String) s1,s2,…,sN satriy kattaliklarni «yig‘indisini» beradi (bu funksiya vazifasini ‘+’ amali ham bajaradi)

Misollar keltiramiz:

Ifoda	**Natija**
‘r1+’+’r2+’+’r3+’+’…+’+’rn’	**‘r1+r2+r3+…+rn’**
Concat (‘Rezistor ‘, ‘-qarshilik ’,’degan ma’’noni ’,anglatadi’)	‘Rezistor-qarshilik degan ma’noni anglatadi’

Copy (st, t1, d) – funksiyasi (tipi String) st satriy kattalikning t1 tartib raqamli belgisidan d ta simvolni (t1 dan boshlab) nusxalash imkonini beradi.

Misollar keltiramiz:

St-qiymati	**Ifoda**	**Natija**
‘Elektr qarshilik o‘’tkazgich uzunligiga bog‘’liq’	Copy(St,8,10)	‘qarshilik’
‘Mol-modda miqdori’	Copy(St,11,6)	‘miqdor’

Delete (st, t1, d) - protsedurasi st catriy kattalikning t1 tartib raqamli belgisidan d-ta simvolni (t1-dan boshlab) o‘chirish imkonini beradi. Delete protsedurasidan foydalanganda uning uzunligi kamayadi.

Misollar keltiramiz:

St-qiymati	**Operator**	**Natija St-ning yangi qiymati**
‘abcdefg‘	Delete(S,3,2)	‘abefg‘
‘abcdefg‘	Delete(S, 2,6)	‘a’

length(st) - funksiyasi (tipi Integer) st satriy kattalikning uzunligini aniqlaydi.

Misollar keltiramiz:

St-qiymati	**Ifoda**	**Natija**
‘Zomin tog‘’-o‘’rmon qo‘’riqxonasi’	Length(St)	30
‘mediana’	Length(St)	6
‘(a+b)*c’	Length(St)	7

Pos (st1, st) - funksiyasi (tipi Integer) st satriy kattalikning tarkibida st1 satriy kattalik “joylashgan” bo‘lsa, uning tartib raqamini beradi, aks holda 0 qiymatni beradi. St1 satriy kattalik bir necha marotaba uchragan holda ham birinchisining tartib raqamini beradi.

Misollar keltiramiz:

st1 qiymati	**Ifoda**	**Natija**
‘abcdef’	Pos(‘cd’,S2)	3
‘abcdcdef’	Pos(‘cd’,S2)	3
‘abcdef’	Pos(‘k’,S2)	0

Val (a2, x,d) - protsedurasi a_2 satriy kattalikni mumkin bo‘lgan holda, x-ga haqiqiy yoki butun tipidagi son sifatida, aks holda x-ga 0-qiymatini o‘zlashtiradi.

Misollar keltiramiz:

1-dastur	**Natija**	**Hulosa**

**program val1;
var
a2:string;d,x:integer;
begin
a2:='rrr'; Val(a2,x,d);
writeln(x);
end.**
	0	A2-satriy miqdorni butun x-soniga o‘zlashtirishida uni son ko‘rinishida ifoda-lay olmasligi tyfayli, x-ga 0 qiymatni o‘zlashtirdi

2-dastur	Natija	Xulosa

program val2; **var** **a2:string;d,x: integer;** **begin** **a2:='2011';Val(a2,x,d);** **writeln(x/20:4:1);** **end.**	100.6	A2-satriy miqdorni butun x-soniga o‘zlashtirdi, uni ko‘rsatilgan formatda 20 ga bo‘lib chop

Quyidagi protseduralarni mustaqil o‘rganing va misollar ko‘rib chiqing:

Insert (st1, st, t1) - protsedurasi st1 satriy kattalikning t1 tartib raqamli belgisidan st satriy kattalik tarkibiga joylashtiradi.

Str (x, st) – protsedurasi x haqiqiy yoki butun tipidagi son bo‘lsa, uni satriy kattalikga aylantiradi.

Yana ikkita satriy kattalik bilan =, <>,<,>, >=,=< munosabat amallari ham bajariladi. Bu munosabatlar ikkita satriy miqdorlarni taqqoslash uchun ishlatiladi. Taqqoslash natijasi *true* yoki *false qiymatlarni qabul qiladi va bu amal* Concat amaliga nisbatan past darajali hisoblanadi. Satriy miqdorlarni taqqoslash quyidagicha bajariladi: chapdan o‘ngga elementlar taqqoslanib boriladi va bu teng indeksli elementlar “mos” kelmaguncha davom ettiriladi. Mos kelmagan elementlarning qaysi biri ASCII jadvalida katta qiymatli indeksga ega bo‘lsa, shu satriy miqdor katta hisoblanadi. Agar satriy miqdorlarning uzunligi har xil bo‘lib, bosh qismlari bir xil elementlardan iborat bo‘lsa, ikkita satriy miqdordan “uzuni” katta hisoblanadi. Bir xil uzunlikka ega va mos indeksli belgilari bir xil bo‘lgan satriy miqdorlar o‘zaro teng hisoblanadilar.

Misollar:

Ifodalar	**Natija**
‘True1’<’True2’	True
‘Mother’>’MOTHER’	True
‘Geografiya’>’Geologiya’	False
‘Cat’=’Cat’	True
‘uzunlik’>’balandlik’	True

Yuqorida keltirilgan funksiya va protseduralardan dasturlarda qo‘llashga misollar keltiramiz:

1-misol.

```
uses crt;
var
s1,s2,s3,s4,s5:string;
begin  clrscr;
s1:='satriy';
s2:='kattaliklar';
s3:=concat(s1,' ',s2);
writeln(s3);
s4:=s1+' '+s2;
writeln(s4);
writeln(length(s3));
s5:=copy(s3,length(s3)-2,3);
writeln(s5);
end.
```

Dastur natijasi:

2-misol.

```
uses crt;
var
s1,s2,s3,s4,s5:string;
i:integer;
begin  clrscr;
s1:='SAMARQAND';
 i:=pos('QAND',s1);
 writeln(i);
 s2:='biologiya';
 s3:='fizika';
```

```
 delete(s2,1,2);
 delete(s3,5,2);
 insert(s2,s3,5);
 writeln(s3);
readln;
end.
```

dastur natijasi:

Kiritilgan ixtiyoriy jumla tarkibidagi “A” belgisini “O” belgisiga almashtirish chop etuvchi dastur tuzing.

1-usul	**2-usul**

PROGRAM sim1;
VAR jumla:string;i,d: INTEGER;
SIM:STRING;
 	
BEGIN
read(Jumla);d:=length(jumla);
writeln(d);
FOR I:=1 TO D DO
BEGIN
IF COPY(JUMLA,I,1)='A'
THEN JUMLA[I]:='O‘;
END;
WRITELN(JUMLA); readln;
END.
	PROGRAM sim1;
	VAR jumla:string;i,d: INTEGER;
	SIM:STRING;
	
	BEGIN
	read(Jumla);d:=ord(jumla[0]);
	writeln(d);
	FOR I:=1 TO D DO
	BEGIN
	IF JUMLA[I]='A'
	THEN JUMLA[I]:='O‘;
	END;
	WRITELN(JUMLA);
	readln;
	END.

Burchak qiymati gradus o‘lchovida kiritilgan holat uchun sinus, kosinus, tangens funksiyalari qiymatini mos belgini tanlash yo‘li bilan hisoblovchi dastur tuzing.

```
PROGRAM char1;
  uses crt;
  VAR sim:char; i,grad:INTEGER;javob,radian:real; label 15,20;
   BEGIN
   writeln('burchak qiymatini gradus o‘lchoviga kiriting‘);
   readln(grad); radian:=(pi*grad)/180;
   writeln('sinus ', grad,' ni hisoblash uchun "s" tugmasini
tanlang‘);
   writeln('kosinus ', grad,' ni hisoblash uchun "c" tugmasini
tanlang‘);
   writeln('tangens ', grad,' ni hisoblash uchun "t" tugmasini
tanlang‘);
   sim:=readkey;
   case sim of
   's': javob:=sin(radian);
   'c': javob:=cos(radian);
   't': if cos(radian)=0 then goto 15 else javob:=sin
(radian)/cos(radian);
   end;
   WRITELN(javob:5:2);goto 20;
   15:writeln ('qiymati mavjud emas');
   20: readln;
   END.
```

Tarkibida raqamlar bo‘lgan s1-satriy kattalikning barcha raqamlridan yangi s2 –satiy kattalik yaratuvchi dastur tuzing.

```
Program Satriy_kattaliklar;
Var s1, s2: string; i: byte;
begin
  writeln('Tarkibida paqamlar bo‘‘lgan satriy kattalikni
kiriting‘);
  readln(s1);
  s2:='';
  for i:=1 to length(s1) do
     if (s1[i]>='0') and (s1[i]<='9')
     then s2:=s2+s1[i];
  writeln('natija ',s2);
end.
```

Dasturni quyidagi qiymat uchun tekshirib to‘g‘ri tuzilganiga ishonch hosil qiling:

S1:=’Balandligi 375 m bo‘lgan poytaxtdagi teleminora’;
Natija: 375

Masala

S-satriy miqdorning boshida joylashgan barcha bo‘shliq belgilarini o‘chiruvchi dastur tuzing.

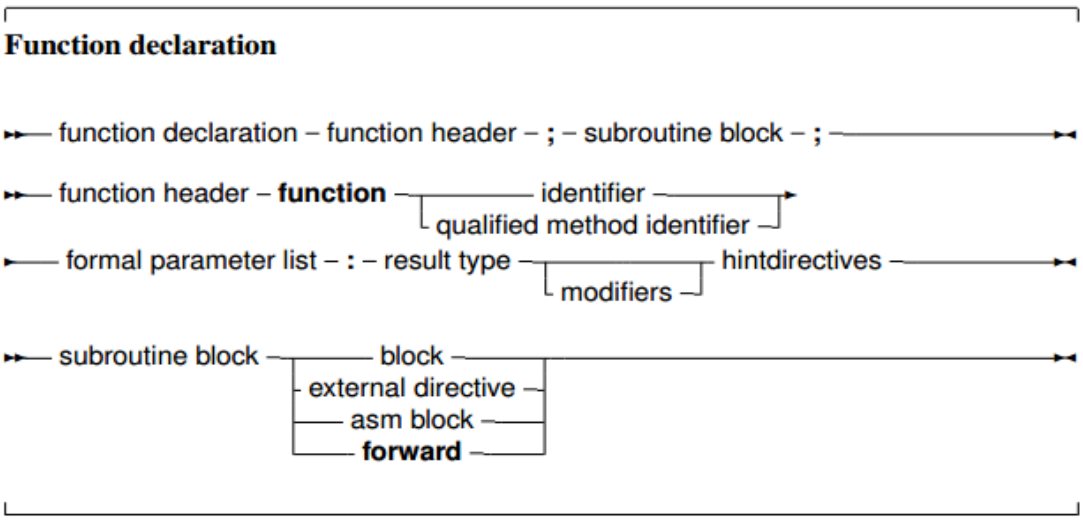
```
Program Sinov1;
Var s: string[80];
Begin
 writeln(Boshida bir necha bo‘shliq belgisi qatnashgan s-satriy miqdor ni kiriting‘);
  readln(s);
 while (pos(‘ ‘,s)=1) and (length(s)>0) do
 delete(s,1,1);
 write(Natija,s);
end.
```

Mustahkamlash uchun savollar

1. Paskal dasturlash tilida matnli ma’lumotlar bilan ishlashda qanday tipidan foydalaniladi?
2. Concat funksiyasining vazifasi nimadan iborat? Misollar keltiring.
3. Copy funksiyasi nima uchun qo‘llaniladi?
4. Delete protsedura nima va unga misol keltiring.
5. Length funksiyasi nima ish bajaradi, misol bilan tushuntiring.
6. Pos funksiyasi qanday vazifani bajaradi va qachon qiymati nolga teng bo‘ladi.
7. Val protsedurasini vazifasini tushuntiring.
8. Paskalda berilgan satr ichiga boshqa satrni joylashtirishning imkoni bormi? Javobingizni tushuntiring.
9. Sonli miqdorni satrli miqdorga o‘tkazib bo‘ladimi? Javobingizni izohlang.

15-§. Qism dasturlar-funksiyalar

Funksiya va protseduralar maxsus tuzilish va nomga ega bo‘lgan dasturning mustaqil qismlari hisoblanishadi. Dastur matnida bu nomlardan foydalanish funksiya va protseduralarga murojaat qilishni anglatadi.

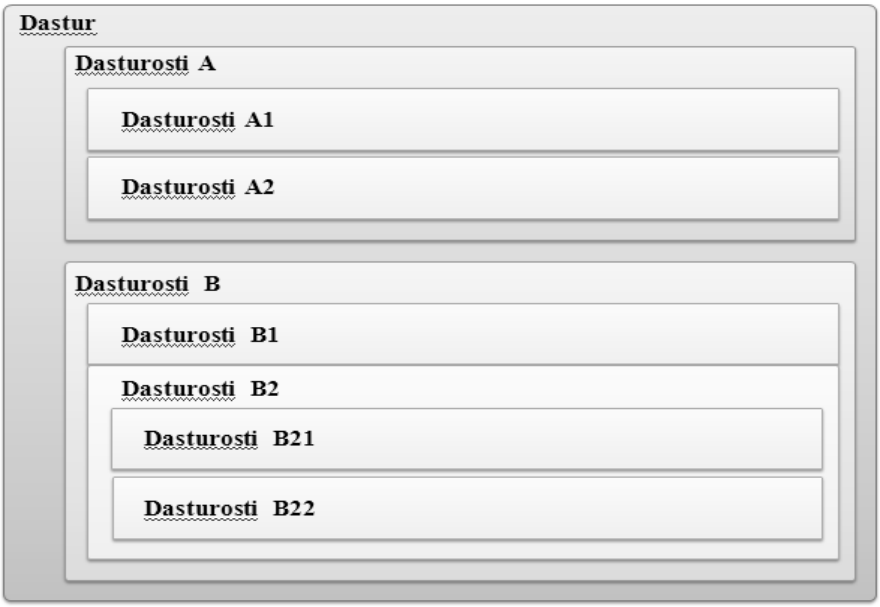


[70]***Function MyFunction : Integer;***
begin
MyFunction:=12; // Return 12
end;

Funksiya va protseduralarni umumiy nom *qism dasturlar* deb yuritishadi. Bu qism dasturlar yordamida har qanday dasturni bir-biriga bog‘liq va bog‘liq bo‘lmagan qimlarga ajratish mumkin. Bu holat dastur xotirani tejash bilan birga, dasturchi uchu qator qulayliklar yaratadi. Har bir qism dasturda bir marotaba uchraydi, ammo undan murojaat qilish usuli bilan bir necha marotaba foydalanisn mumkin. (yaratilgan qism dasturlar yordamida modullar tarkibida foydalanish uyqoru samaralar berishi bilan k…….).

Dastur tarkibida qimdasturlar joylashishuni qyuidagicha tasvir-lash mumkin:

[70] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp-185-186.



Funksiya bu dasturning bir qismi bo‘lib, *function* so‘zi bilan boshlanib, quyidagi tuzilishga ega:

function **<funksiyaning nomi> (<formal parametrlar ro‘yxati>): <tip>;**

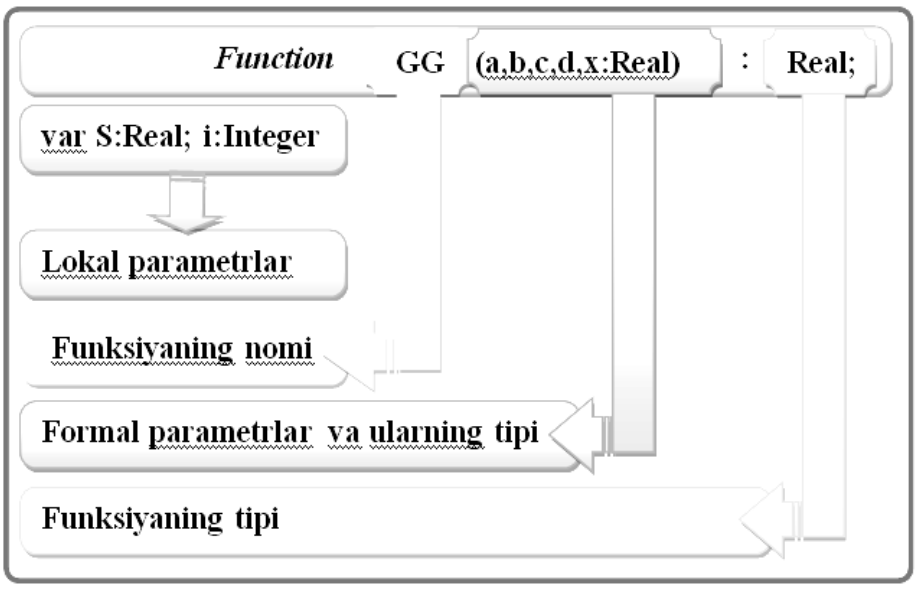
bu yerda tip funksiya qiymatining tipi.

Bu tip tartibli, haqiqiy, String va Point tipida bo‘lishi mumkin. Formal parametrlar quyidagi tuzilishga ega:

<Bir nechta tipga tegishli parametrlar>: <tipning nomi>;

Misollar.

1)



2) *function* RR1(x1,y1,x2,y2: byte): Real;

Dasturda funksiyaga murojaat qilish haqiqiy parametrlar yordamida amalga oshiriladi, masalan

GG (6,5,6,-1,0) yoki RR1(a,b,d,e) va h.

3) *function* Aro(s: String): String;

4) *function* NNC(L: Word; c: Char): String;

Funksiya – dasturning bir qismi bo‘lib, unga dasturda bir necha marotaba murojaat qilinishi mumkin. Umuman:

a) agarda parametrlar qiymatlari yordamida yagona natija olinadigan bo‘lsa, funksiyadan foydalanish mumkin;

b) qism dasturning yakuniy natijasi albatta funksiya nomiga o‘zlashtirilishi shart.

Quyida ikkita har xil sintaksisdan foydalanilgan bir xil funksiya keltirilgan:[71]

```
procedure Hello;
begin
  writeln ('Hello world!');
end;

function Double (Value: Integer) : Integer;
begin
  Double := Value * 2;
end;

// or, as an alternative
function Double2 (Value: Integer) : Integer;
begin
  Result := Value * 2;
end;
```

1-masala.

Funksiya yordamida to‘rtburchak tomonlariga ko‘ra uning perimetri va yuzasini hisoblovchi dastur tuzing.

Program rr;	*dastur nomi*
Var d,h:integer;	
Function dper (a,b:integer):integer;	*funksiya nomi, formal parametrlar va funksiyaning tipi*
Begin	
dper:=a+a+b+b;	*—funksiyaning yakuniy natijasi uning nomiga o‘zlashtiriladi*
end;	
Function ss(a,b:integer):integer;	*— funksiya nomi, formal parametrlar va funksiyaning tipi*

[71] Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. Pp 72.

Begin	
*ss:=a*b;*	*— yakuniy natija ss funksiya nomiga o‘zlashtiriladi*
end;	
Begin	
Readln (d,h);	
Writeln (`to‘g‘ri burchak perimetri`);	
Writeln (dper (d,h));	*—d,h haqiqiy parametrga bog‘liq dper va ss funksiya qiymatlari hisoblanadi*
Writeln (ss(d,h));	
Readln;end.	

Odatda, Pascalda protsedura yoki funksiya parametrlar o‘rnini fiksirlaydi. Siz parametrlarning tipini ko‘rsatasiz, ammo massivda qancha element bo‘lishini bilmaysiz. Quyida ushbu misolni ko‘rib chiqamiz:

```
function Sum (const A: array of Integer): Integer;
var
  I: Integer;
begin
  Result := 0;
  for I := Low(A) to High(A) do
    Result := Result + A[I];
end;
```

2-masala. Funksiya yordamida a, b, d tomonlari berilgan uchburchak yuzasini hisoblovchi dastur tuzing.

```
program ss4;var p:real;
  function geron(a,b,c:real):longint:
   begin
   p:=(a+b+c)/2;
   geron:=sqrt(p*(p-a)*(p-b)*(p-c));
   end;
   var x,y,z :read;
begin
   x:=3; y:=4; z:=5;
   writeln(geron(x,y,z));
```

```
  readln;
end.
```

3-masala. m!-k! –ni hioblovchi dastur tuzing.

```
program funksiya1;
     var f,m,k:integer;
function fact(n:integer):integer;
     var p,i:integer;
    begin p:=1; for i:=2 to n do
     p:=p*i; fact:=p;
    end;
begin   writeln('m!-n! hisoblash uchun ');
     writeln('m va n qiymatini kirit');
     read(m,k); f:=fact(m)-fact(k);
     writeln('f=';f:5); readln;readln;
end.
```

4-masala. Berilgan a,b,d –sonlar uchburchak tomonlari bo'la olishsa, True aksincha False qiymatlarini beruvchi funksiya yarating.

```
function aniqlash(aa,bb,cc:real):boolean;
  begin
  if (aa+bb>cc) and (bb+cc>aa) and (cc+aa>bb) then
   aniqlash:=true else
   aniqlash:=false;
  end;
```

5-masala. Tomonlari a,b,d bo'lgan uchburchakning perimetrini hisoblsh uchun qism-dastur funksiya yarating va undan shu uchburchakning yuzasini hisoblashda foydalaning.

```
var a,b,c:real;
function per(x,y,z:real):real;
begin
per:=x+y+z;
end;
PROCEDURE UZa(x,y,z:real);
 var p,s:real;
begin
```

```
p:=per(x,y,z)/2;
s:=sqrt(p*(p-x)*(p-y)*(p-z));
writeln(s);
end;
begin
uza(3,4,5);
END.
```

6-masala. Ixtiyoriy uchburchak uchun $c^2 = a^2 + b^2 - 2ab \cdot \cos C$, $a^2 = c^2 + b^2 - 2ab \cdot \cos A$, $b^2 = a^2 + c^2 - 2ab \cdot \cos B$, tenglik matematika fanidan o‘rinli ekanligi ma’lum. Siz funksiya yordamida cos C, cos A, cos B qiymatlarini hisoblovchi dastur yarating,

```
var a,b,c:real;
  javob:boolean;
   KosinA,kosinB,kosinC:real;
   label 200,201;
 function aniqlash(aa,bb,cc:real):boolean;
  begin
  if (aa+bb>cc) and (bb+cc>aa) and (cc+aa>bb) then
   aniqlash:=true else
   aniqlash:=false;
  end;
 function burchcos(aa,bb,cc:real):real;
    begin
      burchcos:=(sqr(aa)+sqr(bb)-sqr(cc))/(2*aa*bb);
    end;
 begin
  readln(a,b,c);
  javob:=aniqlash(a,b,c);  writeln(javob);
  if javob=false then goto 200 ;
   KosinC:=burchcos(a,b,c);
    writeln(KosinC);
   KosinA:=burchcos(b,c,a);
    writeln(KosinA);
   KosinB:=burchcos(c,a,b);
    writeln(KosinB); goto 201;
```

```
   200: writeln('bunday uchburchak mavjud emas');
  201: readln;
  end.
```

7-masala. Dastur tarkibidagi qism-dasturlar vazifalarini aniqlang.

```
var m,n:real;
Function Max(a,b:real):real;
 begin
 if a>b  then Max:=a else max:=b;
 end;
 Function Min(a,b:real):real;
 begin
 if a>b  then Min:=a else Min:=b;
 end;

 begin
  readln(m,n);
  Writeln('Max=',max(m,n));
    Writeln('Min=',min(m,n));
  readln;
  end.
```

8-masala. Tarkibida formal parametri satriy kattalikni “teska-rilovchi” funksiyadan foydalanib, kiritilgan ixtiyoriy so‘zni teskari tartibda yozuvchi dastur yarating.

```
var s:string;
   function teskari(t:string):string;
     var w:string; D,i:INTEGER;
    begin  w:=''; D:=LENGTH(T);
     FOR I:=d downTo 1 DO
     w:=w+t[i];
     teskari:=w;
    end;
 begin
    readln(s);
    writeln(teskari(s));
 END.
```

9-masala. 1*2*3*4*..*N- ko‘paytma qiymatini hisoblovchi funksiya yarating.

```
function fakt(n:integer):integer;
  var p,i:INTEGER;
 begin  p:=1;
  FOR i:=1 To n DO
  P:=P*i;
  fakt:=p;
 end;
```

10-masala. Yaratilgan fak-funksiyasi yordamida quyidagi yig‘indini hisoblash dasturini tuzing:

1+1*2+1*2*3+1*2*3*4+1*2*3*4*5

(1!+2!+3!+4!+5!=?)

```
var k,s:integer;
   function fakt(n:integer):integer;
     var p,i:INTEGER;
    begin  p:=1;
     FOR i:=1 To n DO
     P:=P*i;
     fakt:=p;
    end;
  begin
    for k:=1 to 5 do
     s:=s+fakt(k);
     writeln(S);
     readln;
  END.
```

11-masala. x va y- haqiqiy sonlar uchun x^y –qiymatini hisoblovchi dastur tuzing.

```
var a,b:real;
label 5;
function daraja(a,b:real):real;
  begin
   if a>0 then daraja:=exp(b*ln(a)) else
   if a<0 then daraja:=(a/abs(a))*exp(b*ln(abs(a))) else
   if b=0 then daraja:=1 else
        daraja:=0
```

```
end;
 Begin
5: readln(a,b); if (a=0) and (b=0) then goto 5;
 writeln(daraja (a,b));
 readln;
 end.
```

```
PROGRAM HISOBLAGICH ;
VAR FF:text;
 N, P,son: INTEGER;
BEGIN
N:=0; P:=0;
Assign(ff,'c:/musman.txt');
RESET (fF); (*faylni ochish va uning birinchi komponentasini o‘qish*)
WHILE NOT EOF (fF) DO
BEGIN
read(ff, son);
writeln(son);
IF son < 0 THEN N:=N+1 ELSE P:=P+1;
END;
 WRITELN ('musbat sonlar:', P,' ta');
 WRITELN ('manfiy sonlar:', N,' ta');
 readln;
        END.
```

Quyida Funksiyadan foydalanib, massivni ochuvchi dastur tuzamiz:[72] [73]

[72] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 190-191.
[73] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 92.

```
Function Average (Row : Array of integer) : Real;
Var I : longint;
    Temp : Real;
begin
  Temp := Row[0];
  For I := 1 to High(Row) do
    Temp := Temp + Row[i];
  Average := Temp / (High(Row)+1);
end;
```

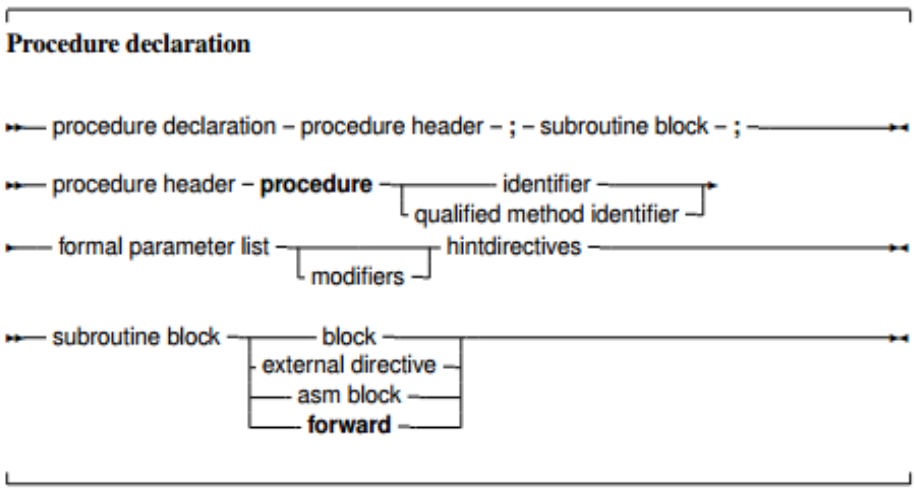
```
function DoubleNumber(x: Integer): Integer;
begin
  Result:= x * 2;
end;

// Main

begin
  Writeln('The double of 5 is : ', DoubleNumber(5));
  Readln;
end.
```

```
function DoubleNumber(x: Integer): Integer;
begin
   Result:= x * 2;
end;

// Main
```

16-§. Qism dasturlar- protseduralar

Funksiyaning natijasi uning nomiga o‘zlashtirilishi shart, ammo qism dasturning natijalari parametrlar ro‘yxatida ham berilishi mumkin. Bu holatda funksiyadan emas, balki Protsedura nomli qism dasturidan foydalaniladi. [74]

[74] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp-184.



Protsedura sarlavhasi quyidagi tuzilishga ega:
Procedure <Protsedura nomi> (<formal parametrlar ro‘yxati >);
Procedure umumiy ko‘rinishi quyidagicha bo‘ladi: [75]

```
procedure name(argument(s): type1, argument(s): type 2, ... );
  < local declarations >
begin
  < procedure body >
end;
```

Protsedura nomi natijani belgilash uchun ishlatilmaydi. Formal parametrlar ro‘yxatida natijalar var xizmatchi so‘zidan keyin tiplari bilan birgalikda beriladi.

Protsedura funksiyaga nisbatan umumiyroq qismdastur bo‘lib, funksiyani Protseduraga aylantirish mumkin va Protseduraning natijaviy parametrlari bir nechta bo‘lishi mumkin (hatto 0 ga teng bo‘lishi ham mumkin).

```
procedure Hello;
begin
  writeln ('Hello world!');
end;

function Double (Value: Integer) : Integer;
begin
  Double := Value * 2;
end;

// or, as an alternative
function Double2 (Value: Integer) : Integer;
begin
  Result := Value * 2;
end;
```

[75] Pascal tutorial. Tutorilaspoint.cm. Pp 62.

```
// call the procedure
Hello;

// call the function
X := Double (100);
Y := Double (X);
writeln (IntToStr (Y));
```
[76]

Quyida dastur kodini o‘zgartirishni keltiramiz:

```
procedure Hello;
begin
  writeln ('Hello world, again!');
end;
```

Protseduraning ko‘rinishlariga misol keltiramiz:

1) Procedure Uch (a,b,c: Real; var ha, hb, hc, ma, mb, mc: real);

2) Procedure Aylana(R: Real; var c,d,s: real);

3) Procedure Max(Var A,B,C: Real);

4) Procedure Beep(H,T: Word).

Protseduraga murojaat qilish quyidagi ko‘rinishga ega:

<Protsedura nomi> (<haqiqiy parametrlar ro‘yxati>)

Misol.

1) Uch(3,5,7); Uch(x,y,z);

2) Aylana (5.1);

Funksiya va Protsedurada formal va haqiqiy parametrlari orasida quyidagi mosliklar bo‘lishi lozim:

a) tiplari mos bo‘lishi, agar parametr tuzilmalar shaklida bo‘lsa unda ham formal va haqiqiy parametrlar tiplari bir xil bo‘lishi shart;

b) ular son jihatdan teng bo‘lishi mumkin;

d) berilish ketma-ketligi mos bo‘lishi.

(1 formal parametr 1-haqiqiy parametr bilan, 2-formal parametr 2-haqiqiy parametr bilan va h.).

1-masala. Protsedura tarkibda umuman parametrdan foydalan-maslik mumkin va shu holat uichun misol keltiramiz:

procedure A;

*begin writeln('**********');end;*

begin A;end.

[76] Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. Pp 72-73.

2-masala. X,y –haqiqiy sonlar. Qism dastur–protsedura yordamida ularga berilgan qiymatlarni almashtiruvchi dastur yarating. (Qiymat berishda x=5 va y=10 ga teng , bo‘sa dastur bajarilishida x=10 va y=5 ga teng bo‘lsin).

```
var x,y:real;
 procedure almashish(xx,yy:real);
  var orkat:real;
 begin
 orkat:=xx;  xx:=yy;  yy:=orkat;  writeln('x=',xx:4:2);  writeln('y=',yy:4:2);
 end;
 begin
 readln(x,y);  writeln('x=',x:4:2);  writeln('y=',y:4:2);  almashish(x,y);
 readln;
  end.
```

3-masala. $Ax^2+bx+c=0$ tenglama yechimlarini aniqlashda protsedura va uning tarkibida funksiyadan foydalaning.

```
var m,n,d:real;
Procedure KVT(a,b,c:real);
 var d,x1,x2,x:real;
 function diskr(bb,aa,cc:real):real;
 begin  diskr:=sqr(bb)-4*aa*cc;  end;
 begin
 d:=diskr(b,a,c);
 if d>0 then
      begin
      x1:=(-b+sqrt(d))/(2*a); writeln('x1=',x1:4:3);
      x2:=(-b-sqrt(d))/(2*a); writeln('x2=',x2:4:3);
      end
     else
      if d=0 then
      begin x:=(-b+sqrt(d))/(2*a); writeln('x=',x:4:3);
      end
     else
      if d<0 then
      begin
```

```
        writeln('yechimga ega emas');
        end;
    end;
  begin
   readln(m,n,d);

   KVT(m,n,d);
   readln;
  end.
```

4-masala. 200 ta aylanani ekranda tarkibida markazi tasodufiy nuqtalarda joylashgan va radiuslari 0..44 qiymatlardan birini qabul qilishni ta’minlovchi protsedura yordamida tasvirlovchi dastur yarating.

```
uses graph;
var i,gd,gm:integer;

PROCEDURE Aylana;
 var p,s:real;
begin
circle(random(getmaxx),random(getmaxy),random(45));
end;
begin
gd:=detect;
initgraph(gd,gm,'');
for i:=1 to 200 do
begin
setcolor(random(2000));
aylana;
end;
readln;
END.
```

5-masala. Protsedura yordamida qo‘shish, ayirish, ko‘paytirish hamda bo‘lish amallarini berilgan a va b –haqiqiy sonlar uchun bajaruvchi dastur yarating.

```
var a,b:real;
procedure Amallar(x,y:real);
    var P,S,F,B:real;
```

```
label 15,20;
begin
S:=x+y; Writeln(x:5:2,' +',y:5:2,'=',x+y:5:2);
F:=x-y; Writeln(x:5:2,' -',y:5:2,'=',x-y:5:2);
P:=x*y; Writeln(x:5:2,' *',y:5:2,'=',x*y:5:2);
if y=0 then goto 15 else
B:=x/y; Writeln(x:5:2,' :',y:5:2,'=',x/y:5:2); goto 20;
15:writeln (x:5:2, '-ni 0 ga bo‘'lish mumkin emas');
20:  end;
Begin
readln(a,b);
amallar(a,b);
readln;
end.
```

6-masala. Protseduradan foydalanib, Oshxona haqidagi masala dasturini tuzish.

```
procedure Menu;
begin
  Writeln('Welcome to Pascal Restaurant. Please select your order');
  Writeln('1 - Chicken      (10$)');
  Writeln('2 - Fish         (7$)');
  Writeln('3 - Meat         (8$)');
  Writeln('4 - Salad        (2$)');
  Writeln('5 - Orange Juice (1$)');
  Writeln('6 - Milk         (1$)');
  Writeln;
end;
```

```
procedure GetOrder(AName: string; Minutes: Integer);
begin
  Writeln('You have ordered : ', AName, ', this will take ',
    Minutes, ' minutes');
end;

// Main application
```

```
var
  Meal: Byte;
begin

  Menu;
  Write('Please enter your selection: ');
  Readln(Meal);

  case Meal of
    1: GetOrder('Chicken', 15);
    2: GetOrder('Fish', 12);
    3: GetOrder('Meat', 18);
    4: GetOrder('Salad', 5);
    5: GetOrder('Orange juice', 2);
    6: GetOrder('Milk', 1);
  else
    Writeln('Wrong entry');
  end;
  Write('Press enter key to close');
  Readln;
end.
```

77

Mustahkamlash uchun savollar

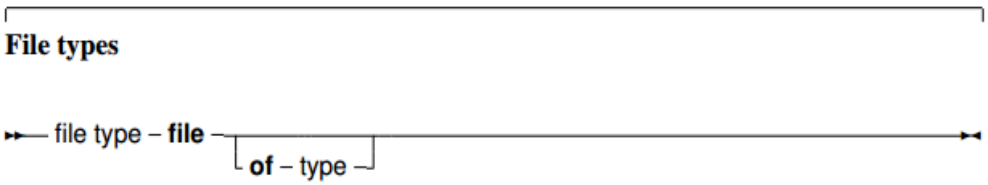
1. Qism dastur - funksiyaning berilishi va funksiyaning nomi.
2. Funksiyaning o‘ziga xos xususiyatlari
3. FUNCTION ning formal va haqiqiy parametrlari.
4. Parametr-qiymat, parametr-o‘zgaruvchi, parametr-doimiylik tushunchalari.
5. Qism dastur protseduraning berilishi va protseduraning nomi.
6. PRECEDURE ning formal va haqiqiy parametrlari.
7. Protseduraning funksiyadan farqi.
8. Protseduraning o‘ziga xos xususiyatlari

17-§. Pascalda fayllar bilan ishlash

Fayl deb, kompyuter tashqi xotirasining nomlangan sohasiga aytiladi va uning uchta xarakterli xususiyatlari mavjud. [78]

[77] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 85.

[78] Michaël Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 48-49



Birinchidan, uning nomi mavjud bo‘lib, bu nomdan dasturda foydalaniladi. Ikkinchidan uning komponentlari bir tipga mansub va bu fayl tipidan boshqa barcha tiplar bo‘lishi mumkin. Uchinchidan yangi yaratiluvchi faylning uzunligi haqida uning e’lon qilish vaqtida “fikr yuritilmaydi” va bu faqat tashqi xotira elementining hajmiga bog‘liq. [79]

```
type
  IntFile = file of Integer;
```

Fayl tipi quyidagi uch yo‘llarning biri bilan yaratiladi:

<nom>=file of<tip>; <nom>=Text; <nom>=File;

Bu yerda <nom>- fayl tipining nomi (to‘g‘ri nomlangan identifikator), file, of xizmatchi so‘zlar, <tip> bu fayl tipidan boshqa barcha tiplar.

Quyidagi deklaratsiya fayllarni yozishni e’lon qiladi:

Type
Point = Record
X,Y,Z : real;
end;
PointFile = File of Point;
Misol: *Type*
Product=record Name:String; Code:Word;End;
Text80=file of String [80];
Var
F1: File of Char; F2: Text; F3: File; F4: Text80;
F5: File of Product;

Fayllarni e’lon qilish usullariga ko‘ra, ularni uch turga ajratish mumkin:

[79] Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008.Pp 58.

tiplashtirilgan fayllar(File of ... bilan beriladi, yuqoridagi misolda,

F1, F4, F5);

matnli fayllar(TextFle tipi bilan aniqlanadi, yuqoridagi misolda, F2);

tiplashtirilmagan fayllar(File tipi bilan beriladi, yuqoridagi misolda, F3);

Quyida fayllarni boshqarish protseduralari jadvalini keltiramiz: [80]

ist below:

Append	FileClose	Flush
AssignFile	FileCreate	GetDir
BlockRead	FileDateToDateTime	IOResult
BlockWrite	FileExists	MkDir
ChangeFileExt	FileGetAttr	Read
CloseFile	FileGetDate	Readln
DateTimeToFileDate	FileOpen	Rename
DeleteFile	FilePos	RenameFile
DiskFree	FileRead	Reset
DiskSize	FileSearch	Rewrite
Eof	FileSeek	RmDir
Eoln	FileSetAttr	Seek
Erase	FileSetDate	SeekEof
ExpandFileName	FileSize	SeekEoln
ExtractFileExt	FileWrite	SetTextBuf
ExtractFileName	FindClose	Truncate
ExtractFilePath	FindFirst	Write
FileAge	FindNext	Writeln

Faylning turi uning saqlanish usulini aniqlaydi va umuman Turbo Paskal tilida oldindan yaratilgan faylni nazorat qilish vositalari mavjud emas va bu vazifani dasturchi o‘z zimmasiga olishi lozim. Fayllar bilan ishlash faqat faylni ochish protsedurasi bajarilgandan so‘ng bajarilishi mumkin. Bu odindan e’lon qilingan fayl o‘zgaruvchisini yaratilgan yoki yaratilishi lozim bo‘lgan fayl nomi bilan bog‘lash protsedurasi bo‘lib, undan so‘ng fayldan o‘qish yoki unga yozish yo‘nalishi beriladi. Har qanday fayllar (yoki mantiqiy qurilmalar) faylni (mantiqiy qurilmani) ochish maxsus protsedurasi yordamida dasturga u bilan ishlash imkoniyatini beradi. Fayl o‘zgaruvchisi avvaldan yaratilgan fayl nomi bilan quyidagi standart protsedura yordamida bog‘lanadi:

Assign (<fayl o‘zgaruvchisi>,<fayl nomi>) protsedurasi fayl o‘zgaruvchisini fayl nomi bilan bog‘laydi.

[80] Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. Pp 129.

AssignFile (<fayl o‘zgaruvchisi >,< fayl nomi >);- bu proseduraning umumiy ko‘rinishi bulib, bu yerda fayl o‘zgaruvchisi - dasturda e’lon qilingan fayl tipidagi o‘zgaruvchi, fayl nomi esa, fayl nomini yoki ungacha bo‘lgan yo‘lni ifodalovchi matn.

Fayl initsializatsiyasi deb, bu faylga ma’lumotlarni jo‘natish yoki undan olish yo‘nalishiga aytiladi.

Faylni o‘qish uchun fayl Reset protsedurasi yordamida initsializatsiya qilinadi va bu protseduraning ko‘rinishi quyidagicha:

Reset (<fayl o‘zgaruvchisi >);

Izoh: fayl o‘zgaruvchisi –avval Assign protsedurasi yordamida mavjud fayl bilan bog‘langan bo‘lishi lozim.

Bu prosedura bajarilishi natijasida fayl o‘qish uchun tayyorlanadi va natijada maxsus ko‘rsatgich bu faylni boshiga, ya’ni 0-tartib nomerli komponentni ko‘rsatib turadi.

Delphi dasturlash tilida Reset protsedurasi yordamida ochilgan tiplashtirilgan fayllarga read protsedurasi bilan murojaat qilish mumkin. Reset protsedurasi yordamida ochilgan matnli fayllar uchun Write yoki Writeln protseduralaridan foydalanib bo‘lmaydi.

Rewrite (< fayl o‘zgaruvchisi >) protsedurasi fayl o‘zgaruvchisi bilan bog‘langan faylga yozish uchun beriladi va bunda mavjud fayldagi barcha ma’lumotlar avval “o‘chiriladi” va yangi ma’lumotlar buyruqlarga ko‘ra faylga yoziladi.

Append (<fayl uzgaruvchisi >) protsedurasi mavjud faylni kengaytirish, ya’ni unga qo‘shimcha ma’lumotlarni yozish imkoniyatini beradi.(faqat Text tipidagi fayllar uchun qo‘llaniladi)

Close (<fayl uzgaruvchisi >) protsedurasi faylni yopish uchun qo‘l-laniladi, ammo fayl o‘zgaruvchisi bilan bog‘lanish o‘z kuchini saqlaydi. [81]

Quyida keltirilgan ikki protseduradan foydalanish uchun Reset, Rewrite, yoki Append protseduralari yordamida ochilgan fayllar avval yopilgan bo‘lishlari shart.

Rename (<fayl uzgaruvchisi >,<yangi nom>) protsedurasi faylni qayta nomlash uchun ishlatiladi.

Erase (<fayl uzgaruvchisi >) fayli o‘chiriladi.

[81] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 50.

Endi yuqoridagi protseduralardan dasturlarda foydalanishga misollar ko‘rib chiqamiz:

1-misol. S-mantiqiy diskda “11.txt” faylini yaratish va unda 'O‘zbekiston ona vatanim' matnini joylashtiruvchi dastur yarating.

```
var f:text;
begin
assign (f,'c:/11.txt');
rewrite(f);
writeln(f,'O‘'zbekiston ona vatanim');
    close(f);
    end.
```

Dastur natijasi(faylga yozish amali bajariladi)::

2-misol. S-mantiqiy diskda joylashgan “11.txt” faylidagi 'O‘zbekiston ona vatanim' matnini “ Toshkent- O‘zbekiston poytaxti” matni bilan to‘ldirish.

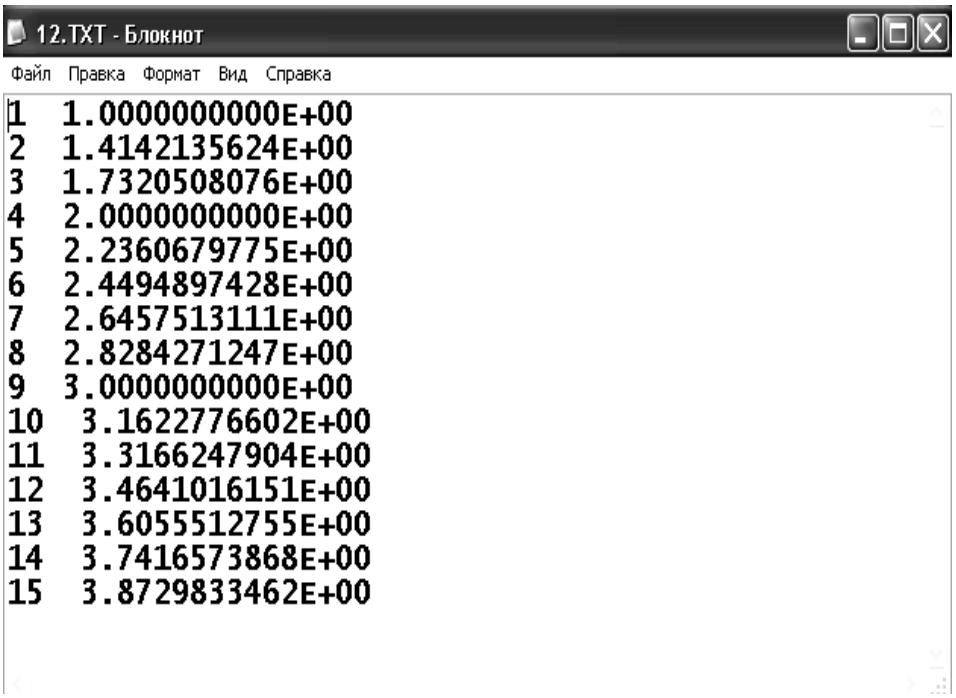
```
var f:text;
begin
assign (f,'c:/11.txt');
rewrite(f);
writeln(f,'O‘'zbekiston ona vatanim');
    close(f);
    end.
```

Dastur natijasi(faylga yozish amali bajariladi):

3-misol.

```
var f:text;
x,i:integer;
begin
assign ( f,'c:/12.txt');
rewrite(f);
for i:=1 to 15 do
begin
  write(f,i);
  write(f,' ');
  write(f,sqrt(i));
  writeln(f);
   end;
    close(f);
    end.
```

Dastur natijasi(faylga yozish amali bajariladi):

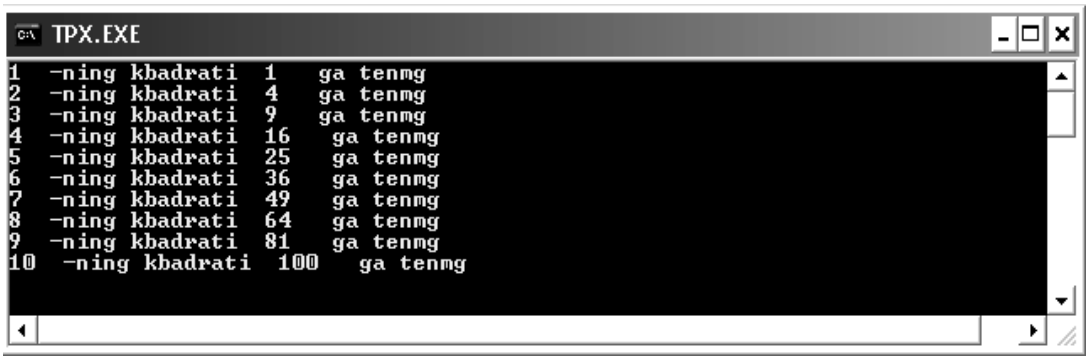


Fayldan o‘qish amali bajarilishini kuzatish uchun avval S mantiqiy diskda (yoki boshqa manbada) o‘qish uchun faylni tayyorlaymiz. Buning uchun «Bloknot» dasturida «15.pas» faylini quyidagi ko‘rinishda tayyorlab olamiz:

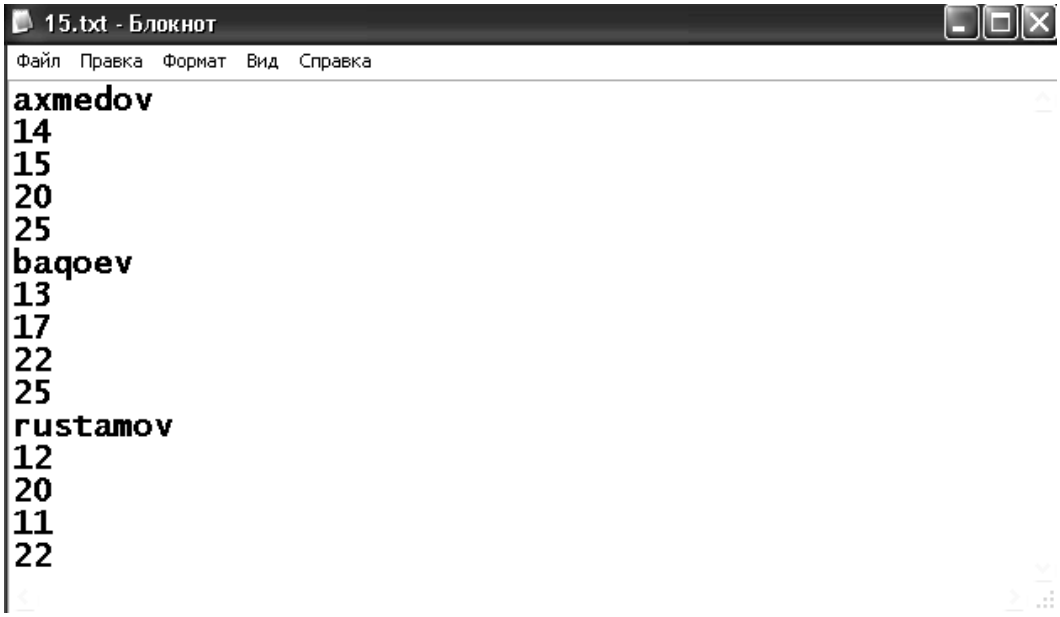
4-misol. S mantiqiy diskdagi «15.pas» faylidan 10 ta soni o‘qib ularning kvadratlarini monitorda aks ettiruvchi dastur yarating.

```
uses crt;
var f:text;
x,i:integer;
begin
assign ( f,'c:/15.txt');
reset(f); clrscr;
for i:=1 to 10 do
begin
  read (f,x);
  writeln(i,'-ning kvadtati ',sqr(x),' ga teng‘);
    end;
   close(f);
   end.
```

Dastur natijasi:



Fayldan o‘qishda faylning oxirini aniqlovchi EOF(f) mantiqiy funksiyasidan foydalanishga misol keltiramiz. Buning uchun siz quyidagi kabi sinfdoshlaringizni 4 fandan olgan test natijalarini aks ettiruvchi «15.txt» hujjatni yaratib kerakli joyda saqlashingiz zarur.



5-misol. O‘quvchilarning 4 fandan olgan natijalarini fayldan o‘qib, har bir o‘quvchining familiyasi va ballari yig‘indisini ekranda chop etuvchi dastur yarating.

Dastur ko‘rinishi:

```
uses crt;
var f:text;
fam:string;
a,b,c,d:integer;
begin
assign (f,'c:/15.txt');
reset(f); clrscr;
while not eof(f) do
begin
  readln (f,fam, a,b,c,d);
  writeln(fam,a+b+c+d);
    end;
   close(f);
   end.
```

(dasturni mustaqil bajarib, xulosa chiqaring)

6-misol. Baho dasturi. [82]

[82] Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 55.

Marks program

```
var
  F: file of Byte;
  Mark: Byte;
begin
  AssignFile(F, 'marks.dat');
  Rewrite(F); // Create file
  Writeln('Please input students marks, write 0 to exit');

  repeat
    Write('Input a mark: ');
    Readln(Mark);
    if Mark <> 0 then  // Don't write 0 value
      Write(F, Mark);
  until Mark = 0;
  CloseFile(F);

  Write('Press enter key to close..');
  Readln;
end.
```

Mustahkamlash uchun savollar:

1. Fayl turdagi o‘zgaruvchi deganda nimani tushinasiz?
2. Paskalda matnli faylni ifodalovchi xizmatchi so‘zni ayting.
3. Assign operatori vazifasini aytib bering.
4. Rewrite operatori vazifasini aytib bering.
5. Rewrite operatori bilan ochilayotgan fayl tashqi xotirada avvaldan mavjud bo‘lsa qanday hodisa yuz beradi?
6. Close operator nima uchun qo‘llaniladi?
7. Append protsedurasi vazifasini aytib bering.
8. Fayldagi ma’lumotlarni ochish uchun qaysi operator yordamida ochiladi.
9. EOF funksiyasini vazifasini aytib bering.

18-§. Grafika moduli va ular bilan ishlash

Turbo Pascal dasturini ishga tushirish bilan ekran matnli holatda bo‘ladi va agarda yaratilayotgan dasturda grafikli obyektlardan foydalanish nazarda tutilsa displey adapterini xrafikali holatga moslashish zarur bo‘ladi. Turbo Pascal dasturlash tilining 4,0

versiyasidan boshlab uning paketi tarkibiga grafikli obyektlar bilan ishlovchi qism dasturlarni qamrab olgan **Graph moduli** qo‘shilgan. Bu modulning qism dasturlaridan foydalanish uchun kerakli drayverni (daryver- maxsus dastur bo‘lib, kompyuterga ulangan texnik qurilmalar faoliyatini boshqarishga mo‘ljallangan) dastur bajarilishi uchun jalb etish zarur. Grafikali drayver displey grafikali holatda displey adapterini boshqaradi. Borland firmasi barcha tipdagi adapterlar uchun bunday drayverlarni ishlab chiqqan. odatda bu drayverlar dastur paketining BGI (Borland Graphics Interfase) nomli papkasida joylashgan bo‘ladi.

Grafikli holatda ekranni bir-biriga juda yaqin joylashgan nuq-talar kabi tasavvur etish mumkin va bu nuqtalar turli ranglarda «yoritilib» tasvir hosil bo‘lishiga xizmat qilishadi. Ayrim adapter-larning grafikali holatlari bilan tanishamiz:

SGA adapteri (Color Graphics Adapter-rangli grafikali adapter). Bu adapter 5 ta grafik holatni qo‘llab-quvvatlaydi. Ularning 4 tasi ekranning quyi imkoniyatlari (gorizontal yo‘nalish bo‘yicha 320 ta piksel (nuqta), vertikal yo‘nalish bo‘yicha 200 ta piksel (nuqta) joylashgan holat)ga mos keladi va ular bir-biridan palitra-ranglar to‘plami bilan farqlanadilar. Har bir palitra 3 xil ranglardan iborat. 5 chi holat yuqori imkoniyatli bo‘lib ekran yechimi 640x200 ga teng.

EGA adapteri (Enhanced Graphics Adapter-takomillashtirilgan grafikali adapter). U SGA adapterini takrorlash bilan birga quyi imkoniyatli yechimni (640x200, 16 xil rang) va yuqori imkoniyatli yechini(650x480, 16 xil rang) qo‘llash imkoniyatini beradi.

MCGA adapteri (Multi-Color Graphics Adapter- rang-barang grafikali adapter). U SGA adapterini takrorlash bilan birga quyi

imkoniyatli yechimni (640x480, 2 xil rang) qo‘llash imkoniyatini beradi.

VGA adapteri (Video Graphics Array grafikali videomassiv). U SGA va EGA adapterlarini takrorlash bilan birga ularni yuqori imkoniyatli yechim (640x480, 16 xil rang) bilan to‘ldiradi.

Bundan tashqari SVGA, EGAVGA kabi qator adapterlar va ularning drayverlari mavjud bo‘lib, siz ular bilan kelgusi faolyatlaringizda batafsil tanishasiz.

Pascal dasturlash tili Graph modulining ayrim qism dasturlari bilan tanishamiz.

InitGgraph-protselurasi adapterning grafikali holatga o‘tishini ta’minlaydi. Uning sarlavhasi quyidagicha:

procedure InitGraph (var Driver, mode: integer;Path:string);

Bu yerda: Driver- integer tipidagi o‘zgaruvchi bo‘lib, grafikali drayver tipini aniqlaydi, mode- ham integer tipli o‘zgaruvchi bo‘lib, grafikali drayverning ishchi holatini aniqlaydi. Path-string tipli o‘zgaruvchi bo‘lib, drayver fayli joylashgan manzil-yo‘lni aniqlash uchun mo‘ljallangan ifodadan iborat. Bu protseduradan foydalanish uchun biror axborot tashuvchi diskda albatta grafikali drayver fayli joylashgan bo‘lishi shart. Dastur bajarilish jarayonida bu drayver protsedura yordamida operativ xotiraga yuklanadi va adapterning grafikali holatga o‘tishini ta’minlaydi. Drayver tipi adapter tipi bilan mos kelishi shart. Graph modulida drayver tipini ko‘rsatish uchun quyidagi doimiyliklar aniqlangan:

```
Const
detect          =0;  {tipni avtomatik aniqlash}
CGA       =1;
MCGA            =2;
EGA       =3;
EGA64           =4;
…
```

Juda ko‘p adapterlar turli holatlarda ishlay oladilar va bu holatlar mode- o‘zgaruvchisi yordamida aniqlanadi.

Protseduradan foydalanishga misol keltiramiz:

CGA.BGI drayveri S diskning TP\BGI papkasida joylashgan bo‘lib, ekranning 320x200 yechimli holatidan foydalanish nazarda

tutilgan bo‘lsa protseduraga murojaat qilishni quyidagi tartibda amalga oshiramiz:

```
 uses Graph;
var
driver,mode:integer;
begin
driver:=CGA;
mode :=CGAC2;
InitGraph (driver, mode,'c:\tp\BGI');
end.
```

Agar kompyuterning adapter tipi noma’lum bo‘lsa yoki yaratilayotgan dastur ixtiyoriy tipdagi adapter uchun mo‘ljallangan bo‘lsa, protseduraga murojaat qilishda drayver tipini avtomatik aniqlashni talab qilish mumkin, ya’ni bu holda

```
driver:=detect;
InitGraph (driver,mode,'c:\tp\BGI');
```

buyruqlaridan foydalanish mumkin. Bunda bir necha grafikali holat ishlovchi adapterlar uchun mode-ning maksimal qiymati avtomatik tanlanadi.

GraphResult funksiyasi (grafikali protseduralarga murojaat qilinganda) integer tipidagi qiymatni qaytaradi: 0-qiymatda xatolik yo‘qligi va -14 dan -1 gacha bo‘lgan qiymatlarda xatoliklarga yo‘l qo‘yilgani haqida «xabar» beradi.

GrapErrorMsg funksiyasi String tipidagi ma’lumotni-xatolikni berish uchun foydalaniladi.

GetDriverName funksiyasi String tipidagi ma’lumotni-yuklangan grafikali drayver nomini aniqlashda foydalaniladi.

GetModName funksiyasi String tipidagi ma’lumotni- ekranning yechim holatlari (gorizontal va vertikal yo‘nalishlar bo‘yicha piksellar miqdorini) va adapterning ish holati nomini parametr bo‘yicha aniqlash imkonini beradi.

GetMaxMode funksiyasi integer tipidagi kattalik bo‘lib, adapterning grafikali holatlarining maksimal miqdorini aniqlaydi.

CloseGraph protsedurasi adapterning grafik holatdagi ish faoliyatini tugatib ekranning matnli holatini tiklaydi.

RestoreCRTMode qisqa muddatga ekranning matnli holatga qaytishini ta’minlaydi. CloseGraph protsedurasildan farqli

o‘rnatilgan grafikali holat parametrlari saqlanashini ta’minlaydi, ya’ni grafik drayver joylashgan xotiraning qismi tozalanmaydi.

GetGraphMode funksiyasi integer tipidagi qiymatni beradi va bu qiymat grafik adpterning o‘rnatilgan grafikali holatini aniqlaydi.

SetGraphMode protsedurasi adapterning yangi grafikali holatini belgilaydi.

Quyida yuqorida keltirilgan funksiya va protseduralardan foydalangan holda Turbo Pascal dasturlash tilida ekranni grafik holatga o‘tkazish va kerakli hollarda uni matnli holatga vaqtincha o‘tkazuvchi dasturni keltiramiz:

```
uses Graph;
var
Driver, Mode,Error:integer;
begin
Driver:= Detect;
InitGraph(Driver,Mode, 'c:\tp\bgi');
Error:=Graphresult;
if Error<>grOk then
Writeln(graphErrorMsg(Error))
else
begin
Writeln('Bu grafikali holat');
Writeln('"Enter"-ni bosing...':20);
Readln;
RestoreCRTMode;
Writeln('Bu matnli holat...');
Readln;
SetGraphMode(GetGraphMode);
Writeln('bu yana grafikali holat...');
Readln;
CloseGraph
end
end.
```

Dasturda keltirilgan «grok»(qiymati 0 ga teng bo‘lgan doimiylik) Graphresult funksiyasini natijasi bo‘lib, protseduralarga murojaat qilishda xatolikga yo‘l qo‘yilmaganligini tekshirishda foydalaniladi.

1-masala. Dastur tarkibida GetDriverName funksiyasidan foydalanib, yuklangan grafikali drayver nomini aniqlang.

```
uses Graph;
var
Driver, Mode,Error:integer;  dr:string;
begin
Driver:= Detect;
InitGraph(Driver,Mode, 'c:\tp\bgi');
dr:=GetdriverName;
Writeln(dr,' ');
Writeln('"Enter" tugmasini bosing...':50);
Readln;
CloseGraph ;
end.
```

Dastur natijasi yuklangan grafikali drayverga bog‘liq va u quyidagicha bo‘lishi mumkin:

2-masala. GetMaxMode funksiyasidan foydalanib, adapterning grafikali holatlarining maksimal miqdorini aniqlovchi dastur yarating.

```
uses Graph;
var
Driver, Mode,Error:integer;  d:integer;
begin
Driver:= Detect;
InitGraph(Driver,Mode, 'c:\tp\bgi');
d:=GetMaxMode;
restorecrtMode;
Writeln(d);
Writeln('"Enter" tugmasini bosing...':50);
Readln;
CloseGraph ;
end.
```

Dastur natijasi:

3-masala. GetModName funksiyasi yordamida ekranning yechim holatlarini va adapterning ish holati nomini parametr bo‘yicha aniqlang.

Dastur ko‘rinishi:

```
uses Graph;
var
Driver, Mode,Error,i:integer;  ds:string;
begin
Driver:= Detect;
InitGraph(Driver,Mode, 'c:\tp\bgi');
writeln (getdrivername);
for i:=0 to 3 do
begin
ds:=GetModeName(i);
Writeln(ds);
end;
Writeln('"Enter" tugmasini bosing...':50);
Readln;
CloseGraph;
end.
```

Koordinatalar, oynalar va sahifalar.

Turbo Pascalning juda ko‘p grafikali protseduralari va funksiyalari ekranda ko‘rinib turuvchi kursordan farqli bo‘lgan “ko‘zga tashlanmas” ikkinchi kursor-ko‘rsatgichdan foydalanishadi. Bu kursorning joylashishi odatda (0,0) koordinatali nuqtaga nisbatan belgilanadi. Umuman, chiziladigan sath - maydon alohida nuqtalar-piksellardan tashkil topgan. Nuqtaning o‘rni uning gorizontal (X) va vertikal (U) koordinatalari bilan belgilanadi. Yuqori chap nuqtaning koordinatalari (0,0) dan iborat. Koordinatalarning qiymatlari vertikal yo‘nalish bo‘yicha tepadan pastga va gorizontal yo‘nalish

bo‘yicha chapdan o‘ngga ortib boradi. GetMaxX va GetMaxY funksiyalari qiymatlari Word tipiga mansub bo‘lib, ekranning joriy ish holatdagi koordinatalaring maksimal qiymatini aniqlashda foydalaniladi.

(0,0)  (GetMaxX, GetMaxY)

Kursor-ko‘rsatgich o‘rnini integer tipidagi qiymatlarni beruvchi GetX va Gety funksiyalari aniqlash imkonini beradilar.

SetViewPort(x1,y1,x2,y2:integer;Clipon:Boolean)-protsedurasi ekranda yuqori chap nuqtasi koordinatalar (x1,y1) va quyi o‘ng nuqtasi koordinatalari (x2,y2) bo‘lgan to‘g‘ri to‘rtburchak shakldagi oynani yaratishda qo‘llaniladi. Boolean tipidagi ifoda qiymatiga ko‘ra mazkur oynaga sig‘mayotgan tasvir elementlarini «qirqish» uchun beriladi. Bu ifodaning True qiymatida mazkur protsedura yordamida aniqlangan sohaga tasvir elementlari sig‘masa, ular qirqiladilar, aksincha False qiymatida soha chegaralari «bekor» qilinadi.

MoveTo(x,y:integer)-protsedurasi ekranda ko‘rsatgichni koordinatalar (x,y) bo‘lgan nuqtaga joylashtiradi va bu joylashtirish oyna o‘rnatilmagan holda (0,0) nuqtaga nisbatan amalga oshiriladi.

MoveRel(dx,dy:integer) - protsedurasi ekranda ko‘rsatgichni yangi nuqtaga joriy koordinatalariga nisbatan dx,dy-orttirmalar bo‘yicha joylashtirish imkonini beradi. SetbkColor(n)- protsedurasi ekran fonini ko‘rsatilgan n-parametrga mos keluvchi (0 dan 15 gacha) rang bilan berilishini ta’minlaydi. ClearDevice- protsedurasi grafik ekranni tozalab, uning fonini SetbkColor bilan aniqlangan rang bilan «bo‘yalishi»ga xizmat qiladi. ClearDevice- protsedurasi yaratilgan grafik oynani, agar u «to‘liq ekran» holatida aniqlanmagan bo‘lsa tozalaydi va ko‘rsatgich oynaning chap yuqori nuqtasiga joylashadi.

Savol va topshiriqlar.

1. Paskalda Graph moduli qanday maqsadda qo‘llaniladi?
2. Grafikli drayver nima va u qanday vaifani bajaradi?
3. Ekranni grafik holatga o‘tkazish uchun Pascalda qanday ko‘rsatma beriladi?
4. Grafik holatdan chiqish uchun qaysi operatordan foydalaniladi?
5. Ekranni grafik holatga o‘tkazilganda yurgichning ko‘rinishi qanday bo‘ladi?
6. Grafik holatda ekranning qaysi nuqtasi joriy nuqta bo‘ladi?
7. GetMaxX va GetMaxY funksiyalarini vazifalarini aytib bering.

19-§. Paskal tilida sodda shakllar chizish

Grafikali muhitda obyektlarni chizish uchun sodda shakllarni chizuvchi protseduralar va ulardan foydalanish uchun funksiyalardan foydalanish zarur. Quyida ularni keltirib o‘tamiz:

PutPixel(x.y:integer;color:word)–protsedurasi ko‘rsatilgan color rangli (x.y) koordinatali nuqtani ekranda tasvirlaydi.

Masalan: PutPixel(100,100,red) - protsedurasi ekranda (100, 100)-koordinatali nuqtada qizil rangli nuqta tasvirini yaratadi.

Bu protseduradan dastur tarkibida foydalanib, samarali “effektlar” olishimiz mumkin.

Masalan, quyidagi dastur yordamida ekranda

Setviewport (20,20,200,200,true) protsedurasi bilan belgilangan oynada 32000 ta nuqtalarni

putpixel(random(getmaxx), random(getmaxy), random(255));

protsedurasi yorlamida turli ranglar bilan bo‘yab ajoyib tasvir yaratishimiz mumkin.

Dastur ko‘rinishi:

```
uses Graph;
var
Driver, Mode,d,m:integer;
begin
Driver:= Detect;
```

```
InitGraph(Driver,Mode, 'c:\tp\bgi');
setbkcolor(black);
Setviewport(20,20,200,200,true);
 for d:=1 to 32000 do
 putpixel(random(getmaxx),random(getmaxy),random(255));
  readln;
CloseGraph ;
end.
```

GetPixel(x,y:integer):***word*** funksiyasi (x,y) koordinatali nuqtaning rangini aniqlashda qo‘llaniladi. Bu funksiyadan foydalanishga misol keltiramiz:

```
uses graph;
var
gd,gm:integer; a:word;
begin
gd:=detect;                                    initgraph(gd,gm,'');
a:=getpixel(25,25);writeln(a);readln;
end.
```

Line(x1,y1,x2,y2:integer)-protsedurasi (x1,y1) va (x2,y2) koordinatali nuqtalarni tutashtiruvchi to‘g‘ri chiziq chizish uchun ishlatiladi.

Misol:

```
uses graph;
var
gd,gm, x1,x2,x3,y1,y2,y3:integer;
begin
gd:=detect; initgraph(gd,gm,'');
x1:=45;y1:=45;x2:=160;y2:=120;x3:=100;y2:=80;
line(x1,y1,x2,y2); line(x3,y3,x2,y2);  line(x1,y1,x3,y3);readln;
end.
```

misol: line(10,100,340,100);

LineTo(x,y:integer)-protsedurasi ko‘rsatgich turgan nuqtadan koordinatalri (x,y) bo‘lgan nuqtagacha to‘gri chiziq chizilishini ta’minlaydi. Bu protseduradan foydalangandan so‘ng ko‘rsatgich koordinatasi shu nuqtaga ko‘chib o‘tadi.

misol:

```
uses graph;
```

```
var gd,gm, i:integer;
begin gd:=detect;initgraph(gd,gm,'');
for i:=1 to 20 do
lineto(random(getmaxx),random(getmaxy));readln;
end.
```

LineRel(dx,dy:integer)-protsedurasi ko‘rsatgich turgan nuqtaga nisbatan koordinatalari mos ravishda (dx,dy) qiymatlarga orttirilgan nuqtagacha to‘g‘ri chiziq chizilishini ta’minlaydi.

```
uses graph;
var gd,gm:integer;
begin
gd:=detect; initgraph(gd,gm,'');
moveto(100,100); Linerel(100,40);readln;
end.
```

SetLineSstyle (type, Pattern,Thick:word)-protsedurasi chizilishi kerak bo‘lgan chiziqning shaklini o‘rnatish uchun qo‘llaniladi.

Bu yerda:

Type- chiziqning shaklini

Pattern- chiziqning namunasi

Thick- chiziqning qalinligini aniqlovchi parametrlar.

Chiziqning tipini quyidagi doimiylar orqali berish mumkin:

```
const
        Solidln    =0; {uzluksiz chiziq}
        Dotteln    =1; {nuqtali chiziq}
        Centerln   =2; {shtrix-punktirli chiziq}
        Dashedln   =3; {punktirli chiziq}
        Userbtln   =4; {chiziq ko‘rinishi foydalanuvchi tomonidan belgilanadi}
```

Pattern-parametri foydalanuvchi tomonidan belgilanadigan chiziqlar uchun aniqlanadi. Pattern parametri uzunligi 16 piksel bo‘lgan kesmani aniqlaydi. Masalan, bu parametr qiymati 100 ga teng bo‘lsa, 16 pikselning shu qismi ”yoritiladi” qolgan qismi fon rangi bilan ifodadanadi. dastur chizilishi zarur bo‘lgan chiziqni ana shu 16 pikselda ketm-ket tasvirlaydi.

Thick-parametri quyidagi qiymatlarni qabul qiladi:

```
const
```

NormWidth =1; {chiziq qalinligi 1 pixel}
ThickWidth =3; {chiziq qalinligi 3 pixel}

dastur tarkibida kattaliklar-doimiyliklar nomini o‘zgartirishimiz mumkinligini hisobga olgan holda, SetLineSstyle protsedurasidan foydalanishga misol keltiramiz:

Misol:

```
uses graph;
const uzluksiz=0; nuqtali=1;Shtrix_punktir=2; punktir=3; foychiz=4;
chizqal=1;
var
gd,gm, i:integer;
begin
gd:=detect;initgraph(gd,gm,'');
setlineStyle(nuqtali,0,chizqal);moveto(100,100);Lineto(400,100);
setlineStyle(uzluksiz,0,chizqal);moveto(100,110);Lineto(400,110);
setlineStyle(Shtrix_punktir,0,chizqal);moveto(100,120);Lineto(400,120);
setlineStyle
(punktir,0,chizqal);moveto(100,130);Lineto(400,130);
setlineStyle
(foychiz,80,chizqal);moveto(100,140);Lineto(400,140);
readln;
end.
```

Izoh: doimiyliklar nomi o‘rniga ularning qiymatidan ham foydalanishimiz mumkin.

SetLineStyle- protsedurasi yordamida o‘rnatilgan chiziqning shakli yordamida to‘rtburchak, ko‘pburchak va boshqa shakllarni chizishda foydalanish mumkin.

rectangle(x1,y1,x2,y2:integer);-protsedurasi chap yuqori burchagi (x1,y1) va o‘ng quyi burchagi (x2,y2) nuqtalarda joylashgan to‘gri to‘rtburchak chizish uchun ishlatiladi.

masalan:

```
uses graph;
var gd,gm:integer;
begin
gd:=detect;
```

```
initgraph(gd,gm,'c:\tp\bgi');
on rangi faqat “qora” rangda emas,
 rectangle(5,5,getmaxx-5,getmaxy-5);
 readln;
 end.
```

SetColor (color:word);- protsedurasi chizilishi va chop etilishi zarur bo‘lgan chiziqlar va simvollar rangini belgilaydi. Graph modulida ranglarni ifodalash uchun Srt modulidagi doimiyliklardan foydalaniladi.

GetColor:word - funksiyasidan joriy rang kodini aniqlashda foydalaniladi.

GetMaxColor:word - funksiyasi SetColor protsedurasida foydalanishi mumkin bo‘lgan maksimal kodni aniqlashda ishlatiladi.

Matnli holatdan farqli f ixtiyoriy bo‘la oladi. Grafikali ekranda rangni aniqlash orqali butun ekran rangini o‘zgartirish mumkin, ya’ni bu ekranni turli sohalarini turli ranglar bilan “bo‘yash”ning iloji yo‘q.

Circle(x,y:integer;r:word); - protsedurasi markazi (x,y) va radiusi *r* – piksel bo‘lgan aylana chizish ishlatiladi. Bu protseduradan foydalanishda chiziq qalinligini parametr orqali o‘zgartirish mumkin, ammo chiziq shakli faqat uzluksiz chiziqdan iborat bo‘ladi.

Masalan:

```
...
setlinestyle(1,0,3);
  circle(50,50,45);
      .....
```

Yuqoridagi protseduralar yordamida ekranda markaz koordinatalari va radiuslari tasodufiy qiymatga ega bo‘lgan 120 ta aylana chizish dasturini keltiramiz:

```
uses graph,crt;
var gd,gm, i:integer;
begin
gd:=detect;initgraph(gd,gm,'');
for i:=1 to 120 do
begin
setcolor(random(getmaxcolor));
circle(random(getmaxx),random(getmaxy),random(100));
```

end;

end.

Arc *(x,y:integer;BegA,EndA,r:word);* - protsedurasi markazi (x,y) va radiusi *r* –piksel bo‘lgan aylana yoyining *BegA va EndA* burchaklariga mos keluvchi qismini chizish uchun ishlatiladi. *BegA va EndA* graduslarda beriladi va soat mili yo‘nalishiga qarshi ravishda hisobga oldinadilar.

Masalan:

arc(200,200,90,180,35);

ellipse(*x,y:integer; BegA, EndA, RX,Ryr:word);* - protsedurasi markazi (x,y) va gorizontal radiusi RX, vertikal radiusi Ry – piksel bo‘lgan ellips yoyining *BegA va EndA* burchaklariga mos keluvchi qismini chizish uchun shlatiladi.

Masalan:

ellipse(100,100,150,60,145,100);

SetFillStyle(fill,color:word);-protsedurasi biror sohani to‘ldirishning stili va rangini belgilaydi. To‘ldirish tipi uchun quyidagi kattaliklardan foydalaniladi:

const

EmptyFill=0;{fon rangi bilan to‘ldirish}

SolidFill=1;{yaxlit-uzluksiz to‘ldirish }

LineFill=2;{- - - - - simvollari bilan to‘ldirish}

LtslashFill=3;{ /////// simvollari bilan to‘ldirish}

SlashFill=4;{ //// qalin simollari bilan to‘ldirish}

BkSlashFill=5;{\\\\\ qalin simvollari bilan to‘ldirish }

LtBkSlashFill=6;{ //// simvollari bilan to‘ldirish }

HatchFill=7;{+++++ simvollari bilan to‘ldirish }

XHatchFill=8;{ *xxxx* simvollari bilan to‘ldirish }

InterLeaveFill=9;{kataklar bilan to‘ldirish}

WideDotFill=10;{kam sonli-zich bo‘lmagan nuqtalar bilan to‘ldirish }

CloseDotFill=11;{zich bo‘lgan nuqtalar bilan to‘ldirish }

UserFill=12;{foydalanuvchi tomonidan aniqlangan shakl bilan to‘ldirish}

floodfill(x,u;integer; Border:word);-protsedurasi ixtiyoriy chegaralangan sohani oldindan aniqlangan sohani to‘ldirishning stili va rangi bilan to‘ldiradi.

masalan: parametrlari (150,150,getmaxx-150,getmaxy-150) bo‘lgan to‘rtburchak sohani turli stillarda to‘ldiruvchi dasturni keltiramiz:

```
uses graph;
var gd,gm:integer; i:byte;
begin
gd:=detect; initgraph(gd,gm,'c:\tp\bgi');setbkcolor(black);
setcolor(red);setlinestyle(0,0,1);
for i:=0 to 12 do
 begin
      setfillstyle(i,green);rectangle(150,150,getmaxx-
150,getmaxy-150);
floodfill(160,160,red);
readln;
 end;
 readln;
 end.
```

Bar *(x1,y1,x2,y2:integer);*-protsedurasi chap yuqori burchagi (x1,y1) va o‘ng quyi burchagi (x2,y2) nuqtalarda joylashgan to‘g‘ri to‘rtburchakli ekran sohasini to‘ldiradi.

masalan:

```
bar(340,125,480,220);
```

Bar va setfillstyle protseduralari yordamida 12 ta to‘g‘ri to‘rt-burchakli sohani turli stillarda to‘ldirib, to‘ldirish stilini namoyish etuvchi dastur yarating.

```
uses graph,crt;
const
stil:array[0..11]of
string=('0','1','2','3','4','5','6','7','8','9','10','11');
var
gd,gm,a,b:integer; i:byte;
begin
gd:=detect;initgraph(gd,gm,'');setbkcolor(black);
a:=5; b:=100;
for i:=0 to 11 do
begin
```

```
setfillstyle(i,4);bar(a,100,a+50,200);outtextxy(a+15,75,stil[i]);
a:=a+50;
end;
readln;
end.
```

Bar3d*(x1,y1,x2,y2,depth:integer;top:boolean);* -protsedurasi oldingi hududi chap yuqori burchagi (x1,y1) va o‘ng quyi burchagi (x2,y2) nuqtalarda joylashgan parallelepiped tasvirini yaratishda ishlatiladi.

bu yerda :

-depth-uch o‘lmali tasvirning uchunchi o‘lchami

-top-parallelepipedning yuqori hududini aks ettirish uchun ishlatiladi, ya’ni “true” qiymtida yuqori hudud aks ettirilsa, aksincha holda u “ko‘rinmaydi”.

Masalan: Bar3d protsedurasi yordamida ekranda parallelepipedning yuqori hududini aks ettirish va aks etirilmasligini namoyish etuvchi dasturni keltiramiz.

```
  uses graph,crt;
var gd,gm:integer;
begin
gd:=detect;initgraph(gd,gm,'');
setbkcolor(black);setcolor (magenta);
 begin
  bar3d(100,100,200,200,25,true);
  bar3d(300,100,400,200,25,false);
 end;
readln;
end.
```

FillEllipse (x,y,Rx,Ry:integer)-markazi (x,y nuqtada bo‘lgan gorizontal radiusi Rx-ga, vertikal radiusi Ry-ga teng ellipsni to‘ldirish uchun ishlatiladi. (Rx va Ry-piksellarda)

Masalan: fillellipse(340,220,100,50);

Sector(x,y:integer;BegA,EndA,Rx,Ry:word)-protsedurasi markazi (x,y) nuqtada bo‘lgan gorizontal radiusi Rx-ga, vertikal radiusi Ry-ga teng ellipsning boshlang‘ich burchagi BegA-ga, oxirgi burchagi EndA ga teng sektorini aniqlangan shakl bilan to‘ldiradi.

Masalan: sector (200,200,180,200,55,300);

PieSlice(x,y:integer;BegA,EndA,R:word)-protsedurasi markazi (x,y) nuqtada bo‘lgan radiusi R-teng aylananing boshlang‘ich burchagi BegA-ga, oxirgi burchagi EndA ga teng sektorni berilgan shakl bilan to‘ldiradi.

Masalan: PieSlice(300,300,145,360,100);

Savol va topshiriqlar.

1. PutPixel operatori haqida so‘zlab bering.
2. Ekranda biror shakl chizish uchun uning rangi qaysi operator yordamida tanlanadi?
3. Paskalda kesma chizish imkoniyatini amalda ko‘rsatib bering.
4. Qaysi protsedura chiziqning shaklini o‘rnatish uchun qo‘llaniladi?
5. *SetLineStyle*- protsedurasi yordamida qanday shakllarni chizishda foydalaniladi?
6. Aylana qaysi operator yordamida chiziladi?
7. SetFillStyle operatorini vazifasini aytib bering.
8. Bar va Bar3D operatorlarini vazifasi nimadan iborat?
9. FillEllipse operatori qanday shakl chizadi?

FOYDALANILGAN ADABIYOTLAR

1. Thomas H. Cormen va b. Intruduction to algorithms. Massachusetts Institute of Technology. London 2009.
2. Robert Sedgewick and Kevin Wayne. Algorithms. FOURTH EDITION. Princeton University. First printing, March 2011.
3. В.Д.Колдаев. Основы алгоритмизации и программирования. Учебное пособие, Москва ИД “Форум”- ИНФРА-М 2006 г.
4. Фаронов В. В. Turbo Pascal. — СПб.: BXB- Санкт-Петербург, 2004. – 1056 с.
5. Слинкин Д.А.Основы программирования на Турбо-Паскаль: Учебно-методическое пособие для студентов вузов. Шадринск: Изд-во Шадринского пединститута, 2003. - 244 s.
6. Michael Van Canneyt. Free Pascal Reference guide. Reference guide for Free Pascal, version 3.0.0. Document version 3.0. November 2015. Pp 11-13, 20-21, 25, 28, 30, 159.
7. Marco Cantu. Essential Pascal. Piacenza, Italy 4th Edition, April 2008. Pp 26, 30-35.
8. Motaz Abdel Azeem. Start Programming using Object Pascal. Edited by: Pat Anderson, Jason Hackney -28 september, 2013y. Pp 18.
9. M.U.Ashurov, N.D.Mirzaxmedova. Turbo Pascal dasturlash tili.(uslubiy qo‘llanma),Toshkent TDPU – 2011.

MUNDARIJA

M.O‘.ASHUROV, SH.A.SATTAROVA,
SH.U.USMONQULOV

ALGORITMLAR

Toshkent – «Fan va texnologiya» – 2018

Muharrir:	F.Ismoilova
Tex. muharrir:	F.Tishabayev
Musavvir:	A.Moydinov
Musahhih:	Sh.Mirqosimova
Kompyuterda sahifalovchi:	N.Raxmatullayeva

E-mail: tipografiyacnt@mail.ru Tel: 245-57-63, 245-61-61.
Nashr.lits. AI№149, 14.08.09. Bosishga ruxsat etildi 16.11.2018.
Bichimi 60x84 $^{1}/_{16}$. «Times New Roman» garniturasi.
Ofset bosma usulida bosildi. Shartli bosma tabog‘i 15,0.
Nashriyot bosma tabog‘i 15,25.
Tiraji 350. Buyurtma № 458.

«Fan va texnologiyalar Markazining bosmaxonasi» da chop etildi.
100066, Toshkent sh., Olmazor ko‘chasi, 171-uy.