O`ZBEKISTON RESPUBLIKASI OLIY VA O`RTA MAXSUS TA`LIM VAZIRLIGI
AJINIYOZ NOMIDAGI NUKUS DAVLAT PEDAGOGIKA INSTITUTI

MATEMATIKA-INFORMATIKA FAKULTETI

Informatika o`qitish metodikasi kafedrasi

ALGORITMLASH

fanidan

Informatika o`qitish metodikasi ta`lim yo'nalishi talabalari uchun

MA'RUZA MATNI

Alaminov M.X

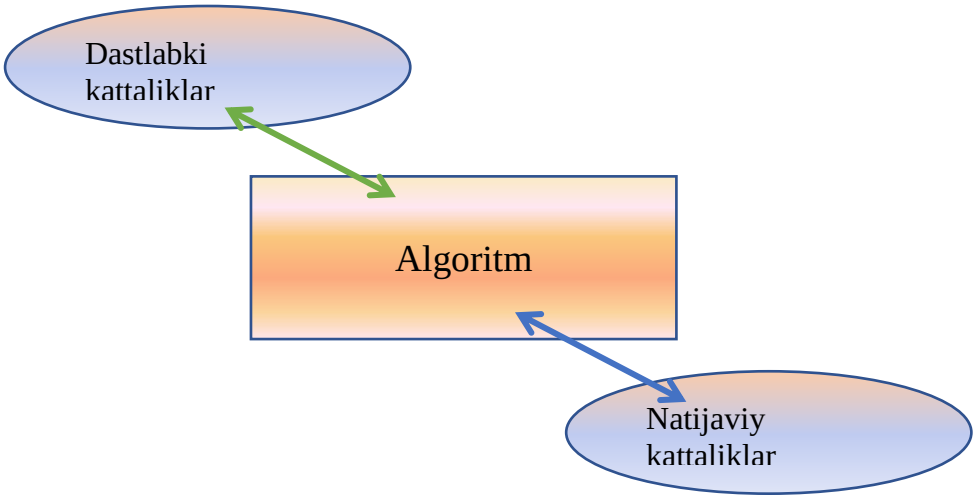
Nukus 2019

1-MAVZU: ALGORITM TUSHUNCHASI VA ULARDAN FOYDALANISH

Reja

1. **Algoritm tushunchasi**
2. **Algoritm xossalari**
3. **Algoritmlarni tasvirlash usullari**

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 ekzemplyari (instanse) deb yuritiladi. Agar algoritm har qanday kiruvchi qiymatlar uchun aniq va mos chiquvchi qiymatlarni bera olsa u aniq (correct) deb yuritiladi. Algorimlardan amaliyotda foydalanishga ayrim misollarni keltiramiz:

- Odam DNK si tarkibidagi 100 ming gen identifikatsiyasi, DNK-ni tashkil etuvchi 3 milliard asosiy juftlikni saralashva tahlili masalasi;
- Internetda ma'lumotlar olish masalasi: kata hajmdagi ma'lumotlarni olish, jo'natish,qidiruv va optimal marshrut tanlash;
- Electron kommertsiya masalalarida(kredit karta nomerlari , parollar, bank xisob-kitob raqamlari himoyasi, raqamli imzo va b);

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 algorutmlari farqini ko'rib chiqamiz:

Saralash algoritmi	Sarflanadigan vaqt	Izoh
Joylashtirish usuli	C_1n^2 bu N^2 -ga propor-tsional	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.

komputerlar	Saralanadigan sonlar soni	Saralovchi algoritm	Talab qilinadigan vaqt
A(tez ishlovchi 1sekundda 10mlrd amal bajaradi)	10 mlnta(taqriban80 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 ishlovch-1sekundda 10 mln amal bajaradi)		Qo'shish usuli (o'rta darajali dasturchi tomonidan yaratilgan algoritm saralash uchun 50nlgnamal bajariladi))	$\frac{50 * 10^7 \ \ lg10^7}{10^7} \approx 1163\ sekund$ $(20\ \text{min}\ dan\ kam)$

Umuman olganda algoritm - bu quyilgan masalaning echimiga 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 barsha boshlang’ish berilganlar ushun umumiy bo’lishi kerak;
- algoritm to’g’ri echimga 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 eshimini toppish ushun tuziladi.

Bitta masalani eshishning bir neshaalgoritmi mavjud bo’lishi mumkin. Ular orasida eng samaralisini, bajarilishi ushun eng kam amallar, mashina vaqti, xotira va h.k.ni talab qiluvshi algoritmni tanlash lozim. Samarali algoritmlar mavjud bo’lishi shartlari va ularni qurish (ishlab shiqish)ni o’rganish algoritmlar nazariyasi asosini tashkil etadi.

Algoritm kibernetika va matematikaning asosiy tushunshalaridan biri bo’lib bu atama o’rta asrlarda yashab ijod etgan buyuk o’zbek matematigi Al-Xorazmiy nomidan kelib shiqqan. U IX asrning 825 yilidayoq o’zi kashf etgan o’nli sanoq tizimida to’rt arifmetikaamallarini bajarish qoidalarini bergan. Arifmetikaamallarini bajarish jarayoni esaalxorazm 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 ataluvshi algoritmlar ma’lum algoritm tushunshasi 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 (somputational problems) eshish ushun tuziladi.

Qo’yilgan masala ushun yaratiladigan algoritmda kiruvshi va shiquvshi ma’lumotlar muxim axamiyatga ega, agar algoritm to’g’ri tuzilgan bo’lsa, ijroshi (kompyuter) aniq natijalar beradi.

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

Aniqlik va tushunarlilik - degandaalgoritmda ijroshiga berilayotgan ko’rsatmalar aniq mazmunda bo’lishi tushuniladi. SHunki ko’rsatmalardagi noaniqliklar mo’ljallangan maqsadga

erishishgaolib kelmaydi. Ijroshiga tavsiya etiladigan ko'rsatmalar tushunarli mazmunda bo'lishi shart, aks holda ijroshi uni bajaraolmaydi.

Ommaviylik -deganda har bir algoritm mazmuniga ko'ra bir turdagi masalalarning barshasi ushun ham o'rinli bo'lishi, ya'ni umumiy bo'lishi tushuniladi.

Natijaviylik -degandaalgoritmdachekli qadamlardan so'ng albatta natija bo'lishi tushuniladi. Shuni ta'kidlash joizki, algoritm avvalfdan ko'zlangan maqsadga erishishga olib kelmasligi ham mumkin. Bunga ba'zan algoritmning noto'g'ri tuzilgani yoki boshqa xatolik sabab bo'lishi mumkin, ikkinshi 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'rtburshak piremetri va yuzasini hisoblash;
- R radusli doirani yuzasini va aylana uzunligini toppish;
- A_1, A_2, A_3,… A_n sonlarni toq elementlarini yig'indisini toppish;
- Berilgan ketma-ketlik sonlarni o'sish (kamayish) tartibda joylashtirish va h.k.

Algoritmning ushta turi mavjud: shiziqli, tarmoqlanuvshi va takrorlanuvshi(tsiklik).

CHiziqli algoritmlar - hesh qanday shartsiz faqat ketma-ket bajariladigan jarayonlardir.

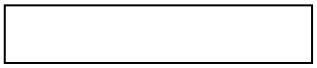
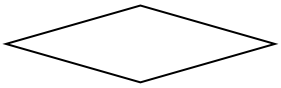
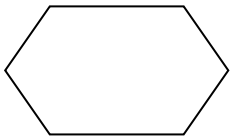
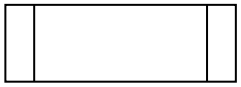
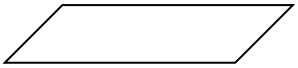
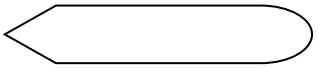
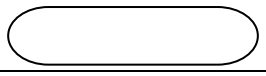
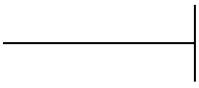
Tarmoqlanuvshi algoritmlar - ma'lum shartlarga muvofiq bajariladigan jarayonlardir.

Takrorlanuvshi algoritmlar - biron bir shart tekshirilishi yoki biron parametrning har xil qiymatlari asasidachekli 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 ushun eng ko'p ishlatiladi. Bunda algoritmdagi har bir ko'rsatma o'z shakliga ega. Masalan: parallelogramm ko'rinishdagi belgi ma'lumotlarni kiritish va shiqarish; to'g'ri to'rtburshak 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 uzgarishi
Karor		Biror shartga boglik ravishda algoritmning bajarilish yunalishini tanlash
SHakl uzgartirish		Dasturni uzgartiruvchi buyruk yoki buyruklar turkumini uzgartirish amalini bajarish
Avval aniklangan jarayon		Oldindan ishlab chikilgan dastur yoki algoritmdan foydalanish
Kiritish CHikarish		Axborotlarni kayta ishlash mumkin bulgan shaklga utkazish yoki olingan natijani tasvirlash
Displey		EXMga ulangan displeydan axborotlarni kiritish yoki chikarish
Xujjat		Axborotlarni kogozga chikarish yoki kogozdan kiritish
Axborotlar okimi chizigi		Bloklar orasidagi boglanishlarni tasvirlash
Boglagich		Uzilib kolgan axborot okimlarini ulash belgisi
Boshlash Tugatish		Axborotni kayta ishlashni boshlash, vaktincha yoki butunlay tuxtatish
Izox		Bloklarga tegishli turli xildagi tushuntirishlar

Algoritmlarni tasvirlash usullariga misollar keltirib o’tamiz:

Masala: to’g’ri to’rtburshakning 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. perimeter, diagonal va yuzasini qiymatini shop etish.
2. Formulalarda berish:

2.1. A va B to’rtburshak tomonlari qiymatlari;

2.2. *P=2*a+2*b;*

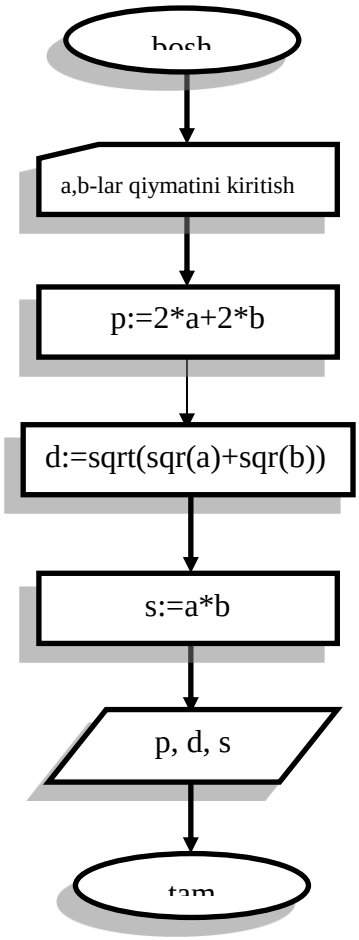
2.3. $D = \sqrt{a^2 + b^2}$;

2.4. S=a*b;

2.5.

2.6. P, D va S qiymatlarini shop etish

3. Blok-sxemalarda tasvirlash:



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

Program to’rtburshak yuzi;
Var a, b: Integer;
P, d, s: real;
Begin
Write (‘a,btomonlarni 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 vaaniq yozish ushun ishlatiladigan belgilashlar va qoidalar tizimidir. Algoritmik til oddiy tilga yaqin bo’lib u matematik belgilarni (yuqorida aytilganidek) o’z ishigaoladi. Qo’yilgan masalalarni eshishga tuzilgan algoritmlarni to’g’ridan-

to’g’ri mashinaga berib, eshib 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 ushun Pascal, Delphi, VBA, java, C++dasturlashtililari 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.
```

FOYDALANILGAN ADABIYOTLAR RO'YHATI:

1. Thomas H. Cormen va b. Intruduction to algorithms. Massachusetts Institute of Technology. London 2009.(5-10pp)
2. Слинкин Д.А.Основы программирования на Турбо-Паскале: Учебно-методическое пособие для студентов вузов. Шадринск: Изд-во Шадринского пединститута, 2003. - 244 с. (10 -р)
3. M.U.Ashurov, N.D.Mirzaxmedova .Turbo Pascal dasturlash tili.(uslubiy qo'llanma),Toshkent TDPU – 2011 (3-10pp)

2-MAVZU: ALGORITMLAR SAMARADORLIGINI BAHOLASH

Reja

1. **Algoritmlar texnologiya sifatida.**
2. **Samaradorligi**
3. **Algoritmlar va boshqa texnologiyalar**

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, echim usulini cheklangan vaqti bor va u to'g'ri javob beradi.

Kompyuterlar juda tez bo'lganda, masalani echishga 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 resurslarini 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, 2-darsta muhokama qilinadi. Birinchisini bajarish uchun, saralashni joulashtirish, bunga vaqt kerk bo'ladi, shunday baxolanmoqda 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 n2 proportsional.

Ikkinchi algoritm amalga oshirish uchun, saralash birlashtirishi, vaqt talab etadi, taxminan $c_2 n \lg n$ ga teng, lg n- 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 kan ta'sir qiladi,n ga bog'liq omillardan ko'ra, shunga ishonch xosil qilaylik.Saralashni joylashtirish algoritmni ish vaqtini shunday yozaylik c_1 n n,birlashtirish saralashini esa $c_2 n \lg n$.

Joylashtirish saralashi n omilga ega, birlashtirish saralashi esa lg n ga ega bu esa sezarli darajada kamligini ko'rishimiz mumkin.Kiritish hajmi n etarlicha katta bo'lganda qo'shish saralashi odatda tezroq bo'ladi, saralash ob'ektlar 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.Saralash elementlarini sono ishshi bilan burilish nuqtasi hosil bo'ladi,shunda birlashish saralashi yanada samarali bo'ladi.

Misol tarzida ikkita A va B kompyuterlarni ko'rib chiqamiz.A kompyuteri ancha tezroq, va unda joylashtirish saralashi 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,shuhday qilib A kompyuteri ming marta B kompyuterdan tez. Saralash birlashishi yuqori darajadagi til yordamida bir programct tomonidan amalga oshirilgan. Bu kompilyator juda samarali emas edi, va natija 50n lg n 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 komilyator bilan ham eng sekin kompyuterda ham17 marta kam vaqt talab qiladi.

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

komputerlar	Saralanadigan sonlar soni	Saralovchi algoritm	Talab qilinadigan vaqt

A(tez ishlovchi 1sekundda 10mlrd amal bajaradi)	10 mlnta (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 ishlovch- 1sekundda 10 mln amal bajaradi)		Qo'shish usuli (o'rta darajali dasturchi tomonidan yaratilgan algoritm saralash uchun 50nlgn amal bajariladi))	$\frac{50 * 10^7 \ \ lg10^7}{10^7} \approx 1163\ sekund$ $(20\ \text{min}\ dan\ kam)$

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);

• Ob'ektga yo'naltirilgan tizimlar;

• Integratsiyalashgan veb texnologiyasi;

• tezroq tarmoqlari, simli va simsiz.

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

Bundan tashqari, bunday yo'nalishlarini topish kabi bir berilgan veb-xizmati tomonidan amalga muayyan operatsiyalar uchun zarur algoritmlarni foydalanish, ko'rish va enterpolasyon manzilini, xaritalar bilan foydalaniladi. Bundan tashqari, dastur,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 asoslanga 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 algoritmlarini samaradorligini ko'rinadi 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 kop narsaga erishish mumkin.

Mashqlar

1.2.1 Dastur darajasida zarur bo'lgan algoritmik content dasturini misol qilib keltiring va bu algoritmlarni funktsiyasini 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 lg n qadamlar kerak bo'ldi. Joylashtirish saralashi birlashtirish saralashidan qiymati oshsa, n ni qiymati qancha bo'lishi kerak?

1.2.3 Ikkita algoritm bitta mashinada amalda bo'lsa, n algoritmni qaysi minimal qiymati, ish vaqti $100n_2$ formula bilan aniqlansa, ish vaqti 2^n formula bilan aniqlangan qaysi biri tezroq ishlaydi. Kimning yugurib vaqt 100p2 bilan belgilanadi, uning ish vaqti, ham algoritmlar Shu mashina amalga bo'lsa, sifatida ifodalangan bir algoritm tezroq n algoritm qaysi minimal qiymati da?

Masala

1.1.1. Algoritmlarni ish vaqtini solishtirish

Quyida bir jadval bo'lib, satrlari turli vazifalarga mos f(n), ustunlaru esa- vaqt qiymatiga t.

N ni maksimal qiymatlari bilar jadvalni toldiring,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 month**	**1 yil**	**1 asr**
$\lg n$							
$\sqrt{n}$							
n							

$nlgn$							
n^2							
n^2							
n^2							
$n!$							

FOYDALANILGAN ADABIYOTLAR RO'YHATI:

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

4-MAVZU. ALMASHISH USULIDA SARALASH,

SARALASHNING SHEYKER USULI

REJA:

1. **Almashish usulida saralash**
2. **Sheker usulida saralash**
3. **Pufakcha (qalqib chiqish) usuli**
4. **Piramida usulida saralash**

Almashish usulida saralash.

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

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

Almashtiriladigan elementlarni sarelkali kvadrat qavslar orqali, taqqoslanayotgan elementlarni esa kvadrat qavslar orqali belgilaymiz. Saralashning birinchi bosqichi 1-rasmda, ikkinchi bosqichi esa 2-rasmda ko'rsatilgan.

Har bir ro'yhatni ko'rishdan so'ng, oxiridan boshlab barcha elementlar o'zlarining oxirgi pozisiyalarini egallashini ko'rish qiyin emas.

[1]

	40	11	83	57	32	21	75	64
DASTLABKI								
BIRINCHI KO'RINISH	11	40	83	83	83	83	83	83
		40	57	32	21	75	64	
HOSIL QILINGAN	11	40	57	32	21	75	64	(83)

	11	40	57	32	21	75	64
DASTLABKI							
IKKINCHI KO'RINISH	11	40	57	57	57	75	75
		40	32	21	57	64	
HOSIL QILINGAN	11	40	32	21	57	64	(75)

[1] Ко
про
обр
71-

2-rasm. Almashish usulida saralash (birinchi ko'rinish)

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 (2- rasmga qarang).

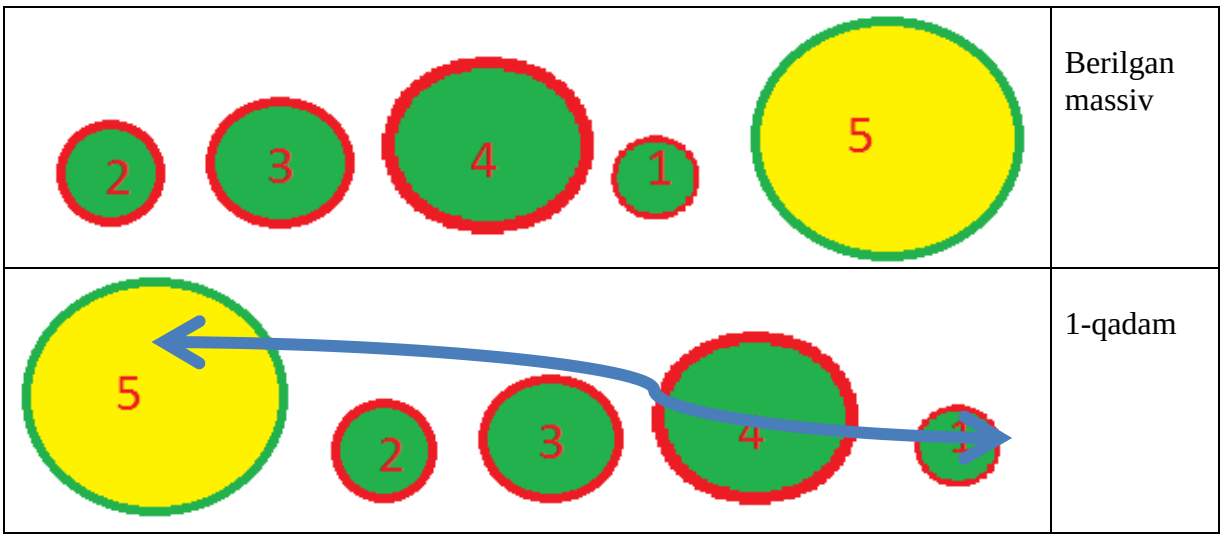
Saralash jarayoni oxirgi ro'yhatning barcha elementlari shakllangunicha davom etadi, aks holda Ayverson sharti bajarilamydi.

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).

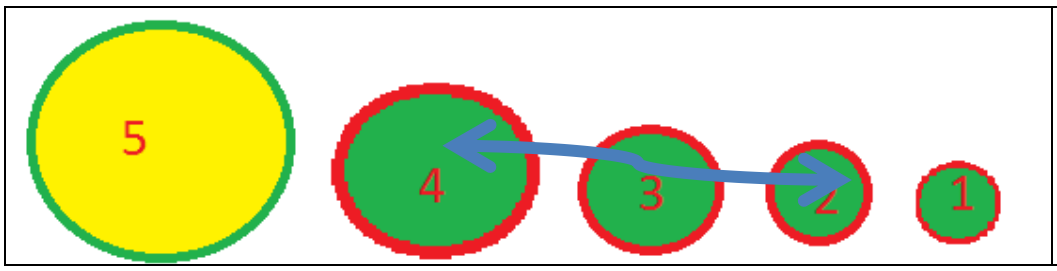
Sheker usulida saralash

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'yhat elementlarning yo'nalishi o'zgaradi.

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



[2] 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 cc.



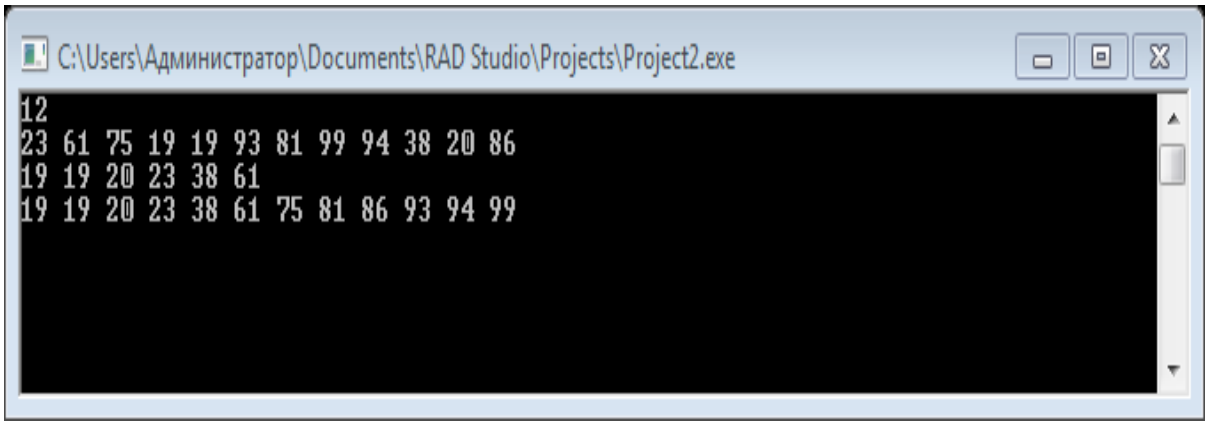
2-qadam

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 TForm1.BitBtn1Click(Sender: TObjest);
var A:ARRAY[1..15] OF INTEGER; I,D,K,Z,QAT:INTEGER;
begin
RANDOMIZE; QAT:=2; StringGrid1.RowSount:=1;
```

```
for I := 1 to 15 do
BEGIN
A[I]:=RANDOM(39); StringGrid1.Sells[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.Sells[Z,QAT]:=FloatToStr(A[Z]); END;
QAT:=QAT+1; StringGrid1.RowSount:=StringGrid1.RowSount+1;
END;
end;
end.
```

Dastur natijasi:

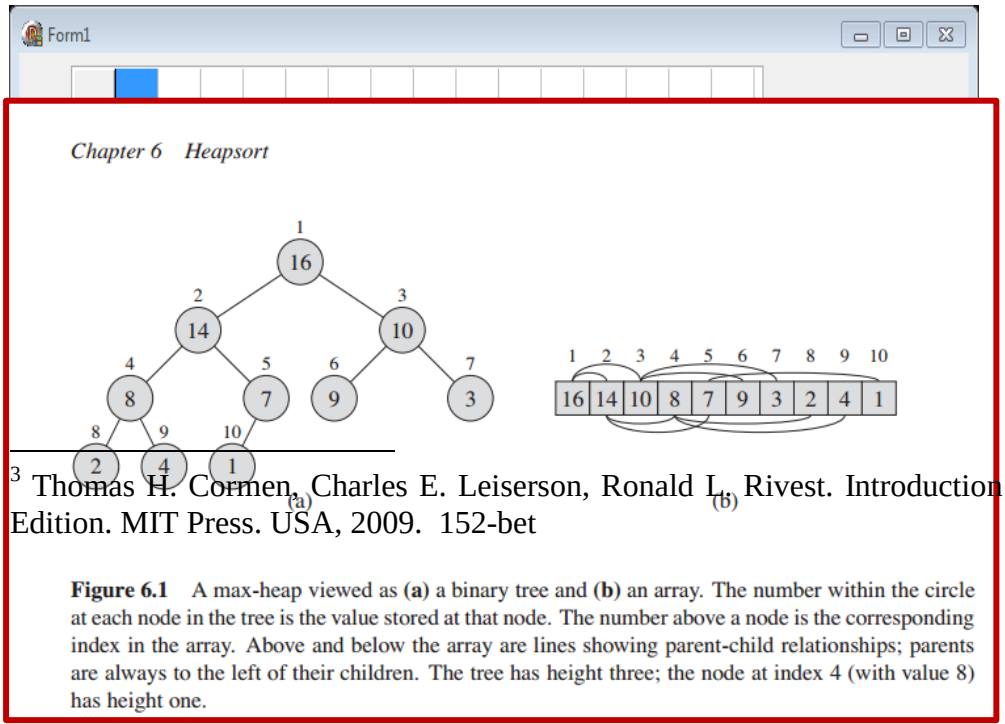


Figure 6.1 A max-heap viewed as **(a)** a binary tree and **(b)** an array. The number within the circle at each node in the tree is the value stored at that node. The number above a node is the corresponding index in the array. Above and below the array are lines showing parent-child relationships; parents are always to the left of their children. The tree has height three; the node at index 4 (with value 8) has height one.

Piramida usulida saralash

[3](Binar) piramidaning tuzilishi butun binar daraxt sifatida qaralishi mumkin bo'lgan

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

obyekt-massivni ifodalaydi (6.1-rasmda keltirilgan).

Ushbu daraxtning har bir tuguni massiv elementlariga mos keladi. Daraxt chapdan o'ngga qarab to'lib boradigan mavjud bo'lishi mumkin bo'lgan eng pastkidan tashqari barcha sathlar bo'yicha to'ldirilgan. piramidani tasvirlovchi $\boldsymbol{A}$ massiv ikkita atributlarga ega obyekt hisoblanadi: odatda massiv elementlar sonini beradigan ***A.length*** va piramidaning nechta elementlari $\boldsymbol{A}$ massivda joylashganini ko'rsatuvchi ***A.heap-size.*** Shuningdek, $\boldsymbol{A[1..A.length]}$ ega faqat $\boldsymbol{0 \leq A.heap-size \leq A.length}$ oraliqagi $\boldsymbol{A[1..A.heap-size]}$ qism massiv elementlari piramidaning to'g'ri elementlari hisoblanadigan ba'zi bir sonlarga ega bo'lishi mumkin. $\boldsymbol{A[1]}$ daraxtning ildizi bo'ladi, berilgan $\boldsymbol{i}$ indeks tuguni uchun uning ota-ona indekslarini chap va o'ng tugunlar orqali oson hisoblash mumkin.

[4]

6.1 Heaps

The *(binary) heap* data structure is an array object that we can view as a nearly complete binary tree (see Section B.5.3), as shown in Figure 6.1. Each node of the tree corresponds to an element of the array. The tree is completely filled on all levels except possibly the lowest, which is filled from the left up to a point. An array A that represents a heap is an object with two attributes: $A.length$, which (as usual) gives the number of elements in the array, and $A.heap\text{-}size$, which represents how many elements in the heap are stored within array A. That is, although $A[1..A.length]$ may contain numbers, only the elements in $A[1..A.heap\text{-}size]$, where $0 \leq A.heap\text{-}size \leq A.length$, are valid elements of the heap. The root of the tree is $A[1]$, and given the index i of a node, we can easily compute the indices of its parent, left child, and right child:

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 $\boldsymbol{i}$ indeksli ildiz tugun uchun quyidagi tengsizlik bajariladi:

$$\boldsymbol{A[PARENT(i)] \geq A[i].}$$

[5]

There are two kinds of binary heaps: max-heaps and min-heaps. In both kinds, the values in the nodes satisfy a ***heap property***, the specifics of which depend on the kind of heap. In a ***max-heap***, the ***max-heap property*** is that for every node i other than the root,

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

that is, the value of a node is at most the value of its parent. Thus, the largest element in a max-heap is stored at the root, and the subtree rooted at a node contains

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 150-151 pp

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 152-153-pp

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:

$$\boldsymbol{A[PARENT(i)] \leq A[i]}.$$

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

values no larger than that contained at the node itself. A ***min-heap*** is organized in the opposite way; the ***min-heap property*** is that for every node i other than the root,

$A[\text{PARENT}(i)] \leq A[i]$.

The smallest element in a min-heap is at the root.

Piramida xossasini saqlanishi

[6]O'smovchi piramida xossalarini saqlab turish uchun **MAX-HEAPIFY** prosedurasini 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 $\boldsymbol{A[i]}$ farzand tugunlardan kichik bo'lishi ham mumkin.

6.2 Maintaining the heap property

In order to maintain the max-heap property, we call the procedure MAX-HEAPIFY. Its inputs are an array A and an index i into the array. When it is called, MAX-HEAPIFY assumes that the binary trees rooted at LEFT(i) and RIGHT(i) are max-heaps, but that $A[i]$ might be smaller than its children, thus violating the max-heap property. MAX-HEAPIFY lets the value at $A[i]$ "float down" in the max-heap so that the subtree rooted at index i obeys the max-heap property.

```
7   largest = r
8   if largest ≠ i
9       exchange A[i] with A[largest]
10      MAX-HEAPIFY(A, largest)
```

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 154-p.

[7]Rasmda **MAX-HEAPIFY** prosedurasi 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 prosedura 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 qismdaraxt o'zining o'smovchi piramida xossalarini buzishiga olib keladi. Shuning uchun ham ushbu daraxt uchun **MAX-HEAPIFY** prosedurasini 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.

Figure 6.2 illustrates the action of MAX-HEAPIFY. At each step, the largest of the elements $A[i]$, $A[\text{LEFT}(i)]$, and $A[\text{RIGHT}(i)]$ is determined, and its index is stored in *largest*. If $A[i]$ is largest, then the subtree rooted at node i is already a max-heap and the procedure terminates. Otherwise, one of the two children has the largest element, and $A[i]$ is swapped with $A[largest]$, which causes node i and its children to satisfy the max-heap property. The node indexed by *largest*, however, now has the original value $A[i]$, and thus the subtree rooted at *largest* might violate the max-heap property. Consequently, we call MAX-HEAPIFY recursively on that subtree.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 154-155 pp

2) Binar daraxt barglari bir sathda yoki ikkilasi qo’shni joylashadi.(3-rasm)

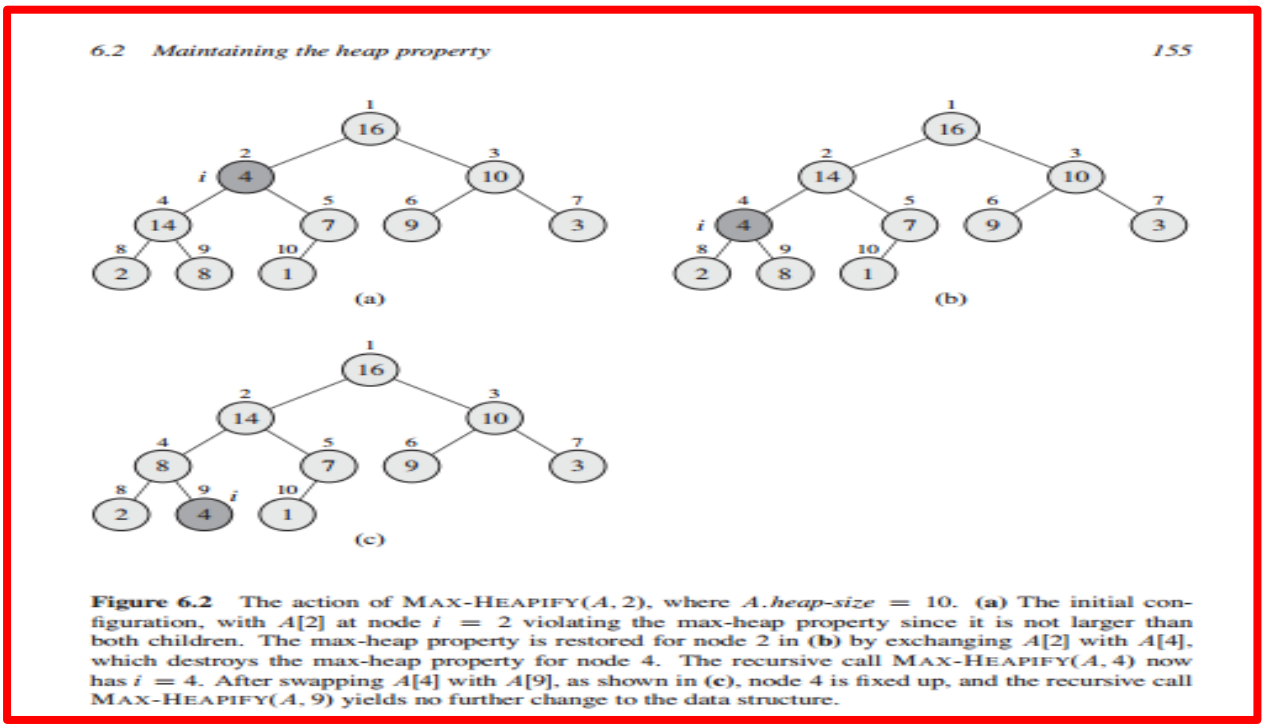
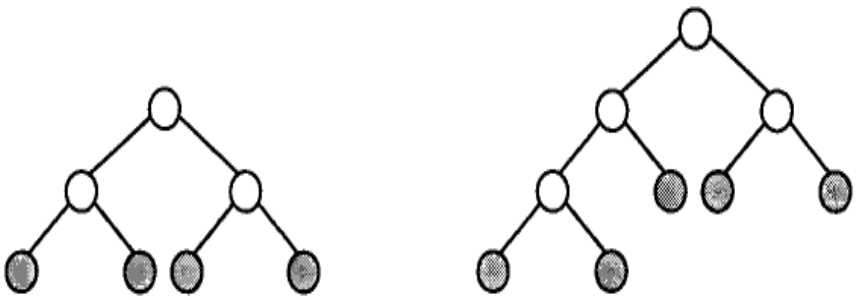


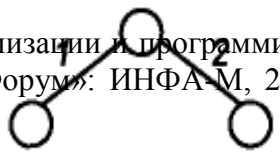
Figure 6.2 The action of MAX-HEAPIFY$(A, 2)$, where $A.heap\text{-}size = 10$. **(a)** The initial configuration, with $A[2]$ at node $i = 2$ violating the max-heap property since it is not larger than both children. The max-heap property is restored for node 2 in **(b)** by exchanging $A[2]$ with $A[4]$, which destroys the max-heap property for node 4. The recursive call MAX-HEAPIFY$(A, 4)$ now has $i = 4$. After swapping $A[4]$ with $A[9]$, as shown in **(c)**, node 4 is fixed up, and the recursive call MAX-HEAPIFY$(A, 9)$ yields no further change to the data structure.

3) Pastki sath barglari baland barglar sathidan chaproqda joylashadi. [89]



3-rasm. Binar daraxt:
a-bir sathdagi barglar; b-qo’shni sathlardagi burglar

Aylantirish jarayonida triad elementlari ikki marotaba taqqoslanadi (4-rasm), shu bilan birga eng katta element yuqoriga, eng kichik element esa pastga o’tadi.



[8] 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-cc.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 155-p.

4-rasm. Triad elementlarini taqqoslash:
1- birinchi taqqoslash; 2- ikkinchi taqqoslash;

[10]

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

FOYDALANILGAN ADABIYOTLAR RO'YHATI

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 150-155 pp, 159-161 pp.
2. 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, 75, 76 сс.

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

5-MAVZU: ALGORITMLAR TAHLIL

REJA

1. Tahlil nima?
2. Kiruvchi berilganlar sinfi
3. Xotira bo‘yicha murakkablik
4. Nimani hisoblash va nimani inobatga olish lozim
5. Kiruvchi ma’lumotlarning sinflari
6. Eng yaxshi holat
7. Eng yomon holat
8. O‘rtacha holat

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 masalalaning 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 nuqtai nazaridan ularning murakkablik darajalari teng. Bu ikki algoritm vaqt nuqtai 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 tahlili 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 tutish 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. 8 .4 § da keltirilgan *N + 5* ta va *N* + 250 ta amal bajariladigan ikki algoritm orasida N ning yetarlicha katta qiymatlarida deyarli farq bo'lmaydi. Shunga qaramay, biz algoritmlarni bajariladigan amallar soniga qarab tahlil qilamiz.

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 hisoblovchi 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 xar bir o'tishda uning sikl chegarasidan chiqmayotganligi tekshiriladi va o'zgaruvchi qiymatini oshiradi. Bu esa sikl bajarilishida 257 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 xajmdagi 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 aloxida 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 mexanizmlari 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. Shuning uchun bu bobni o‘rganishda mantiqiy ifodalarning qiymatini qisqartirib hisoblanishini e’tiborsiz qoldiramiz.

7.2. 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 kata 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.

7.3. 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 soni yozishda ko‘pchilik kompyuterlar 4 dan 8 baytgacha xotira sarflaydi, leki nagar bu soni 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.

7.4 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 harajatlarni yoki ma’lumotlarni qayd eti shva hisobga olishga ketishi tashkil etadi. Ikki xil ahamiyatli jarayon turi mavjud: taqqoslash va arifmetik amallar. Barcha taqqoslash operatorlari ekvivalent hisoblanadi va ularni izlash hamda ajratuv 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. Biz o’z kitobimizda yana bir, ko‘paytirishdan ham ko‘p vaqt talab qiluvchi amallar guruhini hosil qiluvchi logarifmlar va trigonametrik funksiyalardan foydalanuvchi algoritmlarga to‘xtalmadik (odatda kompyuterlar bu ifodalarni bir qatorga ajratish yordamida hisoblaydilar). 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 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 qatiy nazar, bitta taqqoslash kerak bo‘ladi. Shuni eslatish kerakki, ro‘yxatdan izlashda, uning qanchalik uzun bo‘lishidan qatiy 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

$$A(n)\sum_{i=1}^{m} p_i \cdot t_i, \quad (1)$$

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 Xill deb taxmin qilamiz. Boshqacha aytganda, agar guruh 5 ta bo‘lsa, birinchi guruhga tushish extimoli ikkinchi yoki boshqa guruhga tushish extimolidek, ya’ni har bir guruhga tushish extimoli 0,2 ga teng. Bu holda ishning o‘rtacha vaqtini avvalgi formula Bilan yoki unga ekvivalent soddalashtirilgan barcha guruhlarning teng extimolligida haqiqiy bo‘lgan

$$A(n)\sum_{i=1}^{m} p_i \cdot t_i, \textbf{ (2)}$$

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

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

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

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

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

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

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

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

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

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

FOYDALANILGAN ADABIYOTLAR RO'YHATI:

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

6-MAVZU. ALGORITMLARNI ISHLAB CHIQISH USLUBLARI

REJA

1. Algoritmlarni konstruksiyalash

2. Algoritmlarni ekvivalent qayta ishlash.

3. Toraytiruvchi o'zgartirishlar.

4. Formal usulni matematikaga bog'liq bo'lmagan muammoga qo'llash.

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 qattai 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.

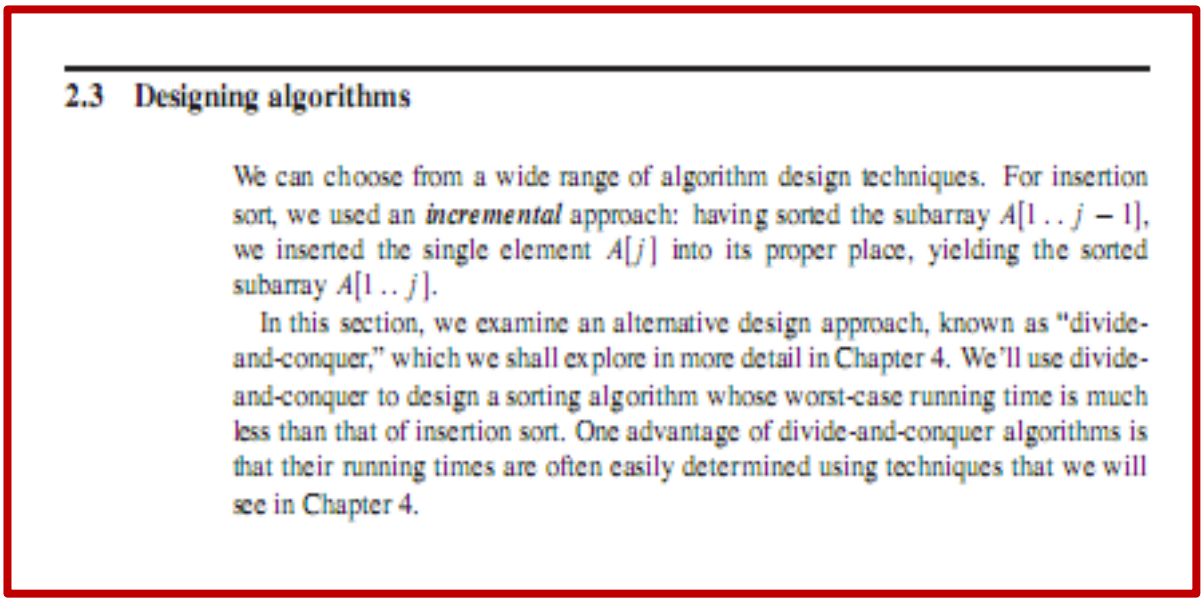
Misol tariqasida, algoritmni bir tildan boshqa tilga o'tkazishni keltirish mumkin. Shu bilan birgalikda algoritmni ekvivalent qayta ishlash usuli bilan keskin o'zgartirish mumkin, lekin bu holda asosiy e'tiborni dastlabki algoritmga nisbatan yahshi algoritmni yaratishga berish kerak.

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 maqsaddida foydalaniladi.

4. <u>Formal usulni matematikaga bog'liq bo'lmagan muammoga qo'llash.</u> Buyerda 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?



2.3 Designing algorithms

We can choose from a wide range of algorithm design techniques. For insertion sort, we used an *incremental* approach: having sorted the subarray $A[1 \ldots j-1]$, we inserted the single element $A[j]$ into its proper place, yielding the sorted subarray $A[1 \ldots j]$.

In this section, we examine an alternative design approach, known as "divide-and-conquer," which we shall explore in more detail in Chapter 4. We'll use divide-and-conquer to design a sorting algorithm whose worst-case running time is much less than that of insertion sort. One advantage of divide-and-conquer algorithms is that their running times are often easily determined using techniques that we will see in Chapter 4.

2.3.1 The divide-and-conquer approach

Many useful algorithms are ***recursive*** in structure: to solve a given problem, they call themselves recursively one or more times to deal with closely related subproblems. These algorithms typically follow a ***divide-and-conquer*** approach: they break the problem into several subproblems that are similar to the original problem but smaller in size, solve the subproblems recursively, and then combine these solutions to create a solution to the original problem.

The divide-and-conquer paradigm involves three steps at each level of the recursion:

Divide the problem into a number of subproblems that are smaller instances of the same problem.

Conquer the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.

Combine the solutions to the subproblems into the solution for the original problem.

The ***merge sort*** algorithm closely follows the divide-and-conquer paradigm. Intuitively, it operates as follows.

Divide: Divide the n-element sequence to be sorted into two subsequences of $n/2$ elements each.

Conquer: Sort the two subsequences recursively using merge sort.

Combine: Merge the two sorted subsequences to produce the sorted answer.

The recursion "bottoms out" when the sequence to be sorted has length 1, in which case there is no work to be done, since every sequence of length 1 is already in sorted order.

The key operation of the merge sort algorithm is the merging of two sorted sequences in the "combine" step. We merge by calling an auxiliary procedure MERGE(A, p, q, r), where A is an array and p, q, and r are indices into the array such that $p \le q < r$. The procedure assumes that the subarrays $A[p..q]$ and $A[q+1..r]$ are in sorted order. It ***merges*** them to form a single sorted subarray that replaces the current subarray $A[p..r]$.

Our MERGE procedure takes time $\Theta(n)$, where $n = r - p + 1$ is the total number of elements being merged, and it works as follows. Returning to our card-playing motif, suppose we have two piles of cards face up on a table. Each pile is sorted, with the smallest cards on top. We wish to merge the two piles into a single sorted output pile, which is to be face down on the table. Our basic step consists of choosing the smaller of the two cards on top of the face-up piles, removing it from its pile (which exposes a new top card), and placing this card face down onto

the output pile. We repeat this step until one input pile is empty, at which time we just take the remaining input pile and place it face down onto the output pile. Computationally, each basic step takes constant time, since we are comparing just the two top cards. Since we perform at most n basic steps, merging takes $\Theta(n)$ time.

The following pseudocode implements the above idea, but with an additional twist that avoids having to check whether either pile is empty in each basic step. We place on the bottom of each pile a ***sentinel*** card, which contains a special value that we use to simplify our code. Here, we use ∞ as the sentinel value, so that whenever a card with ∞ is exposed, it cannot be the smaller card unless both piles have their sentinel cards exposed. But once that happens, all the nonsentinel cards have already been placed onto the output pile. Since we know in advance that exactly $r - p + 1$ cards will be placed onto the output pile, we can stop once we have performed that many basic steps.

```
MERGE(A, p, q, r)
 1  n1 = q − p + 1
 2  n2 = r − q
 3  let L[1 . . n1 + 1] and R[1 . . n2 + 1] be new arrays
 4  for i = 1 to n1
 5      L[i] = A[p + i − 1]
 6  for j = 1 to n2
 7      R[j] = A[q + j]
 8  L[n1 + 1] = ∞
 9  R[n2 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13      if L[i] ≤ R[j]
14          A[k] = L[i]
15          i = i + 1
16      else A[k] = R[j]
17          j = j + 1
```

In detail, the MERGE procedure works as follows. Line 1 computes the length n_1 of the subarray $A[p \,..\, q]$, and line 2 computes the length n_2 of the subarray $A[q + 1 \,..\, r]$. We create arrays L and R ("left" and "right"), of lengths $n_1 + 1$ and $n_2 + 1$, respectively, in line 3; the extra position in each array will hold the sentinel. The **for** loop of lines 4–5 copies the subarray $A[p \,..\, q]$ into $L[1 \,..\, n_1]$, and the **for** loop of lines 6–7 copies the subarray $A[q + 1 \,..\, r]$ into $R[1 \,..\, n_2]$. Lines 8–9 put the sentinels at the ends of the arrays L and R. Lines 10–17, illus-

32 Chapter 2 Getting Started

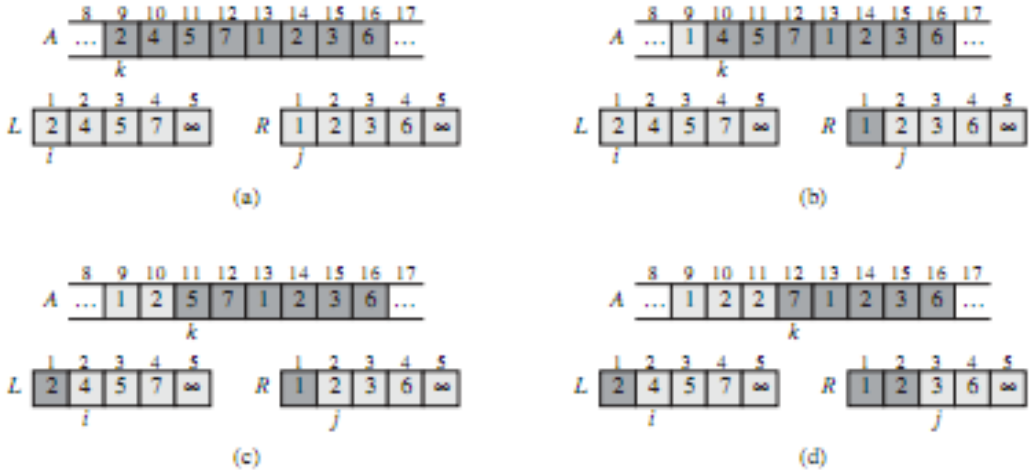


Figure 2.3 The operation of lines 10–17 in the call MERGE($A, 9, 12, 16$), when the subarray $A[9..16]$ contains the sequence $\langle 2, 4, 5, 7, 1, 2, 3, 6\rangle$. After copying and inserting sentinels, the array L contains $\langle 2, 4, 5, 7, \infty\rangle$, and the array R contains $\langle 1, 2, 3, 6, \infty\rangle$. Lightly shaded positions in A contain their final values, and lightly shaded positions in L and R contain values that have yet to be copied back into A. Taken together, the lightly shaded positions always comprise the values originally in $A[9..16]$, along with the two sentinels. Heavily shaded positions in A contain values that will be copied over, and heavily shaded positions in L and R contain values that have already been copied back into A. (a)–(h) The arrays A, L, and R, and their respective indices k, i, and j prior to each iteration of the loop of lines 12–17.

trated in Figure 2.3, perform the $r - p + 1$ basic steps by maintaining the following loop invariant:

> At the start of each iteration of the **for** loop of lines 12–17, the subarray $A[p..k-1]$ contains the $k - p$ smallest elements of $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$, in sorted order. Moreover, $L[i]$ and $R[j]$ are the smallest elements of their arrays that have not been copied back into A.

We must show that this loop invariant holds prior to the first iteration of the **for** loop of lines 12–17, that each iteration of the loop maintains the invariant, and that the invariant provides a useful property to show correctness when the loop terminates.

Initialization: Prior to the first iteration of the loop, we have $k = p$, so that the subarray $A[p..k-1]$ is empty. This empty subarray contains the $k - p = 0$ smallest elements of L and R, and since $i = j = 1$, both $L[i]$ and $R[j]$ are the smallest elements of their arrays that have not been copied back into A.

[11]

FOYDALANILGAN ADABIYOTLAR RO'YHATI:

[11] **Thomas H. Cormen va b. Intruduction to algorithms. Massachusetts Institute of Technology. London 2009. (29-31 pp)**

1.Thomas H. Cormen va b. Intruduction to algorithms. Massachusetts Institute of Technology. London 2009. (29-31 pp)

7-MAVZU: REKURSIYA VA REKURSIV FUNKSIYALAR

REJA

1. **Rekursiya**
2. **Rekursiv funksiyalar**
3. **Fibonashchi sonlarining *n*- hadini hisoblash algoritmi.**

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 funktsiyalarni 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 funktsiya 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 tsikllarni o‘z ichiga olmasligi ham mumkin. Ammo rekursiv algoritmlar, avvalo, yechilayotgan masala, hisoblanayotgan funktsiya 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 protsedurani tarkibiga olsa, u holda R bilvosita rekursiv deyiladi.

Rekursiv protseduralar cheksiz hisoblashlarga yo‘l oshganligini 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.

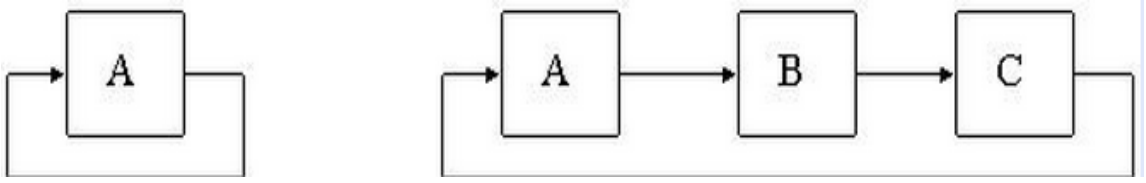
Shuning uchun ham rekursiv algoritmlarning sxemasini quyidagi ko‘rinishlardan biri bilan ifodalash mumkin:

P=IF B THEN P[S,P] END

P=…

Ma’lumki, har qanday protsedura yoki funktsiya boshqa protsedura yoki funktsiyalarga murojaatni o‘z ichiga olishi mukin. Xususiy holda, agar bu murojaat protsedura yoki funktsiyaning o‘ziga murojaatdan iborat bo‘lsa, bu protsedura yoki funktsiya rekursiv deyiladi.[12]

Umumiy holda, rekursiv prtsedura va funktsiyalarni quyidagi sxema ko‘rinishida ifodalashimiz mumkin:



Rekursiv protseduraga misol sifatida quyidagi qism dasturni tahlil etaylik:

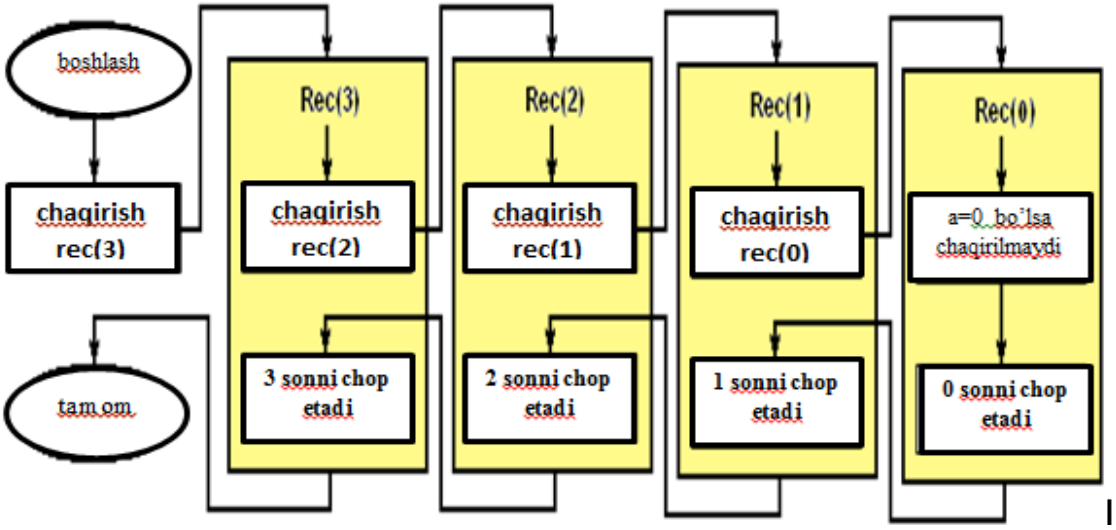
```
prosedure Res(a: integer);
begin
 if a>0 then
 Res(a-1);
 writeln(a);
```

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

end;

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

Quyidagi blok-sxemada operatorlarning bajarilish ketma-ketligi tasvirlangan:



Boshlanish Res(3)ga murojaat Tugash

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

Res protsedurasi dastlab a=3 parametr bilan chaqiriladi, ya’ni Res(3) tarzidagi murojaat amalga oshiriladi. Uning tarkibida esa a=2 parametrli Res(2) tarzidagi murojaat mavjud. Hali dastlabki murojaat tugamasdan keyingi protseduraga murojaat tashkil etiladi, u tugamasa dastlabki Res(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.

Misol. Sonni ikkilik sanoq sistemasiga o‘tkazish masalasini tsikl operatori va rekursiya orqali hal etishni qarab chiqamiz.

Ma’lumki, ikkilik sanoq sistemadagi sonlarni hosil qilish uchun berilgan sonni sanoq sistema asosi bo‘lan 2ga 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:

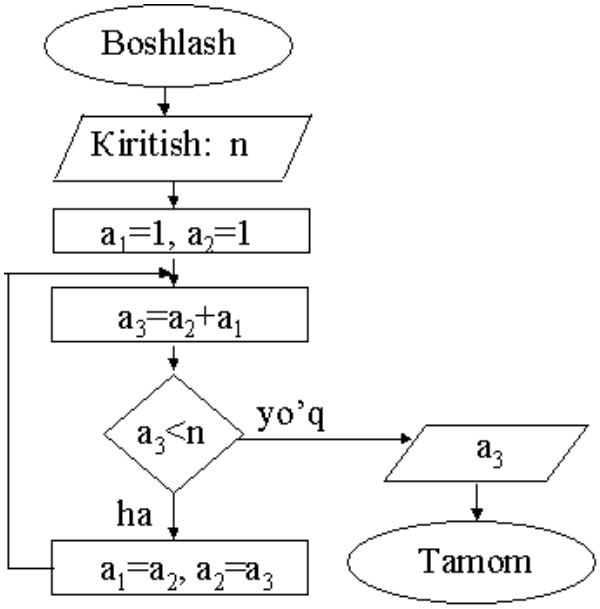
X2=x div 2

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

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

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

$a_0 q a_1 q 1$, $a_i q a_{i-1} + a_{i-2}\ iq 2,3,4,....$ Bu rekkurent ifoda algoritmiga mos keluvchi blok-sxema 2.15-rasmda keltirilgan. Eslatib o‘tamiz formuladagi *i*-indeksga hojat yo‘q, agar Fibonashchi sonining nomerini ham aniqlash zarur bo‘lsa, birorta parametr-kalit kiritish kerak bo‘ladi.



Fibonashchi sonlarining *n*- hadini hisoblash algoritmi.

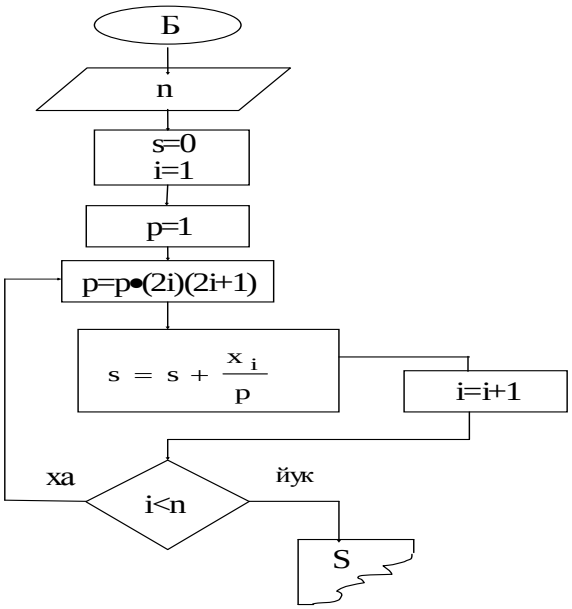
2-misol. $S = \sum_{i=1}^{n} \frac{x_i}{(2i+i)!}$

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 hakozo 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.

6. Ichma-ich joylashgan tsiklik algoritmlar, rekursiya va rekurrent algoritmlarga doir ma’lumotlar tayyorlang

FOYDALANILGAN ADABIYOTLAR RO’YHATI

1. A.U.Ashurov, N.D.Mirzahmedova, N.S.Haytullayeva. Algoritmlash va Dasturlash tillari Informatika O‘qitish metodikasi ta’lim yo‘nalishi uchun uslubiy qo‘llanma. Toshkent – 2015 (113-115 Б)

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

10-MAVZU. GRAFLAR BILAN ISHLOVCHI SODDA ALGORITMLAR

REJA:

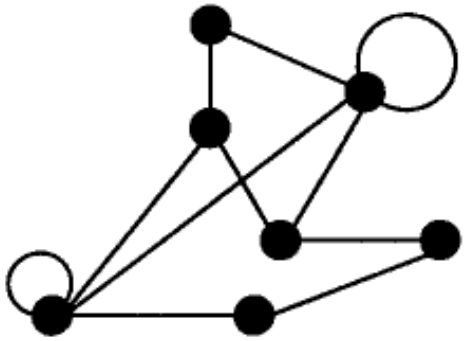
1. **Graf haqida umumiy tushuncha**
2. **Orientirlangan graflar**
3. **Orientirlanmagan graflar**
4. **Qo'shni graflar**

Graflar nazariyasi – bu diskret matematikaning sohasi bo'lib, ayniqsa geometrik yondashuv asosida ob'ektlarni 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.

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

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

[13]Agar **G** da $(\boldsymbol{u, v})$ qirra mavjud bo'lsa, u holda Ikkita $\boldsymbol{u}$ va $\boldsymbol{v}$ uchlar qo'shni bo'ladi,



1-rasm. Grafli tuzilish(tarmoq)

Grafning tasvirlanishi

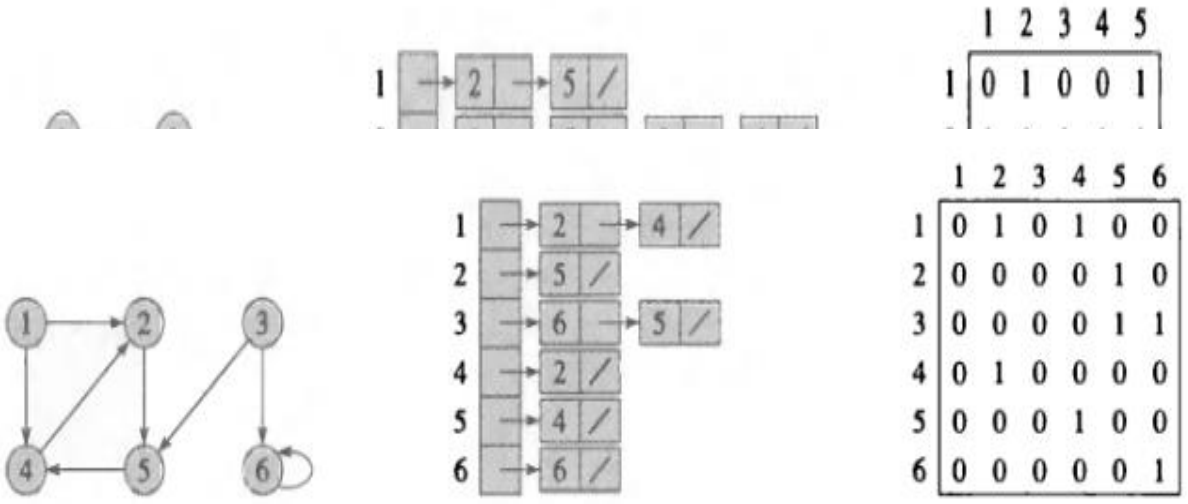
$\boldsymbol{G = (V, E)}$ graflarni ikki xil usulda tasvirlash mumkin: uchlar ro'yhatlari jamlanmasi va qo'shni matrisalar. Ikki usul ham orentirlangan va orentirlanmagan graflarda qo'llaniladi. Ko'p hollarda uchlar ro'yhatlari jamlanmasi foydalaniladi, chunki bu usul ruxsat etilgan graflarni $|E|$ V^2 dan kichik bo'lgan hollarda ixchamroq tasvirlaydi. Orientirlangan graf- bu qirralarida harakat yo'nalishlari bo'yicha garf 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.

---

[13] 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.

[14]

2



a **b** **c**

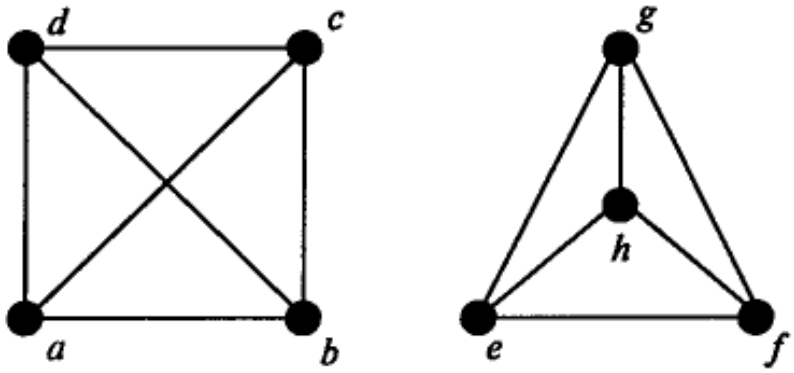
qo'shni ro'yhatlar yordamida tasvirlangan c) G qo'shni matrisalar yordamida tasvirlangan.

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.(4-rasm).

[15]



4-rasm. 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.

---

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

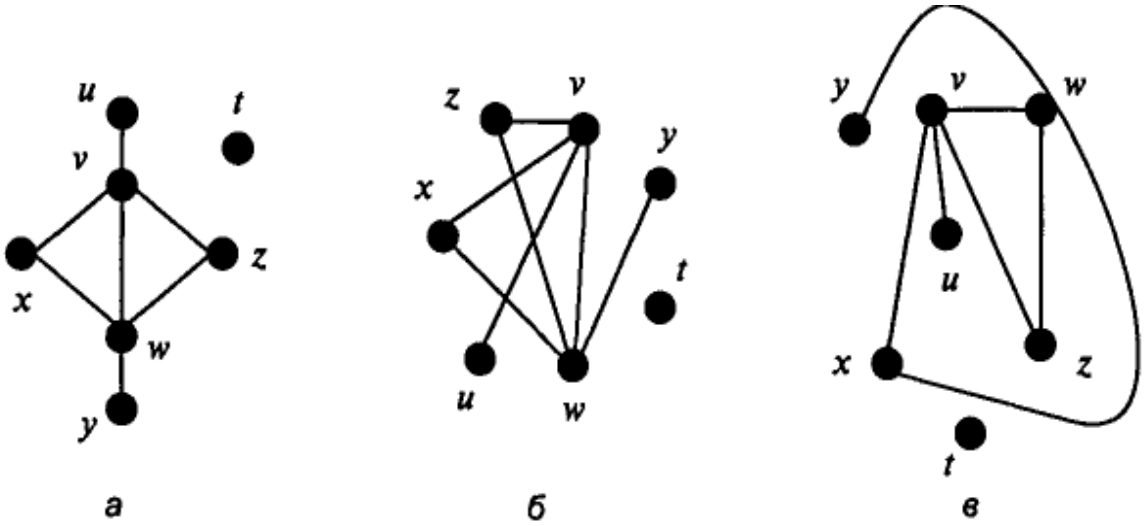
[15] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 109-с.

Graf uchlari darajasi

$\boldsymbol{V}$ graflar uchlari darajasini bildiruvchi $\boldsymbol{d_v}$ yoki $\mathbf{deg}(\boldsymbol{v})$ uning qirralar soniga teng bo'ladi, ya'ni

$$\boldsymbol{d_v} = \mathbf{deg}(\boldsymbol{v}) = |\boldsymbol{N}(\boldsymbol{v})|$$

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.



5-rasm. Tarmoq izomorfizmi

Graf uchi darajasini aniqlashda halqa ikki marotaba hisobga olinadi.

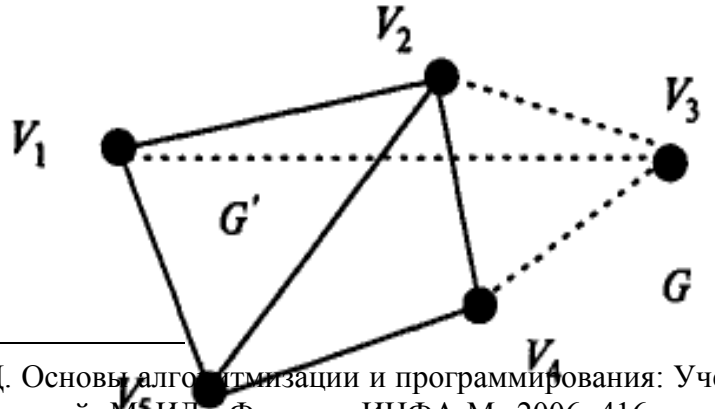
Agar **deg(v)=1** ga teng bo'lsa, u holda V uch qulflangan deyiladi.

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

Qism graf tushunchasi

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

16



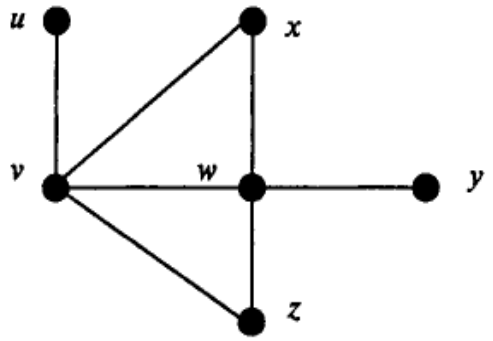
[16] Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие/ Под ред. проф. Л.Г.Гагариной.-М.:ИД «Форум»: ИНФА-М, 2006.-416 с.: ил. –(Профессиональное образование). 109-110 сс.

6-rasm

$G = (V, E)$ 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 $W = u_1, u_2, \dots, u_n$ uchlar ketma-ketligidir.

Agar $u_1 = u_n$ bo'lsa, u holda W yo'nalish yopiq, boshqa hollarda esa ochiq bo'ladi. agar bitta ham uch W bir martadan ko'p payda bo'lmasa.

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.

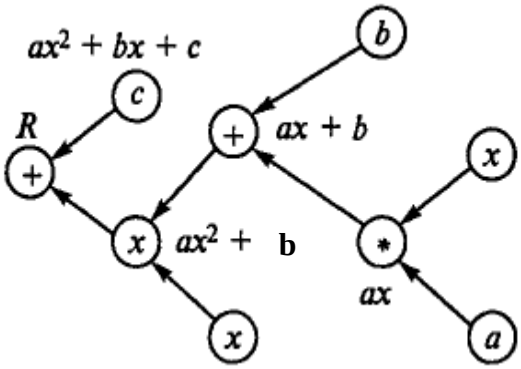


[17]

7-rasm.Graflarda yo'nalish va sikl

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.

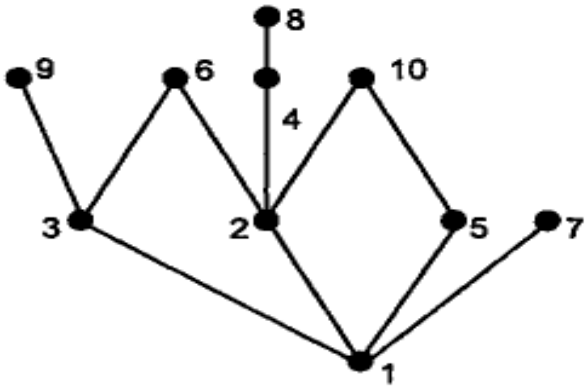
Graflar hisoblash algoritmlarini tasvirlashda qulay vosita hisoblanadi. Misol tariqasida, Gorner sxemasida $ax^2 + bx + c$ kvadrat ko'phadni hisoblashni ko'rib chiqamiz.(8-rasm)



[17] 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. (9-rasm)



9-rasm. To'plamda bo'lish munosabatini tasvirlovchi graf

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

[18]ko'rinishida tasvirlanishda V dagi har bir uchdan $|V|$ ro'yhatning Adj massiv ishlatiladi.

[19]Har bir $u \in V$ uch uchun $Adj[u]$ qo'shni ro'yhat $(u, v) \in E$ bo'lgan barcha v uchlardan tashkil topadi. Qo'shni ro'yhat graf qirrasini tasvirlaganligi sababli, Adj massiv grafning

> The *adjacency-list representation* of a graph $G = (V, E)$ consists of an array Adj of $|V|$ lists, one for each vertex in V. For each $u \in V$, the adjacency list $Adj[u]$ contains all the vertices v such that there is an edge $(u, v) \in E$. That is, $Adj[u]$ consists of all the vertices adjacent to u in G. (Alternatively, it may contain pointers to these vertices.) Since the adjacency lists represent the edges of a graph, in pseudocode we treat the array Adj as an attribute of the graph, just as we treat the edge set E. In pseudocode, therefore, we will see notation such as $G.Adj[u]$. Figure 22.1(b) is an adjacency-list representation of the undirected graph in Figure 22.1(a). Similarly, Figure 22.2(b) is an adjacency-list representation of the directed graph in Figure 22.2(a).

atributi sifatida qaraladi.

[20]Agar G orientirlangan graf bo'lsa, u holda barcha qo'shni ro'yhatlarning yig'indisining uzunligi. (u, v) qirraga $Adj[u]$ ro'yhatdagi v element mos kelganligi uchun. $|E|$ ga teng

---

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

> an undirected graph, the sum of the lengths of all the adjacency lists is $2|E|$, since if (u, v) is an undirected edge, then u appears in v's adjacency list and vice versa. For both directed and undirected graphs, the adjacency-list representation has the desirable property that the amount of memory it requires is $\Theta(V + E)$.

[20] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 590-591 pp.

bo'ladi. Agar G orientirlanmagan graf bo'lsa, u holda barcha qo'shni ro'yhatlarning yig'indisining uzunligi (u, v) qirra u hamda v qo'shni ro'yhatlarda paydo bo'lganligi uchun $2|E|$ ga teng bo'ladi. Orientirlangan va orientirlanmagan graflarning ro'yhatlar ko'rinishiga tasvirlanishi uchun $\theta(V + E)$ hajmdagi xotira talab etiladi.

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

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

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

2 va 3 rasmlarda orientirlangan va orientirlanmagan graflar qo'shni matrisalari ko'rsatilgan. Graflar qo'shni matrisalari grafning qirrasiga bog'liq bo'lmagan holda $\theta(V^2)$ xotirani talab qiladi.

that permit faster edge lookup.)

For the ***adjacency-matrix representation*** of a graph $G = (V, E)$, we assume that the vertices are numbered $1, 2, \ldots, |V|$ in some arbitrary manner. Then the adjacency-matrix representation of a graph G consists of a $|V| \times |V|$ matrix $A = (a_{ij})$ such that

$$a_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Figures 22.1(c) and 22.2(c) are the adjacency matrices of the undirected and directed graphs in Figures 22.1(a) and 22.2(a), respectively. The adjacency matrix of a graph requires $\Theta(V^2)$ memory, independent of the number of edges in the graph.

[21]

2-(b) rasmdagi qo'shni matrisalarning asosiy diagonalga nisbatan simmetrikligiga e'tibor bering graf orientirlanmaganligidan (u, v) va (v, u) bir xil yoqga ega ekanliklarini bildiradi orientirlanmagan grafning qo'shni matrisalari $A = A^T$ transponirlangan qo'shni matrisalarga mos keladi. Ilovalar qatorida ushbu xossa asosiy dioganal va undan

Observe the symmetry along the main diagonal of the adjacency matrix in Figure 22.1(c). Since in an undirected graph, (u, v) and (v, u) represent the same edge, the adjacency matrix A of an undirected graph is its own transpose: $A = A^T$. In some applications, it pays to store only the entries on and above the diagonal of the adjacency matrix, thereby cutting the memory needed to store the graph almost in half.

[21] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 591-p.

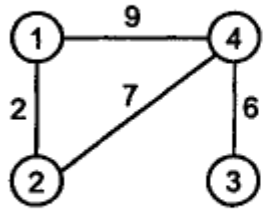
yuqorida joylashgan faqat matrisa elementlarini saqlash imkonini beradi, bu esa ikki marta zarur xotira sig'imini kamayirish imkonini beradi.

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

Agar G grafning uchlari $v_1, v_2, \ldots, v_m$ nishonlar bilan belgilangan bo'lsa, u holda A qo'shni matrisa 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. rasmda graf va qo'shni matrisalar tasvirlangan.



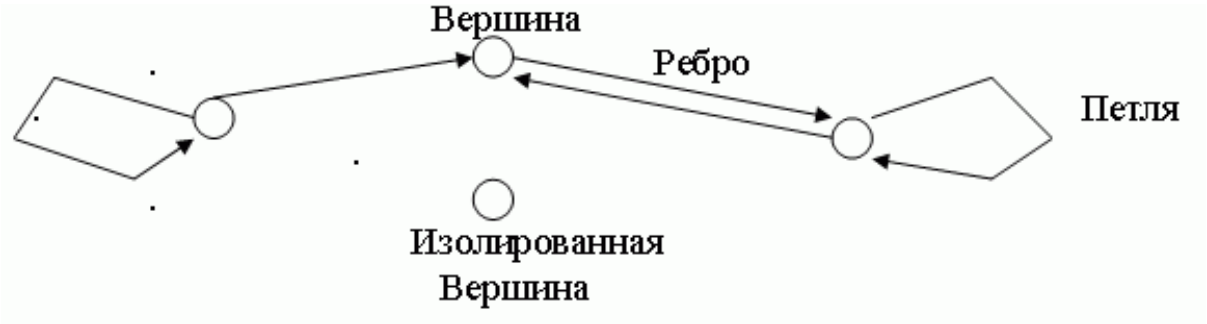
$A =$

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

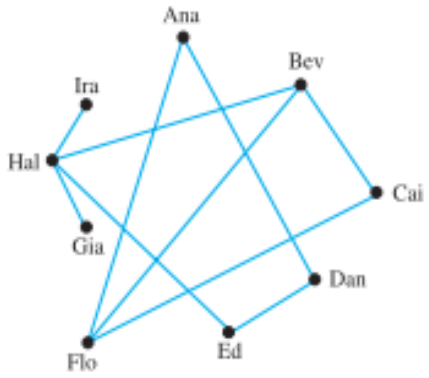
10- *а* *б* rasm. (a) Graf, (b) qo'shni martisa

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

Ushbu sxemalarda chiziqlar uzunligi, eni va shakli hech qanday ahamiyatga ega emas.



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



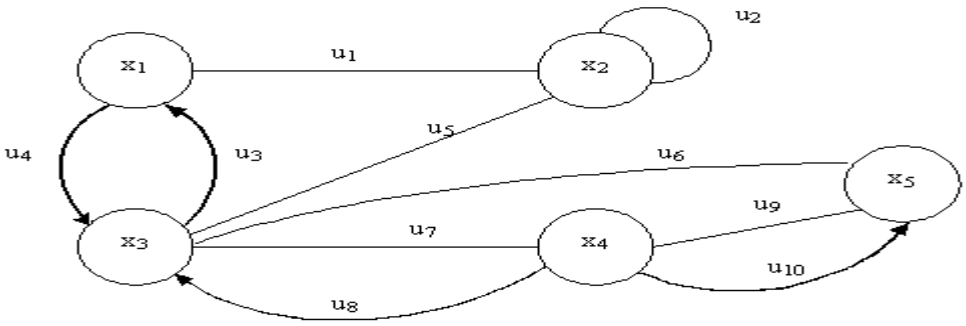
Graflarga misollar

Shunday qilib graf erkin konstruksiyalardir. Bunda ikki uchlari orasidagi bog'lanishning bo'lishi muhimdir, bir xilda ushbu bog'lanishni 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. 11-rasmda 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.

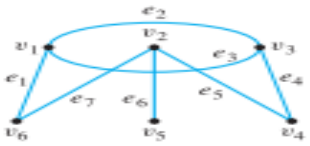
11-rasm

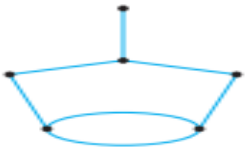


Ko'pincha masalalarda uchlar orasidagi munosabat muhim rol o'ynamaydi. Bunga misol tariqasida tartib munosabat bo'lishi mumkin. Masalan, x_i x_j dan katta bu holda x_j x_i dan katta bo'lishi mumkin emas. Demak, uchlar orasidagi munosabat ma'lum'mo'ljalga egadir. Bunday graflarni mo'ljalga ega graflar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

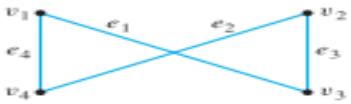
In 5–7, show that the two drawings represent the same graph by labeling the vertices and edges of the right-hand drawing to correspond to those of the left-hand drawing.

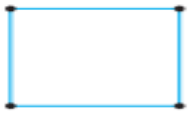
5.



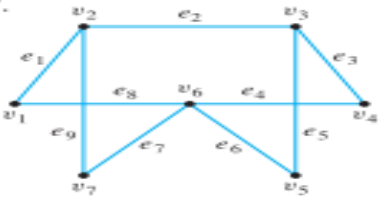


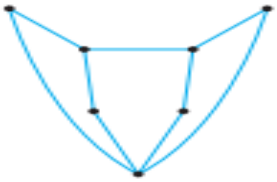
6.





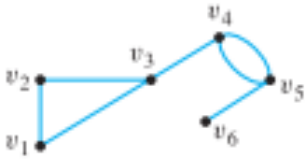
7.



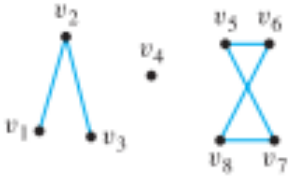


Quyidagi graflarning qaysi biri bog'langan.

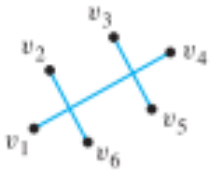
Which of the following graphs are connected?



(a)



(b)



(c)

FOYDALANILGAN ADABIYOTLAR RO'YHATI

3. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to Algorithms, 3rd Edition. MIT Press. USA, 2009. 590-591 pp.
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). 108-110 сс, 112 с.
5. Susanna S. Epp. Discrete Mathematics with Applications, Fourth Edition. Printed in Canada 2011. 639-p.

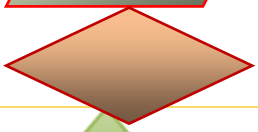
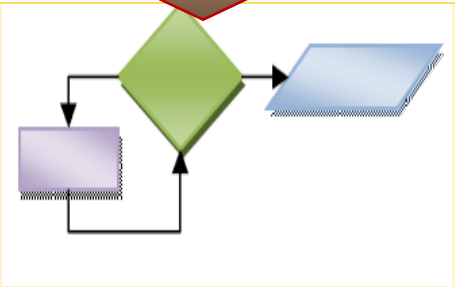
12-MAVZU. CHIZIQLI, TARMOQLANUVCHI VA TAKRORLANUVCHI DASTURLAR

REJA:

1. **Chiziqli dasturlar**
2. **Tarmoqlanuvchi dasturlar**
3. **Takrorlanuvchi dasturlar**

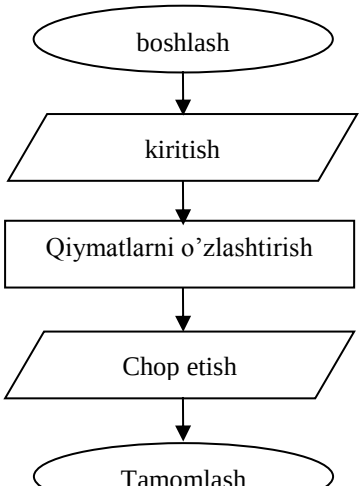
Chiziqli dasturlar tuzilish jihatidan aniq ketma-ketlikdagi amallardan iborat. Bu amallarning bajarilishi dasturda keltirilish tartibi bilan bog‘liq, ya’ni yozilgan ketma-ketlikda bajariladi. Har bir masalani yechishdan oldin algoritimi tuzib olinadi. Algoritim tuzishda quyidagi geometrik figuralardan 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;

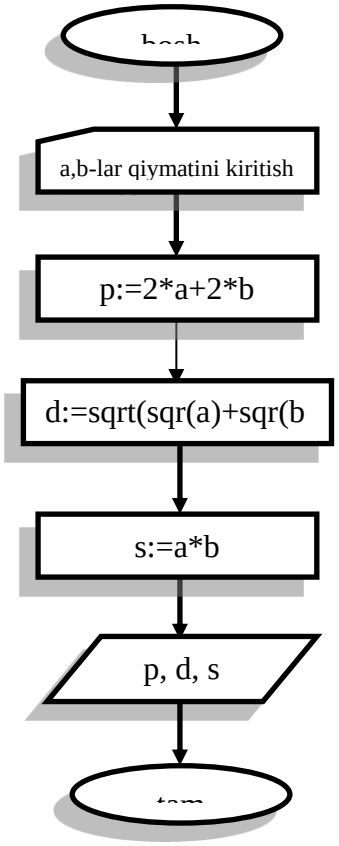
Chiziqli algoritmlarni blok-sxemasi

Hech qanday shart tekshirilmaydigan va tartib bilan faqat ketma – ket bajariladigan algoritmlar ***chiziqli algoritmlar*** 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.

Masalan, 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 dioganperli=’,d);

WriteLn(‘to‘rtburchak yuzasi=’,S);

End.

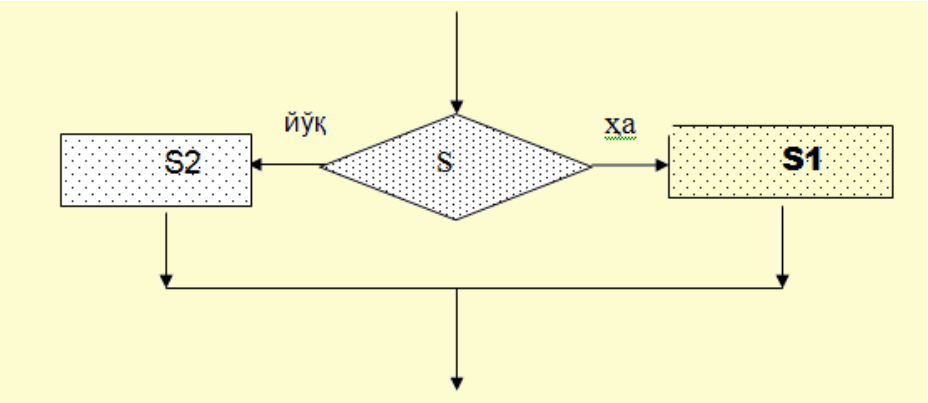
Tarmoqlanish va o‘tish operatorlari.

Turli masalalarni yechganda ko‘rsatmalarni bajarish tartibi biror bir shartning bajarilishiga bog‘liq holda bajariladi. Ya’ni algoritm tarmoqlanadi. Tarmoqlanish «echim» 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 S1 yoki S2 operatorlaridan birini bajarilishiga olib keladi, ya’ni agar S mantiqiy ifoda bajarilishidan so‘ng *true* (rost) qiymati hosil bo‘lsa S1 operatori, aks holda esa S2 operatori bajariladi.

To‘liq shartli operatorga doir misollar:

if a=2 *then* d: = x+2 *else* d: = x-2;

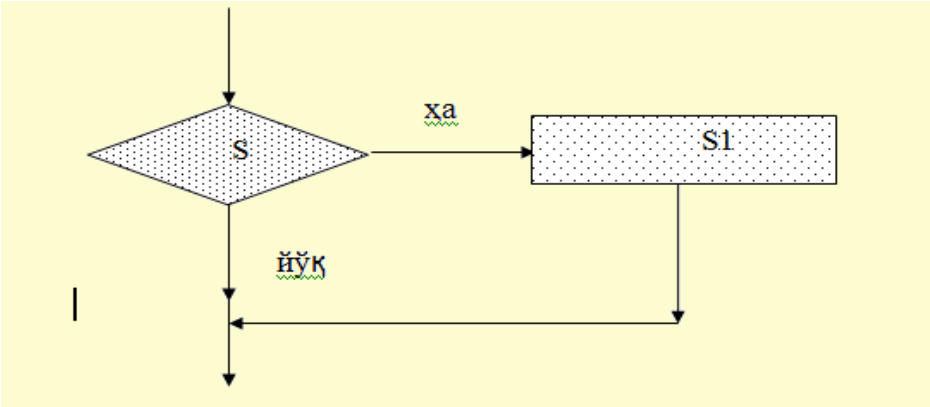
if (x<y) and (z>5) *then begin* y: = x * sin(x);

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 S1;

bu yerda S - mantiqiy ifoda, S1 - operator.

Agar S ifoda qiymati *true* (rost) bo‘lsa S1 operatori bajariladi, aks holda esa boshqarish shartli operatordan keyin yozilgan operatorga uzatiladi.

Shartli operatordan foydalanishga misollar keltiramiz.

1-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.
```

2-misol. Kiritiladigan ixtiyoriy a,b,c sonlar uchun a+b>c, a+c>b, b+c>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.

(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:

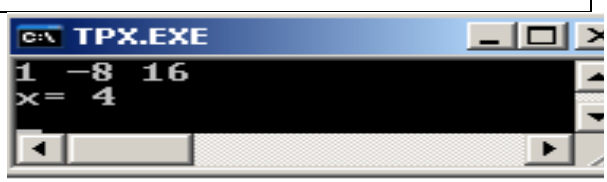
goto belgi;

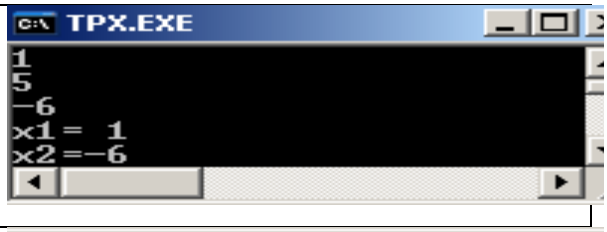
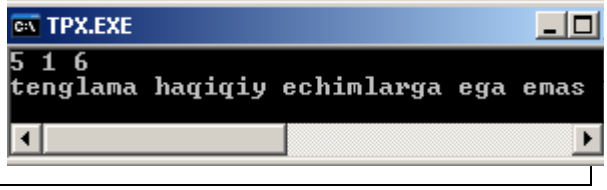
Bu yerda goto xizmatchi so‘z bo‘lib, belgi operator boshqarishni uzatishi zarur bo‘lgan (belilangan) «manzili» hisoblanadi. Belgi sifatida Turbo Paskal dasturlash tilida 0 dan 9999 gacha bo‘lgan butun sonlardan va simvollar birikmasidan(xizmatchi so‘zlardan tashqari) foydalanish mumkin. Belgilar dasturning 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 echimlarga ega emas');
13:end.
```

Keltirilgan dasturni turli holatlarda tenglamning 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-kalid> of <tanlash holatlari ro'yxati-elementlari> else <operatorlar >end;

Bu yerda Case, of, else va end paskalning xizmatchi so'zlari; <tanlash indeksi-kalid> - sonli, belgili yoki matiqiy o'zgaruvchi yoki ifoda; <tanlash holatlari ro'yxati-elementlari> - tanlash indeksi-kalitiga mos qiymatlar.

Tanlash indeksi sifatida haqiqiy tipdan foydalanish mumkin emas va bu indeks tanlash holatlari ro'yxatidagi mos buyruqlni bajarilishini ta'minlaydi. Case operatoridan foydalanishni quyidagi misollarda ko'rib chiqamiz:

1-misol. «Sadaf» kichik tadbirkorlik firmasi bir kecha kunduzda W kVt/soat elektr energiyasini sarflaydi. Bu firmaning 2011 yining 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:

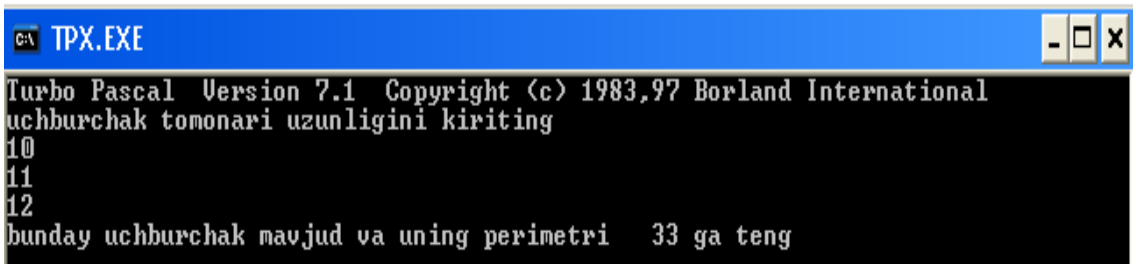
```
case j of                                   Kiritilgan oy tartib raqami j-
      1,3,5,7,8,10,12: r:=31*W;             ga ko‘ra tanlash boshlanadi(oydagi
            4,69,11: r:=30*w;               kunlar miqdoriga nisbatan) va
            2: if (yil mod 4=0) then r:=29*W shunga mos R-ning qiymati
else r:=28*W;                               xisoblanadi
      else writeln('oy tartib raqami xato
kiritilgan')
      end;
```

2-misol. Uchburchak tomonlarini ifodalovchi uchta son kiritilib bu sonlar uchburchak tomonlari bo‘lganda uning perimetrinida 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.
```

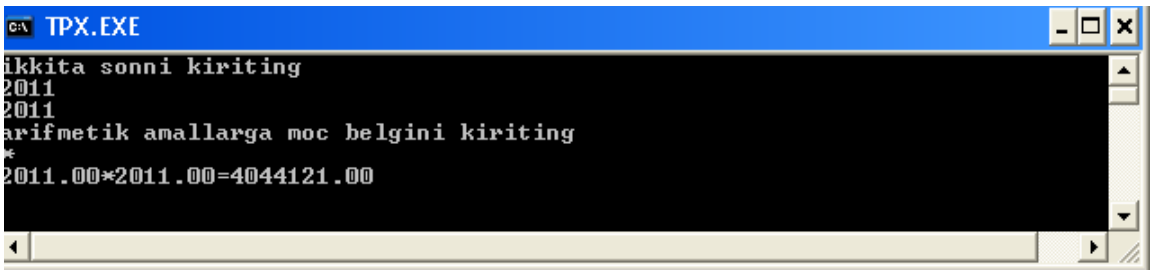
Dastur natijasi:



3-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 moc 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:



Parametrik takrorlash operatori.

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.

Parametrik sikllarning umumiy ko‘rnishi quyidagicha:

For <Sikl parametri>:=<a> to <b> do
< operator yoki operatorlar>

Bu yerda a-parametr bosh qiymatiga, b-parametr oxirgi qiymatiga teng.

1-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’ ortttirish mumkin bo‘lib, bunda «to» o‘rniga «downto» ishlatiladi.

1-misol:

For k:=30 downto 1 do
begin W:=W+sqr(k); R:=r+sqrt(k); end;

<operator> ko‘rinishidagi sikl bo‘lib, ayrim xollarda ham ishlatiladi. Sikl parametrining qiymati faqat butun sonlardan iborat va sikl qadami doimo birga teng.

Parametrik sikllarning o‘ziga xos xususiyatlari quyidagilardan iborat:

For siklidan takrorlanishlar soni aniq bo‘lgan hollarda foydalanish maqsadga muvofiqdir.

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 takorlanishlar «ichma-ich» joylashishlari ham mumkin va bu holat juda ko‘p masalalarni yechishda qo‘llaniladi.

Masalan:

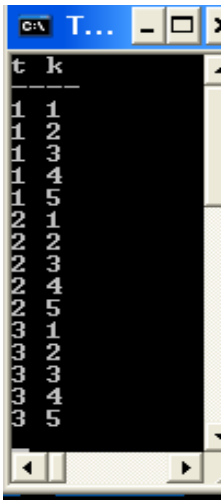
```
for t:=1 to 3 do
for k:=1 to 5 do
writeln(t,k);
```

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 vahokazo.

```
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.
```



2-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:

3-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)>

Bu yerda <operatorlar ketma-ketligi> bajarilishi lozim bo‘lgan amallar yoki sikl tanasida joylashagan operatorlar majmui, <shart> takrorlanishi, bajarilishi yoki to‘xtatilishini boshqaruvchi shartdan iborat. Bu xil ko‘rinishdagi sikl hech bo‘lmaganda bir marotaba bajariladi, negaki operatorlar ketma-ketligi shartni tekshirishdan oldin yozilgan.

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.

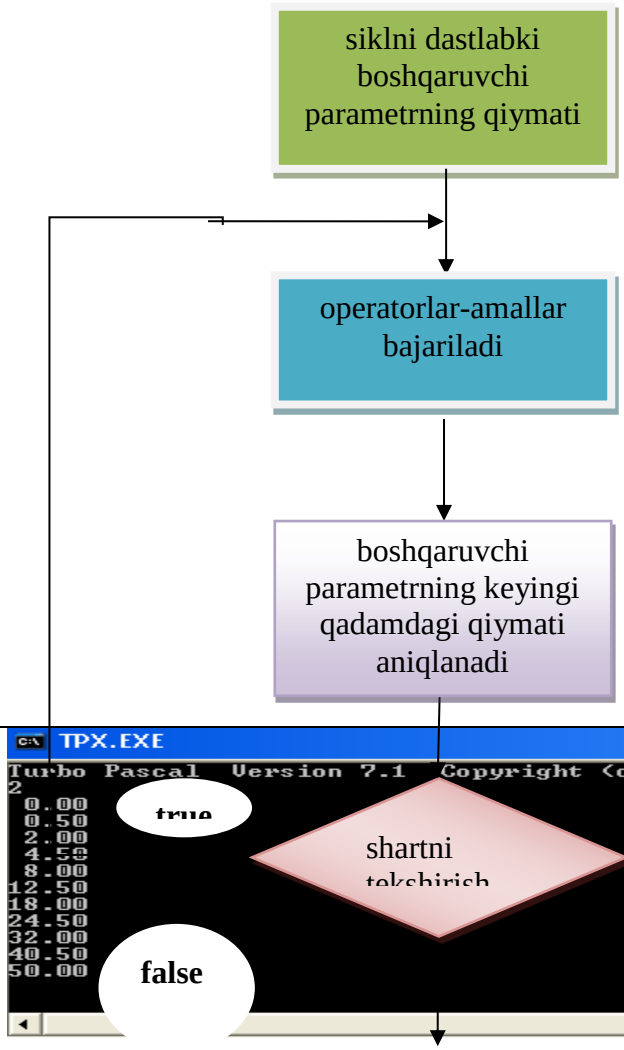
Masala shartiga ko‘ra, foydalanuvchi faqat a-ning qiymatini kiritishi dastur esa $y=ax^2$ funksiya qiytmatini 0,5 qadam bilan hisoblashi zarur. a=2 qiymat uchun natija quyidagicha bo‘lishi zarur, ya’ni dastavval x=0 da funksiya qiymati xisoblanishi (chop etilishi), so‘ngra x-ning qiymati 0,5 ga ottirilishi va hosil bo‘lgan qiymat 5-dan katta bo‘lmasligi tekshirilishi zarur (quyiidagi 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 shatriga mos dastur quyidagicha:

```
var x,a,y:real;
begin
readln (a);
x:=0;
repeat
y:=a*sqr(x);
x:=x+0.5;
writeln(y:5:2);
 until (x>5)
end.
```

Dastur natijasi:

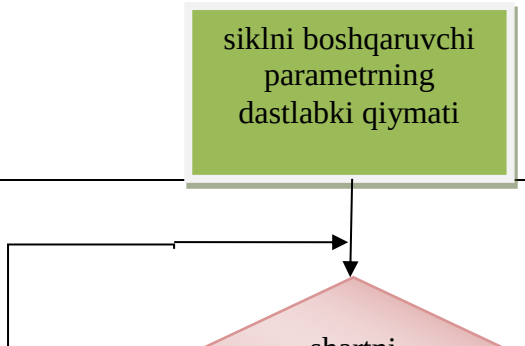
Repeat takrorlash operatori blok-sxemasi	Dastur
	var x,a,y:real; begin readln (a); x:=0; repeat y:=a*sqr(x); writeln(y:5:2); x:=x+0.5; until (x>5) end.

b) while sikli(takrorlanadigan amallar bajarilishi uchun avval shart tekshiriladi). Bu takrorlashning tuzilishi quyidagicha:

While <mantiqiy ifoda (shart)> d o

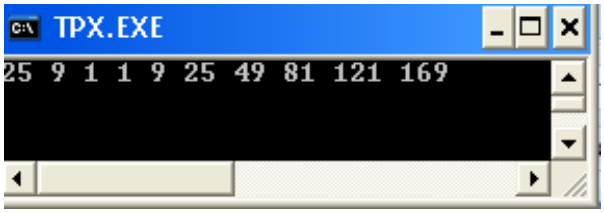
<operator – sikl tanasi>

Bu yerda mantiqiy ifoda (shart) qiymati True bo‘lguncha sikl tanasidagi operatorlar bajariladi, aks hollarda sikl tanasidagi amallar bajarilmaydi.

While shartli tarorlash blok-sxemasi	Dastur-misol
	var x:integer; begin x:=-5; while (x<15) do

true false	begin write(sqr(x), ' '); x:=x+2; end; end.

Dastur natijasi:



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 mukinmi?
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 tushintiring.
9. While operatorining ishlashini tushintiring.
10. Repeat operatorining While operatoridan farqi nimada?

FOYDALANILGAN ADABIYOTLAR

Фаронов В. В. Turbo Pascal. — СПб.: BXB- Санкт-Петербург, 2004. – 1056 с. 15-40 сс, 31-40 сс.

2. Слинкин Д.А.Основы программирования на Турбо-Паскале: Учебно-методическое пособие для студентов вузов. Шадринск: Изд-во Шадринского пединститута, 2003. - 244 с. 45-54 сс.

3. М.У.Ашуров, Н.Д.Мирзахмедова. Turbo Pascal дастурлаш тили.(услубий қўлланма),Тошкент ТДПУ – 20116- 10. 14-17, 36-38, 50-70 betlar.

13-mavzu. Paskalda massivlar

REJA:

1. Massiv haqida tushuncha
2. Chiziqli massivlar
3. Ikki o’lchovli 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 tanishsiz. Odatda jadvalni tashkil etuvchilar uning elementlari deb yuritiladi. Jadvallar birn o’lchovli, ikki o’lchovli, uch o’lchovli va ko’p o’lchovli b o’lishlari mumkin. Hayotda ko’proq bir o’lchovli (chiziqli) va ikki o’lchovli(to’g‘ri to’rtburchakli) jadvallar ko’proq qo’llaniladi.

Avtomobil soatiga 60 km/soat tezlik bilan harakatlansa, uning dastlabki 8 soatda bosib o’tgan y o’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 b o’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

						2		5		8		1		4		7		
4		4		8		1 2		1 0		2 4		2 8		2 2		3 6		3
5		5		0		1 5		1 0		2 5		2 0		3 5		3 0		4 5		4
6		6		2		1 8		1 4		2 0		3 6		3 2		4 8		4 4		5
7		7		4		1 1		2 8		2 5		3 2		4 9		4 6		5 3		6
8		8		6		1 4		2 2		3 0		4 8		4 6		5 4		6 2		7
9		9		8		1 7		2 6		3 5		4 4		5 3		6 2		7 1		8

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:

Massiv-jadval ko'rinishidagi miqdor bo'lib, u ma'lum sondagi bir turli va tartiblangan elementlar majmuasidan iborat.

1. Tiplar yordamida. Bu holda tip-massiv quyidagicha aniqlanadi:

<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;

2. 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 b o’lgan a va b massivlar) aniqlangan b o’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)

Misol: Meteorolgik stanцiyada har bir soatda havo temperaturasi o‘lchanib alohida jadvalga to‘ldirilib boriladi:

O‘lchash vaqti, s							… ……	3	4
Temperatura, S	2	3	1	5	0	2	… ……	1	1

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.

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	Gulrux	Zuhra	Fotima
19	Rustam	Mansur	Botir	Farrux	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 nuqtai 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:

						- elementlar indeksi

2			**2**	**2**	**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.

O‘zbekiston futbol jamolarining turnir natijalari jadval k o’rinishida keltirilgan. Agar uni FF masssivi 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’

№	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. Shungra 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, цeziy) 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 xisoblanadi. 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.

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.
```

1- Masala. Elementlari tasodufiy yaratilgan butun sonlardan iborat bo‘lgan 15 elementli A va B massivlardan kuyidagi shartni qanoatlantiruvchi uchinchi C massivni yarating.

C[i]:=A[i]+B[i], bu erda 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.
```

2-Masala.5-sondan iborat bir o‘lchamli D massiv berilgan. Bu massivning xar bir mos elementining 10 foieidan 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?

FOYDALANILGAN ADABIYOTLAR

1. Фаронов В. В. Turbo Pascal. — СПб.: BXB- Санкт-Петербург, 2004. – 1056 с. 41-44 сс.

2. Слинкин Д.А.Основы программирования на Турбо-Паскале: Учебно-методическое пособие для студентов вузов. Шадринск: Изд-во Шадринского пединститута, 2003. - 244 с. 71-76 сс.

3. М.У.Ашуров, Н.Д.Мирзахмедова. Turbo Pascal дастурлаш тили.(услубий қўлланма), Тошкент ТДПУ – 20116- 10. 25-31 betlar.

14-MAVZU. PASCALDA SATRIY KATTALIKLAR BILAN ISHLASH

REJA:

1. **String tipidan matnli ma’lumotlar**
2. **Satriy kattaliklar ustida funksiya va proseduralar yordamida bajaraladigan amallar**
3. **Satriy kattalikka doir masalalar**

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];
```

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 aniqlanadilar.

Misol:

Satr:=’Geologiya’ bo’lsa, satr[1]=’G’-ga, satr[2]=’e’-ga va h.

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 dasturni 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.
```

Endi satriy kattaliklar ustida ayrim funksiya va proseduralar 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 catriy 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
‘’’’	Length(St)	1
‘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 a2 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 ifodalay 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 proseduralarni mustaqil o’rganing va misollar ko’rib chiqng:

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 darjali 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:

Ифодалар	Натижа
‘True1’<’True2’	True
‘Mother’>’MOTHER’	True
‘Geografiya’>’Geologiya’	False
‘Cat’=’Cat’	True
‘uzunlik’>’balandlik’	True

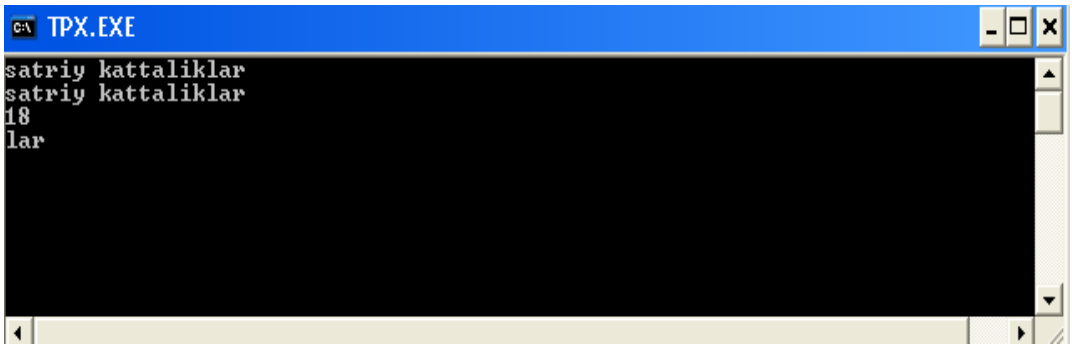
Yuqorida keltirilgan funksiya va protseduralardan dasturlarda qo‘llashga miollar keltiramiz:

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;**	**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.**	**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"lchovida 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 ('qiymatimavjud 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 tekshirb to'g'ri tuzilganiga ishinch 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 kirining');
   readln(s);
  while (pos(' ',s)=1) and (length(s)>0) do
  delete(s,1,1);
  write(Natija',s);
end.
```

MUSTAHKAMLASH UCHUN SAVOLLAR

7. Paskal dasturlash tilida matnli ma'lumotlar bilan ishlashda qanday tipidan foydalaniladi?
8. Concat funksiyasini vazifasi nimadan iborat? Misollar keltiring.
9. Copy funksiyasi nima uchun qo'llaniladi?
10. Delete protsedura nima va unga misol keltiring.
11. Length funksiyasi nima ish bajaradi, misol bilan tushuntiring.
12. Pos funksiyasi qanday vazifani bajaradi va qachon qiymati nolga teng bo'ladi.
13. Val protsedurasini vazifasini tushuntiring.
14. Paskalda berilgan satr ichiga boshqa satrni joylashtirishning imkoni bormi? Javobingizni tushuntiring.

15. Sonli miqdorni satrli miqdorga o‘tkazib bo‘ladimi? Javobingizni izohlang.

FOYDALANILGAN ADABIYOTLAR

4. Фаронов В. В. Turbo Pascal. — СПб.: BXB- Санкт-Петербург, 2004. – 1056 с. 21-23 сс.

5. Слинкин Д.А.Основы программирования на Турбо-Паскале: Учебно-методическое пособие для студентов вузов. Шадринск: Изд-во Шадринского пединститута, 2003. - 244 с. 76-80 сс.

6. М.У.Ашуров, Н.Д.Мирзахмедова. Turbo Pascal дастурлаш тили.(услубий қўлланма),Тошкент ТДПУ – 20116- 10. 74-79 betlar.

15-MAVZU. QISM DASTURLAR-FUNKSIYALAR

REJA

1. **Qism dastur-funsiya tuzilishi**
2. **Funksiya tarkibidagi parametrlar**
3. **Funksiyadan foydalanishning o'ziga xos xususiyatlari**

Funksiya va protseduralar maxsus tuzilish va nomga ega bo'lgan dasturning mustaqil qismlari hisoblanishadi. Dastur matnida bu nomlardan foydalanish funksiya va protseduralarga murojjat qilishni anglatadi. 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 murojjat qilish usuli bilan birnecha marotaba foydalanisn mumkin. (yaratilgan qism dasturlar yordamida modullar tarkibida foydalanish uyqoru samaralar berishi bilan k…….)

Dastur tarlkibida qimdasturlar joylashishuni qyuidagicha tasvrlash mumkin:

Funksiya bu dasturning bir qismi bo'lib, *function* so'zi bilan boshlanib, quyidagi tuzilishga ega:

function **<funksiyaning nomi> (<formal parametrlar ro'yxati>): <tip>;**

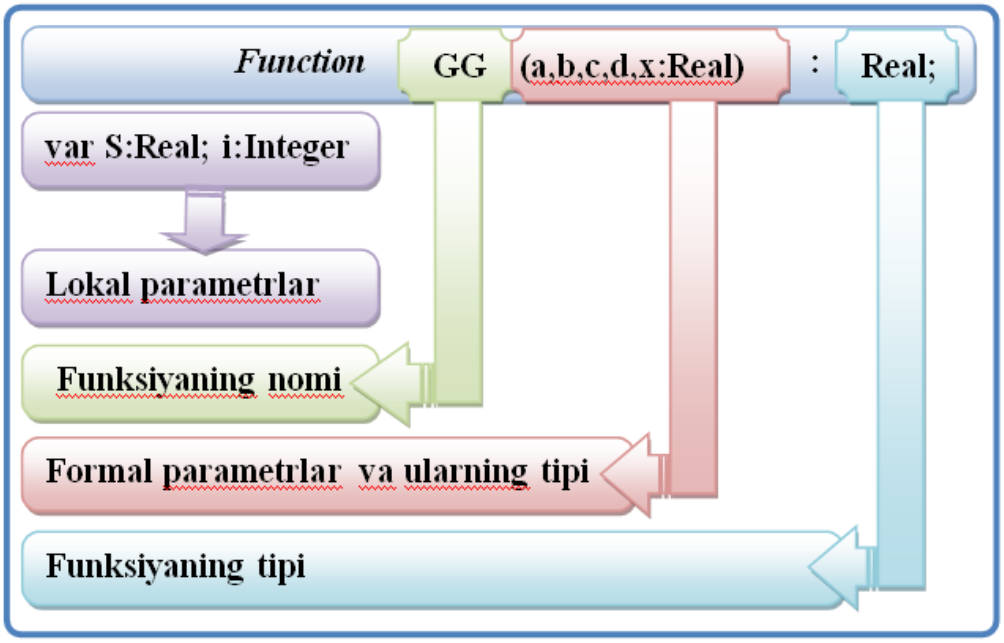
bu yerda tip funksiya qiymatining tipi.

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,c,d) 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.

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 funksiyani 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 funksiyani tipi
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.	

-masala.

Funksiya yordamida a, b, c 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,c –sonlar uchburchak tomonlari bo'la olishsa, True aksincha False qiymatlarini beruvhi 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,c 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 “teskarilovchi” funksiyadan foydalanib, kirtilgan 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.
```

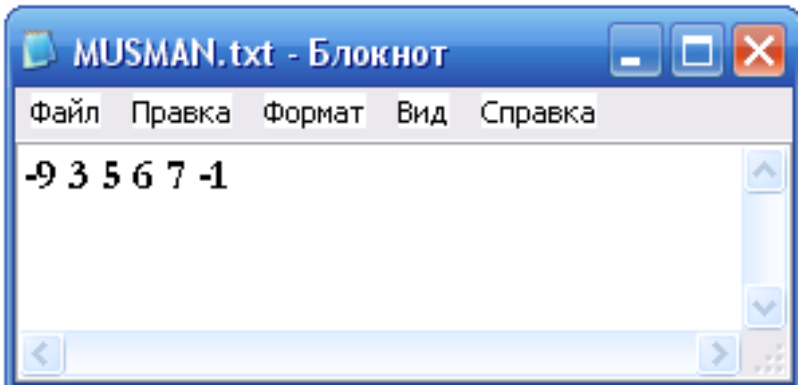
11-masala.

x va y- haqiqiy sonlar uchun x^y –qiymatini hisoblovchi dastur tuzing.

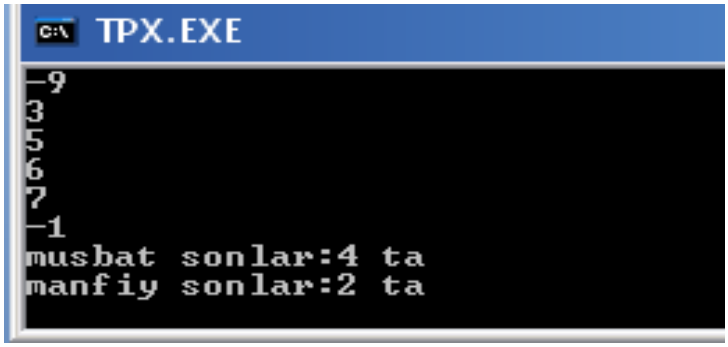
```
var a,b:real;
label 5;
function daraja(a,b:real):real;

  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.
```



ASOSIY



ADABIYOTLAR:

4. Слинкин Д.А.Основы программирования на Турбо-Паскале: Учебно-методическое пособие для студентов вузов. Шадринск: Изд-во Шадринского пединститута, 2003. – 108-110 сс.

5. Фаронов В. В. Turbo Pascal. — СПб.: BXB- Санкт-Петербург, 2004. – 1056 с. (44-51сс)

6. M.U.Ashurov, N.D.Mirzaxmedova. Turbo Pascal dasturlash tili.(uslubiy qo‘llanma),Toshkent TDPU – 2011 (102-112)

16-MAVZU. QISM DASTURLAR- PROTSEDURALAR

REJA

1. **Qism dastur- protseduralar tuzilishi**
2. **Protseduralar tarkibidagi parametrlar**
3. **Protseduradan foydalanishning o'ziga xos xususiyatlari**

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 dasturdan foydalaniladi.

Protsedura sarlavhasi quyidagi tuzilishga ega:

Procedure <protsedura nomi> (<formal parametrlar ro'yxati >);

Prosedura 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).

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;

v) berilish kema-ketligi mos bo'lishi.

(1 formal parametr 1-haqiqiy parametr bilan, 2-formal parametr 2-haqiqiy parametr bilan va h.).

1-masala.

Prosedura tarkibda umuman parametrdan foydalanmaslik mumkin va shu holat uichun misol keltiramiz:

```
procedure A;
begin writeln('**********');end;
begin A;end.
```

2-masala.

X,y –haqiqiy sonlar. Qism dastur–prosedura yordamida ularga berilgan qiymatlarni almashriruvchi 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 prosedura 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 prosedura 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.

Prosedura 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.
```

Mustahkamlash uchun savollar

1. Qism dastur - funktsiyaning berilishi va funktsiyaning nomi.
2. Funktsiyaning 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 funktsiyadan farqi.
8. Protseduraning o’ziga xos xususiyatlari

ASOSIY ADABIYOTLAR:

7. Слинкин Д.А.Основы программирования на Турбо-Паскале: Учебно-методическое пособие для студентов вузов. Шадринск: Изд-во Шадринского пединститута, 2003. – 108-110 сс.
8. Фаронов В. В. Turbo Pascal. — СПб.: BXB- Санкт-Петербург, 2004. – 1056 с. (44-51сс)
9. М.У.Ашуров, Н.Д.Мирзахмедова .Turbo Pascal дастурлаш тили.(услубий қўлланма),Тошкент ТДПУ – 2011 (102-112)

17-MAVZU. PASCALDA FAYLLAR BILAN ISHLASH

REJA:

1. **Fayllar haqida tushuncha**
2. **Fayllarga yozish va ulardan o’qish uchun mo;ljallangan funktsiya va protseduralar**
3. **Fayllar ustida amallar bajarish**

Fayl deb, kompyuter tashqi xotirasining nomlangan soxasiga aytiladi va uning uchta xarakterli xususiyatlari mavjud. Birinchidan, uning nomi mavjud bulib, bu nomdan dasturda foydalaniladi. Ikkinchidan uning kompponentlari 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 xajmiga bog‘liq.

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.

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:

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);

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.

AssignFile (<fayl o‘zgaruvchisi >,< fayl nomi >);- bu protseduraning 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 uzgaruvchisi >);

Izoh: fayl o‘zgaruvchisi –avval Assign protsedurasi yordamida mavjud fayl bilan bog‘langan bo‘lishi lozim.

Bu protsedura 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 qushimcha ma’lumotlarni yozish imkoniyatini beradi.(faqat Text tipidagi fayllar uchun qo‘llaniladi)

Close (<fayl uzgaruvchisi >) protsedurasi faylni yopish uchun qo‘llaniladi, ammo fayl o‘zgaruvchisi bilan bog‘lanish o‘z kuchini saqlaydi.

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.

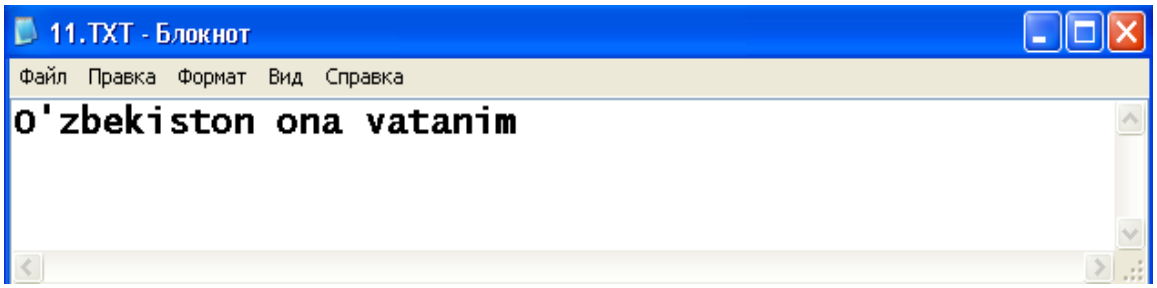
Erase (<fayl uzgaruvchisi >) fayli o‘chiradi.

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.
```

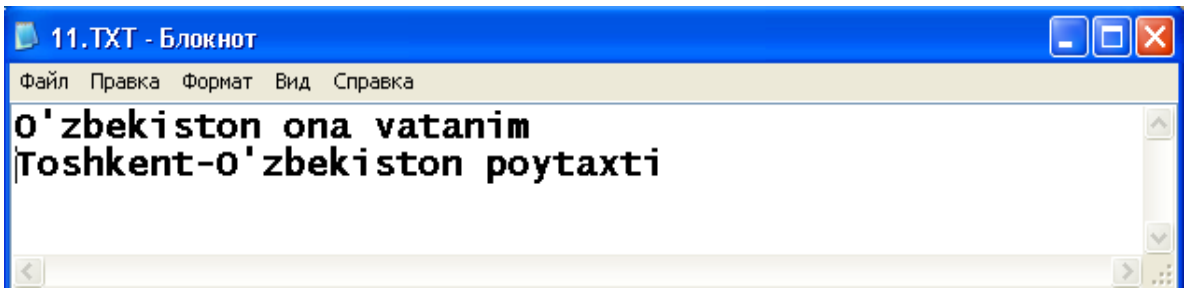
Dastur natijasi(faylga yozish amali bajariladi)::



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.
```

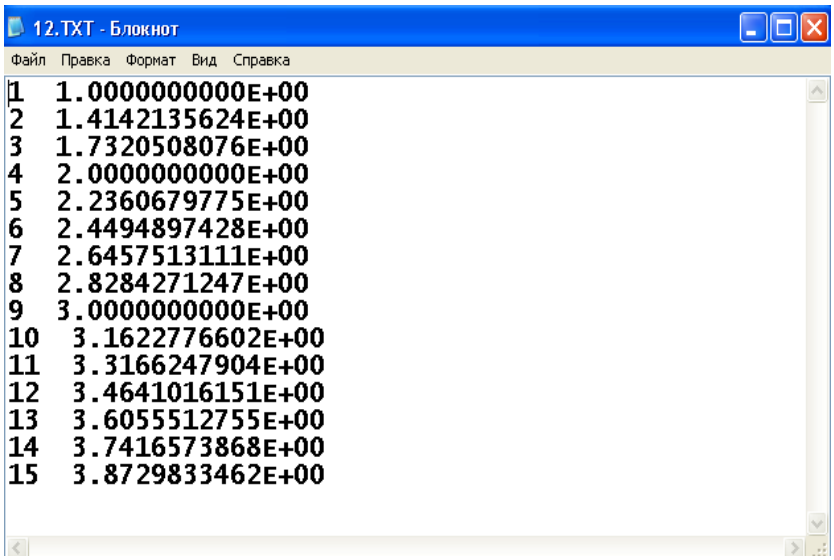
Dastur natijasi(faylga yozish amali bajariladi):



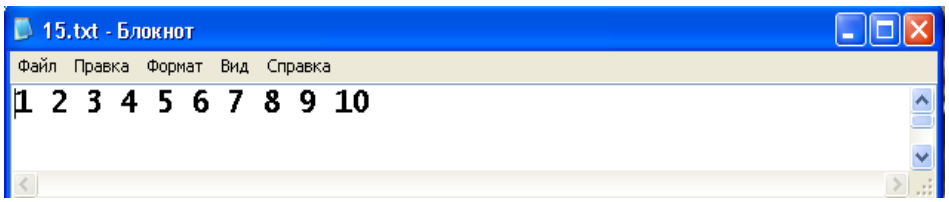
3-misol.

```
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):



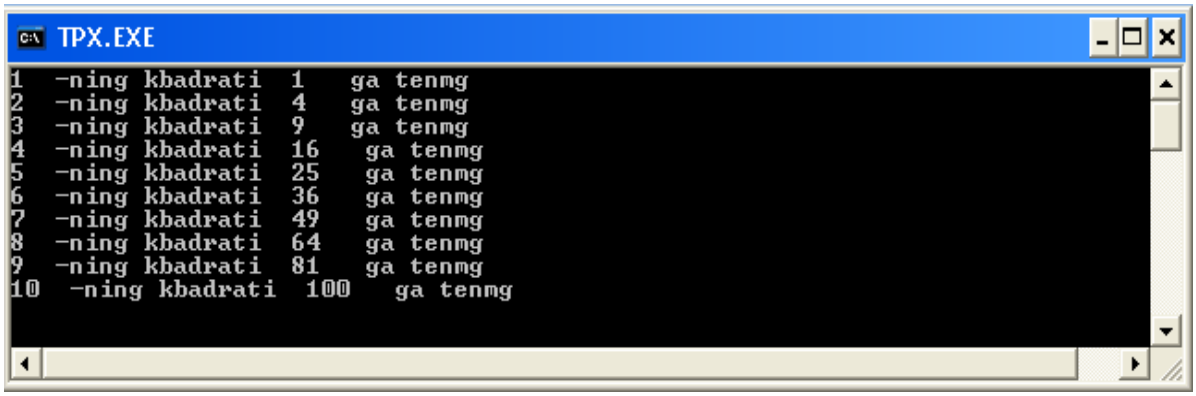
Fayldan o‘qish amali bajarilishi kuzatish uchun avval S mantiqiy diskda (yoki boshqa manbada) o‘qish uchun faylni tayyorlaymiz. Buning uchun «Bloknot» dasturida «15.pas» faylini quyidagi ko‘rinishda tayyorlab olamiz:



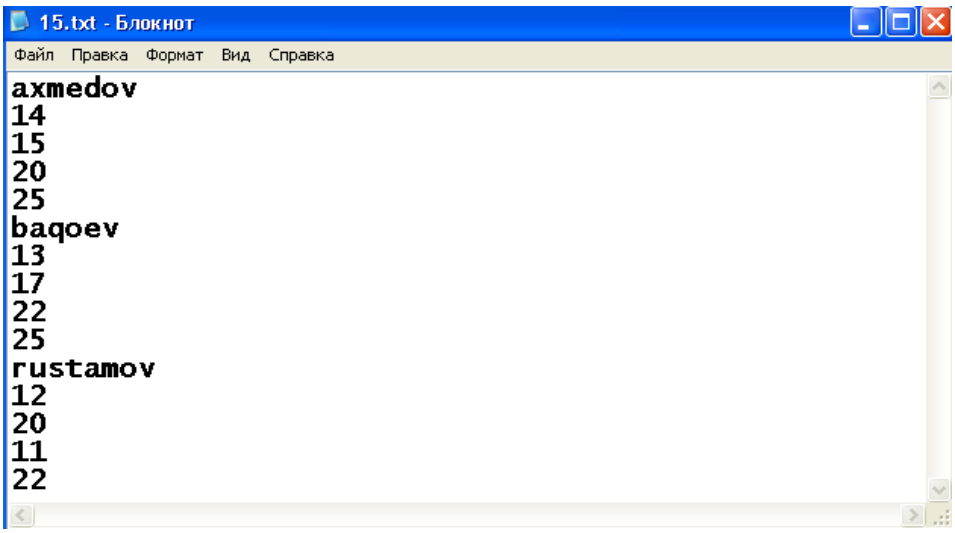
4-misol. S mantiqiy diskdagi «15.pas» faylidan 10 ta soni o‘qib ularning kvadratlarini monitorda aks ettiruvchi dastur yarating.

```
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‘kuvchining 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)

Mustaqil bajarib ko‘ring.

1.

```
var f:text;
x,i:integer;
begin
mkdir ('c:/v1');{yangi katalog ochish uchun}
```

```
assign  ( f,'c:/v1/12.txt');
rewrite(f);
writeln(f,sqrt(2011));
 close(f);
        end.
2.
var f:text;
x,i:integer;
begin
assign  ( f,'c:/v1/12.txt');
rewrite(f);
writeln(f,sqrt(2011));
 close(f);
 rename(f,'c:/v1/y1.txt');
        end.
```

Mustahkamlash uchun savollar:

1. Fayl turdagi to‘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 xodisa yuz beradi?
6. Close operator nima uchun qo‘llaniladi?
7. Append protsedurasini vazifasini aytib bering.
8. Fayldagi ma’lumotlarni ochish uchun qaysi operator yordamida ochiladi.
9. EOF funksiyasini vazifasini aytib bering.

ASOSIY ADABIYOTLAR:

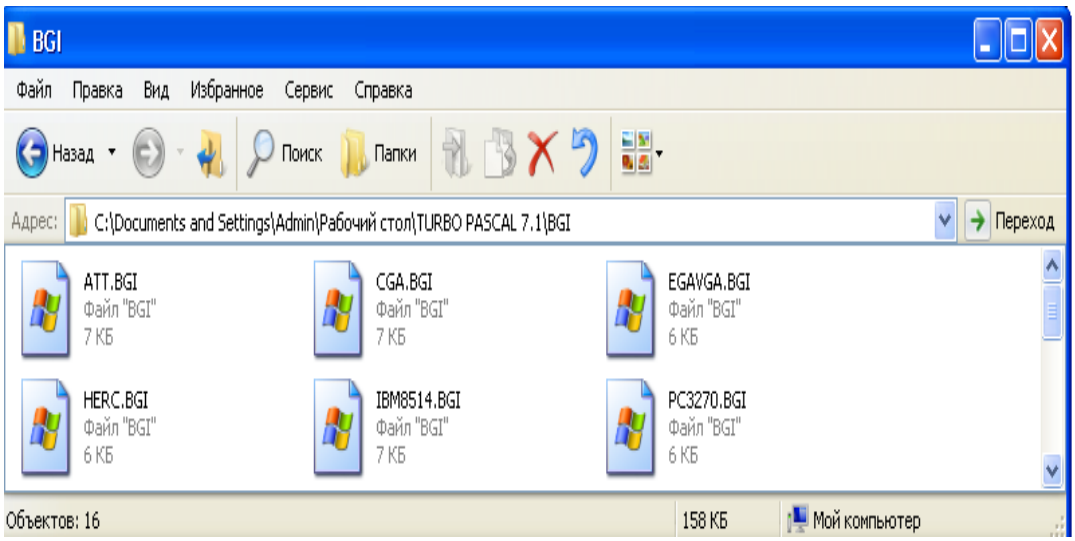
10. Слинкин Д.А.Основы программирования на Турбо-Паскале: Учебно-методическое пособие для студентов вузов. Шадринск: Изд-во Шадринского пединститута, 2003. – 90-97 сс.
11. Фаронов В. В. Turbo Pascal. — СПб.: BXB- Санкт-Петербург, 2004. – 1056 с. (129-150сс)
12. М.У.Ашуров, Н.Д.Мирзахмедова .Turbo Pascal дастурлаш тили.(услубий қўлланма),Тошкент ТДПУ – 2011 (95-101)

18-MAVZU. GRAFIKA MODULI VA ULAR BILAN ISHLASH

REJA

1. **Graph moduli Pascal tilining asosiy moduli sifatida**
2. **Modulning funktsiya va protseduralari**
3. **Koordinatalar, oynalar va sahifalar.**

Turbo Pascal dasturini ishga tushirish bilan ekran matnli holatda bo‘ladi va agarda yaratilayotgan dasturda grafikli ob’ektlardan foydalanish nazarda tutilsa displey adapterini xrafikali holatga moslashish zarur bo‘ladi. Turbo Pascal dasturlash tilining 4.0 versiyasidan boshlab uning paketi tarkibiga grafikli ob’ektlar 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 nuqtalar kabi tasavvur etish mumkin va bu nuqtalar turli ranglarda «yoritilib» tasvir hosil bo‘lishiga xizmat qilishadi. Ayrim adapterlarning grafikali holatlari bilan tanishamiz:

SGA adapteri (Color Graphics Adapter-rangli grafikali adapter). Bu adapter 5 ta grafik holatni qo‘llab-quvvtalaydi. Ularning 4-tasi ekranning quyi imkoniyatlari (gorizontal yo‘nalish bo‘yicha 320 ta piksel (nuqta), vertakal 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- xam 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 garfikali 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 porotsedurasi adpterning 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.
```

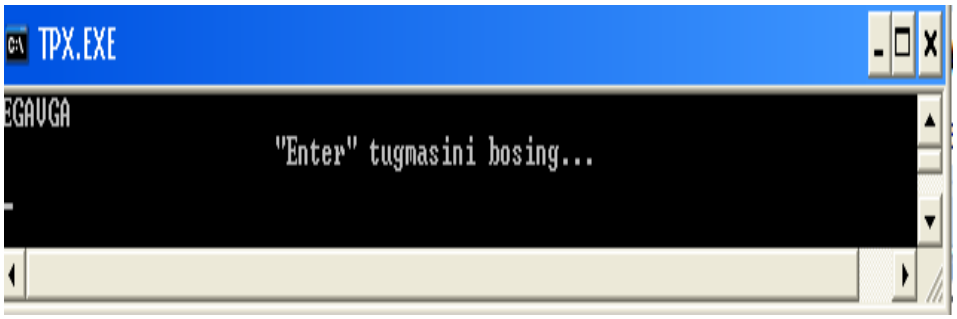
Dasturda keltirilgan «grok»(qiymati 0 ga teng bo‘lgan doimiylik) Graphresult funksiyasini natijasi bo‘lib, protseduralarga murojaat qilishda xatolikga yo‘l qo‘yilmaganligini tekshirishda foydalaniladi.

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.
```

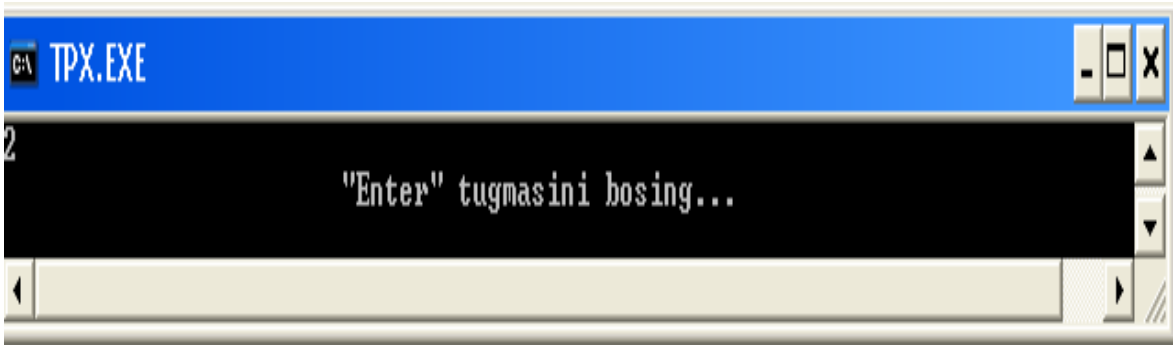
Dastur natijasi yuklangan grafikali drayverga bog‘liq va u quyidagicha bo‘lishi mumkin:



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.

Kursor-ko‘rsatgich o‘rnini integer tipidagi qimatlarni 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 shaklidagi 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.

SetbkSolor(n)- protsedurasi ekran fonini ko‘rsatilgan n-parametrga mos keluvchi (0 dan 15 gacha) rang bilan berilishini ta’minlaydi.

ClearDevice- protsedurasi grafik ekranni toazlab, uning fonini SetbkSolor 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‘rsagich 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 xolatdan chiqish uchun qaysi operatordan foydalaniladi?
5. Ekranni grafik holatga o‘tkazilganda yurgichning ko‘rinishi qanday bo‘ladi?
6. Grafik xolatda ekranning qaysi nuqtasi joriy nuqta bo‘ladi?
7. GetMaxX va GetMaxY funksiyalarini vazifalarini aytib bering.

Asosiy adabiyotlar:

13. Слинкин Д.А.Основы программирования на Турбо-Паскале: Учебно-методическое пособие для студентов вузов. Шадринск: Изд-во Шадринского пединститута, 2003. – 134-136 сс.

14. Фаронов В. В. Turbo Pascal. — СПб.: BXB- Санкт-Петербург, 2004. – 1056 с. (301-320сс)

15. М.У.Ашуров, Н.Д.Мирзахмедова .Turbo Pascal дастурлаш тили.(услубий қўлланма),Тошкент ТДПУ – 2011 (81-87)

19-MAVZU. PASKAL TILIDA SODDA SHAKLLAR CHIZISH

REJA:

1. **Sodda shakllarni chizuvchi protseduralar va ulardan foydalanish**
2. **Shakllarni bo'yash va shablon bilan to'ldirish**

Grafikali muhitda ob'ektlarni 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 ekarnda (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 nuqtalarani 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-doimiylikalr 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: domiyliklar 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 belilaydi. 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 shlatiladi. *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 belgilayd i. 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‘rtburchakli 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 etirilsa, 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?

ASOSIY ADABIYOTLAR:

16. Слинкин Д.А.Основы программирования на Турбо-Паскале: Учебно-методическое пособие для студентов вузов. Шадринск: Изд-во Шадринского пединститута, 2003. – 146-151 сс.

17. Фаронов В. В. Turbo Pascal. — СПб.: BXB- Санкт-Петербург, 2004. – 1056 с. (329-341сс)

М.У.Ашуров, Н.Д.Мирзахмедова .Turbo Pascal дастурлаш тили.(услубий қўлланма),Тошкент ТДПУ – 2011 (88-94)