O'ZBEKISTON RESPUBLIKASI OLIY VA O'RTA MAXSUS TA'LIMI VAZIRLIGI
OLIY VA O'RTA MAXSUS, KASB HUNAR TA'LIMI MARKAZI

P. KARIMOV, S. IRISQULOV, A. ISABOEV

DASTURLASH

Oliy va o'rta o'quv yurtlari uchun uquv qo'llanma

Toshkent 2003 y

22.18
K25

Mazkur uquv qullanma O'zbekiston Respublikasi Oliy va o'rta maxsus talim vazirligining "Elektron hisoblash mashi nalari va tarmoqlari montaji" tarmoq ta'limi standarti va namunaviy dastur asosida yozilgan bo'lib unda algaritmlar, masalani EHM da echish bosqichlari, dasturlari tillari PASKAL, Si uchun dasturlar yaratish haqida ma'lumot lar kiritilgan.

Taqrizchilar: E.G.Hasanov, K.Quziev, M.E.Zayniddinova

Muxarrir: Yu. Muzaffarxo'jaev

ISBN 5-640-03173-5
24010000-37
D----------
M 351(04) 2003

"O'zbekiston" nashriyoti,
2003 y.

So'z boshi

Aziz o'quvchi! Eng avval shuni aytib o'tish joizki, siz ajib bir zamonda yashamoqdasiz. Bu zamonning eng muhim xususiyati xalqimizning istiqlol sharoitida hayot kechirayotgani va ijtimoiy hayotning barcha jabhalarida chuqur inqilobiy o'zgarishlar sodir bo'layotganidir. Respublikamiz Prezidenti I.Karimov XXI asrni ma'naviyat va ma'rifat asri, ilm-fan, madaniyat va axborotlar asri deb ta'riflar ekan, bu bilan hozirgi davrning muhim belgisi va xususiyatini ham belgilab berganini payqab olish qiyin emas.

O'zbekiston darhaqiqat kelajagi buyuk davlat. Biroq bu kelajakning qay darajada yaqin yoki uzoqligi, mamalakatimizni qachon jahonning rivojlangan davlatlari qatoridan o'rin olishi, qolaversa xalqimizning erkin va farovon hayot kechirishi hal qiluvchi darajada uning aqliy salohiyatiga, yoshlarning zamonaviy bilimlarini, fan, texnika va texnologiyalarning eng so'nggi yutuqlarini o'zlashtirishiga bog'liq.

Mamalakatimizda qabul qilingan Kadrlar tayyorlashning milliy dasturi zamon talablariga to'la-to'kis javob bera oladigan, chinakam ma'rifatli, bozor munosabatlari sharoitida o'z bilim va ko'nikmalari bilan vatan istiqboli manfaatlari yo'lida samarali faoliyat ko'rsata oladigan mutaxassislar tayyorlashni nazarda tutadi.

Hozirgi zamonda inson faoliyatining biron bir sohasini hisoblash texnikasi (komputerlar) va axborot texnologiyalarisiz tassavur qilib bo'lmaydi. Ulardan oqilona va samarali foydalana olish bugungi kunda har bir o'qimishli va ziyoli inson uchun suv va havodek zarur bo'lib qolmoqda. Shu nuqtai nazardan qaraydigan bo'lsak, qo'lingizdagi o'quv qo'llanmasi naqadar dolzarb masala va muammolarni hal etishga bag'ishlangani o'z-o'zidan ayon bo'ladi.

Hozirgi kunda mamlakatimiz ta'lim tizimida yangi bosqich bo'lgan kasb-hunar ta'limini zarur me'yoriy hujjatlar, zamonaviy texnika va texnologiyalar bilan ta'minlash borasida bir qator ishlar

amalga oshirildi va oshirilmoqda. Jumladan, O'zbekiston Respublikasi Vazirlar Mahkamasining qarori bilan «O'rta maxsus, kasb-hunar ta'limining davlat ta'lim standarti» va «O'rta maxsus, kasb-hunar ta'limining yo'nalish fanlar davlat ta'lim standarti» tasdiqlandi. Ularda ko'rsatib o'tilganidek, o'rta maxsus, kasb-hunar ta'lim muassasasi bitiruvchilariga fundamental fanlar va aniq kasb sohasi doirasida nazariy hamda amaliy bilimlarga ega bo'lish, kompyuter va telekommunikatsiya vositalaridan foydalana olish, o'quv fanlari bo'yicha oliy ta'lim muassasalarida tahsil olish uchun zarur bo'lgan bilimlar majmuasiga ega bo'lish vazifasi qo'yilgan. Shu talablardan kelib chiqqan holda, kasb-hunar kollejlari uchun «Dasturlash» fanidan namunaviy o'quv dasturi ishlab chiqildi.

Bugungi kunda kasb-hunar kollejlari va ulardagi ta'lim yo'nalishlarining zamonaviyligi ularning kompyuterlashtirilganlik darajasi bilan belgilanmoqda. Informatika va yangi axborotlar texnologiyalari jamiyatimiz ijtimoiy va iqtisodiy hayotiga shiddat bilan kirib kelmoqda. Shu sababli informatika fanining sir-sinoatlarini o'zlashtirish davr talabiga aylanib qoldi.

O'ylaymizki, yuqorida ko'rsatib o'tilgan dastur asosida tayyorlangan ushbu qo'llanma siz aziz o'quvchilarga dasturlash asoslarini o'zlashtirish, kompyuterdan o'z kasbiy faoliyatlaringizda bemalol va samarali foydalana olish uchun zarur bo'lgan malaka va ko'nikmalarni hosil qilishga xizmat qiladi.

Albatta, mazkur qo'llanma ilk bor yozilganligi bois ayrim kamchiliklardan xoli bo'lmasligi mumkin. Uni yanada takomillashtirishga qaratilgan fikr-mulohazalarni mualliflar mamnuniyat bilan qabul qiladilar.

Iqtisod fanlari doktori, prof. **T. E. Ergashev**

I bob

ALGORITMLAR NAZARIYASI ELEMENTLARI

1.1. ALGORITM TUSHUNCHASI VA UNING XOSSALARI

Algoritm hozirgi zamon matematikasining eng keng tushunchalaridan biri hisoblanadi.

Algoritm (algorifm) so'zi o'rta asrlarda paydo bo'lgan bo'lib, buyuk mutafakkir bobokalonimiz Al-Xorazmiyning (783—855) ishlari bilan yevropaliklarning birinchi bor tanishishi bilan bog'liqdir. Bu ishlar ularda juda chuqur taassurot qoldirib algoritm (algorithmi) so'zining kelib chiqishiga sabab bo'ldiki, u Al-Xorazmiy ismining lotincha aytilishidir. U paytlarda bu so'z arablarda qo'llaniladigan o'nlik sanoq tizimi (sistemasi) va bu sanoq tizimida hisoblash usulini bildirar edi. Shuni ta'kidlash lozimki, yevropaliklar tomonidan arab sanoq tizimining Al-Xorazmiy ishlari orqali o'zlashtirilishiga, keyinchalik hisoblash usullarining rivojlanishiga katta turtki bo'lgan.

Hozirgi paytda o'nlik sanoq tizimida arifmetik amallarni bajarish usullari hisoblash algoritmlariga soddagina misol bo'la oladi xolos. Hozirgi zamon nuqtai nazaridan algoritm tushunchasi nimani ifodalaydi? Ma'lumki, inson kundalik turmushida turli-tuman ishlarni bajaradi. Har bir ishni bajarishda esa bir qancha elementar (mayda) ishlarni ketma-ket amalga oshirishga to'g'ri keladi. Mana shu ketma-ketlikning o'zi bajariladigan ishning algoritmidir. Ammo bu ketma-ketlikka e'tibor bersak, biz ijro etayotgan elementar ishlar ma'lum qoida bo'yicha bajarilishi kerak bo'lgan ketma-ketlikdan iborat ekanligini ko'ramiz. Agar bu ketma-ketlikdagi qoidani buzsak, maqsadga erishmasligimiz mumkin.

Masalan, shaxmat o'yinini boshlashda shohni yura olmaymiz, chunki bu o'yin algoritmida yurishni boshqa bir shaxmat donalaridan boshlash kerak yoki palov pishirish algoritmida birinchi navbatda qozonga suv solib ko'ringchi, osh qanday bo'lar ekan. Berilgan matematik ifodani soddalashtirishda amallarning bajarilish ketma-ketligiga e'tibor bermaslik noto'g'ri natijaga olib kelishi barchaga ma'lum.

Demak ishni, ya'ni qo'yilgan masalani bajarishga mayda elementar ishlarni muayyan ketma-ketlikda ijro etish orqali erishiladi. Bundan ko'rinib turibdiki, har bir ish qandaydir algoritmning bajarilishidan iboratdir. Algoritmni bajaruvchi algoritm ijrochisidir. Algoritmning ijrochisi masalaning qanday qo'yilishiga e'tibor bermagan holda natijaga erishishi mumkin. Buning uchun u faqat avvaldan ma'lum qoida va ko'rsatmalarni qat'iy bajarishi shart. Bu esa algoritmning juda muhim xususiyatlaridan biridir.

Umuman, algoritmlarni ikki guruhga ajratish mumkin. Birinchi guruh algoritmning ijrochisi faqat inson bo'lishi mumkin (masalan palovni faqat inson pishira oladi), ikkinchi guruh algoritmlarning ijrochisi ham inson, ham EHM bo'lishi mumkin (faqat aqliy mehnat bilan bog'liq bo'lgan masalalar). Ikkinchi guruh algorimtlarning ijrochisini EHM zimmasiga yuklash mumkin. Buning uchun algoritmni EHM tushunadigan biror tilda yozib, uni mashina xotirasiga kiritish kifoya.

Shunday qilib, biz *algoritm* deganda, berilgan masalani yechish uchun ma'lum tartib bilan bajarilishi kerak bo'lgan chekli sondagi buyruqlar ketma-ketligini tushunamiz.

Biror sohaga tegishli masalani yechish algoritmini tuzish algoritm tuzuvchidan shu sohani mukammal bilgan holda, qo'yilgan masalani chuqur tahlil qilishni talab qiladi. Bunda masalani yechish uchun kerak bo'lgan ishlarning rejasini tuza bilish muhim ahamiyatga ega. Shuningdek, masalani yechishda ishtirok etadigan ob'ektlarning qaysilari boshlang'ich ma'lumot va qaysilari natijaligini aniqlash,

ular o'rtasidagi o'zaro bog'lanishni aniq va to'la ko'rsata bilish, yoki dastur (programma) tuzuvchilar tili bilan aytganda, masalaning ma'lumotlar modelini berish lozim.

Berilgan masala algoritmini yozishning turli usullari mavjud bo'lib, ular qatoriga so'z bilan, blok-tarh (blok-sxema) shaklida, formulalar, operatorlar yordamida, algoritmik yoki dasturlash tillarida yozish va xokazolarni kiritish mumkin.

Endi biror usulda tuzilgan algoritmning ayrim xossalari va algoritmga qo'yilgan ba'zi bir talablarni ko'rib chiqaylik.

1. Algoritm har doim to'liq bir qiymatlidir, ya'ni uni bir xil boshlang'ich qiymatlar bilan ko'p marta qo'llash har doim bir xil natija beradi.

2. Algoritm birgina masalani yechish qoidasi bo'lib qolmay, balki turli-tuman boshlang'ich shartlar asosida ma'lum turdagi masalalar to'plamini yechish yo'lidir.

3. Algoritmni qo'llash natijasida chekli qadamdan keyin natijaga erishamiz yoki masalaning yechimga ega emasligi haqidagi ma'lumotga ega bo'lamiz.

Yuqorida keltirilgan xossalarni har bir ijrochi o'zi tuzgan biror masalaning algoritmidan foydalanib tekshirib ko'rishi mumkin. Masalan:

$$ax^2 + bx + c = 0$$

kvadrat tenglamani yechish algoritmi uchun yuqorida sanab o'tilgan algoritmning xossalarini quyidagicha tekshirib ko'rish mumkin.

Agar kvadrat tenglamani yechish algoritmi biror usulda yaratilgan bo'lsa, biz ijrochiga bu algoritm qaysi masalani yechish algoritmi ekanligini aytmasdan a, b, c larning aniq qiymatlari uchun bajarishni topshirsak, u natijaga erishadi va bu natija kvadrat tenglamaning yechimi bo'ladi. Demak, algoritmni ijro etish algoritm yaratuvchisiga bog'liq emas.

Xuddi shuningdek a, b, c larga har doim bir xil qiymatlar bersak, algoritm har doim bir xil natija beradi, ya'ni to'liqdir.

Yaratilgan bu algoritm faqatgina bitta kvadrat tenglamani yechish algoritmi bo'lib qolmay, balki u a, b, c larning mumkin bo'lgan barcha qiymatlari uchun natija hosil qiladi, binobarin u shu turdagi barcha kvadrat tenglamalarning yechish algoritmi bo'ladi.

Algoritmning oxirgi xossasi o'z-o'zidan bajariladi, ya'ni kvadrat tenglamani yechish albatta chekli qadamda amalga oshiriladi.

Dastur tuzuvchi uchun EHMning ikkita asosiy parametri o'ta muhimdir: hisoblash mashinasi xotirasining hajmi va mashinaning tezkorligi. Shuningdek, algoritm tuzuvchidan ikki narsa talab qilinadi. Birinchidan, u tuzgan dastur mashina xotirasida eng kam joy talab etsin, ikkinchidan, eng kam amallar bajarib masalaning natijasiga erishsin. Umuman olganda, bu ikki talab bir-biriga qarama-qarshidir, ya'ni algoritmning ishlash tezligini oshirish algoritm uchun kerakli xotirani oshirishga olib kelishi mumkin. Bu xol, ayniqsa murakkab masalalarni yechish algoritmini tuzishda yaqqol seziladi. Shuning uchun ham bu ikki parametrning eng maqbul holatini topishga harakat qilish kerak.

1.2. ALGORITMNI TAVSIFLASH USULLARI

Algoritm so'zlar, matematik formulalar, algoritmik tillar, geometrik tarhlar (sxemalar), dasturlash tillari va boshqalar yordamida tavsiflanadi.

Algoritmning so'zlar yordamida berilishiga, tavsiflanishiga misol tariqasida liftda kerakli qavatga ko'tarilish algoritmini keltirish mumkin. Bu quyidagicha ketmaketlikda bajariladi:

1. Liftga kiring.
2. Kerakli qavat tartib soniga mos tugmachani bosing.
3. Liftni harakatga keltiring.
4. Lift to'xtashini kuting.
5. Lift eshigi ochilgandan keyin undan chiqing.

Algoritm matematik formulalar yordamida tavsiflanganda har bir qadam aniq formulalar yordamida yoziladi. Misol tariqasida

$$ax^2 + bx + c = 0 \ (a \neq 0)$$

kvadrat tenglama yechimlari bo'lmish x_1, x_2 ni aniqlash algoritmini ko'rib chiqaylik.

1. a, b, c koeffitsiyentlar qiymatlari berilsin.

2. $D = b^2 - 4ac$ diskriminant hisoblansin.

3. $D < 0$ bo'lsa, tenglamaning haqiqiy yechimlari yo'q. Faqat haqiqiy ildizlar izlanayotgan bo'lsa, masala hal bo'ldi.

4. $D = 0$ bo'lsa, tenglama ikkita bir-biriga teng, ya'ni karrali yechimga ega bo'ladi va ular formulalar bilan hisoblanadi. Masala hal bo'ldi.

5. $D > 0$ bo'lsa, tenglama ikkita haqiqiy yechimga ega, ular

$$x_1 = (b + \sqrt{D}) / 2a \text{ va } x_2 = (-b - \sqrt{D}) / 2a$$

formulalar bilan hisoblanadi. Ya'ni masala hal bo'ldi.

Shunday qilib, kvadrat tenglamaning haqiqiy yechimlarini aniqlashda:

1. «Tenglamaning haqiqiy yechimlari yo'q» matni;

2. «Tenglama karrali yechimga ega, $x_1 = x_2$» matni va x_1, x_2 ning qiymatlari;

3. «Tenglama ikkita yechimga ega» matni, x_1 va x_2 ning qiymatlari natijalar bo'ladi.

Algoritmik tillar — algoritmni bir ma'noli tavsiflash imkonini beradigan belgilar va qoidalar majmuidir. Har qanday tillardagidek ular ham o'z alifbosi, sintaksisi va semantikasi bilan aniqlanadi.

Bizga o'rta maktabdan ma'lum bo'lgan (akademik A. P. Yershov rahbarligida yaratilgan) EHMsiz algoritmlashga mo'ljallangan algoritmik tizim algoritmik tilning namunasidir. Algoritmik tilga misol sifatida yana algoritmlarni belgili operatorlar tizimi shaklida tavsiflashni ham ko'rsatish mumkin. Bu tillar odatdagi tilga o'xshash bo'lib, EHMda bevosita bajarishga mo'ljallanmagan. Ulardan maqsad algoritmni bir xil shaklda va tushunarli qilib, tahlil qilishga oson qilib yozishdir.

Algoritmlarni geometrik tarhlar yordamida tavsiflash ko'rgazmali va, shu sababli tushunarliroq bo'lgani uchun ko'p qo'llaniladi. Bunda xar bir o'ziga xos operatsiya

alohida geometrik shakl (blok) bilan tavsiflanadi va ularning bajarilish tartibi, ular orasidagi ma'lumotlar uzatilishi va yo'nalishi bloklarni bir-biri bilan ko'rsatkichli to'g'ri chiziqlar yordamida tutashtirib ko'rsatiladi. Algoritmning geometrik tarhiga uning *blok-tarhi* (blok-sxemasi) deyiladi.

Bloklarga mos geometrik shakllar, ularning o'lchamlari va ular yordamida blok-tarhlarni chizish qoidalari davlat standartlarida berilgan. 1-jadvalda eng ko'p ishlatiladigan bloklar shakli va ularning ma'nosi keltirilgan. Bu davlat standartlariga ko'ra bloklarni tutashtiruvchi to'g'ri chiziq yozuv tekisligiga vertikal yoki gorizontal holatda bo'lishi kerak, ya'ni ularni og'ma chiziqlar bilan tutashtirish taqiqlanadi. Bloklarni bajarish tabiiy yozish tartibida bo'lsa, ya'ni yuqoridan pastga yoki chapdan o'ngga bo'lsa, tutashtiruvchi chiziq ko'rsatkichsiz bo'lishi mumkin.

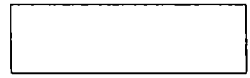
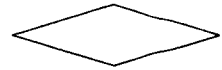
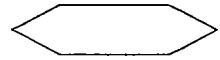
Boshqa barcha hollarda ma'lumot oqimi yo'nalishini ko'rsatuvchi ko'rsatkich qo'yilishi shart. Blokning tartib soni tutashtiruvchi chiziqdan chapga, alohida ajratilgan bo'sh joyga qo'yiladi. Chiziqning birlashgan joyi yirikroq nuqta yordamida ko'rsatiladi. Blokda ko'zda tutilgan operatsiya uning ichiga yozib qo'yiladi. Tarhlar davlat standarti formatlarida bajariladi.

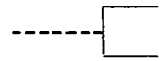
Amalda yechiladigan masalalar va demak, algoritmlar turlari ham juda ko'p bo'lishiga qaramasdan ular asosan besh xil: chiziqli, tarmoqlanuvchi, siklik, iteratsion va cheksiz takrorlanuvchi shakllarda bo'ladi deb aytish mumkin.

Agar murakkab masalalar algoritmlarining blok-tarhini bir bino desak, bu tuzilishdagi algoritmlar uni tashkil qiluvchi rom, g'isht, to'sin, ustun va boshqalarni ifodalaydi deb aytish mumkin. Har qanday murakkab bino ana shu ashyolardan qurilganidek, murakkab algoritmlar ham yuqoridagidek tarhlardan tuziladi. Aslida oxirgi uchta tuzilishdagi algoritmlarni bitta nom bilan takrorlash algoritmlari deb atash mumkin. Ammo ularning har biri o'ziga xos bo'lganligi uchun alohida nomlanadi.

1-jadval

Asosiy bloklarning shakllari va ularning vazifalari

Shakl nomi	Shakl	Vazifasi
Ishga tushirish, to'xtatish		Boshlash, tamomlash, ma'lumotlarni qayta ishlash jarayoni yoki dasturning bajarilishini to'xtatish
Kiritish-chiqarish		Ma'lumotlarni qayta ishlashga (kiritish) yoki qayta ishlash natijalarini akslantirish uchun (chiqarish) yaroqli holga keltirish
Jarayon		Bajarilishi natijasida ma'lumotlarning qiymati, tasavvur shakli yoki o'rnini o'zgartiradigan amal yoki amallar guruhi
Avvaldan ma'lum jarayon		Ilgari tuzilgan va alohida tavsiflangan algoritm va dasturdan foydalanish
Yechim		O'zgaradigan shartga bog'liq holda algoritm yoki dasturni bajarish yo'nalishini tanlash
Modifikatsiya (turlash)		Dasturni o'zgartiradigan buyruq yoki buyruqlar guruhini bajarish
Hujjat		Ma'lumotlarni qog'ozga chiqarish

Bog'lovchi		Ma'lumotlar oqimi-ning uzilgan joylarini tutash-tirish
Betlararo bog'lovchi		Turli varaqda joylashgan algoritm va dastur bo'-laklari orasidagi bog'la-nishni ko'rsatish
Izoh		Tarh elementlari va tu-shuntirish o'rtasidagi bog'lanish

1.3. CHIZIQLI TUZILISHDAGI ALGORITMLAR

Chiziqli tuzilishga ega bo'lgan algoritmlarda ko'rsatmalar yozilish tartibida bajariladi. Ularning blok- tarhlari ishga tushirish, to'xtatish, kiritish-chiqarish jarayoni bloki hamda avvaldan ma'lum jarayon bloklari yordamida tuzilib, bir chiziq bo'ylab ketma-ket joylashgan bo'ladi.

Chiziqli tuzilishdagi algoritmni tuzish masalani yechish uchun kerak bo'ladigan boshlang'ich ma'lumotlarni tashkil qiluvchi o'zgaruvchilar nomi, ularning turi va o'zgarish ko'lamini aniqlashdan boshlanadi. Keyin oraliq va yakuniy natijalar o'zgaruvchilarining nomlari, turlari va mumkin bo'lsa o'zgarish ko'lamini aniqlash kerak. Endi algoritm mana shu boshlang'ich ma'lumotlarni qanday qayta ishlab oraliq va yakuniy natijalarni olish kerakligini aniqlashdan iborat bo'ladi.

M i s o l . Tomonlari mos ravishda a, b, c bo'lgan ABC uchburchak yuzini hisoblash algoritmini tuzaylik.

Tomonlari ma'lum bo'lganda ABC uchburchakning yuzi

$$S = \sqrt{p(p-a)(p-b)(p-c)} \qquad (1)$$

Geron formulasi bilan hisoblanadi. Bunda

$$p = (a+b+c)/2 \qquad (2)$$

uchburchakning yarimperimetri.

1. Boshlang'ich ma'lumotlar: a, b, c uchburchak tomonlari. Shuning uchun a, b, $c \in R$ va $a>0$, $b>0$, $c>0$, ya'ni a, b, c — o'zgaruvchilar nomi; ular haqiqiy son qiymatlar qabul qiladi. Yana, bu uch son uchburchak tomonlarini ifoda qilishi uchun ularning istalgan biri qolgan ikkitasi yig'indisidan katta bo'lmasligi, ya'ni

$$a<b+c,\ b<a+c,\ c<a+b \qquad (3)$$

shartlar bajarilishi kerak. Shunday qilib, o'zgarish ko'lami (3) munosabatlar bilan aniqlanar ekan.

2. *Natijalar:* (1) formula bilan uchburchak yuzini hisoblash uchun uning yarim perimetrining qiymati kerak. Demak, p o'zgaruvchining qiymati oraliq ma'lumot bo'ladi. Yuqoridagi shartlarda $p \in R$ va $p>b$. Yakuniy natija: S — uchburchak yuzi. U $S \in R$ va $S>0$ qiymatlar qabul qiladi.

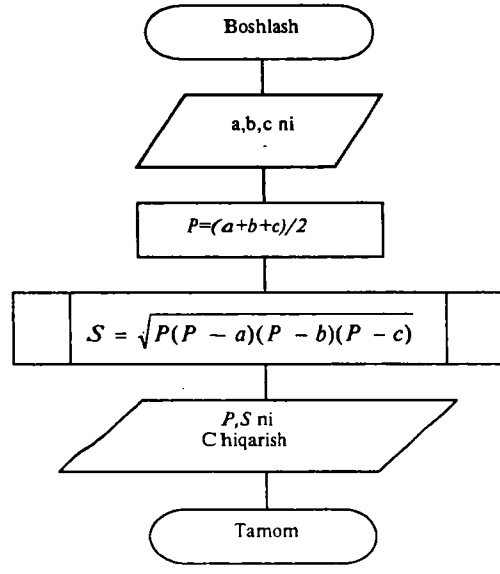
Shunday qilib, ixtiyoriy ABC uchburchak yuzini EHM-da hisoblash va bosmaga (yoki Displey ekraniga) chiqarish:

1) a, b, c qiymatlarini EHM xotirasiga kiritish:

2) p ning qiymatini (2) formula bilan hisoblash;

3) S ning qiymatini (1) formula bilan hisoblash;

4) p va S larning qiymatlarini bosmaga chiqarish operatsiyalaridan iborat bo'ladi (1-rasm).



1-rasm

Har qanday algoritmning blok-tarhi ishga tushirish

blokidan boshlanadi. Uni EHMni ishga tayyorlash, boshlang'ich ma'lumotlarni aniqlash va tayyorlash deb tushunish kerak. Hisoblashlarning tugaganligi ana shunday geometrik shakl bilan ko'rsatiladi. Shuning uchun rasmdagi 1- va 6-bloklar ichiga mos kelgan operatsiyalar nomi yozib qo'yilgan.

Boshlang'ich ma'lumotlarni EHMga har-xil qurilmalardan kiritish mumkin. Aniq bittasini tanlab olish ish sharoitiga bog'liq. Shuning uchun umumiy kiritish-chiqarish bloklaridan (2- va 5-bloklar) foydalaniladi.

Uchinchi blokda bevosita hisoblash jarayoni, to'rtinchi blokda esa kvadrat ildizdan chiqarish uchun tuzilgan kichik algoritm (yordamchi algoritm) dan foydalanish — avvaldan ma'lum jarayon ko'zda tutilgan. Algoritm ko'rsatmalari yozilish tartibida ketma-ket bajariladi. Ma'lumotlar blokdan blokka yuqoridan pastga uzatiladi. Shuning uchun ularni tutashtiruvchi chiziqqa ko'rsatkichlar qo'yilmagan.

Algoritmdan foydalanuvchi boshlang'ich ma'lumotlarni (3) shartlar bajariladigan qilib olishi kerak. Aks holda algoritmni bajarib bo'lmaydi. U natijalilik xossasiga ega bo'lmaydi.

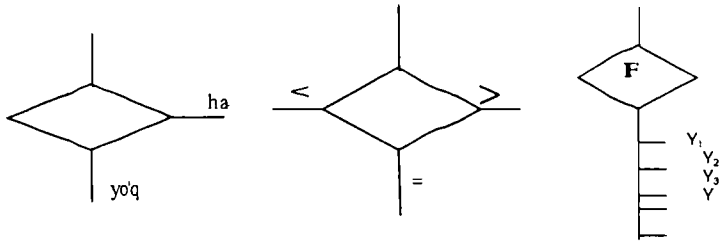
1.4. TARMOQLANUVCHI TUZILISHDAGI ALGORITMLAR

Turli masalalarni yechganda ko'rsatmalarni bajarish tartibi biror bir shartning bajarilishiga bog'liq holda bajariladi, ya'ni algoritm tarmoqlanadi. Tarmoqlanish «Yechim» bloki orqali ifodalanadi.

Shartni tekshirish natijasi faqat ikki hil bo'lganda: bajarilgan hol uchun «Ha» (yoki «+»), bajarilmagan hol uchun «Yo'q» (yoki «—») belgilari qo'yiladi (mantiqiy shart, 2, a-rasm).

Tarmoqlanish matematik ifoda qiymatining ishorasi bo'yicha bo'lganda (arifmetik shart): «>» — musbat, «<» — manfiy va «=» — nolga teng belgilar qo'yiladi (2, b-rasm),

Tekshirish natijasi uchdan ko'p bo'lganda 2, d-rasmdagidek tarmoqlanish bo'lishi mumkin.



a *b* *d*

2-rasm

3-misol. $ax^2+bx+c=0$ tenglama yechimlarini aniqlash algoritmini va uning blok-tarhini tuzing.

Bunda $a, b, c \in R$ va tenglamaning barcha yechimlari izlanayotgan bo'lsin. Agar bu sonlar berilgan bo'lsa, tenglama ildizlari quyidagi tartibda hisoblanadi:

1. Tenglamaning kvadrat tenglama ekanligi tekshiriladi. Buning uchun $a=0$ shart tekshiriladi. Agar bu shart bajarilsa tenglama kvadrat tenglama bo'lmaydi, 4-bandga o'tiladi.

2. Tenglama diskriminanti $D=b^2-4ac$ hisoblanadi.

3. D ning ishorasi tekshiriladi. Agar $D>0$ bo'lsa, tenglamaning ikkita haqiqiy ildizlari bor, ular

$$x_1=(b+\sqrt{D})/2a \text{ ва } x_2=(-b-\sqrt{D})/2a$$

formulalar bilan hisoblanadi.

Natijalar: 1-ma'lumot: «Tenglama ikkita haqiqiy yechimga ega». Bu matn va x_1, x_2 larning qiymatlari bosmaga chiqariladi. Masala yechildi.

2-ma'lumot: Agar D< O bo'lsa, tenglama

$$x_1=(b+\sqrt{|D|}i)/2a \text{ ва } x_2=(-b-\sqrt{|D|}i)/2a$$

formulalar bilan hisoblanadigan ikkita qo'shma kompleks ildizga ega bo'ladi. Ildizlarning haqiqiy qismini $x_1=-b/2a$, mavhum qismini $x_2=\sqrt{|D|}/2a$ bilan belgilansa, x_1, x_2 ni hisoblab, $x_1 \pm x_2 i$ kompleks sonlari hosil qilinadi.

3-ma'lumot: «Tenglama kompleks yechimga ega, haqiqiy qismi =... , mavhum qismi =2», matni va x_1 x_2

larning qiymatlari bosmaga chiqariladi. Bu holda ham masala yechildi.

4. Tenglamaning chiziqliligi $b=0$ shartni tekshirish bilan aniqlanadi. Agar bu shart bajarilsa, tenglama chiziqli emas, 6-bandga o'tiladi.

5. Tenglamaning yechimi: $x=-s/b$. Natijalar: 6-ma'lumot: «Tenglama chiziqli $x=$» matni va x ning qiymati bosmaga chiqariladi. Masala yechildi.

6. Agar $s=0$ bo'lsa, tenglama $0x^2+0x+0=0$ ayniyatga aylanadi va ixtiyoriy $x \in R$ son uning yechimi bo'ladi.

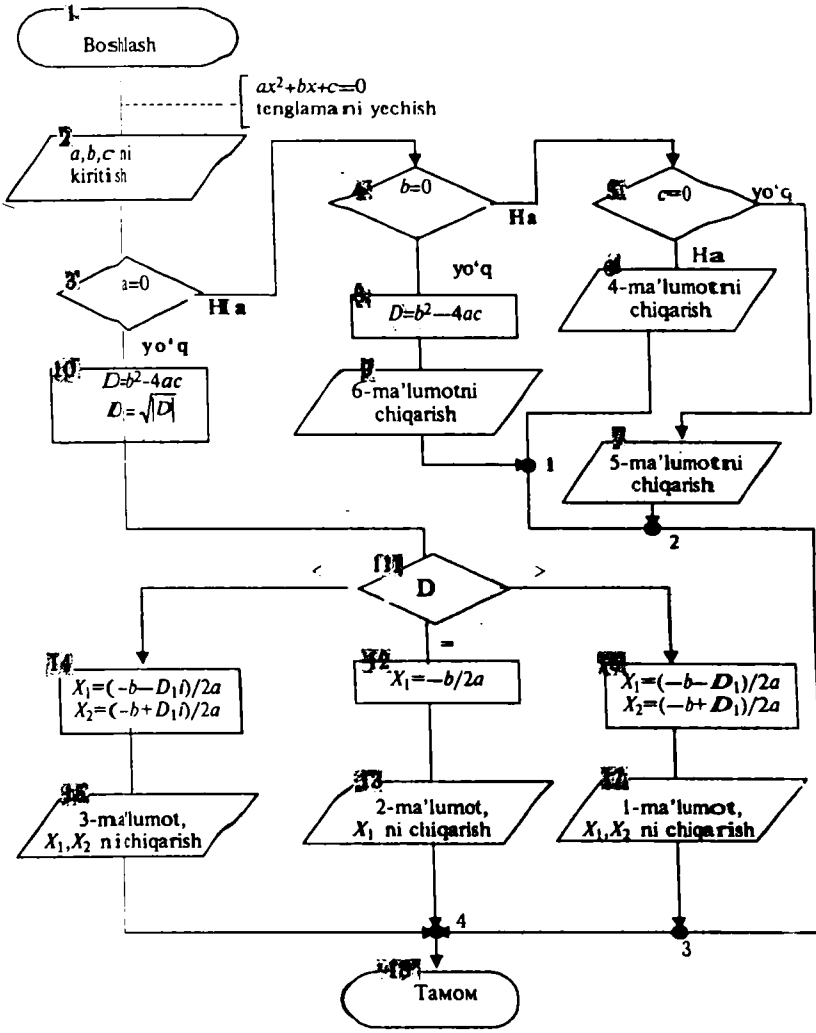
4-ma'lumot: «Tenglama cheksiz ko'p yechimga ega» matni bosmaga chiqariladi. Agar $c \neq 0$ bo'lsa, tenglama ma'noga ega bo'lmaydi. Bu holda natija: 5-ma'lumot: «Tenglama yechimga ega emas» matni bo'ladi.

Shunday qilib, hamma hollarda ham ma'lum bir natijaga kelinadi. Bu algoritmning blok-tarhi 3-rasmda keltirilgandek bo'ladi. Uchinchi, beshinchi va to'rtinchi «yechim» bloklarida mantiqiy shartlar — mos holda a, b va c koeffitsiyentlarning nolga tengliligi tekshirilmoqda. Bu shartlarni tekshirish natijasi ikki xil bo'lishi mumkin: bajarilgan (javob — ha) yoki bajarilmagan (javob — yo'q). Algoritm ikki tarmoqqa ajraladi. Bulardan farqli o'laroq, 11- yechim blokida D o'zgaruvchining ishorasi tekshirilmoqda. Natija uch xil bo'lishi mumkin. Tarhning 1,2,3,4 nuqtalarida ma'lumotlar oqimining qo'shilishi ro'y bermoqda. Shuning uchun bu nuqtalar ajratib ko'rsatilgan. Ma'lumot oqimini ko'rsatuvchi chiziq «singan» va oqim o'ngdan chapga yo'nalgan hollarda tutashtiruvchi chiziqlar ko'rsatkich bilan belgilangan.

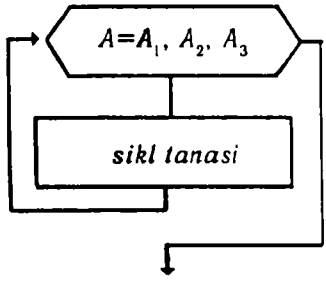
1.5. TAKRORLASH ALGORITMLARI

Takrorlash algoritmlari *sikl tanasi* deb nomlanuvchi ko'p marta takrorlanadigan qismni o'z ichiga oladi. Takrorlash biror shart bajarilguncha davom etadi. Yuqorida aytilganidek siklik, iteratsion va cheksiz davom etuvchi takrorlash algoritmlari farqlanadi.

Siklik tuzilishdagi algoritmlar takrorlash o'zgaruvchisi (sikl parametri) arifmetik progressiya turida o'zgar-


1
Boshlash
$ax^2+bx+c=0$
tenglamani yechish
2
a,b,c ni
kiritish
3
a=0
Ha
yo'q
4
b=0
Ha
yo'q
5
c=0
yo'q
Ha
6
D=b²—4ac
7
6-ma'lumotni
chiqarish
8
4-ma'lumotni
chiqarish
9
5-ma'lumotni
chiqarish
10
D=b²-4ac
D₁=√|D|
11
D
<
=
>
12
X₁=-b/2a
13
2-ma'lumot,
X₁ ni chiqarish
14
X₁=(-b-D₁i)/2a
X₂=(-b+D₁i)/2a
15
3-ma'lumot,
X₁,X₂ ni chiqarish
16
X₁=(—b—D₁)/2a
X₂=(—b+D₁)/2a
17
1-ma'lumot,
X₁,X₂ ni chiqarish
18
Tamom
1
2
3
4

3 - r a s m



4-rasm

ganda hosil bo'ladi. Algoritmning blok-tarhida ular modifikatsiya bloki bilan beriladi (4-rasm). Rasmda A— sikl o'zgaruvchisining nomi, A_1 — sikl o'zgaruvchisining boshlang'ich, A_2 — oxirgi qiymatlari, A_3 — sikl o'zgaruvchisining o'zgarish qadami (sikl raqami yoki oddiygina qadam deyiladi). Bunda A_1, A_2, A_3 ixtiyoriy son yoki ifoda bo'lishi mumkin.

Agar $A_3>0$ bo'lsa, $A_1<A_2$ bo'lishi, $A_3<0$ bo'lganda esa $A_1>A_2$ bo'lishi kerak. $A_3=1$ bo'lganda uni blok ichidagi yozuvda ko'rsatmaslikka kelishilgan.

Algoritmning bajarilishi:

1) $A=A_1$ qilib olinadi;

2) Sikl tanasiga kiruvchi amallar bajariladi, bunda biror shart bajarilganda sikl tashqarisiga chiqib ketish mumkin;

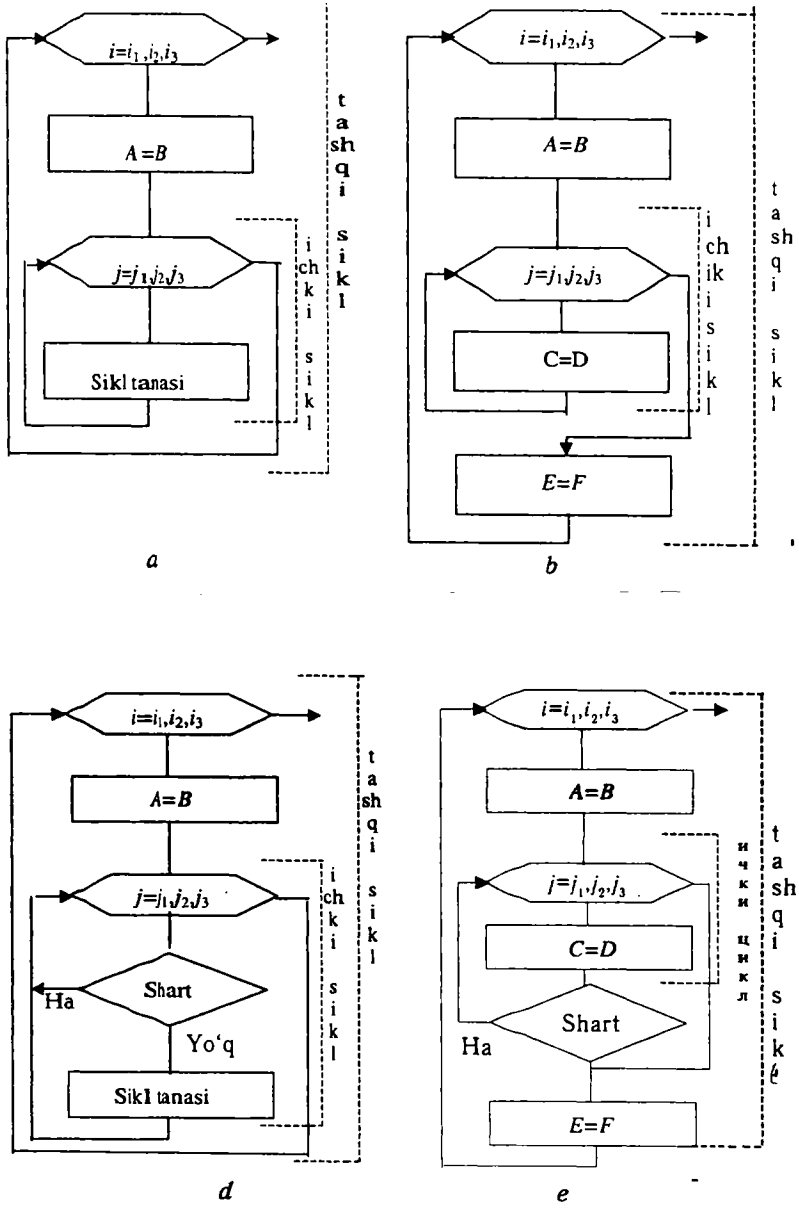
3) $A=A+A_3$ qilib olinib: $A_3>0$ bo'lganda $A<A_3$, yoki $A_3<0$ bo'lganda $A>A_2$ takrorlash sharti tekshiriladi.

Agar takrorlash sharti bajarilsa, sikl tanasidagi amallar uning o'zgaruvchisining yangi qiymatida bajariladi. Bunda ular sikl o'zgaruvchisining qiymatiga bog'liq bo'lishi ham, bo'lmasligi ham mumkin.

Siklik tuzilishdagi algoritmda takrorlash soni avvaldan berilgan bo'lishi mumkin yoki u $n=[(A_2-A_1)/A_3]$ formula bilan hisoblanadi. Bunda $[\bullet]$ belgi sonning butun qismini ifodalaydi.

Ichma-ich joylashgan sikllar. Bir sikl tanasida boshqa bir yoki bir necha sikllar joylashgan algoritmlar ham bo'ladi (5-rasm).

Rasmda j o'zgaruvchili sikl i o'zgaruvchili sikl ichiga joylashgan bo'lib 5, *a*-rasmda j bo'yicha sikl tugashi bilan i ning navbatdagi qiymatiga o'tilishi, 5, *b*-rasmda i o'zgaruvchili sikl tanasiga j bo'yicha sikldan tashqari amal-


i=i1,i2,i3
A=B
j=j1,j2,j3
Sikl tanasi
tashqi sikl
ichki sikl
a
i=i1,i2,i3
A=B
j=j1,j2,j3
C=D
E=F
tashqi sikl
ichki sikl
b
i=i1,i2,i3
A=B
j=j1,j2,j3
Shart
Ha
Yo'q
Sikl tanasi
tashqi sikl
ichki sikl
d
i=i1,i2,i3
A=B
j=j1,j2,j3
C=D
Shart
Ha
E=F
tashqi sikl
ichki tsikl
e

5 - расм

lar ham kirishi, 5, *d*-rasmda berilgan shart bajarilmasa, hech qanday amal bajarmasdan j ning keyingi qiymatiga o'tish, 5, *e*-rasmda berilgan shart bajarilsa, *j* bo'yicha sikl tugashi, bajarilmasa *j* ning keyingi qiymatiga o'tish kerakligi tasvirlangan. Ichma-ich joylashgan sikl turlari bu tuzilishlar bilan tugamaydi. Bunday siklik tuzilishdagi algoritmlarni ishlab chiqqanda tashqi sikldan ichki sikl boshini tashlab o'tib, uning ichiga kirish mumkin emasligini unutmaslik kerak.

Iteratsion algoritmlar. Ko'p hollarda amallarni necha marta takrorlashni avvaldan aniqlab bo'lmaydi. Takrorlash ma'lum bir shart bajarilguncha davom etadi. Bunday hollarda algoritmni 5-rasmda ko'rsatilgandek tashkil qilish mumkin.

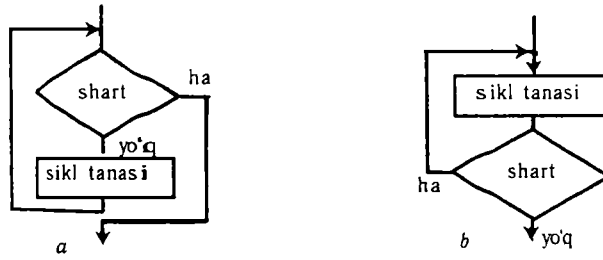
Takrorlash o'zgaruvchisi o'zgarmas qadam bilan o'zgarmaydi. 6, *a*-rasmda takrorlash sharti avvaldan tekshiriladigan, 6, *b*-rasmda esa keyin tekshiriladigan tuzilishdagi iteratsion (italyancha iteratixo — takrorlash so'zidan) algoritmning tarhi ko'rsatilgan.

Birinchi holda sikl tanasidagi amallar biror marta ham bajarilmasligi mumkin bo'lsa, ikkinchi holda esa kamida bir marta bajariladi. Albatta siklik tuzilishdagi algoritmni ham iteratsion tuzilishda tasvirlash mumkin. Lekin modifikatsiya blokini ishlatganda blok-tarh ixchamroq, bo'ladi. Takrorlash algoritmlarini tuzganda sikl tashqarisidan uning tanasiga kirish mumkin emasligini unutmaslik kerak. Chunki bunda sikl o'zgaruvchisining qiymati aniqlanmagan bo'lib qoladi. Aksincha, sikl tanasidan chiqish mumkin. Bunda sikl o'zgaruvchisining qiymati undan chiqqan paytdagidek bo'ladi.

Cheksiz takrorlanuvchi algoritm. Bunday algoritm EHM bilan muloqot olib borishga imkon beruvchi operatorlari mavjud bo'lgan dasturlash tillarida ishlatiladi (masalan, BEYSIKda). Bunda takrorlash foydalanuvchi tomonidan tugatilguncha davom etadi.

1-misol. $y = log_2(ex^2+lx+k)$ funksiyaning $x \in [a, b]$ kesmadagi qiymatlarini h qadam bilan hisoblash algoritmini tuzamiz.

Boshlang'ich ma'lumotlar: a, b, e, l, k, h o'zgaruvchilarning qiymatlari bo'ladi. Ular $a, b, e, l, k, h \in R$ bo'lib, ko'lami $ex^2+lx+k>0$ shart bilan aniqlanadi.



6-rasm

Tuzilgan algoritm: 1. y ning qiymatiga teng bo'lgan xaqiqiy son; 2. «Logarifm ostidagi ifoda qiymati musbat emas» matnli natijalarga ega bo'ladi. Bu natijalarni olishning ikki xil algoritmini ko'rib chiqaylik.

1-algoritm. Modifikatsiya blokidan foydalanilgan hol. Hisoblash ketma-ketligi quyidagicha bo'ladi:

1) h, a, b, e, l, k qiymatlarini EHM xotirasiga kiritish;

2) x o'zgaruvchi bo'yicha a dan b gacha h qadam bilan sikl tuzish;

3) $A = ex^2+lx+k$ ni hisoblash;

4) Agar $A>0$ shart bajarilsa, $y = log_2 A$ ni hisoblash va 6-qadamga o'tish;

5) «Logarifmlanayotgan ifoda musbat emas» matni (1-axborotni chiqarish va x ning boshqa qiymati uchun hisoblashga o'tish;

6) x va y qiymatlarini bosmaga chiqarish va 3-qadamga o'tish;

Bu algoritmning blok-tarhi 6-rasmdagidek bo'ladi.

2-algoritm. Modifikatsiya blokidan foydalanilmagan hol.

Natijalarni olish ketma-ketligi quyidagicha:

1) h, a, b, e, l, k qiymatlarini EHMga kiritish;

2) $x = a$ qilib olish;

3) $A = ex^2+lx+k$ ni hisoblash;

4) $A>0$ shartni tekshirish. Agar bu shart bajarilsa, 5-, bajarilmasa 7- qadamga o'tish;

5) $y = log_2 A$ ni hisoblash;

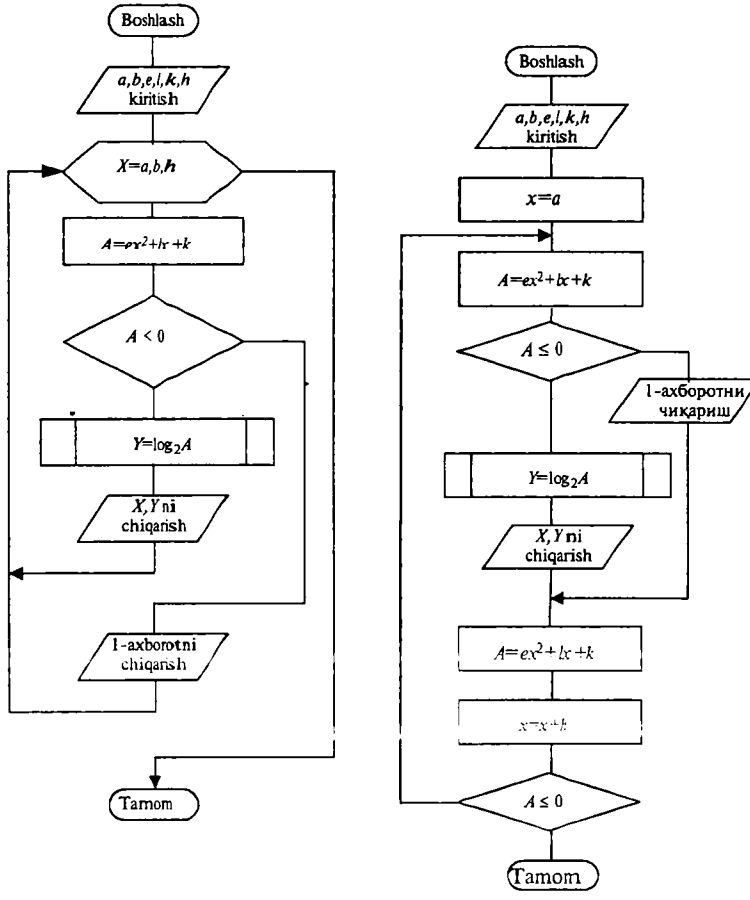
6) x, y qiymatlarini bosmaga chiqarish va 8-qadamga o'tish;

7) 1-axborotni bosmaga chiqarish;

8) $x = x+h$ qilib olib, $x<b$ shartni tekshirish. Agar bu shart bajarilsa, 3-qadamga o'tish, bajarilmasa, hisoblashni tugatish.

Algoritmning blok-tarhi 7-rasmda keltirilgan.

Birinchi algoritmning blok-tarhi ikkinchi algoritmnikidan ixchamroq ekanligini ko'ramiz. Bunga modifikatsiya blokidan foydalanilganligi sababli erishildi.



a) 1- algoritm

b) 2- algoritm

7 - r a s m

2-bob

DASTURLASH TILLARIGA NORASMIY KIRISH

2.1. MASALALARNI EHM DA YECHISH BOSQICHLARI

EHM bilan bevosita ishlashdan oldin qanday bosqichlarni bajarish kerakligini ko'rib chiqamiz. Istalgan hayotiy yoki matematik, fizik va hokazo masala shartlarini ifoda qilish dastlabki ma'lumotlar va fikrlarni tasvirlashdan boshlanadi va ular qat'iy ta'riflangan matematik yoki fizik va hokazo tushunchalar tilida bayon qilinadi. So'ngra yechishning maqsadi, ya'ni masalani yechish natijasida ayni nimani yoki nimalarni aniqlash zarurligi ko'rsatiladi.

Birinchi bosqich — masalani qo'yish. Istalgan masalani yechish uning qo'yilishidan boshlanadi. Masala shartining aniq ifodasi masalaning matematik (fizik va hokazo) qo'yilishi deb ham ataladi. Masalaning qo'yilishida boshlang'ich ma'lumotlar yoki argumentlar hamda qiymatlari aniqlanishi kerak bo'lgan kattaliklar, ya'ni natijalar ajratiladi. Masalani qo'yish uni yechishning birinchi bosqichi bo'ladi. Bunga turli-tuman misollar keltirish mumkin:

1. Tomonlarining uzunligi ma'lum bo'lgan to'g'ri to'rtburchakning yuzi hisoblansin.

2. Bosib o'tilgan yo'l va ketgan vaqt ma'lum bo'lsa, yo'lovchining tezligi aniqlansin.

3. Mashhur Pifagordan so'rashdi: Sizning maktabingizga nechta o'quvchi qatnashadi va suhbatingizni tinglaydi? Pifagor javob berdi: mening o'quvchilarimning yarmi matematikani o'rganadi, choragi musiqani o'rganadi, yettidan biri jimgina fikrlaydi, qolgani esa 3 ta. Pifagor maktabida nechta o'quvchi bo'lgan?

4. Shaxmat taxtasining kataklaridan bir katakka qayta yurmaslik sharti asosida, ot bilan yurib o'ting.

5. O'tloqdagi qo'ylarning sakkizdan birining kvadrati o'tlayotgan, qolgan 12 tasi yotgan bo'lsa, hammasi bo'lib nechta qo'y bor?

Ikkinchi bosqich — matematik modelni qurish.

Amaliy masalalarni hal etishda ob'ektlar — tabiat hodisalari (fizik yoki kimyoviy jarayonlar), mahsulot ishlab chiqarish jarayonlari, mahsulot ishlab chiqarish rejalari va shu kabilar bilan ish ko'rishga to'g'ri keladi. Ana shunday masalalarni qo'yish uchun avval tekshirilayotgan ob'ektni matematik atamalarda tavsiflash, ya'ni iloji bo'lsa, uning matematik modelini (ifodasini) qurish kerak. Mazkur model esa haqiqiy ob'ektni tekshirishni matematik masalani yechishga keltirish imkonini beradi. Modelning haqiqiy ob'ektga moslik darajasi amaliyotda tajriba orqali tekshiriladi. Tajriba—qurilgan modelni baholash va lozim bo'lgan holda uni aniqlashtirish imkonini beradi. Shuni ta'kidlash lozimki, har doim ham qo'yilgan masalaning matematik modelini yaratib bo'lavermaydi.

Yuqorida keltirilgan masalalarning matematik modellarini tuzamiz.

Birinchi masala uchun matematik model $S = ab$ ko'rinishdagi formuladan iborat. Bunda boshlang'ich ma'lumotlar tomonlari uzunligi a va b bo'lsa, natija to'g'ri to'rtburchakning yuzi S dan iboratdir.

Ikkinchi masala uchun bosib o'tilgan yo'lni s, ketgan vaqtni t deb belgilasak, yo'lovchining tezligi v fizika kursidan ma'lum bo'lgan

$$v = s/t$$

matematik model bilan ifodalanadi. Bunda s va t boshlang'ich ma'lumot, v esa natijadir.

Uchinchi masalada x deb o'quvchilar sonini belgilasak, u

$$\frac{x}{2} + \frac{x}{4} + \frac{x}{7} + 3 = x$$

yoki $3x—84=0$ ko'rinishdagi chiziqli tenglamaga keladi. Bu yerda 3 va 84 boshlang'ich ma'lumotlarni, x esa natijani ifodalaydi.

To'rtinchi masala uchun oshkor ko'rinishdagi matematik model mavjud emas, shuning uchun ham bu masalani yechishda birinchi bosqichdan keyin to'g'ridan-to'g'ri uchinchi bosqichga o'tish mumkin.

Beshinchi masalada x ni barcha qo'ylarning soni deb belgilasak, uni topish

$$x - (x/8)^2 = 12$$

ko'rinishdagi kvadrat tenglamani yechishga keladi. Umuman bu masalalar

$$ax^2 + bx + c = 0$$

kvadrat tenglama shaklidagi matematik model bilan ifodalanadi. Bunda a, b, c lar boshlang'ich ma'lumot bo'lsa, x (x_1, x_2) natija bo'ladi.

Shunday qilib, biz xodisalarni ifodalovchi matematik modellar bilan tanishdik. Albatta, hozir qurgan bu modellar juda sodda. Hayotda shunday murakkab masalalar uchraydiki, ular uchun matematik model yaratish juda ko'p kuch va vaqt talab etadi, ba'zi masalalarning esa modelini umuman tuzish mumkin emas.

Uchinchi bosqich — yechish usulini aniqlash.

Masalaning matematik modeli yaratilgandan so'ng, uni yechish usuli izlana boshlanadi. Ayrim hollarda masalani qo'yilishidan keyin to'g'ridan-to'g'ri masalani yechish usuliga ham o'tish kerak bo'ladi. Bunday masalalar oshkor ko'rinishdagi matematik model bilan ifodalanmasligi mumkin. Bu bosqich masalalarni EHM da yechishning uchinchi bosqichini tashkil qiladi. Bunga misol qilib yuqorida keltirilgan matematik modellarni yechish usullarini keltirish mumkin. Ular (1,2,3,5- masalalar) bilan siz matematika kursidan tanishsiz. Xo'sh, to'rtinchi masala uchun yechish usuli nima yoki qanday bo'lishi mumkin? Shaxmatdan xabardor har bir kishiga ma'lumki, shaxmat taxtasining ixtiyoriy katagida turgan otni yuqoridagi shart asosida har doim ham yurish mumkin emas. Hamma kataklardan o'tishning yagona usuli mavjud va u quyidagilardan iborat: faraz qilaylik, ot shaxmat taxtasining ixtiyoriy

bir katagida turibdi. Umuman olganda bu katakdan boshqa 8 ta katakka yurish mumkin. Yurilishi mumkin bo'lgan bu kataklarning har biridan ham yana nechadir kataklarga yurish mumkin. Mana shu mumkin bo'lgan yurishlaming eng kamini tanlash kerak, agar ular bir qancha bo'lsa, ixtiyoriy bittasini tanlash mumkin. Demak, otni shunday bir katakka yurish kerak ekanki, bu katakdan yurilishi mumkin bo'lgan kataklar soni eng kam bo'lsin. Faqat va faqat shu usul bilan qo'yilgan masalani hal qilish mumkin.

To'rtinchi bosqich — yechish algoritmini tuzish.

Navbatdagi bosqichda, ya'ni to'rtinchi bosqichda, masalani EHM dan foydalanib yechish uchun uning yechish algoritmi tuziladi. Algoritmni turli-tuman ko'rinishda yozish mumkin. Dasturlash fanining vazifalaridan biri ham algoritm tuzish usullarini o'rganishdan iboratdir. Bu jarayonda talabalarda masalani yechishning algoritmi, ya'ni algoritmik fikrlash usuli vujudga keladi.

Beshinchi bosqich — algoritmni dasturlash tiliga ko'chirish.

Algoritmning EHM da bajarilishi uchun bu algoritm dasturlash tilida yozilgan bo'lishi lozim. Masalani yechishning bu bosqichida biror bir usulda tuzilgan algoritm ma'lum bir dasturlash tiliga ko'chiriladi. Masalan, agar algoritm blok-tarh ko'rinishida tasvirlangan bo'lsa, uni dasturlash tiliga ko'chirish uchun har bir blokni tilning mos buyruqlari bilan almashtirish yetarli.

Oltinchi bosqich — dasturning bajarilishi.

Bu bosqichda — dastur ko'rinishida yozilgan algoritmni EHM yordamida bajariladi. Bu bosqich dastur tuzuvchilar uchun eng qiyini hisoblanadi. Chunki dasturni mashina xotirasiga kiritishda ayrim xatoliklarga yo'l qo'yish mumkin. Shuning uchun dasturni EHM xotirasiga kiritishda juda ehtiyot bo'lish kerak. Bu bosqich natija olish bilan tugallanadi.

Yettinchi bosqich — olingan natijalarni tahlil qilish.

Nihoyat, masalani yechishning yakunlovchi yettinchi bosqichi olingan natijalarni tahlil qilishdir. Bu bosqich

olingan natijalar qanchalik haqiqatga yaqinligini aniqlash maqsadida bajariladi. Natijalarni tahlil qilish, zarur bo'lgan hollarda algoritmni, yechish usuli va modelini aniqlashtirishga yordam beradi.

Shunday qilib, biz masalalarni EHM yordamida yechish bosqichlari bilan tanishib o'tdik. Shuni ta'kidlash kerakki, har doim ham bu bosqichlar bir-biridan yaqqol ajralgan holda bo'lmasdan, bir-biriga qo'shilib ketgan bo'lishi ham mumkin.

2.2. METALINGVISTIK FORMULALAR TILI

Algoritmik tilning sintaksis qoidalarini yozish va tushuntirish uchun ham qandaydir til lozim bo'ladi. Bu til bilan dasturlashning algoritmik tillarini almashtirib yubormaslik kerak. Kiritilishi kerak bo'lgan bu til dasturlash tilini aniqlash uchun kerak bo'ladi. Bu til «meta tili» deb ataladi. Tilni ifodalashda Bekus-Naurning metalingvistik formulalaridan (BNF) foydalaniladi.

BNF tilida dasturlash tillarining sintaksisi ixcham va formulalar ko'rinishida aniqlanadi. Bu formulalar oddiy arifmetik formulalarga o'xshab ketadi, shuning uchun ham ularni metalingvistik formulalar (qisqacha metaformula) deb ataladi.

Metaformulaning o'ng va chap qismlari «::=» belgisi bilan ajratiladi. Belgining ma'nosi «aniqlanishi bo'yicha shunday» jumlasiga yaqinroq. Bu belgining o'ng tomonida meta o'zgaruvchi, chap tomonida esa meta o'zgaruvchining qiymatlar to'plami yotadi. Tushunishga oson bo'lishi uchun meta o'zgaruvchilar, yoki ularning qiymatlari burchak qavslar («<» va «>») ichiga olib yoziladi. Masalan <son>, <arifmetik ifoda>, <operator> va hokazo.

Meta o'zgaruvchilarning meta qiymatlari bir necha mumkin bo'lgan konstruksiyalardan tashkil topishi mumkin. Bu holda konstruksiyalar o'zaro tik chiziq (|) bilan ajratiladi. Bu belgining ma'nosi «yoki» so'ziga yaqinroq tushunchadir.

Metaformulalarga misollar:

1. <o'zgaruvchi>::= A | B

<ifoda>::= <o'zgaruvchi> | <o'zgaruvchi> + <o'zgaruvchi> | <o'zgaruvchi> — <o'zgaruvchi>

bu formuladan <o'zgaruvchi> deganda A yoki *B* harflari tushuniladi, <ifoda> tushunchasi ostida quyidagi 10 ta holning biri bo'lishi mumkin:

A, B, A + A, A + B, B + A, B + B, A — A, A — B, B — A, B — B.

2. <ikkilik raqam>::= 0|1

3. <ikkilik kod>::=<ikkilik raqam> | <ikkilik kod> <ikkilik raqam>

3-misolda o'ng tomonda aniqlanayotgan tushuncha yotibdi, bu meta formulalarning rekursiv xossasiga ega ekanligini ko'rsatadi.

Meta formulalarni yozishda «{», «}» qavslar uchrab turadi. Bu qavs ichiga olib yozilgan konstruksiya takrorlanuvchi konstruksiya hisoblanadi.

Misol:

<ikkilik kod>::= <ikkilik raqam>{<ikkilik raqam>}

3 - bob

PASKAL TILIGA KIRISH

3.1. ALGORITMIK TILLARNING UMUMIY TAVSIFI

Ma'lumki EHM berilgan algoritmlarni formal bajaruvchi avtomat hisoblanadi, shuning uchun biror masalani EHMda yechishda unga mos algoritmni berish zarur. Algoritmni EHMga uzatishda esa uni maxsus «mashina tili»ga o'girib mashina kodida yozilgan dasturga aylantiriladi. Shu bilan bir qatorda EHMning turli xil tiplari turlicha tillarga ega bo'ladi, ya'ni biror EHM uchun yozilgan dastur boshqa EHM uchun tushunarsiz bo'lishi mumkin. Shunday qilib, har bir EHM faqat o'zining «mashina tili»da yozilgan dasturlarnigina tushunishi va bajarishi mumkin.

Mashina kodida yozilgan dasturlarning ko'rinish sifati juda kambag'aldir, chunki bu dasturlar faqat 0 va 1 larning maxsus ketma-ketligidan tashkil topadi. Bu esa mutaxassis bo'lmagan odamga tushunarsiz bo'lib, dastur tuzishda noqulayliklar keltirib chiqaradi.

Aytib o'tilganlardan shuni xulosa qilish mumkinki, mashina tilidan foydalanish odam uchun uni qiziqtirgan, ya'ni yechishi lozim bo'lgan masalaning algoritmini ishlab chiqishda va yozishda juda katta qiyinchiliklar va muammolar tug'diradi.

Yuqorida aytib o'tilgan qiyinchiliklarni bartaraf qilish, dasturchining ishini osonlashtirish va yaratilgan dasturlarning ishonchlilik darajasini oshirish maqsadida yuqori darajadagi dasturlash tillari yoki algoritmik tillar yaratilgan.

Algoritmik tillarning mashina tillaridan asosiy farqlari sifatida quyidagilarni ko'rsatish mumkin:

— mashina tili alifbosidan algoritmik til alifbosining o'ta kengligi;

— tuzilgan dastur matnining ko'rinish sifatini keskin oshiradi;

— ishlatilishi mumkin bo'lgan amallar majmui mashina amallari majmuiga bog'liq emas;

— bajariladigan amallar odam uchun qulay ko'rinishda, ya'ni amalda qabul qilingan matematik belgilashlarda beriladi;

— amallar operandlari uchun dasturchi tomonidan beriladigan shaxsiy ismlar qo'yish mumkinligi;

— mashina uchun ko'zda tutilgan ma'lumot tiplaridan tashqari yangi tiplar kiritish imkoniyati yaratilganligi.

Shunday qilib, ma'lum ma'noda aytish mumkinki, algoritmik tillar mashina tiliga bog'liq emas.

Yuqorida aytilganlardan kelib chiqqan holda ma'lum bo'ldiki, algoritmik tilda yozilgan masala yechimining algoritmi to'g'ridan-to'g'ri EHMda bajarilishi mumkin emas ekan. Buning uchun esa, algoritm oldindan ishlatilayotgan EHMning mashina tiliga translyator (kompilyator yoki interpretator) yordamida o'girilishi lozim. Translyator — mashina tilida yozilgan maxsus dastur bo'lib, uning asosiy maqsadi algoritmik tillarda yozilgan dastur matnini EHM tiliga tarjima qilishdan iboratdir.

Amalda dasturlashda foydalanilayotgan algoritmik tillar o'z ma'nosiga ko'ra algoritmni so'zli-formulali yozish uslubiga o'xshab ketadi, ya'ni ma'lum bir qism ko'rsatmalar oddiy matematik formulalar, boshqa qismlar esa so'zlar yordamida ifodalanishi mumkin. Misol sifatida n va m natural sonlarning eng katta umumiy bo'luvchisi (EKUB)ni topish algoritmini ko'rib chiqaylik:

1. $A=n$, $B=m$ deylik

2. Agar $A=B$ bo'lsa 5-bandga, aks holda 3-bandga o't.

3. Agar $A>B$ bo'lsa A ning yangi qiymati deb $A-B$ ni qabul qil, B ni qiymatini o'zgartirma; aks holda B ning yangi qiymati deb $B-A$ ni qabul qil, A ning qiymatini o'zgartirma.

4. 2-bandga o't.

5. EKUB=A va hisobni to'xtat.

Ushbu algoritmni qisqaroq ko'rinishda quyidagicha ifodalashimiz ham mumkin:

1. $A=n$, $B=m$ deylik;

2. Agar $A>B$ bo'lsa $A=A-B$ aks holda $B=B-A$, $A=B$ bo'lguncha 2-bandni takrorla.

3. EKUB=A va hisobni to'xtat.

Ushbu misoldan ko'rinib turibdiki algoritmlarni bunday yozish uslubi odam uchun ham qulay va ham tushunarlidir. Lekin bu uslubda ham ma'lum kamchiliklar ko'zga tashlanadi:

— algoritmni ortiqcha ko'p so'zli va uzun deyish mumkin;

— bir xil ma'nodagi ko'rsatmani turli xil uslublarda berish mumkinligi;

— bunday erkin ko'rinishda ifodalangan algoritmni EHM tiliga o'tkazish imkoniyati kamligi.

Yuqoridagi kabi kamchiliklarni bartaraf qilish uchun formallashgan, qat'iy aniqlangan algoritmik tillar ishlab chiqilgan. Algoritmik tillar uchta o'zakdan tashkil topadi: til alifbosi, sintaksisi va semantikasi.

Til alifbosi shu tilgagina tegishli bo'lgan chekli sondagi belgilardan tashkil topadi. Dastur matnini yozishda faqat shu belgilardangina foydalanish mumkin, boshqa belgilarni esa til tanimaydi, ya'ni ulardan foydalanish mumkin emas.

Til sintaksisi alfavit harflaridan tashkil topgan bo'lib, mumkin bo'lgan konstruksiyalarni aniqlovchi qoidalar tizimidir. Mazkur tilda ifoda etilgan to'la algoritm va uning alohida hadlari shu konstruksiyalar orqali ifoda qilinadi. Shunday qilib, belgilarning har qanday ketma-ketligini, hamda mazkur tilning matni to'g'riligi yoki noto'g'riligini til sintaksisi orqali bilib olamiz.

Til semantikasi algoritmik tilning ayrim konstruksiyalari uchun qoidalar tizimini tushuntirishga xizmat qiladi.

Endi algoritmik tillarning qaysilari amalda ko'proq ishlatilishi haqida fikr yuritsak. Ma'lumki, 70-yillarda bir guruh muammoli-yo'naltirilgan algoritmik tillar yaratilgan bo'lib, bu dasturlash tillaridan foydalanib juda ko'p sohalardagi muammoli vazifalar hal qilingan. Hisoblash jarayonlarining algoritmlarini ifodalash uchun Algol-60 va Fortran tillari, iqtisodiy masalalar algoritmlari uchun Kobol va Algek tillari, matnli axborotlarni tahrir qilish uchun esa Snobol tillari ishlatilgan. Sanab o'tilgan bu algoritmik tillar asosan katta hajmli, ko'pchilikning foydalanishiga mo'ljallangan EHMlar uchun mo'ljallangan edi.

Hozirda insoniyat faoliyatining barcha jabhalariga shaxsiy elektron hisoblash mashinalari (SHEHM) shaxdam qadamlar bilan kirib bormoqda. Asosan SHEHMlarga mo'ljallangan, hamda murakkab jarayonlarning hisob ishlarini bajarish va juda katta ma'lumotlar tizimi bilan ishlashni tashkil etuvchi yangi algoritmik tillar sinfi borgan sari kengayib bormoqda. Bu tillar jumlasiga quyidagi tillarni kiritish mumkin:

- Beysik tili;
- Paskal tili;
- Si tili va hokazo.

Dastur tuzishni o'rganishni boshlovchilarga mo'ljallangan, savol-javob tizimida ishlaydigan, turli-tuman jarayonlar algoritmini yozishga qulay bo'lgan tillardan biri BEYSIK(BASIC) tilidir. Beysik tilining nomi ingliz so'zi (Beginner's All-purpose Symbolic Instruction Code) ning o'qilishiga mos kelib, boshlovchilar uchun belgili ko'rsatmalar kodi(tili) degan ma'noni anglatadi. Beysik tilini yaratish ustidagi ishlar 1963 yilning yozidan boshlangan. Tilning ijodkorlari taniqli olimlar T.Kurs va J.Kemenilar hisoblanadi. Hozirga kelib Beysik tilining turli xil yangi ko'rinishlari ishlab chiqilmoqda va ulardan foydalanib millionlab dasturchilar ajoyib dasturlar yaratishmoqda.

Endi nisbatan mukammalroq bo'lgan Paskal va Si algoritmik tillari haqida qisqacha fikr yuritsak.

Paskal tili 1969 yili N.Virt tomonidan yaratilib, mashhur olim Blez Paskal nomi bilan ataldi. Bu til N.Virtning o'ylashi bo'yicha dasturlashning zamonaviy texnologiyasiga va uslubiga, strukturali dasturlash nazariyasiga asoslangan hamda boshqa dasturlash tillariga nisbatan muayyan ustunlikka ega bo'lgan til bo'lishi lozim edi. Mazkur til:

1. Dasturlash konsepsiyasini va strukturasini tizimli (sistemali) va aniq ifodalaydi;

2. Dastur tuzishni tizimli olib borish imkonini beradi;

3. Dastur tuzish uchun boy termin va struktura tarhlariga ega;

4. Yo'l qo'yilgan xatoliklarni tahlil qilishning yuqori darajadagi tizimiga ega.

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

4.0 versiyasidan boshlab dastur yozishni, tahrir qilishni va natijalar olishni osonlashtirish uchun yangi integrallashgan muhit hosil qilindi;

5.5 versiyasining paydo bo'lishi bilan Turbo-Paskalda ob'ektli dasturlash imkoniyati paydo bo'ldi;

6.0 versiyasidan boshlab esa Paskal dasturi ichiga quyi dasturlash tili bo'lmish Assembler tilida yozilgan dasturlarni qo'shish holati hosil qilindi. Shu bilan bir qatorda tilning integrallashgan muhiti ham bir qator o'zgarishlarga duchor bo'ldi.

Si tili 1972 yili D.Richi tomonidan turli xil EHMlar uchun universal til sifatida ishlab chiqilgan va dasturchi dastur tuzish jarayonida hisoblash mashinasining imkoniyatlaridan keng foydalanishi mumkin. Shuning uchun, bu til barcha narsani qilishga qodir degan tushuncha hosil bo'lgan.

Hozirda amalda foydalanilayotgan ko'pgina operatsion tizimlar Si tilida yaratilgan.

3.2. TILNING ALIFBOSI

Ma'lumki, har qanday tilni o'rganish uning alifbosini o'rganishdan boshlanadi. Tilning alifbosi — shu tilgagina tegishli bo'lgan asosiy belgilar va tushunchalar to'plamidan iborat bo'ladi. Paskal tilining alifbosini tashkil etuvchi asosiy belgilar majmuasini uch guruhga ajratish mumkin: harflar, raqamlar va maxsus belgilar.

Til alifbosining metalingvistik (Bekus — Naur) formulasi quyidagicha bo'ladi:

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

Harf sifatida katta va kichik lotin harflari ishlatiladi. Lekin, matnlar va dasturga izohlar yozish uchun kirill alifbosining bosh va kichik harflari ham alifboga kiritilgan.

Raqamlar sifatida oddiy arab raqamlari olingan:

<raqam>::=0|1|2|3|4|...|9

Maxsus belgilar ko'p sonli va bir jinslimas bo'lganligi uchun ularni o'z navbatida to'rtta guruhga ajratamiz:

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

<arifmetik amal belgisi>::= * | / | + | —

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

Solishtirish amallarining belgilari, ularning matematik ifodasi va amallarning ma'nosi 2-jadvalda o'z ifodasini topgan. Bu yerda shu narsaga ahamiyat berish kerakki, ba'zi bir amallar ikkita belgi orqali ifodalangan.

2-jadval

Solishtirish amali belgisining Paskaldagi yozilishi	Amalning matematik ifodasi	Amalning ma'nosi
	=	Teng
<>	≠	Tengmas
<	<	Kichik
<=	⩽	Kichik yoki teng
>	>	Katta
>=	⩾	Katta yoki teng

Ajratkichlar guruhini quyidagi belgilar tashkil qiladi:
<ajratkich>::= . | , | : | ; | (|) | [|] | { | } | ' | :=
Ajratkichlarning vazifalarini tilni o'rganish davomida aniqlab boramiz.

Xizmatchi so'zlar guruhi juda keng, shuning uchun bu so'zlarning hammasini birdaniga yodlab, eslab qolish shart emas, bil aks ulardan foydalanish davomida ketma-ket eslab qolinaveradi:

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

3.3. TILNING ASOSIY TUSHUNCHALARI

3.3.1. Operatorlar

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

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

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

S; S; ...;S.

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

3.3.2. Ismlar va identifikatorlar

Ma'lumki, ma'lumotlarning tahlili jarayonini ifodalovchi algoritm turli xil ob'ektlar (o'zgarmaslar, o'zgaruvchi miqdorlar, funksiyalar va hokazo) ustida ish olib boradi. Bu ob'ektlarga ularning vazifasi va qabul qiladigan qiymatlariga qarab maxsus ismlar beriladi. Shu ismlarni odatda, identifikatorlar deb ataladi. Identifikator deb harf

yoki «_» belgisidan boshlanuvchi harf, raqam va «_» belgisining ixtiyoriy ketma-ketligiga aytiladi:

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

Agar quyidagi oraliq tushunchani kiritsak:

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

Yuqoridagi aniqlashni quyidagicha ham yozish mumkin:

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

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

Ismlarga misollar:

_Burchak, _A1, Ahmad_Berdiev, C, Summa, Time, A, S1, ...

3.3.3. E'lonlar

Paskal tilining asosiy tushunchalaridan biri e'lon qilish hisoblanadi. Dasturda qatnashuvchi barcha ob'ektlarning ismlari mos ravishda dasturning bosh qismida, ularning qanday tipdagi qiymatlar qabul qilishi mumkinligiga qarab, e'lon qilinib qo'yilishi kerak. Paskal tilida e'lon qilishning 5 xil turi mavjud:

- metkalar e'loni;
- o'zgarmaslar e'loni;
- tip aniqlash uchun e'lon;
- o'zgaruvchilar e'loni;
- protsedura va funksiyalar e'loni.

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

3.3.4. O'zgaruvchilar

O'zgaruvchi dastur ob'ekti bo'lib, turli xil qiymatlarni xotirada ma'lum nom bilan saqlab turish uchun ishlatiladi. O'zgaruvchi o'z qiymatini dasturning bajarilish davomida o'zlashtirish operatori yordamida qabul qiladi. Qabul qilingan qiymat o'zgaruvchiga boshqa yangi qiymat berilmaguncha saqlanib turiladi va yangi qiymat berilishi bilan eski qiymat butunlay o'chib, yo'q bo'lib ketadi. Har bir o'zgaruvchiga ma'lum bir tipga tegishli qiymatlarnigina qabul qilish huquqi beriladi. Boshqa tipdagi qiymatlarni o'zlashtirishga urinish dasturning xatoligini ta'minlaydi.

O'zgaruvchi — bu identifikatordir. Uning ismi o'zgaruvchining qiymatiga murojaat qilishda ishlatiladi. Boshqacha aytganda, dastur matnidagi ism shu o'zgaruvchining qiymatini ifodalaydi.

3.3.5. Funksiyalar va protseduralar

O'rta maktab kursidan funksiya tushunchasi bizga yaxshi ma'lum. Algoritmik tillarda faqat qiymatini hisoblash algoritmlari ma'lum bo'lgan funksiyalargina ishlatiladi. Dastur tuzuvchi dastur uchun lozim bo'lgan keraklicha funksiyalarni o'z dasturiga kiritishi mumkin.

Xuddi funksiyalar kabi hal qilinayotgan masalaning ma'lum bir tugallangan bosqichlarini hisoblash vazifasini **protseduralar** zimmasiga yuklasa ham bo'ladi. Funksiyani hisoblash natijasida faqat, yagona natijaviy qiymatga erishiladi, protseduradan foydalanganda esa, natijaviy qiymatlar soni yetarlicha ko'p bo'lishi mumkin.

Dasturda aniqlangan funksiya va protseduralar o'zgaruvchilarning e'loni bo'limida e'lon qilinib qo'yilishi kerak. Bunda har bir funksiya va protseduraga ularning bajaradigan vazifasiga mos ismlar berib qo'yiladi. Ularni aniqlashda formal parametrlardan foydalaniladi. Bu parametrlarning tiplari o'z navbatida, funksiya va protseduraning ichida aniqlanilib, e'lon qilinadi.

Dasturda aniqlangan funksiya va protseduralardan foydalanish uchun dastur matnida ularning ismlari va formal

parametrlarga mos bo'lgan faktik parametrlari berilishi kerak.

Ma'lumki, matematika kursidagi elementar funksiyalardan dastur tuzishda juda ko'p foydalanishga to'g'ri keladi (masalan sin x, cjs x, $\ln x$, e^x va hokazo). Bunday funksiyalarni *standart funksiyalar* deb ataladi va standart funksiyalarning ismlaridan boshqa maqsadda foydalanish maqsadga muvofiq emas.

3.4. DASTUR MATNINI YOZISH QOIDALARI

Har bir algoritmik tilning dastur matnini yozish qoidalari turlicha bo'ladi. Dasturlash tillaridan eng soddasi Beysik tilining ma'lum versiyalarida dasturning har bir operatori qat'iy aniqlangan qator raqamlari orqali yoziladi. Paskal tilida esa operatorlar ketma-ket yozilib, o'zaro «;» belgisi bilan ajratib boriladi. Bundan tashqari, yozilgan dasturning o'qishga oson va undan foydalanish qulay bo'lishi uchun dasturda «matnni ajratish» tushunchasi (bo'sh joy, qatorning tugashi va izohlar) dan foydalaniladi.

Bo'sh joy (probel) grafik tasvirga ega bo'lmagan belgi bo'lib, qatordagi bo'sh joyni anglatadi. Lekin, bo'sh joy belgisi o'zining sonli kodiga ega va dastur matnidagi boshqa belgilar kabi komputerga kiritiladi.

Qator oxiri (tugashi) boshqaruvchi belgi bo'lib, u ham grafik tasvirga ega emas. Ma'lumki, dastur matnini yozish davomida uni tabiiy ravishda yangi qatorlarga ajratilib yoziladi. Chunki, shu matn yozilmoqchi bo'lgan qog'ozning ham, komputer ekranining ham o'lchamlari cheklangan. Dastur matnini alohida qatorlarga ajratmay yozish ham mumkin, lekin bir satrga 256 tadan ortiq belgi sig'maydi. Dastur matnini alohida qatorlarga ajratish dastur tuzuvchining xohishiga qarab bajariladi. Ma'lum bir qator tugamay turib, yangi qatorga o'tish uchun «qator oxiri» tugmachasi bosiladi. Bu tugmacha ham o'zining maxsus sonli kodiga ega.

Izohlar dasturni o'qishga oson bo'lishi, uni qiynalmay tekshirib, yo'l qo'yilgan xatolarni to'g'rilash va dasturda

bajarilayotgan ishlarni tushuntirib borish uchun qo'yiladi. Izohsiz yozilgan dasturni hujjat sifatida qabul qilinmaydi. Muvaffaqiyatli qo'yilgan izoh dasturning va dasturchining katta yutug'i hisoblanadi. Izohlar ixtiyoriy vaqtda dastur matniga kiritilishi yoki olib tashlanishi mumkin. Bu bilan dasturning ishi o'zgarib qolmaydi. Izohlarni «{» va «}» qavslari ichiga olinib yoziladi.

Dastur «matn ajratgich»laridan foydalanishning quyidagi qoidalariga amal qilish lozim:

- tilning ketma-ket yozilgan ikkita konstruksiyasi orasiga albatta bo'sh joy yozilishi kerak;
- ajratgichlarni xizmatchi so'zlar, sonlar va ismlar orasiga qo'yish maqsadga muvofiq emas.

Quyida yuqoridagi qoidalar asosidá yozilgan dasturga doir misol keltirilgan.

Misol. Quyidagi berilgan funksiyalarning qiymatlarini $[a,b]$ oralig'idagi $x=a+ih$, $h=\frac{b-a}{n}$ lar uchun (n—berilgan son) hisoblash dasturini tuzing: $f_1(x)=x_2$, $f_2(x)=3-x$, $f_3(x)=0{,}5-\sin x$

```
Program PI;
  { f1(x)=x*x; f2(x)=3—x; f3(x)=0,5—sin(x) funksiyalar qiymatini [a,b] oralig'ida hisoblash dasturi }
    const
        n=10; {[a,b] oraliqni 10 ta bo'lakchalarga ajratdik}
    Var
        a,b:real;
        i:integer;
        x,h,y1,y2,y3:real;
    Begin
        read(a,b); {[a,b] oraliqning chegaralarini ajratish}
        h:=(b—a)/n;  x:=a;  i:=0;  {Boshlang'ich ma'lumotlar hisoblandi}
    Repeat
        y1:=x*x;
        y2:=3—x;
        y3:=0.5—sin(x);
```

Writeln (x, $y1$, $y2$, $y3$); {Funksiyalar hisoblanib, natijalar chop etilmoqda}
$x:=x+h$; $i:=i+1$;
Until $i=n+1$
{Hisob ishlari yakunlandi}
end.

3.5. TURBO—PASKAL MUHITINI O'RNATISH

Turbo—Paskalning professional dasturlash — 6.0 versiyasi (Turbo Pascal Professional) Borland International Inc firmasining bir nechta disketalarida beriladi. Shu disketalarning biri «INSTALL/COMPILER» deb nomlanadi va bu disketada INSTALL.EXE dasturi mavjud.

Turbo—Paskalni o'rnatish uchun disketani diskovodga qo'yib, dasturni ishga tushiriladi. Dastur Sizga bir necha savollarga javob berishni taklif qiladi:

- qaysi diskovoddan Turbo—Paskalni ishga tushirmoqchisiz?
- Turbo—Paskalni vinchesterga o'rnatasizmi?
- qaysi kataloglarda Turbo—Paskalning ishchi fayllarini joylashtirish lozim?

Agar Sizda yetarli asos bo'lmasa, taklif qilingan katalog ismlari bilan chegaralangani ma'qul. Bu holda sizdan faqatgina disk ismini o'zgartirish talab etiladi. Shundan so'ng INSTALL.EXE dasturi o'zining ishini davom ettiradi va uning so'rovi bo'yicha navbat bilan ko'rsatilgan nomli disketalarni diskovodga qo'yiladi. Natijada Turbo—Paskal ishga tayyor holga keladi.

Faraz qilaylik, Turbo—Paskalni o'rnatishda odatdagi «S» diskning o'rniga «D» diskni tanladingiz va bu holda tizim quyidagi kataloglarga joylashdi:

D:\TP, D:\TP\BGI, D:\TP\DEMOS, D:\TP\DOC,
D:\TP\DOCDEMOS, D:\TP\TURBO3, D:\TP\TV DEMOS,
D:\TP\TVISION, D:\TP\UTILS

Bu yerda D:\TP\ katalogida Turbo—Paskalning asosiy fayllari joylashgan:

- Turbo.exe dasturni yaratish uchun lozim bo'lgan integrallashgan muhit (Turbo—Pascal Integrated Development Environment) fayli;
- Turbo.hlp —tezkor yordam ma'lumotlarini saqlovchi fayl;
- Turbo.tp — Turbo.exe dasturi foydalanadigan sistema konfiguratsiyasining fayli;
- Turbo.tpl — Turbo—Paskalning rezident modullari (Resident units);
- Tptour.exe — integrallashgan muhitda ishlashni tanishtiruvchi dastur.

Bulardan tashqari, D:\TP\BGI katalogida Turbo Paskalning grafik rejim ishini ta'minlovchi fayllar mavjud:

- Graph.tpu — Turbo Paskalning barcha grafik dasturlari ishlashi uchun zarur bo'lgan fayl;
- BGI kengaytmali bir nechta fayl—videotizimlarning turli tiplari bilan ishlashni ta'minlovchi drayverlar;
- CHR kengaytmali bir nechta fayl — vektorli shriftlarni o'z ichiga oluvchi fayllar.

3.6. TURBO—PASKAL TILI

3.6.1. Paskal tilining asosiy tiplari

Odatda, dasturda ishlatiluvchi ma'lumotlar quyidagi tiplarning birortasiga tegishli bo'ladi: butun qiymatli tiplar, haqiqiy qiymatli tiplar, belgili va satrli tiplar, mantiqiy qiymatli va ko'rsatkichli tiplar. Umuman olganda, tiplarni ikkita guruhga ajratish mumkin: asosiy (yoki oddiy) tip va hosilaviy tip. Yuqorida sanab o'tilgan tiplar asosiy guruhga tegishli bo'lgan tiplardir. Hosilaviy tiplar esa, asosiy yoki hosilaviy guruhga tegishli tiplardan hosil qilinadi.

Butun qiymatli tipga tegishli songa misollar: —1501, 0, 9999.

Butun qiymat qabul qiluvchi o'zgaruvchilarni e'lon qilish uchun Integer, ShortInt, Byte, LongInt va Word xizmatchi so'zlaridan foydalanish mumkin.

Haqiqiy qiymatli tipga tegishli sonlarga misollar: 25.0956, 6.75, —321.936, 1.2E+02, —3.57E-01

Haqiqiy (kasr) qiymatli tipga tegishli o'zgaruvchilarni e'lon qilish uchun Real, Single, Double, Extended va Comp xizmatchi so'zlaridan foydalanish mumkin.

Hamma harflar, belgi va raqamlar, (masalan A, b, «, !, $, S) belgili tipga tegishlidir. Belgili tipni qabul qiluvchi o'zgaruvchilarni e'lon qilish uchun Char xizmatchi so'zidan foydalanish mumkin.

Belgilarning ixtiyoriy yig'ilmasi (ketma-ketligi) qatorlar deb ataladi.

Misol: 'Ahmad', '$25', '_START'.

Qator xatto bo'sh ham bo'lishi mumkin (' '). Bu tipdagi o'zgaruvchilami e'lon qilish uchun String xizmatchi so'zidan foydalaniladi.

Mantiqiy o'zgaruvchilar faqat True (rost) va False (yolg'on) qiymatlarining bittasinigina qabul qilishi mumkin. Bu tip o'zgaruvchilarini e'lon qilish uchun Boolean hizmatchi so'zi ishlatiladi.

Ko'rsatkichlar ma'lumotlarning komputer xotirasidagi turarjoyi (adresi)ni aniqlab beradi va ularni e'lon qilish uchun Pointer xizmatchi so'zidan foydalaniladi.

Hosilaviy tiplarni hosil qilish va ularni e'lon qilish yo'llarini kelgusi bo'limlarda to'liq tushuntirib o'tiladi.

Yuqorida sanab o'tilgan tiplar haqida to'liqroq ma'lumotlar keltirib o'tamiz.

3.6.2.Butun sonlar

Butun qiymatli tiplarning barchasi 3- jadvalda keltirilgan.

Bu sanab o'tilgan tiplar o'zlarining qiymatlar qabul qilish oralig'i va xotiradan egallagan joyining katta yoki kichikligi bilan farqlanadi. Shuning uchun, o'zgaruvchilarning qabul qiladigan qiymatlarini katta yoki kichikligiga qarab, yuqoridagi tiplardan mosini tanlash maqsadga muvofiqdir.

3-jadval.

Tip ko'rinishi	Mazkur tipli o'zgaruvchining qabul qiladigan qiymatlar oralig'i	O'zgaruvchining komputer xotirasidan egallaydigan joyi
ShortInt	—128..127	8 bit
Integer	—32768..32767	16 bit
LongInt	—2147483648.. 2147483647	32 bit
Byte	0..255	8 bit
Word	0..65535	16 bit

Endi shu tipdan foydalanishga doir quyidagi **misol**ni ko'rib chiqaylik:

Berilgan *m* va *n* butun sonlari ustida quyidagi arifmetik amallar bajarish dasturini tuzing: *m+n,m—n, m*n*. Umuman Paskal tilida dastur tuzish unchalik murakkab emas, hozir shuni amalda ko'rsatamiz. Sistemali qavs ({,})lar ichiga turli izoh va tushuntirishlar yozib, ular bilan dasturni jihozlaymiz.

{Dastur sarlavhasini yozamiz}

Program Sonlar;

Var {dasturda foydalanish mumkin bo'lgan barcha o'zgaruvchilar shu var so'zidan so'ng e'lon qilinadi}

m,n:integer;{*m* va *n* o'zgaruvchilar o'rtacha kattalikdagi butun sonlar}

*k*1,*k*2,*k*3:integer;{*k*1=*m*+*n*, *k*2=*m*—*n*, *k*3=*m***n* — bajarilgan arifmetik amallar natijasini xotirada saqlash uchun tanlangan butun tipli o'zgaruvchilar}

begin {Paskal dasturi begin (boshlanmoq) so'zi bilan boshlanib, end.(tamom) so'zi bilan tamomlanadi}

Readln(*m,n*); {*m* va *n* butun sonlarini kiritish so'ralyapti, agar kiritilayotgan son butun bo'lmasa «Error 106:Invalid numeric format.» xatosi habar qilinadi va dastur ishini to'xtatadi}

```
  k1:=m+n;
  k2:=m—n;
k3:=m*n;      {so'ralgan amallar bajarildi}
  writeln (k1,k2,k3); {hisoblangan k1=m+n, k2=m—n,
k3=m*n natijaviy qiymatlarni chop etish tashkil etildi}
  end.                          {Dastur tamom bo'ldi}
```

Bundan tashqari, Turbo—Paskalda o'n oltilik sanoq tizimida yozilgan butun sonlardan foydalanishga ham ruhsat beriladi. O'n oltilik sanoq tizimidagi butun sonni aniqlashda uning oldiga «$» (dollar) belgisi qo'yiladi.

Misol: $11 o'nli sanoq tizimidagi 17 ga, $12 soni esa 18 ga teng.

Endi shu holatga doir quyidagi sodda dasturni keltiramiz:

```
Program Sanoq _tizim;
Var
N:integer;
Begin
N:=12; {N butun qiymatli o'zgaruvchiga o'nlik sanoq
tizimidagi 12 soni o'zlashtirilyapti}
N:=$12; {N butun qiymatli o'zgaruvchiga o'n oltilik
sanoq tizimidagi 12 soni o'zlashtirilyapti. Bu son amaldagi o'nli sanoq tizimida 18 ga teng}
End.
```

3.6.3.Haqiqiy sonlar

Haqiqiy sonlar matematika kursidan ma'lum bo'lgan oddiy o'nlik kasr sonlardir. Agar komputeringiz matematik soprotsessorli bo'lsa quyidagi 4-jadvalda sanab o'tilgan barcha tiplar o'rinli bo'ladi.

Agar komputerda matematik soprotsessor bo'lmasa, faqat Real tipinigina ishlatish mumkin.

Haqiqiy sonlarga doir quyidagi **misol**ni ko'raylik:

·Berilgan *m* va *n* haqiqiy sonlari ustida to'rt matematik amalni bajarish dasturini tuzing.

4-jadval

Tip ko'rinishi	Mazkur tipli o'zgaruvchining qabul qiladigan qiymat oralig'i	O'zgaruvchining komputer xotirasida egallaydigan joyi
Real	2.9E-39..1.7E38	6 bayt
Single	1.5E-45..3.4E38	4 bayt
Double	5.0E-324..1.7E308	8 bayt
Extended	3.4E-4932..1.1E4932	10 bayt
Comp	-9.2E18..9.2E18	8 bayt

```
Program arifmetika;
Var
m,n:real; {m va n o'zgaruvchilarni haqiqiy sonlar tipiga tegishli deb e'lon qildik}
Begin
Readln (m); {m sonini kiritish so'ralyapti}
  Readln (n); {n sonini kiritish so'ralyapti}
  Writeln ('sonlar yig'indisi=',m+n);
  Writeln ('sonlar ayirmasi=',m—n);
  Writeln ('sonlar ko'paytmasi=',m*n);
  If n<>0 then Writeln ('sonlar bo'linmasi=',m/n);
{Agar n soni nolga teng bo'lmasa, m/n amalining natijasi mos izoh bilan chop etiladi}
end.
```

3.6.4.Belgilar va qatorlar

Belgili tipli o'zgaruvchilar Char xizmatchi so'zi bilan e'lon qilinib, bu tipning qiymatlari xotiradan 1 bayt joy egallaydi. Paskal tilining barcha belgilari bu tipning qiymatlar sohasiga tegishlidir. Belgili qiymatni ' (apostrof) belgisi ichiga olib, yoki # belgisidan keyin uning ASCII kodini yozib aniqlash mumkin.

Misol: 'A', yoki # 60.

Qator — bu ′ (apostrof) belgisi ichiga olib yozilgan belgilarning oddiy ketma-ketligidir: 'Ab21#9!cd', 'dasturchi Saidkarim G'ulomov'.

Qator bo'sh yoki bitta belgili bo'lishi ham mumkin. Qatorli o'zgaruvchi uzunligi 255 gacha bo'lgan belgili qiymatlarni qabul qilishi mumkin. Umuman olganda, har bir qatorli o'zgaruvchiga xotiradan 256 bayt joy ajratiladi. Xotirani tejash uchun qatorning tipini quyidagicha ko'rsatish maqsadga muvofiqdir: String[N], N — qatordagi belgilar soni. Bu holda belgili o'zgaruvchi uchun N bayt joy ajratiladi.

Belgilar va qatorlar ustida bir qancha amallar bajarish mumkin, ya'ni qatordan kerakli bo'lakni kesib olish, qatorlarni bir-biriga qo'shish va natijada yangi qatorlar hosil qilish. Qatorlar haqidagi to'liq ma'lumotni kerakli bo'limdan olish mumkin.

Belgilar va qatorlarga doir quyidagi sodda dasturni keltiramiz:

```
Program String;
Var
  ch: char; {ch o'zgaruvchi belgili qiymat qabul qiladi}
  qator1,qator2:String; {qator1 va qator2 o'zgaruvchilar uzunligi 255 dan ortmagan qatorlarni o'zlashtirishi mumkin}
  N:String[5]; {N o'zgaruvchisi 5 ta belgidan tashkil topgan qatorlarni o'zlashtiradi}
Begin
ch: = 'A'; {ch o'zgaruvchisi A belgini o'zlashtirdi}
N: = 'Ascar'; {N o'zgaruvchisi 5 ta harfli Ascar so'zini o'zlashtirdi}
qator1: = ch+'li '+N; {qator1 o'zgaruvchisi natijaviy Ali Ascar so'zini o'zlashtirdi}
qator2: = ''; {qator2 o'zgaruvchisi bo'sh qatorni ifodalayapti lekin, bu o'zgaruvchi uchun xotiradan 256 bayt joy ajratilgan}
end.
```

3.6.5.Ma'lumotlarning mantiqiy tiplari

Paskal tilida mantiqiy tip boolean standart nomi bilan aniqlanadi. Mantiqiy tipli o'zgaruvchilar faqat ikki xil qiymat: True(rost) va False (yolg'on) larnigina qabul qilishi mumkin. Mantiqiy tipli qiymatlar ham tartiblangan, ya'ni False<True.

Paskal tilida asosan quyidagi uchta mantiqiy amaldan ko'proq foydalaniladi: not — rad etmoq, and — mantiqiy ko'paytirish, or — mantiqiy qo'shish.

Bu amallarni faqat mantiqiy o'zgarmaslar ustidagina ishlatish mumkin va natijada yana mantiqiy o'zgarmas hosil bo'ladi. Quyida mantiqiy o'zgarmaslar ustida amallar 5-jadvalda ko'rsatilgan:

5-jadval

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

Ixtiyoriy qiymatlarni solishtirish amali ham mantiqiy qiymatni beradi:

Misol: 3>2 natijasi true
0<—1 natijasi false.

3.6.6. Yangi tiplarni loyihalash

Paskal tilida tilning standart tiplaridan yoki oldin hosil qilingan yangi tiplardan foydalanib yana yangi tiplar yaratish mumkin. Dasturda yangi tiplarni kiritish uchun maxsus tip aniqlash bo'limi mavjud. Bu bo'lim type xizmatchi so'zidan keyin boshlanadi.

Har bir yangi tipni e'lon qilishdan oldin uning nomi (tipning identifikatori), so'ng esa tipning nimadan tashkil topganligi ko'rsatiladi.

Yangi tip yozuv ham bo'lishi mumkin, uning maydoni esa standart tipdan yoki oldingi kiritilgan tiplardan tashkil topishi mumkin.

O'z o'rnida kiritilgan yangi tip dasturni yozishda juda qo'l keladi va dasturning sifatini keskin oshiradi.

3.7. PASKAL DASTURINING TUZILISHI

Paskal dasturi dastur sarlavhasi va nuqta bilan tugovchi dastur tanasidan tashkil topgan. Dastur sarlavhasi va dastur tanasini ; (nuqta vergul) belgisi bilan ajratiladi:

<Paskal dasturi>::=<dastur sarlavhasi>;<dastur tanasi>.

Dastur sarlavhasi program xizmatchi so'zidan boshlanadi va undan so'ng dasturga foydalanuvchi bergan nom yoziladi:

<dastur sarlavhasi>::=***program***<dastur ismi>;

Dasturning asosiy qismi uning tanasi hisoblanadi. Dasturning tanasini qisqacha qilib blok deb ham atash mumkin. Umuman olganda, blok qat'iy ketma-ketlikda yoziluvchi oltita bo'limdan tashkil topgan:

<blok>::=<metkalar bo'limi>
<o'zgarmaslar bo'limi>
<yangi tiplar bo'limi>
<o'zgaruvchilar bo'limi>
<protsedura (qism dastur) va funksiyalar bo'limi>
<operatorlar bo'limi>

Dastur tanasining asosiy qismi bu operatorlar bo'limidir. Har qanday dasturda bu bo'lim albatta bo'lishi kerak. Dasturga qo'yilgan masalani yechish shu bo'limda amalga oshiriladi. Boshqa bo'limlar esa yordamchi bo'limlar bo'lib, tiplarni e'lon qilish bo'limlari deb ataladi. Bu yordamchi bo'limlar dasturda qatnashishi yoki qatnashmasligi ham mumkin, lekin ularning yozilish ketma-ketligi saqlanib qolinishi zarur.

Paskal dasturning umumiy ko'rinishini quyidagi ko'rinishda yozib olaylikda, so'ng har bir bo'limni to'laroq tahlil qilib chiqamiz:

Program <dastur ismi>;

```
label
      <metkalar ro'yxati>;
const
      <o'zgarmaslar va ularning qiymatlari>;
type
   <ma'lumotlarning yangi, nostandart tiplarini aniq-
lash>;
var
      <o'zgaruvchilarni, protseduralar va
       funksiyalarni e'lon qilish>;
begin
      <operatorlar bo'limi>
end.
```

Metkalar bo'limi. Dasturning ixtiyoriy operatorini boshqa operatorlar orasida ajratib ko'rsatish mumkin. Buning uchun, operatorning oldidan ikki nuqta (:) belgisi orqali metka (tamg'a yoki ism deb ham atash mumkin) qo'yiladi va bunday operatorni metka bilan jihozlangan operator deb ataladi. Metkalar dasturda o'tish operatoridan foydalangandagina ishlatiladi. Metka sifatida oddiy identifikatorlardan yoki sonlardan foydalanilsa bo'laveradi. Dasturda ishlatilgan barcha metkalar label xizmatchi so'zidan keyin boshlanuvchi metkalar bo'limida e'lon qilinib qo'yilishi kerak:

<metka bo'limi>::=<bo'sh> | label<metkalar ro'yxati>;

Metkalarning nomlari original, ya'ni o'xshashi yo'q bo'lishi kerak.

Misol: label L1, L2, A3;

Bu yerda L1, L2, A3 lar dasturda ishlatiluvchi metkalarning nomlari.

```
Label  m10, m20, StopLabel, 1;
Var
   I:ShortInt;
Begin
   1:
      If  i<10 Then Goto m10 Else Goto m20;
    m10:  Writeln('i  kichik 10 dan');
      Goto StopLabel;
```

```
M20:    i:=i-1;
        Goto 1;
StopLabel:
End.
```

O'zgarmaslar bo'limi. O'zgarmas — dasturning ishlashi davomida o'zgarmay qoladigan miqdordir. Agar miqdor dasturda ko'p marta ishlatilsa, uni dastur matnida qayta-qayta yozgandan ko'ra, bu miqdorni o'zgarmas deb aniqlab olib, dasturdagi miqdorning o'rniga o'zgarmasning ismini yozish qulay bo'ladi. Masalan, hammaga ma'lum (π=3,1415926535...) soni. Bu sonni bir necha marta takroran dasturda yozish noqulay, shuning uchun, uni o'zgarmas sifatida aniqlab olish maqsadga muvofiqdir.

O'zgarmas const xizmatchi so'zidan keyin e'lon qilinadi (aniqlanadi):

<o'zgarmasni aniqlash>::=<o'zgarmas ismi>=<o'zgarmas>;

bu yerda <o'zgarmas>::=<skalyar miqdor>|<belgili qator>|<o'zgarmas ismi>|+<o'zgarmas ismi>|—<o'zgarmas ismi>;

<o'zgarmaslar bo'limi>::=<bo'sh>|const<o'zgarmasni aniqlash>;

Misol:

```
Program L1;
const Pi=3.1415926535;
var A,B: real;
Begin
        A:= Pi+Sin(Pi*(Pi-1));
        B:= sqrt(a);
end.
```

Ma'lumotlarning tiplarini aniqlash bo'limi. Paskal tilida qiymatlarning to'rtta standart tiplari mavjud: integer (butun), real (haqiqiy), Char (belgili) va boolean (mantiqiy) dasturning muallifi bu tiplar bilan bir qatorda, o'zi uchun zarur bo'ladigan tiplarni aniqlab olib, ulardan ham foydalanishi mumkin. Buning uchun, muomalaga kiritilayotgan har bir yangi tipga o'ziga xos ism berish kifoya va o'zgaruvchilarni e'lon qilish bo'limida bu tipdan bemalol foydalanish mumkin bo'ladi.

Tip e'lon qilish quyidagi metaformula asosida amalga oshiriladi:

<tip e'loni>::=<tip ismi>=<tip>;

<tip>::=<tip ismi>|<tip vazifasi>;

Tiplarni aniqlash bo'limi esa quyidagicha aniqlanadi:

<tip aniqlash bo'limi>::=<bo'sh>|type<tip e'loni>

Misol:

```
type
Mantiq=boolean;
b= integer;
        Hafta=(dush, sesh, chor, pay, ju, shan, yaksh);
        Ish_Kuni=dush..ju;
```

O'zgaruvchilar bo'limi. Har qanday dasturda o'zgaruvchilar deb ataluvchi dastur ob'ektlaridan foydalaniladi. O'zgaruvchi — qiymat qabul qiluvchi ob'ektdir. Ularga qiymat dasturning bajarilishi davomida beriladi. Har bir o'zgaruvchiga uning qabul qiladigan qiymati va vazifasiga qarab ismlar beriladi. Shunday qilib, o'zgaruvchi o'zining nomi va qabul qiladigan qiymati bilan xarakterlanadi.

Dasturda ishlatiluvchi har bir o'zgaruvchi o'zi qabul qiladigan qiymatlarining tiplariga mos holda o'zgaruvchilar bo'limida e'lon qilinib qo'yilishi kerak:

<o'zgaruvchilar bo'limi>::=<bo'sh>|Var<o'zgaruvchilar e'loni>;

<o'zgaruvchilar e'loni>::=<o'zgaruvchi ismi>:<tip>

Misol:

```
Var
        n,m,k : integer;
        a : real;
        x,y : real;
S : boolean;
```

Dasturda ishlatilgan o'zgaruvchilar faqat bir martagina e'lon qilinishi kerak.

Protsedura va funksiyalar bo'limi. Boshqa tillardagi kabi, Paskal tilida ham dasturni yozuvchi o'zi uchun qulay va zarur bo'lgan protsedura va funksiyalarni muo-

malaga kiritishi mumkin. Tabiiyki bu protsedura va funksiyalar ham xuddi o'zgaruvchilar kabi e'lon qilinib qo'yilishi kerak. Hamma ishlatiluvchi protsedura va funksiyalarga ism berilib, shu ismi bilan ular chaqirilib dasturning zarur joyida ishlatiladi. Ularga murojaat xuddi oddiy standart funksiyalarga murojaat kabi amalga oshiriladi.

Protsedura va funksiyalar bo'limi o'zgaruvchilar bo'limining davomi bo'lib protsedure yoki function xizmatchi so'zlari bilan boshlanib, ixtiyoriy ketma-ketlikda e'lon qilinadi.

Operatorlar bo'limi. Bu bo'lim dasturning eng asosiy bo'limi bo'lib, dastur bo'yicha hisoblanuvchi barcha ishlar shu bo'limda bajariladi. Bo'limning metaformulasi quyidagicha yoziladi:

<operatorlar bo'limi>::= begin <operatorlar ro'yxati> end.

Dastur o'z ishini shu bo'limda yozilgan operatorlar ro'yxati bo'yicha, qat'iy ketma-ketlikda bajaradi.

3.8. TILNING OPERATORLARI

Dasturning asosiy vazifasi boshlang'ich ma'lumotlarni qayta ishlab, qo'yilgan masalaning natijasini beruvchi amallarni bajarishdan iborat. Algoritmik tillarda biror-bir masalani yechishda ma'lumotlar ustidagi amallarni bajarish operatorlar zimmasiga yuklatiladi. Dasturlash tillaridagi har bir operator ma'lumotlarni qayta ishlash jarayonining mustaqil bosqichi bo'lib, mantiqan yakunlangan hisoblanadi. Dasturda yozilgan operatorlarni to'g'ri talqin qilish uchun ularni yozish qoidalari (operatorning sintaksisi) qat'iy aniqlangan bo'lishi shart.

Shunday qilib, aytish mumkinki dastur bu — turli xil vazifalarni bajaruvchi va yagona maqsadga eltuvchi operatorlarning to'plamidir. Har bir operator ; (nuqta-vergul) belgisi bilan yakunlanadi. Mavjud dasturlash tili ruxsat bergan operatorlardan unumli va oqilona foydalanib, mukammal dasturlar yaratish dasturchining bilimiga, tajribasiga va san'atiga bog'liqdir.

Quyida Paskal tilining asosiy operatorlari bilan to'liqroq tanishib, ulardan dasturlashda foydalanish yo'llarini o'rganamiz.

3.8.1. O'zlashtirish operatori

Odatda dasturning natijasini olish uchun juda ham ko'p oraliq hisob ishlarini bajarishga to'g'ri keladi. Oraliq natijalarni esa ma'lum muddatga saqlab turish lozim bo'ladi. Bu ishlarni bajarish uchun tilning eng asosiy operatorlaridan biri bo'lmish — o'zlashtirish operatori ishlatiladi:

<o'zlashtirish operatori>::=<o'zgaruvchi>:=<ifoda>;

Bu yerda := o'zlashtirish belgisi hisoblanadi, bu belgini = (tenglik) belgisi bilan almashtirmaslik zarur. O'zlashtirish operatorida := belgisining o'ng tomonidagi <ifoda> qiymati aniqlanilib, so'ng chap tomondagi o'zgaruvchiga o'zlashtiriladi yoki boshqacha qilib aytganda, ifoda qiymati o'zgaruvchi nomi bilan xotirada eslab qolinadi. O'zgaruvchining oldingi qiymati esa (agar u bo'lsa) yo'q bo'lib ketadi.

O'zlashtirish operatorini yozishdagi eng muhim narsa bu ifoda va o'zgaruvchilarning bir xil tipli bo'lishidir.

O'zlashtirish belgisining o'ng tomonidagi ifodaning natijaviy tipiga qarab o'zlashtirish operatorini uch xil guruhga ajratish mumkin: arifmetik o'zlashtirish operatori, mantiqiy o'zlashtirish operatori, belgili o'zlashtirish operatori.

3.8.2. Arifmetik o'zlashtirish operatori

Butun yoki haqiqiy tipli, sonli natija beruvchi ifodani (odatda bunday ifodani arifmetik ifoda deb ataladi) hisoblash uchun arifmetik o'zlashtirish operatoridan foydalaniladi. Arifmetik ifodada qatnashuvchi barcha o'zgaruvchilar haqiqiy yoki butun tipli bo'lishi kerak. Arifmetik ifoda— sonlar, o'zgarmaslar, o'zgaruvchilar va funksiyalardan tashkil topadi hamda +, —, *, /, div, mod kabi amallar yordamida

yoziladi. Arifmetik amallarning bajarilishi quyidagi tartibda bo'ladi : *, /, div, mod, +,—

Ifodaning bajarilishidagi bu tartibni o'zgartirish uchun kichik qavslardan foydalaniladi. Ifodaning qavslar ichiga olib yozilgan qismlari mustaqil holda birinchi galda bajariladi.

Sanab o'tilgan arifmetik amallarning vazifalari bizga matematika kursidan ma'lum. Lekin, bu ro'yxatdagi div va mod amallari bilan tanish emasmiz. Div — butun bo'lishni anglatadi, bo'linmani butun qismi qoldirilib, qoldiq tashlab yuboriladi. Misol:

7 div 2 = 3
5 div 3 = 1
—7 div 2 = —3
—7 div—2 = 3
2 div 5 = 0
3 div 4 = 0

Mod — butun sonlar bo'linmasining qoldig'ini aniqlaydi. M mod n qiymat faqat n>0 dagina aniqlangan. Agar m⩾0 bo'lsa m mod n=m—((m div n)*n), m<0 bo'lsa m mod n=m—((m div n)*n)+n, m mod n ning natijasi doim musbat sondir.

Misol:
7 mod 2 = 1
3 mod 5 = 3
(—14) mod 3 = 1
(—10) mod 5 = 0

Paskal tilida ham boshqa algoritmik tillardagi kabi arifmetik standart funksiyalar mavjud. Bu funksiyalarning matematik ifodasi va Paskal tilidagi ifodalanishi 6-jadvalda keltirilgan:

6-jadval

Funksiyalar	Paskal tilida ifodalanishi
sin x	sin(x)
Cosx	cos(x)
Arctg x	arctan(x)
Lnx	ln(x)
e^x	exp(x)
$\sqrt{x}$	Sqrt(x)
$\|x\|$	Abs(x)
x^2	sqr(x)
Argumentning kasr qismini topish funksiyasi	Frac(x)
Argumentning butun qismini topish funksiyasi	Int(x)

Arifmetik ifodaga doir *misollar* :

2*5 — 4*3,

9 div 4/2,

45/5/3,

a + b/2*7.2 — sqrt(7),

exp(2 — a)*9.7 — 6.1*6.1

Paskal tilida darajaga ko'tarish amali yo'q, shuning uchun, bu amalni bajarishda logarifmlash qoidasidan foydalanamiz.

Misol: $y=a^n$, $a>0$ ifodani hisoblashni ko'rib chiqaylik. Tenglikning ikkala tomonini logarifmlaymiz:

$\ln y=\ln a^n$, logarifm xossasiga ko'ra

$\ln y=n\ln a$, bu tenglikdan «y» ni aniqlaymiz,

$u=e^{n\ln a}$ — bu tenglikni Paskal tilida quyidagicha yozish mumkin: $y=\exp(n*\ln(a))$.

Endi sal murakkabroq arifmetik ifodalarning Paskal tilida yozilishini ko'rib chiqaylik.

Matematik ifodasi	Paskal tilidagi ifodasi
$\frac{a+b}{c+d}$	(a+b)/(c+d)
$\frac{a\cdot(a+b)}{bc}$	a*(a+b)/(b*c)
$\frac{1}{1-\frac{1}{1-\frac{1}{x}}}$	1/(1—1/(1—1/x))

$2-(x-b)^2-e^{ax}+\sin\sin x$	2—sqr(x—b)— exp(a*x)+sin(c*x)
$x\cdot\frac{e^{x^2+y^2}-1}{\sqrt{\lvert x^2+y^2\rvert}}$	x*(exp(x*x+y*y)— —1)sqrt(abs(x*x+y*y))

Endi arifmetik o'zlashtirish operatoriga doir misollar ko'rib chiqamiz:

x:= 0;

c:= sqrt(a*a+b*b);

y:= 2*pi*r; i:= i+ 1; i:= 5/4; x:= a — b/2;

O'zlashtirish operatorining o'ng tomonidagi ifodada qatnashuvchi o'zgaruvchilar, albatta, bu operatordan oldin o'zining qiymatlariga ega bo'lishi kerak. Aks holda, o'zlashtirish operatori o'z ishini bajara olmaydi. Dastur tuzishda ko'pchilik yo'l qo'yadigan xatolikni quyidagi misolda tahlil qilib ko'ring:

To'g'ri tuzilgan dastur

```
Program Misol;
Var
a,x,y:Real;
Begin
a:=2.3;
x:=3.1;
y:=a*x;
Writeln('y=',y);
End.
```

Noto'g'ri tuzilgan dastur

```
Program Misol;
Var
a,x,y:Real;
Begin
a:=2.3;
y:=a*x;
{o'zlashtirish operatorining o'ng tomonidagi "X" o'zgaruvchining qiymati aniqlanmagan}
Writeln('y=',y);
End.
```

3.8.3. Mantiqiy o'zlashtirish operatori

Agar o'zlashtirish operatorining chap tomonidagi o'zgaruvchi boolean (mantiqiy) tipiga tegishli bo'lsa, operatorning o'ng tomonida natijasi *true* yoki *false* bo'lgan mantiqiy ifoda bo'lishi shart.

Mantiqiy ifoda arifmetik ifoda, solishtirish belgilari va mantiqiy amallardan tashkil topadi. Mantiqiy ifodaning natijaviy qiymati *true* (rost) yoki *false* (yolg'on) bo'ladi.

Mantiqiy ifodada amallarning bajarilish tartibi quyidagicha:

1. Not
2. *, /, div, mod, and
3. +,—,or
4. =, <, >, <=, >=,<>

Mantiqiy ifodada ham amallar ketma-ketligini o'zgartirish uchun kichik qavslardan foydalaniladi.

Mantiqiy ifodaga doir misollar:

1. $x<2*y$
2. true
3. not not d
4. $(x>y/2)$
5. d and $(x=y)$ and b
6. $(C$ or $D)$ and $(x=y)$ or not B

Mantiqiy o'zlashtirish operatoriga doir misollar:

```
d:= true;
b:= (x>y) and (k=0);
c:=D or B and true;
VAR Global Flag:Boolean;
FUNCTION GETSQR( x:real );
Const SQRMAX=100;
Begin
   X:=x*x;
   GlobalFlag:=( x>SQRMAX );
   If GlobalFlag then x:=SQRMAX;
   GetSQR:=x;
End;
```

3.8.4. Belgili o'zlashtirish operatori

Agar o'zlashtirish operatorining chap tomonida *char* (belgili) yoki String (qatorli) tipdagi o'zgaruvchi ko'rsatilgan bo'lsa, u holda operatorning o'ng tomonida belgili ifoda bo'lishi zarur. Belgili qiymatlar ustida faqatgina qo'shish (ulash) amalinigina bajarish mumkin . Shuning uchun belgili ifoda belgili o'zgarmas, belgili o'zgaruvchi yoki belgili tipli funksiya bo'lishi mumkin.

Belgili o'zlashtirish operatoriga misollar:

```
s:='+—';
d:='*/';
k:=s+d;
p:='Turbo Pascal';
```

```
Program M1;
Var
     s1,s2:String;
Begin
     s1:='oliy';
     s2:=' ta'lim';
     s2:=s1+s2;
     Writeln(s2);
End.
```

Natija:

oliy ta'lim

3.8.5.Tashkiliy operatorlar

Bir nechta operatorlar ketma-ketligini bitta operatorga birlashtirish uchun tashkiliy operator zarur bo'ladi. Tashkiliy operator — begin va end xizmatchi so'zlari orasiga olib yozilgan ixtiyoriy operatorlarning ketma-ketligidir:

<tashkiliy operator>::= *begin* <operator>{;<operator>} end

Xususiy holda, operatorlar ketma-ketligi bitta operatordan ham tashkil topishi mumkin.

Tashkiliy operatorga doir misollar :

1. Begin k:= 5 end
2. Begin y:= $x/7$*exp(x+5); z:= ln(abs(y)) end
3. Begin
   k:=0;
   begin
   i:=0;
   z:=i*(i+k);
   end;
   k:=2*k;
   end
4. if x>0 then begin a:=5; c:=a*Sin(a) end.

Yuqoridagi 3-misolda ko'rsatilganiday, tashkiliy operator rekursiv xarakterga ham ega.

3.8.6. O'tish operatori

Odatda, dastur o'z ishini yozilgan operatorlar ketma-ketligi bo'yicha amalga oshiradi. Operatorlarning tabiiy bajarilish ketma-ketligini buzish uchun shartsiz o'tish operatoridan foydalanish mumkin. Boshqarishni dasturning biror operatoridan boshqa operatorga uzatish uchun boshqarish uzatiladigan operator oldiga tamg'a (metka) qo'yilishi kerak . Boshqarishni shartsiz uzatish operatori quyidagi shaklda yoziladi:

<o'tish operatori>::= goto <metka>

bu yerda goto — ... ga o'tmoq demakdir. Bu operator yordamida boshqarish ko'rsatilgan metkali operatorga uzatiladi. Yuqorida aytganimizdek, dasturda qatnashgan barcha metkalar, dasturning metkalar bo'limida e'lon qilinishi kerak.

O'tish operatoriga doir misollar:

1) a:= 5.75;
   b:= sqr(a); goto L5;
   c:= 9.76;
   L5: d:= a+b;
2) L: a:=5; goto L;
3) 1: x:=0; d:=x*x; goto 1; y:=x;

1) dasturda S:=9.76 operatoridan boshqa barcha operatorlar bajariladi;

2) dastur a:=5 qiymatni tinimsiz hisoblaydi;

3) dastur ham $x:=0$ va $d:=x*x$ ifodani qayta-qayta hisoblab, $y:=x$ ifodani hisoblashga navbat kelmaydi.

Umuman olganda, dastur tuzuvchi iloji boricha o'tish operatoridan foydalanmaslikka harakat qilgani ma'quldir. Chunki o'tish operatoridan foydalanish dasturning o'qilishini qiyinlashtirib, uning sifatini keskin pasaytiradi.

O'tish operatoridan foydalanishga doir quyidagi to'liq dasturni ko'rib chiqaylik:

```
Program m1;
    Label 1;
    Var a,y:real;
Begin
1: Readln (a);
If a<=0 then goto 1;{a ning qiymati a>0 shartini qanoatlantirmaguncha qaytadan kiritilmoqda}
y:=ln(a)
Writeln('y= ',y);
end.
```

3.8.7.Shartli operator

Algoritmlar nazariyasidan ma'lumki, hisoblash jarayonlarini shartli ravishda uch xil guruhga ajratish mumkin:

1. Chiziqli jarayonlar;
2. Tarmoqlanuvchi jarayonlar;
3. Takrorlanuvchi jarayonlar.

*Chiziqli jarayon*ni hisoblash algoritmi qat'iy ketma-ketlik asosida amalga oshiriladi. Bunday jarayonni hisoblash uchun o'zlashtirish operatorining o'zi yetarli bo'ladi.

*Tarmoqlanuvchi jarayon*ni hisoblash yo'li ma'lum bir shartning bajarilish yoki bajarilmasligiga qarab tanlanadi. Tarmoqlanuvchi jarayonlarni hisoblash uchun shartli operatordan foydalaniladi. Shartli operator ikki xil ko'rinishda bo'ladi:

- to'liq shartli operator;
- chala shartli operator.

To'liq shartli operator quyidagi shaklda yoziladi:

<to'liq shartli operator>::= *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 soddaroq quyidagicha yozish mumkin:

if S then S1 else S2;

bu yerda S — mantiqiy ifoda;

S1 — S mantiqiy ifoda rost qiymat qabul qilganda ishlovchi operator;

S2 — S mantiqiy ifoda yolg'on qiymat qabul qilganda ishlovchi operator.

Shartli operatorning bajarilishi unda yozilgan S1 yoki S2 operatorlardan faqat birining 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:

1. if a=2 then d:= x+2 else d:= x—2;
2. if (x<y) and z then begin y:= x * sin(x);
   t:=x * cos(x) end else begin y:= 0; t:=1 end;
3. 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;

Chala (to'liqmas) shartli operatorning yozilishini quyidagicha ifodalasa bo'ladi:

if S then S1;

bu yerda S — mantiqiy ifoda, S1 — operator.

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

Yuqorida aniqlangan shartli operatorlardan bir xil maqsadda bemalol foydalanish mumkin.

Bu ikkala operatordan foydalanib dastur tuzish uchun quyidagi misolni ko'rib chiqaylik:

$$y = \begin{cases} ax + b & \text{arap } x > 0, \\ cx + d & \text{arap } x \le 0 \end{cases}$$

bu yerda faraz qilaylikki $a = 1,5$; $b = 4$; $c = 3,7$; $d = -4,2$. x — qiymati beriladigan noma'lum o'zgaruvchi.

«y» tarmoq funksiyasini hisoblash dasturini tuzish talab etilsin.

1. To'liq shartli operatordan foydalanib tuzilgan dastur:

```
program misol 1;
var  x, y, a, b, c, d: real;
begin
readln (x); { x ning qiymatini klaviaturadan kiritish so'ralmoqda}
a:=1.5; b:=4; c:=3.7; d:=—4.2;
if x>0 then y:= a*x+b else y:=c*x+d;
writeln (y);
end.
```

2. Chala shartli operatordan foydalanib tuzilgan dastur:

```
program misol2;
label L1;
var  x, y, a, b, c, d: real;
begin
      readln (x);
  a:=1.5; b:=4; c:=3.7; d:=—4.2;
  if x>0 then begin y:=a*x+b; goto L1 end;
  y:= c*x+d;
L1:   writeln (y);
end.
```

Shartli operatorning sintaksis qoidasiga ko'ra then va else xizmatchi so'zlaridan so'ng faqat bitta operator yozilishi mumkin, agar bir nechta operatorlarni yozish lozim bo'lsa u holda, bu operatorlar ketma-ketligi begin va end xizmatchi so'zlari orasiga olinib tashkiliy operator hosil qilinadi.

Misol:

```
if a>b then
begin
y:=a*cos(a);
z:=Sqr(y);
p:=Sqrt(abs(y+z));
```

```
        writeln(z)
    end
    else
    begin
        y:= a*sin(a);
        z:= Sqr(y)*y;
        p:= tan(y+z);
        writeln(z)
    end;
```

Ko'pgina operatorlar kabi shartli operator ham rekursivlik xossasiga ega, ya'ni shartli operator ichida yana shartli operator qatnashishi mumkin. Lekin, chala shartli operatorning ichida yana shartli operator yozishda ehtiyot bo'lmoq zarur, chunki yozilgan operatorni ikki xil ma'noda tushunish mumkin:

if B1 then if B2 then S1 else S2

bu operator quyidagicha tushunilishi mumkin:

1) if B1 then begin if B2 then S1 else S2 end
2) if B1 then begin if B2 then S1 end else S2

3.8.8.TAKRORLASH (SIKL) OPERATORLARI

Yuqorida sanab o'tilgan jarayonlardan biri — *takrorlanuvchi jarayonlarni* hisoblashni shartli operatorlardan foydalanib ham tashkil etsa bo'ladi, lekin bunday jarayonlarni hisoblashni takrorlash operatorlari yordamida amalga oshirish osonroq kechadi.

Takrorlash operatorlarining 3 xil turi mavjud:

- parametrli takrorlash operatori;
- repeat takrorlash operatori;
- while takrorlash operatori.

Yechilayotgan masalaning mohiyatiga qarab, dasturchi o'zi uchun qulay bo'lgan takrorlash operatorini tanlab olishi mumkin.

Parametrli takrorlash operatori. Operatorning quyidagi ko'rinishi amalda ko'proq ishlatiladi:

for *k*:= *k*1 *to* *k*2 *do* S;

bu yerda *for* (uchun), *to* (gacha), *do* (bajarmoq) — xizmatchi so'zlari;

k — sikl parametri (haqiqiy tipli bo'lishi mumkin emas);
$k1$ — sikl parametrining boshlang'ich qiymati;
$k2$ — sikl parametrining oxirgi qiymati;
S — sikl tanasi.

Operatorning ishlash tamoyili (prinsipi):

- sikl parametri (sp) boshlang'ich qiymat $k1$ ni qabul qilib, agar bu qiymat $k2$ dan kichik bo'lsa, shu qiymat uchun S operatori bajariladi;
- sp ning qiymati yangisiga o'zgartirilib (agar k son bo'lsa o'zgarish qadami 1 ga teng, belgili o'zgaruvchi bo'lsa navbatdagi belgini qabul qiladi va h.k.) yana S operatori bajariladi va bu jarayon $k > k2$ bo'lguncha davom ettiriladi. Shundan so'ng sikl operatori o'z ishini tugatib boshqarishni o'zidan keyingi operatorga uzatadi.

Agar biz operatorlarning necha marta takroran hisoblanishini aniq bilsak, u holda parametrli takrorlash operatoridan foydalanish maqsadga muvofiqdir.

Misol: $S = \sum_{i=1}^{n} \frac{1}{i}$ yig'indining chekli n ta hadining yig'indisini topish dasturini tuzish.

```
Program suml;
var
      S: real;
        i,n: byte; {i va n o'zgaruvchilar 255 dan katta
bo'lmagan, butun, natural sonlar}
begin
readln (n); S:= 0;
   for i:=1 to n do
         S:= S + 1 /i;
writeln (S);
end.
```

Ayrim paytlarda sikl parametrini o'sib borish emas, balki kamayish tartibida o'zgartirish mumkin, bu holda sikl operatori quyidagi formada yoziladi:

```
for k:= k2 downto k1 do S;
```

bu yerda *down to* (gacha kamayib) — tilning xizmatchi so'zi.

Bu operatorda k parametri $k2$ dan $k1$ gacha kamayish tartibida (agar k — butun qiymatli o'zgaruvchi bo'lsa sikl qadami — 1 ga teng) o'zgaradi. Operatorning ishlash tamoyili esa oldingi operatornikiday qolaveradi.

Misol: yuqorida ko'rsatilgan misolning dasturini qaytadan tuzaylik.

Bu holda dasturdagi sikl operatorigina o'zgaradi xolos:

for i:= n downto 1 do

qolgan operatorlar esa o'z o'rnida o'zgarmay qoladi.

Dasturda parametrli takrorlash operatoridan foydalanish jarayonida sikl parametrining qiymatini sikl tanasi ichida o'zgartirmaslik lozim, aks holda operatorning ish ritmi buzilishi mumkin. Buni quyidagi misollarda ko'rish mukin:

To'g'ri tuzilgan dastur qismi	*Noto'g'ri tuzilgan dastur qismi*
for i:=1 **to** 10 **do**	**for** i:=1 **to** 10 **do**
Begin	Begin
s:=i*i;	s:=i*i;
writeln(s);	writeln(s);
end;	i:=i+3
	end;

Ma'lum jarayonlarning takrorlash parametrlari haqiqiy qiymatlar qabul qilishi mumkin, bu holda parametrli takrorlash operatoridan to'g'ridan-to'g'ri foydalanib bo'lmaydi. Quyidagi misolda bunday takrorlashlarni qanday tashkil qilish mumkinligini ko'ramiz:

Misol: $y=e^x$ funksiyaning $[-2;2]$ oraliqdagi «x» lar uchun hisoblash dasturini tuzing («x» ning o'zgarish qadami 0,5 ga teng deb hisoblansin).

Funksiyani necha marta hisoblash kerakligini $N = \frac{2-(-2)}{0,5} = \frac{4}{\frac{1}{2}} = 8$ formula bilan aniqlaymiz.

```
Program Function;
  Var x:real;
```

```
        y:real;
        i:integer;
begin
    x:= —2;
     for i:=1 to 9 do
     begin
          y:=exp(x);
          writeln(x,y);
          x:=x+0.5
     end
end.
```

3.8.9.Repeat takrorlash (sikl) operatori

Yuqorida aytib o'tganimizdek, sikldagi takrorlanishlar soni oldindan ma'lum bo'lsa, parametrli (for) sikl operatori foydalanish uchun juda qulay. Lekin, ko'pgina hollarda takrorlanuvchi jarayonlardagi takrorlanishlar soni oldindan ma'lum bo'lmaydi, sikldan chiqish esa ma'lum bir shartning bajarilish yoki bajarilmasligiga bog'liq bo'ladi. Bu hollarda *repeat* yoki *while* sikl operatorlaridan foydalanish zarur. Agar sikldan chiqish sharti takrorlanuvchi jarayonning oxirida joylashgan bo'lsa *repeat* operatoridan, bosh qismida joylashgan bo'lsa *while* operatoridan foydalanish maqsadga muvofiqdir.

Repeat operatori quyidagi shaklda yoziladi:

repeat S1; S2; ... SN ***until*** *B*;

bu yerda *repeat* (takrorlamoq), *until* (gacha) — xizmatchi so'zlar;

S1, S2, ..., SN lar esa sikl tanasini tashkil etuvchi operatorlar;

B — sikldan chiqish sharti (mantiqiy ifoda).

Operatorning ishlash tamoyili juda sodda, ya'ni siklning tanasi *B* mantiqiy ifoda rost qiymatli natija bermaguncha takror-takror hisoblanaveradi. Misol sifatida, yana yuqoridagi yig'indi hisoblash misolini olaylik.

```
Program Sum2;
var  i, n: Byte;
```

```
    S: real;
begin
      readln(n);
      S: =0;  i:=1;
      repeat
          S:= S+1/i;
          i:=i+1;
      until  i>n;
      writeln (S)
end.
```

Ayrim takrorlanish jarayonlarida sikldan chiqish shartini ifodalovchi mantiqiy ifoda hech qachon True (rost) qiymatga erishmasligi mumkin. Bu holda dasturning takrorlash qismi cheksiz marta qaytadan hisoblanishi mumkin, ya'ni dasturchilar tili bilan aytganda «dastur osilib qoladi» shuning uchun, operatordagi shartni tanlashda e'tiborli bo'lish lozim.

E'tiboringizga ya'na bir misol tariqasida ismni qidirib topish dasturini havola qilamiz:

```
Program ism;
Var
   a,b:String[20];
Begin
    a:='Jamshid';
    Repeat
       Writeln('Tanlagan ismingizni kiriting');
        Readln(B);
       if a<>b Then writeln('Noto'g'ri') else writeln
('Yashang to'g'ri topdingiz');
    Until A=B;
End.
```

3.8.10. *While* takrorlash (sikl) operatori

Ahamiyat bergan bo'lsangiz, *repeat* operatorida siklning tana qismi kamida bir marta hisoblanadi. Lekin, ayrim paytlarda, shu bir marta hisoblash ham yechilayotgan masalaning mohiyatini buzib yuborishi mumkin. Bunday hollarda, quyidagi shaklda yoziluvchi while sikl operatoridan foydalanish maqsadga muvofiqdir:

while *B* do S;

bu yerda ***while*** (hozircha), ***do*** (bajarmoq) — xizmatchi so'zlari;

B — sikldan chiqishni ifodalovchi mantiqiy ifoda;

S — siklning tanasini tashkil etuvchi operator.

Bu operatorda oldin *B* shart tekshiriladi, agar u *false* (yolg'on) qiymatli natijaga erishsagina sikl o'z ishini tugatadi, aks holda siklning tana qismi qayta-qayta hisoblanaveradi.

While operatoriga misol sifatida yana yuqorida berilgan yig'indi hisoblash misolini ko'rib chiqaylik:

```
program sum3;
var     i, n: byte;
   S: real;
begin
        readln(n);
       i:=1; S:= 0;
       while i<=n  do
        begin
     S:= S + 1/i;
            i:= i+1;
       end;
       writeln (S)
end.
```

3.8.11.Bo'sh operator

Bo'sh operator o'zidan keyingi operatorni aniqlab beradi. U operatorlar ketma-ketligi orasida boshqa operatorlardan «;» belgisi bilan ajratiladi. Bundan tashqari, bo'sh operator metka bilan jihozlangan bo'lishi ham mumkin.

Misol:

1) **begin** L1:; *k*:=5; *M*:=*k*+6; **end.**

2) **begin** *M*:=5; *k*:=*M*—2.7; L4: **end.**

Ayrim paytlarda ba'zi operatorlarga bir nechta metka bilan murojaat qilishga to'g'ri kelganda bo'sh operatordan foydalanish qo'l keladi.

S5:; S6:; S7: *x*:=0.5;

Shu paytgacha biz qiymatlarning standart-skalyar tiplari ustida so'z olib bordik va ulardan dasturlashda foydalandik. Bu tiplar Paskal tilining o'zida aniqlangan tiplar edi. Lekin, Paskal tili dastur tuzuvchi o'zi uchun qulay bo'lgan yangi tiplar kiritish imkoniyatini ham beradi. Shunday yangi tiplardan biri sifatida cheklangan va sanalma tiplarni ko'rsatish mumkin.

3.9.1.Sanalma tiplar

Tilning standart tiplariga biz «butun son»lar, «haqiqiy son»lar, «belgi»lar va «mantiqiy qiymat»larni kiritgan edik. Lekin, amalda turli xil tipdagi qiymatlar bilan ishlashga to'g'ri keladi. Masalan, rang tushunchasi qizil, qora, oq, sariq, kulrang va h. k.larni o'z ichiga oladi, yoki yil oylari tushunchasi yanvar, fevral, ..., dekabr kabi 12 ta oyni o'z ichiga oladi. Bunday qiymatli tiplarni sonlar orqali ifodalab olsa ham bo'ladi, lekin bu belgilab olish ularning mohiyatini yo'qotib, tushunishga qiyin bo'lgan holni yuzaga keltiradi.

Masalan:

if k = 7 then

dastur qatorini o'qib, gap nima haqida ketayotganligini dabdurustdan anglash qiyin. Ehtimol, gap bu yerda 7 — oy haqidadir, balki «*k*» o'zgaruvchini 7 butun soni bilan tekshirilayotgandir. Shunday qilib, «*7*» soni ostida nima yashiringanini bilish juda qiyin. Lekin, dasturning bu qatori

if k = iyul *then*

bo'lsa, gap yilning iyul oyi haqida ketayotganligini osongina anglash mumkin.

Yuqoridagi kabi tushunmovchiliklarni bartaraf qilish, dasturning o'qishga qulayligini oshirish uchun qiymatlar tiplarining sanalma tipi kiritilgan.

Standart tiplar ichida bu tipga misol qilib *boolean* (mantiqiy) tipini ko'rsatish mumkin: *boolean = (false, true)*.

Sanalma qiymat tipini quyidagicha aniqlanadi:

<sanalma tipi>::=(<ism>,<ism>,...) yoki <sanalma tipi>::=(<ism> {,<ism>})
bu yerda kichik qavs ichidagi o'zaro vergul bilan ajratilgan <ism>lar aniqlangan tipning o'zgarmaslari hisoblanadi, ularning qavs ichiga olib yozilgan birikmasi esa, shu tipning qiymatlar to'plami hisoblanadi. Sanalma tip qiymatlari qat'iy noldan boshlab raqamlangan. Masalan, (dushanba, seshanba, chorshanba, payshanba, juma, shanba, yakshanba) sanalma tipi 7 ta haddan iborat bo'lib, bu yerda quyidagi hol o'rinlidir:

dushanba < seshanba < chorshanba < payshanba < juma < shanba < yakshanba, ya'ni dushanba 0 — tartib raqamiga, seshanba 1—tartib raqamiga ega va h.k.

Bu tip dasturning yangi tiplar bo'limida aniqlanadi.

Sanalma tipni aniqlashga doir misollar:

```
type
  Rang = (qizil, safsar, sariq, ko'k, havorang, kulrang, qora, oq);
  Hafta = (dush, sesh, chor, pay, jum, shan, yaksh);
  Mevalar = (olma, nok, shaftoli, uzum);
  Gul = Rang;
```

bu yerda biz to'rtta sanalma tip kiritdik, oxirgi *Gul* tipi *Rang* tipi bilan bir xil qilib aniqlandi.

Shuni esda tutish kerakki, bir ismning har xil tip qiymatlari bo'lishi mumkin emas. Masalan yuqoridagi tiplarning safiga

Ziravor = (zira, qalampir, olma)

tipini qo'shish mumkin emas, chunki **olma** qiymati **Mevalar** tipida aniqlangan edi. Bunday tip e'lon qilish, dasturning xatoligini anglatadi.

O'zgaruvchilarning tiplarini e'lon qilish bo'limida dasturning type bo'limida aniqlab qo'yilgan tiplardan xuddi standart tiplar kabi foydalansa bo'ladi:

```
var
  Kun: Hafta;
  shar, kub: Rang;
```

Yangi sanalma tiplarni o'zgaruvchilarning tiplarini e'lon qilish bo'limida ham kiritish mumkin:

var

A, B: (stul, divan, stol, shkaf, parta);

Lekin, kiritilgan bu tip ismsiz bo'lganligi uchun bu tipga dasturning boshqa joylaridan murojaat qilish mumkin emas. Shuning uchun sanalma tipni aniqlashning birinchi usuli ma'qulroqdir.

Sanalma tipli «*x*» argument uchun *Succ(x), pred(x)* va *ord(x)* standart funksiyalari mavjud. Yuqoridagi aniqlangan tiplarga doir misollardan ko'rib chiqaylik.

Faraz qilaylik, **x= sesh** qiymatli bo'lsin.

succ (*x*) = chor (x dan keyingi qiymat),

pred (*x*) = dush (x dan oldingi qiymat),

org (*x*) = 1 (*x* qiymatning tartib raqami),

for x = sesh to yaksh do S;

Sanalma tipli qiymatlar ustida hech qanday amalni bajarib bo'lmaydi.

Sanalma tipli qiymatlarni chop etish uchun odatda variant tanlash operatoridan foydalaniladi.

3.9.2.Variant tanlash operatori

Ayrim algoritmlarning hisoblash jarayonlari o'zlarining ko'p tarmoqliligi bilan ajralib turadi. Umuman olganda, tarmoqli jarayonlarni hisoblash uchun shartli operatordan foydalanish yetarlidir. Lekin, tarmoqlar soni ko'p bo'lsa, shartli operatordan foydalanish algoritmning ko'rinishini qo'pollashtirib yuboradi. Bu hollarda shartli operatorning umumlashmasi bo'lgan variant tanlash operatoridan foydalanish maqsadga muvofiqdir.

Variant tanlash operatorining sintaksis aniqlanmasi quyidagicha:

<variant tanlash operatori>::= ***case*** <operator selektori>

of <variant ro'yxatining hadlari> *end;*

bu yerda <operator selektori>::= <ifoda>;

<variant ro'yxatining hadi>::=<variantlar metkalarining ro'yxati>:<operator>;

<variantlar metkalarining ro'yxati>::=<variant metkasi>{,< variant metkasi>};

<variant metkasi>::=<o'zgarmas>

Variant tanlash operatori bajarilish paytida oldin selektorning qiymati hisoblanadi, shundan so'ng selektorning qiymatiga mos metka bilan jihozlangan operator bajariladi va shu bilan variant tanlash operatori o'z ishini yakunlaydi. Shuni esda tutish kerakki, <variant metka>si bilan <operator metka>si bir xil tushuncha emas va variant metkasi metkalar (Label) bo'limida ko'rsatilmasligi kerak. Bundan tashqari, ular o'tish operatorida ishlatilishi mumkin emas.

Misollar:

```
1. Case i mod 3 of
   0: m: = 0;
   1: m: = -1;
   2: m: = 1
  end.
2. Case sum of
     '=' : k: = 1;
   '*', '+', '/', '-' : ;
   '!' : k: = 2;
         ':', ';' : k: = 3
  end.
3. Case kun of
dush, sesh, chor, pay, jum: writeln('ish kuni');
shan, yaksh: writeln('dam olish kuni')
 end.
```

Variant tanlash operatorining tana qismiga kirish faqat *case* orqali amalga oshiriladi.

Endi shartli operatornng variant tanlash operatori orqali ifodalanishini ko'rib chiqaylik:

Quyidagi shartli va variant tanlash operatorlari bir-biriga mos keladi:

1. **if** *B* **then** S1 **else** S2 va Case *B* of
   true: S1;
   false: S2;
   end.

Quyidagi chala shartli va variant tanlash operatorlari bir-biriga mos keladi:

2. **if** *B* **then** S va Case *B* of
true: S;
false:
end.

Variant tanlash operatorlaridan sanalma tiplar qiymatlarini ko'rishga qulay holda chop etish uchun foydalanish mumkin. Buni quyidagi misolda ko'rib chiqamiz:

```
type
hafta=(Du,Se,Cho,Pa,Ju,Sha,Yak);
var
kun:hafta;
begin
kun:=Du;
case kun of
Du,Se,Cho,Pa,Ju: writeln('Ish kuni');
Sha,Yak: writeln('Dam olish kuni');
end;
for kun:=Du to Jak do
begin
case kun of
Du: writeln('Dushanba');
Se: writeln('Seshanba');
Cho: writeln('Chorshanba');
Pa: writeln('Payshanba');
Ju: writeln('Juma');
Sha:writeln('Shanba');
Yak:writeln('Yakshanba');

end ;
end;
end.
```

Sanalma tipli qiymatlarni kiritish ancha qiyin masala hisoblanib, bu ishni tashkil qilish uchun to'g'ridan-to'g'ri variant tanlash operatoridan foydalanish mumkin emas. Ma'lumotlarni kiritishda asosan qatorli (String) tiplarning imkoniyatlaridan foydalaniladi.

3.9.3.Cheklangan tiplar

Faraz qilaylik, «*n*» o'zgaruvchisi dasturda qaysidir oyning biror kunini ifodalovchi butun son bo'lsin. Bu o'zgaruvchini integer tipi bilan e'lon qilsak, «n»ga ixtiyoriy butun sonni o'zlashtirish mumkin. Lekin, yechilayotgan masalaning mohiyatiga ko'ra «*n*»ning faqat [1; 31] oraliqdagi qiymatlarigina ma'noga ega xolos. O'zgaruvchining boshqa qiymatga ega bo'lishi uning xato hisoblanayotganligini yoki dasturga berilgan ma'lumotlarning xatoligini anglatadi. Shunga o'xshash, dasturchi dasturni tuzish davomida ma'lum bir o'zgaruvchilar qiymatlarining o'zgarish oraliqlari haqida ma'lumotga ega bo'ladi. Bunday ma'lumotlarning dasturda ko'rsatib qo'yilishi, dastur ishining to'g'ri bajarilayotganligi ustidan nazorat qilib turish imkoniyatini yaratadi.

Shunday cheklashlarni amalga oshirish uchun Paskal tilida cheklangan tiplar kiritilgan. Har bir shunday tip oldindan ma'lum bo'lgan tiplarga cheklashlar kiritish orqali aniqlanadi.

Cheklangan tiplar quyidagicha aniqlanadi:

<cheklangan tip>::=<o'zgarmas1>..<o'zgarmas2>

bu yerda <o'zgarmas1> va <o'zgarmas2>lar cheklangan tip kiritilayotgan, oldindan aniqlangan tipning o'zgarmaslaridir hamda <o'zgarmas1> < <o'zgarmas2> sharti bajarilishi kerak.

Cheklangan tiplarga doir misollar:

1...20 (integer tipidagi cheklanma);

1) dush..jum (sanalma tipli cheklanma);

2) 'A'...'Z' (char tipidagi cheklanma).

Yuqorida aytganimizdek, cheklanma oldindan aniqlangan tipga nisbatan aniqlanadi. Masalan, (dush, sesh, chor, pay, jum, shan, yaksh) sanalma tipini aniqlamasdan turib dush...jum cheklangan tipini kiritish mumkin emas.

Cheklanma tip ham boshqa tiplar kabi yangi tip aniqlash bo'limida yoki o'zgaruvchilarning tiplarini e'lon qilish bo'limida aniqlanishi mumkin.

Cheklangan tiplarni e'lon qilishga doir bo'lgan misollardan yana ko'rib chiqaylik:

```
type
  Hafta = (dush, sesh, chor, pay, jum, shan, yaksh);
  Figura = (piyoda, ot, fil, ferz, shoh);
  Engil_Figura= piyoda..fil;
  Ish_Kunlari = dush..jum;
  Indeks = 10..20;
var
  x, y, z: real;
  i, j: integer;
Kun: Hafta;
L: Indeks;
  Ish_Kuni: Ish_Kunlari;
  Dam_Olish_Kuni: Shan..yaksh.
```

3.10. KOMBINATSIYALI TIPLAR (YOZUVLAR)

E'tiboringizga yana bir yangi, boshqa algoritmik tillarda mavjud bo'lmagan Paskal tilining hosilaviy tiplaridan birini — *kombinatsiyali tipni* havola qilamiz. Kombinatsiyali tipning qiymati ham xuddi massivlarniki kabi bir nechta haddan tashkil topadi. Lekin massivdan farqli o'laroq, uning har bir hadi turlicha tipli bo'lishi mumkin. Bu tipning hadlariga ularning joylashgan tartib raqamlari bilan emas, balki ismlari orqali murojaat qilinadi.

Kombinatsiyali tipni odatda soddagina qilib — *yozuvlar* deb ataladi.

Kombinatsiyali tip qiymatlari asosan murakkab, bir jinsli emas tashkil etuvchilarga ega bo'lgan ob'ektlarni ifodalashga bag'ishlangan. Bu tipdan amalda turli xil ma'lumotlar majmuasini yaratishda keng foydalaniladi. Masalan, biror tashkilotda ishlovchi xodimlar haqidagi ma'lumotlar majmuasi — xodimlarning turli xil anketali xabarlarini o'zida jamlaydi:

familiyasi,
ismi-sharifi,
tug'ilgan yili—oyi—kuni,

yashash manzili, ishchi va uy telefon raqamlari,
ma'lumoti, mutaxassisligi,
oilaviy ahvoli,
harbiyga aloqadorligi va h.k.

Sanab o'tilgan va bitta shaxsga tegishli bo'lgan ushbu ma'lumotlarning turli xil tipga tegishligiga ahamiyat bering: telefonlar, tug'ilgan yili—oyi—kunlari — butun sonlardan, boshqa ma'lumotlar esa belgili qatorlardan tashkil topgan.

Shunday qilib, kombinatsiyali tip qiymati — maydonlar deb atalmish chekli sondagi hadlardan tashkil topgan ma'lumotlar strukturasidir. Yozuvning har bir maydoniga o'ziga xos ism beriladi va bu maydon qiymatining tipi ko'rsatiladi. Bunda maydon qiymatining tipiga hech qanday cheklashlar qo'yilmaydi. Shuning uchun yozuv hadlari o'z navbatida yana yozuv bo'lishi mumkin. Demak, yozuv aniq ifodalangan iyerarxik (shajarali) tuzilishga ega bo'lishi mumkin. Bir xil darajada turgan bitta yozuvning barcha ismlari turli xil bo'lishi lozim, xuddi shuningdek, turli yozuvlar bir xil ismli maydonlarni o'z ichiga olishi mumkin. Bunda bu maydonlarga murojaat qilishda hech qanday anglashilmovchilik bo'lmaydi, chunki murojaat tashqi yozuv orqali amalga oshiriladi.

Yozuvlardan foydalanib mukammal dasturlar yaratishdan oldin sodda hollarda uning imkoniyatlari bilan tanishib chiqaylik.

Oddiy kombinatsiyali tiplar. Soddalik uchun, bir avlodli struktura orqali yozilgan kombinatsiyali tip ma'lumotlari bilan tanishib chiqaylik.

Yozuvlarni aniqlash (kiritish) quyidagi sintaksis qoida bo'yicha amalga oshiriladi:

<kombinatsiyali tipni aniqlash>::= record < maydonlar ro'yxati> end

<maydonlar ro'yxati>::=<`yozuv seksiyasi>{;<`yozuv seksiyasi>}

<`yozuv seksiyasi>::=<maydon ismi>{,<maydon ismi>}: <tip>

Shu qoidaga asoslanib matematika fanidan yaxshi tanish bo'lgan kompleks sonli tip kiritaylik ($a + bi$ — kompleks son, a, b — haqiqiy sonlar, $i2$=—1) va tip nomini complex deb ataylik:

```
type
  CompLex = record
              re: real;
              im: real;
  end
```

Bu tipni quyidagicha tushuntirish mumkin: *CompLex* tipiga tegishli ixtiyoriy qiymat ikkita hadli (maydonli) yozuvdan tashkil topgan strukturadir. Yozuv maydonlari re va im nomlari bilan ataladi va ular real tipiga tegishlidir.

re va im bir xil tipli bo'lgani uchun ularni bitta ro'yxatga birlashtirib yozsa ham bo'ladi:

```
type
  ComplEx = record
              re, im: real;
            end
```

Endi barcha kompleks (mavhum) qiymatlar qabul qiluvchi o'zgaruvchilarning tiplarini var bo'limida aniqlash mumkin:

```
var
    x, y, z: CompLex;
```

Bu tipdagi o'zgaruvchilarga biror qiymat o'zlashtirish uchun ularning maydonlarini tashkil etuvchilarga qiymat berish kerak bo'ladi. Masalan, x o'zgaruvchiga 4,5 + $+i$ * 6,75 qiymatni o'zlashtirish uchun, re va im ismli maydonlarga qiymat berish kerak:

```
x.re := 4.5;   x.im := 6.75;
```

Bir xil kombinatsiyali tipga tegishli o'zgaruvchilar uchun faqat o'zlashtirish amaligina o'rinli xolos:

```
y := x;
```

Yozuvlarga doir quyidagi misol ustida ular bilan ishlash malakamizni oshiraylik:

Misol: x va y mavhum sonlari ustida qo'shish, ayirish va ko'paytirish amallarini bajarish dasturini tuzing.

Masalani yechish algoritmi:

Agar $x = \mathrm{Re}\ x + i\ \mathrm{Im}\ x$, $y = \mathrm{Re}\ y + i\ \mathrm{Im}\ y$ bo'lsa, ular ustida sanab o'tilgan amallarni bajarish algoritmi quyidagicha bo'ladi:

$u = x + y$, $\mathrm{Re}\ u = \mathrm{Re}\ x + \mathrm{Re}\ y$, $\mathrm{Im}\ u = \mathrm{Im}\ x + \mathrm{Im}\ y$;

$v = x - y$, $\mathrm{Re}\ v = \mathrm{Re}\ x - \mathrm{Re}\ y$, $\mathrm{Im}\ u = \mathrm{Im}\ x - \mathrm{Im}\ y$;

w = x * y, $\mathrm{Re}\ w = \mathrm{Re}\ x * \mathrm{Re}\ y - \mathrm{Im}\ x * \mathrm{Im}\ y$, $\mathrm{Im}\ w = \mathrm{Re}\ y * \mathrm{Im}\ x + \mathrm{Re}\ x * \mathrm{Im}\ y$

Endi mazkur algoritmni dasturda ifoda etamiz:

```
Program LI;
type
  comp = record
  re, im: real
end;
var
x, y, u, v, w: comp;
begin {x va u mavhum sonlarning haqiqiy (Re x, Re y) va
  mavhum (Im x, Im y) qismlarini kiritish}
  readln (x.re, x.im, y.re, y.im);
  { u=x+y}
  u.re:= x.re + y.re;  u.im := x.im + y.im;
  {v=x—y}
  v.re :=x.re — y.re;   v.im := x.im — y.im;
  {w=x*y}
  w.re  := x.re * y.re — x.im * y.im;
  w.im := x.re * y.im + x.im * y.re;
  writeln ('x + y =',u.re,'+',u.im,'*i');
  writeln ('x — y =',v.re,'+',v.im,'*i');
  writeln ('x * y =',w.re,'+',w.im,'*i');
  end.
```

Iyerarxik (shajarali) yozuvlar. Oldingi mavzuda biz kiritgan yozuvlarning maydonlari skalyar (o'zgarmas) miqdorlar edi. Paskal tilida esa, yuqorida ta'kidlaganimizdek, yozuv maydonining tiplariga hech qanday cheklashlar qo'yilmaydi. Shuning uchun yozuvning hadlari yana yozuvlardan, ularning hadlari ham o'z navbatida yozuvlardan tashkil topishi mumkin.

Yozuv maydonining tipi ikki xil usul bilan aniqlanishi mumkin:

- to'g'ridan-to'g'ri kombinatsiyali tip ichida;
- oldindan aniqlangan tiplar orqali.

Iyerarxik (avlodli) struktura bo'yicha aniqlangan yozuvdagi eng past darajada faqat oddiy skalyar tiplargina qatnashadi. Avlod darajasining o'sib borishi davomida tiplarning murakkablik darajasi ham ortib boradi.

Misol sifatida «Ilmiy_xodim» degan yozuvli tipni ko'rib chiqaylik. Bu tipni kiritishdan avval bir nechta yordamchi tiplarni aniqlab olamiz. Bu tiplar asosiy «Ilmiy_xodim» tipini aniqlashda kerak bo'ladi:

```
type rab = packed array [1..15] of char;
   data = record
      kun : 1..31;
      oy : (yanv, fev, mart, apr,may, iyun, iyul, avg,
sent, oct, noy, dek);
      yil: integer;
           end;
Ilmiy_xodim = record
   fish : record {fish — familiyasi, ismi, sharifi}
fam, ism, shar: rab
end;
   tug : data;
   jins: (erkak, ayol);
   malum: (boch, o'rta, o'rtmax, oliy);
   maosh: integer;
   ildar: (darajasiz, fan_nom, doktor); {ildar— ilmiy
darajasi}
   ilunvon: (unvonsiz, dos, kat_il_xodim, prof);
{Ilunvon — ilmiy unvoni}
   manzil: record
        ind: integer;
             shah, ko'cha: rab;
             uy, kvar: integer;
   end;
   tel: record
uy, hizmat: integer
          end;
```

```
end;
```

Endi «*shaxs*» degan «*Ilmiy_xodim*» tipidagi o'zgaruvchini kiritamiz:

```
var
  shaxs: Ilmiy_xodim;
```

Bu o'zgaruvchining qiymatlarini barcha maydonlarning qiymatlarini aniqlash orqali beriladi, masalan:

```
shaxs.fish.ism:= 'Ahmad';
shaxs.fish.fam:='Po'latov';
shaxs.fish.Shar:='Anvarovich';
shaxs.tug.kun:='15';
shaxs.tug.oy:=may;
shaxs.tug.yil:='1953';
shaxs.jins:=erkak;
shaxs.malum:=oliy;
shaxs.maosh:='3000';
shaxs.ildar:=fan_nom;
shaxs.ilunvon:=dos;
shaxs.manzil.ind:='717325';
shaxs.manzil.shah:='Namangan';
shaxs.manzil.ko'cha:='Navoiy';
shaxs.manzil.uy:='20';
shaxs.manzil.kv:='5';
shaxs.tel.uy:='45215';
shaxs.tel.hiz:='40625';
```

Bog'lama operatori. Oldingi mavzuda havola qilingan dasturdan ko'rinib turibdiki, yozuvlar bilan ishlash dasturning matnini juda uzun qilib yuborib, har doim yozuvning to'liq ismlarini yozishga to'g'ri keladi. Jumladan, bu misolda 18 marta «shaxs» so'zi takrorlandi. Shunday takrorlanuvchi yozuv hadlarini kamaytirish maqsadida bog'lama operatori kiritilgan:

<bog'lama operatori>::=<sarlavha> <operator>

<sarlavha>::=with<yozuvli o'zgaruvchilar ro'yxati> do

<yozuvli o'zgaruvchilar ro'yxati>::=<yozuvli o'zgaruvchi>{,<yozuvli o'zgaruvchi>}

Endi bu operatorlarni amalda ishlatilishini ko'rib chiqamiz:

with R ***do*** S

bu yerda ***with*** va ***do*** — xizmatchi so'zlar;

R — kombinatsiyali tipga tegishli o'zgaruvchi;

S — ixtiyoriy operator.

Bog'lama operatorining bajarilishi S operatorining bajarilishi demakdir. Faqat S operatorining ichki qismida R.p yozuvi o'rniga (p— maydon ismi) faqat p yoziladi xolos.

Masalan, oldingi misolimiz uchun *with* shaxs *do* operatoridan keyin o'zgaruvchilarning qiymatlarini yozsak:

fish.ism :='Ahmad'; va h.k.

Bu yerda ko'rinib turibdiki, dastur matni o'zlashtirish operatorining chap tomonidagi ismlarda «Shaxs» so'zi tushirib qoldirilganligi sababli ancha ixchamlashdi.

Operatorning ishlashini yaxshiroq tushunish uchun qisqaroq dasturni ko'rib chiqaylik.

Quyidagi o'zgaruvchilarning qiymatlar tiplarini e'lon qilish bo'limi mavjud bo'lsin:

```
var
   R2: record
A, B, C: integer
        end;
   R3: record
A, D: integer;
        B: record
        C, E : integer
              end
         end;
```

U holda bog'lama operatorining kiritilishi:

```
with R3, B, R2 do
begin
A:=1;  B:=2;  C:=3; D:=4; E:=5
end
```

quyidagi tashkiliy operatorga teng kuchlidir:

```
begin
R2.A:=1;  R2.B:q2; R2.C:q3; R3.D:q4;  R3.B.E:q5
End.
```

3.11.1. Paskal tilida to'plamlarni belgilash

To'plam tushunchasi matematika kursidan yaxshi ma'lum bo'lib, dasturlash tillarida ham keng ma'noda ishlatiladi. To'plamning hadlari bir xil tipli va chekli sondagi bo'lishi kerak. To'plamdagi hadlarning soni 256 tadan ortmasligi yoki umuman to'plam bo'sh bo'lishi ham mumkin. To'plamning hadlari o'rta qavs ichiga olinib, o'zaro vergul bilan ajratilib yoziladi.

Paskal tilida to'plam tiplari sifatida oldingi mavzularda ko'rib chiqilgan ixtiyoriy skalyar tip qabul qilinishi mumkin, faqat Real tipini qabul qilishi mumkin emas.

Paskal tilida yozilgan to'plamlarga misollar:

[ ]	— bo'sh to'plam;
[2,3, 5, 7, 11]	— 2, 3, 5, 7, 11 butun sonlaridan tuzilgan to'plam;
[1..10]	— 1 dan 10 gacha bo'lgan butun sonlar to'plami;
['a', 'b', 'd']	— a, b, d harf (belgi)lardan tuzilgan to'plam;
[oq,qora, sariq]	— 3 ta hadli, sanalma tipdagi qiymatlardan hosil qilingan to'plam;
['d'..'a']	— bo'sh to'plam;
['a'..'d','f'..'h', 'k']	— a, b, c, d, f, g, h, k harflaridan tashkil topgan, belgili tipli to'plam;
[1..1,5..1]	— bir qiymatli, bitta hadli, sonli to'plam, ya'ni [1] ga teng kuchli.

To'plamlarning umumiy nazariyasidagi kabi Paskalda ham to'plam hadlarining tartib joyi hech qanday vazifa bajarmaydi va har bir had faqat bir marta hisobga olinadi:

1) [true, false] va [false, true] to'plamlari ekvivalentdir;

2) [1,2,3,2..5,6,4,3] to'plam [1,2,3,4,5,6] to'plamga ekvivalent, u esa o'z navbatida [1..6] to'plamiga ekvivalent.

Noto'g'ri yozilgan to'plamlardan namunalar:

1) [5.3, 2.5] — to'plam hadlari Real tipidagi son bo'lishi mumkin emas;

2) [9, 7, «P»,0] — to'plam hadlari bir xil tipli bo'lishi lozim edi;

3) ['abc','Axad'] — to'plam hadlari hosilaviy tiplarga tegishli bo'lishi mumkin emas.

To'plamlar ustidagi munosabat amallarining quyidagi jadvalini e'tiboringizga havola qilamiz:

Paskaldagi yozilishi	Matematik yozilishi	Natijaviy qiymat	
		True	False
$A=B$	$A=B$	A va B to'plamlari mos tushadi	aks holda
$A<>B$	$A\neq B$	A va B to'plamlari mos tushmaydi	aks holda
$A<=B$	$A\subseteq B$	A to'plamining barcha hadlari B to'plamga tegishli	aks holda
$A>=B$	$A\supseteq B$	B to'plamining barcha hadlari A to'plamga tegishli	aks holda
x in A	$X\in A$	x hadi A to'plamga tegishli	aks holda

3.11.2.To'plamlar ustida amallar

Agar A va B to'plamlarning barcha hadlari o'zaro bir xil bo'lsa ($A=B$), ular teng (bir xil) deyiladi.

Misol: A=[4,1,3], B=[4,1,3].

Agarda A ning barcha hadlari B ning ham hadlari bo'lsa ($A<B$), A to'plam B to'plamning to'plam ostisi deyiladi.

Misol: A=[1, 2, 3] , B=[1, 2, 3, 4, 5, 6].

A va B to'plamlarining kesishmasidan yana to'plam hosil bo'ladi va natijaviy to'plamning hadlari A to'plamga ham, B to'plamga ham tegishli bo'ladi ($C=A*B$).

Misol: [1,2,3,4,5] * [2,5,6,7,8] = [2,5].

Agar A va B to'plamlar bir xil hadlarga ega bo'lmasa, $C = A*B$ natija — bo'sh to'plam bo'ladi C = [ ].

A va *B* to'plamlarning birlashmasi ($C = A+B$) ularning hech bo'lmasa birortasiga tegishli bo'lgan hadlardan tashkil topadi.

Misol: [1,2,3,4,5] + [2,5,6,7,8] = [1,2,3,4,5,6,7,8].

1) *A* to'plamdan *B* to'plamning ayirmasi deb, shunday to'plamga aytiladiki, natijaviy to'plamning hadlari *A* to'plamga tegishli, lekin *B* to'plamga tegishli bo'lmaydi ($C=A-B$).

Misol: [1,2,3,4,5] — [2,5,6,7,8] = [1,3,4].

2) *x* qiymatning *A* to'plamga tegishliligini aniqlash uchun «in» amalidan foydalaniladi. Agar *x* qiymat *A* to'plamga tegishli bo'lsa; *x* ***in*** *A* = ***true***. Agar *x* qiymat *A* to'plamga tegishli bo'lmasa *x* ***in*** *A* = ***false***.

Misol: *A* =[2, 3, 4, 7] bo'lsa, 6 in A ifodaning natijasi false: 3 in A ifodaning natijasi esa True.

3.11.3. To'plamli tipni berish va to'plamli o'zgaruvchilar

To'plamli tiplarni aniqlash quyidagicha amalga oshiriladi:

Set of <to'plam tipi>;

bu yerda *Set, of* — Paskal tilining xizmatchi so'zlari.

Misol:

1. *Set of* 1..3 — to'plamning bazaviy tipi 1 dan 3 gacha bo'lgan butun sonlar oralig'i, shuning uchun bu to'plamning qiymatlari sifatida 1, 2 va 3 sonlaridan tuzilgan barcha to'plamlarni olish mumkin (xatto bo'sh to'plamni ham): [ ], [1], [2], [3], [1, 2], [1, 3], [2, 3] va [1 2, 3].

2. Faqat shu to'plamlargina, yuqorida aniqlangan to'plamli tipning qiymatlari bo'lishi mumkin.

Set of boolean;

bunday aniqlangan to'plamning qiymatlari to'plami:

[ ], [*true*], [*false*], [*true, false*]

3. type

```
        oy = (yanv, fev, ..., dek);
        yiloyi = Set of oy
        bool = Set of boolean;
    var
```

```
bolalar: Set of (o'g'il, qiz);
oylar: yiloyi;
mantiq: bool;
```

To'plamli qiymatlarni to'plamli o'zgaruvchilarga o'zlashtirish uchun o'zlashtirish operatori ishlatiladi:

```
V:=S;
```

bu yerda V — to'plam tipiga tegishli o'zgaruvchi;
S — to'plamli ifoda.

To'plamli ifodaga misollar:

1) [1, 2, 5, 6,7] * [2, 6] + [3,9], natijasi [2, 3, 5, 6, 9];
2) ([3,4,5]+[1-,3,6,7]) * [5..7]—[6], natijasi [5,7].

Endi to'plamlar ustida bajariladigan amallarga doir quyidagi misol dasturini yarataylik.

Kiritilgan qatorning faqat son, lotin harflari va bo'sh joylardan tashkil topganligini aniqlash dasturini tuzing.

```
Program To'plamlar;
Var
  Str:string;
  L:Byte;
  Tru:Boolean;
Begin
Writeln('Qatorni kiriting');
Readln(Str);
L:=Length(Str); {Kiritilgan simvollar soni}
Tru:=L>0; {True, agar bo'sh qator bo'lmasa}
While Tru and (I>0) do {Qator oxirigacha tekshirish }
  Begin
  Tru:=Str[L] in ['0'..'9', 'A'..'Z', 'a'..'z', ' '];
       {Simvollar to'g'riligini tekshirish}
       Dec(L);                    {Oldingi simvol}
  End;
  If Tru Then Writeln('To'g'ri yozilgan qator')
Else Writeln('Noto'g'ri yozilgan qator');
End.
```

Quyida faqat haqiqiy sonlarni qabul qila oladigan funksiyani yaratish dasturi keltirilgan:

```
Program Set_Of;
```

```
Uses Crt;
Var
   R:Real;
Function Input_R:Real; {Haqiqiy sonlarni kiritish funksiyasi}
Var
S:String[15]; {S o'zgaruvchiga ko'pi bilan 15 belgi sig'adi}
S1:Set Of Char; {Belgilarni tekshirish uchun to'plam}
Ch:Char; {Belgilarni qabul qiladigan o'zgaruvchi}
Code:Integer;
R1:Real;
Begin
S:='';
S1:=['0'..'9','.','—'];
Repeat
Ch:=Readkey;
If Ch in S1 Then {Ch ga kiritilgan belgini S1 to'plam ichidan tekshirish}
          Begin
               S:=S+Ch;
               Write(Ch);
          End;
Until Ch=#13; {Enter klavishi bosilguncha sikl aylanadi}
        Val(S,R1,Code);
        Input_R:=R1;
End;
Begin
   R:=Input_R; {Input_R funksiyasidan foydalanish}
    Writeln;
    Writeln('1-natija= ',R:10:4);
    Writeln('2-natija= ',Input_R:10:4); {Input_R funksiyasidan foydalanish}
    Readln;
End.
```

3.12. MASSIVLAR (JADVAL KATTALIKLAR)

Biz shu paytgacha qiymatlarning oddiy (skalyar) tiplaridan foydalanib, turli xil dasturlar tuzishni o'rgandik. Skalyar tipga tegishli har bir qiymat yagona ma'lumot hisoblanib, **trivial** (qat'iy) tuzilishga egadir. Amalda esa, turli xil hosilaviy tiplar bilan ishlashga, ulardan foydalanib murakkab dasturlar yaratishga to'g'ri keladi. Bu tiplarga tegishli qiymatlarning har biri trivial bo'lmagan tuzilishga ega, ya'ni bu qiymatlar o'z navbatida yana bir nechta qiymatlardan tashkil topadi.

Endi shunday tiplardan biri bo'lgan, dasturlashda eng ko'p qo'llaniladigan dastur ob'ekti — ***massivlar*** bilan tanishib chiqamiz.

3.12.1. Bir o'lchamli massivlar

Massiv — bu bir xil tipli, chekli qiymatlarning tartiblangan to'plamidir. Massivlarga misol sifatida matematika kursidan ma'lum bo'lgan vektorlar, matritsalar va tenzorlarni ko'rsatish mumkin.

Dasturda ishlatiluvchi barcha massivlarga o'ziga xos ism berish kerak. Massivning har bir hadiga murojaat esa uning nomi va o'rta qavs ichiga olib yozilgan tartib hadi orqali amalga oshiriladi:

<massiv nomi>[<indeks>]

bu yerda <indeks> — massiv hadining joylashgan o'rnini anglatuvchi tartib qiymati.

Umuman olganda, <indeks> o'rnida <ifoda> qatnashishi ham mumkin. Indeksni ifodalovchi ifodaning tipini — *indeks tipi* deb ataladi. Indeks tipining qiymatlar to'plami albatta raqamlangan to'plam bo'lishi, shu bilan bir qatorda massiv hadlari sonini aniqlashi va ularning tartibini belgilashi kerak.

Massivlarni e'lon qilishda indeks tipi bilan bir qatorda massiv hadlarining tipi ham ko'rsatilishi kerak. Bir o'lchamli massivni e'lon qilish quyidagicha amalga oshiriladi:

array [<indeks tipi>] of <massiv hadining tipi>;

Ko'pincha <indeks tipi> sifatida cheklanma tiplardan foydalaniladi, chunki bu tipga tegishli to'plam tartiblangan va qat'iy raqamlangandir. Misol uchun, 100 ta haqiqiy sonli hadlardan iborat massiv quyidagicha e'lon qilinadi:

```
array [1..100] of real;
```

Massivlarni e'lon qilish haqida to'liqroq ma'lumot berish uchun turli tipdagi indekslarga oid misollarni e'tiboringizga havola qilamiz:

```
1. array [1000..5000] of integer;
2. array [—754..—1] of byte;
3. array [0..100] of real;
4. array [0..10] of boolean;
5. array [10..25] of char;
6. type
     chegara = 1..100;
     vektor = array [chegara] of real;
     massiv1 = array [115..130] of integer;
     massiv2 = array [—754..—1] of integer;
   var
  A,B: vektor;
     c,d : massiv1;
     e: massiv2;
7. var
     r, t: array [chegara] of real;
     s, q: array [115..130] of integer;
     p: array [—754..—1] of integer;
     k, m: array [1..50] of (shar, kub, doira);
8. type kv1 = (yanvar, fevral, mart);
     var t, r: array [kv1] of real;
9. type
     belgi = array [boolean] of integer;
     belgi_kodi = array [char] of integer;
     var
     k : belgi;
     p : belgi_kodi;
```

Endi massivlar ustida tipik amallar bajaruvchi bir nechta dastur bilan tanishib chiqaylik.

1. Bir o'lchamli n ta hadli (n=30) massiv hadlarini yig'ish.

```
Program L1;
const n=30;
var
i: integer;
x: array [1..n] of real;
S: real;
begin
   for i: =1 to n do readln (x[i]); { massiv hadlarini
kiritish}
   S: = 0;
for i: =1 to n do S: =S+x[i];
   writeln ('natija=', S)
end.
```

2. Bir o'lchamli, n ta hadli (n=30) massiv hadlarining eng kattasini topish va uning joylashgan o'rnini aniqlash.

```
Program L2;
const n=30;
type
gran = 1..30;
   vector = array [gran] of real;
var
x: vector;
S: real;
i, k: integer;
begin
writeln (' x — massivi hadlarini kiriting');
   for i: =1 to n do readln (x[i]);
S: =x[1]; k: =1;
   for i: =2 to n do
   if x[i] > S then
begin
S: =x[i]; k: = i
end;
   writeln ('x massivining eng katta hadi');
   writeln (S);
   writeln ('max(x) ning o'rni', k)
```

```
end.
```

3. n ta hadli ($n = 15$) vektorlarning skalyar ko'paytmasini aniqlash.

```
Program L3;
const n=15;
type
gran = 1..n;
  mas = array [gran] of real;
var
i: byte;
S: real;
  x, y: mas;
begin
writeln ('x va u massiv hadlarini kiriting');
  for i:=1 to n do readln (x[i]);
  for i:=1 to n do readln (y[i]);
S:=0;
  for i:=1 to n do S:= S + x[i] * y[i];
  writeln ('natija', S)
end.
```

3.12.2. Ko'p o'lchamli massivlar

Bir o'lchamli massivlarning hadlari skalyar miqdorlar bo'lgan edi. Umumiy holda esa, massiv hadlari o'z navbatida yana massivlar bo'lishi mumkin. Agar bu massivlar skalyar miqdorlar bo'lsa, natijada ikki o'lchamli massivlar hosil bo'ladi. Ikki o'lchamli massivlarga misol sifatida matematikakursidagi matritsalarni keltirish mumkin. Agar bir o'lchamli massivning hadlari o'z navbatida matritsalar bo'lsa natijada uch o'lchovli massivlar hosil qilinadi va h.k.

Ikki o'lchamli massiv tipini ko'rsatish quyidagicha bajariladi:

array [<indeks tipi>] of array [<indeks tipi>] of <skalyar tip>;

Ikki o'lchamli massivlarning tiplarini bir necha xil usulda aniqlashni quyidagi misolda ko'rib chiqaylik: A matritsa 10 ta satr va 20 ta ustundan iborat bo'lib, uning xadlari haqiqiy tipga tegishli bo'lsin:

1. var
   A: array [1..10] of array [1..20] of real;
2. type matr = array [1..10] of array [1..20] of real;
   var
      A: matr;
3. type gran1 = 1..10; gran2 = 1..20;
   matr = array [gran1, gran2] of real;
   var A: matr;
4. var A: array [1..10, 1..20] of real;

Yana shuni ham aytish mumkinki, ikki o'lchamli massiv indekslarining tiplari turli xil bo'lishi ham mumkin. Bu holni quyidagi misol ustida ko'rib chiqaylik:

```
Program L1;
const n = 24;
type hafkun = (dush, sesh, chor, pay, jum, shan,
yaksh);
   Ishkun = dush..jum;
   detson = array [1..n] of char;
var A: array [boolean] of array [1..n] of char;
    B: detson;
    S: array [1..365] of detson;
```

Ikki o'lchamli massivlar ustidagi bir nechta tugallangan dasturlar bilan tanishib chiqaylik.

1. Matritsalarni qo'shish.

```
Program L2;
const n = 3; m = 4;
  { n — matritsa satrlari soni, m — ustunlar soni}
var i, j: integer;
  A, B, C: array [1..n,1..m] of real;
begin {A, V matritsa hadlarini kiritish}
  for i : = 1 to n do
 for j: =1 to m do
   readln (A[i,j], B[i,j]);
  for i : = 1 to n do
          for j: = 1 to m do
  begin
    C[i,j] := A[i,j] + B[i,j];
      writeln (C[i,j])
  end
```

```
end.
```

2. Matritsani vektorga ko'paytirish.

```
Program L3;
const n =3; m = 4;
type matr = array [1..n, 1..m] of real;
  vect = array [1..m] of real;
var   i, j: byte;
      A: matr;
      B, C: vect;

begin
writeln ('A matritsa hadlarini kiriting');
  for i:=1 to n do
            for j:=1 to m do
              readln (A[i,j]);
  writeln ('B vektor hadlarini kiriting');
  for i:=1 to n do readln (B[i]);
  for i:=1 to n do
    begin
C[i]:=0;
            for j:=1 to m do
            C[i]:= C[i] + A[i,j] * B[j];
            writeln (C[i]);
      end
end.
```

3. Matritsa hadlarining eng kattasini topish va uning joylashgan joyini aniqlash.

```
Program L4;
const n=3; m=4;
var   A: array [1..n, 1..m] of real;
  R: real;
  i, j: byte; K, L: byte;
begin {A matritsa hadlarini kiritish}
  for i: =1 to n do
for j: =1 to m do
            readln (A[i,j]);
  R:= A[1,1]; L:= 1; K:= 1;
  for i: =1 to n do
              for j: =1 to m do
```

```
begin
            if  R< A[i,j]  then
          begin
          R: =A[i,j];
           L: = i;  K: =j;
          end;
                end;
   writeln ('max A=', R);
   writeln ('satr=',L,'ustun =',K);
end.
```

3.13. PROTSEDURA-OPERATORLAR

Dastur tuzish jarayonida uning turli joylarida ma'nosiga ko'ra bir xil, mustaqil xarakterga ega bo'lgan va yechilayotgan asosiy masalaning biror qismini hal qilishga mo'ljallangan murakkab algoritmdan bir necha marotaba foydalanishga to'g'ri keladi. Masalan, matritsalarni ko'paytirish, matritsani vektorga ko'paytirish, chiziqli bo'lmagan tenglamani yechish, chiziqli algebraik tenglamalar tizimini yechish, faktorial hisoblash, yig'indi hisoblash va hokazo kabi masalalarni hal qilish algoritmlari ko'plab masalalarni yechishning bosh algoritmlarida qayta-qayta ravishda, turli boshlang'ich ma'lumotlar bilan birgalikda qatnashishi mumkin. Bunday hollarda, malakali dasturchi dastur matnini ixchamlashtirish, dasturning ishonchlilik darajasini oshirish, dasturni tahrir qilish (otladka) ni tezlashtirish va dasturning umumiyligini (universalligini) ta'minlash uchun protsedura hamda funksiyalardan kengroq foydalangan holda mukammal dastur yaratishga harakat qiladi.

Protsedura va funksiyalar mustaqil dasturli ob'ektlar hisoblanadi. Bu mustaqil dasturli ob'ektni dasturchi o'z xohishiga va undan olinadigan natijalarga ko'ra protsedura yoki funksiya ko'rinishida aniqlashi mumkin. Odatda olinadigan natija yagona qiymatli bo'lsa funksiyadan, olinadigan natijalar soni bir nechta bo'lsa protseduradan foydalanish maqsadga muvofiqdir.

Protsedurani yozish strukturasi xuddi asosiy dastur strukturasi kabi bo'lib, faqat sarlavhalari bilangina farq qiladi xolos:

```
procedure <protsedura ismi>(<formal parametrlar ro'yxati>);
label <metkalar ro'yxati>;
const <o'zgarmaslarni kiritish>;
type <yangi tiplarni aniqlash>;
var <o'zgaruvchilarning tiplarini e'lon qilish>;
<qism dasturgagina tegishli bo'lgan ichki protsedura va funksiyalar e'loni>;
begin
   <protseduraning tana qismi>
end;
```

Protseduralar va funksiyalarni aniqlash asosiy dasturning var (o'zgaruvchilarning tiplarini e'lon qilish) bo'limida bajariladi. Protseduradan dasturda foydalanish uchun uning ismi va faktik parametrlar ro'yxati yoziladi. Shunda protsedura o'ziga belgilangan ishni bajarib, o'zining faktik parametrlari orqali asosiy dasturga o'z natijasini beradi.

Protseduraning e'loni va unga murojaat qilishni kelgusida ko'riladigan misollar orqali o'zlashtirib olamiz.

3.13.1. Parametrsiz protseduralar

Yuqorida aytib o'tilganidek, protsedura hisoblab bergan natijalar uning faktik parametrlari orqali asosiy dasturga uzatiladi. Lekin, ayrim paytlarda protsedura parametrsiz ham bo'lishi mumkin. Bu holda asosiy dasturning barcha parametrlari protsedura parametrlari rolini bajaradi. Parametrsiz protsedurada ham protseduraning barcha bo'limlari saqlanib qoladi, faqat parametrlar ro'yxatigina qatnashmaydi.

Protseduralarni aniqlash va ulardan foydalanishni quyidagi misolda ko'rib chiqaylik:

Misol: $u = \max(x + y,\ x * y)$, $v = \max(0.5, u)$ — berilgan x va y haqiqiy sonlardan foydalanib u va v qiymatlarni aniqlash.

Bu yerda x, y — qiymatlari kiritiladigan haqiqiy tipli o'zgaruvchilar.

1. Masalani yechish dasturining protseduradan foydalanmay tuzilgan shakli:

```
Program max;
var
x, y, u, v: real;
a, b, s: real;
begin
{x, y — miqdorlarni kiritish};
   readln (x,y);
a:= x +y;  b:= x*y;
   if  a > b  then S:= a else S:=b;
u := S;
   a:= 0.5; b:=u;
   if  a > b  then  S:= a   else  S:=b;
   v:=S;
   {olingan natijalar};
   writeln (u, v)
end.
```

Ahamiyat bersangiz, dasturdagi shartli operator ikki marta takrorlanib, bir xil ish bajardi.

2. Masalani yechish dasturining parametrsiz protseduradan foydalanib tuzilgan shakli (endi yuqoridagi dasturda yo'l qo'yilgan kamchilikni protseduralar orqali tuzatishga harakat qilamiz):

```
Program max;
var x, y, u, v: real; a, b, S: real;
procedure max1;
begin
if  a>b  then  S:=a  else  S:=b;
end;
begin
readln (x, y);
   a:= x + y;  b:=x * y;
max1; {max1 protsedurasiga 1-marta murojaat qilinmoqda}
```

```
u:=S;
  a:= 0.5;  b:= u;
max1; {max1 protsedurasiga 2-marta murojaat qilin-
moqda}
v:=S;
  writeln (u,v);
end.
```

Asosiy dasturning operatorlar qismida ikki marta yozilgan max1 parametrsiz protsedurasiga murojaat e'lon qilingan protsedurani ikki marta asosiy dasturga olib kelib ishlatishni ta'minlaydi. Ahamiyat berilsa, ikkinchi dastur birinchi protsedurasiz tuzilgan dasturga ko'ra ixchamroq va soddaroqdir. Biz kiritgan protsedura hozircha faqat ikkita haqiqiy son ichidan kattasini aniqlab berdi xolos. Shuning uchun dasturning hajmini kamaytirishdan erishgan yutuq salmoqli bo'lmadi. Lekin, protseduralar asosan ko'p hajmli matndagi amallarni, vazifalarni bajarishga mo'ljallanadi va bu holda erishilgan yutuq salmog'i ancha yuqori bo'ladi.

Parametrsiz protseduraning asosiy kamchiligi — uning asosiy dasturga va undagi ma'lum parametrlarga bog'lanib qolganligidir.

3.13.2. Parametrli protseduralar

Protsedura bilan asosiy dasturni bog'laydigan bosh omil — protsedura parametrlaridir. Parametrlarni ikkita tipga ajratiladi: qiymatli parametrlar (parametr-qiymat), o'zgaruvchili parametrlar (parametr - o'zgaruvchi).

Parametr-qiymatlar protseduraning ishlash jarayonini ta'minlovchi parametrlar hisoblanadi, ya'ni ular asosiy dastur qiymatlarini protseduraga uzatadigan parametrlardir.

Endi yuqorida ko'rib chiqilgan sonlarning eng kattasini topish algoritmining dasturini qiymatli parametr bilan yozilgan protseduralar orqali amalga oshiraylik:

```
Program max;
var x, y, u, v: real;  S: real;
procedure max2 ( a, b: real );
```

```
begin
if a>b  then S:=a  else  S:=b;
end;
begin
read (x, y);
   max2 (x + y, x * y);  u:=S;
   max2 (0.5 , u);  v:=S;
   writeln (u, v)
end.
```

bu yerda *a, b* — protseduraning qiymatli formal parametrlari.

Protseduraga murojaat qilishda formal va faktik parametrlarning tiplari o'zaro mos kelishi kerak, aks holda dastur xato tuzilgan hisoblanadi. Yuqoridagi dasturdan ko'rinib turibdiki, *a* va *b* formal parametrlar o'rniga natijaviy qiymatlari ma'lum bo'lgan ifodalar qo'yildi. Demak, qiymatli faktik parametrlar o'rniga shu tipli natijaga erishuvchi ifodani yozish mumkin. Bundan tashqari, protsedurada kiritilgan *a* va *b* parametrlar faqat protseduraning ichidagina ma'noga ega, tashqarida, misol uchun asosiy dasturda ular tushunarsiz, qiymatlari aniqlanmagan miqdorlardir. Shuning uchun qiymatli parametrlarga protsedura natijalarini o'zlashtirib, asosiy dasturga uzatib bo'lmaydi.

Yuqorida tuzilgan dasturning asosiy kamchiligi shundaki, topilgan katta son doim *S* o'zgaruvchiga o'zlashtirilayapti. Misolimiz shartiga ko'ra esa natijalar u va v o'zgaruvchilarga o'zlashtirilishi kerak. Shuning uchun, dasturda ikki marta qo'shimcha *u*:=*S* va *v*:=*S* o'zlashtirish operatorlari yozildi.

Bu kamchilikni tuzatish uchun protseduraga yana bir parametrni kiritamiz. Lekin, kiritilgan bu parametr protseduraga qiymat olib kirmaydi balki, protsedura natijasini asosiy dasturga olib chiqib ketadi. Bunday parametrni *parametr-o'zgaruvchi* deb ataladi.

Parametr-o'zgaruvchini parametr-qiymatdan farq qilish uchun protsedurani aniqlashdagi parametrlar ro'yxatida o'zgaruvchi oldidan *var* xizmatchi so'zi yoziladi. Parametr-o'zgaruvchidan so'ng albatta uning tipi ko'rsatib qo'yila-

di. Yuqorida aytganimizdek, formal parametr-qiymat o'rniga protseduraga murojaat qilganda shu tipli ifodani yozish mumkin bo'lsa, parametr-o'zgaruvchi uchun bu hol mutlaqo mumkin emas.

Protsedurani mukammallashtirib borish dinamikasini his etish uchun yana, yuqorida ko'rilgan maksimumni topish misolini parametr-o'zgaruvchi ishlatgan holda ko'rib chiqamiz:

```
Program max;
var
x, y, u, v: real;
procedure max3(a, b: real; var S: real);
begin
if a>b then S:=a else S:=b;
end;
begin
readln (x, y);
max3( x + y, x * y, u); {x+y va x*y ifodalarining kattasi u o'zgaruvchiga o'zlashtirilmoqda}
max3( 0.5, u, v); {0.5 va u ifodalarining kattasi v o'zgaruvchiga o'zlashtirilmoqda}
    writeln( u, v)
end.
```

Shunday qilib, bitta dasturni protseduraning uch xil varianti uchun tuzib chiqib, natijada ixcham va sodda dasturga ega bo'ldik.

Protseduralarni aniqlashda shu paytgacha oddiy tipli parametrlardan foydalanib keldik. Lekin, biz shuni yaxshi bilamizki, Paskal tilida hosilaviy tiplar ham mavjud. Parametr-o'zgaruvchiga hosilaviy yangi tiplar berish xuddi oddiy skalyar tip berish kabi amalga oshirilaveradi. Ammo parametr-qiymatlarda yangi tiplar masalasiga batafsilroq yondashish kerak.

Biz yuqorida eslatib o'tdikki, faktik parametr formal parametr-qiymatga mos tipli ixtiyoriy ifoda bo'lishi mumkin. Lekin, Paskal tilida ixtiyoriy tipli qiymatlar uchun shu tipdagi natija beruvchi hech qanday amal ko'zda tutilmagan. Shuning uchun, bu tiplar uchun faktik

parametrlar faqat shu tipga mos o'zgaruvchilar bo'lishi mumkin xolos. Bunday hol, xususiy holda massivlar uchun ham o'rinlidir.

Faraz qilaylik, dasturda o'zgaruvchilar quyidagicha e'lon qilingan bo'lsin:

```
const  n=20;
type vector = array [1..N]  of  real;
var      u, v: real;
   x, y: vector;
```

Bu yerda $u = \max\{x_i\}$, $v = \max\{y_i\}$ larni aniqlash talab qilinayotgan bo'lsin.

Vektorning eng katta hadini topishni albatta protsedura ko'rinishida tashkil qilamiz:

```
procedure max1 (A:vector;  var S: real);
var   i: integer;
begin
S:=A[1];
   for i:=2  to  n  do  if  A[i] > S     then    S:= A[i]
end.
```

Asosiy dasturda bu protseduraga murojaat

```
max1 (x, u);  max1 (y, v);
```

ko'rinishda amalga oshiriladi.

Protseduradagi A vektorni parametr-qiymat sifatida yozib qo'yganimiz uchun, protseduraga qilinayotgan har bir murojaatda A vektorga mos ravishda X va Y vektorlar ko'chirib yoziladi va so'ng protsedura o'z ishini bajaradi. Biz bilamizki, bir tarafdan, massivlarning ustida ko'chirish amalini bajarishga ancha vaqt ketadi, ikkinchi tarafdan, har safar yangidan protseduraga qilingan murojaatda A vektor uchun xotiradan qo'shimcha joy ajratiladi. Shuning uchun, protseduraning sarlavhasida quyidagicha almashtirish qilsak, yuqoridagi ikki kamchilikni bartaraf qilgan bo'lamiz:

```
procedure max1 (var A: vector; var S: real);
```

Endi protsedurani e'lon qilish, undan foydalanib dastur yaratish malakasini hosil qilganimizdan so'ng uni e'lon qilishning sintaksis qoidalarini ko'rib chiqaylik.

Protsedurani aniqlash (e'lon qilish) quyidagicha amalga oshiriladi:

<protsedurani aniqlash>::=<protsedura sarlavhasi>; <blok>

Bu yerda <blok> tushunchasi to'liqligicha <dastur tanasi> tushunchasi bilan bir xil sintaksis qoida asosida aniqlangani tufayli bu tushunchaga ortiq qaytib o'tirmaymiz.

Endi esa <protsedura sarlavhasi>ga ta'rif beramiz:

<protsedura sarlavhasi>::=Procedure <protsedura ismi>| Procedure <protsedura ismi>(<formal parametrlar ro'yxati>)

Protsedura ismi dasturchi tomonidan tanlanadigan oddiy identifikator hisoblanadi.

Formal parametrlar ro'yxati quyidagicha aniqlanadi:

< formal parametrlar ro'yxati >::=<formal parametrlar seksiyasi > {; < formal parametrlar seksiyasi >}

Formal parametrlar seksiyasi deganda protsedura parametrlarining parametr-qiymat va parametr-o'zgaruvchilardan iborat bo'lishi tushuniladi:

< formal parametrlar seksiyasi >::={< ism > {, < ism >}: <ism tipi> var <ism>{,<ism>}:<ism tipi>

bu yerda <ism> — formal parametrlar sifatida ishlatiladigan identifikator.

Endi yuqoridagi aniqlashlarga tushuntirishlar berib o'tamiz.

Yuqoridagi **Bekus—Naur formulalari**dan ko'rinib turibdiki, formal parametrlar ro'yxati (agar u mavjud bo'lsa) bitta yoki bir nechta o'zaro nuqta- vergul (;) belgisi bilan ajratilgan seksiyalardan tashkil topgan. Har bir seksiyada esa o'z navbatida bitta yoki bir nechta o'zaro nuqta-vergul bilan ajratilgan formal parametrlar qatnashishi mumkin. Protseduradagi formal parametrlar sonini dasturchining o'zi protsedurani aniqlash mohiyatidan kelib chiqqan holda tanlaydi.

Misol:

Procedure *P*(*A*:Char; *B*:Char; Var *C*:Real; Var *D*:Real; *E*:Char);

bu yerda formal parametrlar ro'yxati beshta seksiyadan iborat: *A,B,E* lar Char tipli qiymatlar, *C, D* lar Real tipidagi o'zgaruvchilar. Shu bilan bir qatorda har bir seksiya faqat bitta parametrni o'z ichiga olmoqda.

Bir xil tipli, hamda ketma-ket joylashgan qiymatlarni va o'zgaruvchilarni bitta seksiyaga birlashtirib protsedura sarlavhasini quyidagicha yozish ham mumkin:

Procedure P(A,B:Char; Var C,D:Real; E:Char);

Shunday qilib, *seksiya* deganda bir xil tipli parametr-qiymatlar yoki parametr-o'zgaruvchilarning ro'yxatini tushunish mumkin.

Ko'pchilik boshlovchi dasturchilar yo'l qo'yadigan quyidagi xatoliklardan ehtiyot bo'lmoq zarur:

Procedure *P*(Var *X*:Real; *Y*:Real);

bu sarlavha

Procedure *P*(Var *X,Y*:Real);

sarlavha bilan bir xil emas.

Aniqlangan protseduraga murojaatni yoki protsedura operatoridan qanday foydalanishni aniqlashni ko'rib chiqaylik (yaratilgan protsedurani «Aktivlashtirish», ya'ni ishlatish):

<protsedura operatori>::=<protsedura ismi>|<protsedura ismi>(<faktik parametrlar ro'yxati>)

Agar protsedura aniqlanishida parametrsiz bo'lsa, unga murojaat qilish ham faqat protsedura ismini yozish bilangina amalga oshiriladi.

Agar protsedura aniqlanishida parametrli bo'lsa, albatta protsedura-operator ham unga mos faktik parametrlar ro'yxatiga ega bo'ladi. Shu parametrlar orqali protseduraga murojaat qilinayotganda formal parametrlar faollashtiriladi:

< faktik parametrlar ro'yxati >::=< faktik parametr > {, < faktik parametr >}

3.14. LOKALLASHTIRISH TAMOYILI

Hajmi katta va murakkab dasturlarni ishlab chiqishda, tabiiyki katta qiyinchiliklarga duch kelinadi. Katta, kompleks dasturlarni lozim bo'lgan muddatda yaratishga esa bitta dasturchining vaqti yetmaydi. Bunday hollarda dasturlarni ishlab chiqish uchun dasturchilarning katta guruhini jalb etishga to'g'ri keladi. Yagona dasturni yaratishda parallel ravishda ish olib boriladi va protsedura hamda funksiyalarning roli juda katta bo'ladi. Bajarilishi kerak bo'lgan ishni mustaqil bo'limlarga ajratilib, har bir mustaqil ish alohida dasturlanib, keyin ular yagona — asosiy dasturga birlashtiriladi.

Asosiy dasturda ishlatiluvchi o'zgaruvchilar va protseduralar parametrlarini qanday tanlab olish kerak degan muammo bajariladigan ishning eng og'ir qismlaridan biri bo'lib qoladi. Agar ularni bir-birlariga bog'lab yuborilsa u holda asosiy dasturdagi biror o'zgaruvchiga kiritilgan o'zgartirish, protsedurada ishlatilgan va shu o'zgaruvchiga bog'liq barcha ishlarni qaytadan tahlil qilib, tekshirib chiqishga to'g'ri keladi. Bunday chalkash va og'ir ishni bajarishning qiyinligi dastur tuzishda bir nechta dasturchining parallel ravishda, ish olib borishiga to'sqinlik qiladi.

Shuning uchun protsedura va funksiyalarni yozishda har bir dasturga o'zi yechayotgan masalaga muvofiq holda turli xil ichki o'zgaruvchilar, dasturli ob'ektlar o'zgaruvchilarining turli qiymatlarini tanlab olish huquqi beriladi. Hattoki bitta o'zgaruvchini turli xil vazifalarda ishlatsa ham bo'ladi. Paskal tilida bunday masalani hal qilish uchun lokallashtirish tamoyili ishlab chiqilgan. Bunda protsedura yoki funksiyada ishlatilgan o'zgaruvchi shu protsedura yoki funksiyaning ta'sir doirasida (ichida) gina o'z qiymatini saqlab qoladi. Protsedura va funksiyalarning ichida aniqlanib, qiymatlangan o'zgaruvchilarni *lokal (ichki) o'zgaruvchilar* deb ataladi. Tashqarida, ya'ni asosiy dasturda kiritilgan o'zgaruvchilar esa umuman olganda dasturning ixtiyoriy joyida o'z qiymatini saqlab qola oladi.

Bu o'zgaruvchilarni *global (tashqi) o'zgaruvchilar* deb ataladi.

Quyidagi misolda lokallashtirish tamoyili yaqqol ko'zga tashlanadi:

```
Program L1;
  const
 n = 1;
  var
 t: real;
 x: char;
 procedure P (x, y: real);
 var n: real;
 begin
    n:= x+t; t:=y;
    writeln( n, t, x);
 end;
begin
  t:= n/2; x:= '+';
 P(n,0.8); writeln(n,t,x);
end.
```

bu yerda t — asosiy dasturning global o'zgaruvchisi; x, y — R protsedurasining formal parametrlari; n — P protseduradagi lokal o'zgaruvchi.

3.15. PROTSEDURA-FUNKSIYALAR

Matematika kursidan funksiya tushunchasi bizga yaxshi tanish bo'lib, uning yordamida funksiya va argument o'rtasidagi bog'liqlik aniqlanadi. Paskal tilida ham funksiya tushunchasi kiritilgan bo'lib, uni shartli ravishda ikki turga ajratsak bo'ladi: standart funksiyalar, dasturchi tomonidan aniqlangan protsedura-funksiyalar. Standart funksiyalar har bir algoritmik til uchun aniqlanib, amalda ko'p uchrab turuvchi funksiyalarning qiymatlarini hisoblab berishga mo'ljallangan. Masalan, sin (x), cos(x), exp(x), abs(x), sqrt(x) va h.k.

Xuddi standart funksiyalar kabi dasturchi ham o'zi uchun zarur, mustaqil dastur ob'ektlarini funksiyalar ko'rinishida aniqlab, undan kerakli paytda foydalanishi mumkin.

Funksiya Paskal tilida quyidagi struktura bo'yicha aniqlanadi:

```
function <funksiya ismi>(<formal parametrlar ro'yxati>):
  <funksiya qiymatining tipi>;
  label
    <metkalar ro'yxati>;
  const
    <o'zgarmaslarning qiymatlarini aniqlash>;
  type    <yangi tiplarni kiritish>;
  var
    <o'zgaruvchilarning tiplarini e'lon qilish>;
    <funksiya uchun ichki protseduralar va funksiyalarni aniqlash>;
  begin
          <funksiyaning tana qismi>
  end;
```

Yuqorida eslatib o'tganimizdek, funksiyalar ham protseduralar kabi mustaqil dasturlar hisoblanib, asosiy dastur orqali boshqariladi va xuddi asosiy dastur va protseduraga o'xshash strukturada yoziladi.

Funksiyani ham protsedura kabi dasturning var (o'zgaruvchilar tiplarini e'lon qilish) bo'limida aniqlab qo'yiladi. Protsedura uchun aytilgan gaplarning deyarli barchasi funksiya uchun ham o'rinlidir. Funksiyaning protseduradan asosiy farqi quyidagilardir:

- funksiya sarlavhasi boshqacha aniqlanadi;
- funksiyaning ishi davomida olinadigan natija funksiyaning ismiga o'zlashtiriladi, ya'ni funksiyaning tana qismida albatta funksiya ismiga mos tipli qiymat o'zlashtirilgan bo'lishi kerak;
- funksiyadan asosiy dasturga uning ismi orqali bittagina qiymat beriladi.

Funksiyaga murojaat ham xuddi protseduradagi kabi amalga oshiriladi, lekin funksiyaning mos tipli ifodada qatnashish kabi qo'shimcha imkoniyati ham mavjud.

Endi funksiyani aniqlash va unga murojaat qilishni to'liqroq o'rganish uchun quyidagi misolni e'tiboringizga havola qilamiz:

Misol: $f(n) = n!$ ($n!$ = 1 * 2 * 3 * ...* n— faktorial) funksiyadan foydalanib,

$Y = \frac{20!+3!}{5!+31!} * \frac{(k+1)}{m!}$ — ifodani hisoblashni tashkil qiling:

```
Program L1;
var
k, m, i :integer; y: real;
function fact (n: integer): integer;
var
j: integer; P: byte;
begin
j:=1;
for p:=1 to  n do j:= j * p;
fact :=j;
end;
begin
readln (k, m);
   y:= (fact (20) + fact(3))/(fact(5) + fact (31)) *
fact(k+1) / fact(m);
   writeln( y);
end.
```

Funksiyalarni aniqlashda doim shunday harakat qilish lozimki, uning tana qismida formal parametrlar va funksiyani aniqlash uchun zarur bo'lgan lokal o'zgaruvchilargina qatnashsin. Dasturning global o'zgaruvchisiga iloji boricha protsedura yoki funksiya ichidan turib qiymat bermaslik kerak, aks holda dastur xato natija berishi mumkin.

Misol:

```
   Program m1;
Var      x,y: integer;
Funktion  f(t: integer): integer;
Begin
   f:=t*t;
   x:=7;
end;
begin
   x:=5; writeln(x);
   y:=f(2)+x
   writeln(x,y)
end.
```

Bu dasturning ishlashi natijasida $x=5$, $y=11$ va $x=7$ qiymatlar ekranga chiqariladi, ya'ni funksiyaning ichki qismidagi $x=7$ qiymati asosiy dasturdagi natijaviy qiymatlarga o'z ta'sirini o'tkazmoqda.

Paskal tilida protsedura-funksiyalar bilan ishlashda funksiyalarning rekursivlik xossasidan foydalanish imkoniyati yaratilgan.

Rekursiya tushunchasiga misol qilib oddiy faktorial hisoblashni keltirish mumkin:

$$n! = \begin{cases} 1 & \text{agar} \quad n = 0, \\ n \cdot (n-1)! & \text{agar} \quad n > 0; \end{cases}$$

bu yerda ko'rinib turibdiki $n!$ qiymati $(n-1)!$ orqali aniqlanayapti (rekursiya so'zi o'zi orqali o'zini aniqlash ma'nosini anglatadi).

Paskal tili ham funksiyalarni rekursiv aniqlash imkoniyatini beradi. Funksiyani rekursiv aniqlash uning tana qismida o'ziga o'zi murojaat qilish orqali amalga oshiriladi.

Yuqoridagi faktorial hisoblashni rekursiv funksiyalar orqali amalga oshiraylik:

```
program L1;
var
n: integer; y: integer;
function fact(m: integer): integer;
var
k: integer;
begin
if  m=0  then fact:=1 else fact:= fact(m—1) * m;
end;
begin
readln (n);
   y:= fact (n);
   writeln(y);
end.
```

Funksiyalarni rekursiv aniqlash qisqa va tushunarli tilda bo'ladi. Rekursiv bo'lmagan tarzda aniqlash esa uzoq vaqtni oladi va funksiyaning samarasini kamaytiradi. Rekursiv

aniqlash holida esa sarflangan EHM vaqti va xotira ancha yuqori bo'ladi.

Yuqorida ko'rib chiqilgan va aniqlangan barcha protsedura hamda funksiyalarning parametrlari qandaydir tipli qiymat yoki o'zgaruvchilar bo'lgan edi. Ammo shunday hollar ham uchrab turadiki, ayrim parametrlarni funksiyalar yoki protseduralar orqali aniqlash lozim bo'ladi. Bu holga misol sifatida

$$y=\int_a^b f(x)dx \quad \text{va} \quad z=\int_c^d g(x)dx$$

aniq integrallarni trapetsiya usulida taqribiy hisoblash dasturini ko'rib chiqamiz.

Bu yerda a, b, c, d — qiymatlari beriladigan o'zgaruvchilar; $f(x)=e^{2x}+\sin 6x$, $g(x)=x^2-3x^3\cos x$.

Aniq integrallarni trapetsiya usuli yordamida hisoblash algoritmi quyidagi formula asosida bajariladi:

$$y=\int_a^b f(x)dx \approx h\left[\frac{f(a)+f(b)}{2}+\sum_{k=1}^{n-1} f(a+kh)\right],$$

bu yerda $h=\frac{b-a}{n}$, n — $[a, b]$ oraliqni bo'lishlar soni.

```
Program L1;
var
a, b, c, d, y, z: real;
function f(x: real) : real;
begin
f:= exp(2*x)+sin(6*x)
end;
function g(x: real) : real;
begin
g:= sqr(x) — 3*x*sqr(x)*cos(x)
end;
procedure int(A, B: real; function F(x: real): real; var R:
real);
const
n=20;
var
```

```
   x, h: real; k: integer;
   begin
   h:= (B — A)/n;  R:=(f(A)+f(B))/2;
   for k:=1  to  n—1  do
                 R:= R+ f(a + k * h);
   R:= R * h;
   end;
{asosiy dasturning operatorlar bo'limi}
   begin {1-integral chegaralarini kiriting}
      read ( a, b);
   int (a, b, f, y);
      writeln( 'y=',y);
{2-integral chegaralarini kiriting}
   read (c, d); int ( c, d, g, z);
   write ('z=', z);
   end.
```

3.16.TURBO — PASKALDA MODULLAR

3.16.1. Turbo—Paskalning modullari. Foydalanuvchi modulini yaratish.

Shaxsiy kompyuterlarning eng katta kamchiligi ularda amaliy dastürlar kutubxonasining to'liq emasligidadir. Katta EHMlarda dastur tuzuvchilar uchun juda katta dasturlar kutubxonasi xizmat qilar va ulardan foydalanib tuzilgan dasturlar o'zlarining ishonchlilik darajasi bilan yuqori turar edi. Shaxsiy kompyuterlarning bu kamchiligini yo'qotish uchun Turbo—Paskalda modullar tushunchasi kiritilgan. Umuman olganda, har bir malakali dasturchi o'z dasturini protsedura va funksiyalardan foydalanib tuzadi. Lekin, bu protsedura va funksiyalardan boshqa dasturlarda foydalanish uchun ularning matnlarini qayta ko'chirib yozish lozim bo'ladi.

Turbo — Paskalda bu masalani yechish uchun modullar yaratilib, ularni kompilyatsiya qilinadi va bu moduldan boshqa dasturlarda bemalol foydalanilaveriladi.

Turbo — Paskal tilining yaratuvchilari quyidagi zarur va foydali modullarni yaratib, dasturchilar uchun juda katta qulayliklar yaratishgan:

1) *System moduli*— standart protsedura va funksiyalarni o'z ichiga olib, avtomatik tarzda barcha dasturlar uchun ochiqdir;
2) *DOS moduli* — MS DOS operatsion tizimi bilan ishlashni tashkil qiluvchi funksiya va protseduralardan tashkil topgan;
3) *Crt moduli* — ekran, klaviatura va IBM rusumidagi kompyuterlarning tovushli dinamigi bilan ishlash protseduralarini o'z ichiga olgan;
4) *Graph moduli* — kompyuterning grafik imkoniyatlaridan foydalanib yaratilgan funksiya va protseduralarning katta to'plami;
5) *Printer moduli* — bu kichkinagina modul printer qurilmasi bilan ishlashni osonlashtiradi;
6) *Overlay moduli* — katta dasturlarni bir nechta bo'laklarga ajratishning kuchli vositasi bo'lib, bir qancha protseduralar va funksiyalardan tashkil topgan.

Modullardan foydalanish uchun dastur sarlavhasi

Program <programma nomi>;

dan keyin quyidagi qator yozilishi kerak:

Uses <modul ismi>;

Agar dasturda bir nechta modul ishlatilsa, ularning ismlari ketma-ket yozib qo'yiladi:

Uses <modul ismi1>,<modul ismi2,>,...,<modul ismiN>;

Turbo — Paskal bizga o'zimizning modullarimizni yaratib olish imkonini ham beradi. Foydalanuvchi modullari quyidagicha tuzilishda bo'ladi:

```
Unit<modul ismi>;
Interface
    ...
    { ochiq e'lonlar bo'limi — interfeys seksiyasi}
    ...
Implementation
```

{ yopiq e'lonlar bo'limi}

...

Begin

...

{ Initsializatsiya bo'limi}

...

End.

Agar modul o'z ichida boshqa modullardan foydalansa Interface xizmatchi so'zidan keyin

Uses <modullar ro'yxati>;

yoziladi.

Interfeysli bo'lim modulning bir qismi bo'lib, Interface va Implementation so'zlari orasida joylashadi. Bu bo'limda o'zgarmaslar, ma'lumotlar tipi, o'zgaruvchilar, protsedura va funksiyalarni aniqlash mumkin. Bu kiritilganlar mazkur modulda qatnashuvchi barcha dasturlar va modullarda bemalol ishlatilishi mumkin. Bo'limda sanab o'tilgan protsedura va funksiyalarning tana qismlari *Implementation* so'zidan keyin aniqlanadi (ularning sarlavhalari aynan saqlanib qolishi kerak). Bu bo'limda ham, faqat shu bo'lim uchungina «Ko'rinadigan» (ishlatilishi mumkin bo'lgan) e'lonlar bo'limi qatnashishi mumkin.

Initsializatsiya* bo'limi *Begin* va *End* so'zlari ichiga olib yoziladi. Agar *Begin* so'zi tushirib qoldirilgan bo'lsa, demak bu bo'lim yo'q hisoblanadi. Initsializatsiya bo'limida boshqarishni asosiy dasturga uzatgungacha bajariladigan operatorlar joylashgan bo'ladi. Bu operatorlar asosan dasturni ishga tushirishga tayyorlab beradi.

Misol sifatida *X* va *Y* sonlarining maksimumi va minimumini aniqlovchi modulni yarataylik:

```
Unit Stud;
Interface      {ochiq e'lonlar bo'limi — interfeys bo'limi}
        function min(x,y:integer):integer;
        function max(x,y:integer):integer;
Implementation      {yopiq e'lonlar bo'limi}
```

* Initsializatsiya — ob'ektni tanitish jarayonini anglatadi.

```
function min(x,y:integer):integer;
Begin
      if  x<=y     then min:=x else  min:=y;
End;
function max(x,y:integer):integer;
Begin
      if  x>y then max:=x else  max:=y;
End;
Begin
  {Initsializatsiya bo'limi yo'q}
End.
```

Biz zarur modulni hosil qildik, endi uni kompilyatsiya qilishimiz lozim. Kompilyatsiya natijasida *Stud.tpu* ismli fayl hosil qilinishi kerak. Kompilyatsiya qilinmagan modulning ismi esa shunga mos holda *Stud.pas* bo'lishi kerak.

Bu moduldan foydalanish dasturi quyidagicha bo'lishi mumkin:

```
Uses Stud;
Var
   A,b,c,d:integer;
Begin
    Write('A va B larni kiriting >');
     Readln(a,b);
     C:=max(a,b);
     Writeln('Max= ',C);
      C:=min(a,b);
      Writeln('Min= ',C);
      D:=max(a,b)+min(a,b);
      Writeln('Max+Min=',D);
End.
```

Quyida esa ekran rangini tanlash moduli misol sifatida ko'rsatilgan:

```
Unit Colors;
Interfase
Type
     Colortype =Array[0..15] of Byte;
Const
```

```
        Black:byte=0;blue:byte=1;
        Green:byte=2;cyan:byte=3;
        Red:byte=4;magenta:byte=5;
        Brown:byte=6;lightgray:byte=7;
        Darkgray:byte=8;lightblue:byte=9;
        Lightgreen:byte=10;lightcyan:byte=11;
        Lightred:byte=12;lightmagenta:byte=13;
        Yellow:byte=14;white:byte=15;
Var
        Currcolors:colortype absolute Black;
        Procedure setMonoColors;
      Procedure setColorColors;
Implementation
Const
ColorColors:Colortype=(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15);
MonoColors:ColorType=(0,1,7,7,7,7,7,7,7,7,7,7,7,7,15,15);
Procedure SetMonoColors;
Begin
        CurrColors:=MonoColors;
End;
Procedure SetColorColors;
Begin
        CurrColors:=ColorColors;
End;
Var
        Ch:Char;
Begin
   Write( );
    Readln(ch);
   If ch in ['M','m','M','m'] then SetMonoColors;
End.
```

3.16.2. System modulining protsedura va funksiyalari

Yuqorida aytganimizdek, System modulining protseduralari va funksiyalari barcha dasturlar uchun ochiq bo'lib, ulardan keng foydalanish mumkin. System modulining ismini modullar ro'yxatida ko'rsatish shart emas.

Quyida ushbu modulning ma'lum bir qism protseduralar va funksiyalari bilan qisqacha tanishib chiqamiz:

1. Dastur ishini bajarish protseduralari.

Exit ***protsedurasi.*** Vazifasi: aktiv ishchi blokidan chiqish, bajarilayotgan ishni yakunlash;

Aniqlanishi: Exit.

Halt ***protsedurasi.*** Vazifasi: dastur bajarilishini to'xtatadi va OS ga boshqarishni qaytaradi;

Aniqlanishi: Halt[(ExitCode:Word)];

bu yerda majburiy bo'lmagan (o'rta qavs majburiy emas belgisi) ExitCode parametri dasturning yakunlanish kodini beradi, agar bu parametr bo'lmasa bu kod nolga teng bo'ladi.

RunError ***protsedurasi.*** Vazifasi: dasturning bajarilishini to'xtatadi va bajarilish vaqtidagi xatolarni aniqlaydi;

Aniqlanishi: RunError[(ErrorCode:Word)];

bu yerda ErrorCode parametrining xatolik raqami haqidagi ma'lumoti ekranga chop etiladi.

2. Tiplarni almashtirish funksiyalari.

Chr ***funksiyasi.*** Vazifasi: ASCII jadvalidan berilgan butun songa mos bo'lgan belgini aniqlaydi;

Aniqlanishi: Chr(N:Byte):Char;

bu yerda N — belgining ASCII jadvalidagi qiymatini ifodalovchi butun musbat son.

Ord ***funksiyasi.*** Vazifasi: sanalma tipli qiymatlar bo'yicha uning tartib sonini aniqlash;

Aniqlanishi: Ord(X):LongInt;

bu yerda X — sanalma tipli qiymat.

Round ***funksiyasi.*** Vazifasi: haqiqiy tipli qiymatni yaxlitlab, katta butun, son hosil qiladi;

Aniqlanishi: Round(X:Real):LongInt;

Trunc ***funksiyasi.*** Vazifasi: haqiqiy tipli qiymatning kasr qismini tashlab yuborib, butun son hosil qiladi;

Aniqlanishi: Trunc(X:Real):LongInt;

3. Arifmetik funksiyalar.

Bu funksiyalarning yozilishi va ulardan foydalanish qoidalari o'zlashtirish operatori mavzusida to'liq berib o'tilganligi uchun ularga to'xtalib o'tirmaymiz.

4. Sanalma tip funksiyalari va protseduralari.

Dec ***protsedurasi.*** Vazifasi: o'zgaruvchi qiymatini kamaytiradi;

Aniqlanishi: Dec(Var X[;n]:LongInt);

bu yerda yozilishi majburiy bo'lmagan «*n*» o'zgaruvchi argumentning qiymatini qanchaga kamaytirish lozimligini ko'rsatadi. Bu o'zgaruvchi yozilmasa argument qiymati bir soniga kamayadi.

Inc ***protsedurasi.*** Vazifasi: o'zgaruvchi qiymatini orttiradi;

Aniqlanishi: Inc(Var X[;n]: Integer);

bu yerda Dec protsedurasiga teskari ish bajariladi.

Odd ***funksiyasi.*** Vazifasi: argumentning toq yoki juft sonligini tekshiradi;

Aniqlanishi: Odd(X:LongInt):Boolean;

bu yerda natijaviy qiymat rost (True) bo'lsa son toq, aks holda son juft.

Pred ***protsedurasi.*** Vazifasi: argumentning oldingi qiymatini aniqlash;

Aniqlanishi: Pred(X);

Succ ***protsedurasi.*** Vazifasi: argumetning keyingi qiymatini aniqlash;

Aniqlanishi: Succ(X);

5. Satrlar bilan ishlash funksiyalari va protseduralari.

ConCat ***funksiyasi.*** Vazifasi: satrlarni ketmaket ulaydi;

Aniqlanishi: ConCat(S1[,S2,...,SN]:String):String;

bu yerda S1 qatorni keyingi sanab o'tilgan qatorlar bilan ularning yozilish tartibida ulaydi.

Copy ***funksiyasi.*** Vazifasi: satr ichidan yangi satr hosil qilish;

Aniqlanishi: Copy(S:String;Index,Count:Ineger):String;

bu yerda S — berilgan satr;

Index-S satrining nechanchi belgisidan boshlab yangi satr hosil qilish kerakligini aniqlaydi;

Count — yangi satrdagi belgilar soni.

Delete ***protsedurasi.*** Vazifasi: berilgan satr ichidan satr ostini olib tashlaydi;

Aniqlanishi: Delete(Var S:String;Index:Integer; Count:Integer);

bu yerda S — berilgan satr;

Index — shu tartib raqamli belgidan boshlab S satrdan satr osti olib tashlanadi;

Count — olib tashlanadigan satrdagi belgilar soni.

Insert ***protsedurasi.*** Vazifasi: berilgan satrga yangi satr qo'shadi;

Aniqlanishi: Insert(S1:String; Var S2:String; Index: Integer);

bu yerda S2—berilgan satr;

S1— qo'shiladigan satr;

Index—S2 satrning qaysi tartib raqamli hadidan boshlab yangi satr qo'shilishini anglatadi.

Length ***funksiyasi.*** Vazifasi: satr uzunligini aniqlaydi;

Aniqlanishi: Length(S:String):Integer;

bu yerda S satridagi belgilarning soni aniqlanadi:

Pos funksiyasi. Vazifasi: satrdan satr ostini qidiradi;

Aniqlanishi: Pos(SubStr,S:String):Byte;

bu yerda SubStr — S satrda qidirilayotgan satr osti.

Agar SubStr satr osti S satrida topilsa, Pos funksiyasi mos kelgan birinchi belgining tartib raqamini beradi, agar bu satr osti S satrda bo'lmasa, funksiya nul qiymat beradi.

Str ***protsedurasi.*** Vazifasi: sonli qiymatni uning satrli ko'rinishiga o'tkazadi;

Aniqlanishi: Str(X[:Width[:decmals]];Var S:String);

bu yerda yozilishi shart bo'lmagan Width va decmals parametrlari mos ravishda S satrining hadlar sonini va haqiqiy sonning verguldan keyingi hadlar sonini ifodalaydi.

Val ***protsedurasi.*** Vazifasi: satrli qiymatni uning sonli ko'rinishiga o'tkazadi;

Aniqlanishi: Val(S:String;Var V; Var Code: Integer);

bu yerda S — berilgan satr;

V — S satrning uning sonli ko'rinishiga o'tkazganidan keyin hosil bo'lgan sonni saqlash joyi;

Code — butun tipli o'zgaruvchi.

Agar S satrni songa aylantirib bo'lmasa Val protsedurasi bajarilganidan so'ng Code o'zgaruvchisi nul qiymatni qabul qiladi.

6. Parametrlar bilan ishlash funksiyalari.

Dasturni ishga tushirishda unga uzatiladigan qiymatlarni parametrlar deb hisoblaymiz.

Misol: Tizimli dasturlashning quyidagi **Copy** buyrug'ini ko'raylik:

Copy file.dat c:\prog\file2.dat

ya'ni, aktivlashgan katalogdagi file.dat faylining nusxasini «S» diskning prog katalogiga file2.dat nomi bilan ko'chirish. Bu yerda Copy buyrug'iga parametr sifatida «filel.dat»va «file2.dat» qatori uzatilmoqda.

Turbo—Paskalda parametrlar bilan ishlash uchun ParamCount va ParamStr funksiyalari mavjud.

ParamCount ***funksiyasi.*** Vazifasi: buyruqli qatordan dasturga uzatilgan parametrlar sonini aniqlaydi;

Aniqlanishi: ParamCount:Word;

ParamStr ***funksiyasi.*** Vazifasi: ko'rsatilgan raqamdagi parametrni aniqlaydi;

Aniqlanishi: ParamStr(N:Integer):String;

7. Adreslar bilan ishlash funksiyasi.

Addr ***funksiyasi.*** Vazifasi: ko'rsatilgan ob'ektning adresini (o'rnini) aniqlaydi;

Aniqlanishi: Addr(X):Pointer;

bu yerda X — ixtiyoriy tipli o'zgaruvchi yoxud dasturda e'lon qilingan protsedura yoki funksiya ismi.

Seg ***funksiyasi.*** Vazifasi: ko'rsatilgan o'zgaruvchi adresining segment qiymatini aniqlaydi;

Aniqlanishi: Seg(X):Word;

bu yerda X — ixtiyoriy tipli o'zgaruvchi yoxud dasturda e'lon qilingan protsedura yoki funksiya ismi.

8. Boshqa protsedura va funksiyalar.

FillChar ***protsedurasi.*** Vazifasi: ko'rsatilgan qiymat bilan ketma-ket kelgan va chekli sondagi baytlarni to'ldiradi;

Aniqlanishi: FillChar(Var X;Count:Word;Value);

Hi ***funksiyasi.*** Vazifasi: argumentning katta baytini aniqlaydi;

Aniqlanishi: Hi(X):Byte;

Lo ***funksiyasi.*** Vazifasi: argumentning kichik baytini aniqlaydi;

Aniqlanishi: Lo(X):Byte;

Move ***protsedurasi.*** Vazifasi: ko'rsatilgan miqdordagi baytlardan tezkor (operativ) xotiraning bir qismidan ikkinchi qismiga nusxa ko'chiradi;

Aniqlanishi: Move(Var Cource,Dest;Count:Word);

Random ***funksiyasi.*** Vazifasi: tasodifiy sonni aniqlaydi;

Aniqlanishi: Random([Range:Word]);

bu yerda aniqlanadigan tasodifiy son 0 va Range sonlari oralig'ida yotadi.

Randomize ***protsedurasi.*** Vazifasi: tasodifiy sonni hosil qiluvchi generatorni ishga tushiradi;

Aniqlanishi: Randomize;

UpCase ***funksiyasi.*** Vazifasi: kichik lotin harflarni katta harflarga o'tkazadi;

Aniqlanishi: UpCase(Ch:Char):Char;

funksiya rus alifbosi harflarini ham katta harflarga o'tkazadi.

3.16.3. Crt modulining protsedura va funksiyalari

Yuqorida qisqacha aytib o'tganimizdek, Crt moduli matnli ekran, klaviatura va tovush dinamigi bilan ishlashni tashkil etuvchi dasturlar bilan jihozlangan.

Matnli rejimdagi ekran odatda 25 ta qator va har bir qatorda 80 ta belgi sig'dirish imkoniyatiga ega. Ekranda hosil qilish mumkin bo'lgan ranglar soni esa 16 ta (oq-qora rangli ekran rangini ko'zda tutmayapmiz).

Ekranda kerakli belgini zarur joyda chiqarish, matn va fon rangini o'zgartirish hamda ekranning ishini to'liq boshqara olish Crt modulining vazifalaridan hisoblanadi:

- **GotoXY(I,J:Byte)** *protsedurasi* ekranning I ustun va J satriga kursorni ko'chirib beradi;
- **Write(S)** *protsedurasi* kursor turgan joydan S qatorni chop etadi;
- **TextColor(Color:Byte)** *protsedurasi* ekranga chiqariladigan matn rangini o'rnatadi;
- **TextBackGround(Color:Byte)** *protsedurasi* chop etiluvchi matn uchun ekran fonining rangini o'rnatadi;
- **ClrScr** *protsedurasi* TextBackGround protsedurasi o'rnatgan rang bilan ekranni tozalaydi, agar rang o'rnatilmagan bo'lsa ekran qora rang bilan tozalanadi, kursor ekranning chap yuqori qismiga chiqariladi;
- **AssignCrt(Var F:Text)** *protsedurasi* Crt modulini o'rnatish uchun matnli faylni ta'minlaydi;
- **ClrEol** *protsedurasi* kursor turgan joydan boshlab ekranni tozalaydi. Agar Window protsedurasi bilan oyna ochilgan bo'lsa tozalov shu oyna ichida bo'ladi. Kursor o'z o'rnini o'zgartirmaydi;
- **Delay(N:Word)** *protsedurasi* N millisekund davomida to'xtashni tashkil etib beradi;
- **DelLine** *protsedurasi* kursor turgan qatorni o'chiradi. Qolgan qatorlar bir satrga yuqoriga siljiydi va ekranning oxirgi qatori tozalanadi;
- **HighVideo** *protsedurasi* chop etilayotgan belgilar uchun yuqori yorug'lik o'rnatadi;
- *InsLine protsedurasi* kursor turgan joyga bo'sh qator o'rnatadi;
- **KeyPressed:Boolean** *funksiyasi* agar klavish bosilsa True, klavish bosilmagan bo'lsa False qiymatni qaytaradi;
- **LowVideo** *protsedurasi* chop etilayotgan belgilar uchun past yorug'lik o'rnatadi;
- **NormVideo** *protsedurasi* chop etilayotgan belgilar uchun o'rtacha yorug'lik o'rnatadi;
- **NoSound** *protsedurasi* tovush dinamigini o'chiradi;
- **ReadKey:Char** *funksiyasi* klaviaturadan belgini o'qiydi;

- **Sound(X:Word)** *protsedurasi* tovush dinamigini ulaydi va uni berilgan X chastotada tovush chiqarishga majbur qiladi;
- **TextMode(N:Integer)** *protsedurasi* kerakli N—matnli rejimni o'rnatadi;
- **WhereX:Byte** *funksiyasi* kursor turgan joyning ustun raqamini (qaralayotgan oynada), ya'ni X koordinatasini aniqlaydi;
- **WhereY:Byte** *funksiyasi* qaralayotgan oynaga nisbatan kursor joyining qator raqamini, ya'ni Y koordinatasini aniqlaydi;
- **Window(X1,Y1,X2,Y2:Byte)** *protsedurasi* ekranda matnli oyna hosil qiladi. Bu oyna o'zini xuddi to'liq ekrandek his etadi.

Ranglar bilan ishlashni qulaylashtirish maqsadida Crt modulida quyidagi o'zgarmaslar kiritilgan:

```
Const
Black=0; {qora}
Blue=1; {ko'k}
Green=2; {yashil}
Cyan=3; {havorang}
Red=4; {qizil}
Magenta=5;{malinarang}
Brown=6;
LightGray=7;{och kulrang}
DarkGray=8;{to'q kulrang}
LightBlue=9; {och havorang}
LightGreen=10; {och zangori}
LightCyan=11; {feruzarang}
LightRed=12; {och qizil}
LigthMagenta=13;{och malinarang}
Yellow=14; {sariq}
White=15;{oq}
Blink=128; {yonib-o'chish}
```

3.16.4 MSDOS modulining protsedura va funksiyalari

DOSning tizimli moduli 6 kilobayt atrofidagi joy egallab, MSDOS muhiti bilan ishlashga hamda uning imko-

niyatlaridan foydalanishga mo'ljallangan protseduralar va funksiyalarga ega. Aslida bu protseduralar va funksiyalar Paskal tilining sintaktik qoidalariga moslashtirilgan MSDOSni chaqirish funksiyalaridir.

DOS modulidagi protsedura va funksiyalarni bajaradigan ishlarining ma'nolariga ko'ra oltita funksional guruhga ajratish mumkin:

- MSDOS parametrlarining so'rovi va ularni o'rnatish;
- SHEHMda taqvim (kalendar) va soatlar bilan ishlash;
- Disk resurslarining tahlili (analizi);
- Katalog va fayllar bilan ishlash;
- MSDOS uzilishlari(prerivaniye) bilan ishlash;
- Rezident dasturlar va subprotsesslar bilan ishlash.

1. MSDOS parametrlarining so'rovi va ularni o'rnatishning protseduralari hamda funksiyalari:

- **DOSVersion:Word** *funksiyasi* MSDOS versiyasining kodlashtirilgan raqamini aniqlaydi;
- **GetCbreak (Var B: Boolean)** *protsedurasi* Break parametrining qiymatini o'qiydi;
- **SetCbreak(Var B: Boolean)** *protsedurasi* Break qiymatini o'rnatadi;
- **GetVerify(Var V:Boolean)** *protsedurasi* Verify parametrining qiymatini o'qiydi;
- **SetVerify(V:Boolean)** *protsedurasi* Verify qiymatini o'rnatadi;
- **EnvCount:Integer** *funksiyasi* MSDOSning tizimli o'zgaruvchilari sonini aniqlaydi;
- **EnvString(N:Integer):String** *funksiyasi* N raqamli MSDOS o'zgaruvchisining to'liq vazifa qatorini beradi;
- **GetEnv(E:String):String** *funksiyasi* E tizimli o'zgaruvchining qiymatini aniqlaydi.

2. SHEHMda taqvim va soatlar bilan ishlash protsedura hamda funksiyalari:

DOS modulida kompyuter soati va taqvim bilan ishlash hamda fayllarni tashkil qilish kuni va vaqtini o'rnatish protseduralari mavjud:

- **GetDate(Var Year,Month,Day,Dw:Word)** SHEHM soatidan yil, oy, kun va hafta kunini o'qiydi;
- **SetDate(Year,Month,Day:Word)** SHEHMning soatiga yil, oy va kunni o'rnatadi;
- **GetTime(Var Hour,Min,Sec,Sec100:Word)** SHEHM soatidan hozirgi vaqtni aniqlaydi;
- **SetTime(Var Hour,Min,Sec,Sec100:Word)** SHEHM soatiga yangi vaqtni o'rnatadi;
- **PackTime(Var Dt:Datetime; Var T:Longint)** kun,oy, yil va vaqt ma'lumotlarini faylga yozish uchun ixcham holda tayyorlaydi.
- **UnPackTime(Var T:Longint;Var Dt:Datetime)** fayldan o'qilgan kun, oy, yil va vaqt yozuvini ochib beradi;
- **GetFtime(Var F; Var T:Longint)** ochiq F fayl uchun kun, oy, yil va vaqtning ixcham yozuvini o'qib beradi;
- **SetFtime(Var F; Var T:Longint)** ochiq F fayl uchun yil, oy, kun va vaqtning ixcham yozuvini yozib qo'yadi.

3. Disk resurslari tahlilining funksiyalari:

DOS moduli o'z ichiga disk holatini tahlil qilishning ikkita funksiyasini oladi:

- **DiskFree(D:Word):Longint** diskdagi bo'sh joy o'lchovini aniqlaydi;
- **DiskSize(D:Word):Longint** diskning to'liq hajmini baytlarda aniqlab beradi. Bu yerda D butun qiymatli parametr yoki butun son bo'lib, uning qiymati aniq bir diskni ko'rsatadi:
- Agar D=0 bo'lsa ishlatilayotgan joriy disk, D=1 bo'lsa A disk, D=2 bo'lsa B disk va hokazolar tahlil qilinadi;
- Agar mazkur tizim D ning kiritilgan qiymatiga mos diskni topa olmasa funksiya — 1 ga teng qiymat beradi;

4. Kataloglar hamda fayllar bilan ishlash funksiyalari va protseduralari.

Paskal tilining tashqi fayllar bilan ishlash imkoniyatlari juda cheklangan bo'lib, ular asosan quyidagilardan tashkil topgan: faylni ochish, yopish, qayta nom berish va o'chirish. Bu kamchiliklarni bartaraf qilish uchun DOS

modulida quyidagi funksiyalar va protseduralar ko'zda tutilgan:

- **FindFirst(Path:String;Attr:Word;Var Sr:SearchRec)** *protsedurasi* Path so'rovi bo'yicha Attr atributli birinchi mos ismni topadi;
- **FindNext(Var Sr:SearchRec)** *protsedurasi* FindFirst protsedurasidan keyin chaqirilib, keyingi mos ismlarni topish uchun ishlatiladi;
- **FSearch(Path:PathStr; DirList:String):PathStr** *funksiyasi* DirList kataloglar ro'yxatidan Path ismli faylni qidiradi;
- **GetFattr(Var F:File; var Fa:Word)** *protsedurasi* F fayl bilan bog'liq diskdagi faylning Fa atributini o'qiydi;
- **SetFattr(Var F:File; var Fa:Word)** *protsedurasi* F fayl bilan bog'liq diskdagi faylga atribut o'rnatadi;
- **Fsplit(Path:PathStr; Var Dir:DirStr; Var Name:NameStr; Var Ext:ExtStr)** *protsedurasi* Path faylining to'liq ismini uni tashkil etuvchilariga ajratadi: Dir—yo'li, Name—ismi, Ext—kengaytmasi.

5. MSDOS uzilishlari bilan ishlash.

MSDOS ning tizimli funksiyalarini chaqirish uzilishlar ko'rinishida hal qilinadi. Har bir uzilish aktivlashtirilgandan keyin turli xil funksiyalar to'plamiga murojaat qilish imkoniyatini yaratadi. Masalan, 16H raqamli uzilish operatsion tizim darajasidagi klaviatura bilan muloqotni tashkil etuvchi funksiyalarga yo'l ochib beradi, 25H raqamli uzilish esa diskni o'qish ishlarini boshqarishni o'z zimmasiga oladi va hokazo. Juda katta vazifalar 21H uzilish zimmasidadir, chunki u operatsion tizim (OT)ni tashkil etuvchi o'nlab funksiyalardan foydalanishni tashkil qilib beradi.

Quyida e'tiboringizga Paskal dasturidan turib MSDOSning funksiyalariga murojaat qilishni tashkil qilib beruvchi DOS moduli protseduralarining vazifalari va aniqlanishlarini havola qilamiz:

- **GetIntVec(N:byte; Var Adress:pointer)** *protsedurasi* Adress ko'rsatkichiga berilgan N raqamli uzilish qism dasturining adresini aniqlab beradi;

- **SetIntVec(N:byte; Var Adress:pointer)** *protsedurasi* N-raqamli uzilishning yangi qism dasturini DOS ga o'rnatib, adresning eski qiymatini Adress ko'rstkichiga joylaydi;
- **Intr(N:byte; Var R:Register) protsedurasi** N dasturli uzilishni aktivlashtirib, funksiya raqami va parametrlarini R o'zgaruvchiga uzatadi;
- **MSDOS(Var R:Register)** *protsedurasi* 21H raqamli uzilishni maxsus chaqirish vazifasini bajaradi.

Oxirgi protsedurada yangi Register tipi kiritildi. Bu tip DOS modulida aniqlangan quyidagi yozuvdan iborat:

```
Type
  register=Record
     case Integer of

   0:(AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags:Word);
                1:(AL,AH,BL,BH,CL,CH,DL,DH:Byte);
        end;
```

Bu tipdagi o'zgaruvchilar mikroprotsessor registrlarga Intr va MSDOS larni chaqirishda yo'l ochish uchun ishlatiladi:

0—variantida 16 razryadli registrlarga 1—variantda esa 8 razryadli protsessor yacheykalariga murojaat qilinadi.

6. Rezident dasturlar va subjarayon (subprotsess)larni tashkil qilish.

MSDOS operatsion tizimi subjarayonlarni tashkil qilish bo'yicha katta imkoniyatlarga ega. Subjarayonda bir dastur boshqa bir dasturni ishga tushirsa, boshqa bir dastur esa o'z navbatida keyingi dasturni ishga tushiradi va hokazo. Bu holda ishga tushiruvchi dastur (jarayon) subjarayonning ishini tashkil qilib o'zi ishdan to'xtaydi, lekin SHEHM xotirasida saqlanib turadi. Subjarayon esa xotiraning qolgan qismida ixtiyoriy dastur kabi bajariladi. U o'z ishini yakunlagach keyingi jarayon dasturi ish boshlaydi.

MSDOS ning bu xossasi integrator-dasturlarni yaratish imkoniyatini hosil qiladi. Hammaga tanish bo'lgan Norton Commander dasturi subjarayonlardan aktiv foydalanuvchi, rezident bo'lmagan dastur hisoblanadi.

Oddiy dasturlar o'z ishini yakunlagach SHEHM xotirasini bo'shatib qo'yadi. Subjarayonlar esa ishini yakunlagach, boshqarishni o'zlarini chaqirgan dasturlarga, ular esa o'z navbatida keyingisiga uzatadi va jarayon oxirida boshqarish OT (operatsion tizim)ga uzatiladi.

Rezident dasturlar esa boshqacha ishlaydi. Ular ishga tushgach darrov qandaydir amallarni bajarishi mumkin, yoki OT uzilishlari bilan turli xil tushunarsiz ishlarni amalga oshirishi mumkin. Ularning asosiy hislati shundaki, o'z ishini yakunlagach SHEHM xotirasida saqlanib qoladi va biror bir shartning bajarilishiga qarab qayta «tirilishi», ya'ni yana ishlay boshlashi mumkin. Rezident dasturlarga misol qilib quyidagi dasturlarni ko'rsatish mumkin: Side Kick tizimi, ALFA va BETA kirill drayverlari, antivirus dasturlari, virus dasturlari va hokazo.

Rezident dasturlarning barchasini bir xil narsa birlashtiradi, bu ham bo'lsa ularning ichki(tezkor) xotira (OZU)da joylashishi va maxsus shartlarda (ALFA, BETA klavishlarini bosishda, antivirus dasturlarga murojaatda, qo'yilgan vaqt yetib kelganida va h.k.) qayta «tirilishi». Passiv holatlarda rezident dasturlar faqat ortiqcha joy ushlab turishadi hamda boshqa oddiy dasturlar ishiga halaqit berishmaydi.

SHEHM xotirasidan unumli foydalangan holda subjarayonlarni tashkil qilish va rezident dasturlar bilan ishlashning maxsus protseduralaridan foydalanish mumkin:

- **SwapVectors** *protsedurasi* ishga tushirilayotgan subjarayonlar uchun tizimli yoki vaqtinchalik uzilishlar vektorlarini tiklaydi;
- **Exec (ExeFile, ComLine: String)** *protsedurasi* ComLine satrli parametr bilan ExeFile (subjarayon) bajariluvchi faylni ishga tushiradi;
- **DosExitCode: Word** *funksiyasi* subjarayon yakunlanganligi haqidagi kodni aniqlaydi;
- **Keep(ExitCode:Word)** *protsedurasi* SHEHM xotirasidan o'chmagan holda dastur ishini yakunlaydi (rezident kodini tashkil qiladi).

3.16.5. Printer moduli

Paskal tilining kutubxonasida sanab o'tilgan modullar qatorida Printer moduli ham o'zining munosib joyini egallagan, chunki olingan natijalarni printer orqali qog'ozga chop etish muhim ishdir. Bu modulning asosiy vazifasi dastur va chop etuvchi qurilma (odatda printer) o'rtasida aloqa o'rnatishdir.

Bu kichik modul bor-yo'g'i bitta o'zgaruvchi va bajariluvchi matnning ikki qatoridan tashkil topgan. Bu modul Lst o'zgaruvchisi va LPT1 (birinchi parallel port) tizimli qurilma mosligini o'rnatadi. Modulning to'liq matni quyidagicha:

```
Unit Printer;
Interface
Var
   Lst:Text;
Implementation
Begin
   Assign(Lst,'LPT1');
   Rewrite(Lst)
End.
```

Printer moduli ulangandan so'ng Write(Lst,...) va WriteLn(Lst,...) operatorlari natijaviy ma'lumotlarni to'g'ridan-to'g'ri printerga chiqaradi.

Agar printer boshqa qurilma orqali, masalan ikkinchi port LPT2 yoki ketma-ket SOM1 porti orqali ulangan bo'lsa, u holda dasturchi o'zining Printer2 nomli modulini yaratib olishi mumkin. U holda yuqoridagi modul dasturi matnidagi LPT1 yozuvi boshqa ismga (LPT2 yoki SOM1) almashtirilib translyatsiya qilib olinadi.

Natijalarni yoki maxsus buyruqli kodlarni printerga Lst fayli (yoki shunga o'xshash fayl) orqali chiqarish, chop etishni boshqarishning yagona yo'li emas. BSVV17N uzilishi orqali printerga belgilar yoki ularning ketma-ketligini uzatish mumkin. Bundan tashqari mazkur uzilishdan foydalanish chop etish qurilmasining holatini so'rash imkoniyatini yaratadi.

17N uzilishi printerga xizmat ko'rsatishning uch xil funksiyasidan foydalanishga ruxsat beradi:

— belgilarni printerda chop etish (0-raqamli funksiya);
— printer portini ishga sozlash (1-raqamli funksiya);
— printer holatini o'qish (2-raqamli funksiya).

3.16.6. Overlay modulining protsedura va funksiyalari

Juda ko'p holatlarda dastur kompilyatsiya qilingandan so'ng ma'lum bo'ladiki, dastur uchun SHEHM ning ichki tezkor xotirasi yetishmay qoladi, ya'ni dastur talab qilgan xotira joyining miqdori SHEHM ning xotirasidan katta bo'ladi. Bu holatda ekranda Not enough memory («xotira yetishmayapti») ma'nosidagi xato ko'rsatilgan ma'lumot hosil bo'ladi. Bunday holatlarda dasturchiga Overlay moduli yordamga oshiqadi

Dasturni overleyli qurish. Overlay moduli dasturni alohida bo'laklarga ajratishning kuchli vositasi hisoblanadi. Bunday bo'laklar soni ixtiyoriy sonda bo'lishi va bu bo'laklarga zarur bo'ladigan joy miqdori SHEHM imkoniyatidan ancha yuqori bo'lishi mumkin. Overleyli tarzda tuzilgan dastur bitta YEXE kengaytmali fayldan va shunday nomli lekin OVR kengaytmali yana bitta fayldan tashkil topgan bo'ladi. Bunda YEXE fayli dasturning doimiy qismini, OVR fayli esa zarur paytda xotiraga yuklanadigan kodlarni saqlab turadi. Overleyli tarzda tuzilgan dasturda xotirada faqat xozirdagina zarur bo'ladigan overleyli protsedura va funksiyalargina saqlanadi. Ular kerakli paytdagina o'zlarining joylarini boshqa protsedura va funksiyalarga bo'shatib beradilar. Bu holda qo'shimcha xotira joyi talab qilinmaydi, chunki dasturning overleyli qismlari xotiraning faqat bir qisminigina navbat bilan ishlatadi xolos. Xotiraning mazkur qismi *overleyli bufer* deb ataladi. Overleylarni xotiraga yuklash va xotiradan bo'shatish kabi ichki amallarning barchasini overley administratsiyasi avtomatik tarzda bajaradi. Dasturchidan

esa faqat dasturning overleyli parchalarini e'lon qilish va uning administratsiyasini ishga solish talab qilinadi xolos.

Overleyli dasturlarni rasmiylashtirish qoidalari. Turbo—Paskalda overleylar modullar darajasidagina yaratilishi mumkin. Overleylar xuddi oddiy modullar kabi e'lon bo'limi (INTERFACE) va hisoblash bo'limi (IMPLEMENTATION) hamda uni faollashtiradigan qismlardan tashkil topadi. Shu bilan bir qatorda quyidagi shartlarning bajarilishi ham talab qilinadi:

- xar bir overley modul uchun unga oldindan overleylilik huquqini beruvchi {$F+} kompilyatsiya rejimi o'rnatilishi lozim;
- overleyli qism-dasturlardan bevosita va bilvosita foydalanuvchi barcha protsedura va funksiyalar {$F+} kompilyatsiya qilinishi lozim.

Bu shartlarni bajarish uchun overleyli modulning bosh qismiga {$F+, $O+} direktivalarini o'rnatib, asosiy dasturning birinchi qatorlariga {$F+} kompilyatsiya rejimining kaliti yozib qo'yiladi.

Bundan tashqari, asosiy dastur overlay moduliga ulangan bo'lishi va uses direktivasidagi qaysi modullar overleyli ekanligi ko'rsatib qo'yiladi.

Misol sifatida quyidagi dasturning bosh qismini ko'rsatib o'tamiz:

```
Program MultiCalc;
{$F+}
uses crt,dos,Overlay, MultiCalc, divCalc AddCalc,
Subcalc;
{ $O MultiCalc}    { MultiCalc moduli —overley }
{ $O DivCalc}      { DivCalc moduli —overley }
{ $O AddCalc}      { AddCalc moduli —overley }
{ $O SubCalc}      { SubCalc moduli —overley }
```

Ahamiyat bergan bo'lsangiz { $O <modul ismi >} direktivasi overleyli modullarni ko'rsatish uchun yozilishga majburdir. bundan tashqari, uses direktivasining modullari ro'yxatida overlay moduli overleyli modullar ro'yxatining boshida kelishi shart.

Overleyli dasturlar xotirada kompilyatsiya qilinmaydi, endi ular faqat disklardagina hosil qilinadi. Dastur kompilyatsiyasidan so'ng (bu dasturni MULTICALC.PAS nomi bilan ataylik) MULTICALC.EXE bajaruvchi fayli va overleyli deb e'lon qilingan barcha modullardagi protseduralarning kodlarini saqlovchi yo'ldosh fayli hosil qilinadi.

3.16.6.1. Overley ishini initsializatsiya qilish

1. Overleylar administratorini yuklash

Overley administratorini yuklash dasturning ishlashi davomida faqatgina bir marta bajarilishi shart. Bu ish quyidagi overlay modulida e'lon qilingan protseduraga murojaat orqali amalga oshiriladi:

OvrInit (Ovr File Name: string)

Bu protsedura overley administratorini ishga yuklaydi va ovr overleyli faylni ochadi.

2. Overleyni yuklash natijalarining tahlili uchun Overlay modulidagi oldindan aniqlab qo'yilgan butun tipli OvrResult o'zgaruvchisi mavjud bo'lib, u mazkur modul protsedura va funksiyalarining, shu jumladan OvrInit protsedurasining ishlarini yakunlash kodini o'zida saqlaydi. OvrResult o'zgaruvchisi yetti hil qiymat qabul qilishi mumkin va bu qiymatlarning har biriga mos ravishda o'zgarmaslar biriktirib quyilgan.

Bu ma'lumotlar 7- jadvalda keltirilgan

7-jadval

O'zgarmaslar	O'zgarmasning ma'nosi
OvrOk*0	to'g'ri yakunlanishi
OvrEror *—1	Overleyni boshqarish xatosi
OvrNotFound*—2	Over fayli topilmadi
Ovr NoMemory*—3	Bufer uchun xotira yetishmayapti
OvrIOEror*—4	Overleylifaylni o'qishda buzilish ro'y bergan
OvrNoEMSDriver*—5	EMS draveyri o'rnatilmagan
OvrNoEMSMemory*—6	EMS-xotira hajmi yetarli emas

Endi sanab o'tilgan xatoliklar haqida to'liqroq to'xtalib o'taylik:

- OvrEror xatosi — overleyli bo'lmagan faylni yuklashga xarakat qilinganda ro'y beradi ;
- OvrNotFound xatosi — overleyli faylni diskka noto'g'ri joylash oqibatida yuzaga kelishi mumkin ;
- OvrNoMemory xatosi — SHEHM ning bo'sh xotirasi yetishmasligidan ro'y berishi mumkin ;
- OvrIOEror xatosi — kelib chiqishiga ko'proq tashqi omillar sababchi bo'ladi (faylning shikastlanishi, MS — DOS ichidagi buzilishlar) ;
- OvrEMSDriver va OvrNoEMSMemory xatoliklari — OvrInit protsedurasining emas, balki OvrInitEMS qo'shimcha protsedurasining ishidagi xatoliklaridan kelib chiqadi.

Overley buferini boshqarish. Overley buferining o'lchovlarini boshqarish va uni tozalash uchun Overlay modulida maxsus protsedura va funksiyalar kiritilgan:

OvrGetBuf *funksiyasi*. Vazifasi : ishchi buferning baytlardagi o'lchovlarini aniqlaydi ;

Aniqlanishi: OvrGetBuf :Longint ;

bu yerda funksiyaning qiymati ba'zi hollarda 64 kb dan oshishi mumkin, shuning uchun, funksiya qiymatini longint deb e'lon qilinadi.

OvrSetBuf *protsedurasi*. Vazifasi: bufer o'lchovini moslashtirib turadi;

Aniqlanishi: OvrSetBuf (Size: longint); bu protsedura OvrInit va OvrInitEMS protseduralari yordamida overleylar administratsiyasi yuklangandan so'ng ishga tushishi lozim. Buferning talab qilinayotgan baytlardagi miqdori Size parametri orqali beriladi.

OvrClearBuf *protsedurasi*. Vazifasi: overley buferini majburiy tarzda tozalash uchun ishlatiladi.

Aniqlanishi: OvrClearBuf ;

bu protsedura buferda turgan barcha overleyli dasturlarni chiqarib tashlash vazifasini bajaradi. Overleylar administratorining o'zi bu protsedurani chaqirmaydi.

3.16.7. Graph modulining protsedura va funksiyalari

Barcha dasturlash tillari kabi Paskal tili ham dasturchiga faqat matnli axborotlar bilangina emas, balki grafik ma'lumotlar bilan ham ishlash imkoniyatini beradi. Graph moduli turli xil displey adapterlari (CGA,EGA,VGA,MCGA,Hercules,PC3270,AT&T6300 va IBM8514)ning grafik rejimlarini to'liq boshqarish imkoniyatini yaratuvchi saksondan ortiq protsedura va funksiyalarning kutubxonasini ifoda etadi.

Grafika rejimida ekran holati. Ma'lumki, ekran to'rtburchak maydon bo'lib, juda ko'p nuqta (Pixel) lardan tashkil topgan. Grafik ekran ishchi maydon va bordyur (ekranning chetki qismi)dan iborat bo'ladi.

Grafik rejimda ekrandagi barcha nuqtalarning ranglarini o'zgartirib chiqish mumkin (eslatib o'tamiz, matnli rejimda bu mumkin emas). Turli rangga bo'yalgan nuqtalar yordamida chiziqlar, matnlar va boshqa har xil tasvirlar hosil qilinadi. Ekran turiga qarab ranglar soni turli hil bo'lishi mumkin, eng kami bilan esa 2 xil rang bo'ladi.

Kompyuter ekrani yo matnli rejim yoxud grafik rejim holatida bo'ladi. Bir vaqtning o'zida ekranning bir qismi grafik rejimda, bir qismi matnli rejimda bo'lishi mumkin emas. Chunki ekranda ko'rinayotgan barcha tasvirlar video xotiradagi ma'lumotlarning aksi — tasviridir.

Ekranning grafik yoki matnli rejimlariga qarab videoxotiradagi ma'lumotlar turlicha bo'ladi.

Videoxotira, yorug'lik trubkasining kontrolleri, kirish-chiqish portlari va shu kabilar bitta platada joylashadi va *displey adapteri* deb ataladi. Amalda bir necha xil displey adapterlari mavjud bo'lib, ular quyidagi ko'rsatkichlari bilan bir-biridan farq qiladi:

— ekranning tasvirni ko'rsatish sifati;

— ekranda bir vaqtda ko'rsatish mumkin bo'lgan ranglar soni.

Quyida amalda keng tarqalgan videoadapterlar sanab o'tilgan:

- CGA(Color Graphics Adapter) ;
- MCGA(Multi Color Graphics Array);
- EGA(Enhanced Graphic Adapter);
- VGA(Video Graphics Array).

Kompyuterda qanday adapterning o'rnatilishidan qat'i nazar Turbo—Paskalning protsedura va funksiyalaridan to'liq foydalanish mumkin. Ularning o'zaro sozlashuvi avtomatik tarzda kechadi. Bu sozlashlarni *grafik drayverlar* deb ataluvchi maxsus dasturlar bajaradi. Drayverlar *. BGI kengaytmasi bor bo'lgan fayllarda joylashgan.

Misol uchun, EGA va VGA adapterlari bilan ishlash uchun zarur drayverlar EGAVGA.BGI faylida,CGA va MCGA adapterlariga mos drayverlar esa CGA.BGI faylida joylashgan.

Turli xil drayverlarni ko'rsatish uchun Graph modulida quyidagi o'zgarmaslar aniqlangan:

```
const
Detect=0;{Drayverni avtomatik tarzda aniqlash}
CGA=1;
MCGA=2;
EGA=3;
EGA64=4;
EGAMONO=5;
IBM8514=6;
HERCMONO=7;
ATT400=8;
VGA=9;
PC327=10.
```

Graph modulida grafik rejimlarni ko'rsatish uchun esa quyidagi o'zgarmaslar aniqlangan (CGA,EGA va VGA adapterlari uchun);

```
const
CGAC0=0;   {320x200 ta nuqta, 4 xil rang}
CGAC1=1;   {320x200 ta nuqta, 4 xil rang}
CGAC2=2;   {320x200 ta nuqta, 4 xil rang}
CGAC3=3;   {320x200 ta nuqta, 4 xil rang}
CGAHi=4;   {640x200 ta nuqta, 4 xil rang}
EGALo=0;   {640x200 ta nuqta, 16 xil rang, 4 ta varaq}
```

EGAHi=1; {640x350 ta nuqta, 16 xil rang, 2 ta varaq}
EGA64Lo=0; {640x200 ta nuqta, 16 xil rang,1 ta varaq}
EGA64Hi=1; {640x350 ta nuqta, 4 xil rang, 4 ta varaq}
VGALo=0; {640x200 ta nuqta, 16 xil rang, 4 ta varaq}
VGAMed=1; {640x350 ta nuqta, 16 xil rang, 2 ta varaq}
VGAHi=2. {640x480 ta nuqta, 16 xil rang, 1 ta varaq}

Displeyni grafik rejimga o'tkazish. Displeyning doimiy rejimi — matnli rejim hisoblanadi. Displeyni grafik rejimiga o'tkazish uchun Graph modulining InitGraph protsedurasidan foydalaniladi:

InitGraph(GD,GM,Path)

bu yerda GD — drayver raqami ;
GM — rejim raqami ;
Path — kerakli drayver joylashgan fayl yo'li.

GD va *GM* raqamlarni qanday aniqlashni oldingi mavzuda ko'rib chiqdik. Agar ***Path*** o'zgaruvchisi bo'sh satr qatorini tashkil qilsa (Path=' ') drayverni ishchi katalogdan qidiriladi.

GD va *GM* o'zgaruvchi parametrlar hisoblanadi. Agar *InitGraph* protsedurasini ishga tushirishda *GD*=0 bo'lsa, zarur drayver va bu drayver uchun eng yaxshi rejim avtomatik tarzda aniqlanadi.

Graph modulida qulaylik uchun *Detect* o'zgarmasi (Detect=0) kiritilgan. Misollarda undan qanday foydalanganligiga ahamiyat bering.

InitGraph protsedurasiga simmetrik protsedura CloseGraph hisoblanadi. Bu protsedura drayverni xotiradan chiqarib tashlaydi va oldingi videorejimni tiklaydi.

Quyidagi dastur displeyni grafik rejimga o'tkazadi va shu zahotiyoq uni asli holiga qaytaradi:

```
uses Graph;
var
    Gd,Gm:Integer;
begin
GD:=detect;{Drayverni avtomatik tarzda aniqlaydi,
            chunki Detect=0}
```

```
    InitGraph(GD,GM,'d:g'tp'); {Grafik rejimni o'rnatish}
    ReadLn; {Enter bosqichining bosilishini kutish}
    CloseGraph;
end.
```

Ba'zi hollarda, masalan .BGI kengaytmali fayl yo'lini xato ko'rsatilsa yoki grafik rejimni o'rnatish uchun tezkor xotira yetishmasa, InitGraph protsedurasi o'z ishini muvaffaqiyatli yakunlay olmaydi. Bunday hollarda yo'l qo'yilgan xatoliklarni aniqlash uchun GraphResult funksiyasidan foydalanish mumkin. Bu funksiya grafik rejimni ishga tushirishga harakat qilganda quyidagi qiymatlardan birini beradi:

- 0 — xatolar yo'q;
- 2 — grafik plata o'rnatilmagan;
- 3 — drayver fayli topilmadi;
- 4 — noto'g'ri drayver o'rnatilgan;
- 5 — grafik rejimni o'rnatish uchun tezkor xotira yetishmagan;

GraphResult funksiyasidan foydalanishga doir quyidagi dasturni ko'rib chiqaylik:

```
uses Graph;
var
      GD,GM,ErrCode:integer;
begin
   GD:=Detect;     {Drayverni avtomatik tarzda aniqlash}
   InitGraph(GD,GM,'      '); {Grafik rejimni ishga tushirish}
   ErrCode:=GraphResult;
   if ErrCode<>0 then WriteLn(ErrCode,'raqamli xato') {Xatoga yo'l qo'yilgan}
   else CloseGraph;{Xato topilmagan va grafik rejimning ishi yakunlandi}
end.
```

Nuqta, chiziq va ranglar. Biz ekranda grafik rejimini qanday o'rnatishni ko'rib chiqdik. Endi ekranda turli xil nuqtalar, chiziqlar va shakllar chizishni tashkil

qilaylik. Graph modulida 80 taga yaqin funksiya va protseduralar mavjud bo'lib, ulardan foydalanib quyidagi ishlarni qilish mumkin:

- nuqtalar yasash;
- kesmalar chizish;
- ellips va aylanalar chizish;
- to'g'ri to'rtburchaklar va ko'pburchaklar chizish;
- yopiq shakllarni turli ranglarga bo'yash hamda bir necha xil standart va keragicha nostandart usullar bilan sohalarni shtrixlash;
- ekranga turli shriftdagi matnlarni chiqarish;
- ekran sohalarini eslab qolish va ularni surish.

Grafik rejimda ishlash xuddi matematika fanidagi Dekart koordinatalar tizimida nuqtalar orqali turli xil shakllar yasashga o'xshab ketadi. Ekrandagi har bir nuqta o'zining koordinatalariga ega. Ekranning chapdan eng tepadagi nuqtasining koordinatasi (0,0)ga teng. *X* koordinatasi chapdan o'ngga o'sib borsa, *Y* koordinatasi yuqoridan pastga qarab o'sib boradi. Nuqta koordinatasini aniqlovchi (*x,y*) juftlikdagi birinchi qiymat OX o'qidan, ikkinchi qiymat esa *OY* o'qidan aniqlanadi. Ekranning o'ngdan eng pasidagi nuqtasi oxirgi nuqta hisoblanadi va uning koordinatasi grafik rejimning turiga bog'liq. Masalan, VGAHi rejimida ekrandagi nuqtalar soni 640x480ga teng. O'ngdan eng pastki nuqta koordinatasi esa (639,479) ga teng bo'ladi.

Endi *Graph* modulining eng ko'p ishlatiladigan funksiya va protseduralari bilan tanishib chiqaylik.

1. PutPixel(x,y,color) *protsedurasi* — (*x,y*) koordinatali nuqtani *Color* parametri bilan muayyan rangga bo'yab beradi;

Misol: PutPixel(100,120,Red) — ekranda (100,120) koordinatali qizil nuqta paydo bo'ladi.

2. *GetPixel*(x,y) *funksiyasi* — (x,y) koordinatali nuqtaning rangini aniqlab beradi;

Misol: Color:=GetPixel(100,120); (100,120) koordinatadagi nuqtaga qo'yilgan rangni aniqlaydi.

3. *Line*(x1,y1,x2,y2) *protsedurasi* —(*x*1,*y*1) va (*x*2,*y*2) koordinatali nuqtalarni tutashtiruvchi kesma chizadi;

4. ***Circle*****(x,y,Radius)** *protsedurasi* — markazi (*x,y*) nuqtada joylashgan radiusi Radius ga teng aylana chizadi;

5. ***Rectangle*****(x1,y1,x2,y2)** *protsedurasi*—chapdan yuqoridagi burchagi (*x*1,*y*1) va o'ngdan pastki burchagi (*x*2,*y*2) koordinatali nuqtalar orqali o'tkazilgan to'g'ri to'rtburchak chizadi;

6. ***SetColor*****(Color)** *protsedurasi* — chizish rangini o'rnatadi;

Grafik rejimda ranglarni belgilash uchun xuddi matnli rejimdagi kabi quyidagi o'zgarmaslar ishlatiladi:

```
const
Black=0;{Qora}
Blue=1;{Ko'k}
Green=2;{Yashil}
Cyan=3;{Zangori}
Red=4;{Qizil}
Magenta=5;{Pushti}
Brown=6;{Jigarrang}
LightGray=7;{Och kulrang}
DarkGray=8;{To'q kulrang}
LightBlue=9;{Och havorang}
LightGreen=10;{Och ko'k}
LightCyan=11;{Feruzarang}
LightRed=12;{Och qizil}
LigthMagenta=13;{Och pushti}
Yellow=14;{Sariq}
White=15;{Oq}
```

Misol:

```
uses Graph;
var
   GD,GM:integer;
   Rang,Radius:word;
begin
   GD:=Detect;          {Drayverni avtomatik tarz-
da aniqlash}
   InitGraph(GD,GM,'    '); {Grafik rejimni ishga
tushirish}
   for Rang:=15 downto 0 do
   begin
```

```
        setcolor(Rang); {Chizish rangini o'rnatish}
        Radius:=Rang*10;
          Circle(GetMaxX div 2, GetMaxY div
2,Radius);
      {Markazi ekran markazida joylashgan aylana chi-
zish}
    end;
    Readln;
    CloseGraph;
  end.
```

Turli xil shakllar. Yuqoridagi mavzuda ko'rib chiqilgan funksiya va protseduralar yordamida faqat chiziqlar chizish mumkin. Endi boshqa turli xil ranglar bilan to'ldirilgan shakllar chizishni tashkil etishga yordam beruvchi yana bir nechta protsedura va funksiyalar bilan tanishib chiqamiz.

1. SetFillStyle(Style,Color) protsedurasi — Color rangi bilan sohalarni to'ldirish va ularni ko'rsatilgan uslubda to'ldirish (shtrixlash) uchun ishlatiladi;

Sohani turli ranglar bilan to'ldirish o'zgarmaslari:

```
const
EmptyFill=0;{sohani ekran fonining rangiga bo'yaydi}
SolidFill=1;{sohani belgilangan rangda uzluksiz to'ldi-
rish}
LineFill=2;{sohani qalin gorizontal (———————) chi-
ziqlar bilan to'ldiradi}
LtSlashFill=3;{sohani ingichka»/// « belgilari bilan
to'ldiradi}
SlashFill=4;{sohani qalin «/ / / « belgilari bilan to'ldi-
radi}
BkSlashFill=5;{sohani qalin «\\\» belgilari bilan
to'ldiradi}
LtBkSlashFill=6;{ sohani «\\\» qatlami bilan to'ldi-
radi}
HatChFill=7;{sohani to'r bilan to'ldiradi}
XHatChFill=8;{ sohani egri to'r bilan to'ldiradi}
InterLeaveFill=9;{ sohani zich egri shtrixlar bilan to'ldi-
radi}
```

WideDotFill=10;{sohani kam uchrovchi nuqtalar bilan to'ldiradi}

CloseDotFill=11;{sohani zich nuqtalar bilan to'ldiradi}

UserFill=12;{sohani dasturchi belgilagan shtrixlar bilan to'ldiradi}

2. Bar($x1,y1,x2,y2$) protsedurasi ekrandagi rang va shtrixlar ustiga to'g'ri to'rt burchak quradi;

3. Bar3D($x1,y1,x2,y2$,Depth,Top) protsedurasi ham shunday rang va shtrixlar bilan to'ldirilgan parallelepiped chizadi. Depth o'zgaruvchisi parallelepiped balandligini anglatadi. Top mantiqiy o'zgaruvchisi true qiymatli bo'lsa parallelipipedning yuqori asosi chiziladi, aks holda chizilmay ochiq qoladi;

4. FillEllipse(x,y,XRadius,YRadius) protsedurasi oldin o'rnatilgan rangga to'ldirilgan ellips chizadi. Ellips o'qlari koordinata o'qlariga paralel deb olinadi. XRadius — ellips eni, YRadius — ellips balandligi.

Grafik rejimdagi matnlar. Grafik rejimda matnlarni yozish uchun ikki xil tipdagi shriftdan foydalanish mumkin: nuqtalar matritsasi hamda simvolni tashkil etuvchi vektorlar qatori orqali.

Shriftlar fayllari .CHR kengaytmasiga ega bo'ladi va shriftni ishga sozlaganda kerakli fayllar ishchi katalogda yoki .BGI grafik drayveri joylashgan katalogda bo'lishi kerak.

Shriftni tanlash va masshtab o'rnatish SetTextStyle protsedurasi yordamida amalga oshiriladi:

SetTextStyle(Font, Direction, Size)— kerakli shriftni o'rnatadi, matnni chiqarish yo'nalishini aniqlaydi va belgilar o'lchovini belgilab beradi. Font— shriftni aniqlovchi o'zgaruvchi, Direction—matnni chop etish yo'nalishini ko'rsatuvchi o'zgaruvchi (chapdan o'ngga yoki pastdan yuqoriga), Size—shrift o'lchovini aniqlovchi o'zgaruvchi. Matritsali shriftda o'lchovga Size=1, vektor shriftida esa Size=4 qiymatlarida erishiladi.

Turli xil shriftlarni ko'rsatish va matnlarni chop etish yo'nalishlarini tanlash uchun quyidagi o'zgarmaslar aniqlangan:

```
const
  {shriftlar}
DefaultFont=0;{8x8 nuqtali standart matritsali shrift}
TriplexFont=1;{vektorli shrift}
SmallFont=2;{vektorli shrift}
SansSerifFont=3;{vektorli shrift}
GothicFont=4;{vektorli shrift}
{matn yo'nalishi}
HorizDir=0;{chapdan o'ngga}
VertDir=1;{pastdan yuqoriga}
```

OutTextXY(*x*,*y*,TextString) *protsedurasi* — oldindan aniqlangan shriftda, yo'nalishda va belgi o'lchovida TextString qatorini (*x*,*u*) nuqtadan boshlab chop etadi.

SetTextJustify(Horiz,Vert) *protsedurasi* — OutTextXY protsedurasi chop etadigan matnni avtomatik tarzda tekislab beradi. Horiz — gorizontal, Vert — vertikal tekislashlar.

Matnlarni tekislash uchun quyidagi o'zgarmaslar aniqlangan:

```
const {gorizontal tekislash uchun}
LeftText=0;{chap tomonga nisbatan tekislash}
CenterText=1;{markazga nisbatan tekislash }
RightText=2;{o'ng tomonga nisbatan tekislash}
{vertikal tekislash uchun}
BottomText=0;{pastki tomonga nisbatan tekislash}
CenterText=1;{markazga nisbatan tekislash}
TopText=2;{yuqori tomonga nisbatan tekislash}
```

Ekran sohalari. GetImage, PutImage protseduralari va ImageSize funksiyasi yordamida tasvirlarning to'g'ri to'rtburchakli sohalarini hotirada eslab qolish va ularni ekranga chiqarishi mumkin.

1. ImageSize(*x*1,*y*1,*x*2,*y*2) *funksiyasi* — ekranning to'g'ri to'rtburchakli sohasini saqlash uchun zarur bo'lgan xotira o'lchovini (baytlarda) beradi.(*x*1,*y*1) to'g'ri to'rtburchakli ko'rinishning chapdan yuqoridagi, (*x*2,*y*2) — esa pastdan o'ngdagi burchak nuqtalari uchun koordinatalar.

2. GetImage(*x1,y1,x2,y2*,Area) *protsedurasi* — xotiraning *Area* sohasida to'g'ri to'rtburchakli ekran tasvirini saqlaydi. (*x*1,*y*1) va (*x*2,*y*2) lar yuqoridagi ma'noda qayta ishlatilmoqda.

3. ***PutImage*(*x,y*,Area,Mode)** *protsedurasi* — ekranning ko'rsatilgan joyiga tasvir ko'rinishini chop etadi. (*x,y*) — xotiraning Area sohasidagi tasvir ko'rinishi nusxasini chop etiladigan ekranning chapdan yuqoridagi nuqtasining koordinatasi. *Mode*—tasvirni ekranga chiqarish rejimi.

Tasvirlarni ekranga chiqarish rejimini aniqlash uchun foydalaniladigan o'zgarmaslar:

```
const {PutImage protsedurasi uchun o'zgarmaslar}
NormalPut=0;{ mavjud tasvirni almashtirish}
XorPut=1;{XOR mantiqiy amali}
OrPut=2;{OR mantiqiy amali}
AndPut=3;{AND mantiqiy ko'paytirish amali}
NotPut=4;{NOT mantiqiy rad etish amali}
```

Xatolar tahlili. Grafik rejimni o'rnatish mavzusida yo'l qo'yilgan xatolar diagnostikasi uchun *GraphResult* funksiyasidan foydalangan edik. Hozir shu funksiya beradigan xatolar kodi bilan to'liqroq tanishib chiqaylik.

Quyida *GraphResult* funksiyasi beradigan kodlarga mos o'zgarmaslar ro'yhati keltirilgan:

```
const
grOk=0;{xatolar yo'q}
grNoInitGraph=—1;{Grafik rejim o'rnatilmagan
(InitGraph protsedurasini ishga tushiring )}
grNotDetected=—2; {Grafik plata o'rnatilmagan}
```

—6 — sohalarni ko'chirishda xotira chegarasidan chiqish;

—7 — zarur sohani bo'yash paytida xotira chegarasidan chiqish;

—8 — shrift fayli topilmagan;

—9 — shrift faylini ishga tushirish uchun xotira yetishmayapti;

—10 — tanlangan drayver uchun noto'g'ri grafik rejim.

Graph modulining protsedura va funksiyalari.

1. Arc *protsedurasi* — aylana yoyini chizadi.

Aniqlanishi : Arc(*x,y* : integer; StAng, EndAng, Radius:Word);

x,y — aylana markazining koordinatasi;

StAng, EndAng — mos ravishda yoyning boshlang'ich va oxirgi burchaklari;

Radius — aylana radiusi.

2. Bar *protsedurasi* — rangga bo'yalgan to'g'ri to'rtburchak chizadi.

Aniqlanishi: Bar(*x*1,*y*1,*x*2,*y*2:integer);

(*x*1,*y*1) va (x2,y2) mos ravishda to'g'ri to'rtburchakning chetki nuqtalari koordinatalari.

3. Bar3D *protsedurasi* — rangga bo'yalgan parallelipiped chizadi.

Aniqlanishi:Bar3D(*x*1 ,*y*1,*x*2,*y*2:integer;Depth:word; Top:boolean);

(*x*1,*y*1) va (*x*2,*y*2) asosni tashkil etuvchi to'g'ri to'rtburchak uchlarining koordinatalari;

Depth — parallelipiped chuqurligi;

Top — mantiqiy o'zgaruvchi.

4. Circle *protsedurasi* — aylana chizadi;

Aniqlanishi : Circle(*x,y*:integer;Radius:word);

(*x,y*) aylana markazining koordinatasi;

Radius—aylana radiusi.

5. CloseGraph *protsedurasi* — grafik rejimini uzadi.

Aniqlanishi :Closegraph;(parametrsiz protsedura)

6. DrawPoly *protsedurasi* — ko'pburchak chizadi.

Aniqlanishi :DrawPoly(NumPoints:word; var PolyPoints);

NumPoints — ko'pburchak tomonlari soni;

PolyPoints— ko'pburchak uchlarining koordinatalaridan tuzilgan massiv.

7. Ellipse *protsedurasi* — ellips yoyini chizadi.

Aniqlanishi :

Ellipse(*x,y*:integer;StAng,EndAng:word;*X*Radius,*Y*Radius: word);

(*x,y*) — ellips markazining koordinatasi;

StAng va EndAng — yoyning boshlang'ich va oxirgi burchaklari;

Xradiues va Yradius mos ravishda ellips balandligi va eni.

8. FillPoly *protsedurasi* — rangli ko'pburchak chizadi.

Aniqlanishi: FillPoly(NumPoints:word; var PolyPoints);

NumPoints — ko'pburchakning uchlari soni;

PolyPoints — ko'pburchak uchlari koordinatalaridan tuzilgan massiv.

9. GetArcCoords *protsedurasi* — oxirgi marta ishlatilgan Arc protsedurasining koordinatalarini aniqlaydi.

Aniqlanishi:GetArcCoords(var ArcCoords: ArcCoords Type);

10. GetColor *funksiyasi* — ekran rangini aniqlaydi.

Aniqlanishi: GetColor:word;

11. GetGraphMode *funksiyasi* — grafik ekranni qaytaradi.

Aniqlanishi: GetGraphMode:integer;

12. GetImage *protsedurasi* — ekranning berilgan sohasini Area da saqlaydi.

Aniqlanishi: GetImage(*x*1,*y*1,*x*2,*y*2:integer;var Area);

13. GetMaxColor *funksiyasi* — rangning eng katta qiymatini hisoblaydi.

Aniqlanishi: GetMaxColor:word;

14. GetPixel *funksiyasi* — berilgan nuqta rangini aniqlaydi.

Aniqlanishi: GetPixel(*x*,*y*:integer):word;

15. GraphErrorMsg *funksiyasi* — berilgan kod bo'yicha xato haqida ma'lumot beradi.

Aniqlanishi: GraphErrorMsg(Code:integer):string;

16. LineTo *protsedurasi* — oldingi aniqlangan nuqtadan berilgan nuqtagacha kesma chizadi.

Aniqlanishi: LineTo(x,y:integer);

17. PieSlice *protsedurasi* — sektor chizadi.

Aniqlanishi: PieSlice(x,y:integer;StAng,EndAng, Raduis:word);

3.17.1. Faylli tiplar

Faylli tipdagi o'zgaruvchilarni diskdan ma'lumot o'qib oluvchi yoki diskka ma'lumot yozib qo'yuvchi dasturlarda ishlatish mumkin. Faylli tipdagi o'zgaruvchilarni e'lon qilishda file va text xizmatchi so'zlari ishlatiladi:

```
var     mfile 1, mfile 2: file;
    afile: file;
    Prima: text;
```

text xizmatchi so'zi faylning matnli ekanligini anglatadi. Matnli fayllar maxsus belgilar bilan ajratilgan, uzunligi noma'lum bo'lgan qatorlardan tashkil topadi.

Ayrim paytlarda fayllarni bir xil tipli hadlar ketma-ketligi ko'rinishida qarash qulayroq bo'ladi. Bu ketma-ketlik qatorlar, butun sonlar yoki yozuvlardan tashkil topishi ham mumkin:

```
var     A1: file of byte;
  {A1 fayli baytlar ketma-ketligidan tashkil topgan}
    A2: file of integer;
  {A2 fayli butun sonlar ketma-ketligidan tashkil topgan}
    A3: file of string;
    {A3 fayli qatorlar ketma-ketligidan tashkil topgan}
    A4: file of string[20];
  {A4 fayli 20ta belgili qatorlarning ketma-ketligidan tashkil topgan}
  A5: text;
  {A5 fayli matnli fayl hisoblanadi}
```

Agar faylning hadlari uchun tip aniqlangan bo'lsa, bunday fayllarni *tiplashtirilgan,* aks holda esa *tiplashtirilmagan* deb ataladi:

```
var     A: file ; { tiplashtirilmagan fayl}
    B: file of char; { tiplashtirilgan fayl}
```

Fayllar bilan ishlaydigan quyidagi dasturni ko'rib chiqaylik:

```
Var
   mydata: file of integer;
   i, j, sum: integer;
begin
   assign (mudata, 'd:g'tpg'myfile.dat');
   {mydata fayl o'zgaruvchisi bilan faylning ismi
myfile.dat va uning aniq yo'li aniqlanmoqda}
   rewrite (mydata);  {fayl yozuv uchun ochiq}
  writeln ('Salom noma'lum o'rtoq...');
  writeln ('Birinchi sonni kiriting');
  readln (i);
  writeln ('Kiritilgan sonni diskdagi myfile.dat
  fayliga yozilmoqda');
  write (mydata, i); {bu operator yordamida diskdagi
myfile.dat  fayliga i sonini yoziladi}
   writeln ('Ikkinchi sonni kiriting');
   readln (j);
  writeln ('Kiritilgan ikkinchi sonni diskdagi myfile.dat
  fayliga yozilmoqda');
   write (mydata, j); {Diskka yozish bajarilmoqda}
   sum := i + j;
   writeln ('Yig'indi =', sum);
  writeln ('Yig'indi diskdagi myfile.dat fayliga yozilmoq-
da');
   write (mydata, sum);        {Diskka yozish bajaril-
moqda}
   close (mydata); {mydata fayli yopildi}
   writeln ('Xayr noma'lum o'rtoq...');
end.
```

E'tiboringizga havola etilgan dasturda *Assign, Rewrite, Write* va *Close* protseduralaridan foydalanildi. Endi shu protseduralarning va keyingi dasturda ishlatiluvchi Reset va Read protseduralarning vazifalari hamda qanday aniqlanganligi haqida qisqacha ma'lumot berib o'taylik:

Assign protsedurasi.

Vazifasi: Faylli o'zgaruvchiga tashqi fayl ismini o'zlashtiradi.

Aniqlanishi: Assign (f; name: string);bu yerda f — ixtiyoriy tipli faylli o'zgaruvchi; name — qatorli tipdagi ifoda yoki qator, fayl ismi (agar faylning to'liq yo'li ko'rsatilmagan bo'lsa fayl ishlanayotgan katalogda joylashgan bo'ladi).

Close protsedurasi.

Vazifasi: ochiq faylni yopadi.

Aniqlanishi: Close (f);bu yerda f — oldindan ochilgan faylga mos keluvchi faylli o'zgaruvchi.

Read protsedurasi.

Vazifasi: fayl hadini o'zgaruvchiga o'qiydi.

Aniqlanishi: Read (*f, v*);bu yerda *f* — faylning ixtiyoriy tipiga mos faylli o'zgaruvchi (faqat matnli tipli emas); *v*— fayl hadining tipi bilan bir xil tipli o'zgaruvchi.

Reset protsedurasi.

Vazifasi: mavjud faylni ochadi.

Aniqlanishi: Reset (f: file);bu yerdaf — faylning ixtiyoriy tipiga mos faylli o'zgaruvchi va bu o'zgaruvchi fayl bilan Assign protsedurasi orqali bog'langan bo'lishi kerak. Reset protsedurasi mazkur faylni ochadi.

Rewrite protsedurasi.

Vazifasi: yangi faylni yaratadi va ochadi.

Aniqlanishi: Rewrite (f: file);bu yerda *f* — ixtiyoriy faylli tipdagi faylli o'zgaruvchi. Rewrite protsedurasini ishlatishdan oldin *f* o'zgaruvchi Assign protsedurasi yordamida diskdagi fayl bilan bog'lanishi kerak. Rewrite protsedurasi yangi fayl tashkil qiladi.

Write protsedurasi.

Vazifasi: fayl hadiga o'zgaruvchini yozib qo'yadi.

Aniqlanishi: Write (*f, v*);bu yerda *f* — faylli o'zgaruvchi; *v* — *f* faylining hadi bilan bir xil tipli o'zgaruvchi.

Oldingi tuzgan dasturimiz «d:» diskdagi tp katalogida myfile.dat faylini tashkil qildi. Endi shu fayldan qanday qilib ma'lumotlarni o'qishni ko'rib chiqaylik.

```
Var
  mydata: file of integer;
  i, j, sum: integer;
```

```
begin
   assign (mydata, 'd:\'tp\'myfile.dat');
   reset (mydata);          {fayl o'qish uchun ochilmoq-
da}
   writeln ('Salom noma'lum o'rtoq...');
   read (mydata, i);
   writeln ('myfile.dat faylidan birinchi son o'qildi');
   read (mydata, j);
   writeln ('diskdagi myfile.dat faylidan
ikkinchi son o'qildi');
   read (mydata, sum);
   writeln ('myfile.dat faylidan uchinchi son o'qildi');
   close (mydata);          {mydata fayli yopiladi}
   writeln ('Xayr noma'lum o'rtoq...');
end.
```

Text standart fayli tip matnli fayllarni aniqlaydi. Matnli fayllar o'zaro yangi qatorga o'tish belgilari bilan ajratilgan qatorlardan tashkil topadi.

Matnli fayllar bilan ishlash uchun maxsus kiritish (Readln), chop etish (Writeln) protseduralari ko'zda tutilgan. Bu protseduralar uzunligi noma'lum qatorlarni fayllardan o'qish va fayllarga yozish uchun ishlatiladi.

Endi matnli fayllar bilan ishlashga doir quyidagi dastur bilan tanishib chiqaylik:

```
var
   mytext: text;
   s: string;
begin
   assign (mytext, 'd:\'tp\'mytext.txt');
                {mytext faylli o'zgaruvchi orqali fayl
ismi va yo'li aniqlanmoqda}
   rewrite (mytext);
                {fayl yozish uchun ochiq}
   writeln ('Sizning ismingiz?');
   readln (s);
```

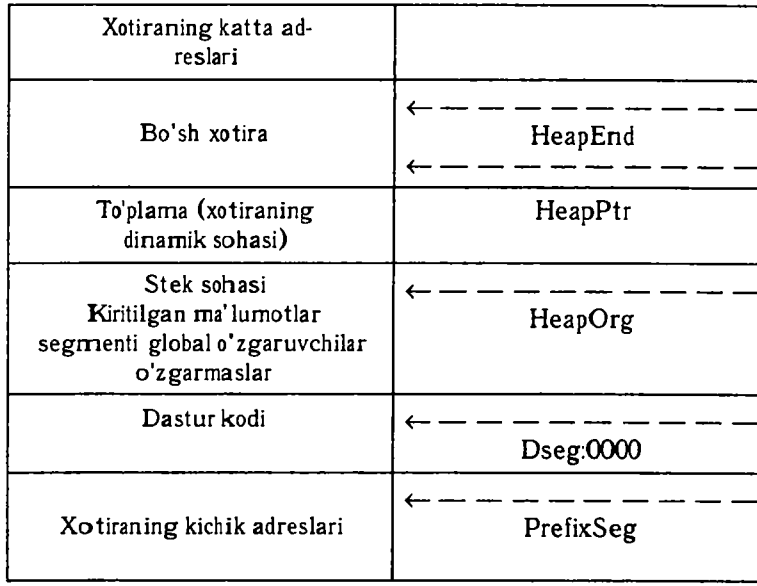
```
    writeln ('Ismingiz diskdagi mytext.txt fayliga yozil-
moqda');
    writeln (mytext, s);
        {s — qatori mytext.txt fayliga yozilmoqda}
    close ( mytext);
        {mytext fayli yopildi}
end.
```

3.17.2. Xotiraning dinamik sohasi

Dasturning (tashqi) o'zgaruvchilari komputer xotirasining ma'lumotlar segmentida joylashgan bo'ladi. Segment uzunligi esa 64 Kb ni tashkil qiladi. Demak, shunday hollar bo'lishi mumkinki, dasturning barcha o'zgaruvchilarini ma'lumotlar segmentiga joylab bo'lmaydi, ya'ni ma'lumotlar segmentga sig'maydi. Masalan, ushbu

```
var
    A: array [1..40000] of  byte;
    B: array [1..25534] of  byte;
begin
end.
```

dasturni kompilyatsiya qilishga urinish natijasiz tugaydi, ya'ni ekranda «*Error 49: Data Segment too large*» (berilgan ma'lumotlar segmenti juda katta) deb yo'l qo'yilgan xato haqida ma'lumot chiqadi. Shuning uchun katta xajmli ma'lumotlar bilan ishlashga to'g'ri kelganda bu ma'lumotlarni xotiraning dinamik sohasiga yozish maqsadga muvofiqdir. Xotiraning dinamik sohasi (to'plama deb ham atashadi) dastur egallagan deyarli barcha xotira hisoblanadi. Faqat bu sohaga ma'lumotlar kodlari, stek va ma'lumotlar segmenti xotiralari kirmaydi. Odatda dastur egallagan xotira ko'rinishi quyidagicha bo'ladi:



To'plamaning boshi va oxirini *System* modulining *Heaporg* va *HeapEnd* maxsus o'zgaruvchilari ko'rsatib turadi. Zarur paytdagi to'plama sohasining tugash joyini *HeapPtr* o'zgaruvchisi aniqlab beradi. To'plamaning o'lchovi kiritilgan ma'lumotlar segmentining o'lchovidan bir necha marta katta bo'lib, taxminan 500 kb (agar kompyuterning umumiy xotirasi 640 Kb bo'lsa) joy egallashi mumkin.

Ma'lumotlar segmentida joylashgan ma'lumotlar *statik ma'lumotlar* deb ataladi. Bunga asosiy sabab, ular uchun xotira dasturni kompilyatsiya qilish davomida ajratib qo'yiladi. Dasturning ishlashi davomida esa bu xotira o'zgarmay qoladi. To'plamadagi xotira esa dasturning ishlashi davomida to'ldirib boriladi va zarur bo'lgan paytda bu xotira bo'shatib qo'yilishi mumkin. Shuning uchun to'plamada joylashgan o'zgaruvchilarni *dinamik o'zgaruvchilar* deb ataladi. To'plamadan xotira ajratishni GetMem va New protseduralari orqali amalga oshiriladi, uni bo'shatib qo'yishni esa FreeMem va Dispose protseduralari bajaradi.

3.17.3. Ko'rsatkichlar haqida boshlang'ich ma'lumotlar

Ko'rsatkich shunday o'zgaruvchiki, u ma'lumotning qiymatini emas, balki ularning xotiradagi joylashgan adresini o'zida saqlaydi. Ko'rsatkichlarning ikki xil tipi mavjud: tiplangan va tiplanmagan. Tiplanmagan ko'rsatkichlarni e'lon qilish uchun pointer xizmatchi so'zidan foydalaniladi, tiplangan ko'rsatkichlarni e'lon qilish uchun esa ko'rsatilgan tip oldidan «^» belgisi qo'yiladi.

Misol:

```
var
   p: pointer; {tiplanmagan ko'rsatkich}
   pint: ^ integer ; {pint — tiplangan, butun tipli ko'rsatkich}
   x, y: ^ byte; {x, y — bayt tipidagi ko'rsatkich}
   pst: ^ string ; {pst — qator tipidagi ko'rsatkich}
```

Adresi ko'rsatkichda joylashgan zarur ma'lumotga ko'rsatkich qilish uchun ko'rsatkich ismidan keyin «^» belgisini yozish kerak:

```
GetImage (0, 0, 100, 100, p^); {grafik ekran sohasining ma'lumotlari r da ko'rsatilgan adresdagi xotira sohasiga yoziladi}
pint^ :=2;    {pint adresi bo'yicha 2 butun soni yoziladi}
x^ := # 12; y^ := $2; {x^ baytiga # 12 qiymatini o'zlashtiriladi, y^ baytiga esa $ 2}
pst^ :=' Ahmad'; {pst qatoriga 'Ahmad' so'zi o'zlashtiriladi}
```

Ko'rsatkichni ishlatish uchun uni faqat e'lon qilibgina qolmay, uning qanday joyni ko'rsatayotganligiga ham katta ahamiyat berish zarur. Ya'ni, ko'rsatkich operativ xotiraning kerakli ma'lumotlar joylashgan adresini aniq ko'rsatishi kerak. Ko'rsatkich adresini xato ko'rsatish kutilmagan natijalar olishga yoki butunlay dastur ishini chippakka chiqarishga olib kelishi mumkin. Shuning uchun Paskal tilida maxsus protseduralar mavjudki, ular yordamida xotira adresi aniq hisoblanadi.

1. GetMem *protsedurasi* (tiplanmagan ko'rsatkichlar uchun) — tezkor xotiradan zarur miqdorda joy ajratadi va uning adresini ko'rsatkichga joylashtiradi.

Aniqlanishi: GetMem (var p: Pointer; size : word);

p — ko'rsatkich, size — ajratilgan xotiraning bayt o'lchovidagi miqdori.

Misol:

```
var
  p: pointer;
begin  GetMem (p, 1000); {1000 bayt uzunlikdagi xotira sohasi ajratildi}
{p^  sohadan foydalanish bloki}
  FreeMem (p, 1000); {r adresdagi 1000 bayt uzunlik xotira sohasi bo'shatib qo'yildi}
end.
```

2. New *protsedurasi* (tiplangan ko'rsatkichlar uchun) — xotiradan zarur joyni band qilib, uning adresini berilgan ko'rsatkichga jo'natadi.

Aniqlanishi: New (var p: pointer);

p — berilgan ko'rsatkich.

New va *GetMem* protseduralari nisbatan juda katta hisoblanmish xotiraning dinamik sohasidan zarur miqdordagi xotira so'rab oladi. Shu xotirani band qilib, uning adresini mos ko'rsatkichga jo'natadi. Shunday hollar ham bo'ladiki, xotiraning dinamik sohasida bo'sh joy qolmaydi, u holda shu xotiraga yuqoridagi protseduralar bilan qilingan ko'rsatkich quyidagi ma'lumotli xatoni yuzaga keltiradi:

Runtime Error 203 ...

Shuning uchun xotiraning dinamik sohasini vaqti-vaqti bilan bo'shatib turish kerak. Bu ishlarni *Dispose* va *FreeMem* protseduralari bajaradi.

1. Dispose *protsedurasi* — tiplangan ko'rsatkichdagi adresda joylashgan xotirani bo'shatadi.

Aniqlanishi: Dispose (var p: pointer);

2. FreeMem *protsedurasi* — tiplanmagan ko'rsatkichda ko'rsatilgan adresdagi xotirani bo'shatadi.

Aniqlanishi: FreeMem (var p: pointer; size: word);

Ko'rsatkichning vaqtincha ishlamay turishini bildirish uchun Nil xizmatchi so'zidan foydalanish mumkin.

Bu protseduralar bilan bir qatorda quyidagi funksiyalar bilan ham tanishib chiqaylik:

1. MemAvail *funksiyasi* — xotiraning dinamik sohasidagi bo'sh sohalarning umumiy xajmini aniqlaydi.

Aniqlanishi: MemAvail : LongInt;

2. *MaxAvail* funksiyasi — xotiraning dinamik sohasidagi eng katta soha o'lchovini aniqlaydi.

Aniqlanishi: MaxAvail: LongInt;

Misol sifatida GetMem va FreeMem protseduralaridan foydalanilgan quyidagi dasturni ko'rib chiqaylik:

```
Var
P: Pointer;
Begin
GetMem(P,1024); {Xotiradan 1024 bayt joy ajratildi,
R shu sohani ko'rsatmoqda}
.....
FreeMem(P,1024); {R sohadagi 1024 bayt joy bo'shatildi}
End.
```

New va Dispose protseduralari esa quyidagicha ishlatiladi:

```
Var
        PS:^String;
  Begin
New(PS); {Xotiradan 256 bayt joy ajratildi }
PS^:q'Dinamik xotira';
Writeln(PS^);
Dispose(PS); {RS sohasigi tegishli bo'lgan joy bo'shatildi}
End.
```

4 bob

SI PROGRAMMALASH TILI

4.1. SI TILI ELEMENTLARI

4.1.1. Tilning asosiy tushunchalari

Si dasturlash tili Yuniks operatsion tizimi bilan bog'langan bo'lib, shu tizimda ishlatiladigan bir qancha dasturlar Si tilida yaratilgan. Si dasturlash tili mashinaning barcha imkoniyatlaridan to'laligicha foydalanishga imkon beradi.

Si tili mo'ljallanishiga ko'ra umumiy bo'lib, u yuqori darajali dasturlash tillari bilan Assembler orasida oraliq vaziyatni egallaydi.

Dasturni boshqaruvchi operatorlar strukturali dasturlash talablariga javob beradi. Unda kiritish-chiqarish, xotirani dinamik taqsimlash, multidasturlash, parallel hisoblash vositalari yo'q. Bu vositalar tashqi funksiyalar orqali amalga oshiriladi.

Si tilida dastur tuzish hisoblash mashinalari ishini va uning operatsion tizimini tushunish imkoniyatini yaratadi, shuning uchun ham u kasbiy dasturchilarga mo'ljallangan.

Lekin, Si dasturlash tili biron-bir tizim yoki mashina bilan qattiq bog'lanib qolmagan.

Si tilida dasturlar ma'lumotlar ustida natija olish maqsadida qilinadigan harakatni bildiradi. Dasturda harakatlar operatorlar orqali beriladi, malumotlar esa ob'ektlarni aniqlash va tavsiflash orqali amalga oshiriladi. Dasturda foydalaniladigan har bir ob'ekt tavsiflanishi zarur. Tavsiflash ob'ekt bilan bir necha xarakteristikalarni bog'laydi. Bu xarakteristikalar: tip, belgilash, xotira sinfi, harakat sohasi, boshlang'ich qiymatlar kabilardir.

Tilda asosiy tiplar: belgi (liter) bilan birgalikda butun va suzuvchi vergulli sonlar ishlatiladi. Bundan tashqari ko'rsatkichlar, massivlar, operandalardan chiqariluvchi ma'lumotlarning to'la iyerarxiyasini hosil qilish mumkin.

Ifoda operandalar va ko'rsatkichlardan tashkil topadi. Qiymat uzatish funksiyasini chaqarishni qo'shib hisoblaganda, har bir ifoda ko'rsatma bo'lishi mumkin.

Ko'rsatkich bu berilgan tipdagi ob'ektni dalillashdan iborat.

Massiv bu bir tipdagi ob'ektlar ketma-ketligi bo'lib, ularga yaqinlashish indekslari orqali amalga oshiriladi, lekin qo'shimcha aniqlash rekursiv qoidasi yordamida istalgan o'lchamdagi massivlarni aniqlash mumkin.

Tuzilmalar turli tipdagi ob'ektlar ketma-ketligi bo'lib, ularga yaqinlashish ismlar bo'yicha amalga oshiriladi. Ikki xil tuzilmalar mavjud bo'lib, oddiy tuzilmalarda (struct) elementlar xotirada tavsiflangan tartibda ketma-ket joylashadi. Istalgan paytda ularga yaqinlashish mumkin.

Variantli tuzilmalarda (union) hamma elementlar xotirada tuzilma boshlanishida joylashtiriladi. Bu holat eng keyingi qiymat berilgan elementgagina yaqinlashish imkonini beradi xolos.

O'tkazish bu butun tipdagi ob'ektlar ketma-ketligi bo'lib, ularga yaqinlashish ismi bo'yicha amalga oshiriladi.

Funksiya bu murakkab ob'ekt bo'lib, uni hisoblash natijasida berilgan tipdagi qiymat hosil qilinadi.

Ko'rsatkichlar bog'liqmas adresli mashina arifmetikasini belgilaydi. Si da ko'rsatma tuzuvchi ({ }), shart bo'yicha tarmoqlanish (if), ko'pdan biron-bir muqobillarini (switch) tanlash, yuqoriga qarab takrorlash (for, while) quyiga qarab takrorlash (do), shuningdek takrorlashni to'xtatish (break) kabi boshqaruvchi tuzilmalar mavjud.

Si tilida har bir funksiyaga rekursiv (yakka tartibda) yondashiladi. Lekin bir funksiyaniig tasviri ikkinchi funksiyaning orasida yotishi mumkin emas.

Mazkur tilning muayyat kamchiligi ham mavjud bo'lib, til tuzilmalari sintaksisi noqulay.

Si dasturlash tili birinchi marta Denis Ritchi («BELL LABORATORIES» AQSH) tomonidan 1972 yilda loyihalashtirilgan. Bu tilda skalyar qiymatlar bilan ham ish olib borib, ular o'rtasida operatsiyalar bajarish mumkin. Taqribiy qiymatlar esa elementlarga yaqinlashishni tartiblash uchun ishlatiladi.

Tilning leksikasi va sintaksisida Bekus-Nuar formasiga yaqin qoidalar qabul qilingan va bu Metaformalar «=» ifoda bilan ajratiladi. Bu belgining chap tomonida metao'zgaruvchi va o'ng tomonida esa uning qiymati yotadi.

Si tili alfavitiga quyidagilar kiradi:

— lotin alifbosining bosma va yozma harflari (A,V,...,Z,a,b,...,z)

— raqamlar: 0,1,2,3,4,5,6,7,8,9

— maxsus belgilar: «, [], {}, p, (), +, -, /, g', ; G' . ! ? < = > _ ! & * # ~ ^

— akslanmaydigan belgilar (bo'sh joy, tabulyatsiya, yangi qatorga o'tish)

Izohlarda esa boshqa belgilar ishtirok etishi ham mumkin.

/* */ belgisi yordamida izohlar keltiriladi.

[], (), {}, ;, :, ..., *, = ajratuvchi belgilar deyiladi.

[ ] — katta qavs bir yoki ko'p o'lchamli massivlarni chegaralash uchun ishlatiladi.

Masalan:

a[5] — a bir o'lchamli 5 ta elementdan iborat bo'lgan massiv;

e [3] [2] — ye ikki o'lchamli massiv (matritsa);

( ) — kichik qavslar quyidagi hollarda ishlatiladi:

1) agar operatorda ifoda shartini ajratilsa:

If (x<0) x=—x;

2) funksiyani aniqlashda va uni tavsiflashda:

float F(float x, int k) /* funksiyani aniqlash */ {funksiya — tanlash};

float F(float x, int); /* funksiyani tavsiflash */;

3) funksiyaning ko'rsatuvchilarini aniqlashda int (*pfink)(); /* pfink funksiya ko'paytmasini aniqlash uchun */;

4) operatsiyalarni bajarish ketma-ketligini aniqlashda $y=(a+b)/c$;

5) takrorlashning elementi sifatida for(i=0, j=1, i=<j;i +=r ;j+2; j++) sikl tanasi; do sikl tanasi while (k>0);

6) makroaniqliklarda

#define $R(x,y)$ sqrt((x)*(x)+(y)*(y))

Tashkiliy operator yoki blokning boshlanishini yoki tugallangan qismini bildirish uchun figurali qavslar ishlatiladi. Ularni massivlarni initsializatsiyalashda va ular tarkibini aniqlashda ham ishlatib, nuqta vergul bilan tugallanadi.

Int day []={1,2,3,4,5,6,7,8,9,10,11,12};

, — vergul belgisi ajratuvchi yoki operatsiya sifatida ishlatilishi mumkin.

; — Si tilida har bir aniqlash nuqta vergul bilan yakunlanadi.

: — metkani aniqlash uchun ishlatiladi.

... — ko'p nuqta funksiyada o'zgaruvchining parametrlari sonini aniqlashda va tasvirlashda ishlatiladi.

Qiymat uzatishni belgilash "=" tenglik belgisi yordamida amalga oshiriladi.

* — operatsiya sifatida ko'paytirish,

— belgisi esa protsesor oldi buyruqlarini aniqlash uchun xizmat qiladi.

Operatsiyalar sifatida: +, —, *, /,<, >, !, &, ~, >, <, ^ belgilari ishlatiladi.

Ular bilan keyingi paragraflarda tanishib o'tamiz.

4.1.2. Identifikatorlar

Harflardan yo belgilardan yoxud ostki chiziq bilan boshlangan raqamlar ketma-ketligi identifikator hisoblanadi.

Masalan:

KOM_ 15, sizl 98, - MAX,TIME, time.

Bosma va yozma alifboda yozilgan identifikatorlar bir-biridan farqlanadi. Jumladan, yuqoridagi oxirgi ikkita identifikator bir-biridan farqli.

Identifikator turli uzunlikka ega bo'lishi mumkin, lekin kompyuter 31 tagacha bo'lgan belgini hisobga oladi xolos. Ayrim kompyuterlarda bu cheklov yanada qattiqroq qo'yilib, faqat dastlabki ixtiyoriy identifikatorning 8 ta belgisigacha hisobga olinadi.

Bu holda NUMBER_OF_ROOM va NUMBER—OF_TECT identifikatorlari dastur bajarilishida bir xil bo'ladi.

Xizmatchi (kalit) so'zlar. Dasturchi tomonidan erkin tanlab olib qo'llanilmaydigan identifikatorlar *xizmatchi so'zlar* deb ataladi. Xizmatchi so'zlar ma'lumotlar tipini, xotira sinfini, tip xususiyatini va operatorlarini aniqlash uchun qo'llanilib, ular quyidagilardir:

auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, siseof, static, struct, with, typedef, union, unsigned, void, volatile, while.

Xizmatchi so'zlar ma'no jihatdan quyidagi guruhlarga ajratiladi.

Ma'lumotlar turlarini sinflarga ajratish uchun:

char — belgili

double —ikkilangan aniqlikdagi suzuvchi vergulli haqiqiy sonlar

enum — sanalma tip

float — suzuvchi vergulli haqiqiy sonlar

int — butun tip

long — uzun butun tip

short — qisqa butun tip

struct — tuzilmali tip

signed — ishorali tip

union — birlashtiruvchi tip

unsigned — ishorasiz tip

void — qiymat ishtirokisiz tip

typedef — tipni aniqlash sinonimini kiritish.

Tip xususiyatlarini aniqlash:

const — yagona qiymatga ega bo'lgan faqat o'qish uchun aniqlanadigan sinf xususiyati.

volatile — dasturchining aniq ko'rsatmalarisiz qiymat o'zgartiradigan ob'ekt sinfi.

Xotira sinfini belgilash uchun:

auto — avtomatik

extern — tashqi

register — registrli

static — statik

identifikatorlari ishlatiladi.

Operatorlarni qurish uchun quyidagi xizmatchi so'zlar ishlatiladi.

Break — sikldan yoki o'tkazuvchidan chiqish

continue — siklni davom ettirish

do — bajarmoq(shart ostida siklni bajarish operatorining bosh qismi)

for — uchun(parametrli takrorlash operatorining bosh qismi)

goto — o'tish(shartsiz o'tish)

if — agar—shartli operatorning belgilanishi

return — qaytish (funksiyadan)

switch — o'tkazuvchi

wile — hozircha(do sikli yakunlovchisi)

Xizmatchi so'zlarga kiradigan quyidagi identifikatorlar mavjud:

default — switch operatorida kerakli variant topilmasa harakatning davomini aniqlash

case— switch operatori variantlarini tanlash

else — yoki — if shartida muqobil tarmoqni tanlash.

4.1.3. Operatsiya belgilari

Quyidagi 8-jadvalda operatsiyalar ranglari bilan guruhlarga ajratilgan.

8-jadval

№	Operatsiya	Assotsiativ-ligi
1.	() [] → ← &	→
2.	! ~ + — ++ C & * (min) sizeof	←
3.	* / % (mul'tiplikativ) binar	→
4.	+ — (additiv binar)	→
5.	>> << (razryadli surish)	→
6.	>< = > = > (munosabat)	→
7.	==! (munosabat)	→
8.	& (razryadli kon'yunksiya "va")	→
9.	^ (razryadli "yoki")	→
10.	| (diz'yunksiya "yok")	→
11.	&& (kon'yunksiya "va")	→
12.	|| (diz'yunksiya "yok")	→
13.	? : (shriftli operatsiya)	←
14.	= * = / = % = + = & = ^ = \ = << 1 >> 1	←
15.	, (vergul operatsiyasi)	→

Munosabatlar va mantiqiy ifodalar. Munosabat operatsiyalari quyidagicha aniqlanadi:

= = teng; ! = teng emas;
< kichik; < = kichik yoki teng;
> katta; > = katta yoki teng;

1-rang operatsiyalarni yuqorida ko'rib o'tdik.

2-rang operatsiyalar unar yoki bir joyli operatsiyalar deb ataladi.

! — emas — mantiqiy inkorni bildiradi;
masalan: !2=0; !(-5)=0; !0=1;
~ — bitli inkorni ifodalaydi;
— — unar minus, arifmetik operatsiyaning ishorasini unar minusga o'zgartiradi;

+ — unar plyus, unar minusga simmetrik ravishda kiritilgan (inkrement yoki avtomatik orttirish);

++ — bir birlikka ortishi;

ikki xil shakli mavjud:

prefiksli operatsiya — operanda qiymati u foydalanguncha bittaga ortishi;

postfiksli operatsiya — operanda qiymati foydalanilgandan keyin bittaga ortishi;

— — dekrement yoki avtomatik kamaytirish; kamaytirish ikki xil shaklda ishlatiladi:

prefiksli operatsiya — operanda qiymatining bittaga u foydalanilguncha kamayishi;

postfiksli operatsiya — operanda qiymatining bittaga u foydalanilgandan keyin kamayishi;

Masalan

++m m ning qiymatini bittaga ortiradi;

—— n n ning qiymatini bittaga kamaytiradi;

i++ *i* ning qiymatini operatsiya bajarilgandan so'ng bittaga ortiradi;

j— — jning qiymatini operatsiya bajarilgandan so'ng bittaga kamaytiradi;

n=4 bo'lsa *n*++*2 ifodaning qiymati 8 bo'lib, *n* ning qiymati esa 5 ga teng bo'ladi.

++*n*2 ifodaning qiymati 10 bo'lib *n* ning qiymati 5 ga teng bo'ladi.

x+++*b* yoki *z*———*d* ifodani ko'rsak, bu ifodalar (*x* ++)+*b* yoki (*z*——)—*d* ifodalar bilan teng kuchlidir.

Multiplikativ operatsiyalar:

* — arifmetik tipdagi operandalarni ko'paytirish;

— binar minus — arifmetik operandalarni ayirish yoki ko'rsatkichlarini ayirish uchun;

/ — arifmetik tipdagi operandalarni bo'lish;

% — modul bo'yicha bo'lish;

Surilish operatsiyalari

<< — chapga surilish;

>> — o'ngga surilish;

Razryadli operatsiyalar:

& — razryadli kon'yunksiya (va);

! — razryadli diz'yunksiya («yoki»);

^ — razryadli yoki;

Surilish razryadli operatsiyalarning bajaralishi natijasida :

4	<< 2	16 ga teng
5	>> 1	2 ga teng

Eslatib o'tamizki $4_{(10)}=100_{(2)}$; $5_{(10)}=100_{(2)}$; $6_{(10)}=110_{(2)}$ va h.k.

2 ta surilish natajasida 100 kodi 1000 ga aylanadi, uning o'nli qiymati 16 ga teng.

Munosabat operatsiyalari

< — kichik;

> — katta;

<= — kichik yoki teng;

>= — katta yoki teng ;

= = — teng;

!= — teng emas.

Munosabat operatsiyasining operandalari arifmetik tipga tegishli yoki ko'rsatkichdan iborat bo'lishi lozim .

Binar (ikki joyli) operatsiyalar quyidagi guruhlarga bo'linadi:

- additiv;
- multiplikativ;
- surilish;
- razryadli;
- munosabat operatsiyalari;
- mantiqiy;
- qiymat uzatish;
- «vergul» operatsiyasi;
- qavslar operatsiya sifatida.

Additiv operatsiyalar

+ — binar plyus — arifmetik operandalari qo'yish yoki butun tipli operandalar ko'rsatkichini qo'yish uchun;

— — binar minus — arifmetik operandalarni ayirish yoki ko'rsatkichlarini ayirish uchun;

Mantiqiy binar operatsiyalar

&& — arifmetik operandalar yoki munosabatlar kon'-yunksiyasi (va).

Butun sonli qiymat 0 (yolg'on) yoki 1 (rost) qiymat qabul qiladi;

!! — arifmetik operandalar yoki munosabatlar diz'yunksiyasi (yoki) .

Butun qiymatli ifoda (0) yolg'on yoki (1) rost qiymat qabul qiladi;

Munosabat va mantiqiy qiymat natijalari:

3<5 1 ga teng

3>=5 0 ga teng;

3= =5 0 ga teng;

3 ! = 5 1 ga teng;

3!=5 SS 3= =5 1 ga teng;

3+4>5 && 3+5>4 && 4+5>3 1 ga teng

Qiymat uzatish operatorlari

= — oddiy qiymat uzatish ;

masalan: P=10.3—2*x;

*= — ko'paytirishdan so'ng qiymat uzatish;

P*=2 ifoda P=P*2 bilan teng kuchli

/= — bo'lishdan keyingi qiymat uzatish;

P/=2.2-d ifoda $P=P/(2.2—d)$ ifoda bilan teng kuchli;

%= — modul bo'yicha bo'lishdan keyin qiymat uzatish;

+= — qo'shishdan keyin qiymat uzatish;

a+=b ifoda $a=a+b$ ifoda bilan teng kuchli;

— = — ayrishdan keyin qiymat uzatish x—=4,3—z ifoda— $x=x$— (4,3—z) ifoda bilan teng kuchli;

<< = — razryadlarni chapga surishdan keyin qiymat uzatish a<<=4 ifoda $a=a$<<4 ifoda bilan teng kuchli;

>> = — razryadni o'nga surishdan keyin qiymat uzatish;

a>>=4 ifoda $a=a$>>4 ifoda bilan teng kuchli;

& = — razryadli konyunksiyadan keyin qiymat uzatish;

e& = 44 ifoda *ye*=*e*&44 ifoda bilan teng kuchli;

! = — razryadli dizyunksiyadan keyin qiymat uzatish; *a*!=*b* ifoda *a*=*a*!*b* ifoda bilan teng kuchli;

^ = — razryadli yoki chiqarib tashlashdan keyin qiymat uzatish *z*^=*x*+*y* ifoda *z*=^(*x*+*y*) ifoda bilan teng kuchli;

Vergul operatsiya sifatida

Vergul bilan ajratilgan ifodalar chapdan o'ngga qarab hisoblanadi. Natija tipi sifatida eng o'ngdagi ifoda qiymati tipi saqlanadi:

x butun tipga tegishli bo'lsa, *x*=3 bo'lganda *x*=2.3**x* ifodaning qiymati 6 bo'ladi.

4.2. DASTURNING TUZILISHI

4.2.1.Boshlang'ich dastur

Si tilida dasturni aniqlash va yozish ixtiyoriy shaklda, matnli fayl sifatida yozilishi mumkin.

Unda har bir dastur protsessoroldi direktivalari, bosh funksiya, ob'ektni tavsiflash va aniqlash ketma-ketligidan iborat.

Protsessoroldi direktivalari buyruqlari dastur matnini kompilyatsiya qilingungacha qayta o'zgartirishni boshqaradi.

'##' belgisi protsessor direktivalarini belgilash uchun ishlatiladi.

Masalan:

include <stdio.h> protsessor oldi buyrug'i dastur matnini kiritish va chiqarish funksiyalari kutubxonasini kiritadi.

Std — standart (standart); i — input (kiritish), o — output (chiqarish); h — head (sarlavha); ma'nosini bildiradi.

Formatlangan chiqarishni amalga oshirish uchun printf() buyrug'i ishlatiladi.

printf() funksiyani chaqirish quyidagicha amalga oshiriladi:

printf(format qatori, argumentlar ro'yxati);

Format qatori qo'shtirnoq ichiga olinib, qator o'zgarmaslari, ixtiyoriy matnli boshqaruvchi belgilar va ma'lumotlarni o'zgartirish xususiyatlari ko'rsatilishi mumkin. Masalan ekranga «Calom Olam» so'zini chiqarish uchun quyidagicha dastur tuzish zarur:

```
# include <stdio.h>
void main()
{
  printf("Salom olam\n");
}
```

«\n» belgisi qator o'tkazib yuborishni bildirib, kursorni yangi qatorga o'rnatadi.

Shuningdek quyidagi belgilardan ham foydalanish mumkin:

'\t' — gorizontal tabulyatsiya;
'\r' — kursorni koordinata boshiga qaytarish;
'\' — teskari og'ma chiziq;
'\' ' — apostrof;
'\"' — qo'shtirnoq;
'\0' — nol belgisi;
'\a' — signal qo'ng'iroq;
'\b' — bitta belgiga qaytish;
'\f' — qator o'tkazib yuborish;
'\v' — vertikal tabulyatsiya;
'\?' — so'roq belgisi.

Yuqoridagi belgilar *boshqaruvchi ketma-ketliklar* deb yuritiladi. Bu boshqaruvchi belgilar yordamida displeyda axborotning joylanishini o'zgartirish mumkin.

Axborotlarning tashqi shaklini o'zgartirishni boshqarish uchun o'zgartirishning maxsuslatgich belgilaridan foydalaniladi va ular % belgisi bilan birgalikda ishlatiladi.

Hisoblash xarakteriga ega bo'lgan masalalarda biz ko'proq quyidagi *o'zgartirish maxsuslatgichlaridan* foydalanamiz.

%d — o'nli butun sonlar uchun(int—tipi);
%u — ishorasiz o'nli butun sonlar uchun(unsigned-tipi);
%f — haqiqiy sonlar uchun(floatba double—tipi);

%l — suzuvchi vergulli haqiqiy sonlar uchun(double va fioat-tipi);

Agar summa haqiqiy o'zgaruvchi qiymati 2702.3 ga teng bo'lsa:

printf(« \n summa=%f, summa);

funksiyasi ekranga:

summa=2702.3

ni chiqaradi.

Quyidagi operatorlar

```
float   m,n;
int k;
```

ning bajarilishi natijasida

m=15.7; k=—75; n=15.33 ni hosil qilamiz.

O'zgaruvchilar dasturning asosiy ob'ekti hisoblanib, uning bajarilishi davomida dastur o'z qiymatini o'zgartiradi.

Arifmetik ifodada sonlar, harflar va o'zgaruvchilar qatnashib +,—, *,/ kabi amallar yordamida yoziladi. Si tilida boshqa standart matematik funksiyalar mavjud. Bu funksiyalar 9-jadvalda keltirilgan.

9-jadval

Funksiya	Si tilida ifodalanishi
$\sin x$	sin (x)
$\cos x$	cos (x)
$\mathrm{tg}\, x$	tan (x)
$\ln x$	log (x)
$\lvert x \rvert$	abs (x)
e^x	exp (x)
$\sqrt{x}$	sqrt (x)

Si dasturlash tilida o'zgaruvchilar ro'yxati foydalanilmasdan oldin kiritilishi zarur, odatda hamma o'zgaruvchilar foydalaniladigan birinchi funksiyadan avval yoziladi. Funksiyadagi o'zgaruvchilarning xususiyati avvaldan e'lon qilinadi.

Int turidagi sonlar 16 razryadli (—32768 dan 32767 gacha)

float turidagi sonlar esa 32 razryadli ($3.4*10^{-38}$dan $3.4*10^{38}$gacha).

Si da int va float dan tashqari ma'lumotlarning bir qancha bazaviy tiplari mavjud.

Ko'rsatilgan tiplarning o'lchamlari mashina turlariga qarab o'zgarishi mumkin.

Bazaviy tipdan massivlar, tuzilmalar birlashmalar va ko'rsatkichlarni hosil qilish mumkin.

Yuqorida ko'rib o'tganimizdek printf funksiyasi ob'ektni ekranga chiqaribgina qolmay balki uni istalgan ko'rinishda chop etadi.

Axborotni kiritish uchun tilda scanf, read funksiyalaridan foydalaniladi.

scanf ((format qatori), (adres1), (adres2)...)

Bu yerda printf dagi singari format qatorlarini qo'llash mumkin.

1-Misol

```
# include <stdoi.h>
main ()
{
char name [ 30 ];
printf («Sizning ismingiz nima ?\p»);
scanf («% s», name);
printf  («Salom  % s\n», name);
        }
```

Bu dasturning bajarilishi natijasida:

Sizning ismingiz nima? .....

Salom.....

ifodalari hosil qilinadi.

2-Misol

$y = \frac{x^2-3x+2}{\sqrt{2x^3-1}}$ funksiyaning berilgan x uchun qiymatini hisoblash dastursini tuzing.

```
# include <stdio.h>
# include <math.h>
main()
{
float x,y;
printf ("x ning qiymatini kiriting:" );
scanf ("%f", x);
y=(x*x—3*x+2)/sqrt (2*x*x*x—1 );
printf ("x=% At y=%f", x,y);
}
```

3 - misol

$$y=\begin{cases}\frac{(m-2)(m-1)m}{6}, & \text{arap } n\le 9;\\ 2^{29-n}, & \text{arap } 9<n<29;\\ 1, & \text{arap } n=29;\\ (n-9)!, & \text{arap } n>29.\end{cases}$$

ifodaning qiymatini berilgan n uchun hisoblash dasturini tuzing.

```
# include <stdio.h>
# include <math.h>
 main()
{
float y ,a,p;
int n,i,a;
printf ("n ni kiriting:" );
scanf ("%d", n);
if (n<=9)
y=(n*(n—1)*(n—2))/6;
 else if(n>29)
 i=1;
 p=1;
 while (i<=n—9)
a=a*i;
 i++;
```

```
y=a;
 else if(n=29)
y=l;
    else
y=exp ((29—n)*log(2));
printf ("y=% f n=%d", y,n);
}
```

scanf funksiyasidan foydalanish boshqa bir muammoni vujudga keltiradi. Birinchi misolimizni qayta bajarib, ism va familyamizni kiritsak, faqat ism chiqadi xolos. Nima uchun? Chunki, ismdan keyin quyiladigan bo'sh joy scanf funksiyasiga qator tugaganligini bildiradi. Bu vaziyatdan qanday chiqish mumkin?

Ikki xil usul mavjud. Mana bulardan biri:

```
# include <stdio.h>
main( )
{
char name [60];
printf («Ismingiz nima?» );
gets (name);
printf («Salom,% s\n», name);
}
```

gets funksiyasi siz nimani tersangiz shuni o'qiydi lekin qator oxiriga \O belgisini qo'shadi. Ikkinchi ucul quyidagicha:

```
# include <stdio.h>
main()
{
char first [20],middle[20],last[20];
printf («Ismingiz nima?» );
scanf(«%s %s %s», firs,middle,last)
printf («Salom,% s, yoki %s \n»,first, last);
}
```

Yuqoridagilardan tashqari getch funksiyasi ham mavjud bo'lib, u tugmachadan kiritilgan yagona belgini o'qiydi.

4.2.2. Dastur ob'ektlarining «yashash vaqti» va «harakat» ko'lami

O'zgaruvchining (global yoki lokal) «yashash vaqti» quyidagi qoida orqali aniqlanadi:

1. E'lon qilingan global o'zgaruvchilar dasturning bajarilishi davomida saqlanadi;

2. Lokal o'zgaruvchilar register yoki auto sinfidagi o'zgaruvchilar e'lon qilingan blok bajarilishi davomidagina «yashaydi». Agar bu o'zgaruvchi static yoki extern tipiga ta'lluqli bo'lsa, u holda o'zgaruvchi butun dastur bajarilishi davomida «yashaydi».

O'zgaruvchi va funksiyalarning dasturdagi ko'rinishi quyidagi qoidalar yordamida aniqlanadi:

1. E'lon qilingan yoki global aniqlangan o'zgaruvchi e'lon qilingan nuqtadan yoki bajariluvchi fayl oxirigacha aniqlangan o'zgaruvchida ko'rinadi. O'zgaruvchini boshqa bajariluvchi fayllarda xam ko'rinadigan qilish mumkin. Buning uchun bu fayllarda uni extern xotira sinfi bilan e'lon qilish kerak.

2. E'lon qilingan yoki lokal aniqlangan o'zgaruvchi e'lon qilingan nuqtadan yoki shu blok oxirigacha ko'rinadi. Bunday o'zgaruvchi *lokal o'zgaruvchi* deb ataladi.

3. O'zgaruvchi qamrab oluvchi bloklarda ko'riladi. Shuningdek, global e'lon qilingan o'zgaruvchilar ham ichki bloklarda ko'rinadi. Bu ichma-ich ko'rinish deb ataladi. Gárchi blok ichida e'lon qilingan o'zgaruvchi qamrab oluvchi blokdagi o'zgaruvchining nomi bilan bir xil bo'lsa ham, ammo ular aslida har-xil o'zgaruvchilar bo'ladi. Qamrab oluvchi blokdagi o'zgaruvchi ichki blokda ko'rinmaydi

4. static xotira sinfdagi funksiyalar ular aniqlangan bajariluvchi fayldagina ko'rinadilar. Boshqa har qanday funksiyalar esa butun dasturda ko'rinadi.

Funksiyalardagi metkalar butun funksiya bajarilish davomida ko'rinadilar.

Funksiya prototipida e'lon qilingan formal parametrlar ismlari uni e'lon qilish nuqtasidan boshlab funksiyani e'lon qilish tugagungacha ko'rinadi.

4.3. OPERATORLAR

4.3.1. Shartli operatorlar

Yana oddiy dasturning tuzilishiga qaytamiz. U faqat bittagina main() funksiyasidan iborat edi.

```
Protsessor oldi direktivalari
main()
 obektlarni_aniqlash;
   bajariluvchi_operator
     }
```

Har bir bajariluvchi operator dasturning keyingi qadamdagi harakatini ifoda etadi. Operator qiymatga ega emas. Harakat xususiyati jihatidan operatorlar ikki tipga ajratiladi: ma'lumotlarni o'zgartirish operatorlari va dastur ishini bajaruvchi operatorlar.

Ma'lumotlarni o'zgartirish tipik operatorlari—qiymat uzatish hamda nuqta va vergul bilan yakunlanuvchi ixtiyoriy ifoda bo'ladi.

Dastur ishini boshqaruvchi operatorlar dastur **konstruksiyasini boshqaruvchi operatorlar** deyiladi. Ularga:

— tarkibiy operatorlar;
— tanlash operatorlari;
— takrorlash operatorlari;
— o'tish operatorlari kiradi.

Quyida *tarkibiy operatorlar*ning qismi keltirilgan:

```
{ n++ ;
sum+= (float)n
 }
```

Tanlash operatorlari esa — shartli operator (if) va qayta ulovchi (switsh) dan iborat.

Takrorlash (sikl) operatorlari: sharti oldindan berilgan operator (while), parametrli takrorlash (for) va sharti keyin keluvchi operator (do) lardan iborat.

O'tish operatorlari boshqarishni shartsiz uzatishni ta'minlaydi. Ularga goto (shartsiz o'tish), continue (takrorlash hozirgi iteratsiyasini yakunlash), break (takror-

lashdan yoki qayta ulovchidan chiqish) va return (funksiyalardan qaytish) kabilar kiradi.

Biz bu paragrafda *shartli operatorlar* bilan tanishib o'tamiz. U quyidagi qisqa ko'rinishga ega:

```
If
If(x<o && x>—10) x=—x;
```

Mazkur shartli operator qisqa shakldan tashqari, yana to'liq shaklga ham ega bo'lib, uning ko'rinishi:

```
If(shartli_ifoda)
1-operator;
else
2-operator;
```

Masalan:

```
If(x>0)
b=x;
else
b=—x;
```

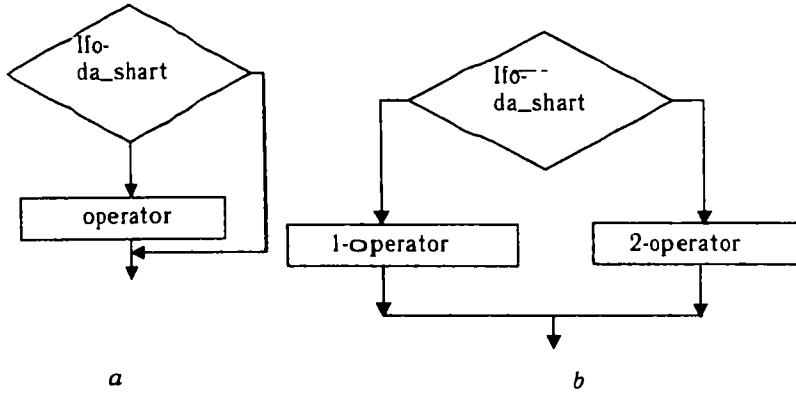
Ularning bajarilish tarhi 8-rasmda keltirilgan.

Misol: $ax^2+bx+c=0$ kvadrat tenglamaning x_1, x_2 — haqiqiy ildizlarini aniqlash:

```
# include(stdio.h)
main( )
{
float a,b,c,d,x1,x2;
d=b*b—4*a*c
if (d>=0.0)

   x1=(—b+sqrt(d)/2/a;
   x2=(—b—sqrt(d)/2/a:
   printf («\n x1= %f  x2= %f», x1, x2 );

   else
   printf («\n Haqiqiy ildizlar mavjud emas»);
   } ko'rinishida aniqlanadi.
```



8-rasm. a) Qisqa shakl; *b)* To'liq shakl.

4.3.2. Takrorlash operatorlari (for, while, do)

Boshqa dasturlash tillari singari SI tilida ham takrorlashlarni hosil qilish uchun maxsus vositalar mavjud. Ko'pchilik dasturlash tillarini takrorlash operatori ikki elementdan: bosh qism va tanadan iborat. Tana takrorlashda bajariladigan operatorlarni o'z ichiga oladi. Bosh qism esa tanadagi operatorlarning takroriy bajarilishini ta'minlaydi. SI tilida takrorlash operatorlarining uch turi ishlatiladi, ular quyidagi xizmatchi so'zlar bilan belgilanadi.

while, for, do.

while va for quyidagi tarh bo'yicha tuziladi:

Takrorlash bosh qismi

takrorlash tanasi

do takrorlashi boshqacha tuzilmaga ega bo'lib, yuqoridan va pastdan konstruksiyasiga asoslanadi. Shuning uchun ham do takrorlashida takrorlashning bosh qismi haqida gapirish o'rinsizdir.

Shuni eslatib o'tamizki, har uchala takrorlash operatorida ham takrorlash tanasi {} belgisi orasidagi alohida yoki tashkiliy operator bo'lishi mumkin. Takrorlash tanasi bo'sh operator bo'lishi ham mumkin.

while — takrorlashi quyidagi ko'rinishga ega:

while (ifoda_shart)
takrorlash tanasi

Ifoda_shart sifatida ko'proq munosabat yoki mantiqiy ifodadan foydalaniladi. Agar u rost qiymat qabul qilsa, takrorlash tanasi ifoda_shart yolg'on bo'lguncha bajariladi.

Shunga e'tibor beringki, ifodaning to'g'riligi takrorlash tanasining bajarilishini ta'minlaydi. Shunday qilib, oldindan yolg'on bo'lgan ifoda_shartda takrorlash biror marta ham bajarilmaydi.

Ifoda_shart arifmetik ifoda bo'lishi ham mumkin

do takrorlashi quyidagi ko'rinishga ega:

do
takrorlash tanasi
while (ifoda_shart)

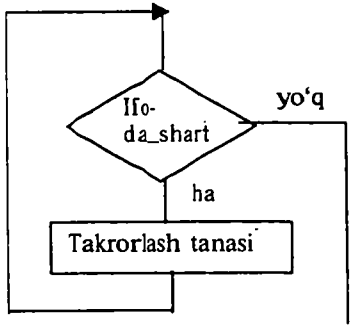
Ifoda_shart while takrorlashdagi kabi mantiqiy yoki arifmetik ifoda bo'lishi mumkin. do takrorlashda takrorlash_tanasi hech bo'lmaganda bir marta amalga oshadi.

for takrorlashi (parametrli takrorlash) quyidagi ko'rinishga ega:

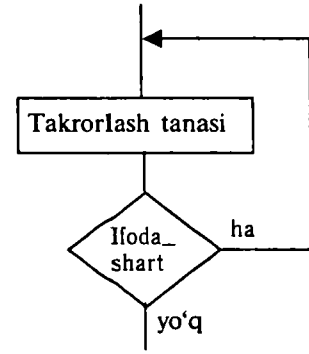
takrorlash_tanasi for (ifoda —1; ifoda_shart; ifoda—3) operatordagi birinchi va uchinchi ifoda vergul bilan ajratilgan bir necha ifodalardan iborat bo'lishi mumkin.

Ifoda_1— takrorlash boshlanguncha bo'lgan harakatni bildiradi, ya'ni takrorlash uchun boshlang'ich shartni ifodalaydi. Ko'proq u qiymat uzatishni ifodalaydi. Ifoda_shart — odatda mantiqiy yoki arifmetik ifoda bo'lishi mumkin. U shartning tugashi yoki takrorlashning davom ettirilishini anglatadi. Agar u rost bo'lsa, u holda takrorlash_tanasi bajariladi. So'ngra ifoda_3 hisoblanadi u bajarilgach, ifoda_shartning to'g'riligi hisoblanadi va hammasi yana qaytariladi. Ifoda_1 faqat bir marta bajariladi, ifoda_shart va ifoda_3 takrorlash tanasining ifoda_sharti yolg'on bo'lguncha davom etadi.

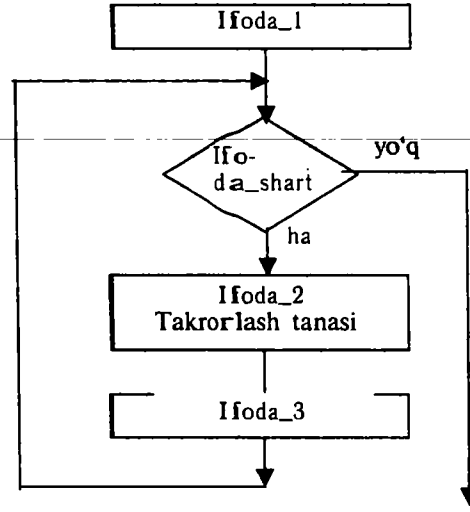
9-rasmda while, do va for takrorlashlarini tashkil qilish keltirilgan.



a) while buyrug'i takrorlashi



b) do buyrug'i takrorlashi



d) for buyrug'i takrorlashi

9-rasm.

Masalan e^x ning taqribiy qiymatini hisoblashni ko'rib o'taylik. Bizga matematik analiz kursidan ma'lumki

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{i=0}^{n} \frac{x^n}{n!}$$

formula bilan hisoblanadi.

Biz quyida dasturning bir qismini ko'rib o'tamiz.

```
:
i=2;
b=1.0;
r=x;
while (r>eps!!r<—eps)
{
b=b+r;
x=r*x/i;
i++;
}
do takrorlashi orqali
i=1;
b=0.0;
r=1.0;
do {
b=b+r;
r=r*x/i;
i++;
}
while(r>=eps!!r<=—eps);
for parametrli takrorlash orqali
i=2;
b=1.0;
r=x ;
for(;r>eps!!r<—eps;)
{
b=b+r;
r=r*x/i;
i++;
}
```

Lekin for sikli yuqoridagilarga qaraganda birmuncha afzalliklarga ega bo'lib, uni quyidagicha yozishimiz mumkin:

```
for(i=2,b=1.0,x=x;x>eps!!x<—eps;i++)
{
```

```
b=b+x;
x=x*x/i;
}
```

yig'indini hisoblash dasturining to'la matni quyidagicha:

```
/*eksponentani hisoblash dasturi */
Dastur bajarilishi natijasida:
X ning qiymatini kiriting: x=1
Aniqlikni kiriting : eps=0.01
natija: 2.708333
aniqlik      0.008333
Qator hadlari soni: 6 ni hosil qilamiz.
Dastur bajarilishining boshqa varianti.
X ning qiymatini kiriting: x=1
Aniqlikni kiriting: eps = —0.3
Aniqlikni kiriting:eps =—0.0001
Natija: 2.718254
Aniqlik: 0.000025
Qator hadlari soni: 9ta
```

Oxirgi variantda dastlab noto'g'ri yers=—0.3 qiymat kiritildi shuning uchun ham aniqlikni qayta kiritishga to'g'ri keldi.

4.3.3. Break operatori

Yuqorida ko'rib o'tgan misolimizda boshlang'ich ma'lumotlarni qayta kiritish uncha qulay emas, ekranda xatolik haqida hech qanday xabar berilmadi. Bunday holatlardan chiqishning turli yo'llari mavjud.

Ulardan biri metka va o'tkazish operatori goto dan foydalanishdir. Lekin strukturali dasturlash nuqtai nazaridan bunday usul noto'g'ri hisoblanadi.

Bunday natijaga strukturali dasturlash tamoyillarini buzmagan holda takrorlash tanasida uzilish operatori break ni qo'llash orqali ham erishish mumkin.

Bu operator takrorlash bajarilishini tugatadi va boshqarishni takrorlashdan keyingi operatorga uzatadi.

Misol: $c = \sum_{i=1}^{n} a^i$ ifodani $a,n(n>0)$ — o'zgaruvchilarni kiritish yordamida hisoblang.

Dasturning ko'rinishi quyidagicha:

```
/ *son darajalari yig'indisi  */
# include <stdio.h>
main()
{
double a, c; /* c —yig'indi * /
int i,n;
printf(«\n a ning qiymatini kiriting a=»);
scanf («%f», &a );
while(1)
{
printf(«\n n — ning qiymatini kiriting n=»);
scanf(«%d»,&n) ;
if (n>0) break ;
printf(«xato ! n>0 bo'lishi kerak \n»);
}
for( c=0.0, p=1.0, i=1 ; i<=n ;  i++)
{
p*=a;
c+=p;
}
printf( «\n yig'indi c=%f», c);
}
```

Dasturning bajarilish natijasi:

```
a ning qiymatini kiriting  a=8.8
n ning qiymatini kiriting  n=—15
xato! n>0 bo'lishi kerak
n ning qiymatini kiriting: n=15
yig'indi c=1. 65817e +14
```

4.3.4. Continue operatori

Takrorlash tanasida operatorlarning bajarilishiga aralashish imkoniyatini beruvchi operatorlardan biri continue operatoridir.

U takrorlashning ixtiyoriy qismida takrorlashning bajarilish shartini tekshirishga va uning natijasida takrorlashni tugatish yoki davom ettirishga xizmat qiladi. Bu operatorning bajarilishini quyidagi dastur orqali ko'rishimiz mumkin:

```
# include <stdio.h>
/* musbat sonlar yig'indisini toping */
main()
float s,x ; /* s —yig'indi, x—kiritiluvchi son*/
int k;  /* k — musbat had soni*/
printf («\ n sonlarni kiriting oxirida 0 bilan»)
for (x=1.0, s=0.0 , k=0; x!= 0.0 )
{
scanf(«%f», &x) ;
if  (x<= 0.0) continue;
   k++; s+=x;
}
printf(«\n yig'indi =%f  musbat element soni
=%d»,s,k) ;
}
```

Dasturning bajarilish natijasida:

```
Sonlarni kiriting oxirida 0 bilan
5. 0        —7.0  11    —4   10    0. 0
yig'indi = 26.0    musbat element soni = 3.
```

4.4. MASSIVLAR

Massivlar bir hil tipdagi (double, float, int va h.k.) elementlar guruhidan iborat bo'ladi. Kompilyator e'lon qilingan massivdan uning elementlari tipi va ularning soni to'g'risida ma'lumotlarga ega bo'lishi kerak. Massivni e'lon qilishning ikki xil formati mavjud:

maxsus_tip [konstantali_ifoda];

maxsus_tip [ ];

Maxsus tip massivning identifikatoridir.

Maxsus_tip esa e'lon qilinayotgan elementlarning tipini beradi. Massiv elementlarining tipi funksiyaga va void tipiga taalluqli bo'lmasligi kerak.

Konstantali-ifoda massiv elementlari sonini bildiradi. Massivni e'lon qilishda quyidagi xollardagina konstantali-ifodani tushirib qoldirish mumkin:

— massiv funksiyaning formal parametri sifatida e'lon qilinganda,

— e'lon qilingan massiv boshqa fayldagi aniq massivga yo'llangan bo'lsa.

Oxirgi misolda w[3][3] massivi e'lon qilingan. Figurali qavslar ichiga olingan ifodalar massivning satriga mos keladi, qavslar qoldirib ketilsa initsializatsiya noto'g'ri bajariladi.

SI tilida massivlarning kesishmasidan ham foydalanish mumkin, lekin bu kesishmalardan foydalanishga ko'p cheklovlar qo'yiladi. Kesishmalar bir yoki bir necha juft kvadrat qavslarni olib tashlash orqali amalga oshiriladi. Massivlarning kesishmasi hisoblash jarayonini tashkil qilishda ishlatiladi.

Misollar:

int *s*[2][3];

Agar ayrim funksiyaga s[0] orqali murojaat qilinsa, s massivning nolinchi satri olinadi.

int *b*[2][3][4];

b massivga *b*[1][2] orqali murojaat qilinganda to'rt elementdan iborat vektor uzatiladi, *b*[1] orqali murojaat qilinganda esa 3x4 ikki o'lchamli massiv uzatiladi.

Belgili massivlarni e'lon qilish quyidagicha bo'ladi:

char str[ ] = «belgili massivni e'lon qilish».

4.4.1. Massivlar ustida amallar

Massivlarning qo'llanilishiga doir misollar ko'rib o'tamiz.

1-misol.

$$S = \sum_{j=1}^{10} \left\lceil \prod_{i=1}^{5} b_{ij} + \sum_{i=1}^{5} b_{ij} \right\rceil.$$

Dasturini tuzamiz:

```
# include <stdio.h>
main ()
```

```
{ float p,c;
int i , j;
double b[10] [5]
printf(«\ n  Massiv elementlarini kiriting») ;
for ( j=1 ; j<10 ; j++)
{
for ( i=1 ; i<5 ; i++)
{
printf(«b[ %d ] [ %d]=», j,i )
scanf(« %f %f , & x[ j] [ i ]»)
}
 for ( s=0.0 , j=0, ; j< 10 ; j++)
  {
   for (p=1.0 , c=0. 0 , i=0 ; i<5 ; i++)
   {
        p*= a[ j] [i] ;
        c += a[ j] [i] ;
        }
        s+= c+p ;
   }
}}
```

2-misol.

Bir o'lchamli massiv elementlarini tartiblash.

$a(n)$ $1 < n < 100$ massivning elementlarini qiymatlari o'sib borish tartibida joylashtiring.

```
# include <stdio.h>
main (){ int n , i, j ;
float a [100] ,b ;
while (1)
{
  printf(«\n  Elementlar sonini kiriting n=») ;
  scanf(«%d»,& n);
  if ( n> 1  && <=100) break ;
  printf(«xato! 1<n<=100 bo'lishi zarur !») ;
}
  printf(«\n massiv elementlari qiymatlarini kiriting
: \n») ;
```

```
for ( j=0 ; j<n ; j++)
  {
    printf( a [%d ] , j+) ;
    scanf(«%f» , &a[ j] ) ;
   }
  for (i=0; i<n—1; i++)
      for (j=i+1; j<n; j++)
      if (a[i]>a[j])
                  {
                    b= a[i];
                    a[i]=b[j];
                  }
            printf(«\n Tartiblangan massiv: ) n»);
                              for (j=0; j<n; j++)
                        printf(«a(%d)=%f \n»,j+1,a[j]);
}
```

Dasturning bajarilishi
Elementlar sonini kiriting n=3
a[1]=15.8
a[2]=11.2
a[3]=—2.3
Tartiblangan massiv:
a[1]=—2.3
a[2]=11.2
a[3]=15.8

4.5. FUNKSIYA

4.5.1. Funksiyani aniqlash

Si tili sintaksiga ko'ra unda uchta hosilaviy tip aniqlanadi: massiv, ko'rsatkich, funksiya. Biz bu bo'limda funksiya haqida fikr yuritamiz.

Funksiyalar borasida, Si tilida ikki tomonlama qarash mavjud. Birinchi tomondan, funksiya hosilaviy tipga taalluqli bo'lsa, ikkinchi tomondan Si tilida funksiya bajariluvchi modul hisoblanadi. Bajaruvchi modul tushunchasi boshqa dasturlash tillaridagi protsedura, qism dastur, qismdastur—funksiya sinonimlari bilan mos kela-

di. SI tilida hamma funksiyalar standart talabiga ko'ra yagona ko'rinishga ega:

tip funksiya _ismi_ (parametrlar _ xususiyati)
funksiya _tanasi.

Birinchi qator — mazmuniga ko'ra funksiyaning bosh qismi bo'lib, o'zining timsolidan oxirida [;] nuqta vergul qatnashmasligi va albatta rasmiy parametrlarning qatnashishi bilan farq qiladi.

Bu yerda tip qiymat qaytarmaydigan void turiga yoki qiymat qaytaruvchi funksiyalar turiga taalluqli bo'lishi mumkin.

Biz oldingi bo'limlarda qiymat qaytaruvchi bazaviy turlarni ko'rib o'tdik (char, int,double va boshqalar).

Funksiyaning ismi yo bosh funksiya uchun main(), yoki dasturchi tomonidan dasturdagi ixtiyoriy tanlanadigan xizmatchi so'zlar va boshqa ob'ektlar ismi bilan mos tushmaydigan ism (identifikator) bo'lishi lozim.

Parametrlar _ xususiyati esa — bo'sh yoki rasmiy parametrlar ro'yxati bo'lib, ular quyidagi ko'rinishga ega bo'ladi:

tipning _belgilanishi
parametr_ ismi.

Parametrlar ro'yxati nuqta verguldan keyin [. . .] (ko'p nuqta) bilan tugashi mumkin. Ko'p nuqta funksiyaga ko'p sonli haqiqiy parametrlar bilan murojaat qilishni anglatadi. Bunday imkoniyat funksiya tanasiga maxsus vositalar yordamida biriktirilishi zarur.

Masalan:

*int printf(conct char * format, . . .);*
*int scanf(conct char * format, . . .);*

funksiyalari nazariy jihatdan formatlangan kiritish va chiqarishni cheklamagan miqdordagi haqiqiy parametrlar uchun qo'llashni ta'minlaydi.

Funksiya_tanasi — dasturning figurali qavslar ichiga olingan sarlavhadan keyin keluvchi funksiyalarni aniqlash qismidir.

Funksiya_tanasi blok yoki tarkibiy operator bo'lishi mumkin. Si tilida funksiya tanasi ichida boshqa funksi-

yani aniqlash imkoni yo'q. Funksiya tanasida funksiya chaqirilgan nuqtasidan chiqish operatori bo'lishi hamisha ham talab etilavermaydi. U ikki xil shaklga ega:

return;

return _ifoda;

Birinchi shakl hech qanday qiymat qaytarmaydigan void tipiga to'g'ri keladi.

Shunday qilib Si tilida funksiya ko'rsatilgan tipda qiymat qaytaruvchi, yoki hech narsa qaytarmaydigan parametrlar bilan hamda parametrlarsiz ham qo'llanishi mumkin.

Funksiyani aniqlashning umumiy shakli quyidagicha:

tip_natija

funksiya_ismi (*parametrlar_ro'yxati*) *parametrlar_xususiyati;*

funksiya_tanasi

4.5.2. Funksiyani e'lon qilish va chiqarish.

Funksiyaga aniq murojaat qilish uchun u haqidagi barcha ma'lumotlar kompilyatorga oldindan ma'lum bo'lishi zarur, ya'ni funksiyani chaqirishdan avval standartga ko'ra o'sha funksiya uning aniqlanishini yoki tavsifini dasturga joylashtirish kerak.

Standart bo'yicha funksiyani uning timsoli aniqlaydi:

tip funksiya_ismi (*parametrlar _ xususiyatlari*);

Masalan:

double f (*int n, float x*);

double f (*int, float*);

irodalar bir-biri bilan ekvivalent.

Funksiyaga murojaat qilish uchun «()» operatsiyasi bilan ifodalardan foydalaniladi:

funksiyaning_belgilanishi (*haqiqiy_parametrlar_ro'yxati*)

«()» operatsiyaning operandalari sifatida *funksiyaning_belgilanishi* va *haqiqiy_parametrlar_ro'yxati* qatnashmoqda.

Funksiyaning belgilanishi bu uning ismi, *haqiqiy_parametrlar_ro'yxati* argumentlar deb atalib, ularning soni funksiyaning rasmiy parametrlari soniga teng.

Haqiqiy va rasmiy parametrlar orasidagi moslik ularning ro'yxatda o'zaro joylashishiga qarab aniqlanadi. Haqiqiy parametrlarni hisoblash (o'ngdan chapga yoki chapdan o'nga) Si tili standartida aniqlanmagan. Rasmiy va haqiqiy parametrlar tiplari bo'yicha mos bo'lishi zarur. Haqiqiy parametr tipi rasmiy parametr tipi bilan bir xil bo'lgani yaxshiroq. Aks holda, bunday tiplarni keltirish mumkin bo'lsa, kompilyator tiplarni yaratish buyruqlariga avtomatik ravishda qo'shadi.

Masalan funksiya timsoli bilan quyidagicha aniqlangan bo'lsin:

```
int g (int,long);
```

Dasturda uni chaqirish quyidagicha:

```
g (5.0+m,6.3e+2)
```

Bu yerda ikkala parametr ham double tipiga ega. Kompilyator funksiya timsoliga asoslanib, avtomatik ravishda mana bunday almashtirishni qaraydi:

```
g ((int)(3.0+m), (long) 6.3e+2)
```

Funksiya ifoda hisoblanganligi sababli, uning tanasidagi operatorlar bajarilgach, chaqirish nuqtasiga ayrim qiymatlar qaytariladi, uning tipi esa qat'iy ravishda uning timsolida aniqlangan funksiya ismi oldida keluvchi tipga mos keladi.

Masalan:

```
float ft (douible x, int n)
{
if (x<n) return x;
return n;
}
```

hamisha float tipidagi qiymatni qaytaradi.

Funksiyani chaqirish—ifoda hisoblanib, bunday ifoda dastur matnida funksiyaning qaytaradigan qiymatlariga bog'liq bo'ladi.

Masalan:

```
void print (int gg, int mm, int dd)
 {
printf («\n yil:%d», gg);
```

```
printf («\n oy:%d», mm);
printf («\n kun:%d», dd);
}
```

Unga murojaat

```
print (2002, 10, 15);
```

Ekranga quyidagicha natijani chiqarish bilan yakunlanadi.

Yil: 2002 oy:10 kun:15

Misol. Uchburchakning tomonlari berilsa, uning perimetri va yuzini hisoblovchi dasturni ko'rib o'taylik.

Bizga geometriya kursidan ma'lumki, tomonlari a, b, c bo'lgan uchburchakni yasash uchun $a+b \geqslant c$, $a+c \geqslant b$, $b+c \geqslant a$ tengsizliklar bajarilishi kerak. Bu holda, uchburchak perimetri $r=a+b+c$, uning yuzi esa

$$S = \sqrt{\frac{p}{2}\left(\frac{p}{2}-a\right)\left(\frac{p}{2}-b\right)\left(\frac{p}{2}-c\right)}$$

formulalari yordamida topiladi.

Dasturning ko'rinishi quyidagicha:

```
# include <stdio.h>
# include <match.h>
main()
{
  float x,y,z,pp,ss;
/ *timsol:* /
int triangle(float,float,float,float*,float*);
printf(«\ n x ni kiriting x=»);
scanf(«%f», &x);
printf(«\t y ni kiriting: y=»);
scanf(«%f» &y);
printf(«\t z ni kiriting: z=»);
scanf(«%f «, &z);
if (triangle(x,y,z,&pp,&ss)= =1)
{
print(«Perimetr=% f «,pp);
print («yuza =% f», ss);
}
```

```
else
print ("\n malumotlar xato»);
}
/* funksiyani aniqlash */
int triangle(float a,float b,float c,float *perimeter,float
*arrea);
{
float e;
* perimeter=*arrea=0.0;
if (a+b<=c:: a+c<=b:: b+c <=a);
return 0
*perimeter= a+b+c
e=*perimeter/2;
*arrea=sqrt(e*(e—a)*(e—b)*(e—c));
return 1
}
```

Dasturning bajarilishi:

x ni kiriting: x=3
y ni kiriting: y=4
z ni kiriting: z=5

Natija:

perimetr=12.000000
yuza=6.000000

4.6 PROTSESSOR DIREKTIVALARI VA KOMPILYATORGA KO'RSATMA

4.6.1. Nomlangan o'zgaruvchi va makroaniqlashlar

Bizga ma'lumki, Si tilida har bir dastur protsessor oldi direktivalaridan iborat. Protsessoroldi buyruqlari direktivalar deb ataladi. Ularning har biri # belgisi bilan belgilanadi. Biz bundan oldingi boblarda # include direktivasini ko'rib o'tdik. Direktivalarning umumiy formati quyidagicha:

#direktiva_ismi protsessor_leksemalari

#belgisidan oldin va keyin probellar ruxsat etiladi. Matnli qator oxiri protsessor oldi direktivasining tugashini bildiradi.

#define direktivasi ko'p ishlatiladigan identifikatorlarni o'zgarmaslarga almashtirish uchun xizmat qiladi va *nomlangan o'zgaruvchi* deb ataladi.

Bu direktiva ikki xil sintaktik shaklga ega bo'ladi:

#define identifikator matn;

#define identifikator(parametrlar ro'yxati)matn.

Masalan:

#define WIDTH 80

#define LENGTH (WIDTH+10)

Agar ekranda biror o'zgaruvchining qiymatini tez-tez chop qilish kerak bo'lsa, uni quyidagicha amalga oshirish mumkin:

```
#define AK printf («\element raqami=%d.»,N);
```

bu direktivadan so'ng,

```
int N=4;
AK;
```

operatorlar ketma-ketligi,

Dasturning bajarilishi natijasida:

Element raqami =4 yozuvini chiqaradi.

```
#define k 50
#define PK printf(«n Elementlar soni %d.»,K)
....
RK;
....
```

Ekranga «Elementlar soni RK=50» ni chiqaradi.

#undef ism (identifikator)

buyrug'i bajarilishi bilan ism yoki makros aniqlanishini bekor qilish mumkin.

Masalan:

```
#define M 16
#undef M
```

M = 16 ni bekor qiladi.

```
....
A = 10
```

....
#define A x
....
A = 5
#undef A
....
B = A
B = 10 ga teng qiymatni qabul qiladi.

4.6.2. Shartli kompilyatsiya

Si tilida shartli kompilyatsiyalash quyidagi direktivalar yordamida amalga oshiriladi:

#if butun_o'zgarmasli_ifoda
#ifdef identifikator
#ifndef identifikator
#else
#endif
#elif

Birinchi uchta direktiva shartni tekshiradi, keyingi ikkitasi esa shartning harakat ko'lamini aniqlaydi.

Shartli kompilyatsiyalash direktivalarini qo'llashning umumiy tuzilishi quyidagicha:

#if...
1-matn
#else
2-matn
#endif

Shartli kompilyatsiyalash tashqi qurilmalar uchun dasturlar yozishda, sozlanayotgan chop qilishni ajratish va shunga o'xshash holatlarda ishlatiladi.

#ifdef identifikator direktivasi *#define* yordamida identifikator aniqlangan bo'lsa, u holda 1- kompilyator 1-matndan foydalanadi.

Quyidagi:

#ifndef identifikator direktivasi *#define* yordamida esa teskari shart — identifikatorning aniqlanmaganligi ya'ni identifikator #define funksiyasi foydalanilmaganligi yoki

aniqlash #undef buyrug'i bilan bekor qilinganligi tekshiriladi.

Dasturni sozlashda nazorat qiluvchi axborot vositalarini yoki chiqarib tashlash uchun shartli kompilyatsiyani qo'shish qulay.

Masalan:

```
#define DEBUG
....
#ifdef DEBUG
printf («Sozlashda chop qilish»);
#endif
```

Multitarmoqlanishlarni amalga oshirish uchun:

#elif *butun_o'zgarmasli_ifoda*

direktivasi kiritilgan.

Bu yerda *butun_o'zgarmasli_ifodasiga* qo'yiladigan talablar #if direktivasida keltirilganiga o'xshash bo'ladi.

Bu direktivani qo'llash strukturasi bilan berilgan matn quyidagicha tus oladi:

```
#if shart
if_uchun_matn
#elif ifoda_1
Matn_1
#elif ifoda_2
Matn_2
#else
Matn_3
Else_uchun_main
# endif
```

Shunday qilib matnning faqat shartli direktivalar tomonidan ajratilgan qismi ishlatiladi.

Qatorlarni raqamlash uchun:

line o'zgarmas

direktivasi kompilyatorga keyingi qator o'nli butun o'zgarmas bilan aniqlanadigan raqamga ega bo'lishini ko'rsatadi. Direktiva bir vaqtning o'zida nafaqat qator raqamini, balki fayl ismini ham o'zgartirish imkonini beradi. Uning umumiy shakli:

```
# line o'zgarmas «fayl ismi».
Masalan AAA.S  dastur matni;
# define  N 3
voidmain ()
{
# line 23 «files.c»
double  z(3*n);
}
```

protsessor oldi ishlovidan keyin "aaa.i" ismli faylda quyidagi qator hosil bo'ladi:

```
aaa.c 1;
aaa.c  2;
aaa.c  3;{
aaa.c  4;
file.c  23;  double z(3*3)
file.c  24;}
```

Quyidagi direktiva:

error leksem ketma-ketligi xatoliklarning oldini olish xabarlarini yetkazadi.

Masalan:

```
# define aa 5
# if (aa ! =5)
# error  aa  5 ga teng bo'lishi kerak!
```

integrallashgan muhitda (masalan TURBO C da) axborot quyidagi ko'rinishga ega bo'ladi:

```
Fatal <fayl ismi><qator raqami>;
Error directiv: AA 5 ga teng bo'lishi kerak!
```

4.6.3. Bo'sh direktiva

Si tilida hech qanday harakatni ifodalamaydigan direktiva mavjud bo'lib, u quyidagi ko'rinishga ega:

```
#
dastur buyruqlari;
```

pragma *leksem_ketma-ketligi*

Mazkur buyruq kompilyator bilan bog'liq aniq harakatni ijro etishni bildiradi.

Bu direktiva tarkibiga ayrim kompilyatorlar dastur matni tarkibida mavjud bo'lgan assembler buyruqlari soni to'g'risidagi variant ham kiradi.

Bu yerda # pragma buyrug'i muhimdir va u turlicha bo'lishi mumkin. Uning uchun standart yo'q. Agar aniq bir protsessor oldi ishlovi unga notanish dasturni uchratsa, uni bo'sh direktiva singari bekor qiladi.

Ayrim kompilyatorlarda

pragma pack(n)

n bu yerda, 1,2 yoki 4 bo'lishi mumkin.

Bunda tuzilmalardagi va birlashmalardagi pack dasturi qo'shma elementlarga ta'sir etadi.

Oldindan aniqlangan makroismlar mavjud bo'lib, ular quyidagi xabarlarni olish imkonini beradi:

_ _ LINE _ _ — o'nli o'zgarmas — Si programasi bilan hozirgi ishlanayotgan qator raqami.

_ _ FILE _ _ — kompilyatsiya qilinayotgan fayl nomi.

_ _ DATE_ _ — «oy : kun : yil» shaklidagi belgilar qatori.

_ _ TIME _ _ «soat: minut: sekund» belgilari qatori — ishlash vaqtini aniqlaydi.

4.7. XOTIRA MODELI

Si dasturlash tili beshta turdagi xotira modeliga ega. Bu modellar uchun quyidagi cheklovlar o'rinli:

— Hech qaysi takrorlanuvchi modul 64 kb dan yuqori hajmni egallamaydi.

— Agar juda katta xotira modeli qo'llanilmasa va ma'lumot elementi (masalan masiv) huge kalit so'zi bilan yozilmasa, u 64kb dan ortiq bo'la olmaydi.

4.7.1. Model turlari

Kichik (smail) model

Bunda dastur bir segment ma'lumoti va bir buyruq segmenti bilan yaratiladi (segment 64 kb.ga teng). Bundan kelib chiqadiki, bu modellarda dastur 128 kb dan oshmaydi. Ushbu modelda hech qanday kalitlar ishlatilmasa, hamma ko'rsatkichlar near tipiga ega bo'ladi, lekin far,

huge kalit so'zlari bilan ham kiritishni amalga oshirish mumkin.

O'rta (meditum) model

Dastur buyruqlarning bir necha segmenti va bitta ma'lumot segmenti bilan yaratiladi. Odatda dasturlar uchun buyruqlarning katta o'lchami qo'llaniladi. Bunda har bir bajariluvchi modul o'zining buyruq segmentiga ega bo'ladi.

Kompakt (compact) model

Dastur bir necha ma'lumot segmentlari va bitta buyruq segmenti bilan yaratiladi. Oldindan ko'rsatilmasa hamma kod ko'rsatkichlari bu modelda near tipiga, ma'lumotalar esa far tipiga ega bo'ladi. Boshqa kod ko'rsatichlarini far kalit so'zi yordamida va ma'lumotlarni near va huge kalit so'zlari yordamida ham kiritish mumkin.

Katta (large) model

Dastur bir necha ma'lumotlar segmenti va bir necha buyruqlar segmenti yordamida yaratiladi. Avvaldan ko'rsatilmasa bu modelda element kodi va ma'lumotlar far tipidagi adresga ega bo'ladi. Boshqa kod ko'rsatkichlarini near va huge kalit so'zi yordamida ham kiritish mumkin.

Juda katta (huge) model

Dastur bir necha buyruq va ma'lumot segmentlari bilan to'ldiriladi. Bunday segmentlar bir qancha bo'lishi mumkin. Bu modelda ma'lumot o'lchamiga (masalan massivga) 64 kb. li cheklov olib tashlanadi, ammo segmentlarga quyidagi cheklovlar qoladi:

— ma'lumotning yagona elementi (masalan massiv elementi) 64 kb dan yuqori bo'lmasligi;

— ixtiyoriy 128 kb. dan yuqori massivlar uchun hamma elementlar 2 ga karrali o'lchamga ega bo'lishi kerak.

Sizeof opretorining bajarilish natijasi va ikki ko'rsatkichning farqi int tipiga ega bo'ladi, buning uchun sizeof operatoridan quyidagi konstruksiyada foydalanish kerak:

(long)sizeof(huge_item),

Bunda *huge_item* talab qilingan element.

Ikkita ko'rsatkichning farqini hugeda olish uchun, elementlarni quyidagi konstruksiyada olish kerak:

(long) (huge_ptr1-huge_ptr2).

4.7.2.Turbo Si xotira modeli

Turbo-Si dasturlash tili yuqorida ko'rsatilgan modellardan tashqari yana bir xotira modeli (ting)ga ega. Bu eng kichik xotira modelidir. Bunda hamma to'rta *(CS,DS,SS,ES)* registrlar bitta yagona adresni saqlaydi, shuning uchun 64 kb ma'lumot kodi massivlarni saqlashga ajratiladi. Hamisha near tipidagi ko'rsatkichlar qo'llaniladi. Tiny tipidagi xotira modelidan foydalanuvchi dastur *.com tipidagi formatga ega bo'ladi.

Elementlar yoki ko'rsatkichlarni elementga near, far va huge kalit so'zlari yordamida modifikatsiyalash uchun quyidagi qoidalarni doimo esda tutish zarur:

1. Kalit so'zlar element yoki ko'rsatkichni o'ngdan modifikatsiyalaydi. Masalan, *char far ** p*; shuni bildiradiki, *r*— bu far ga ko'rsatkich char ga ko'rsatkichdir;

2. Agar o'ngda kalit so'zidan keyin identifikator kelsa, u holda bu element standart ma'lumot segmenti nearda yoki far yoxud hugeda joylashishini aniqlaydi. Masalan, *char far A*; bu yerda *A char* tipidagi *far* adresli elementlarga ega;

3. Agar o'ngda kalit so'zdan keyin ko'rsatkich kelsa, u holda u adres o'lchami: 16 bit near yoki 32 bit (*huge* yo *far*)ni aniqlamaydi. Masalan *char far*p*. Bu yerda *r char* tipidagi element *far* ko'rsatkichini (32 bit) bildiradi;

Misol:

Char a [3000] — a standart ma'lumot segmentida joylashadi.

Near va far lardan funksiyalarda foydalanish aynan ma' lumotlar uchun qo'llash qoidalariga o'xshamaydi, lekin *huge* kalit so'zini funksiyalarga qo'llab bo'lmaydi.

Misol:

Char far fun();

Char far fun() {z..*z}* — bu yerda *fun(), char* tipidagi elementni qaytaruvchi va 32 bitni adresdan chaqiriluvchi funksiyalardir.

4.8. FAYLLAR USTIDA OPERATSIYALAR

Fayllardan matnlarni o'z ichiga olish uchun #include direktivasi qo'llaniladi va uni yozishning quyidagi uch shakli mavjud:

#include <fayl_nomi>
#include «fayl_nomi»
#include makros_ismi

Shu vaqtgacha biz misollarda birinchi shaklni qo'lladik. Fayl nomi <> belgisi ichiga olinsa, protsessor faylni katalogning standart tizimidan qidiradi. Agar fayl nomi «» ichiga olinsa, protsessor foydalanuvchining hozirgi katalogini, so'ngra katalogning standart tizimini qidiradi.

Makros_ismi — bu #define direktivasi orqali kiritilgan ism yoki makroslardan iborat.

Foydalanuvchi Si tili bilan ish boshlashda kiritish-chiqarish vositalari bilan to'qnash keladi. Buning uchun dastur matnida quyidagicha direktiva joylashtirilar edi:

#include <stdio.h>

Bu direktivaning bajarilishida dastur kutubxonadan kiritish va chiqarishning aloqalarini ulaydi. Bunda .h kutubxona fayllari tirnsolini bildiradi.

TURBO C da include fayllarining ayrimlari quyidagilar:

alloc.h — xotirani boshqarish funksiyasini e'lon qiladi (bo'lish, joylashtirish va hokazolar).

asert.h — sozlovchi makrobuyruq assert ni e'lon qiladi.

bios.h — IBM—PC ROM BIOS protsedurasini chaqirishda turli funksiyalarni e'lon qiladi.

conio.h — DOS tizimida konsolga kiritish-chiqarishda turli funksiyalarni e'lon qiladi.

ctype.h — makrobuyruqlarda foydalaniladigan belgilarni o'zgartirish va sinflarga ajratish strukturasini saqlaydi.

dir.h — kataloglar bilan ishlash funksiyalarini ishlash protseduralarini o'z ichiga oladi.

float.h — suzuvchi vergulli ma'lumotlar bilan operatsiyalar bajarganda protseduralar uchun saqlaydi.

math.h — matematik funksiyalar timsollari kutubxonasini o'z ichiga oladi.

stddef.h — ayrim ma'lumotlarning umumiy tiplari makrobuyruqlarini o'z ichiga oladi.

stdio.h — kiritish chiqarish protseduralarini o'z ichiga oladi.

stdlib.h — almashtirish protseduralari, izlash protsedurasi, tartiblash protsedurasi va shu kabilarni o'z ichiga oladi.

string.h — belgilar qatori bilan ishlash.

time.h — vaqtni belgilash uchun.

graphics.h — grafiklar bilan ishlaydi.

4.9. FUNKSIYALAR GRAFIKLARINI HOSIL QILISH

Ranglar bilan ishlash uchun Si tilida turli funksiyalar mavjud bo'lib, ular quyidagilardan iborat:

Setsolog(rang) — bu funksiya ko'rsatilgan chizish rangini o'rnatadi;

GetColor — chizish rangini qaytaradi;

GetBkColor — fon rangini qaytaradi;

GetMaxColor — ranglar sonini qaytaradi;

Ranglarning nomi va ularning belgilanishi 10-jadvalda keltirilgan.

10-jadval

Ranglar va ularning belgilanishi.

Raqamlanishi	Belgilanishi	Rang nomi
0	Black	Qora
1	Blue	Ko'k
2	Green	Yashil
3	Cyan	Feruza (bargi karam)
4	Red	Qizil
5	Magenta	Qirmizi
6	Brown	Jigarrang
7	LIGHTGRAY	Och kulrang
8	DARKGRAY	To q kulrang
9	LIGHTBBLUE	Havorang
10	LIGHTGREEN	Och yashil
11	LIGHTCYAN	Och feruzarang
12	LIGHTRED	Och qizil
13	LIGHTMA-CENTA	Och qirmizi
14	MELLOW	Sariq
15	WHITP	Oq

Masalan:
SetBkColor (GREEN) — ekranga yashil fon beradi.

4.9.1. Nuqta tasvirini hosil qilish

Nuqtani tasvirlash uchun putpixel operatoridan foydalanamiz.

Uning formati quyidagicha:

putpixel(x,y, rang) — x,y lar nuqtaning koordinatlari, rang — esa uning rangi.

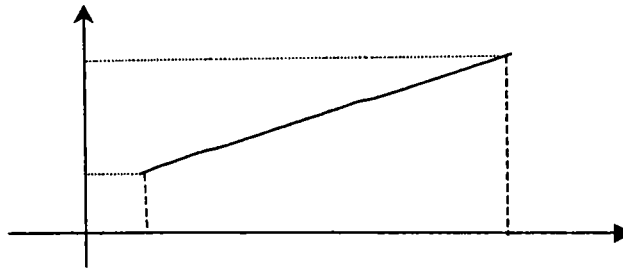
Masalan: (240,220) koordinatali nuqtaning tasvirini qizil rangda yasaylik:

```
#ihclude<conio.h>
# include<graphic.s.h>
rnain()
{ float x,y;
int gd=DETECT,gm; initgraph(&gd,&gm, «\\tc»);
putpixel(240,220,RED);
getch();
```

4.9.2. To'g'ri chiziq tasvirini hosil qilish

To'g'ri chiziq (to'g'rirogi kesma)ni yasash uchun line funksiyasidan foydalaniladi. Uning formati quydagicha :

line($x1$,$y1$,$x2$,$y2$)



10-расм. Кесма чизиш.

Bu yerda: $x1$, $y1$ — kesmaning boshlang'ich nuqtasining koordinatalari,

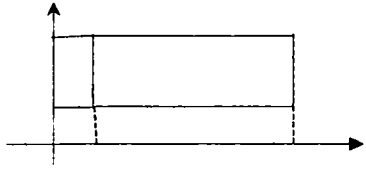
$x2$,$y2$ — kesma oxirining koordinatalari.

Masalan:

line (100,29,80,15) — ko'rsatilgan (100,29) va (80,15) nuqtalar orasida kesmani yasaydi.

Rectangle($x1,y1,x2,y2$) dioganal bo'yicha ($x1,y1$) va ($x2,y2$) koordinatalarda yotgan to'g'ri to'rtburchakni yasaydi (11-rasm).

bar($x1,y1,x2,y2$) — uchlari diagonal bo'yicha ($x1,y1,$) va ($x2,y2$) — koordinatalarda bo'lgan bo'yalgan to'g'ri to'rtburchakni yasaydi (12-rasm).



11-rasm. To'g'ri to'rtburchak chizish.

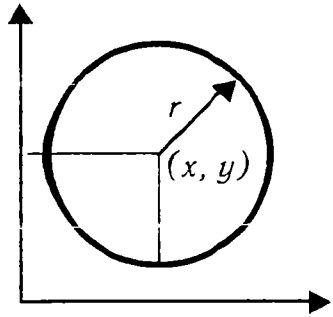
12-rasm. Bo'yalgan to'g'ri to'rtburchak chizish.

4.9.3. Aylana va yoy tasvirini hosil qilish

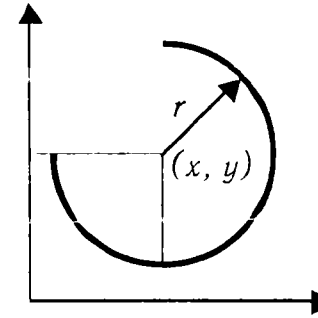
Circle—funksiyasi yordamida aylanalarni yasash mumkin. Uning formati:

Circle (x,y,r) — bunda *x, y* aylana markazining koordinatalari, *r* — radiusi (13-rasm).

Circle (x,y,r) — yordamida yoy yasaladi bunda *x, y* — yoy markazining koordinatalari, α, — yoyning boshlang'ich burchagi, β– oxirgi burchagi va *r* — *yoyning radiusi* (14-rasm).



13-rasm. Aylana chizish.



14-rasm. Yoy chizish.

Ellipsni hosilqilish uchun Ellipse funksiyasidan foydalanamiz. Uning formati.

Ellipse $(x,y,\alpha,\beta,x_r,y_r)$ — bunda (x,y) — ellipsning markazi koordinatalari, α — boshlang'ich burchak, β — oxirgi burchak, x_r — obsissalar o'qi bo'ylab radiusi, y_r — kordinatalar o'qi bo'ylab radiusi.

Soha chegaralarini bo'yash.

Ko'rsatilgan soha chegarasini bo'yash uchun

floodfill (x,y, rang chegarasi)

— funksiyadan foydalaniladi, u *setfillstyle* bilan birgalikda ishlaydi.

void sefillstyle(int bo'yash turi, int rang)

Bu funksiya shablon turini va rangini tanlash imkonini beradi. To'ldirishning 12 xil turi bo'lib, u quyidagi 11-jadvalda keltirilgan.

11-jadval

Ismi	Qiymati
EMPTY_FILL	0 (soha rangi bilan to'ldiriladi)
SOLID_FILL	1 (to'ldiruvchi rang bilan butunicha bo'yash)
LINE_FILL	2 (gorizontal-chiziqlar bilan to'ldiriladi)
LTSLAHS_FILL	3 (og'ma chiziqlar bilan to'ldiriladi)
SLAHS_FILL	4 (yo'g'on og'ma chiziqlar bilan to'ldiriladi)
BKSLAHS_FILL	5 (teskari yo'g'on og'ma chiziqlar bilan to'ldiriladi)
LTBKSLASH_FILL	6 (teskari ingichka og'ma chiziqlar bilan to'ldiriladi)
HATCH_FILL	7 (to'rtburchakli gorizontal shtrixlar bilan to'ldiriladi)
XHATCH_FILL	8 (egri shtrixlar bilan to'ldiriladi)
INTERLEAVE_FILL	9 (bekiluvchi egri shtrixlar bilan to'ldiriladi)
WIDE_DOT_FILL	10 (siyrak nuqtalar bilan to'ldiriladi)
CLOSE_DOT_FILL	11 (o'zaro yakin joylashgan nuqtalar bilan to'ldiriladi)

Si dahar-xil yo'g'onlikdagi chiziqlarni selinestyle funksiyasi yordamida yasash mumkin.

void selinestyle(int linesigned, uppatern, int thickness);

linestyle(" chiziq to'g'ri") quyidagi 5 ta qiymatdan birini qabul qilishi mumkin.

Raqamlanish	Belgili konstanta	Chiziq turi
1	SOLID_LINE	Uzluksiz
2	DOTER_LINE	Uzuq-uzuq
3	CENTER_LINE	Markaziy
4	DAHSED_LINE	Shtrixli
5	USRBIT_LINE	Foydalanuvchi shabloni

Ikki argument upatter («foydaluvchi shabloni») birinchi argument USERBIT_LINE bo'lgan holdagina foydalaniladi.

Ikkinchi argument thickness chiziqning yo'g'onligi haqidagi axborotni bildiradi. Afsuski, uning faqatgina quyidagi ikki ko'rinishi mavjud xolos:

Raqamlanish	Belgili konstanta	Izoh
1	NORM_WIDTH	eni bir nuqta qalinligida
3	THIC_WIDTH	eni uch nuqta qalinligida

Matnlarni joylashtirish uchun outtext (qator) funksiyasidan foydalanib, qatorni ekranga chiqarish mumkin.

Masalan:

Outtextxy (x, y, «NaMPI») — NaMPi qatori ekranga chikadi

Outtextxy (x, y, «matn») — kerakli axborotni ko'rsatilgan koordinatadan boshlab ekranga chiqarish.

Masalan:

Outtextxy (50, 50, «O'zbekiston kelajagi buyuk davlat»)

textwidth (qator) — qator uzunligini qaytaradi.

textheight (qator) — qator bo'yini qaytaradi.
settextstyle (shrift, yo'nalish, o'lcham)
void far settextstyle (int shrift,int yo'nalish, int o'lchami)
font — o'lchamni quyidagi qiymatlardan birini qabul qiladi:

Raqamlanishi	Belgili konstanata	Yo'nalishi
0	DEFALT_FONT	Umumiy holda
1	DEFALT_FONT	Tripleks
2	DEFALT_FONT	Kichik
3	DEFALT_FONT	Sanserif
4	DEFALT_FONT	Gotik

Ikki o'zgaruvchi direction yo'nalishi quyidagi qiymatlarni qabul qiladi:

Raqamlanishi	Belgili konstanata	Yo'nalishi
0	HORIZ_DIR	Gorizontal
1	VERT_DIR	Vertikal

Uchunchi o'lcham charsize belgi o'lchamini aniqlaydi va 1,2,..,10 qiymatlarini qabul qiladi.

charsize($1 \leqslant i \leqslant 10$)

void far setjistify (int horiz, int xert)

funksiyasi yordamida matnni istalgan joyga joylashtirish mumkin.

Uning gorizontal, vertikal bo'yicha qiymatlari quyidagi 11-jadvalda joylashtirilgan.

11-jadval

Qiymati	Belgili konstanata	Mo'ljallanishi
0	LEFT_TEXT,BOTTOM_TEXT	o'ngdan,quyidan
1	CENTER_TEXT	O'rtasidan
2	RIGHT_TEXT,TOP_TEXT	O'ngdan, yuqoridan

Closegraph()-funksiyasi grafik rejimidan chiqish uchun ishlatiladi.

FOYDALANILGAN ADABIYOT

1. *A. Siddiqov* Sonli usullar va dasturlashtirish. T. : Mehnat. 2002 y.
2. *A. Sattarov, Qurmonboyev.* Informatika va hisoblash texnikasi asoslari. T. O'qituvchi, 1996 y.
3. *Grexem R.* Prakticheskiy kurs yazika Paskal dlya mikroEVM. M.: Radio i svyaz, 1986.
4. *Abramov V., Trifonov N., Trifonova G.* Vvedeniye v yazik Paskal. M.: Nauka, 1988.
5. *Faysman A.* Professionalnoye programmirovaniye na Turbo Paskale. T.: «Infomeks Korporeyshn», 1992.
6. *Findlay W., Watt D. A., Paskal.* An introduction to methodical programmin. Third edition. London: Pitman, 1985.
7. *Kernigan, D. Ritchi.* «Yazik programmirovaniya Si» M.: Finansi i statistika. 1992 g.
8. Rukovodstvo polzovatelya po yaziku Si «Opisaniye yazika Si» 1, 2 knigi.
9. *D. Maslova.* «Vvedeniye v yazik Si». M., 1991 g.
10. *L. Ameral.* «Programmirovaniye grafiki na TURBO C». M.: «Sol sistem», 1992 g.
11. *Virt N.* Algoritmi strukturi dannix programmi. M.: Mir, 1985.

MUNDARIJA

P. Karimov, S. Irisqulov, A. Isabayev

DASTURLASH

Kasb-hunar kollejlari uchun o'quv qo'llanmasi

Toshkent — „O'ZBEKISTON" 2003

Badiiy muharrir *U. Solihov*
Tex. muharrir *T. Xaritonova*
Musahhih *Sh. Moqsudova*

Terishga berildi 18.02.03. Bosishga ruxsat etildi 07.04.03 Bichimi $84\times108^{1}/_{32}$. Shartli bosma taboq 10,92. Nashr taboq 10,95. Adadi 6000 nusxa. Buyurtma № 18. Bahosi shartnoma asosida.

Toshkent, 700129 ,,O'zbekiston" nashriyoti, Navoiy ko'chasi 30.
Nashr №18-2003.

O'zbekiston Matbuot va axborot agentligining Toshkent kitob-jurnal fabrikasida chop etildi.
700194, Yunusobod dahasi, Murodov ko'chasi, 1-uy.

K 25

Karimov N. va b. Dasturlash. Kasb-hunar kollejlari uchun o'quv qo'llanma. — T.: "O'zbekiston", 2003. 208 b.
I. Muallifdosh.

ISBN 5-640-03173-5

BBK 22. 18ya73132.973-01

D $\frac{2404010000-37}{M\ 351(04)\ 2003}$ 2003