N. A. Otaxanov

"Programmalash asoslari"

fani bo'yicha

M'ARUZALAR MATNI

I-kurs "Amaliy matematika" mutaxassisliklari uchun

Namangan-2005 yil

§-1. MATEMATIK MODELLASHTIRISHNING ASOSIY TUSHUNCHALARI. MASALALARNI EHMDA YECHISH BOSQICHLARI

1.1. **KIRISH**

Qadim zamonlardan beri inson o‘z imkoniyatlarini kengaytirishga xarakat qilib, turli mehnat qurollarini yaratib kelgan. Masalan, uzoqni ko‘rolmaslikni mikroskop, teleskop, radiolokator kabi buyumlarni yaratish bilan qoplagan bo‘lsa, bir-biriga ma’lumotlar uzatishdagi cheklangan imkoniyatlarini telefon, radio va televideniya xisobiga kengaytirmoqda. Elektron hisoblash mashinalarining yaratilishi va ularning keskin rivojlanib borishi inson ongining imkoniyatlarini to‘ldiribgina qolmay, uning turli-tuman ma’lumotlarni tahlil qilish va o‘zining ish faoliyatida uchraydigan masalalar yechimini topish tezligini ham jadal sur’atda o‘stiradi.

Shunday qilib, fan va texnikaning rivojlanishi va o‘ta murakkab jarayonlarning hisob ishlarini sifatli va tez bajarilishini talab etayotgan bir paytda,- yuqori texnologiyali elektron hisoblash mashinalarining ishlab chiqilishi tabiiy bir holdir. XXI asr - kompyuterlashtirish asrida insoniyat faoliyatining barcha jabhalariga kompyuterlar jadal sur’atda kirib bormoqda.

Zamonaviy kompyuterlarning ko‘payib borishi esa tabiiy ravishda undan foydalanuvchilarning safini ortib borishiga turtki bo‘ladi. Odatda kompyuterdan foydalanuvchilar sinfi juda ham xilma-xildir. Lekin, umumiy qilib ularni kompyuterlardan o‘z ishlarini bajarishda tayyor vosita sifatida foydalanuvchi-operatorlar sinfi va ular uchun zarur bo‘lgan dasturiy ta’minotlarni yaratuvchi dasturchilar sinfiga ajratish mumkin. Dasturchilar sinfini esa o‘z navbatida shartli ravishda sistemali va amaliy dasturchilar guruhlariga ajratamiz.

Mazkur «Programmalash asoslari» fanidan yozilgan ma’ruzalar matni amaliy dasturchilar guruhiga tegishli mutaxassislarni, institutning «Informatika va axborotlar texnologiyasi» kafedrasida «Informatika va AT» yo‘nalishi bo‘yicha ta’lim olayotgan bakalavriat va magistratura talabalari, dastur tuzishni mustaqil o‘rganmoqchi bo‘lgan o‘quvchilar ommasiga mo‘ljallangan.

1.2. MATEMATIK MODEL TUSHUNCHASI

Elektron hisoblash mashinalari uchun dastur yozishni o‘rganishdan avval nimalarni bilish kerakligini ko‘rib chiqaylik. Istalgan xayotiy, matematik yoki fizik va xokazo masala shartlarini ifoda qilish dastlabki ma’lumotlar va fikrlarni tasvirlashdan boshlanadi va ular qat’iy ta’riflangan matematik yoki fizik va xokazo tushunchalar tilida bayon qilinadi. So‘ngra masalani Yechishning maqsadi, ya’ni masalani Yechish natijasida ayni nimani yoki nimalarni aniqlash zarurligi

ko‘rsatiladi. Masalani o‘rganish uning matematik modelini tuzishdan boshlanadi, ya’ni uning o‘ziga xos asosiy xususiyatlari ajratiladi va ular o‘rtasidagi matematik munosabat o‘rnatiladi. Boshqacha qilib aytganda, dastlab o‘rganilayotgan fizik xodisaning mohiyati, belgilari, ishlatiladigan ko‘rsatkichlar so‘zlar yordamida batafsil ifoda etiladi, so‘ngra extiyojga qarab kerakli matematik tenglamalar keltirilib chiqariladi. Bu tenglamalar o‘rganilayotgan fizik jarayon yoki xodisalarning matematik modeli deb ataladi. Matematik modelni haqiqiy ob’ektga moslik darajasi amaliyotda tajriba orkali tekshiriladi. Odatda, matematik model qaralayotgan ob’ektning hususiyatlarini aynan, to‘la o‘zida mujassam qilmaydi. U har xil faraz va cheklanishlar asosida tuzilgani uchun taqribiylik xarakteriga ega, tabiiyki uning asosida olinayotgan natijalar xam taqribiy bo‘ladi. SHuning uchun, tajriba qilib ko‘rish orqali yaratilgan modelni baholash va lozim bo‘lgan holda uni aniqlashtirish imkoniyati yaratiladi.

Matematik modelning aniqligi, uning korrekt qo‘yilganligi, olinadigan natijalarning ishonchlilik va turg‘unlik darajasini baxolash masalasi modellashtirishning asosiy masalalaridan biridir.

Matematik modellarni shartli ravishda quyidagi turlarga ajratish mumkin.

1.3. STATSIONAR VA NOSTATSIONAR MODELLAR

Bu modellarda qaralayotgan jarayon vaqt bo‘yicha turg‘unlashgan deb qaraladi, ya’ni matematik modelni ifodalovchi tenglamalarda vaqtni ifodalovchi ko‘rsatkichi qatnashmaydi. Modelda qatnashuvchi ko‘rsatkichlar, parametrlarning bir qismi yoki barchasi faqat fazoviy o‘lchovlarga bog‘liq bo‘ladi. Bunday modellarga misol qilib inshoot devoridan o‘tuvchi stasionar issiqlik oqimi tenglamasi, qurilish to‘sinlarining stasionar egilishi va buralishi tenglamalarini keltirish mumkin. Stasionar modellar algebraik tenglamalar, oddiy differensial tenglamalar yoki ularning sistemasi kabi ifodalanadi.

Bu modellarda jarayon ko‘rsatkichlari vaqtga bog‘liq deb qaraladi. Umumiy holda esa, bu ko‘rsatkichlar fazoviy o‘lchovlarga xam bog‘liq bo‘lishi mumkin. Bunday modellarga qurilish inshootlarida nostasionar issiqlik oqimi tenglamalari, tebranish jarayonlarining tenglamalari, diffuziya tenglamalarini misol qilib ko‘rsatish mumkin. Nostasionar jarayon o‘zi va xosilalari vaqtga bog‘liq funksiya qatnashgan differensial tenglama yoki shunday tenglamalar sistemasi, hususiy xosilali differensial tenglamalar yordamida yoziladi.

1.4. PARAMETRLARI TO‘PLANGAN MODELLAR VA PARAMETRLARI TARQOQ MODELLAR

Bunday modellarda jarayon ko‘rsatkichlari fazoviy o‘lchovlar bo‘yicha o‘rnatiladi. Natijada model ko‘rsatkichlari faqat vaqtga bog‘liq bo‘ladi. Bu jihatdan

parametrlari to‘plangan modellar fazoviy o‘lchovga bog‘liq bo‘lmagan nostasionar modellarga o‘xshashdir. Modellar chiziqli va chiziqli bo‘lmagan algebraik, chiziqsiz tenglamalar, vaqt buyicha xosilalar qatnashuvchi oddiy differensial tenglamalar yoki shunday tenglamalar sistemasi kabi tenglamalar bilan ifodalanadi.

Bunday modellarda umuman olganda qaralayotgan jarayon ko‘rsatkichlari xam vaqtga, xam fazoviy o‘lchovlarga bog‘liq bo‘ladi. Modellar asosan xususiy xosilali differensial tenglamalar yordamida ifodalanadi. Hususiy holda, modellar vaqtga bog‘liq bo‘lsa, ular stasionar modellar bilan bir xil bo‘ladi. Lekin, parametrlari tarqoq modellarning mazkur guruhga kiritilishida ularda qatnashuvchi ko‘rsatkichlarning fazoviy o‘lchovlarga bog‘liqligi belgilovchi omil bo‘lgan bo‘lsa, stasionar modellarning alohida guruhga birlashtirilishida asosiy omil - ulardagi ko‘rsatkichlarining vaqtga bog‘liq emasligidir.

Yuqorida keltirilgan tavsif ma’lum darajada shartlidir. Matematik modellarning boshqa ko‘rinishdagi tavsiflari xam berilishi mumkin. Masalan, ularni chiziqli va chiziqli bo‘lmagan, bir o‘lchamli va ko‘p o‘lchamli kabi guruhlarga ajratish mumkin. SHuni ta’kidlash lozimki, xar doim ham qo‘yilgan masalaning matematik modelini yaratib bo‘lavermaydi.

1.5. MASALALARNI EHMDA YECHISH BOSQICHLARI

Matematik model xar xil vositalar yordamida berilishi mumkin. Bu vositalar fizik qonuniyatlar hamda funksional analiz elementlarini ishlatib differensial va integral tenglamalar tuzishdan to hisoblash algoritmi va EXM dasturlarini yozishgacha bo‘lgan bosqichlarni o‘z ichiga oladi. Har xil bosqich yakuniy natijasiga ko‘ra o‘ziga xos ta’sir ko‘rsatadi va ulardagi yo‘l qo‘yiladigan xatoliklar oldingi bosqichlardagi xatoliklar bilan ham belgilanadi.

Ob’ektning matematik modelini tuzish, uni EHM da bajariladigan hisoblashlar asosida tahlil qilish - hisoblash tajribasi deyiladi. Hisoblash tajribasining umumiy sxemasi quyidagi bosqichlar orqali amalga oshiriladi:

1. Masalaning qo‘yilishi va tahlil.
2. Masalaning matematik modelini yaratish.
3. Hisoblash algoritmini qurish.
4. Dasturiy ta’minot yaratish va uni EHM ga tatbiq etish.
5. EHM da olingan natijalarni tahlil qilish.

Birinchi bosqichda masalaning aniq qo‘yilishi, berilgan va izlanuvchi miqdorlar, ob’ektning matematik modelini tuzish uchun ishlatish lozim bo‘lgan boshqa hususiyatlari tasvirlanadi.

Ikkinchi bosqichda fizik, mexanik, kimyoviy va boshqa qonuniyatlar asosida matematik modeљ tuziladi. U asosan algebraik, differensial, integral, integro-

differensial va boshka turdagi tenglamalardan iborat bo‘ladi. Ularni tuzishda o‘rganilayotgan jarayonga ta’sir ko‘rsatuvchi omillarning barchasini bir vaqtning o‘zida hisobga olib bo‘lmaydi, chunki, matematik modelь juda murakkablashib ketadi. SHuning uchun, modelь tuzishda qaralayotgan jarayonga eng kuchli ta’sir etuvchi asosiy omillargina hisobga olinadi.

Masalaning matematik modeli yaratilgandan so‘ng, uni Yechish usuli izlana boshlanadi, ya’ni, mos tenglamalar echilishi va kerakli ko‘rsatkichlar aniqlanishi lozim. Ayrim hollarda masalaning 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 EXMda Yechishning uchinchi bosqichini tashkil qiladi.

Navbatdagi bosqichda, ya’ni, to‘rtinchi bosqichda, masalani EHM dan foydalanib Yechish uchun uning Yechish algoritmi ishlab chiqiladi, hamda shu algoritm asosida biror-bir zamonaviy algoritmik tilda EXM da ishlatish uchun dastur tuziladi. Dastur ma’lum talablar asosida tuziladi. Masalan, u umumiylik xususiyatiga ega bo‘lishi kerak, ya’ni, matematik modelda ifodalangan masala parametrlarining etarlicha katta sohada o‘zgaruvchi qiymatlarida dastur ishonchli natija berishi kerak. U bir necha mustaqil qismlar (proseduralar) dan iborat bo‘lishi mumkin.

Nihoyat masalani Yechishning yakunlovchi beshinchi bosqichida yaratilgan dastur EXMga kiritiladi va sozlanadi xamda olingan natijalar chuqur taqlil qilinib, baxolanadi. Natijalarni tahlil kilish, zarur bo‘lgan xollarda algoritmni, Yechish usulini va modelni aniqlashtirishga yordam beradi, xattoki masalani noto‘g‘ri qo‘yilganligini xam baxolab berishi mumkin.

SHunday qilib, biz masalalarni EXM lar yordamida Yechish bosqichlari bilan tanishib chikdik. SHuni ta’kidlash lozimki, xar doim xam bu bosqichlar bir-biridan yaqqol ajralgan xolda bo‘lmasdan, bir-biriga ko‘shilib ketgan bo‘lishi xam mumkin.

Nazariy savollar va tayanch iboralar:

1. Masalani o‘rganish nimadan boshlanadi?
2. Matematik model deb nimaga aytiladi?
3. Stasionar misollarga misollar keltiring.
4. Qanday modellar nostasionar modellar deb ataladi?
5. Faqat vaqt faktoriga bog‘liq modellar qanday modellar deb ataladi?
6. Parametrlari tarqoq modellarni qanday tushunasiz?
7. Masalaning modelini doim ham qurish mumkinmi?
8. Hisoblash tajribasi deb nimaga aytiladi?
9. Masalani EXM yordamida Yechish necha bosqichda amalga oshiriladi?
10. Tadqiqot ob’ektini tanlash va masala shartlarini aniqlash bosqichining vazifasi

nimadan iborat?

11. Matematik model tuzish va uni asoslash bosqichining vazifalarini tushuntiring.
12. Diskret model va hisoblash algoritmi bosqichi kanday ishlarni o‘z ichiga oladi.
13. Masalani Yechishning dastur ta’minotini yaratish bosqichining vazifalarini tushuntiring.
14. Dastur natijalarini tahlil kilish va baxolash bosqichida bajariladigan ishlar ketma-ketligini ayting.
15. Masalaning noto‘g‘ri qo‘yilganligini qanday aniqlash mumkin?
16. Matematik modellarga doir misollar keltiring.

§-2. ALGORITM. ALGORITMIK TIL

2.1. ALGORITM VA UNGA QO‘YILADIGAN TALABLAR

Inson butun hayoti davomida algoritmlar ichida yashaydi, lekin buni u odatda sezmaydi. U dunyoga kelishidan tortib, to dunyodan ketishigacha bo‘lgan faoliyati davomida o‘z oldiga doim qandaydir masalalar qo‘yadi, bu masalalarni Yechishning yo‘l-yo‘riqlarini qidiradi. Natijada ma’lum bir qonun-qoidalarni o‘ylab topadi, belgilangan tartibda ularni bajarib, ko‘zlagan natijasiga erishadi. Agar ana shu qonun-qoidalarni ixtiyoriy odam ko‘rsatilgan tartibda bajarishga muvafaq bo‘lsa, u ham ana shu natijalarga erishishi, tartib buzib bajarganda esa olingan natija uni qanoatlantirmasligi mumkin.

Qandaydir maqsadga erishish yo‘lida belgilangan amallar ketma-ketligini bajarayotgan inson yoki texnik vositani ijrochi deb ataymiz.

Ta’rif: Algoritm deb qo‘yilgan masalani to‘la hal uchun ijrochining bajarishi lozim bo‘lgan amallar ketma-ketligining qat’iy tartibiga aytiladi.

1-misol: CHoy damlash algoritmi. Bu masalani hal qilish uchun keragicha qaynab turgan suv, quruq choy hamda 1 litrli choynak berilgan deb hisoblaymiz.

1. CHoynakni chaying.
2. CHoynakka 5 gramm quruq choy soling.
3. CHoynakni qaynab turgan suv bilan to‘ldiring.

4 CHoynakning qopqog‘ini yoping.

5. CHoynakni o‘t ustiga qo‘yib, 5 minut dam bering.
6. CHoy tayyor.

2-misol. Ko‘chani xavfsiz kesib o‘tish qoidasi.

1. Yo‘lning chetiga kelib to‘xtang.
2. Yo‘lning chap tomoniga qarang.
3. Agar chap tomonda transport vositalari yaqin kelib qolgan bo‘lsa, o‘tib ketguncha kuting.

4. CHap tomoningizda transport vositalari qolmagan bo‘lsa, yo‘lning o‘rtasiga o‘tib to‘xtang.
5. Yo‘lning o‘ng tomoniga qarang.
6. Agar o‘ng tomonda transport vositalari yaqin kelib qolgan bo‘lsa, o‘tib ketguncha kuting.
7. O‘ng tomoningizda transport vositalari qolmagan bo‘lsa, yo‘lning qolgan qismini kesib o‘ting.

1-misolda choy damlayotgan shaxs, 2-da esa yo‘lni kesib o‘tayotgan shaxs ijrochi hisoblanadi. SHuningdek, ixtiyoriy dorilarni tayyorlash yo‘llari, ovqatlarni tayyorlash usullari, xakim belgilagan dorilarni iste’mol qilish, bankomyotdan pul olish kabi amallarni algoritm sifatida qabul qilish mumkin. Algoritmlarga turli fan sohalaridagi masalalarni Yechish yo‘llari ham kiradi.

3-misol: $y = x^{20}$ ni 5 ta amal yordamida hisoblang.

1	$a1 := x * x$	x^2 hisoblandi
2	$a2 := a1 * a1$	x^4 hisoblandi
3	$a3 := a2 * a2$	x^8 hisoblandi
4	$a4 := a3 * a3$	x^{16} hisoblandi
5	$y := a4 * a2$	x^{20} hisoblandi

Algoritmlarga quyidagi talablar qo‘yiladi :
1.Boshlanishi va tugashi ko‘rsatilishi kerak.
2. Har qanday amal buyruq tarzida ifodalanishi shart.
3. Har bir amal ijrochiga tushunarli bo‘lgan ko‘rinishda ifodalangan bo‘lishi shart.
4. Har bir amalda qatnashayotgan o‘zgaruvchilarning qiymatlari oldindan aniqlangan bo‘lishi kerak.
5. Har qanday amal natijasi bir qiymatli bo‘lishi kerak.
6. Bajariladigan amallar soni cheklangan bo‘lishi kerak.
7. YAkuniy natijalarni ajratib ko‘rsatish va chiqarish shart.
8. Qo‘yilgan masalani to‘la yechish uchun berilgan hamma ma’lumotlar va mumkin bo‘lgan barcha imkoniyatlar hisobga olingan bo‘lishi kerak.
9. Algoritm ommaviy, ya’ni bitta sinfga taaluqli bo‘lgan ko‘plab masalalarni Yechishga mo‘ljallangan bo‘lishi kerak.

Yuqoridagi talablarning birortasi buzilgan bo‘lsa, qo‘yilgan masalani Yechish uchun qurilgan algoritm to‘laqonli bo‘la olmaydi, ya’ni masalaning to‘la echimini bera olmaydi. Masalan: Agar biron bir amalni bajarishda qatnashayotgan har bir o‘zgaruvchining qiymati oldindan aniqlanmagan (4-talab) bo‘lsa, u holda ana shu o‘zgaruvchining o‘rniga odatda nol qo‘yib hisoblanadi. Bu esa har doim ham to‘g‘ri

natija beravermaydi. Faraz qilaylik, K-o‘zgaruvchining qiymati oldindan aniqlanmagan bo‘lsin. U holda $D = (A + B)/K$ ifodaning qiymatini hisoblashning iloji yo‘q, chunki K ning o‘rniga kompilyator nol qiymatini qo‘yadi. Natijada nolga bo‘linish holati ro‘y beradi. Bunday bo‘lishi esa mumkin emas. Endi 6-talabni buzib ko‘raylik.

1. *Hisoblansin* $i := 1$;
2. *Hisoblansin* $i := i + 1$;
3. *1-ga o‘tilsin.*

Bu holda qurilgan algoritm «cheksiz algoritm» bo‘lib qoladi, yani uni «ijrochi» hech qachon tugata olmaydi.

2.2. ALGORITMLARNING BERILISH USULLARI

Algoritmlar uch hil usulda qurilishi mumkin.

A) Algoritmni so‘zlar orqali qurish. Bunda algoritmning har bir buyruq-amali ijrochiga tushunarli bo‘lgan so‘zlar orqali ifodalanadi.

1-misol: AB kesmani teng ikkiga bo‘lish algoritmi.

1. Boshlansin.
2. Sirkulning bir uchi A nuqtaga qo‘yilsin.
3. Radiusi AB bo‘lgan aylana chizilsin.
4. TSirkulning uchini V nuqtaga quyilsin.
5. Radiusi BA bo‘lgan aylana chizilsin.
6. Aylanalarning kesishish nuqtalarini CD kesma bilan birlashtirilsin.
7. CD va AB kesmalarning kesishish nuqtasi M ni belgilansin.
8. M nuqtani izlangan nuqta deb hisoblansin.
9. Ishni tugatilsin.

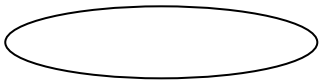
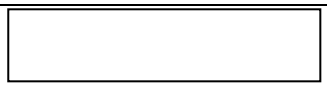
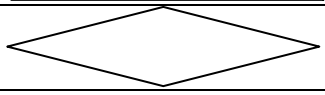
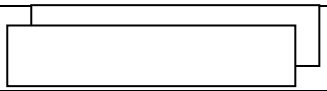
B) Matematik formulalar usuli. Bu usulda algoritmning har bir amali matematik formulalar yordamida hosil qilinadi. Algoritm amallarini ifodalashda oddiy matematik yozuvlardan foydalanish mumkin.

2-misol: $ax^2 + bx + c = 0$ kvadrat tenglamaning echimi topilsin.

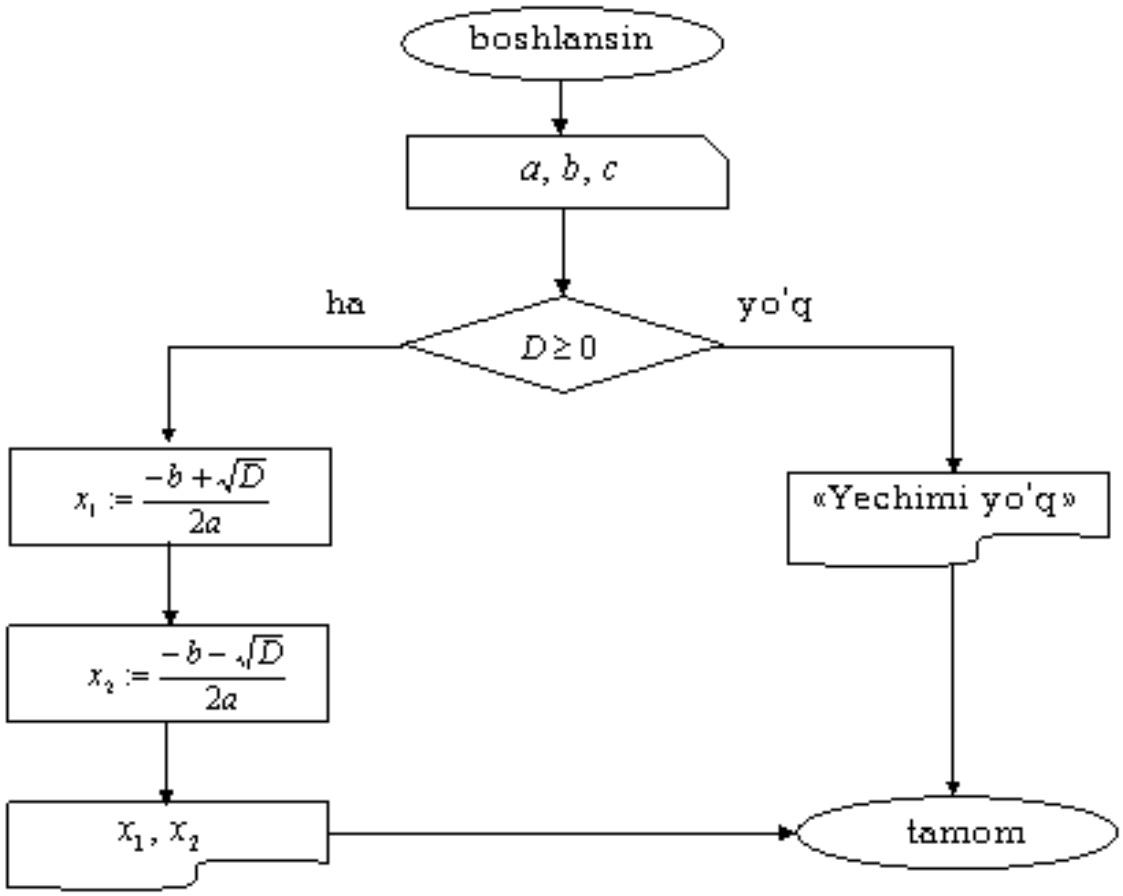
1. Boshlansin.
2. Aniqlansin a, b, c.
3. Hisoblansin $D := b^2 + 4ac$.
4. Agar D<0 bo‘lsa chiqarilsin «Echimi yo‘q» ; 7 ga o‘tilsin.
5. Hisoblansin $x_1 := \frac{-b + \sqrt{D}}{2a}; \quad x_2 := \frac{-b - \sqrt{D}}{2a}$.
6. CHiqarilsin x_1, x_2.
7. Ishni tugatilsin.

S) Blok-sxemalar usuli. Bunda algoritmning har bir buyrug‘i maxsus geometrik shakllar yordamida ifodalanadi. Blok-sxemalarni qurishda 1-jadvaldagi shakllardan foydalanish mumkin.

1-jadval

1		algoritmning boshlanishi va oxirida qo‘yiladi
2		o‘zgaruvchilarga qiymat berish
3		ma’lumotlarni kiritish
4		mantiqiy ifodalarni hisoblash
5		amallarni bajarish yo‘nalishi
6		yordamchi algoritmga murojaat
7		yakuniy natijalarni chiqarish

3-misol: $ax^2 + bx + c = 0$ kvadrat tenglama uchun blok-sxema.



1.3. ALGORITMIK TIL HAQIDA QISQA MA’LUMOT

Yuqorida uch hil ko‘rinishdagi algoritmga misollar keltirildi.
Ularning bilim darajasi o‘rtacha bo‘lgan ixtiyoriy o‘quvchi bajara oladi, CHunki bu algortimlarda uning uchun tushunarli bo‘lgan so‘zlar va formulalardan foydalanilgan. Informatika fani odatda algortimlarning ijrochisi deganda EXM ni nazarda tutadi. EXM yuqoridagi usullarda ifodalangan algortimlarni tushunmasligi mumkin. Demak biz turli masalalarning yechish algortimlarni EXM larga tushunarli ko‘rinishda ifodalash uchun qo‘shimcha vositaga muxtoj bo‘lib qoldik. SHunday vosita bo‘lib algoritmik til xizmat qiladi. Ko’pincha algoritm va algoritmik til tushunchalarini chalkashtirib qo‘yishadi yoki bir xil narsa deb qarashadi. Bu noto‘g‘ri.

Algoritmik til deb algoritmlarni ijrochiga tushunarli va bir xil ko‘rinishda ifodalash uchun zarur bo‘lgan belgilar va qonun-qoidalar majmuasiga aytiladi.

Algoritmik tillarni ko‘pincha dasturlash tillari deb xam yuritiladi. Xozirgi vaqtda zamonaviy EXM lar uchun ko‘plab dasturlash tillari ishlab chiqilgan bo‘lib, hammasining o‘ziga yarasha imkoniyatlari hamda qonun-qoidalari mavjud. BEYSIK, TURBO PASKAL, FORTRAN, SI dasturlash tillari ana shular jumlasidandir. Bu tillar imkoniyatlarining turlichaligi bilan bir-birlaridan farq qiladilar. Masalan, BEYSIK algoritmik tili o‘rganish uchun sodda va qulay bo‘lib, unchalik murakkab bo‘lmagan injenerlik masalalari uchun mo‘ljallangan. PASKAL tili esa dastur yozish jarayonida yo‘l qo‘yilishi mumkin bo‘lgan hatoliklarning oldini olish, yangi tipdagi funksiyalarni aniqlash, yangi tipdagi ma’lumotlarni hosil qilish, rekursiv funksiyalar bilan ishlash, grafik imkoniyatlarining kengligi va boshqa ko‘plab hususiyatlari bilan boshqa tillardaq farq qiladi. Bundan tashqari bu til zamonaviy DELPHI dasturlash muhitini o‘rganish uchun asosiy poydevor hisoblanadi. PASKAL dasturlash tiliga 1069 yilda TSyurixdagi informatika institutining hisoblash texnikasi bo‘yicha mutaxassisi Nikolas Virt asos solgan. 1981 yilda bu tilning halqaro standart versiyasi taklif qilingan. IBM PC kompyuterlari uchun BORLAND firmasi PASKAL tilining TURBO PASKAL versiyasi ishlab chiqildi. Hozirgacha TURBO PASKAL ning 7 dan ortiq versiyalari ishlab chiqilgan bo‘lib, ular halq xo‘jaligining turli tarmoqlari masalalarini Yechishda keng foydalanilmoqda. Jahondagi etuk dasturchilarning ko‘pchiligi shu tilda ijod qilishgan. TURBO PASKAL dasturlash tili EHM dasturiy ta’minotini hamda turli amaliy xarakterdagi dasturlar dastasini yaratishda eng ko‘p qo‘llaniladigan dasturlash tillaridan biri bo‘lib qolmoqda.

Biz ushbu qo‘llanmada TURBO PASKAL o‘rniga soddagina qilib PASKAL so‘zidan foydalanamiz.

§-3. TURBO-PASKAL MUXITIDA ISHLASH ASOSLARI

3.1. TURBO-PASKAL MUXITINI O‘RNATISH

Turbo Paskalning professional programmalash - 7 versiyasi (Turbo Passal Professional) Borland International Ins firmasining bir nechta disketalarida beriladi. SHu disketalarning biri «INSTALLG‘COMPILER» deb nomlanadi va bu disketada INSTALL.EXE programmasi mavjud.

Turbo Paskalni muhitini o‘rnatish uchun yuqorida ta’kidlangan disketani diskovodga qo‘yib, programmani ishga tushiriladi. Programma Sizdan bir necha savollarga javob berishni taklif qiladi:

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

Agar Sizda etarli asos bo‘lmasa, taklif qilingan katalog ismlari bilan chegaralangan ma’qul. Bu holda sizdan faqatgina, disk ismini o‘zgartirish talab etiladi. SHundan so‘ng, INSTALL.EXE programmasi o‘zining ishini davom ettiradi va uning so‘rovi bo‘yicha navbat bilan, ko‘rsatilgan nomli disketalarni diskovodga kuyiladi. Natijada Turbo Paskal ishga tayyor holga keladi.

Faraz qilaylik, Turbo-Paskalni urnatishda odatdagi «S» diskning o‘rniga «D» diskni tanladingiz, bu holda sistema quyidagi kataloglarda joylashdi:

$D:\TP$, $D:\TP\BGI$, $D:\TP\DEMOS$, $D:\TP\DOC$, $D:\TP\DOCDEMOS$,
$D:\TP\TURBO3$, $D:\TP\TVDEMOS$, $D:\TP\TVIZION$, $D:\TP\UTILS$

3.2. PASKAL MUXITINING ASOSIY FAYLLARI

Turbo Paskalning asosiy fayllari $D:\TP\$ katalogida joylashgan bo‘lib, ular quyidagilardir:

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

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

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

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

Tptour.exe - integrallashgan muhitda ishlashni tanishtiruvchi dastur.

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

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

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

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

3.3. TURBO-PASKAL MUHITIDA ISHLASH

Bugungi kunda kasb-xunar kollejlarida va oliy ta’lim muassasalarida Turbo Paskalning 5-7 versiyalari keng tarqalganligini va undan foydalanish qulayligini hisobga olingan xolda, quyida Turbo Paskalning shu versiyalari uchun umumiy bo‘lgan tomonlarini ko‘ramiz.

Turbo Paskal muhiti *Turbo.exe* faylidan ishga yuklanib, uning asosiy vazifasi Paskal tilidagi programmani taxrirlash va mashina kodiga o‘tkazishdan iboratdir.

Turbo Paskal muhitini yuklash natijasida bosh menyu hosil bo‘lib, uni aktivlashtirish uchun, *F10* tugmachasidan foydalaniladi.

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

File	*.pas kengaytmali fayllari va Turbo Paskalga tegishli bulgan kataloglar ustida turli amallar bajaradi;
Edit	yuqori menyudan matnlarni taxrirlash oynasiga qaytish;
Run	programmani turli ko‘rinishda ishga yuklaydi va natijalar oynasiga o‘tishni ta’minlaydi;
Compile	programmani kompilyasiya kilish va uni tekshirishni ta’minlaydi;
Options	Turbo Paskal muhitini va undagi programmalarga tegishli hususiyatlarni taxrirlaydi;
Debug	programmadagi proseduralarni izlash, hisoblash jarayonlari va o‘zgaruvchilarni tekshirish, chaqirilgan stekni ko‘rish va shunga o‘xshash amallarni bajaradi;
Break + Watsh	agarda *Debug + Integrated Debugging* ustanovkasi *On* xolatda bo‘lsa, u holda *Break + Watsh* menyusi *Watsh window* (tekshirish oynasi) punktini oynaga o‘rnatadi yoki olib tashlaydi;

Quyida bosh menyuning mos ostki menyulariga qisqacha ta’riflar berilgan:

Files bo‘limi:

Load (F3)	fayl yoki dasturni chaqirish uchun xizmat qiladi;
Pisk (ALT + F3)	tanlashni belgilaydi;
New	yangi dastur uchun oynani tozalaydi;
Save (F2)	ekrandagi dasturni doimiy xotirada saqlaydi.
Write to	ishchi faylni yangi nom bilan saqlaydi;
Change dir	faol katalogni almashtiradi;
Os shell	vaqtincha OS ga chiqishni tashkil qiladi;

Quit (Alt + X)	TURBO PASCAL muhitidan chiqish.

Run bo‘limi:

Run (Ctrl + F9)	yuklangan dasturni ishga tushiradi ;
Program reset (Ctrl + F2)	oldindagi parametrlarni taxrirlash;
Goto sursor (F4)	kursor turgan joyga qaytishni amalga oshiradi;
Trase into (F7)	dasturni qadamlab bajarish imkoniyatini hosil qiladi;
Setup over (F8)	proseduraga kirishni ta’minlaydi;
User ssreen (Alt-F5)	natijalar oynasiga o‘tish.

Compile bo‘limi:

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

Options bo‘limi:

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

Debug bo‘limi:

Evaluate	har qanday o‘zgaruvchining qiymatini ko‘rish imkoniyatini yaratadi;
Call stask	dastur bajarilish vaqtida joriy chaqiruvchi stekni

	ko‘rsatadi;
Find prosedure	prosedurani izlash;
Integrated debugging	dasturni qadamlab bajarilishi uchun yo‘l ochib beradi;
Standalone debugging	dasturni **.exe* faylga aylantirish uchun yo‘l ochib beradi;
Display swapping	ekran hususiyatini o‘rnatadi;
Refresh display	ekran hususiyatini aktivlashtiradi.

Break + ***Watsh*** bo‘limi:

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

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

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

3.4. PASKAL MUHITIDA ENG KO‘P UCHRAYDIGAN XATOLIKLAR

Dasturlarni EHM xotirasiga kiritish va bajarishda ayrim xatoliklarga yo‘l qo‘yilgan bo‘lishi mumkin. Qo‘yilgan xatoliklar xarakteri haqida Paskal kompliyatori standart axborotlar beradi. Ana shu axborotlarga qarab, yo‘l qo‘yilgan xatolikni aniqlab, bartaraf qilish lozim. Dasturda bitta ham xato qolmagandan keyingina uni EHM yordamida bajarib natijasini olish mumkin.

Quyidagi jadvalda ana shunday xatoliklar ichida amaliyotda eng ko‘p

uchraydigan xatoliklar, ularning mazmuni va bartaraf qilish yo‘llari tavsiya qilingan.

Paskal axboroti	xatolikning mazmuni	bartaraf qilish yo‘li
“ “ expested	“ ” o‘rtasida ko‘rsatilgan belgi etishmayapti	“ “ o‘rtasida ko‘rsatilgan belgini qo‘ying
begin expested	begin o‘rniga boshqa xizmatchi so‘z kelgan	xizmatchi so‘zlarni to‘g‘ri qo‘ying
boolean expression expested	Mantiqiy ifodada "=" o‘rniga ":=" kelgan	belgilarni to‘g‘ri qo‘ying.
sonstant out of range	Ruxsat etilgan diapazondan chetga chiqilgan	ma’lumotlar tipning diapazoni doirasida bo‘lishini ta’minlang
duplisate identifier	Identifikator ikki marta aniqlangan	identifikatorga boshqa nom bering
division by zero	nolga bo‘linish	bo‘luvchining nolga teng bo‘lmasligini ta’minlang
error in expression	qiymat berishda amal belgisi tushib qolgan	tushib qolgan belgini o‘rniga qo‘ying
file not found	izlangan fayl yo‘q	fayl adresining to‘g‘ri ekanligini tekshiring
invalid numeris format	kiritilgan ma’lumot tipiga o‘zgaruvchi tipi mos emas	o‘zgaruvchining tipiga mos keladigan qiymat kiriting
invalid qualifier	e’lon qilinmagan massiv elementidan foydalanish	massivni to‘g‘ri e’lon qiling yoki element indeksini tekshiring
line too lange	juda uzun satr	satrdagi belgilar sonini kamaytiring
syntax error	mumkin bo‘lmagan belgi (apostrof o‘rnida qo‘shtirnoq kelgan bo‘lishi mumkin	shu belgini yo‘qoting.
type mismatsh	o‘zgaruvchi va uning qiymati tiplari mos emas	o‘zgaruvchi tipiga mos qiymatlar tanlang
unknown identifier	noma’lum o‘zgaruvchi	o‘zgaruvchining tipini aniqlang
unexpested end of file	dastur oxirida END operatori yetishmayapti	dastur oxiriga END operatorini qo‘ying

Nazariy savollar va tayanch iboralar:

1. Install.exe programmasining vazifasini tushuntiring.
2. Turbo-Paskalning asosiy fayllarini yozib bering va ularning vazifalarini ayting.
3. Turbo-Paskalning qaysi versiyasi amalda ko‘proq ishlatiladi?
4. Turbo.exe faylining vazifasini tushuntiring.
5. Turbo-Paskal muxitining bosh menyu bo‘limlari vazifasini tushuntirib bering.
6. Files bo‘limining ostki menyulari vazifasini ayting.
7. Run bo‘limining ostki menyulari vazifasini ayting.
8. Compile bo‘limining ostki menyulari vazifasini ayting.
9. Opions bo‘limining ostki menyulari vazifasini ayting.
10. Debug bo‘limining ostki menyulari vazifasini ayting.
11. Break + Watsh bo‘limining ostki menyulari vazifasini ayting.
12. Programma matnini taxrirlash uchun ishlatiladigan funksional tugmachalar vazifalarini tushuntiring.
13. Paskal tilidagi biror tayyor programma bilan muhitda ishlab ko‘ring.
14. Kompilyatorning xatoliklar haqidagi axborotlarini yod oling.

§-4. PASKAL HAQIDA BOSHLANG‘ICH MA’LUMOTLAR

4.1. PASKAL TILINING ALIFBOSI

Paskal dasturlash tilining alifbosi deb, shu tilda ma’lumotlarni ifodalash va dasturlar yozish jarayonida kompilyator tomonidan qabul qilishga ruxsat berilgan belgilar yoki maxsus belgilardan iborat bo‘lgan zanjirlar to‘plamiga aytiladi.

Bu alifbo o‘z ichiga ASCII (halqaro belgilar va ularning uodlari) jadvalining hamma belgilarini, ya’ni quyidagilarni oladi:

1. Lotin alifbosining katta va kichik sharflari;
2. 0 dan 9 gacha arab raqamlari;
3. Tagiga chizish belgisi (_);
4. Bo‘sh joy belgisi; (bu belgi paskal tilida barcha xizmatchi so‘zlar va ma’lumotlarni bir-biridan ajratish uchun foydalaniladi;
5. Boshqaruvchi belgilar : ASCII jadvalidagi kodlari 0 dan 31 gacha bo‘lgan belgilar. Bu belgilar satr va konstantalarni ifodalashda qo‘llanishi mumkin;
6. Turli ko‘rsatmalarni yozish uchun ishlatiladigan maxsus belgilar : + − * / = {} () [] . , @ # ' : ; $ % ^ ! ? & \ < >

7. Asosiy bo‘lmagan belgilar: (ASCII ni kengaytiruvchi, ya’ni kodi 128 dan 255 gacha bo‘lgan belgilar. Rus alifbosining katta va kichik harflari, psevdografika elementlari shu sinfga kiradi. Bu belgilar turli konstantalarni hosil qilishda, matnlarni yozishda, izoxlarni tashkil qilishda qo‘llanishi mumkin.);
8. Murakkab belgilar: <= >= := .. * *;
9. Xizmatchi so‘zlar. Ular paskal tilida ma’lum bir ma’no yoki ko‘rsatmani anglatuvchi maxsus belgilar zanjiridan iborat bo‘lib, bu zanjirni o‘zgartirish yoki qisqartirib qo‘llash mumkin emas. Masalan: ***begin, program, end*** va hokazo.

Eslatma: Agar dasturlar yozishda yuqorida sanab o‘tilgan belgilardan boshqa belgi yoki xizmatchi bo‘lmagan so‘zlar uchrab qolsa, bu haqida kompilyator maxsus axborotni EHM ekraniga chiqaradi.

4.2 TURBO PASKAL TILIDA O‘ZGARUVCHILAR

Ma’lumki, dasturlar turli sonli va boshqa turdagi ma’lumotlarni qayta ishlash uchun yoziladi. Ana shu ma’lumotlarni bir-biridan farqlash uchun ularni nomlashga to‘g‘ri keladi. Bu nom sifatida identifikatordan (o‘zgaruvchi nomi) foydalaniladi.

Identifikator muayyan bir vaqtda ifodalab turgan son yoki boshqa turdagi ma’lumot uning qiymati xisoblanadi. Dasturda qatnashgan xar bir identifikator uchun EXM xotirasidan ma’lum bir joy ajratiladi xamda bu joyda uning qiymati yozib qo‘yiladi va saqlanadi.

Identifikatorlar doim lotin xarflari bilan boshlanadi. Uni yozish uchun zarur bo‘lgan keyingi belgilar esa lotin xarflari, raqamlar va“_ ” belgisidan iborat bo‘lishi mumkin:

X, xl, s4, absl2d, fam, kitob_soni.

Dastlabki to‘rtta identifikator sintaktik jixatdan to‘g‘ri yozilgan bo‘lishiga qaramay, katta hajmli dasturlarni yozishda ma’lum bir qiyinchiliklarni tug‘dirishi mumkin. CHunki, ular anglatayotgan ma’lumotlar to‘la va tushunarli qilib ifodalanmaydi. Natijada bunday identifikatorlarni boshqasi bilan almashtirib yuborish ehtimolligi ortib ketadi hamda ana shu dasturni o‘qish va tushunish qiyinlashib ketadi. SHuning uchun identifikatorlarni keyingi ikkitasi kabi belgilagan maqsadga muvofiq hisoblanadi. CHunki ular o‘zlari ifodalayotgan o‘zgaruvchilarni ma’lum bir darajada izoxlab turadi va anglashilmovchiliklarga barham berishda muhim o‘rin tutadi. Identifikator tanlaganda ham ma’lumotlarning shakli va mazmunini hisobga olish ham ana shu omillardlan biri hisoblanadi. Masalan, uchburchak haqidagi masalada ehtiyojga qarab

a_tomon, b_tomon, s_tomon, yarim_perimetr, yuza

kabi identifikatorlar maqsadga muvofiq hisoblanadi.

Identifikatorlarni yozishda Paskal tili kompilyatori katta va kichik harflarni ajratmaydi, ya'ni yuza, Yuza, YUZA, YuZa kabi identifikatorlarni bitta identifikator sifatida qabul qiladi.

Identifikator sifatida xizmatchi so‘zlar, turli tinish belgilari, munosabat belgilari kabi belgilardan foydalanib bo‘lmaydi. Ularni quyidagicha yozish noto‘g‘ri hisoblanadi:

$$4x,\ X-y,\ G=dr,\ AB,\ !gamma,\ a?b,\ begin,\ end.$$

Agar identifikator dasturning bajarilishi davomida o‘z qiymatini o‘zgartirmasa, ularni o‘zgarmaslar yoki konstantalar, aks holda o‘zgaruvchilar deb ataladi. O‘zgarmas ma'lumotlar dastur matnida maxsus xizmatchi so‘z ***sonst*** yordamida alohida ta'kidlab ko‘rsatiladi. Masalan:

$$const\ \ gamma = 1.23;.$$

SHundan keyin bu o‘zgarmaslarning qiymatlarini dasturning bajarilishi davomida o‘zgartirib bo‘lmaydi. Dasturda qatnashayotgan barcha o‘zgaruvchilar esa biror tipga mansub bo‘lishi shart.

Dasturda qatnashayotgan barcha identifikatorlarga EHM xotirasidan joy ajratiladi. Bu joyda ana shu identifikatorning qiymati saqlanadi. SHu identifikatorga murojaat qilinganda, uning uchun ajratilgan joy, ya'ni yacheykada saqlanayotgan ma'lumotni EHM o‘qiydi va shu ma'lumotni identifikatorning o‘rniga qo‘yadi.

4.3. MA'LUMOTLARNING TIPLARI

Aslini olganda, 2 va 2.0 sonlari bir hil miqdorni anglatadi. Lekin Paskal tili kompilyatori ularni bir-biridan farqli sonlar sifatida, ya'ni 2 sonini butun, 2.0 ni esa haqiqiy son deb qabul qiladi. Butun sonlar bilan kompilyator odatdagi barcha amallarni bajara oladi. Ammo, haqiqiy sonlar bilan ishlaganda ularni yaxlitlash hisobiga taqribiy hisoblashga yo‘l ko‘yadi.

Paskal tili haqiqiy va butun sonlarni bir nechta turga bo‘ladi va har bir turdagi sonlarni saqlash uchun o‘z xotirasidan ma'lum bir hajmdagi joyni ajratib beradi. (2-jadval.)

sonli tiplar jadvali 2-jadval

tipi	hajmi	diapazoni	razryadi
integer	2 bayt	-32767 ... 32767	
longint	4 bayt	-2 147 483 647 ... 2 147 483 647	
shortint	1 bayt	-128 ... 128	
byte	1 bayt	0 ... 255	

word	2 bayt	0 ... 65535	
real	6 bayt	$-2.9\cdot10^{-39}$ ... $1.7\cdot10^{38}$	11-12
single	4 bayt	$-1.5\cdot10^{-45}$ ... $3.4\cdot10^{38}$	7-8
double	8 bayt	$-5.0\cdot10^{-324}$ ... $1.7\cdot10^{308}$	15-16
extented	10 bayt	$-3.4\cdot10^{-4932}$ ... $1.1\cdot10^{4932}$	19-20
somp	8 bayt	$-9.2\cdot10^{-18}$ ... $9.2\cdot10^{18}$	19-20

Haqiqiy sonlarni kompilyator ikki hil usulda yozishga ruxsat beradi: oddiy yozuv va ko‘chuvchi vergulli yozuv. Haqiqiy sonlarni oddiy usulda yozishda ularni odatdagidek yozish mumkin:

2.0, 3.4563, -1.025, -0.00125.

Ko‘chuvchi vergulli yozuvda esa haqiqiy sonning butun va kasr qismini ajratib turuvchi vergulni ehtiyojga qarab o‘ngga yoki chapga surish mumkin. Bunda vergulning surilish xonalariga qarab, uning tartibini o‘zgartiriladi. Masalan:

$123.0 = 12.0\cdot10^{1} = 0.123\cdot10^{3} = 12300.0\cdot10^{-2}$.

Sonning tartibini ko‘rsatuvchi «o‘nning darajasi» yozuvi o‘rniga paskal kompilyatori «e» belgisidan foydalanadi.

Sonli ma’lumotlardan tashqari Paskalda ASCII jadvalidagi ixtiyoriy bitta belgidan iborat bo‘lgan ***shar*** tipidagi ma’lumot bilan ham ishlash imkoniyati yaratilgan. '*A*', '*c*', '–', '%', '!', '' kabi ma’lumotlarni ***shar*** tipidagi ma’lumot deb qabul qilish mumkin. Bu tipdagi ma’lumotlarni saqlash uchun EHM o‘z xotirasidan 1 bayt joy ajratadi.

Boolean tipidagi ma’lumotlar mantiqiy ifodalarning qiymatlarini qayd etish uchun foydalaniladi. Ravshanki, mantiqiy ifodalar yoki true (rost) yoki false (yolg‘on) qiymatlarini qabul qiladi halos. Bu tipdagi ma’lumotlarni saqlash uchun 1 bayt joy ajratiladi.

String (satr) tipidagi ma’lumotlar 256 tagacha belgilar ketma-ketligini o‘z ichiga olishi mumkin. Bu tipdagi ma’lumotlarni saqlash uchun EHM xotirasidan 256 baytgacha joy ajratiladi. EHM xotirasidan unumli foydalanish maqsadida ***string*** tipidagi ma’lumotdagi belgilar sonini ehtiyojga qarab chegaralab qo‘yish mumkin. Masalan, familiyalarda ko‘pi bilan 20 tagacha harflar bo‘lishini hisobga olib, ***string[20]*** tipidagi ma’lumot aniqlansa, bu tipdagi ma’lumotni saqlash uchun 20 bayt joy ajratiladi halos.

Yuqorida keltirilgan ma’lumotlar ichida asosiy tip sifatida ***real, integer, boolean, shar*** tiplari qabul qilingan. Agar zarurat tug‘ilsa, bu tiplardan tashg‘ari yangi tiplarni ham aniqlash mumkin. Biz bu masala bilan keyinroq tanishamiz.

4.4. MA’LUMOT TIPINI ALMASHTIRISH FUNKSIYALARI

Ko‘pincha ma’lumotlarni bir tipdan bosh qa tipga o‘tkazishga to‘g‘ri kelib qoladi. Bunda foydalanish mumkin bo‘lgan ayrim funksiyalarni 3-jadvalda keltirib o‘tamiz.

3-jadval

funksiya	vazifasi va tipi
shr(n)	ASCII jadvalida n-o‘rinda turgan belgini aniqlaydi. ***n-byte*** tipidagi ma’lumot. Funksiyaning qiymati ***shar*** tipida bo‘ladi.
ord(x)	Tartiblangan tiplar uchun qo‘llanadi va shu tipdagi ma’lumotlar ichida *x*-qiymatli ma’lumot nechanchi o‘rinda turganligini aniqlaydi. Funksiyaning qiymati ***shar, integer, byte, word, longint, shortint*** kabi tiplarga mansub bo‘lishi mumkin.
round(x)	x-haqiqiy qiymatli ma’lumotni yaxlitlash uchun ishlatiladi.
truns(x)	x-haqiqiy qiymatli sonning butun qismini ajratib olish uchun xizmat qiladi. SHuning uchun tipi integer bo‘ladi.
odd(N)	Berilgan N butun sonining juft yoki toqligini aniqlaydi. Tipi ***boolean***. N soni juft bo‘lsa ‘false’, aks holda ‘true’ qiymatini oladi.

Tiplarni almashtirish uchun qo‘llaniladigan boshqa funksiyalar bilan extiyojga qarab, o‘rni kelganda tanishib boramiz.

4.5. TARTIBLANGAN TIPLAR UCHUN AYRIM FUNKSIYALAR

funksiya	vazifasi
des(x [N:longint])	Berilgan x miqdorni N ga kamaytirish uchun ishlatiladi. Agar N ko‘rsatilmagan bo‘lsa, 1 ga kamaytiriladi.
ins(s [N:longint])	Berilgan x miqdorni N ga orttirish uchun ishlatiladi. Agar N ko‘rsatilmagan bo‘lsa, 1 ga orttiriladi.
pred(x)	Bu funksiya x dan bitta avvalgi elementni aniqlash

	uchun foydalaniladi.
suss(x)	Bu funksiya x dan bitta keyingi elementni aniqlash uchun qo‘llaniladi.

4.6. SONLAR, ARIFMETIK AMALLAR VA IFODALARNI YOZISH

Butun sonlarni Paskal tilida ham odatdagidek yozish mumkin:

Oddiy yozuv	Paskal tilida
1	1
123	123
-1024	-1024

Haqiqiy sonlarni yozishda, ularning butun va kasr qismini ajratish uchun uchun vergul belgisi o‘rniga nuqtadan foydalaniladi.

Oddiy yozuv	Paskal tilida
1,0	1.0
12,3	12.3
-1,024	-1.024

Juda katta yoki kichik, ya’ni eksponensial ko‘rinishidagi sonlarni yozishda odatdagi «o‘nning darajasi» yozuvi o‘rniga «e» belgisidan foydalaniladi.

Oddiy yozuv	Paskal tilida
$1{,}2 \cdot 10^{25}$	1.2e25
$1{,}23 \cdot 10^{-19}$	1.23e-19
$-2{,}8 \cdot 10^{-27}$	-2.8e-27

Paskal tilida sonlar va o‘zgaruvchilar ustida qo‘shish $(+)$, ayirish (-), ko‘paytirish (*), bo‘lish $(/)$, qoldiqni aniqlash (MOD), bo‘linmaning butun qismini topish (DIV) kabi amallardan foydalanish mumkin.

Arifmetik amallar odatdagi yozuvdan bitta satrga yozilish bilan farqlanadi.

Oddiy yozuv	Paskal tilida
$a+b$	$a+b$
b^2-4ac	$b*b-4*a*c$
$z(a+b)$	$z*(a+b)$
$\frac{a}{b}$	a/b
$\frac{x-y}{x+y}$	$(x-y)/(x+y)$

$\dfrac{x}{1+\dfrac{c+d}{nm}}$	$x/(1+(c+d)/(n*m))$

Arifmetik ifodalarda standart funsiyalarning ko‘p uchrashini hisobga olib, ularning xam yozilishini keltirib o‘tamiz. Bu funksiyalarning argumentlari qavslar ichida yoziladi.

oddiy yozuv	ma’nosi	Paskal tilida
$\lvert x\rvert$	absolyut qiymat	abs(x)
$\sqrt{x}$	kvadrat ildiz	sqrt(x)
x^2	kvadrat	sqr(x)
e^x	eksponenta	exp(x)
$\ln x$	natural logarifm	ln(x)
π	pi (3.14...)	pi
$\sin x$	sinus	sin(x)
$\cos x$	kosinus	sos(x)
arctg x	arktangens	arstan(x)
	argumentning butun qismi	truns(x)
	argumentning kasr qismi	fras(x)

Eslatma: 1. Jadvaldagi trigonometrik funksiyalarning argumentlari radianlarda ifodalanishi lozim. 2. Jadvalga kirmagan funksiyalarni jadvaldagi funksiyalar orqali ifodalash shart. Masalan: $\log_a b = \dfrac{\ln b}{\ln a}$ yoki $ctg\, x = \dfrac{\cos x}{\sin x}$ kabi.

§-5. SODDA DASTURLAR YOZISH

5.1. DASTURNING UMUMIY KO‘RINISHI

Paskal tilida eng sodda dasturlarni yozish uchun quyidagi umumiy qoida qabul qilingan:

1. Dastur ***program*** so‘zi bilan boshlanadi, undan keyin dasturning nomi yoziladi.
2. Dastur ***end*** so‘zi bilan tugaydi.
3. O‘zgarmaslar ro‘yxati odatda ikkinchi satrda joylashib, ***sonst*** so‘zidan keyin keltiriladi.
4. Dasturda qatnashadigan barcha o‘zgaruvchilar va ularning tiplari ***var*** so‘zidan keyin ko‘rsatiladi.
5. Masalaning Yechish uchun paskal tilidagi buyruqlar ketma-ketligi ***begin*** dan keyin yoziladi.
6. Dasturdagi har bir ko‘rsatma yoki buyruq boshqalaridan «;» belgisi bilan

ajratiladi.

7. Dasturdagi oxirgi ***end*** dan keyin nuqta qo‘yiladi.

Agar yozilayotgan dasturda o‘zgarmas ma’lumotlar qatnashmasa, ushbu umumiy sxemadagi 3-punktni tushirib qoldirish mumkin.

Dasturda o‘zgaruvchilardan ham foydalanishga ehtiyoj bo‘lmasa, 4-punktni ko‘rsatmaslik mumkin.

5.2. QIYMAT BERISH BUYRUG‘I

O‘zgaruvchilarning qiymatlarini o‘zgartirish uchun ularga qiymat beriladi. Bu ishni qiymat berish buyrug‘i, ya’ni ":=" yordamida amalga oshiriladi. Buyruqning umumiy ko‘rinishi quyidagicha:

$$\alpha := \beta;$$

Bu erda α-qiymat olayotgan o‘zgaruvchi, $\beta -$ esa qiymati α ga beriladigan sonli, arifmetik, mantiqiy yoki xarfiy ifoda. Bu buyruqning ma’nosi quyidagicha: $\beta -$ ifodaning qiymati hisoblanadi va bu qiymat α ga beriladi, ya’ni EHM xotirasidan α uchun ajratilgan yacheykaga yozib qo‘yiladi. Masalan:

$$x := 20.25;\ y := (x + 0.75) * 2;$$

$$a1 :=' paskal\ \ tili';$$

Bu buyruqlar bajarilganidan so‘ng, *x*-o‘zgaruvchi 20,25 ni, *u*-esa 42 ni, xarfiy *a1* o‘zgaruvchi esa «raskal tili» degan qiymatlarni oladi. Demak, bu o‘zgaruvchilar uchun ajratilgan yacheykaga ana shu qiymatlar yozib qo‘yiladi.

Agar zarur bo‘lsa, qiymat olayotgan o‘zgaruvchi qiymat berish buyrug‘ining o‘ng tomonida ham kelishi mumkin. Bu holda o‘ng tomondagi ifodaning qiymatini hisoblash uchun uning «eski» qiymatidan foydalaniladi. Ifodaning qiymati hisoblab bo‘linganidan keyin yacheykadagi «eski» qiymat o‘chiriladi va uning o‘rniga «yangi» qiymat yozib qo‘yiladi. Masalan:

$$alfa := 15;\quad alfa := alfa * 2;$$

buyruqlari bajarilganidan keyin, ***alfa*** ning qiymati 30 ga teng bo‘lib qoladi.

Eslatma: Qiymat berish buyrug‘ining o‘ng tomonidagi ifodada qatnashayotgan har bir o‘zgaruvchining qiymati oldindan aniqlangan bo‘lishi lozim.

5.3. KIRITISH VA CHIQARISH BUYRUQLARI

Ko‘pincha masalalarni Yechish jarayonida masalaning shartida berilgan ma’lumotlarni klaviatura orqali kiritishga to‘g‘ri kelib qoladi.

read operatori o‘zgaruvchilarga qiymatlarni klaviatura yordamida berishni tashkil qilish uchun ishlatiladi. Bu operator umumiy ko‘rinishda quyidagicha yoziladi:

read (o‘zgaruvchilar ro‘yxati);

Ro‘yxatdagi o‘zgaruvchilar bir-birlaridan vergul bilan ajratiladi. Masalan:

read (r,k,h); .

Read buyrug‘ini bajargan EHM ishdan to‘xtaydi va ro‘yxatda ko‘rsatilgan barcha o‘zgaruvchilar uchun qiymat kiritilishini kutadi. Klaviaturadan kiritilayotgan ma’lumotlar bir-biridan bo‘sh joy belgisi bilan ajratiladi. Kiritilgan ma’lumotlar tartib raqamlariga qarab mos ravishda berilgan ro‘yxatdagi o‘zgaruvchilarga qiymat qilib beriladi. Boshqacha aytganda, birinchi kiritilgan ma’lumot ro‘yxatdagi birinchi o‘zgaruvchiga, ikkinchi ma’lumot ikkinchisiga va xokazo tartibda beriladi. Yuqoridagi opreator uchun klaviaturadan quyidagi ma’lumotlar kiritilgan bo‘lsin:

2.34 15 Paskal

U xolda *r* ga 2.34, *k* ga 15, *h* ga esa «Paskal» qiymatlari beriladi va dasturning keyingi buyruqlari o‘zgaruvchilarning ana shu qiymatlari uchun bajariladi.

Qiymat olayotgan o‘zgaruvchi bilan unga berilayotgan qiymat bir xil tipga mansub bo‘lishi lozim. *Char* yoki *string* tipidagi ma’lumot kiritilayotganda ularni apostrof orasiga olish shart emas. *Real* tipida ma’lumot kiritilayotganda esa butun sonlarni xam kiritishga ruxsat beriladi. (Bu holda kiritilgan 10 sonini 10.00 tarzida qabul qilinadi.) *Boolean* tipidagi ma’lumot sifatida faqat *false* yoki *true* qiymatlaridan birini kiritish mumkin xalos.

Klaviaturadan kiritilgan ma’lumotlar soni *read* operatorida berilgan ro‘xatdagi o‘zgaruvchilar sonidan kam bo‘lmasligi lozim. Aks holda, ro‘yxatdagi qaysidir o‘zgaruvchi qiymat olmagani sababli navbatdagi operatorlar bajarilmay turaveradi.

Agar kiritilgan ma’lumotlar soni *read* operatori bilan ko‘rsatilgan o‘zgaruvchilar sonidan ko‘p bulsa, buning zarari yo‘q. CHunki ortiqcha qiyomatlar yoki navbatdagi *read* dagi o‘zgaruvchilarga qiymat qilib beriladi. Masalan, bitta dasturda

read (a,b,s); read (x,y);

operatorlariga javoban klaviaturadan

2.3 -1.5 2.4 22 -0.05 4.125

ma’lumotlari kiritilgan bo‘lsa, *a* ga 2.3, *b* ga -1.5, *s* ga 2.4 qiymatlari o‘zlashtirilsa, *x* o‘zgaruvchi 22, *y* esa -0.05 qiymatlari beriladi. Ortiqcha kiritilgan 4.125 dan esa EHM foydalanmaydi, ya’ni tashlab yuboradi.

Re*ad* ln opreatori ro‘yxatda ko‘rsatilgan o‘zgaruvchilarga qiymat kiritilganidan so‘ng, kursorni yangi satrning boshiga o‘tkazib qo‘yadi. Bu xolda ortiqcha ma’lumotlarning barchasi tashlab yuboriladi, navbatdagi *read* yoki *read* ln yordamida berilgan o‘zgaruvchilarga esa qiymat qilib yangi satrning boshida kiritilgan ma’lumotlar olinadi. Masalan:

read ln *(a,b,s); read (x,y);*

operatorlari uchun klaviatura orqali

2.3 -1.5 2.4 22 3.75

-0.05 4.125

ko‘rinishidagi ma’lumotlar kiritilgan bo‘lsa, *a*, *b, s* o‘zgaruvchilar mos ravishda 2.3, -1.5, 2.4 qiymatlarini olsa, *x* bilan *y* o‘zgaruvchilar -0.05 va 4.125 ni oladi.

Write operatori turli hisoblash natijalarini, matnlarni hamda arifmetik ifodalarning qiymatlarini hisoblab displey ekraniga chiqarish uchun hizmat qiladi. Bu operator umumiy holda quyidagicha yoziladi:

write (chiqariladigan ma’lumotlar ro‘yxati); .

CHiqariladigan ma’lumotlar bir-birlaridan vergul bilan ajratiladi.

Ekranga chiqarish kerak bo‘lgan matnlarni apostrof belgisi bilan ko‘rsatiladi. Masalan:

write (‘Paskal tili’);

buyrug‘i bajarilganda ekranga

Paskal tili

yozuvi chiqariladi.

Write operatori yordamida paskal dasturlash tili qoidalari bilan yozilgagn arifmetik ifodalarning qiymatlarini ham hisoblash mumkin. Masalan:

$write(3*4+15/3-10.5);$

buyrug‘ining natijasi qavslar ichida berilgan ifodaning qiymati bo‘lgan $6{,}5000000000E+00$ ko‘rinishdagi sonni ekranga chiqarishdan iborat bo‘ladi.

Agar *write* yordamida ekranga chiqarish talab qilingan ma’lumotlar sifatida o‘zgaruvchilar ro‘yxati berilgan bo‘lsa, u holda bu o‘zgaruvchilarning qiymatlari EHM xotirasidan qidirib topiladi va ekranga chiqariladi. Faraz qilaylik, biror dasturning bajarilishi davomida *x, y, x* o‘zgaruvchilar mos ravishda 23, 123.12, ‘Paskal’ qiymatlarini olgan bo‘lsin. U holda

write (x,y,z);

operatorini bajarilishi natijasida ekranda

$2.3000000000E+01$ $1.2312000000E+02$ Paskal

ko‘rinishidagi ma’lumotlar chiqariladi. Bu yozuvlardan ko‘rinib turibdiki, ekrandagi sonli ma’lumotlar o‘qish va tushunish uchun bir oz noqulay. Ana shu noqulaylik oldini olish uchun ekranda uzatiladigan sonli ma’lumotlarni ma’lum bir o‘lchamga solishga (formatlashga) to‘g‘ri keladi. O‘lcham odatda ikki butun sondan iborat bo‘lib, ularning birinchisi umumiy raqamlar sonini, ikkinchisi esa kasr qismning raqamlari sonini belgilaydi. O‘lcham formatlanayotgan o‘zgaruvchidan keyin ikki nuqta (:) orqali aniqlanadi va faqat shu o‘zgaruvchigagina tegishli bo‘ladi. Yuqoridagi ma’lumotlarnip formatlab, ekranga uzataylik:

write (*x:4:2, y:6:2, z*);

Bu buyruq ta’sirida ma’lumotlar quyidagi ko‘rinishda ekranga chiqariladi:

23.00 123.12 Paskal

Butun son tipidagi ma’lumotlarda haqiqiy qism bo‘lmasligini yodda tutish zarur.

Ekranda (ko‘rsatkich) kursor mavjud bo‘lib, u ma’lumotlarni qaysi joydan boshlab kiritilishi yoki chiqarilishi kerakligini ko‘rsatadi. Navbatdagi ma’lumot kursor turgan joydan boshlab kiritiladi yoki chiqariladi. Yuqorida biz *readln* yordamida ana shu kursorni navbatdagi satrning birinchi pozisiyasiga o‘tkazishni ko‘rgan edik. SHu holatni *writeln* yordamida ham takrorlash mumkin.

Writeln operatori talab qilingan ma’lumotlarni ekranga chiqarganidan keyin, kursorni yangi satrning boshiga o‘tkazadi. Masalan:

write (x); write (y); write (z);

buyruqlari ma’lumotlarni

$2.3000000000E+011.2312000000E+02$ Paskal

ko‘rinishida ekranga chiqarsa,

writeln(x); writeln(y); write(z);

buyruqlari ma’lumotlarni ekranga

$2.3000000000E+01$

$1.2312000000E+02$

Paskal

tarzida chiqarilishini ta’minlaydi.

Readln va *writeln* operatorlaridan hech qanday argumentsiz ham foydalanish mumkin. *Readln* buyrug‘i ENTER tugmasini bosilishi kutadi va kursorni yangi satrning boshiga o‘tkazsa, *writeln* esa shunchaki kursorni yangi satrning boshiga ko‘chiradi.

5.4. CHIZIQLI DASTURLAR YOZISH

5.1-dagi ma’lumotlarni hisobga olsak, dasturlarning paskal tilidagi umumiy ko‘rinishi quyidagicha bo‘lishini ko‘rish qiyin emas:

Program dasturning nomi;

Const o‘zgarmaslar ro‘yxati;

Var o‘zgaruvchilar va ularning tiplari ro‘yxati;

Begin

Paskal tilida echilayotgan masalaning algoritmiga

mos keluvchi buyruqlar ketma-ketligi;

End.

Agar yozilayotgan dasturda o‘zgarmas ma’lumotlar qatnashmasa, ushbu umumiy sxemadagi *sonst* ni tushirib qoldirish mumkin.

Dasturga majburiy bo‘lmagan, ammo ehtiyoj paydo bo‘lganda yozish shart bo‘lgan yangi qo‘shimcha turli elementlarni qo‘shish mumkin. Biz bu elementlar bilan keyinchalik tanishamiz.

CHiziqli dastur deganda masalaning algoritmiga mos keluvchi buyruqlar ketma-ketligi dasturda uchrash tartibiga mos ravishda bajariladigan dasturlarni tushuniladi. Demak, dasturni bajarish boshlanganda, dastlab 1-buyruq, keyin 2-buyruq va x.k. tarzida bajariladi. Bunda *begin* va *end* larga alohida e’tibor beriladi. Ular operatorlar qavsi hisoblanadi va amallarni bajarish paytida ustunlikka ega bo‘ladi.

Dasturning bitta satrida bir nechta buyruqlar kelishi mumkin. Bunda ular bir-biridan «;» belgisi bilan ajratiladi.

Paskal tilidagi dastur qo‘yilgan masalaning Yechish algoritmidagi buyruqlarni kompilyatorga «tushunarli» bo‘lgan ko‘rinishda yuqoridagi umumiy sxemaga muvofiq ifodalash gatijasida hosil bo‘ladi.

Masala: Uzunligi L bo‘lgan aylana bilan chegaralangan doira yuzini toping.

Masalaning Yechish g‘oyasi: Ma’lumki, doiraning yuzi $S = \pi R^2$ formula bilan hisoblanadi. Bu formulani qo‘llash uchun bizga R ning qiymati zarur. Uni aylana uzunligi L dan foydalanib topish mumkin, ya’ni $R = L/(2\pi)$. Bu ma’lumotlarni hisobga olib, masalaning algoritmi va dasturini yozamiz.

algoritmi	*dasturi*
1. Boshlansin; 2. Aniqlansin L; 3. Hisoblansin $R = L/(2\pi)$; 4. Hisoblansin $S = \pi R^2$; 5. CHiqarilsin S; 6. Ishni tugatilsin.	*program yuza;* *var L,R,S:real;* *begin* *readln(L);* *R := L/(2 * Pi);* *S := Pi * R * R;* *writeln(‘S = ’,S:6:2)* *end.*

Ushbu dasturni EHM da bajarish uchun, avval Turbo Paskal muxitini ishga tushiramiz. So‘ngra dastur matnini kiritamiz. Bu matnni xatoliklarga tekshiramiz, ya’ni kompilyasiya qilamiz. Buning uchun $ALT + F9$ yoki $ALT + C$ tugmalaridan foydalanamiz. Matnni kiritishda yo‘l qo‘yilgan xatoliklar mavjud bo‘lsa, ularni bartaraf etamiz. Xatoliklar qolmagandan so‘ng, dasturni bajarishga ko‘rsatma beramiz. Buning uchun $CTRL + F9$ yoki $ALT + R$ tugmalaridan foydalanish mumkin.

Yuqoridagi dasturni $L = 100$ uchun bajarilsa, ekranda $S = 795.82$ natija paydo bo‘ladi.

§-6. SATRLI MA'LUMOTLAR BILAN ISHLASH

6.1. CHAR TIPIDAGI MA'LUMOTLAR

EHM da eng ko'p foydalaniladigan 256 ta belgi ajratib olingan bo'lib, bu belgilar o'z ichiga lotin va kirill alifbosining xarflari, raqamlar, amal va munosabat belgilari, tinish belgilari, psevdografika elementlari va boshqalarni oladi. Bu belgilarga 0 dan 255 gacha bo'lgan raqamlar, ya'ni kodlar mos qo'yilgan. Masalan: bu jadvalda 0 dan 9 bo'lgan raqamlarga 48 dan 57 gacha bo'lgan kodlar, katta lotin xarflariga 65 dan 90 gacha, kichik lotin xarflariga esa 97 dan 122 gacha, «ENTER» tugmasiga esa 32 kod to'g'ri keladi. Xosil bo'lgan jadvalni ASCII kodlar jadvali deb ataladi.

ASCII jadvali EHM da qo'llaniladigan belgilarning xalqaro standarti hisoblanadi. Ammo, zarurat bo'lsa, har bir davlat bu jadvalga o'zi uchun hos bo'lgan ayrim belgilarni qo'shishi mumkin. Bunga misol qilib kirill alifbosini olish mumkin.

Char tipidagi ma'lumot deganda, ASCII jadvalining ixtiyoriy bitta belgisidan iborat bo'lgan ma'lumot tushuniladi. Masalan: «F», «5», «*», «#», «}», «←», «⌈» kabi belgilarning xar birini *shar* tipidagi ma'lumot deb qabul qilish mumkin.

ORD(x) funksiyasi *shar* tipidagi *x* ma'lumot ASCII jadvalida nechanchi o'rinda turganligini aniqlash uchun ishlatiladi. Masalan, $x = "A"$ uchun *ORD(x)* yoki *ORD('A')* funksiyalari 65 natijani beradi, ya'ni «A» belgisi ASCII jadvalida 65 o'rinda joylashgan.

CHR(N) funksiyasi *ORD(x)* ga teskari bo'lib, N-o'rinda qaysi belgi turganligini aniqlaydi. Bu erda N sifatida *byte* tipidagi ma'lumotlar, ya'ni 0 dan 255 gacha bo'lgan sonlarni olish mumkin. Masalan: $CHR(65) = 'A'$. Ko'rinib turibdiki, SHR(N) funksiyasining qiymati *shar* tipida bo'lar ekan.

6.2. STRING TIPIDAGI MA'LUMOTLAR

ASCII jadvaliga kirgan turli belgilar ketma-ketligidan iborat bo'lgan, ya'ni 256 tagacha belgini o'z ichiga olgan matn yoki satr ko'rinishidagi ma'lumotlarni s*tring* tipidagi ma'lumotlar deb qabul qilingan. «Men paskal tilini o'rganaman», «1-kursda o'qiysizmi?», «av@r123%:!!» kabi belgilar ketma-ketligi *string* tipidagi ma'lumotlarga misol bo'lishi mumkin. SHuni ta'kidlash kerakki, *string* tipidagi ma'lumot biror ma'noni anglatishi shart emas. (Oxirgi misolga qarang.)

Bu tipdagi o'zgaruchilarni ham boshqa tipdagi o'zgaruvchilar kabi e'lon qilinadi:

var s,a0,tekst: string;.

Har bir *string* tipidagi o'zgaruvchi uchun EHM xotirasidan 256 bayt joy ajratiladi. Ammo undagi belgilar soni n ($(n \le 255)$ ta bo'lsa bu joydan noo'rin foydalanish xolati yuzaga keladi, ya'ni *string* tipidagi o'zgaruvchi nechta belgidan

iborat bo‘lgan qiymatni qabul qilsa, bu qiymatni saqlash uchun shuncha bayt joy band qilinadi. Ana shu xolatni oldini olish uchun *string* tipidagi o‘zgaruvchilar bilan ishlaganda, ehtiyojga qarab, ularning uzunliklarini (belgilar sonini) chegaralash mumkin. Masalan, odamlarning familiyalari ko‘pi bilan 20 xarf atrofida bo‘lishini e’tiborga olinsa, ularni saqlash uchun belgilar soni 20 tagacha bo‘lgan o‘zgaruvchini

var fam : string[20];

tarzida e’lon qilish mumkin.

6.3. SONLI TIPDAGI BUYRUQLAR

Length(x) **-**buyrug‘i x-xarfiy kattalikning uzunligi (belgilar soni) ni aniqlash uchun xizmat qiladi va umumiy holda quyidagicha yoziladi:

butun tipli o‘zgaruvchi:= length(satrli o‘zgaruvchi yoki matn);

Bu funksiya butun tipdagi qiymatni qabul qiladi.

Misol: *program uzunlik;*

var x:string; n: byte;

begin

x:= ’Kush bilimdadur’;

n:= length(x);

writeln(‘n:= ’,n)

end.

Ushbu dastur *n:= 15* natijani beradi.

Val(x,a,b) buyrug‘i satrli ma’lumotni sonli ma’lumotga aylantirish uchun ishlatiladi. Bu erda *x*-satrli, *a* va *b* –butun sonli tipdagi ma’lumotlar. Agar x-satr faqat raqamlar ketma-ketligidan iborat bo‘lsa, *a*-ning qiymati shu ketma-ketlik hosil qilgan songa teng. Aks xolda *a* - nolga teng bo‘ladi. b-satrli ma’lumotni songa aylantirishdagi xatolik kodini ifodalab, qiymati sonlarni yozishda qo‘llanmaydigan belgi uchragan pozisiya nomeriga teng bo‘ladi.

satr	ifoda	*a*	*b*
‘2005’	*val(satr,a,b)*	2005	0
‘142.24E-12’	*val(satr,a,b)*	0	4
‘Paskal’	*val(satr,a,b)*	0	1

Pos(1_satr, 2_satr) – funksiyasi ***2_satr*** tarkibida ***1_satr*** ning bor yoki yo‘qligini tekshiradi. Agar yo‘q bo‘lsa funksiyaning qiymati nolga teng, aks holda nechanchi pozisiyadan boshlangan bo‘lsa, shu son funksiyaning qiymati bo‘ladi. Agar ***1_satr - 2_satr*** ning tarkibida bir necha marta uchrashi mumkin. Bu holda funksiya faqat birinchi marta uchrash pozisiyasini aniqlaydi halos.

Misol: *program tek;*

var a:integer;

```
        b,s:string;
begin
        b:= 'Gulbaxor'; s:= 'baxo';
        a:=pos(s,b);
        writeln(a)
end.
```

Ushbu dastur 4 natijani beradi.

6.4. XARFIY TIPDAGI QIYMATLI BUYRUQLAR

Consat(satr1, satr2, …, satrN) buyrug‘i ***satr1, satr2, …, satrN*** satrli ma’lumotlarni bir-biriga ko‘rsatilgan ketma-ketlikda ulash (birlashtirish) uchun xizmat qiladi. Masalan: a:= *’Paskal’* va *b:= ’tili’* bo‘lsa, *sonsat(a,’ ‘,b)* buyrug‘ining natijasi qiymati *‘Paskal tili’* bo‘lgan yaxlit bitta satrdan iborat bo‘ladi.

Str(a,satr) - buyrug‘i ***a***-sonli kattalikni ***satr*** - satrli ma’lumotga aylantirish uchun ishlatiladi. Agar ***a***-butun son bo‘lsa, ***satr*** - ning qiymati shu sonni tashkil qiluvchi raqamlar ketma-ketligidan iborat bo‘ladi. ***a***-haqiqiy bo‘lganda esa, satr –ning qiymati shu sonning normal ko‘rinishidan olinadi. Masalan: ***str(123.45, x)*** bo‘lsa, ***x*** - ning qiymati ‘1.2345e+02’ ga teng bo‘ladi. Zarur bo‘lganda sonli ma’lumotga o‘lcham e’lon qilinishi mumkin. Bu xolda ***satr***-ning uzunligi shu o‘lchamga teng bo‘ladi. Masalan: ***str(123.45:20, x)*** uchun ***x***-ning qiymati ‘ 1.2345000000e+02’ ga teng.

Upsase(x) –funksiyasidan kichik lotin xarflarini katta lotin xarflari bilan almashtirish uchun foydalaniladi. *Upsase*(‘Paskal’) funksiyasining qiymati ‘PASKAL” dan iborat bo‘ladi. Bu funksiya boshqa alifbodagi xarflarga ta’sir qilmaydi.

Copy(A, N, M)-funksiyasi A satrning N – chi belgisidan boshlab M ta belgisini ajratib olish vazifasini bajaradi va umumiy holda

S:= sopy(a,n,m);

ko‘rinishida yoziladi. Copy(‘Gulbaxor’,4,3) funksiyasi‘bax’ qiymatini qabul qiladi. Agar M soni A-satrning uzunligidan katta bo‘lsa, u xolda A-satrning N-chi belgidan keyingi qismi funksiyaning qiymati hisoblanadi.

Delete(A, N, M)-buyrug‘i A-satrning N-chi belgisidan boshlab M ta belgini o‘chirish uchun ishlatiladi va umumiy holda

Delete(A, N, M);

tarzida yoziladi. ***Delete(‘Paskal’, 4, 2)*** buyrug‘ining natijasi ***‘Pasl’*** bo‘ladi. N soni A-satrning uzunligidan katta bo‘lsa, u holda A-satrning belgilari o‘chirilmaydi. Aks holda agar $M + N$ miqdor satrning uzunligidan katta bo‘lsa, u holda A-satrning N –chi belgisidan boshlab xamma belgilari o‘chiriladi.

Insert(1_satr, 2_satr, N) – buyrug‘idan ***1_satr*** – ni ***2_satr*** ga uning N –chi

belgisidan boshlab qo‘shish maqsadida foydalaniladi. Agar N soni ***2_satr*** uzunligidan katta bo‘lsa, u holda ***1_satr 2_satr*** ning oxiriga qo‘shiladi.

```
Program kushish;
    var a,b:string;
begin
    a:= 'O‘zbekiston buyuk davlat';
    b:= 'kelajagi ';
    insert(b,a,12);
    writeln(a)
end.
```

dasturi ‘Uzbekiston kelajagi buyuk davlat’ natijasini ekranga chiqaradi.

§-7. O‘TISH, TARMOQLANISH VA TANLASH BUYRUQLARI

7.1. MANTIQIY IFODALAR

Mantiqiy ifoda qiymati «rost» yoki «yolg‘on» bo‘lishi mumkin bo‘lgan ilmiy-nazariy va hayotiy mulohazalar, soddaroq qilib aytilganda turli shartlardan iborat bo‘ladi.

Inson hayoti davomida doimo qandaydir masalalarni hal qilishga xarakat qiladi. SHu masalalarni Yechish jarayonida u mumkin bo‘lgan turli mulohazalar va ularning oqibatlarini hisobga olgan holda u yoki bu ishga qo‘l uradi. Masalan, ishga otlanayotgan kishi ertalab uydan chiqishidan oldin «hozir kuchli yomg‘ir yog‘moqda» mulohazasini hayolan tahlil qiladi va soyabonni o‘zi bilan olish olmaslik masalasini hal qiladi. «Hozir harorat 20^0 dan yuqori» mulohazasining natijasi uning kiyadigan kiyimlarini belgilab beradi.

Mantiqiy mulohaza natijalarini EHM yordamida tahlil qilish uchun ularni Paskal tilida maxsus usulda ifodalash lozim. Buning uchun munosabat belgilaridan foydalaniladi. Ularni yozish tartibi quyidagicha:

munosabat	Paskalda	oddiy yozuv	Paskalda
$=$	=	$x^2 = 10$	$x * x = 10$
$\neq$	<>	$x \neq 5$	$x <> 5$
$>$	>	$x^2 - 3x > 0$	$x * x - 3 * x > 0$
$\geq$	>=	$5x \geq 7$	$5 * x >= 7$
$<$	<	$x^2 - 4ac < 0$	$x * x - 4 * a * c < 0$
$\leq$	<=	$ab \leq cd$	$a * b <= c * d$

Ikki va undan ortiq hartdar yoki mantiqiy mulohazalarni tekshirish uchun mantiqiy «xa» (bir nechta mulohazalarni bir vaqtda o‘rinli bo‘lishi), «yoki» (bir nechta mulohazalardan kamida bittasini o‘rinli bo‘lishi) hamda «emas» (mulohazaning teskarisi) kabi amallardan foydalanish mumkin. Bu mulohazalarni

tekshirishda mantiqiy ifodalar qavslar ichida ko‘rsatilishi sharti.

1. («Hozir yomg‘ir yog‘moqda») va («harorat 20^0 dan past»).
2. («Hozir yomg‘ir yog‘moqda») yoki («harorat 20^0 dan past»).
3. emas («Hozir yomg‘ir yog‘moqda»).

1-mulohaza faqatgina har ikki shart o‘rinli bo‘lgandagina «rost» qiymatini oladi. Qolgan hamma hollarda «yolg‘on» bo‘ladi. 2-esa ikki mulohazadan kamida bittasi «rost» bo‘lganda «rost» qiymatini oladi. «emas» amali qavs ichidagi mulohazaning natijasini teskarisiga almashtiradi, ya’ni yomg‘ir yog‘ayotgan bo‘lsa «yolg‘on», aks holda «rost» qiymatiga ega bo‘ladi.

Mantiqiy amallarni Paskal tilida yozish uchun “AND” (va), “OR” (yoki) va “NOT” (emas) hizmatchi so‘zlaridan foydalaniladi. A va B mantiqiy mulohazalar berilgan bo‘lsin. Bu mulohazalar uchun mantiqiy amallar natijasi quyidagicha:

A	B	A and B	A or B	not B
rost	rost	rost	rost	yolg‘on
rost	yolg‘on	yolg‘on	rost	rost
yolg‘on	Rost	yolg‘on	rost	yolg‘on
yolg‘on	yolg‘on	yolg‘on	yolg‘on	rost

7.2. TARMOQLANISH BUYRUG‘I

Hozirgacha keltirilgan dasturlarimiz chiziqli dasturlar toifasiga kiradi. Ulardagi buyruqlar ko‘rsatilgan tartibda bajariladi. bunday holat xar doim ham bizni qanoatlantiravermaydi.

Ko‘pincha biror mantiqiy ifodaning qiymatiga («rost» yoki «yolg‘on» bo‘lishiga) qarab, ikki yoki undan ortiq yo‘nalishlardan birini tanlab olishga to‘g‘ri keladi.

Faraz qilaylik, korxona o‘z xodimlariga ularning bir oyda ishlagan umumiy ish soatlariga qarab ish haqi to‘lashga shartnoma tuzgan bo‘lsin. Bu shartnomada 200 soatgacha ishlagan xodimlarga bir hil ish haqi, 200 soatdan ortiq ishlagan xodimlarga esa 200 soatdan ortiq ishlagan xar bir soati uchun ikki barobar ish haqi to‘lash ham ko‘rsatib o‘tilgan bo‘lsin. Xodimning bir soatlik ish haqi *haq*, bir oyda ishlagan umumiy ish soati *ish_soat* bo‘lsin. U holda 200 soatdan ko‘p ishlagan xodimlarning maoshlari

$$ish\_haqi := haq * 200 + (ish\_soati - 200) * haq * 2;$$

formula bilan topiladi. 200 va undan kam ishlagan xodimning ish haqini hisoblash uchun bu formuladan foydalanib bo‘lmaydi. CHunki, 200 soatdan k am ishlagan xodim «haqdor» bo‘lish o‘rniga «qarzdor» bo‘lib qolishi ham mumkin. Bu holatni oldini olish uchun dasturchi har ikki holatni hisobga olishi lozim. Ushbu masala quyidagi algoritm yordamida hal qilinadi: dastlab xodimning ishlagagn umumiy ish

soati, so‘ngra "$ish\_soati \geq 200$" mantiqiy ifodaning qiymati aniqlanadi. Agar u «rost» bo‘lsa, xodimning ish haqi

$$ish\_haqi := haq * 200 + (ish\_soati - 200) * haq * 2,$$

aks holda esa

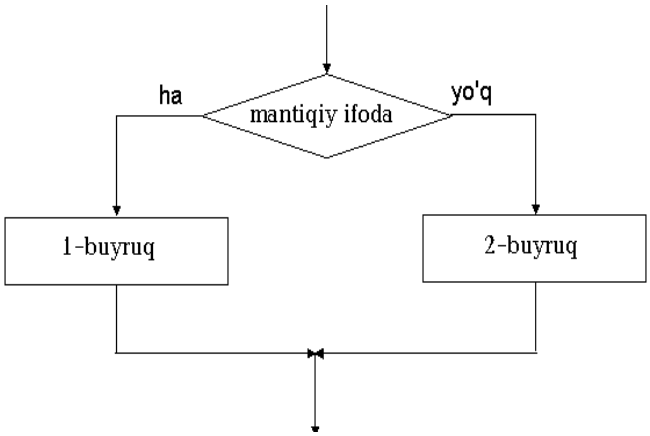
$$ish\_haqi := haq * ish\_soati$$

formula bilan topiladi.

Paskal tilida bunday jarayonlarni ifodalash uchun tarmoqlanish buyrug‘i deb ataladigan buyruq kiritilgan. U umumiy ko‘rinishda quyidagicha yoziladi:

if mantiqiy ifoda ***then*** 1-buyruq ***else*** 2-buyruq ; .

Bu buyruqni bajarishda EHM dastlab «mantiqiy ifoda» ning qiymatini aniqlaydi. Agar u «rost» bo‘lsa 1-buyruqni bajaradi, 2-buyruqni esa bajarmaydi. Agar «yolg‘on» bo‘lsa, aksincha, ya’ni 2-buyruqni bajarib, 1-ni bajarmaydi. (Tarmoqlanish buyrug‘ining bajarilish tartibi ko‘rsatilgan blok-sxemaga e’tibor bering.) SHundan keyingina ***if*** dan keyingi buyruqni bajarishga o‘tadi. Tarmoqlanish buyrug‘idan foydalanib yuqoridagi jarayonni quyidagicha yozish mumkin:



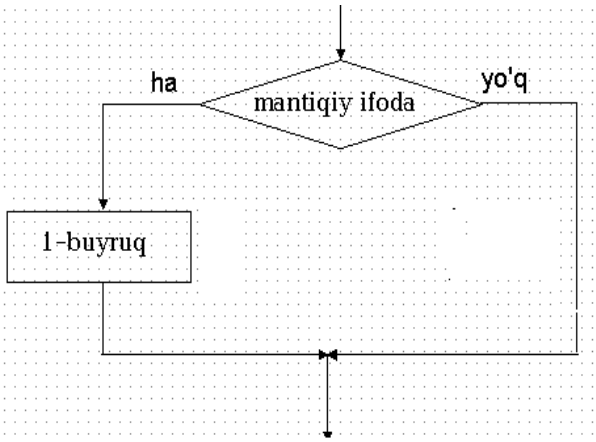
if $ish\_soati <= 200$ ***then*** $ish\_haqi := haq * 200 + (ish\_soati - 200) * haq * 2$

else $ish\_haqi := haq * ish\_soati$;

Agar ehtiyoj bo‘lsa, tarmoqlanish buyrug‘ining qisqartirilgan variantidan ham foydalanish mumkin. U umumiy holda quyidagicha yoziladi:

if mantiqiy ifoda ***then*** 1-buyruq; .

Bu yerdagi 1-buyruq mantiqiy ifoda «rost» qiymat olgan holdagina bajariladi. Aks holda hech bir amal bajarilmaydi va ***if*** dan keyingi buyruqqa o‘tiladi. (Bu buyruqqa mos keladigan blok-sxemaga qarang.)



Yuqoridagi jarayon tarmoqlanish buyrug‘ining to‘liqsiz varianti yordamida quyidagicha ifodalanishi mumkin:

$ish\_haqi := haq * ish\_soati$;

if $ish\_soati <= 200$ ***then*** $ish\_haqi := haq * 200 + (ish\_soati - 200) * haq * 2$;

Bu buyruqlar ketma-ketligini quyidagicha izohlash mumkin: Hodimlarning ish haqlari hamma uchun bir hil, ya'ni $ish\_haqi := haq * ish\_soati$ formula bilan hisoblanadi. Qo'shimcha ish haqini esa faqat bir oyda 200 soatdan ortiq umumiy ish soatiga ega bo'lgan hodimlargina oladi.

Ikki va undan ortiq mulohazalardan "and", "or" va "not" mantiqiy amallaridan foydalanib ifodalar tuzish mumkin. Bunda bir mulohaza qavslar ichida ko'rsatilishi shart. Quyidagi misollarga e'tibor bering:

1-misol: ***if*** $(x >= 2)$ ***and*** $(x <= 5)$ ***then*** *write* ('x soni [2, 5] ga tegishli')
else *write* ('x soni [2, 5] ga tegishli emas');

2-misol: ***if*** ('bugun yakshanba') ***or*** ('ish vaqti tugadi')
then *writeln* ('dam oling') ***else*** *writeln* ('ish bilan shugullaning');

3-misol: ***if not*** $(x >= 2)$ ***then*** *write* ('x soni 2 dan katta emas')
else *write* ('x soni 2 dan katta'); .

Yuqoridagi barcha mulohazalarni jamlagan holda, ish haqini hisoblash dasturini quyidagicha yozish mumkin:

```
program maosh;
        var ish_haqi, ish_soati, haq : integer;
begin
        writeln('hodim nesha soat ishladi ?'); readln(ish_soati);
        writeln('bir soat ushun qansha haq oladi ?'); readln(haq);
        ish_haqi := ish_soati * haq;
        if ish_soati >= 200 then
          ish_haqi := haq * 200 + (ish_soati - 200) * haq * 2;
        writeln('hodim ',ish_haqi,' sum oladi');
end.
```

Yuqoridagi dastur uchun EHM quyidagi natijalarni beradi: (? belgisidan keyin klaviatura orqali kiritilgan ma'lumotlar keltirilmoqda.)

```
hodim nesha soat ishladi ?  150
bir soat ushun qansha haq oladi? 100
hodim 150000 sum oladi

hodim nesha soat ishladi ?  300
bir soat ushun qansha haq oladi? 100
hodim 400000 sum oladi
```

Ko'pincha ***then*** yoki ***else*** xizmatchi so'zlaridan keyingi operatorlar tarkibida boshqa ***if*** buyrug'i ham kelishi mumkin. Buni ichki tarmoqlanish deyiladi. Ichki tarmoqlanishda ichki ***if*** o'zidan avval kelgan va eng yaqin turgan ***then (else)*** ga

ta’aluqli bo‘ladi. Quyidagi dasturga e’tibor bering;

```
program masala;
    var x: integer;
    s: string;
begin
    writeln(‘x ning qiymatini kiriting‘); readln(x);
    s:= ’x soni 2 dan kichik’);
    if x>=2 then if x<=5 then s:= ’x soni [2, 5] oraliqqa tegishli’
                         else s:= ’x soni 5 dan katta’;
    writeln(s)
end.
```

Bu dastur uchun EHM quydagi natijalarni beradi:

```
x ning qiymatini kiriting  4
x soni [2, 5] oraliqqa tegishli
x ning qiymatini kiriting  7
x soni 5 dan katta
x ning qiymatini kiriting  1
x soni 2 dan kichik
```

Agar ichki tarmoqlanishlar ko‘p bo‘lsa, ularni yozish va o‘qish ishlarida dasturchi uchun bir muncha noqulayliklar kelib chiqishi mumkin. Buni oldini olish uchun PASKAL tiliga tanlash buyrug‘I kiritilgan.

7.3. TAMG'ALAR. SHARTSIZ O‘TISH BUYRUG'I

Ayrim bir buyruqlarni boshqalaridan biror belgi bilan ajratib qo‘yilishi mumkin. Ana shu belgini tamg‘a deb ataymiz.

Tamg‘a sifatida butun son yoki lotin alifbosidagi xarflarni olish mumkin. Tamg‘adan so‘ng ikki nuqta (:) belgisi qo‘yiladi. Masalan:

```
10: x:=2;
‘s’: d:=x*x-4*a*c;
```

SHundan keyin dasturning biror eridan turib, shu buyru=a murojaat qilish mumkin. Buning uchun shartsiz o‘tish buyrug‘i bo‘lgan ***goto*** dan foydalanamiz. Bu buyruq umumiy holda

goto tamg‘a ;

ko‘rinishida yoziladi.

SHartsiz o‘tish buyrug‘i ***goto*** dan keyin ko‘rsatilgan tamg‘ali buyruqqa o‘tiladi.

Dasturda qatnashgan barcha buyruqlar odatda dasturning ikkinchi satrida, ***label*** xizmatchi so‘zidan keyin keltiriladi.

Avvalgi bo‘limda ish haqini hisoblash dasturi bitta xodim uchun yozilgan edi.

Ana shu korxonada 3 ta xodim ishlab, ular 1 dan 3 gacha tartib raqamlari bilan nomerlangan bo‘lsin. Dasturdagi amallar ketma-ketligini xodimlarning hammasi uchun bajarilishini ta’minlaydigan dastur quyidagicha yoziladi:

```
Program utish;
        label 2;
         var ish_haqi, ish_soati, haq, tab_nom : integer;
begin
        tab_nom:=1;
  2 :   writeln('hodim nesha soat ishladi ?'); readln(ish_soati);
        writeln('bir soat ushun qansha haq oladi ?'); readln(haq);
        ish_haqi := ish_soati * haq;
        if ish_soati >= 200 then
          ish_haqi := haq * 200 + (ish_soati - 200) * haq * 2;
        writeln(tab_nom, '-hodim ',ish_haqi,' sum oladi');
        tab_nom:=tab_nom+1;
        if tab_nom<=3 then goto 2;
end.
```

```
hodim nesha soat ishladi ?  150
bir soat ushun qansha haq oladi? 50
1-hodim 7500 sum oladi

hodim nesha soat ishladi ?  300
bir soat ushun qansha haq oladi? 100
2-hodim 40000 sum oladi

hodim nesha soat ishladi ?  200
bir soat ushun qansha haq oladi? 125
3-hodim 25000 sum oladi
```

Bu dastur yordamida xuddi shunday tarzda ixtiyoriy sondagi xodimlarning ish haqlarini hisoblani tashkil qilish mumkin. Buning uchun dasturdagi 3 sonini xodimlar soni bilan almashtirish lozim bo‘ladi.

7.4. TANLASH BUYRUG'I

Case buyrug‘i bir nechta imkoniyatlardan birini tanlab olish uchun mo‘ljallangan buyruq bo‘lib, ichki tarmoqlanishlarni chiroyli va tushunarli qilib yozish maqsadida foydalanilishi mumkin. Bu buyruq umumiy ko‘rinishda

quyidagicha yoziladi:

Case ifoda ***of***

1-tamg‘a: 1-operator;

....

n-tamg‘a: n-buyruq ;

end;

Case **va** ***of*** xizmatchi so‘zlari orasida ko‘rsatilgan ifodani selektor, ya’ni tanlash ifodasi deb ataladi. Buyruq quyidagicha bajariladi: dastlab selektorning qiymati hisoblanadi. So‘ngra ana shu qiymatga mos keladigan tamg‘a qidirib topiladi va shu tamg‘a ostidagi operatorlar ketma-ketligi bajariladi. Agar izlangan tamg‘a topilmasa, ***case*** ga taalluqli ***end*** operatoridan keyin ko‘rsatilgan operatorlar ketma-ketligi bajariladi.

<u>*Masala:*</u> Sportchi musobaqada A – o‘rinni egalladi. U qanday medal olganini aniqlang.

```
Program sport;
    label 1,2,3,4;
    var a: integer;  s : string[16];
begin
    write('sportchi nechanchi urinn oldi?');
    readln(a);
    s:= 'sportchi medal olmagan'
    case a of
        1: s:= 'oltin';
        2: s:= 'kumush';
        3: s:= 'bronza';
    end;
    writeln(s)
end.
```

EHM bu dastur uchun quyidagi natijani beradi:

sportchi nechanchi urinn oldi? 5
sportchi m,edal olmagan

sportchi nechanchi urinn oldi?1
oltin

Selektor ifodaning qiymati *real* bo‘lmagan ixtiyoriy skalyar tipda bo‘lishi

mumkin. Tamg‘alar ham ana shu tipga oid bo‘lib, bitta tamg‘a bitta *case..of* buyrug‘ida bir necha marta kedmasligi kerak.

Tanlash buyrug‘idan odatda biron bir oraliqqa tushmaydigan qiymatlar uchun ma’lum bir buyruqlarni bajarilishi yoki bitta ifodaning qiymati bir nechta ma’lumotlardan biriga bog‘liq bo‘lgan holatlarda foydalanilsa yaxshi samara beradi.

```
Program baho;
        label 0,1,2,3,4,5;
        var a: integer; s: string[9];
begin
        write(‘bahoni kiriting‘);
        readln(a);
        case a of
                5:  s:= ’alo‘;
                4:  s:= ’yaxshi’
                3:  s:='qoniqarli' ;
        0,1,2:  s:='yomon' ;
       end;
         wriieln(s)
end.
```

EHM bu dastur uchun quyidagi natijalarni beradi:

Bahoni kiritng 4 yahshi

Bahoni kiritjng 1 uomon

§-8. SIKLLARNI TASHKIL QILISH.

8.1. WHILE OPERATORI.

Ko‘plab masalalarni Yechish jarayonida ayrim bir amallar ketma-ketligini takror va takror ko‘rsatishga va demak bajarishga to‘g‘ri keladi. 4.3- masalani esga oling. U erda bir nechta buyruqlarni 3 ta ishchisi bor korxona uchun 3 marta bajarishga to‘g‘ri keladi. Dasturning ichida bir necha marta bajariladigan buyruqlar ketma ketligi tsikl deyiladi.

TURBO PASKAL da tsikllarni tashkil qilishning bir necha usullari mavjud. Ulardan biri *while* operatoridir. *While* (inglizcha so‘z-"toki") buyrug‘i umumiy ko‘rinishda quyidagicha yoziladi:

while (mantiqiy ifoda *) do*
begin
1-operator ; 2-operator ; ... p-operator ;
end ;

Agar tsikl bitta operatordan iborat bo‘lsa, u quyidagicha yoziladi:

While (mantiqiy ifoda***) do*** operator ;

While operatorining bajarilish tartibi mana bunday: dastlab *while* buyrug‘idan keyin qavslar ichida ko‘rsatilgan mantiqiy ifoda qiymati hisoblanadi. Agar uniig qiymati "rost" bo‘lsa tsikldagi operatorlar ketma-ketligi bir marta bajariladi, so‘ngra yana mantiqiy ifodaning qiymati hisoblanadi. Qiymat yana "rost" bo‘lsa, tsikl yana bir marta bajariladi va hokazo. Bu jarayon toki mantiqiy ifodaning qiymati "yolg‘on" bo‘lib qolguncha bajarilave-radi. Agar mantiqiy ifodanipg qiymati birinchi marta tekshirilgan paytdayoq "rost" qiymatini qabul qilmasa, ya’ni o‘rinli bo‘lmasa, tsikldagi buyruqlar ketma-ketligi bir marta ham bajarilmaydi.

```
Program salomlashish;
   var marta : integer ;
begin
write('necha marta salomlashamiz ?') ;
readln( marta ) ;
 while marta>0 do begin readln('SALOM') ;
 marta:= marta-1; end ;
writeln( 'HAYR, JANOB!) end.
```

EX,M bu dastur uchun quyidagi natijalarni beradi:

```
Necha niarta salomlashamiz 3
SALOM
SALOM
SALOM
HAYR, JANOB
Necha marta salomlashamiz? 0
HAYR, JANOB
```

Odatda *while* operatoridan takrorlanishlar soni oldindan noma’lum bo‘lgan hollarda foydalanish maqsadga muvofiq hisoblanadi.

Masala-1. A va V natural sonlash berilgan bo‘lsin. Ularning eng katta umumiy bo‘luvchisini toping.

Yechish g‘oyasi: Bu masalani Yechish "Evklid algoritmi" asosida amalga oshiriladi. Bu algoritmga kg‘ra toki A va V sonlari bir-biriga teng bo‘lib qolmaguncha, kattasidan kichigi ayrilaveradi. Xar gal ayirma shu sonlardan kattasini ifodalab turran o‘zgaruvchi nomi bilan belgilab qo‘yiladi. Bu jarayon A=V bo‘lganda to‘xtaydi. Echim sifatida A yoki V ni olish mumkin.

```
Program Yevklid ;
```

```
        var a,b : integer ;
    begin
      readln(a,b); while a<>b then a:=a-b else b:=b-a;
write ( 'EKUB= ', a )
end.
```

Ushbu dasturni a=60 va b=36 uchun bajargan EOM

EKUB = 12

ko‘rinishidagi natijani displeyga chiqaradi.

Eslatma: Mantiqiy ifodani o‘rinli bo‘lish yoki bo‘lmasligiga qarab, *while* tsikli bir marta ham bajarilmasligi mumkin.

8.2. REPEAT ... UNTIL OPERATORI.

TSikllarni tashkil qilishning boshqa bir usuli *iepeat-until* operatoridir. U umumiy ko‘rinishda quyidagicha yoziladi:

```
repeat
        1-buyruq; 2-buyruq; ...; p-buyruq;
until (mantiqiy ifoda);
```

TSiklni bu ko‘rinishda tashkil qilinganda mantiqiy ifoda tsiklning oxirida tekshiriladi. Uning qiymati "yolg‘on" bo‘lsa, tsikl yana bir marta bajariladi, so‘ngra yana mantiqiy ifoda tekshiriladi. Bu jarayon toki mantiqiy ifodaning qiymati "rost" bo‘lib kolmaguncha takrorlanaveradi. By usulda tsikl tashkil qilishnish avvalgisidan farqi shuki, mantiqiy ifodaning qiymati tsiklning oxirida tekshirilishi hisobiga tsikl hech bo‘lmaganda bir marta bajariladi. 5-1 dagi dasturni

```
program salomlashish;
        var marta ; integer ;
    begin
        write('Necha marta salomlashamiz  ?') ;
        readln( marta ) ;
        repeat
        writeln(‘SAL       OM’) ;
        marta:=marta-l;
        until marta<=0 ;
        writeln( 'HAYR, JANOB')
    end.
```

tarzida ham yozish mumkin. By dasturni EXM bajarganda, displeyga quyidagi yozuvlar chiqariladi:

Necha marta salomlasnanuz ? 3

SALOM

SALOM
SALOM
HAYR, JANOB
Necha marta salomlashamiz ? 0
SALOM
HAYR, JANOB

Repeat...until operatoridan takrorlanishlar soni oldindan noma'lum bo‘lgan hollarda foydalanish yaxshi samara beradi.

<u>*Masala.*</u> Fibonachchi sonlari $F_0 = F_1 = 1$ va $F_n = F_{n-1} + F_{n-2}$ formulalar bilan topiladi. 1000 dan katta bo‘lgan birinchi Fibonachchi sonini toping.

<u>*Yechish g‘oyasi:*</u> Formulalardan ko‘rinib turibdiki. topilayotgan yangi had o‘zidan avvalgi ikki hadnipg yig‘indisiga teng ekan. SHuning uchun topilayotgan yangi xadni F_2, undan oldin keladigan ikki had uchun mos ravishda F_0 va F_1 degan belgilash qabul qilinadi. Unga ko‘ra, $F_2 = F_1 + F_0$ formula bilan yangi had topiladi. Undan keyingi hadni topish uchun, yuqoridagi belgilashga ko‘ra, topilayotgan yangi had ham F_2 bo‘lishini hisobga olinsa, $F_1 = F_2$ va $F_0 = F_1$ almashtirishlarni bajarish kerak. Bu jarayonni to $F2 > 1000$ bo‘lguncha davom ettirish talab qilinadi. Bu fikrlarni nazarda tutib, dastur yozamiz:

```
program fihonachchi ;
   var f0, f1, f2 : integer ;
begin
      f0:=1 ;  f:= 1 ;
      repeat
      f2:=f1+f0 ;
      f0:=f1; f1:=f2;
      until f2<1000 ;
      writeln ('izlangan son:= ', f2)
end.
```

Bu dastur uchun EHM quyidagi natijani beradi :
izlangan son:= 1597

<u>Eslatma</u>: until...repeat tsikli kamida bir marta bajariladi.

8.3. FOR OPERATORI

Ko‘plab tsikllarni biron bir o‘zgaruvchining m, m+1, ..., N bo‘lgan qiymatlarida bajarishga to‘g‘ri kelib qoladi. Ana shunday jarayonlarni ixcham qilib yozish maqsadida yangi tsikl operatori *for* kiritilgan. Bu buyruq umumiy ko‘rinishda quyidagicha yoziladi:

for x:=xl to x2 do begin

1-operator ; ..., p-operator ;

end;

Agar tsiklda takroran bajarilishn talab qilingan operator bitta bo‘lsa, *for* operatorini mana bunday yozish mumkin :

for x:=xl to x2 do operator ;

Bu erda x tsikl jarayonni boshqaradigai o‘zgaruvchi, xl-x ning boshlang‘ich qiymati, x2-esa x ning oxirgi qiymati. *For* buyrug‘i birinchi marta bajarilganda, x-o‘zgaruvchiga xl boshlang‘ich qiymat beriladi. So‘ngra x-nnng ana shu qiymati uchun tsikl bir marta bajariladi. Keyin x o‘zgaruvchiiing qiymati 1 ga orttiriladi va x ning yangi qiymatini x2 bilan solishtiriladi. Agar $x \le x2$ shart o‘rinli bo‘lsa, tsikl yana bir marta bajariladi va hokazo. Bu jarayon to x>x2 shart o‘rinli bo‘lib qolguncha davom ettirilaveradi. *For* tsikli hech bo‘lmaganda bir marta bajariladi.

<u>*Masala:*</u> $1+2+3+1000$ yig‘indini toshshg.

<u>*Yechish g‘oyasi:*</u> Yig‘indini hisoblab borish uchun S o‘zgaruvchi kiritiladi. So‘ngra i ning 1 dan 1000 gacha bo‘lgan qiymatlari uchun $S = S + i$ ifoda qiymatini topiladi.

```
Program yigindi ;
        var s, i : integer ;
begin
        s:=0;,
        for i:=1 to 1000 do  s := s + i ;
        writeln( 'S:= ', s)
end.
```

EHM bu dastur uchun quyidagi natijani beradi:

S:=500500

YUqoridagi uch xil usul bilan tashkil qilingan tsikllarning bajarish yoki bajarilmasligi *mantiqiy ifoda* ning qiymatiga bog‘liqligi ko‘rinib turibdi.

While tsiklida *mantiqiy ifoda* "rost" bo‘lgan holda bajarilsa, *repeat-until* tsiklida yolg‘on bo‘lsa bajariladi. Demak, har bir tsikl bajarilgan paytda ana shu *mantiqiy ifoda* ning qiymati o‘zgarishi mumkin. Lekin *while* tsikli jarayonida ana shu ifodaning qiymati o‘zgarmasa va birinchi marta *mantiqiy ifoda* "rost" qiymatini olsa, (*repeat...until* tsikli uchun "yolg‘on" qiymatini olsa) tsikl "cheksiz" marta bajariladi. Bunday tsikllarni "cheksiz tsikllar" deyiladi. Uni bajarayotgan EHM ning ishi hech qachon tugamaydi. Dastur tuzishda mana shunday hollarni nazarda tutish lozim. Quyidagi dasturda tsikl har gal bajarilganda, *k* ning qiymati 2 ga ortib boradi va $k = 12$ bo‘lganda tsikl bajarilishi tugaydi.

```
Program cheksiz_cikl;
        var k: integer;
 begin
         k:=2 ;
        while k<>12 do k:=k+2 ;
         writeln(k)
 end.
```

Lekin k:=2 buyrug‘i k:=1 bilan almashtirilsa, k-ning qiymati hech qachon 12 ga teng bo‘lmaydi. Demak dasturning bajarilishi hech qachon tugamaydi, ya’ni "cheksiz tsikl" hodisasi ro‘y beradi.

Ayrim hollarda tsiklning bajarilish jarayonini oxirigacha tugatmay turib, to‘xtatishga to‘g‘ri keladi. Bu ishni *exit* yoki *halt* buyruqlari bilan amalga oshirish mumkin. Agar dasturda *exit* yoki *halt* buyruqlaridan biri uchrab qolsa, EHM faqat tsikl jarayoninigina emas, balki dasturning keyingi buyruqlarini ham bajarishni to‘xtatadi.

```
Program tuxtatish;
         var s,k: integer;
 begin
       for k:=1 to 10 do begin s:= s+k ; exit;
       writeln(‘yigindi:=’,s) ; end;
       writeln('sikl tugarait qadam k:=’,k);
 end.
```

Berilgan dastur uchun EHM hech qanday natija bermaydi va dastur ishini to‘xtatib qo‘yadi.

TSikl buyrug‘i o‘z ichiga ko‘plab buyruqlarni olishi mumkin. SHu jumladan bitta tsikl ichida boshqa bir tsikl ham kelishi mumkin. Ana shunday tsikllarni ichma-ich joylashgan tsikllar deb ataladi. Bunda tsikl buyrug‘ini o‘z ichiga olgan tsiklni tashqi, ichkaridagisini esa ichki tsikl deb yuritiladi. Tashqi tsiklning boshqarayotgan o‘zgaruvchining bir marta o‘zgarishi uchun ichki tsikl to‘la bir marta bajariladi. (Soatning soat va minut kursatkichlarini esga oliig. Unda soatning soat milini bir marta o‘zgarishiga minut mili 0 dan 60 gacha bir marta to‘la aylanib chiqishi mos keladi. Bu erda soat milini tashqi tsiklning boshqaruvchi o‘zgaruvchi deb qarasak, minut milini ichki tsiklning boshqaruvchi o‘zgaruvchisi sifatida qarash mumkin.) TSikllarning boshqa buyruqlari boshqaruvchi o‘zgaruvchilarniig ana shu qiymatlari uchun bajariladi. Keltirilayotgan dasturda 1 dan 10 gacha bo‘lgan sonlarni dastlab 1 ga, 2 ga, 3 ga va 4 ga ko‘paytirish natijasida hosil bo‘ladigan sonlarning yig‘indisi topilmoqda.

```
Program yig‘indi ;
        var i, j, s: integer ;
 begin
        for i:=1 to 4 do begin s:=0 ;
        for j:=1 to 10 do s:=s+j*i ;
         writeln (i, ' ga kupaytirib qoshsak’,s); end ;
 end.
```

*B*y dasturni bajargan EHM quyidagi natijaga beradi:

1 ga kupaytirib qoshsak 55
2 ga kupaytirib qoshsak 1 1 0
3 ga kupaytirib qoshsak 165
4 ga kupaytirib qoshsak 220

§-9. MASSIVLAR

9.1. MASSIVLAR VA ULARDAN FOYDALANISH

Jadval kattaliklari yoki massivlar bir xil tipdagi va ko‘plab sondagi ma’lumotlarni saqlash hamda qayta ishlash uchun mo‘ljallangan. Masalan: familiyalar ro‘yxati, imtixondan talabalarni olgan baholari, kundalik o‘rtacha xarorat va hokazolarni massiv sifatida qabul qilish mumkin. Faraz qilaylik, 5 ta bahoni o‘qish, ular ichidagi eng katta bahoni topish va qolgan hamma baholarning undan qanchaga farq qilishini topish talab qilingan bo‘lsin. By dasturda 5 ta bahoning hammasini kiritib bo‘lmaguncha, ularning eng kattasi va boshqa baholarni undan qanchaga farq, qilishini topib bo‘lmaydi. Buning uchun hamma baholarni EHM xotirasida saqlashga to‘g‘ri keladi. Turgan gapki, baholarning hammasi *integer* tipida bo‘ladi. Ularni saqlash uchun *integer* tipidagi 5 ta turli o‘zgaruvchilarni kiritish mumkin. Baholarni bildiradigan o‘zgaruvchilarni boshqalari bilan almashtirib qo‘ymaslik uchun VANO1, VANO2, BANO3 yoki shunga o‘xshash qilib tanlash maqsadga muvofiq hisoblanadi. Agar baholar soni ko‘p bo‘lsa (aytaylik 100 ta bo‘lsa), bu usul ham yaxshi natija bera olmaydi. O‘zgaruvchilarni bunday tanlash dasturni murakkablashtirib yuboradi. CHunki, dasturda qatnashadigan o‘zgaruvchilar soni qancha ko‘p bo‘lsa, uni o‘qish va tushunish shuncha qiyin bo‘ladi. Bunday holatlariing oldini olish uchun TURBO PASKAL tilida massivlar tushunchasi kiritilgan.

Massivlar quyidagicha e’lon qilinada:

var massiv nomi : *array [A .. B]* of massivning tipi *;*

Bu erda A-massivdagi birinchi element indeksini, V-esa oxirgi element indeksini bildiradi. Massivlarni nomi ikki qismdan iborat bo‘lgan hamda bir hil tipdagi

o‘zgaruvchilar guruhi deb qarash ham mumkin. Nomning birinchi qismi bir guruhdagi hamma o‘zgaruvchilar uchun bir hil bo‘lib, massiv nomidan, ikkinchi qismi esa massiv elementlari diapazonidan iborat. Masalan :

var VAHO [1..5] of integer;

Massivdagi xar bir o‘zgaruvchini massivning elementi, kvadrat qavs ichidagi sonni esa massiv elementining indeksi (turgan o‘rni) deb ataladi. E’lon qilingap xar bir o‘zgaruvchi uchun ma’lum bir hajmda xotira yacheykalari (o‘zgaruvchi tipiga bog‘liq ravishda) ajratiladi va EHM bu yacheykalar adreslarini esda saqlab turadi. Massivlarda esa, faqat birinchi element turgan adresni yodda saqlaydi halos, qolgan elementlar adreslarini ana shu adresga massiv tipini hisobga olgan holda ma’lum bir sonni qo‘shish orqali hosil qiladi. Faraz qilaylik, *integer* tipidagi Z massiv e’lon qilingan bo‘lsin hamda Z[1] element 1000-adresga yozilgan bo‘lsin. *Integer* tipidagi ma’lumot 2 bayt joyni band qilishini bilgan holda EHM Z[2] ni o‘qish uchun Z[1] adresiga 2 ni qo‘shadi va hosil bo‘lgan sondagi adresli, ya’ni 1002-yacheykadagi ma’lumotni o‘qiydi.

Var BAH0 : array [1..5] of integer ;

R, X : array [10.l00] of real ;

yozuvlari 5 ta *integer* tipidagi elementlari bo‘lgan VANO hamda elementlari soni 91 ta bo‘lgan, indekslari esa 10 dan 100 gacha bo‘lgan *real* tipli ikkita R va X massivni e’lon qilmoqda.

Massivning biron bir elementiga murojaat qilish uning nomi va kvadrat qavs ichida shu elementning massivda turgan o‘rnini ko‘rsatish orqali amalga oshiriladi. Masalan: VANO[4], R[50], X[9] kabi.

Massiv elementining indeksi kvadrat qavslar ichida ko‘rsatilgan sonlar orasidan chetga chiqmasligi kerak. VANO[0] yoki VANO[6], shuningdek, R[9] hamda X[101] kabi elementlar mavjud emas. CHunki, bu elementlarni indekslari e’lon qilingan oraliqqa kirmaydi.

Massiv elementlariping indekslari o‘rnida ihtiyoriy tartiblangai tipdagi (masalan *integer)* va qiymati indekslar diapazonidan chetga chiqmaydigan turli ifodalar ham kelishi mumkin. R[12*8-3] bilan R[93] yozuvlari bitta elementni anglatadi.

Massiv elementlari o‘rtasida turli arifmetik amallarni boshqa o‘zgaruvchilar bilan qanday bajarilsa, huddi shunday tartibda bajariladi:

*R[12|:=10*2 ; X[12]:=R[12]/2 ;*

Massivlar elementlarining joylashish tartibiga qarab, ikki hil bo‘ladi: Bir o‘lchovli va ikki o‘lchovli massivlar. Bir o‘lchovli massivning elementlari faqat bitta satr yoki ustun bo‘ylab joylashadi. Ihtiyoriy vektor,

guruhdagi talabalar ro‘yxati, talabalarning bitta imtixondan olgan baholaridan tuzilgan jadvallar bir o‘lchovli massivlarga misol bo‘la oladi.

Ikki o‘lchovli massivlarning elementlari esa ham satrlar bo‘ylab, ham ustunlar bo‘ylab joylashgan bo‘ladi. Ularni umumiy holda

massiv nomi: *array* [al..a2] [bl..b2] *of* tip ;

tarzida e’lon qilinadi. Bu erda al va a2-satr nomerlarining diaiazoni, b1 va b2 ustun nomerlarining diapazoni. Matritsalarni ikki o‘lchovli massiv sifatida qabul qilish mumkin. Ikki o‘lchovli massivlar satrlari va ustunlari tartib raqamlarining o‘zgarish diapazonlarini ko‘rsatish orqali e’lon qilinadi. Masalan:

$$
\begin{array}{llll}
A_1 & B_1\ B_2\ B_3\ B_4\ B_5\ B_6 & C_{11}\ C_{12}\ C_{13}\ C_{14} \\
A_2 & & C_{21}\ C_{22}\ C_{23}\ C_{24} \\
A_3 & & C_{31}\ C_{32}\ C_{33}\ C_{34} \\
A_4 & & C_{41}\ C_{42}\ C_{43}\ C_{44}
\end{array}
$$

kabi jadvallarni quyidagicha e’lon qilish mumkin:

var A: array [1.. 4] of real; V: array 11..6] of real;

C: array [1..4][1..4J of integer ;

va hokazo.

<u>*Masala:*</u> A[l..l00] butun sonli massivda qiymati 7 ga teng elementlar sonini aniqlang.

<u>*Yechish g‘oyasi:*</u> Dastlab 7 ga teng bo‘lgan elementlarni sanash uchun s:=0 o‘zgaruvchi tanlanadi. So‘ngra, massiv elementlarini bittadan kiritiladi. Har bir kiritilgan elementni 7 soni bilan taqqoslanadi. Agar a[i] element 7 ga teng bo‘lsa, s ning qiymatini 1 ga oshiriladi. Keyin navbatdagi elementga o‘tiladi. Bu jarayon 100 marta takrorlanadi. Qo‘yilgan masala uchun dastur quyidagicha yoziladi:

```
program sanash ;
    var s, i : integer ;
    A: array [I.. 100] of integer;
 begin
    s:=0 ;
     for i:=1 to 100 do begin
    readln ( a[i] ) ;
    if a[i]=7 then s:= s+1 ; end ;
    writeln('7 ga teng elementlar soni=',s);
 end.
```

<u>*6.2-masala:*</u> 10 ta haqaqiy elementli massivdagi eng katta va eng kichik elementlarni toping.

Yechish g‘oyasi: Dastlab massivning barcha elementlarini aniqlab olinadi. So‘ngra, birinchi elementni izlangan element deb faraz qilinadi va qolgan elementlar uchun qilingan farazni to‘g‘ri-noto‘g‘riligini tekshiriladi. Agap eng katta (eng kichik) degan elementdan ham kattaroq (kichikroq) element topilib qolsa, izlangan element sifatida uni qabul qilinadi va tekshirishni kelgan eridan ana shu element uchun davom ettiriladi. Bu masalaning dasturi quyidagicha yoziladi :

```
program massiv ;
        var a : array[1.. 10] of real ;
        i : integer ; max,min : real ;
 begin
        for i:=1 to 10 do begin
        write (i, '- element:=') ; readln (a[i]) ; end;
        max:=a[1]; min:=a[1];
        for i:=2 to 10 do
        if a[i]>max then max:=a[i]
              else if min>a[i] then min:=a[i];
         writeln( 'kiritilgan elementlar ichida');
         writeln ( 'eng katta element',max);
         writeln (‘eng kichik element', min)
 end.
```

Bu dastur uchun avval EHM ga massiv elementlari kiritiladi.

1 – element 12
2 - element 10
3 - element 17
4 - element 21
5 - element 19
6 - element 23
7 - element 3
8 - element 12
9 - element 19
10 - element 14

Bu ma’lumotlar uchun EHM bergan natija g‘uyidagicha bo‘ladi:

kiritilgan elementlar ichida
eng katta element 23
eng kichik element 3

String tipidagi ma’lumotlarni *char* tipidagi elementlar massivi sifatida qabul qilish ham mumkin. Masalan:

Var r : string[18] ;

yozuvi R massiv» *char* tipidagi 18 ta elementdan iborat zkanligini bildiradi. r ning biron bir belgisiga murojaat qilish zarur bo‘lsa, uning turgan o‘rni ko‘rsatiladi. Misol uchun:

K:=' MATEMATIKA-GEOMETRIYA'

bo‘lsa, R[10]=’A’ R[16]='E' bo‘ladi. E’tibor bering, agar *string* tipidagi o‘zgaruvchiga belgilanganidan uzunroq so‘zni qiymat qilib berilsa, EHM so‘zning faqat belgilangan uzunlikdagi qismini ajratib oladi, ortiqcha belgilarni esa tashlab yuboradi. SHuning uchun yuqoridagi misoldagi R ning qiymati

‘MATEMATIKA-GEOMETR’

ga teng bo‘ladi.

6.3-masala: 18 tagacha belgisi bo‘lgan W so‘zni teskarisidan o‘qish dasturini yozing.

Yechish g‘oyasi: Dastlab W so‘zni kiritiladi. So‘ngpa, hech qanday belgisi bo‘lmagan bo‘sh S o‘zgaruvchi olinadi. Uning chap tomoniga berilgan so‘zning birinchi belgisidan boshlab, hamma belgilarni bitta-bittadan keltirib yonma-yoi g‘o‘yiladi.

```
Program teskari ;
    var w,s:string[20] ; i:integer ;
 begin
    write( ‘kiritilgan suz…’ );
    for i:=1 to 20 do readln (w[i]);
    s:=’’;
    for i:= 1 to 20 do  s:= w[i]+ s  ;
    writeln('uning teskarisi…’,s) ;
 end.
```

Ushbu dastur uchun EHM quyidagi natijani berdi :

kiritilgan suz… MATEMATIKA GEOMETRIYA

uning teskarisi... YIRTEMOEG AKITAMETAM

4-masala: Guruhdagi 20 talabaning informatika fani bo‘yicha imtixonda olgan baholari berilgan bo‘lsin. Xar bir talabaning bahosi o‘rtacha o‘zlashtirish bahosidan qanchaga farq qiladi ?

*Yechish g‘*oyasi: Har bir talaba olgai baholarni kiritib, umumiy baholar yig‘indisini topiladi. Umumiy yig‘indini talabalar soni 20 ga bo‘lib, o‘rtacha o‘zlashtirish aniqlanadi. Keyin har bir talabaning bahosidan o‘rtacha o‘zlashtirish bahosini ayirib, oradagi farq topiladi.

Program informatika ;

```
var buho : array[1..20] of integer ;
i : integer ; s, farq : real;
begin
for i:=1 to 20 do begin
readln (baho[i]) ;  s := s + baho[i] ; end ;
 s := s / 20;
for i:=1 to 20 do  begin
farq:=baho[i]- s;
write(j, ' talaba uchun farq:=',farq ) ; end;
end.
```

Ushbu dasturni tekshirib ko‘rish o‘zingizga havola qilinadi. Baholarni 1 bilan 5 orasida tanlash tavsiya etiladi.

Eslatma: 1. Massiv e’lon qilishni unutmang. 2. Massiv elementlarini for tsikli yordamida kiritish qulay hisoblanadi. Z. Elementning indeksi belgilangan diapazonidan chetga chiqmasin.

9.2. MASSIV ELEMENTLARINI TARTIBLASH

Massivlar haqidagi masalalar ichida eng ko‘p uchraydigani bu uning elementlarini o‘sish yoki kamayish tartibida tartiblash masalasidir. Uni Yechish usullari ko‘p bo‘lib, ulardan birortasini boshqasidan ustun qo‘yib bo‘lmaydi. Har bir usul elementlarning joylashishiga ko‘ra boshqasidan yaxshi bo‘lishi mumkin.

1-usul. A[1..N] massiv elementlarini o‘sish tartibida tartiblash talab qilingan bo‘lsin.

Berilgan massiv elementlari ichidan eng kichigini topib, uning o‘rnini 1-element bilan almashtiriladi. Demak, 1-element tartiblandi. Endi 2-element tartiblanadi. Buning uchun qolgan elementlar ichidan eng kichigi topilib, uning o‘rnini 2-element bilan almashtiriladi va hokazo. Bu jarayon N-1 marta takrorlanganda massiv elementlarini tartiblash tugaydi. Bu usul uchun dastur quyidagicha yoziladi:

```
program tartiblash ;
const n=100 ;
var  c, min : real ;
i, j, k  : integer;
a : array[1..n] of real;
begin
for i:=1 to n do readln (a[i]);
for i:=1 to n do begin
min:=a[i] ; k:=i;
for  j:=i+1 to n  do
```

```
 if min>a[j] then begin     (*eng kichik element
min:=a[j];                  va uning tartib raqami
      k:=j;  end            aniqlanmoqda*)
      s:= a[i];             (*eng kichik element
      a[i]:=a[k];           bilan tartiblanayotgan
      a[k]:= s ;            element o‘rinlari almashtirilib,
      writeln(a[i]:8:3)  displeyga chiqrilmoqda.*)
      end;
end.
```

Bu usulni eng kichik elementni chiqarish usuli deyiladi.

2-usul. Ko‘piksimon usul. Bu usulning asosiy g‘oyasi engil elementlarni yuzaga chiqarishdan iborat bo‘lib, xuddi suv ichidan chiqayotgan pufakchalarni eslatadi. Engil elementlar "vazni" darajasida borgan sari yuqorilab boradi. Buning uchun birinchi elementdan boshlab, hamma elementlari o‘z yonida turgan element bilan taqqoslanadi. Agar $a[i] > a[i+1]$ bo‘lsa, u holda ular o‘rni o‘zaro almashtiriladi. Tekshirish yana boshidan boshlanadi.

```
Program tartihlash_2 ;
      const n=100 ;
      var i, j : integer; s : real ;
      a : array [1..n j of real ;
begin
      for i:=1 to n do read(a[i]);
      for i:=1 to n-1 do begin
      if a[i] > a[i+1]  then begin
      c:=a[i];              (*engil element oldinga
      a[i]:= a[i+1] ;       o‘tkazilmoqda *)
       a[i+1]:= c ;
      i:=0;                 (*tekshirish boshidai boshlanadi*)
      end;
      writeln(a[i]:8:3); end;
end.
```

<u>Masala.</u> Birinchi xarflari katta, qolgani kichik xarflar bilan yozilgan familiyalardan A[1..N] massiv tuzilgan bo‘lsin. SHu massiv elementlarini alifbo tartibida displeyga chiqaring.

<u>Yechish:</u> Dastur juda sodda bo‘lgani uchun g‘oyasi keltirilmadi.

```
Program tartihlash_3;
      const n=100 ;
```

```
        var a : array[1..n] of string(20);
         h, s : char; i, j : integer ;
begin
        fori:=1 to n do readin (a[i]);
        for b:= 'A' to 'Z' do for c:='a' to 'z' do
        for i:=1 to n do
        if (A[i][1]=b) and (a[i][2]=c
        then writeln(a[i]);
end.
```

10. YANGI TIPLARNI KIRITISH

10.1. ELEMENTLARI CHEGARALANGAN TIPLAR

6.4-masalani EHM yordamida bajarib ko‘ring. U erda kiritilayotgan ma’lumotlar sifatida ihtiyoriy butui sonni (10, 50, 1000 kabi sonlarni ham) talabaning bahosi sifatida kiritishingiz mumkin va EHM u ma’lumotni hech qanday e’tirozsiz qabul qiladi hamda kiritilgan baholar uchun talab qilingan natijalarni topadi. Noto‘g‘ri kiritilgan sonlar hisobiga o‘rtacha baholar 5 dan yuqori yoki noldan past bo‘lishi ham mumkin. Ko‘rinib turibdiki, bu narsa noto‘g‘ri natijaga olib keladi. SHunday holatlarni oldini olish uchun TURBO PASKALda standart bo‘lmagan yangi tiplarni aniqlash va ulardan foydalanish imkoniyatlari yaratilgan.

Bunday tiplardan birinchisi elementlari chegaralangan tiplardir. Bu tip faqat elementlari tartiblangan bazaviy tiplarning ma’lum bir qismini chegaralar yordamida ajratib olish orhali hosil qilinadi va umumiy ko‘rinishda quyidagicha yoziladi:

```
type tip nomi  b_ch..o_ch ;
```

Bu erda b_ch-6oshlang‘ich chegara, o_ch-oxirgi chegara. Bu tipga mansub bo‘lgan o‘zgaruvchilar faqat b_ch-dan o_ch – gacha bo‘lgan oraliqdagi qiymatlarni qabul qila oladi, boshqa hech qanday tsiymatlarni qabul qilmaydi.

Tip aniqlangandan so‘ng, shu tipdagi qiymatlarni qabul qiladigan o‘zgaruvchilar boshqa (standart tipdagi o‘zgaruvchilar kabi e’lon g‘ilinadi. SHundan keyingina ulardan foydalanish mumkin. Masalan: baholarning 1 dan 5 gacha bo‘lishini bilgan holda

```
type V=1..5 ;
```

yangi tipni hosil qilinadi. Endi *V* tipidagi qiymatlarni qabul qiladigan o‘zgaruvchilarni e’lon qilish mumkin :

```
var   baxo : b ;
```

Demak, dasturda qatnashadigan hamma *baxo* o‘zgaruvchilari faqat V tipidagi, ya’ni 1 dan 5 gacha bo‘lgan qiymatlarni qabul qilishi mumkin.

EHM chegaralangan tipdagi o‘zgaruvchilar ko‘rsatilgan doiradagi qiymatlarni olishini nazorat qilib boradi. Agar chegaralangan tipdagi o‘zgaruvchi o‘z chegarasidan tashqaridagi qiymatni oladigan bo‘lsa, dasturning bajarilish jarayoni to‘xtatiladi va yo‘l qo‘yilgan xatolik haqida 'TYPE MISMATH' axboroti beriladi. YAngi tiplarni aniqlashning eng muhim tomoni ham ana shu.

YUqoridagi ma’lumotlarlarni hisobga olib, 6.4-masala dasturining yangi ko‘rinishini keltiramiz:

```
program infor ;
        type baxo=1 .. 5 ;
        var a : array[1.. 20] of baxo :
        i : integer ; s, farq : real;
begin
        for i:=1 to 20 do begin
        readln (a[i]) ;  s := s + a[i]   ; end ;
         s := s / 20 ;
        for i:=1 to 20 do begin  farq:=a[i]-s;
        write(i,'talaba uchun farq:=', farq); end;
end.
```

Eclatma: Elomentlari chegaralangan tipdagi o‘zgaruvchilarning qiymatlarini kompilyator faqat dacturni bajarish davomida nazorat qiladi halos, lekin *read* operatori yordamida ma’lumotlarning to‘g‘ri kiritilishini nazorat qilish dasturchining vazifasi hisoblanadi.

10.2. ELEMENTLARI SANALADIGAN TIPLAR

Ko‘pincha ma’lum bir oiladagi qiymatlarni (haftaning kunlari-dushanba, seshanba, chorshanba, payshanba, juma, shanba, yakshanba yoki yilning fasllari-qish, bahor, yoz, kuz kabi) qabul qiladigan kattaliklar bilan ishlashga to‘g‘ri keladi. Zarur bo‘lsa, yangi tip orqali bu qiymatlarni aniqlash va foydalanish mumkin. Bu ish umumiy ko‘rinishda

type *tip nomi =(elementlar ro‘yxati) ;*

tarzida tashkil qilinadi. Masalan:

```
type KUN=dushanba, seshanba, chorshanba,
        payshanba, juma, shanba, uakshanba) ;
var X: KUN ;
```

Endi X-o‘zgaruvchi faqat KUN tipidagi ma’lumotlarni qabul qiladi halos. O‘zgaruvchilarning qiymatlarini yuqoridagi kabi oldindan aniqlab qo‘yish,

dasturlarning bajarilishi davomida yanglishib boshqacha qiymat berib qo‘yishning oldini oladi. Agar sanaladigan tipdagi o‘zgaruvchi o‘z chegarasidan tashqaridagi qiymatlarni oladigan bo‘lsa, dasturning bajarilish jarayoni to‘xtatiladi va yo‘l qo‘yilgan hatolik haqida 'TYPE MISMATH' axboroti beriladi.

10.3. TO‘PLAMLAR VA ULAR USTIDA AMALLAR BAJARISH

To‘plam deb bir hil tipdagi ma’lumotlardan tuzilgan birlashmaga aytiladi. To‘plamdagi ma’lumotlar soni ihtiyoriy bo‘lib, umumiy ko‘rinishda quyidagicha e’lon qilinadi:

type to‘plam nomi=***set of*** ma’lumotlar tipi ;

Masalan:

type *tup=set of char ;*

e’loni yordamida elementlari *char* tipida bo‘lgan to‘plam e’lon qilinmoqda. Umuman olganda to‘plamlarni tipi *real* bo‘lmagan va tartiblangan ihtiyoriy tipdagi ma’lumotlardan hosil qilish mumkin.

To‘plamning hamma elementlari tegishli bo‘lgan tipni to‘plamning bazaviy tipi deb ataladi. To‘plamlar bo‘sh bo‘lishi o‘am mumkin. To‘plamdagi elementlar soni eng ko‘pi bilan bazaviy tipga kirgan elementlar sonidan ortiq bo‘lmasligi lozim hamda EHM ning imkoniyatlariga qarab 256 tagacha bo‘lishi mumkin. SHuning uchun *set of 0..50 ;* e’loni to‘g‘ri bo‘lgani holda, *set of integer ;* e’loni noto‘g‘ri hisoblanadi.

Ture ou= (jan, fev, mar, apr,may,yun,yul, avg, sen, okt, noy, dek) ;

sana: set of oy ;

var m : oy ; mm : sana ;

Bu holda M o‘zgaruvchining qiymati bo‘lib biror oyning nomi qatnashsa, *mm* o‘zgaruvchining qiymatlari oylarning ihtiyoriy to‘plamlaridan (masalan: {shag, arg, may}) iborat bo‘lishi mumkin. Turbopaskalda to‘plam elementlarini kvadrat qavslar "[]" yordamida beriladi. Kvadrat qavslar ichida bir yoki bir necha element ko‘rsatilishi mumkin, shu jumladan xech narsa ko‘rsatilmasligi ham mumkin.

m*m*:=*[] ;* (*bo‘sh to‘plam *)

mm:=*[yan];* (* bitta elementli to‘plam*)

mm:=*[uan .. arg,sen, noy] ;* (* to‘plamga uan, fev, mar, arg, sen, nou kiradi *)

mm:=*[feb,M] ;* (*to‘plamga feb va M ning qiymatlari kiradi*)

Faraz qilaylik,

K:=[1, 3, 4, 5, 7];

L:=[2, 4, 6, 7];

bo‘lsin. Ikki to‘plam o‘rtasida quyidagi amallarni bajarish mumkin:

1) Ikki to‘plam R va L ning yig‘indisi $(R+L)$ bo‘lib, ularning hech bo‘lmaganda bittasiga tegishli bo‘lgan elementlar to‘plami xizmat kiladi. $(R+L)$ amalining natijasi [1, 2, 3, 4, 5, 6, 7| bo‘ladi.

2) Ikki to‘plamning ayirmasi (R-L) deb, R ning L ga kirmagan elementlaridan tuzilgan to‘plamga aytiladi. (R-L) ning natijasi [1, 3, 5] to‘plam bo‘ladi.

3) Ikki to‘plam ko‘paytmasi R*L deb, bir vaqtda har ikki to‘plamga tegishli bo‘lgan elementlar to‘plamiga aytiladi. R*L ning natijasi [4, 6] to‘plam bo‘ladi.

4) Elementni to‘plamga tegishli yoki tegishli emasligini aniqlash uchun ***in*** amalidan foydalaniladi. Uning natijasi "true' (element to‘plamga tegishli) yoki "false" (tegishli emas) bo‘lishi mumkin. ***O in R*** amalining natijasi "false", ***4 in L*** amalining natijasi esa "true”.

5) To‘plamlar ustida taqqoslash amallarini bajarish mumkin. "=" yoki "<>" belgisi ikki to‘plamning teng yoki teng emasligini tekshiradi; "<=” yoki ">=" belgilari bir to‘plam ikkinchi to‘plamga tegishli ekanligini aniqlash uchun ishla-tiladi. R<>L amali natijasi "false", [4]< =R*L esa "true", [4]> =R*L amali natijasi "false".

Quyidagi dasturda QIZLAR to‘plami e’lon qilinadi. So‘ng ulardan ikkita yangi BILIMDON va GUZAL to‘plamlarini hosil qilinadi. Dastur ZEBO ismli element har ikki to‘plamga tegishlimi yoki yo‘qmi degan savolga javob berish bilan tugaydi.

```
program yangi_tip;
  type qizlar=(xalima,salima,zebo,guli,asila,umida );
  var x,guzal, bilimdon : set of qizlar ;
begin
     guzal:=[ salima, zebo, asila, umida ] ;
     bilimdon:=[ xalima, zebo, guli, umida ] ;
     x:=guzal+bilimdon;
     writeln [zebo in x);
end.
```

EHM uni bajarib, ***true*** yozuvini displeyga chiqaradi.

§-11. YOZUVLAR YOKI ARALASH TIPLAR

Maktab o‘quvchisi tabelini ko‘z oldingizga keltiring. Unda viloyat, tuman, maktab nomeri, sinfi, o‘quvchining familiyasi, ismi, otasining ismi, o‘quv yili hamda o‘quvchining turli fanlar bo‘yicha choraklardagi va yillik baholari saqlanadi. Demak, bitta sinfda 25 ta o‘quvchi bo‘lsa, ular haqidagi hamma ma’lumotlarni saqlash uchun 25 dona tabel zarur bo‘ladi. Endi har bir tabeldagi barcha ma’lumotlarni bitta satrga

yozilgan holatini ko‘z oldingizga keltiring. Demak, 25 ta satrdagi ma’lumotlar jamg‘armasi hosil bo‘ladi. Bu jamg‘armadan ma’lumotlarni olish yoki yozish ishlarini osonlashtirish maqsadida mazmun jihatidan bir hil bo‘lgan ma’lumotlarni alohida ustunlarga yoziladi. Masalan: viloyatlar bitta ustunga, o‘quvchilarning familiyalari bitta, ismlari boshqa ustunga kabi yoziladi.

Hosil bo‘lgan ustunlarni maydon deb ataladi. Har bir satrni esa yozuv deyiladi. Demak, yozuvlar maydonlar to‘plamidan iborat. Har bir maydon mazmun jihatidan qandaydir bir hil tipdagi ma’lumotlarni o‘z ichiga oladi.

Ko‘rinib turibdiki, bir yozuvni bitta o‘zgaruvchi dob qarasak, u o‘zgaruvchi turli tipdagi qiymatlarni qabul qiladi. Bunday o‘zgaruvchilarni aralash tipli o‘zgaruvchilar deb ham yuritiladi. Ular umumiy ko‘rinishda

type *yozuv nomi=**record** 1-maydon:1-tip;... ; n-maydon:n-tip **end** ;*

tarzida aniqlanadi. Masalan:

```
type tab=record vil, tum:string[20]; mak:integer;
     sinf:l..12; fam, ism:string[20]; uquv:integer end;
var uquvchi: tab ;
```

YUqoridagi e’lon yordamida ***uquvchi*** nomli o‘zgaruvchining ***vi], tum*** maydonlari uzunligi 20 tagacha bo‘lgan matnli, ***mak*** maydoni butun sonli, ***sinf*** maydoni 1 dan 12 gacha bo‘lgan sonli, ***fam*** va ***ism*** maydonlari uzunligi 20 gacha bo‘lgan matnli, ***uquv*** maydoni esa butun sonli ma’lumotlarni saqlash uchun mo‘ljallanganligi haqida EHM ga axborot berilmoqda. CHunki bu o‘zgaruvchi ***tab*** tipidagi qiymatlarni qabul qiladi. SHu o‘zgaruvchining biror maydoniga murojaat qilish uchun avval o‘zgaruvchining nomi, "." belgisi va maydon nomi

uquvchi.vil, uquvchi.sinf, uquvchi.fam

tarzida ko‘rsatiladi. Bu erda *uquvchi* o‘zgaruvchisining *vil, sinf, fam* maydonlariga murojaat qilinmoqda.

```
Uquvchi. vil:='Namangan'; uquvchi.sinf:=11;
 uquvchi.fam:='Aliyev';
```

Ko‘rinib turibdiki, yozuvning hamma o‘zgaruvchi va maydon nomlariini ko‘rsatishni tashkil qilish ancha murakkab ish. Bu ishni foydalanuvchi uchun qulayroq holga keltirish uchun ***with*** xizmatchi so‘zidan foydalanish mumkin. Uning umumiy ko‘rinishi quyidagicha:

with yozuv nomi do begin ... end;

Bu imkoniyatdan faqat bitta yozuvning bir nechta maydonlari uchun foydalanish mumkip. Masalan:

with uquvchi do begin vil:='Namangan' ;

```
sinf:=ll ; fam:='Otaxanov'; end ;
```

7.4-masala. Sinfdagi 25 ta o‘quvchiiing familiyasi, ismi va tug‘ilgan yillari haqidagi ma’lumotlar fam, ism va t_yil maydonlarida berilgan bo‘lsin. SHu sinfda 1990 yilda tug‘ilgan Abdulla ismli bolalar sonini aniqlang.

Yechish g‘oyasi: Dastlab sinfdagi barcha o‘quvchilarning familiyasi, ismi va tug‘ilgan yillarini aniqlab olinadi. So‘ngra ularning hammasini ismi va tug‘ilgan yillarini 'Abdulla' va 1990 bilan solishtiriladi.

```
Program maktab ;
    type uquv=record fam, ism:string[20];
    t_yil:integer; end;
    var d: array [1.25] of uquv ;
    s, i.integer;
begin
    for i:=1 to 25 do readln(d[i]);
    s:=0;
    for i:=1 to 25 do
        if (d[i].ism='Abdulla') and ( d[i].t_yil=90) then s:=s+1;
    writeln('1990 yilda tugilgan Abdullalar soni= ',s)
end.
```

12. PROTSEDURALAR.

12.1. FORMAL VA JORIY O‘ZGARUVCHILAR.

Odatda ko‘plab masalalarni Yechish jarayonida oldindan masalani qanday qiymatlar (o‘zgaruvchilar) uchun berilganini bilib bo‘lmaydi. Lekin uni Yechish uchun shartli ravishda turli o‘zgaruvchilar kiritiladi va shu o‘zgaruvchilar uchun qo‘yilgan masalani to‘la Yechishning qonun - qoidalari (algoritmi) yaratiladi. Ana shu o‘zgaruvchilar formal o‘zgaruvchilar deyiladi. Keyin shu sinfga taalluqli bo‘lgan konkret masalani olinadi. Bu masalada odatda hamma o‘zgaruvchilar aniq ko‘rsatib qo‘yiladi. Masalani ana shu o‘zgaruvchilar uchun hal qilish talab qilinadi. Bunday o‘zgaruvchilar joriy o‘zgaruvchilar deb ataladi. Endi masalani Yechish uchun yaratilgan hamma qonun-qoidalarni formal o‘zgaruvchilarning o‘rniga masala shartida berilgan joriy o‘zgaruvchilarni qo‘yib bajariladi.

Ma’lumki, to‘g‘ri to‘rtburchak yuzi $S:=A*B$ formula bilan topiladi. Bu erda A-to‘rtburchak bo‘yini, V-esa enini ifodalaydi.

Masala: Bo‘yi N, eni M bo‘lgan to‘g‘ri to‘rtburchak yuzini toping.

Yechish g‘oyasi: To‘g‘ri to‘rtburchakning yuzi bo‘yi bilan enining ko‘paytmasiga teng ekanligi yuqorida aytilgan edi. Uni qo‘llab yuza topiladi: S = N*M. Bu misolda A va V formal o‘zgaruvchi, N va M esa joriy o‘zgaruvchi hisoblanadi. Formal

o‘zgaruvchi bilan joriy o‘zgaruvchi ustma-ust tushishi ham mumkin.

12.2. PROTSEDURALAR

Ayrim bir masalalarni Yechish jarayonida bitta masalani bir nechta kichik masalalarga bo‘lib Yechish anchagina qulay hisoblanadi. Agar hosil bo‘lgan masalalar bitta sinfga tegishli bo‘lsa yana ham yaxshi. Bu holda bir sinfga tegishli bo‘lgan xar bir masalaga alohida dasturlar yozish o‘rniga ulardan bittasi uchun formal o‘zgaruvchilar o‘ylab topiladi va masalani shu o‘zgaruvchilar ychun Yechish buyruqlari ketma-ketligi tashkil qilinadi. Ana shu buyruqlar ketma-ketligi protsedura hisoblanadi. Xar bir protsedura uchun beriladigan (asosiy dasturdan o‘tadigan) uzgaruvchilar ro‘yxati hamda protsedurada hisoblanishi talab qilinadigan o‘zgaruvchilar ro‘yxati aniqlab olinadi. 3apup bo‘lganda undagi formal o‘zgaruvchilar o‘rniga joriy o‘zgaruvchilarni qo‘yib, (ro‘yxatdagi 1-normal o‘zgaruvchi o‘rniga joriy o‘zgaruvchi, 2-formal o‘zgaruvchi o‘rniga 2-joriy o‘zgaruvchini o‘yiladi va hokazo) protseduralarga murojaat qilish mumkin.

Protseduralar ishini tashkil qiluvchi dasturni asosiy dastur deyiladi. Protseduraga asosiy dasturdan turib murojaat qilinadi. Buning uchun protsedura nomi ko‘rsatiladi, so‘ng qavslar ichida joriy o‘zgaruvchilar ro‘yxati beriladi. Xar galgi murojaat qilishda asosiy dasturdan protseduraga tushadigan va protseduradan asosiy dasturga qaytib chiqib ketadigan o‘zgaruvchilar po‘yxati ko‘rsatiladi. Asosiy dasturdan protseduraga murojaat qilinganda, asosiy dasturning bajarilish jarayoni to‘xtaydi va EHM protsedurani bajara boshlaydi. Protseduradagi barcha buyruqlar joriy o‘zgaruvchilar uchun to‘la bajarilgandan so‘ng EHM yana asosiy dasturning kelgan eridan boshlab navbatda turgan buyruqlarni bajarishda davom etadi.

Protseduralar umumiy quyidapgcha yoziladi :

procedure pr_nomi (var ruyxat1 ; var ruyxat2) ;
var ruyxat3 ;
begin
protsedura buyruqlari ketma-ketligi ;
end;

Bu erda *ruyxatl*-asosiy dasturdan protseduraga o‘tadigan formal o‘zgaruvchilar va ularning tiplari; *ruyxat2*-npotsedupadan asosiy dasturga chiqib ketadigan formal o‘zgaruvchilar va ularning tiplari; *ruyxat3* –oraliq o‘zgaruvchilar va ularning giplari.

Masala shartida ko‘rsatilmagan, lekin masalani Yechish uchun hisoblanishi zarur bo‘lgan o‘zgaruvchilarni oraliq o‘zgaruvchilar deyiladi. Oraliq o‘zgaruvchilar odatda faqat bitta protsedura uchun taalluqli bo‘ladi. SHunipg uchun ularni lokal (mahalliy) o‘zgaruvchilar deb ham yuritiladi.

Faraz qilaylik, kvadrat tenglamani Yechish uchun protsedura tashkil qilingan

bo‘lsin. Bu protsedura uchun diskriminantni ifodalovchi o‘zgaruvchi lokal o‘zgaruvchi hisoblanadi. Protsedurada qatnashayotgan barcha lokal o‘zgaruvchilar protseduraning birinchi *begin* xizmatchi so‘zidan avval e’lon qilinishi shart. Protsedurada lokal o‘zgaruvchilarning qatnashmasligi ham mumkin.

Global o‘zgaruvchilar deb bir vaqtning o‘zida ham asosay dasturga, ham protseduralarga taalluqli bo‘lgan o‘zgaruvchilarga aytiladi.

Prezident farmonlarini global desak, viloyat Hokimining qarorlarini lokal deb karash mumkin.

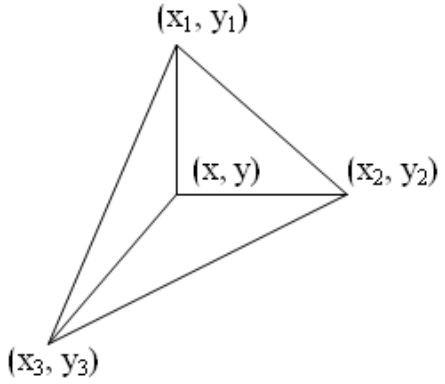
Bitta dasturda bir nechta protseduralar qatnashishi ham mumkin. Protseduralar bitta dasturda bir marta aniqlanadi, lekin ehtiyojga qarab bir necha marta foydalanish mumkin.

Protseduralarni asosiy dasturda o‘zgaruvchilar tiplari ko‘rsatilgandan so‘ng, birinchn BEGIN xizmatchi so‘zidan avval tashkil qilinadi. Protseduralar qatnashgan dasturning umumiy ko‘rinishi quyidagicha:

program dasturning nomi ;
    label tamg‘alar ro‘yxati ;
    const o‘zgarmaslar ro‘yxati ;
    type yangi tiplarni aniqlash ;
var asosiy dasturda katnashadigan o‘zgaruvchilar;
procedure prots_nomi(vag formal o‘zg:tip; var natijalar: tip);
var lokal o‘zgaruvchilar : tip ;
begin
    protsedura buyruqlari ketma-ketligi ;
end ;
begin
    asosiy dastur buyruqlari ketma-ketligi;
end.

Agar protsedura e’lon qilingan bo‘lsa, undan dasturning keyingi qismlarida foydalanish mumkin. Buning uchun avval protsedura nomi, so‘ngra qavs ichida joriy o‘zgaruvchilar ro‘yxati ko‘rsatiladi.

<u>*Masala-1;*</u> Uchburchak uchlarining koordinatalari (x1,y1), (x2, y2). (x3, y3) berilgan bo‘lsin. (x, u) koordinatali nuqta shu uchburchakka tegishli bo‘la oladimi ?



<u>*Yechish g‘oyasi:*</u> Berilgan uchburchak yuzasi S bo‘lsin. Uchburchak uchlarini (x, u) koordinatali nukta bilan tutashtirib, 3 ta uchburchak hosil qilinadi.

Ularning yuzalari mos ravishda S1, S2 va S3 bo‘lsin. Agar (x, u) nuqta berilgan uchburchakning ichida yotsa, $S = S1 + S2 + S3$ bo‘lishi kerak. Aks holda nuqta uchburchak ichida yotmaydi. Ko‘rinib turibdiki, bu erda uchlarining koordinatalari berilgan to‘rtta uchburchak yuzalarini hisoblashga to‘g‘ri kelmoqda. Uchburchak uchlarining koordinatalari uchun formal o‘zgaruvchilarni (a1,b1), (a2,b2) va (aZ,bZ) tarzida tanlanadi. Bu uchburchakning tomonlari va yarim perimetrini belgilash uchun mos ravishda A, V, S va R o‘zgaruvchilarni olinadi. Ular masala shartida ko‘rsatilmagani uchun lokal o‘zgaruvchilar hisoblanadi. Tanlangan formal o‘zgaruvchilarni hisobga olib, uchburchak yuzini topish buyruqlari ketma-ketligidan protsedura hosil qilinadi. So‘ngra ketma-ket to‘rt marta (xl, x2, x3, yl, y2, y3), (x, xl, x2, y, yl, y2), (x, x2, xZ, u, u2, uZ) va (x, xl, x3, y, yl, y3) joriy o‘zgaruvchilar uchun protseduraga murojaat qilinadi YA’ni uchlarining koordinatalari ana shu nuqtalarda yotgan uchburchaklariing yuzalari S1, S2, S3 va S larni hisoblab topiladi. So‘ng masalani Yechish g‘oyasiga ko‘ra $S = S1 + S2 + S3$ munosabatning o‘rinli bo‘lishi yoki bo‘lmasligiga qarab hulosa chiqariladi.

```
Program nuqta ;
    var x,xl,x2,x3,y,yl,y2,y3,sl,s2,s3,s: real;
    r : string[13] ;
    procedure uza(al,a2,a3,bl,b2,b3:real; var yuza: real);
        var a,h,c,p : real;
    begin
        a:=sqrt(sqr(a2-al)+sqr(b2-bl)) ;
        b :=sqrt(sqr(a3-a2)+sqr(b3-b2)) ;
        c:=sqrt(sqr(a3-al) +sqr(b3-bl)) ;
        p:=(a+b+c)/2 ;
        yuza:=sqrt(p*(p-a) *(r-b) *(r-s)) ;
    end ;
begin
    write (‘ucbburcbakning birinchi uchi'); readln(xl, yl);
    write ('uchburchakning ikkinchi uchi’); readln(x2, y2);
    write ('uchburchakning uchinchi uchi'); readln(x3, y3);
    write( 'nuqtaning koordinatasi’); readln(x,y) ;
    uza(xl,x2,x3,yl,y2,y3,sl); uza(x,xl,x2,y,yl,y2,s2);
    uza(x,x2,x3,y,y2,y3,s3); uza (x,xl, x3,y,yl,y3, s) ;
    if S = S1 + S2 + S3 then r:= 'tegishli ' else r:= 'tegishli emas';
    writeln(‘berilgan nuqta uchburchakka’, r)
end.
```

uchburchakning birinchi uchi 2 3
uchburchakning ikkinchi ucni 1 2
uchburchaiming uchinchi uch 0 —3
nuqlaning koordinatasi 0 0
berilgan nuqta uchburchakka tegishli emas

uchburchakning birinchi uchi 1 2
uchburchakning ikkinchi ucni 2 3
uchburchakning uchinchi uchi 3 4
nuqtaning koordinalasi 2 2.5
berilgan nuqta uchburchakka tegishli

Agar *exit* yoki *halt* operatorlari protsedura amallari tarkibida uchrab qolsa, bu protseduraning *exit* yoki *haft* operatorlaridan keyingi ko‘rsatilgan buyruqlari bajarilmay, boshqaruv navbatdagi blokka (protseduraga murojaat qilingan joyga) o‘tadi. Quyidagi dastur natijasiga e’tibor bering.

```
Program tuxtash_ 1 ;
    var i,k:integer ;
    procedure bekor( var a:integer; var b:integer ) ;
        begin
        for b:=1 to 20 do begin b:= a+b ;
        exit ; end;
    end;
begin
    j:=0 ; bekor(i,k);
    writeln('sikl tugaram qadam:= ',k);
end.
```

YUqoridagi dastur uchun EHM mana bu natijani beradi:
sikl tugagan qadam:= 1

12.3. PROTSEDURA-FUNKSIYA.

Turli ifodalarning qiymatlarini hisoblashga qandaydir funksiyalarning qiymatlaridan foydalanishga to‘g‘ri keladi. Masalan: $y = \cos x + 3$ ifodaning qiymatini hisoblaganda, avval *cosx* o‘rniga uning qiymatini topib qo‘yiladi va 3 ga qo‘shiladi. Agar *cosx* funksiyasining o‘rnida murakkab funksiya yoki uzundan-uzun arifmetik ifoda tursachi?

Bunday hollarda dasturchining ishini bir muncha soddalashtirish uchun Turbopaskalda protsedura-funksiyalar yoki foydalanuvchi funksiyasini tuzish imkoniyati yaratilgan.

Protsedura-funksiyalar odatda murakkab ifodalarni, uzun arifmetik ifodalarni yoki biror ifoda qiymatini argumentlarning turli qiymatlari uchun hisoblashga to‘g‘ri kelgan hollarda tashkil qilinishi mumkin. Funksiyalar dasturning qismi bo‘lib, umumiy ko‘rinishda quyidagicha aniqlanadi:

function funksiya nomi (formal o‘zgaruvchilar):tip ;
const o‘zgarmaslar ro‘yxati *;*
type yangi tiplar *;*
var lokal o‘zgaruvchilar ;
begin
protsedura-funksiya buyruqlari ketma-ketligi ;
end ;

Protsedura-funksiyalar yoki funksiyalar protseduralar kabi asosiy dastur tarkibida, o‘zgaruvchilarning tiplari aniqlangandan so‘ng, asosiy dasturning birinchi BEGIN xizmatchi so‘zidan avval ko‘rsatiladi. Protseduradan funksiya quyidagi tomonlari bilan farq qiladi:

1) Funksiya nomini ifodalayotgan o‘zgaruvchiga albatta qiymat berilishi kerak.
2) Funksiya nomini ifodalayotgan o‘zgaruvchi tipini ko‘rsatiladi.
3) Protsedura bir nechta o‘zgaruvchi qiymatlarini hisoblash uchun mo‘ljallanadi, funksiyada esa uning nomini ifodalayotgan o‘zgaruvchining qiymatini topiladi.

Protseduralardan foydalanish uchun alohida buyruq yozilsa, funksiyani biron bir ifodaning tarkibpda chaqiriladi. Buning uchun uning nomi ko‘rsatiladi va qavslar ichida joriy o‘zgaruvchilar ro‘yxati ko‘rsatiladi. SHundan so‘ng funksiya ana shu joriy o‘zgaruvchilarni formal o‘zgaruvchilar o‘rniga qo‘yib (1-formal o‘zgaruvchi o‘rniga 1-joriy o‘zgaruvchi, 2-formal o‘zgaruvchi o‘rniga 2-joriy o‘zgaruvchi va hokazo), funksiyadagi amallar ketma-ketligini bajariladi va funksiya nomi bilan atalgan o‘zgaruvchining qiymati hisoblanadi. Bu qiymat funksiyani chaqirayotgan yozuv bilan almashtiriladi va ifoda qiymatini hisoblashning keyingi amallariga o‘tiladi.

<u>*8.3-misol.*</u> S va T haqiqiy sonlar berilgan bo‘lsin. Quyidagi ifodaning qiymatini toping:

$$r = g(1.2, s) + g(t, 2s) + g(2s - 1, t)$$

Bu erda

$$g(a,b) = \frac{a^2 + b^2}{a^2 + 2ab + 3b^2 + 4}$$

<u>*Yechish g‘oyasi:*</u> Ko‘rinib turibdiki, *g(a,b)* funksiyasining qiymatini 3 marta hisoblashga to‘g‘ri kelmoqda. SHuning uchun *g(a,b)* funksiyaning qiymatini hisoblash maqsadida funksiya tashkil qilinadi. So‘ngra, birinchi marta

$a = 1.2, b = s$ bo‘lgan hol uchun, ikkinchi marta $a = t, b = 2s$ uchun va uchinchi marta $a = 2s - 1, b = t$ bo‘lgan hollar uchun unga murojaat qilinadi.

```
Program funksiya;
        var s, t,r: real;
function g(a, b:real) : real ;
begin
 g := (a * a + b * b) /(a * a + 2 * a * b + 3 * b * b + 4);
end;
begin
        write ('S va T ni kiriting‘); readln (s,t);
        r := g(1.2,s) + g(t,2s) + g(2s - 1,t) ;
        wrileln('R := ’,r:10:5 )
end.
```

Bu dastur uchun EHM quyidagi natijani berdi:

S va T ni kiritiiig 12 -5

R := 2.7373

Agar funksiya nomi protsedura-funksiyada bir necha marta qiymat olca, uning eng oxirgi olgan qiymati asosiy dasturga qaytadi. Agar funksiya nomi biror marta ham qiymat olmagan bo‘lsa, u holda protsedura-funksiya aniqlanmagan hisoblanadi. Agar *exit* yoki *halt* buyruqlari funksiya amallari tarkibida uchrab qolsa, bu funksiyaning *exit* yoki *halt* dan keyingi ko‘rsatilgan buyruqlari bajarilmay, boshqaruv navbatdagi blokka (funksiyaga murojaat qilingan joyga) o‘tadi.

```
Program tuxtash_2 ;
        var i,k;integer ;
        function bekor(a:integer) integer;
                var b.integer ;
        begin
                for b := l to a do begin
                bekor := a + b ; exit ; end;
        end;
begin
        i := 20;
        k := bekor(i);
        writeln('sikl tugagan qadam’, k);
end.
```

dasturi uchun EHM olgan natija quyidagicha:

sikl tugagan qadam 21

§-13. REKURSIYA

Agar dastur o‘zini-o‘zi protsedura yoki funksiya sifatida foydalanadigan bo‘lsa, bunday dasturlarni rekursiv dastur deyiladi. Rekursiv dasturlar ikki turga bo‘linadi:

a) To‘g‘ri rekursiya. Bunda funksiya o‘ziga-o‘zi murojaat qiladi.

b) YOndosh rekursiya. Bunda 1-funksiya 2- funksiyaga, 2-funksiya esa 1-funksiyaga murojaat qiladi.

Rekursiv dastur yozish uchun avvalo 1) rekkurent munosabat; 2) shu munosabat uchun boshlang‘ich holat aniqlangan bo‘linshi shart. Rekkurent munosabat deganda biror jarayonning N va N-1 qadamlarni bog‘lovchi munosabatlar tushuniladi. Masalan, $N!=N(N-1)!$ formulani N! uchun rekkurent munosabat deb qarash mumkin. Bu munosabat uchun boshlang‘ich holat bo‘lib, $1!=1$ xizmat qiladi. Bu ma’lumotlarni hisobga olsak, faktorialni hisoblash masalasi uchun rekkurent munosabatlar quyidagicha bo‘ladi:

$$N!=\begin{cases} N\cdot(N-1)!, & agar\ N>1\ bo'lsa \\ 1, & agar\ N=1\ bo'lsa \end{cases}$$

Ko‘rinib turibdiki, N! ni hisoblash uchun avval (N-1)! ni hisoblab topish kerak. Lekin, $(N-l)!=(N-2)!$ $(N-1)$ bo‘lganligi uchun o‘z navbatida (N-2)! ni topishga xarakat qilinadi. (N-2)! esa (N-3)!(N-2) ga teng va hokazo. Demak, N! ni hisoblash algoritmi o‘zining ichiga o‘zi "cho‘kib" bormoqda. Bu jarayon boshlang‘ich holat sodir bo‘lgunga, ya’ni 1! gacha davom etadi. SHundan keyin, "cho‘kish" jarayoni to‘xtaydi, $1!=1$ ekanligi haqida ko‘rsatma olgan EHM endi yuqoriga qarab "suzib" chiqish bosqichini bajaradi. YA’ni, $2!=1$, $2!=1\cdot 2=2,\ 3!=2!\cdot 3=6$ va hokazo. Bu holat to N! hisoblanmaguncha davom etaveradi. SHu jarayonning dasturi quyidagicha yoziladi :

```
program factorial ;
        var r.n: integer ;
        function fak(k:integer): integer;
        begin
                if k>1 then fak:=fak(k-1)*k else fak:=1 ;
        end;
begin
        read(n) ;
        write(fak(n));
end.
```

$N=4$ uchun "cho‘kish" va "suzib chiqish" jarayoni quyidagicha :

N ning qiymatlari	4 4 3 2 1	yakuniy rekursiyadan chiqishdagi oraliq qiymatlar
N=1 sharti natijasi	yo‘q yo‘q yo‘q ha	
	fak(3)*4 fak(2)*3 fak(1)*2 1	p:=p*4=24 p:=p*3=6 p:=p*2=2 p:=1

Demak, EHM N=4 bo‘lgan hol uchun 24 natijani beradi.

SHunday masalalar sinfi mavjudki, bu masalalarni rekursiyadan foydalanmay turib Yechishning boshqa usullari yo‘q.

Masala-8.4. f(n) funksiyaning qiymatlari f(0)=1, f(2n)=f(n) va f(2n+1)=f(n)+1 ifodalar yordamida topiladi. Berilgan k natural soni uchun f(k) ni toping.

Yechish g‘oyasi: YUqoridagi masalani K ta elementli massiv yordamida Yechish ko‘pchilikning nazarida oson usulga o‘xshaydi. Lekin bu to‘g‘ri ish emas. CHunki, K etarlicha katta son bo‘lsa, K ta elementli massiv EHM xotirasiga sig‘may qolishi mumkin. Qolaversa, siqqan takdirda ham, bu elementlarning hammasi ishlamaydi. k=1000 bo‘lganda massivning 1000 ta elementidan ko‘pi bilan 11 tasi masalani Yechish uchun kerak bo‘ladi halos. Qolgann esa EHM xotirasini bekordan bekorga band qiladi. SHuning uchun qo‘yilgan masalani massivdan foydalanmay Yechish eng yaxshi usul hisoblanadi. Bu usul rekursiyadan foydalanish usulidir. Rekkurent munosabat va bu munosabat uchun boshlang‘ich holat masala shartida keltirilganligi dasturchi ishini yanada osonlashtiradi.

```
Program rekursiya_8_2 ;
    var k,n : integer ;
    function f(m:integer) : integer ;
        var h: integer;
    begin
        if m=0 then f:=1 else  begin
        h:=m div 2 ;
        if m mod 2=0 then f:=f(h)
        else f:=f(h)+1; end;
    end;
begin
    read(k) ; n:=f(k) ;
    write('f(',k,') := ',n:10)
end.
```

Bu dasturni $k = 10$ bo‘lgan hol uchun bajargan EHM

$f(10) := 3$

yozuvlarni displey ekraniga chiqaradi.

YUqoridagi ma’lumotlardan ko‘rinib turibdiki, rekursiya oddiy protsedura yoki funksiyaga nisbatan murakkabroq tushuncha, lekin u mohir dasturchi qo‘lida juda ham yaxshi qurol bo‘lishi mumkin.

§-14. FAYLLAR BILAN ISHLASH.

14.1. FAYL TUSHUNCHASI.

YAqin yillargacha EHM lar faqat sonli ma’lumotlarni qayta ishlashga mo‘ljallangan edi. Xozirga kelib, EHM xalq xo‘jaligining deyarli hamma sohalarini qamrab olmoqda. SHuning uchun ham xar bir sohadagi ma’lumotlarni o‘qish, saqlash, qayta ishlash hamda boshqa ma’lumot iste’molchilariga uzatish kabi eng muhim va dolzarb masalalar zamonaviy EHM lar zimmasiga yuklanmoqda. Natijada, 70-yillarning oxiriga kelib, yuqoridagi masalalarni hal qilish uchun informatika fani yuzaga keldi. Xalk xo‘jaligining boshqaruv sistemalari, aloqa vositalari kabi ko‘plab sohalarida EHM lar asosan ma’lumotlarni saqlash vositasi sifatida qo‘llanmoqda. Uzoq muddat saqlashga to‘g‘ri keladigan ma’lumotlar maxsus vositalarda, magnit disklarida, magnit lentalarida fayllar ko‘rinishida saqlanadi.

Fayl deb, xotira qurilmalaridan birida saqlanayotgan (ma’lum bir hajmni egallagan) va o‘zining nomiga ega bo‘lgan ma’lumotlar to‘plamiga aytiladi.

Fayl bo‘sh bo‘lishi ham mumkin. Fayllar ma’lumot saqlashning eng qulay usuli ekanligining sababi quyidagilardan iborat:

1) Odatda dasturni bajarib, olingan natijalar dastur o‘z ishiii tugatgandan so‘ng, EHM xotirasidan o‘chib ketadi. Bu ma’lumotlar yana zarur bo‘lsa, dasturni yangitdan bajarishga to‘g‘ri keladi. Buni oldini olish uchun olingan natijalarni fayllarga yozib qo‘yilishi mumkin.

2) Bitta faylda saqlanayotgai ma’lumotlar ko‘plab masalalar (dasturlar) uchun asos bo‘liish mumkin.

3) Ma’lumotlar soni EHM ning operativ xotirasiga sig‘maydigan darajada ko‘p bo‘lishi mumkin.

4) Fayllardan ulardagi ma’lumotlar doirasidagi ihtiyoriy maqsadlar uchun foydalanish mumkin.

Dasturchi fayllar ishini tashkil qilar ekan, faqat dastur va uning natijasi haqida qayg‘uribgina qolmasdan, balki ko‘plab qo‘shimcha dasturlar yordamida fayllarni yaratish, faylda saqlanayotgan ma’lumotlarni boshqarish, tahlil qilish, tartnblash, ehtiyojga qarab displey yoki qog‘ozda akslantirish kabi masalalarni ham xal qilishi kerak. YAna, ilgari ko‘zda tutilmagan yangi zaruratlar

uchun qo‘shimcha dasturlar yaratish haqida ham o‘ylashi kerak.

Turbopaskalda fayllar deb, EHM da saqlanayotgan bir xil tipga mansub bo‘lgan ma’lumotlar (komponentalar) to‘plamiga aytiladi.

Fayldagi ma’lumotlardan foydalanish uchun ularni o‘qiladi va o‘qilgan ma’lumotlarni o‘zgaruvchilarga qiymat qilib beriladi.

Ihtiyoriy vaqtda faylning faqat bitta komponentasi bilan ishlash mumkin xalos. Bu ma’lumotni ko‘rsatkich (kursor) ko‘rsatib turadi. Ko‘rsatkich birinchi komponentadan boshlab, xar bir ma’lumot o‘qilgandan keyin, navbatdagi o‘qish kerak bo‘lgan ma’lumotni ko‘rsatib turadi. (Boshlang‘ich sinflardagi xatcho‘plarni eslab ko‘ring.)

Fayldagi ma’lumotlar soni o‘zgarib turadi, u dastlab nolga teng, keyinchalik faqat faylga yangi ma’lumotlar qo‘shilganda o‘sishi yoki o‘chirilganda noltagacha kamayishi mumkin. YAngi ma’lumotlar doim faylning oxiriga qo‘shiladi.

14.2. FAYLLI TIPLAR.

Faylda saqlanayotgan ma’lumotlarni faylning komponentalari deyiladi. Bitta faylning hamma komponentalari bitta qandaydir tipga xos bo‘ladi. Bu tip standart bo‘lishi yoki dasturchi tomonidan aniqlanadigan yangi tip bo‘lishi mumkin. Fayllardan foydalanish uchun avval, ularni belgilovchi faylli o‘zgaruvchilarni e’lon qilish kerak. Faylli o‘zgaruvchi boshqa o‘zgaruvchilar kabi biror tipga mansub bo‘lib, o‘zi belgilayotgan fayl komponentalari tipida bo‘ladi.

Faylli o‘zgaruvchilarni umumiy ko‘rinishda quyidagicha e’lon qilish mumkin:

o‘zgaruvchi : ***file of*** ma’lumot tipi ;

Masalan:

type a: file of integer ;
var b : a ;
s : file of real ;

yozuvlari yordamida *integer* tipidagi ***a*** faylli tip, ***a*** tipli ***b*** o‘zgaruvchi, *real* tipidagi ***c*** faylli o‘zgaruvchilar e’lon qilinmokda.

Fayllar tiplariga ko‘ra ikki turga bo‘linadi: tiplashgan va tiplashmagan fayllar. Tiplashgan fayllarning komponentalari biron bir tipta mansub bo‘ladi, tiplashmagan fayllarning komponentalari esa odatda *text* tipiga mansub hisoblanadn.

E’lon qilingan fayllardan foydalanish uchun ularni ochish lozim. Fayllarni ikki xil maqsadl,a ochish mumkii:

a) fayldagi ma’lumotlarni o‘qish uchun ;

b) faylga ma’lumotlarni yozish uchun .

Fayldagi ma’lumotlarni o‘qish maqsadida ochish *reset* xizmatchi suzi yordamida, faylga ma’lumotlarni yozish uchun ochish esa *rewrite* xizmatchi

so‘zlari yordamida amalga oshiriladi.

reset(b) ; b fayli ukish uchun tayyorlandi.

Rewrite(s); s fayli yozish uchun tayyorlandi.

Assign operatori e’lon qilingan faylli o‘zgaruvchilar qaysi faylni bildirishini va bu faylni qaerdan qidirish lozimligini ko‘rsatish uchun xizmat qiladi va umumiy ko‘rinishda

assign (faylli o‘zgaruvchi,'faylning adresi va nomi');

tarzida yoziladi. Masalan:

assign (f,'C:\TP\BAZA.PAS') ;

assign (g, 'F: \OMBOR\KITOB.DBF') ;

operatorlari yordamida f-faylli o‘zgaruvchi S diskning TR katalogidagi BAZA. PAS faylini bildirishi va g o‘zgaruvchi F-diskning OMBOR katalogidagi KITOB.DBF faylini anglatishi ko‘rsatilmokda. SHundan keyin f yoki g faylli o‘zgaruvchilar uchun bajariladigan barcha ishlar ana shu fayllar ustida bajariladi. Bitta faylni bir vaqtda faqat bitta maqsad bilan, yoki o‘qish uchun, yoki yozish uchun ochish mumkin. Bir vaqtda xar ikkala ishni bajarish uchun fayllarni ochib bo‘lmaydi.

O‘qish uchun ochilgan fayllardagi ma’lumotlarni *read* operatori yordamida o‘qiladi va o‘zgaruvchilarga qiymat qilib beriladi. Masalan:

read(f,x) ;

buyryg‘i yordamida f o‘zgaruvchi bilan belgilab qo‘yilgan fayldagi (*C:\TP\BAZA.PAS* faylidagi) ma’lumot o‘qiladi va x-o‘zgaruvchiga qiymat qilib beriladi.

Fayl ochilgan paytda ko‘rsatkich undagi birinchi ma’lumotni ko‘rsatib turadi. *Read* buyrug‘i bilan ana shu ko‘rsatkich ko‘rsatib turgan ma’lumot o‘qiladi, so‘ngra ko‘rsatkich navbatdagi ma’lumotga o‘tadi. Faylli o‘zgaruvchining tipi bilan undagi ma’lumotlarni qiymat qilib oladigan o‘zgaruvchilarning tiplari bir xil bo‘lishi shart.

YOzish uchun ochilgan fayllarga *write* operatori yordamida ma’lumotlar yoziladi. Bunda qavs ichida avval faylli o‘zgaruvchi yoziladi, co‘ngra ana shu o‘zgaruvchi bilan belgilab qo‘yilgan faylga yozilishi kerak bo‘lgan qiymatlarni bildiruvchi o‘zgaruvchilar ro‘yxati beriladi. Masalan,

write (g,x,y) ;

yordamida g faylli o‘zrapyvchiga (*F:\OMBOR\KITOB.DBF* fayliga) X va Y o‘zgaruvchilarning k,iymatlari yozib qo‘yiladi. Bu erda faylli o‘zgaruvchining tipi bilan qiymatlari faylga yoziladigan o‘zgaruvchilariing tiplari bir xil bo‘lishi kerak.

Joriy vaqtda ochiq bo‘lgan fayllarni *close(u)* buyrug‘i yordamida yopish

mumkin. Bu erda u-faylli o‘zgaruvchi. Ochiq fayllar keyinchalik foydalanilmaydigan bo‘lsa yoki bir maqsad bilan ochilgan faylni boshqa maqsad bilan ochishga extiyoj tushgan xollarda yopish mumkin.

YUqorida aytib o‘tildiki, fayl o‘qish uchun ochilgan vaqtda ko‘rsatkich-kursor o‘qish uchun navbatda turgan ma’lumotlarning birinchisini ko‘rsatib turadi. Bu ma’lumot o‘qilgandan so‘ng ko‘rsatkich keyingi ma’lumotga o‘tadi. Uning xolatini tekshirib turish uchun Turbopaskalda mantiqiy *eof(y)* va *eoln(y)* funksiyalari kiritilgan. *Eof(y)* fuiktsiyasi agar ko‘rsatkich fayldagi biror ma’lumotni ko‘rsatib turgan bo‘lsa "FALSE" (yolg‘op), o‘qiladigan ma’lumotlar qolmagan, ya’ni fayldagi hamma ma’lumotlar o‘qib bo‘lingan bo‘lsa "TRUE" (rost) qiymatini oladi. Xuddi shuningdek, *eoln(y)* funksiyasn satrdagi ma’lumotlarning hammasi o‘qilgan bo‘lsa "TRUE", aks holda "FALSE" qiymatini oladi.

Faraz qilaylik, y-faylida quyidagi ma’lumotlar saqlanayotgan bo‘lsin:

3 2 5 4 2 3 5 4 3 4 5 4

Fayl o‘qish uchun ochilgan bo‘lsa, dastlab ko‘rsatkich holati

3 2 5 4 2 3 5 4 3 4 5 4
↑

ko‘rinishida bo‘ladi. Demak, *EOF(y)* ning qiymati FALSE bo‘ladi. Uchta ma’lumot o‘qilgandav so‘ng ko‘rsatkich to‘rtinchi ma’lumotni ko‘rsatadi:

3 2 5 4 2 3 5 4 3 4 5 4
↑

Bu holda ham *EOF(y)* funksiyasining qiymati - FALSE. Hamma ma’lumotlar o‘qib bo‘lingandan so‘ng, ko‘rsatkich faylning oxiriga o‘tadi

3 2 5 4 2 3 5 4 3 4 5 4
↑

va *EOF(y)* funksiyasi TRUE qiymatini qabul qiladi.

Odatda fayllar bilan ishlagan vaqtda, ihtiyoriy masala kamida ikki bosqichdan iborat bo‘ladi: a) faylni yaratish; b) fayldan foydalanish. Agar talab ilingan fayl ilgari yaratilgap bo‘lsa, to‘g‘ridan to‘g‘ri nkkinchi bosqichga o‘gib, fayldagi ma’lumotlar doirasida qandaydir savollarga javob qidirish mumkin. Aks holda avval birinchi bosqichni albatta bajarish lozim.

<u>*Masala-9.*</u> 1 dan 1000 gacha bo‘lgan butun sonlar ichidagi tub sonlarni F-fayliga yozing.

<u>*Yechish g‘oyasi:*</u> Ma’lumki, agar 1 dan katga bo‘lgan butun son faqat 1 ga va o‘ziga qoldiqsiz bo‘linsa, bu sonni tub son deb ataladi. Ihtiyoriy N-butun sonning 1 dan farqli bo‘luvchilari $[2, \sqrt{N}]$ oraliqda bo‘ladi. Qo‘yilgan savolga javob berish uchun 1 dan boshlab 1000 gacha bo‘lgan hamma sonlarni ana shu oraliqdagi

sonlarga birma-bir bo‘lib chiqiladi. Agar son xech bo‘lmaganda shu sonlardan bittasiga bo‘linsa, demak bu son tub emas. Uni faylga yozilmaydi va navbatda turgan bo‘luvchilarini qidirishga o‘tiladi. Agar bo‘luvchilari bo‘lmasa, bu son faylga yoziladi va navbatdagi sonning tub yoki tub emasligini tekshirishga o‘tiladi. 2 dan boshqa barcha tub sonlar toq son bo‘lgani uchun, hisoblash ishlarini tezlashtirish maqsadida 2 ni to‘g‘ridan-to‘g‘ri faylga yoziladi. So‘nra tub sonlarni faqat toq sonlar ichidan qidiriladi.

```
Program tub ;
        var f: file of integer ;
        x, n, k : integer ;
        u : string[3] ;
begin
        assign(f, 'D:\TP\TUB.CON');
        rewrite (f) ; (*fayl yozish uchun ochildi*)
        n:=2 ; write (t,n) ;  n:=3 ;
        while n<=1000 do begin
        y:='ha' ;
        k:=3;                                   (*N sonining tub yoki tub
        while (k<=sqrt(n))and(y='ha')           emasligi tekshirilmoqda. *)
        begin
        if n mod k=0 then y:='yuq' ;
k:=k+2  end;
        if u='ha' then write(f,n);              (*tub son faylga yozildi*)
n:=n+2  end;
end.
```

YUqoridagi dasturni EHM yordamida bajarsak, D diskning TR katalogida TUB.CON fayli hosil qilinadi va unga 1 dan 1000 gacha bo‘lgan sonlar ichidagi tub sonlar yoziladi. Quyida bu faylga yozilgan ma’lumotlar ko‘p bo‘lgani uchun faqat bir qismi keltirilmokda:

```
2   3   5   7   11   13   17   19   23   29
31   37  41    43   47   53   59  61    67  ...
```

Endi bu fayldagi ma’lumotlardan foydalanib, ko‘plab masalalarni xal kilish mumkin. Keltirilayotgan masalalar ana shular jumlasidandir:

1. 1 dan 1000 gacha bo‘lgan tub sonlar fayli TUB.CON dagi sonlarning urta arifmetigini toping.
2. 1 dan 1000 gacha bo‘lgan tub sonlar fayli TUB.CON dagi "egizak" tub sonlarni aniqlang.

3. 1 dan 1000 tacha bo‘lgan tub sonlar fayli TUB.CON dagi eng katta sonnn toping. Va hokazo.

YUqoridagi ikkinchi masalani Yechish uchun quyidagicha fikr yuritish tavsiya qilinadi :

Yechish g‘oyasi: "Egizak" tub son deb orasidagi farqi 2 ga teng bo‘lgan tub sonlarga aytiladi. SHuning uchun fayldagi ketma-ket kelgan ikki tub sonning ayirmasi olinadi. Agar ayirma 2 ga teng bo‘lmasa, navbatdagi tub songa o‘tiladi va hokazo. Aks holda, orasidagi farq 2 ga teng bo‘lgan tub sonlar ekranga chiqariladi va keyingi songa o‘tiladi. Aniqlik uchun birinchi oonni A, ikkinchi sonni V bilan belgilangan bo‘lsin. Fayldagi ma’lumotlar soni avvaldan ma’lum bo‘lmagani uchun *eof()* funksiyasidan foydalaniladi.

```
program tub 2 ;
        var f: file of integer ; x, a, h : integer ;
begin
        assign(f, 'd: \tp\tub.con');
        reset(f) ;                          (* fayl o‘qish uchun ochildi.*)
        read(f,a) ;                         (* Birinchi son o‘qildi.*)
        while not (eof(f)) do begin
        read (f,b);                          (* Ikkinchi son o‘qildi.*)
        if b-a=2 then write (a,' ',b, ', ')
                                        (*"egizak" tub sonlar chiqarildi*)
        a:=b ;                              (* navbatdagi songa o‘tildi*)
        end ;
end.
```

EHM quyidagi sonlarni displeyga uzatadi:

3 5, 5 7, II 13, 17 19, ... , 857 859, 881 883

§-15. MODULLAR BILAN ISHLASH.

15.1. MODUL TUSHUNCHASI.

Ayrim protseduralardan ko‘plab masalalar uchun foydalanish mumkin. Buning uchun xar doim ana shu protseduralarni dasturlar tarkibiga kiritishga to‘g‘ri keladi. Faraz qilaylik, ikki sondan kattasi yoki kichigini topish uchun protsedura yaratilgan bo‘lsa va bu protseduradan foydalanib 10 ta masala Yechishga to‘g‘ri kelsa, 10 ta dastur tarkibiga ularni kiritish lozim bo‘ladi. Dasturchining bunday holatdagi ishini osonlashtirish maqsadida TURBO PASKAL da modul tushunchasi kiritilgan.

Modul deb protseduralar va funksiyalar kutubxonasiga aytiladi.

Turbopaskalda EHM nnng imkoniyatlaridan to‘laroq foydalapish uchun eng

ko‘p ishlatiladigan protseduralardan maxsus modullar yaratilgan. Masalan:

SYSTEM - "klassik PASKAL" ning standart buyruqlarini (protseduralari va funktsnyalarini) o‘z ichiga oladi. Bu modul hozirgacha biz ko‘rib chiqqan barcha operatorlarni o‘z ichiga oladi. Bu modul hamma dasturlar uchun ochiq.

DOS - MS DOS operatsion sistemasining turli vositalari bilan ishlash uchun protseduralar jamg‘armasi.

CRT - EHM ning klaviaturasi, displeyi va dinamigi bilan ishlash uchun protseduralar to‘plamidan iborat.

GRAPH - EHM grafik imkoniyatlaridan to‘la foydalanish uchun yig‘ilgan protseduralar to‘plami.

PRINTER - printer bilan ishlash uchun dasturlar kutubxonasi.

Turbo Paskal dasturi ishga tushgandan so‘ng, avtomatik tarzda SYSTEM moduli ishga tushadi. Agar undan boshqa modullardan foydalanishga ehtiyoj tug‘ilsa, bu modullarni ochish kerak. Buni odatda *program* buyrug‘idan keyin amalga oshiriladi va *uses* xizmatchi so‘zidan foydalaniladi.

Program dastur nomi ;

uses modullar ro‘yxati ;

Agar ko‘plab modullardan foydalanishga *to‘g‘ri* kelsa, ularni

uses *modul-1,modul-2, … , modul-N;*

tarzida ochiladi.

Quyidagi dastur "tomchilarning tushishi" jarayonini ifodalash uchun tuzilgan:

```
program tomchi ;
      uses crt ;
 begin
      repeat
      sound((140)+random(600)) ;
      delay (random(10)) ;
      nosound ;
      delay (random (1300));
      until keypressed ;
       nosound ;
 end.
```

YUqoridagi dasturda CRT modulining quyidagi protseduralaridan foydalanildi:

Sound- dinamikni ishga tushiruvchi protsedura.

Nosound- dinamikni o‘chiradigan protsedura.

delay(n) - Avvalgi buyruqni N - millisekund vaqt davomida bajarilishini ta’minlaydigan protsedura.

Keypressed - mantiqiy funksiya bo‘lib, biror tugmani bosilganda TRUE, aks holda FALSE qiymatini oladi.

15.2. YANGI MODUL HOSIL QILISH VA FOYDALANISH.

Standart bo‘lmagan va siz uchun "sevimli" bo‘lgan protseduralardan doimo modul yaratish imkoniyati mavjud. Bu ishni quyidagi ko‘rinishda amalga oshiriladi:

unit *modul nomi ;*
interface
interfeys sektsiyasi-ochiq ifodalashlar bo‘limi ;
inplementation
yopiq ifodalashlar bo‘limi ;
begin
initsializatsiya bo‘limi
end.

unit xizmatchi so‘zidan keyin modulnnng nomi yoziladi. So‘ngra *interface* so‘zidan so‘ng, ushbu modulni ishlatish uchun mo‘ljallangan ihtiyoriy dasturlar uchun modulning ochiq bo‘lgan qismi ko‘rsatiladi. Agar bu modul avval yaratilgan modullardan foydalanishni ko‘zda tutsa, ularning nomlari *interface* so‘zidan keyin, *uses* xizmatchi so‘zi yordamida aniqlanishi kerak.

Interfeys bo‘limi *interface* va *implementation* xizmatchi so‘zlari orasida joylashib, shu yaratilayotgan modulni ichiga kirayotgan protsedura va funksiyalar sarlavhalarining ro‘yxatidan iborat bo‘ladi. Bulardan tashqari modulda foydalanish uchun mo‘ljallangan konstantalar, o‘zgaruvchilar va boshqa ma’lumotlarning tiplari ko‘rsatiladi. Protsedura yoki funksiyalarning o‘zlari esa yopiq ifodalashlar bo‘limida keltiriladi. Interfeys bo‘limida berilgan xar bir protsedura va funksiyalarning to‘liq matni yopiq ifodalashlar bo‘limida yana bir marta ko‘rsatiladi. Ularning sarlavhalari har ikki bo‘limda ham aynan bir xil bo‘lishi shart. Initsializatsiya bo‘limi *begin* va *end* operatorlari orasida ko‘rsatiladi. Agar *begin* bo‘lmasa, u holda initsializatsiya bo‘limi ham bo‘lmaydi. Initsializatsiya bo‘limidan boshqaruv asosiy dasturga uzatilishidan avval bajarilishi kerak bo‘lgan operatorlar ketma-ketligi beriladi. Odatda bu operatorlar dasturni ishga tayyorlash uchun xizmat qiladi. Masalan. initsializatsiya bo‘limida o‘zgaruvchilar aniqlanishi mumkin, fayllar ochilishi mumkin va hokazo. Initsializatsiya bo‘limi ko‘rsatmalari shu modulni ishlatish uchun mo‘ljallangan dasturlarning asosiy buyruqlaridan avval bajariladi.

Agar bir nechta modullardan bitta dasturda foydalanishga to‘g‘ri kelib qolsa, ularning initsializatsiya bo‘limlari modullarning *uses* operatori yordamida ochilish navbati bilan chaqiriladi. Quyidagi kichik bir misolda modullarni yaratish

va foydalanish jarayoni keltirilmoqda. YAratilayotgan modul o‘z ichiga ikki sondan kattasi va kichigini topish uchun tashkil qilingan funksiyalarni olmoqda.

```
Unit namuna ;
    Interface (* ochik ifodalashlar bo‘limi *)
        function max(a,b:integer ): integer ;
        function mm(a,b:integer ) : integer ;
    implementation (* yopiq ifodalashlar bo‘limi *}
        function  max(a,b:integer ) : integer ;
            begin
                if a>=b then max:=a else max:=b ;
            end;
        function min(a,b:integer ) : integer ;
            begin
                if a>=b then min:=b else min:=a ;
            end;
    (* initsializatsiya bo‘limi yo‘q*)
end.
```

Endi bu dasturni kompilyatsiya qilish lozim. Buning uchun modul matnini EHM xotirasiga kiritilgandan so‘ng, modul nomi bilan bir xil bo‘lgan NAMUNA.PAS nomi bilan yozib qo‘yiladi. So‘ngra bu matnni kompilyatsiya qilinadi. Kompilyatsiya qilish uchun odatdagidek, ALT+C tugmalarini bosib, TURBO PASKAL bosh menyusining COMPILE menyusiga chiqiladi. Undagi DESTINATION ostmenyusiga kelib, MEMORY o‘rniga DISK rejimi tanlanadi. (So‘ngra yana ALT-F9 orqali COMPILE menyusiga qaytib, modul matnini kompilyatsiya qilinadi. Kompilyatsiya natijasida modul matni nomi bilan bir xil bo‘lgan NAMUNA.TPU fayli hosil bo‘ladi. SHundan keyingina NAMUNA.TPU moduli ishga tayyor hisoblanadi. Undan ikki sonning kattasi yoki kichigiii topishga to‘g‘ri keladigan barcha masalalarda foydalanish mumkin.

Eslatma : Modul matnining nomi bilan UNIT xizmatchi so‘zidan keyin kelgan modul matnining nomi bir xil bo‘lishi shart.

Masala-1: D:\TP\KITOB.DBF faylida kitoblar do‘konidagi kitoblarniig tiyinlardagi baholari saqlanadi. SHu kitoblarning eng arzoni va eng qimmatini aniqlang.

Yechish g‘oyasi: Bu masalani hal qilish uchun fayldagi eng katta va eng kichik sonni aniqlash kifoya. Buning uchun D:\TP\KITOB.DBF faylidan bnrinchi kitobning bahosini o‘qib, uni eng kattasi va eng kichigi bo‘lsin deb faraz qilinadi. Qolgan baholarni bitta-bittadan o‘qib, uni eng katta va eng kichik qiymat bilan solishtiriladi. Eng katta va eng kichik sonlarni saqlab qo‘yish uchun ***katta*** va ***kichik*** degan ikki o‘zgaruvchi kiritiladi. Solishtirish uchun NAMUNA modulining *min* va

shax funksiyalaridan foydalaniladi.

```
Program masala_ 1;
        uses namuna ;
        var x,katta,kichik: integer ;
        f: file of integer ;
begin
        assign (f, 'D:\TP\KITOB.DBF ');
        reset(f) ;
        read(f,x) ;
        katta:=x;  kichik:=x ;
        while not(eof(f)) do begin
        read(f, x);
        katta:=max(katta, x) ; kichik:=min(kichik,x) ;
        end;
        writeln ('dukondagi eng qimmat kitob:= ',katta) ;
        writeln('dykonimgi eng arzon kitob:= ',kichik)
end.
```

§-16. GRAFIKLAR BILAN ISHLASH.

16.1. EKRANNING ISH REJIMLARI.

Displey ekrani to‘g‘ri to‘rtburchak shaklidagi maydon bo‘lnb, o‘z ichiga ko‘plab nuqtalar to‘plamini oladi va ikki xil rejimda ishlashi mumkin: a) matnli rejim. b) grafik rejim.

Matnli rejimda ekranning rangi ikki xil bo‘lib, u faqat dasturlar matni hamda ma’lumotlarni kiritish uchun mo‘ljallangandir.

Grafik rejim esa tasvirlar bilan ishlash, ya’ni turli chizmalarni yaratish, o‘zgartirish va ko‘rishni uchun mo‘ljallangan bo‘lib, matnli rejimdan farqli ravishda ekranning ihtiyorny nuqtasiga turli ranglar berish mumkin. Boshqa rangga bo‘yalgan nuqtalar to‘plami to‘g‘ri chiziqlarni, turli belgi va tasvirlarni hosil qilishi yaxshi ma’lum. Matnli rejimda ekranning imkoniyati 24×80, ya’ni ekran 24 qatorga ega va har bir qatorga 80 tagacha belgini yozish mumkin.

Displey ekrani bir vaqtning o‘zida faqat yuqoridagi ish rejimlaridan birida ishlay oladi, ikki rejim bilan ishlash imkoniyatlari hozircha yaratilmagan. Buni sababi shuki, ekranda odatda EHM videoxotirasidagi ma’lumotlar tasviranadi. Joriy vaqtda videoxotirada qanday rejim belgilanganligiga qarab, undagi ma’lumotlar turlicha talqin qilinadn. Videoxotira, nur trubkasining kontrolleri, kiritish va chiqarish qurilmalari displey adapteri deb ataladigan plata tarkibida joylashgan.

EHM imkoniyatlariga karab, bir necha xil displey adapterlari yaratilgan. Ular

orasidagi asosiy farq quyidagilardan iborat:

1. Ruxsat etilgan imkoniyatlari (eng ko‘p nuqtalar sig‘imi) ;
2. Ranglarining soni bilan.

EHM zarurat bo‘lsa, matnli ish rejimidan grafik ish rejimiga o‘tkazish mumkin. SHundan keyingina, Turbopaskalning grafik protseduralari va funksiyalaridan foydalangan holda tasvirlar bilan ishlash mumkin. Bu ishni grafik drayverlar deb ataluvchi maxsus dasturlar yordamida amalga oshiriladi. Bu drayverlar vazifasini kengaytmasi .BGI (BORLAND GRAPHICS INTERFACE) bo‘lgan fayllar bajaradi.

Xozirgi vaqtda VGA (VIDEO GRAPHICS ARRAY) yoki SVGA (SUPER VGA) tipidagi grafik adapterlardai keng foydalanilmoqda. VGA tipidagi adapterlarning nuqtalar sig‘imi tanlangan drayverga bog‘liq ravishda 640x200 dan 640x480 gacha bo‘lishi mumkin. Ular bilan ishlash uchun EGAVGA.BGI drayveri yaratilgan.

16.2. EKRANNI GRAFIK HOLATGA O‘TKAZISH.

Turbopaskalda grafiklar bilan ishlash uchun *graph* moduli yaratilgan bo‘lib, bu modul o‘z ichiga grafiklar balan ishlash uchun mo‘ljallangan 80 dan ziyod protsedura va funksiyalarni oladi. *Graph* modulida VGA tipidagi adapterlar uchun drayverlarni ko‘rsatish uchun konstanta-9 raqami aniqlangan. Undan tashqari *EHM* uchun eng yaxshi drayverni tanlash operatori *detect* kiritilgan.

Ekranni grafik rejimga utkazish uchun *graph* modulining *initgraph* prodedurasidan foydalaniladi. U quyidagicha yoziladi:

initgraph (gd, gm, ‘yo‘l’) ;

Bu erda *gd* — drayver nomeri, *gm* — rejim nomeri, yo‘l-tanlangan ish rejimi uchun zarur bo‘lgan drayverlar joylashgan manzilni bildiradi. Agar *gd=0* bo‘lsa, EHM uchun eng yaxshi drayver avtomatik tarzda aniqlanadi. Eng yaxshi drayverni aniqlash uchun qiymati nolga teng bo‘lgan *detect* operatori ham kiritilgan.

Yo‘l umuman ko‘rsatilmagan bo‘lsa, drayverlar joriy katalogdan qidiriladi. Yo‘l noto‘g‘ri ko‘rsatilgan bo‘lsa, EHM zarur drayverni topa olmaganligi uchun dastur umuman ishlamasligi mumkin.

Zarur bo‘lsa, displeyni grafik rejimdan avvalgi ish rejimiga qaytarish mumkin. Buning uchun closegraph operatoridan foydalaniladi. Quyidagi dasturda grafik ish rejimini ta’minlaydigan va shu zahoti yana displeyning dastlabki ish rejimini tiklaydigan dastur matni keltirilmoqda.

```
Program grafik;
    Uses graph ;        (* graph modulini ochish *)
    var x, u : integer ;
```

```
begin
    gd:=detect; (*drayverni avtomatik tarzda aniqlash*)
    initgraph (x, u, 'S:\TR'); (*grafik rejimga o‘tish*)
    readln  ;     (* «ENTER» tugmasi bosilishini kutish*)
    closegraph ;        (* avvalgi rejimga qaytish *)
end.
```

Eslatma; Yo‘l-kengaytmasi .BGI bo‘lgan drayverlar yozilgan katalog adresidan iborat. Agar drayverlar bu adresdan topilmasa, qidirish joriy katalogda davom ettiriladi.

16.3. RANGLAR. NUQTALAR. CHIZIQLAR.

TURBO PASKAL da ranglar bilan ishlash uchun maxsus konstantalar hamda xizmatchi so‘zlar kiritilgan :

inglizcha	№	o‘zbekcha	inglizcha	№	o‘zbekcha
black	0	qora	darkgray	8	Kul rang
blue	1	ko‘k	lightblue	9	moviy
green	2	yashil	lightgreen	10	och yashil
cyan	3	kulrang	lightcyan	11	havorang
red	4	qizil	lightred	12	pushti
magenta	5	binafsha	lightmagenta	13	safsar
brown	6	jigarrang	yellow	14	sariq
lightgray	7	och kulrang	white	15	oq

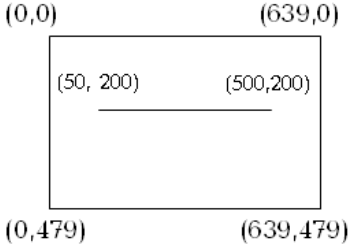
Displeyda hosil qilinadigan geometrik tasvirlarga avval raig tanlash lozim. Bu ishni setcolor operatori yordamida amalga oshiriladi va umumiy ko‘rinishda

setcolor (rang nomi yoki rangning tartib raqami) ; tarzida yoziladi. Masalan,

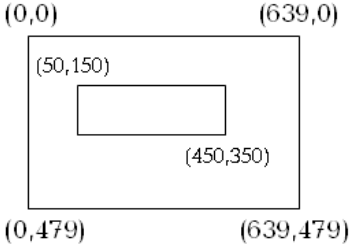
setcolor(4) ; yoki *setcolor(red) ;*

buyruqlaridan biri bajarilgandan so‘ng, to navbatdagi *setcolor* yordamida boshqa rang tanlanmaguncha chizmalar qizil rangda hosil qilinaveradi. Agar rang tanlanmagan bo‘lsa, joriy bo‘lib oq rang xizmat qiladi, ya’ni hamma tasvirlar oq rangda yasaladi.

Biz odatda geometrik chizmalar chizgan paytimizda koordinatalar boshini markazda deb



1-расм



2-расм

qabul qilib o‘rganganmiz. Displey ekrani koordinatalari esa boshqacharok. 1-rasmda uni koordinatalari qay tarzda belgilanishi ifodalangan.

Putpixcl protsedurasi displeyda nuqta tasvirini hosil kilish uchun ishlatiladi va umumiy ko‘rinishda

Putpixel (x,y, color) ;

tarzida yoziladi. Bu erda *x* va *u* nuqtaning koordinatalari, *color* -rang nomeri yoki nomi.

(*x,y*) koordinatali nuqtadagi rang nomerini aniqlash uchun *getpixel(x, u)* funksiyasidan foydalaniladi. Faraz qilaylik, ekranning (100,125) nuqtasi qizil bo‘lsin. U holda getpixel(100,125) funksiyasining qiymati 4 ga teng bo‘ladi.

line(x1,u1,x2,u2) — protsedurasi uchlari (*x1, y1*) va (*x2, y2*) nuqtalarda yotgan kesma tasvirini hosil qilish uchun ishlatiladi. Masalan, *line (50,200,500,200)* buyrug‘i yordamida 1-rasmdagi tasvirni yasash mumkin.

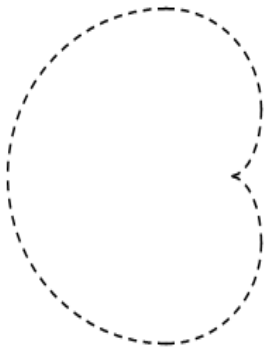
Diagonalining uchlari (x1,y1) va (x2,y2) nuqtalarda yotgan to‘g‘ri to‘rtburchakni chizish *rectangle(x1,y1,x2,u2)* protsedurasi bilan amalga oshiriladi. Rectangle(50,60,450,350); buyrug‘ining natijasi 2- rasmda ko‘rsatilgan.

Markazlari (x, y) nuqtada yotgan aylana tasvirini hosil qilish uchun *circle (x,u, radius);* protsedurasidan foydalaniladi. Bu erda *radius*- aylananing radiusini bildiradi.

<u>*Masala-11*</u>. $x = a\cos t(1+\cos t)$ va $y = a\sin t(1+\cos t)$ $(t \in [0,2\pi])$ tenglamalar bilan berilgan kardioida tasvirini hosil qiling.

<u>*Yechish g‘oyasi:*</u> Bu masalani Yechish uchun t-ning hamma qiymatlari uchun x va u o‘zgaruvchilarni ifodalovchi berilgan formulalardan foydalanib birma-bir hisoblab topiladi. So‘ngra koordinatalar boshi displey o‘rtasida (unnng koordinatasi taxminan (320,240)) ekanligini hisobga olgan holda topilgan (x,u) koordinatali nuqtalarni ekranga qo‘yib chiqiladi. Tasvirni biron bir tugmani bosilgunga qadar ekranda saqlash uchun readln protsedurasidan foydalaniladi. (3-rasm)

```
program kardioida;
uses crt, graph ;
var x,y,gd,gm:integer ; a, i : real ;
begin
write ('kardioida koyeffitsienti:=');
readln(a) ;
gd:=detect ;
initgraph(gd,gm, 'd:\tp\bgi') ;
t:=0 ;
while t<=2*pi do begin
 x := trunc(320 + a * cos(t) * (1 + cos(t));
 y := trunc(240 – a * sin(t) * (1 + cos(t));
putpixel(x,y) ;
 t := t + 0.001;
end ; readln ;
end.
```



3 – расм

16.4. CHIZMALARNI TO‘LDIRISH.

Circle, line va *rectangle* protseduralari chizmalarning faqat chetki chiziqlarini chizadi xalos. Turbopaskalda ana shu chizmalar egallaydigan sohalarni shtrixovka (maxsus belgilar) bilan to‘ldirish imkoniyati yaratilgan. *Setfillstyle* protsedurasi ana shu shtrixovkalarni belgilash uchun xizmat kiladi va umumiy ko‘rinishda

setfillstyle (styl, color)

tarzida qo‘llanadi. Bu ko‘rsatmada *styl* - joriy shtrixovka nomi yoki nomerini, *color* - joriy rang nomi yoki nomerlarini belgilaydi. Shtrixovka nomi va nomerlari quyidagicha:

emptyfill	- 0	fon rangi bilan to‘ldirish;
solidfill	- 1	berilgan rang bilan to‘ldirish;
linefill	- 2	qalin gorizontal chiziqlar bilan to‘ldirish;
Itslashfill	- 3	ingichka // chiziqlar bilan to‘ldirish;
Slashfill	- 4	qalin // chiziqlar bilan to‘ldirish;
Bkslashfill	- 5	qalin \\ chiziqlar bilan to‘ldirish ;
ltbkslashfill	- 6	ingichka \\ chiziqlar bilan to‘ldirish;
hatchfill	- 7	katakchalar bilan to‘ldirish ;
xhatchfill	- 8	qiyshiq katakchalar bilan to‘ldirish;

interleavefill - 9 zich va qalin /// chiziqchalari bilan to‘ldirish;
widedotfill -10 siyrak nuqtalar bilan to‘ldirish;
Closedotfili - 1 zich nuqtalar bilan to‘ldirish ;

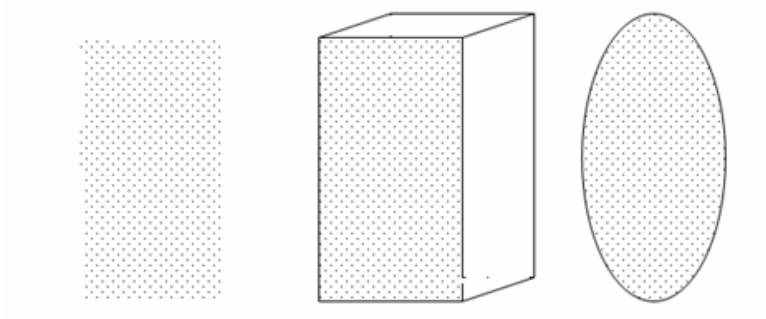
Egallaydigan sohasi, joriy shtrixovkalar bilan to‘ldiriladigan figuralar uchun quyidagi protseduralar kiritilgan:

Bag(x1,u1,x2,u2) - diagonalining uchlari (xl,yl) va (x2,u2) nuqtalarda yotgan va joriy shtrixovka bilan to‘ldirilgan to‘g‘ri to‘rtburchakni yasash uchun xizmat qiladi.

Bar3d(xl, yl, x2, y2, depth, top) - old tomoni joriy shtrixovka bilan to‘ldirilgan parallelepiped tasvirini hosil kilish uchun ishlatiladi. Bu erda *depth* - parallelopiped "chuqurligini" ifodalash uchun xizmat qilsa, *top* - parallelopiped yuqori chiziqlarini chizish yoki chizmaslikni bildiradi. Agap *top* - ning qiymati "TRUE" bo‘lsa, yuqori chiziqlari chiziladi, "FALSE" bo‘lsa - yo‘q.

Fillellipse(x, u, xradius, yradius)-protsedurasi ichki sohasi joriy shtrixovka bilan to‘ldirilgan ellipsni chizish uchun ishlatiladi. Ellipsning o‘qlari koordinatalar o‘qiga nisbatan parallel bo‘ladi. *Xradius*-ellips kengligini bildirsa, yradius - uzunligini bildiradi. Quyidagi dastur natijasi 4-rasmda keltirilgan.

```
Program tasvir ;
        uses crt, graph ;
        var x, u, gd, gm : integer ;
        a, t : real ;
begin
        gd:=detect;
        initgraph(gd,gmt 'd:\tp\bgi') ;
        setfillstyIe(11,15) ;
        bar(20,100,100,300) ;
        bar3d(200,100,300,300,40,true) ;
        fillellipse(500,200,20,100);
        readln ;
end.
```



4 – rasm

16.5. MATNLAR BILAN ISHLASH.

Ko‘pincha grafik holatda ishlab turgan vadtda ekranga qandaydir matnlarni ham chikarishga 'gugri kelib k.oladi. Bu x.olda avval chikariladigan matnlar *uchun* shrifg tanlanadi. TURBO PASKAL da ikki xil shrift bilan ishlash imkoniyati mavjud:

a) Nuqtali shriftlar; b) vektorli shriftlar.

Nuqtali shriftlarni kattalashtirgan vaqtda ular nuqgalar to‘plamidan iborat ekanligi ko‘rinadi. Vektorli shriftda esa bunday emas. SHriftlar kengaytmasi *.CHR* bo‘lgan fayllar yordamida aniqlanadi. SHuning uchun biron bir shriftdan foydalanmoqchi bo‘lsak, avval joriy katalogda yoki *.BGI* fayllari joylashgan katalogda kerakli *.CHR*-fayllari bor-yo‘qligini aniqlab olish darkor. SHriftlarni tanlash va ularni ma’lum bir masshtab bilan chiqarish *settextstyle* protsedurasi yordamida amalga oshiriladi. Uni umumiy xolda quyidagicha yozish mumkin:

settextstyle(shrift, yo‘nalish, masshtab);

Bu erda *shrift* — shriftlarni nomi yoki nomerini ko‘rsatib, quyidagi son yoki so‘zlardan biri shaklida bo‘lishi mumkin:

Defaultfont - 0 8x8 standart nuqtali shrift;
triplexfont - 1 vektorli shrift;
smallfont - 2 vektorli shrift;
sansseriffont -3 vektorli shrift;
gothicfont -4 vektorli shrift;

yo‘nalish - matnlarni qanday yo‘nalishda chiqarish kerakligini ko‘rsatadigan o‘zgaruvchi bo‘lib, ikki xil qiymat olishi mumkin:

horizdir - 0 chapdan o‘ngga;
vertdir - 1 pastdan yuqoriga.

Masshtab — belgilariing qanday kattalikda chiqarishni belgilaydigan miqdor hisoblanadi. Normal nuqtali shrift uchun bu miqdor 1 ga, vektor shriftlar uchun esa 4 ga teng.

Ekranga chiqariladigan matn

outtextxy (A, V, 'matn') ;

protsedurasi yordamida ko‘rsatiladi. Bu erda A va V sonlari matnni ekranning qaysi koordinatali nuqtasidan boshlab chiqarilishini belgilab berish uchun hizmat qiladi. Matnlarni faqat lotin xarflaridan foydalanib yozish shart.

Settextjustufy(horiz, vert) – protsedurasi ekranga chiqariladigan matnni avtomatik tarzda tenglash uchun xizmat qiladi. Bu erda *horiz* — gorizontal tenglashni, *vert* — esa vertikal tenglashni anglatadi. Tenglash uchun quyidagi nomlar va kattaliklardan foydalanish mumkin: gorizontal tenglash uchun:

lefttext - 0 chap tomondan tenglash ;
centertext - 1 markazga nisbatan tenglash ;
righttext - 2 *o*‘ng tomondan tenglash.
Vertikal tenglash uchun:
bottomtext - 0 past tomondan tenglash ;
centertext - 1 markazga nicbatan tenglash ;
toptext - 2 yuqoridan tenglash.

YUqoridagi ayrim protseduralardan foydalangan holda 'ISTIQLOL' degan matnni ikki xil shrift bilan (0 va 3 shriftlar) displeyga chiqarildi.

5-расм

Quyidagi dastur 5-rasmni hosil qilish dasturidir.

```
Program shriftlar;
        uses crt, graph ;
        var x, u, gd, gm : integer ;
begin
        gd:=detect ;
        initgraph (gd,gm, 'd:\tp\bgi') ;
        settextstyle(0,0,8) ;
        outtextxy(20,100, 'ISTIQLOL') ;
         settextstyle(3,0,8) ;
        outtextxy(20,300,'ISTIQLOL');
        readln ;
end.
```

16.6. ASOSIY GRAFIK PROTSEDURALAR BA FUNKSIYALAR.

GRAPH moduli hammasi bo‘lib 80 dan ortiq protsedura va funksiyalarni o‘z ichiga oladi. Ularning hammasini keltirib o‘tish anchagina qiyin bo‘lganligi uchun, yuqorida aytilmagai va eng ko‘p ishlatiladigan funksiyalar, protseduralar va ularning vazifalarini keltirib o‘tamiz:

ags(x, u : integer; bosh_bur, ox_bur, radius: word) - protsedurasi aylana yoyini chizish uchun xizmat qiladi. Bu erda ***bosh_bur*** - yoyning boshlang‘ich burchagi, ***ox_bur*** - *esa* oxirgi burchagi bo‘lib graduslarda ifodalanadi.

Ellipse(x, u: integer; bosh_bur, ox_hur, radius: word) - protsedurasi ellips yoyini chizish uchun xizmat qiladi. Bu erda ***bosh_bur*** - yoyning boshlang‘ich burchagi, ***ox_bur*** - esa oxirgi burchagi bo‘lib, graduslarda ifodalanishi shart.

fillpoly(bur_soni: word; var uch_koor: tipi) - protsedurasi ko‘pburchak tasvirini hosil qilish uchun ishlatiladi. Bu erda ***bur_soni*** - burchaklar soni, ***uch_koor*** - ko‘pburchak uchlari koordinatalarining massivi.

Floodfill(x, u : integer ; ch_r : word) — protsedurasi (x, u) nuqta tegishli bo‘lgan sohani bo‘yash uchun ishlatiladi. Bu erda ***ch_r*** — soha chegarasining rangi.

Getcolor(x,y : word) —funksiyasidan joriy rang nomerini aniqlash maqsadida foydalanish mumkin.

getmaxX (getmaxY) - funksiyalari joriy drayver uchun gorizontal (vertikal) nuqtalarning eng katta sonini aniqlaydi.

Linerel(x,y:integer)-ppotsedypacu ko‘rsatkich turgan pozitsiyadagi nuqtani (x,u) koordinatali nuqta bilan birlashtiradi.

Moveto(x,u: integer) - protsedurasi kursorni (x,u) koordinatali nuqtaga o‘tkazish uchun ishlatiladi.

Pieslice (x,u: integer; bosh_bur, ox_bur, radius: word)- protsedurasi sektor chizish uchun ishlatiladi. Bu erda ***bosh_bur*** va ***ox_bur*** - mos ravishda sektoriing boshlang‘ich va oxirgi burchaklarini gradusda ko‘rsatuvchi o‘zgaruvchi.

Outtext(s:string) — protsedurasi kursor turgan joydan boshlab S- matnni chiqaradi.

Textheight (textwidth)- funksiyalaridan matnlar balandligini (kengligini) belgilash maqsadida foydalaniladi.

§-17. XARAKATLI TASVIRLAR BILAN ISHLASH ASOSLARI

Bizga yaxshi ma’lumki, EHM ekranidagi nuqtalar soni o‘rtacha 640x480. Demak ekranda xarakat qilayotgan figuraning koordinatalari ana shu oraliqda bo‘lishi kerak, Biz buni albatta yodda saqlamog‘imiz lozim.

Ekranda biron bir figurani xarakatlantirish uchun quyidagi ishlarni bajarishga to‘g‘ri keladi:

1. Dastlabki pozitsiyada figura tasvirini hosil qilinadi.
2. Tasvir ma’lum bir vaqt mobaynida ekranda ushlab turiladi.
3. Tasvir o‘chiriladi (tasvirga ekran foni rangi bilan bir xil bo‘lgan rang beriladi).
4. Tasvirning navbatdagi pozitsiyasi koordinatalari aniqlanadi.
5. Tasvir yangi iozitsiyadan hosil qilinadi.
6. Ekranning chegaraviy nuqtalarini hisobga olgan holda 1-5 bosqich ehtiyojga qarab bajariladi.

<u>2-masala</u>. Ekranning o‘rtasida gorizontal yo‘nalishda xarakat qilayotgan nuqta

tasvirini hosil qiling.

Yechish g‘oyasi: YUqorida sanab o‘tilgan bosqichlarni hisobga olib, qo‘yilgan masalaning dasturi quyidagicha yoziladi:

```
program  nuqta;
        uses graph, crt;
        var gd,gm,x,k: integer;
begin
        gd:=detect;
        initgraph ( gd,gm, 'd:\tp\bgi');
        k:=1;
        x:=1;
        repeat
        if x=1 then k:=1 else
        if x=620 then k:=1;
        x:=x+k;
        putpixel(x,l50,15);
        delay(20);
        putpixel(x, 150,0) ;
        until keypressed;
end.
```

2-masala: Aylana bo‘ylab xarakat qilayotgan nuqta tasvirini hosil qilish.

Yechish g‘oyasi: Biz aylana bo‘ylab xarakat qilayotgan nuqta tasvirini hosil qilish uchun qutb koordinatalar sistemasidan foydalanamiz. Ma’lumki, nuqtaning koordinatalari $x=r\cos t$ va $y=r\sin t$ formulalar bilan aniqlanadi. Bu formulalarni hisobga olinsa, ishning qolgan qismi avvalgi masala kabi hal qilinadi:

```
program  aylana;
        uses graph, crt;
        var gd,gm,x,y,k: integer;
        fi: real;
begin
        gd:=detecl;
        initgraph ( gd,gm, 'd:\tp\bgi');
        fi:=0;
        k:=50;
        repeat
        fi:=fi+0.01;
        if fi=6.28 then fi:=0;
```

```
        x:=trunc(k*cos(fi));
        y:=trunc(k*sin(fi));
        putpixel(x+320,240-y,15);
        delay(20);
        putpixel(x+320,240-y,0);
        until keypressed;
    end.
```

3-masala: O‘z o‘qi atrofida aylanayotgan kesma tasvirini yasang.

Yechish g‘oyasi: Ma’lumki, kesmaning ikki uchi mavjud. Ularning xar ikkisi uchun yuqoridagi ikki masala birlashtiriladi.

```
    Program   kesma;
        uses graph, crt;
        var gd,gm,x1 ,y1 ,x,y,k: integer;
        fi: real;
    begin
        gd:=detect;
        initgraph ( gd,gm, 'd:\tp\bgi');
        fi:=0; k:=50;
        repeat
        fi:=fi+0.01;
        if fi=6.28 then fi:=0;
        x:=trunc(k*cos(fi) ) ;
        y:=trunc(k*sin(fi));
        x1:=trunc(k*cos(fi+3.14));
        y1:=trunc(k*sin(fi+3.14));
        setcolor(15);
        line(x+320, 240-y, x1+320, 240-y1);
        delay(20);
        setcolor(0);
        line(x+320, 240-y, x1+320, 240-y1);
        until keypressed;
    end.
```

4-masala: O‘z o‘qi atrofida aylanayotgan uchburchak tasvirini hosil qiling.

Yechish g‘oyaci: Aniqlik uchun uchburchak uchlarning koordinatalari (300,200), (400,100), (200,180) bo‘lsin. Bu uchburchakni aylantirish masalasi deyarli 3-masalaning mantiqiy davomi bo‘la oladi. SHuning uchun dasturni quyidagicha yozish mumkin:

```
program   uchhurchak;
        uses graph, crt;
        var gd,gm,x,y,x1,y1,x2,y2,k; integer;
        fi: real;
begin
        gd:=detect;
        initgraph ( gd,gm, 'd:\tp\bgi');
        fi:=0; k:=50;
        repeat
        fi:=fi+0.01;
        if fi=6.28 then fi:=0;
        x:=trunc(k*cos(fi));
        u:=trunc(k*sin(fi));
        setcolor(15);
        Line(x+300, y+200, x+400, y+100);
        Line(x+300, y+200,x+200, y+180);
        Line(x+400, y+100, x+200, y+180);
        delay(20):
        setcolor(0);
        Line(x+300, y+200, x+400, y+100);
        Line(x+300, y+200,x+200, y+180);
        Line(x+400, y+100, x+200, y+180);
        until keypressed;
end.
```

§-18. KO‘RSATKICHLAR.

Odatda dasturda qatnashayotgan har bir o‘zgaruvchi uchun EHM xotirasidan bittadan yacheyka ajratiladi. Siz o‘zgaruvchining EHM xotirasidagi qaysi yacheykaga yozilganligi bilan qiziqmasligingiz mumkin. Bu yacheykalardagi ma’lumotlarga o‘zgaruvchilar nomini ko‘rsatish orqali muomala qilinadi

Lekin, TURBO PASKAL o‘zgaruvchilarning qiymatlari saqlanayotgan adresni aniq biladi. Masalan,

```
var uzg : integer ;
```

e’loni yordamida *uzg*-o‘zgaruvchisiga EHM xotirasidan 2 bayt (butun son uchun) joy ajratish haqida kompilyatorga buyruq berilmoqda. Zarur bo‘lsa, *uzg* — o‘zgaruvchi uchun ajratilgan adresni @ operatori yoki *addr* funksiyasi yordamida aniqlash mumkin:

```
p1 := @ uzg ;
```

Endi p1 - o‘zgaruvchisining qiymati *uzg* - joylashgan adres nomeriga teng bo‘ladi.

Ko‘rsatkich deb o‘zgaruvchilarning adreslariii caqlash uchun mo‘ljallangan o‘zgaruvchilarga aytiladi.

Operativ xotiraning dasturlar, ulardagi ma’lumotlar va steklarni saqlash bilan band bo‘lmagan qismi dinamik xotira deyiladi.

Ba’zi masalalarni Yechish jarayonida ko‘rsatkichlardan foydalanishga to‘g‘ri kelib qoladi. Mana ularning ayrimlari:

-Dastur umumiy xajmi 64 kilobaytdan katta bo‘lgan ma’lumotlar bilan ishlashi zarur bo‘lsa. Dasturdagi ma’lumotlar to‘laligicha EHM ning ma’lumotlar segmentida saqlanadi. Uning hajmi esa 64 kilobayt. (Ortiqcha ma’lumotlarni iima qilish kerak ?)

- Dastur umumiy xajmi oldindan ma’lum bo‘lmagan ma’lumotlar bilan ishlashiga to‘g‘ri kelib qolsa. Ba’zi hollarda, masalan, satrlar yoki massivlarning barcha elementlarini saqlash uchun mumkin bo‘lgan eng katta joyni band qilish lozim bo‘ladi. Bu joyning hammasiga ham ma’lumot yozilmasligi mumkin. Demak, xotiradan noo‘rin va samarasiz foydalanishga yo‘l qo‘yiladi.

-Ma’lumotlarni saqlash uchun xotira buferlaridan foydalanilsa. Ko‘pincha ma’lumotlarni, masalan, matnlarni EHM xotirasiga doimiy saqlash uchun yozishdan avval, vaqtincha buferlarda saqlab turishga to‘g‘ri keladi. Buning uchun ma’lum bir hajmdagi buferlarni ajratish lozim. Bufer hajmi kichik bo‘lib, matn katta bo‘lishi yoki katta hajmdagi buferda kichkina matn saqlanishi mumkin. Bu holda ham xotira noto‘g‘ri taqsimlanmoqda.

Ko‘rsatkichlardan foydalanish xotiradan unumli foydalanishga yordam beradi. Ko‘rsatkich EHM xotirasida bor-yo‘g‘i 4 bayt joyni band qilgan holda, bir necha o‘n kilobayt joyni band qilayotgan ma’lumotlarni ko‘rsatishi mumkin.

TURBO PASKAL da ikki xil, tiplashgan na tiplashmagan ko‘rsatkichlar bilan ishlash imkoniyati mavjud.

Tiplashgan ko‘rsatkichlar ma’lum bir tipdagi ma’lumotlar yozilgan yacheykalarning adreslarini bildiradi. Tiplashgan ko‘rsatkichlarni e’lon qilish uchun avval identifikator yoziladi, so‘gra «^» belgisi qo‘yiladi va undan keyin til ko‘rsatiladi.

Tiplashmagai ko‘rsatkichlar esa ma’lum tip bilan bog‘lanmagan bo‘ladi. SHuning uchun ulardan tipi o‘zgarib turadigan ma’lumotlarni saqlash uchun qulay bo‘ladi. Tiplashmagan ko‘rsatkillarni e’lon qilish uchun *pointer* xizmatchy so‘zidan foydalaniladi.

*Var R : pointer ;(*R — tiplashmagan ko‘rsatkich)

pin : ^ integer ; (butun tiplashgan ko‘rsatkich)

X, Y : byte ; (byte tipidagi ko‘rsatkich)

p_string : ^ string ; (satr tipidagi ko‘rsatkich)

Adresi ko‘rsatkichda saqlanayotgan ma’lumotga murojaat qilish uchun ko‘rsatkich nomidan keyin "^" belgisini qo‘yish zarur.

Pin:=2 ; (*Pin*-ning adresiga 2 soni yozib qo‘yildi)

*X^:=12 ; Y^:=2 ; (*X baytga #12 qiymat, Y baytga esa #2 qiymat yozib qo‘yildi)

p_string:- ‘TURBO PASCAL' (*p_string* "*TURBO PASCAL* " qiymatini olmoqda.)

Ko‘rsatkichdan foydalangan vaqtda, u kerakli ma’lumotlarni saqlab turgan adreslarni ko‘rsatib turishi kerak. Aks holda turli anglashilmovchiliklar kelib chiqishi mumkin.

Tiplashgan ko‘rsatkichlarga qiymat berish uchun *new* protsedurasidan, tiplashmagan ko‘rsatkich uchun esa *getmem* protsedurasidan foydalaniladi.

Getmem protsedurasi — berilgan hajmdagi operativ xotirani band qiladi va uning adresini berilgan ko‘rsatkichga yozib qo‘yadn. *Getmem* protsedurasi umumiy ko‘rinishda quyidagicha yoziladi:

GetMem(var P : pointer; size : word) ;

Bu erda R - ko‘rsatkich nomi, *size* - band qilinadigan xajm.

```
Var R: pointer ;
begin
      GetMem(p,1000) ;
```

(* xotiradan xajmi 1000 bayt bo‘lgan joyni ajratib, unnng adresini R^ ga yozib qo‘yadi*)

R^dan foydadaniladigan buyruqlar ketma-ketligi}

```
FreeMem(P,l000);
```

(* *R^* ga jo‘natilgan uzunligi 1000 bayt bo‘lgan soha bo‘shatilmoqda.*)

```
end.
```

Bu dasturdagi *FreeMem* protsedurasi tiplashmagan ko‘rsatkich band qilgan joyni bo‘shatish uchun xizmat qiladi. Vaqti-vaqti bilan, nokerak ko‘rsatkichlar band qilib turgan joylarni bo‘shatib turilmasa, xotira to‘lib, yangi ma’lumotlar sig‘may qolishi mumkin.

New - protsedurasi operativ xotiradan zarur xajmdagi joyni ajratadi va uning adresini berilgan ko‘rsatkichga jo‘natadi va umumiy holda quyidagicha yoziladi :

```
new ( var P: pointer) ;
```

bu erda R — ko‘rsatkich nomi.

```
Var ps; ^string ;
begin
New(ps);
```

(*xotiradan uzunligi 256 bayt bo‘lgan (chunki *string* tipi 256 bayt joy egallaydi.) joyni ajratib, uning adresini ps^- ga jo‘natadi.*)

```
ps^:="dinamik soha "; writeln(ps^);
Dispose(ps^) ;
```

(* xotiraning ps^ qismi bsshatildi.*)

```
end.
```

New va *GetMem* protseduralari asosan to‘da (kucha) lar sohasidan berilgan o‘lchamdagi xajmni ajratib oladi va uning adresini ko‘rsatkichga yozib qo‘yadi. Agar tudada talab qilingan hajm etarlicha bo‘lmasa

runtime error 203 ... (to‘da to‘lib ketgan)

ko‘rinishidagi axborot ekranga uzatiladi. Quyida berilgan dastur ana shu holatni namoyish etishi mumkin, sababi doimo yangi soha talab qilinmoqda, ammo avvalgi sohalarni bo‘shatishga e’tibor berilmayapti.

```
Var p:pointer; i: integer;
begin
i:=0;
```

(* keltirilayotgan tsikl "cheksiz" tsikl hisoblanadi, chunki i<>i sharti hech qachon o‘rinli bulmaydi.*)

```
repeat
GetMem(p,1000);  i:=i+i; Write(i,' ');
until i<>i
end.
```

Biz yuqorida to‘da (kucha) tushunchasidan foydalandik. U nima? Dasturdagi barcha o‘zgaruvchi va konstantalar xotiraning ma’lumotlar segmentida joylashadi. Bu segment xajmi 64 kilobaytdan iborat bo‘lib, dasturdagi barcha ma’lumotlarni saqlash uchun etarli bo‘lmasligi mumkin.

```
Var A:array[1..40000] of byte; V: array [1..25534] of byte;
begin
end.
```

Dasturni kompilyatsiya qilinganda , EHM

"error 46 : data segment too Large"

(ma’lumotlar juda ham ko‘p) tarzidagi axborotni displeyga chiqarishi mumkin.

Saqlash zarur bo‘lgan ma’lumotlar juda ham ko‘p bo‘lgan holda ularni saqlash uchun xotiraning dinamik sohasi yoki to‘dadan (dastur band qilgan va ma’lumotlar segmenti, kodi, steklaridan tashqari barcha xotira qismidan) foydalanish maqsadga muvofiq hisoblanadi. To‘daning boshi va oxiri SYSTEM modulining maxsus

o‘zgaruvchilari *HeapOrg* va *HeapEnd* lar yordamida ko‘rsatiladi. To‘da band qilgan xotiraning oxiri HeapPtr funksiyasi yordamida aniqlanadi. To‘daning xajmi odatda ma’lumotlar segmentidan bir necha marta katta bo‘lib, odatda 640 kilobaytgacha (EHM ning operativ xotirasining xajmi) bo‘lishi mumkin.

To‘da xotirasi ehtiyojga qarab band qilinishi va bo‘shatilishi mumkin. SHuning uchun to‘dadagi o‘zgaruvchilarni dinamik o‘zgaruvchilar deb ataladi. Ular uchun xotirani *New* va *GetMem* irotseduralari yordamida band qilinib, *FreeMem* va *Dispose* protseduralari yordamida bo‘shatiladi.

Joriy vaqtda ko‘rsatkichlar hech narsani ko‘rsatmayotganligini ***Nil*** xizmatchi so‘zi bilan bildiriladi. Dinamik sohani bo‘shatish usullari ***Mark*** va ***Release*** protseduralaridan iborat. ***Mark*** protsedurasi *HeapPtr*-o‘zgaruvchisi joriy qiymatini (band bo‘lgan to‘da sohasi oxirini ko‘rsatuvchi ko‘rsatkich) berilgan ko‘rsatkichga yozib qo‘yadi va umumiy ko‘rinishda quyidagicha yoziladi:

Mark(var P: pointer);

Release - protsedurasi to‘dani ko‘rsatkich bilan aniqlangan adresgacha bo‘lgan qismini bo‘shatish uchun ishlatiladi.

Eslatma: *Mark* va *Release* protseduralarini *FreeMem* va *Dispose* protseduralari bilan birga ishlatish tavsiya etilmaydi.

```
    Var P: pointer; r1,r2,rZ : ^string;
begin
    new (pi) ;     (* ^pi tudada *)
    mark(p) ;      (* To‘daning joriy holati R ga yozildi *)
    new(p2) ;      (* ^p1,  r2 tudada *)
    new(p3) ;      (* ^r1, ^r2 va ^r3 tudada *)
    release (R) ; (* tudada ^r1 qoldi xalos, ^r2 va ^r3 bilan
                     band qilingan xotira qismi bo‘shatildi.*)
end.
```

19. RO‘YHATLAR VA ZANJIRLAR

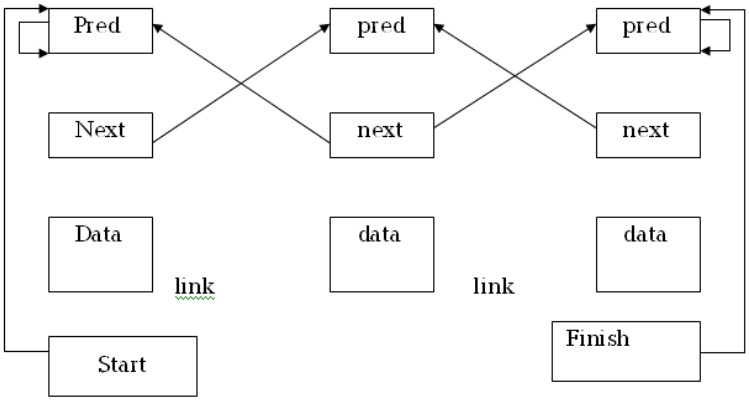
19.1. RO‘YHAT VA ZANJIR TUSHUNCHALARI

Ushbu ma’ruzadan maqsadimiz ma’lumotlarni bitta ketma-ketlikka yoki boshqacha ayttanda zanjirga birlashtirishni o‘rganishdan iborat.

Start va finish - zanjirning boshlanishi va oxirini ko‘rsatuvchi ikki ko‘rsatkich bo‘lsin. (Keyingi sahifadagi rasmga qarang.)

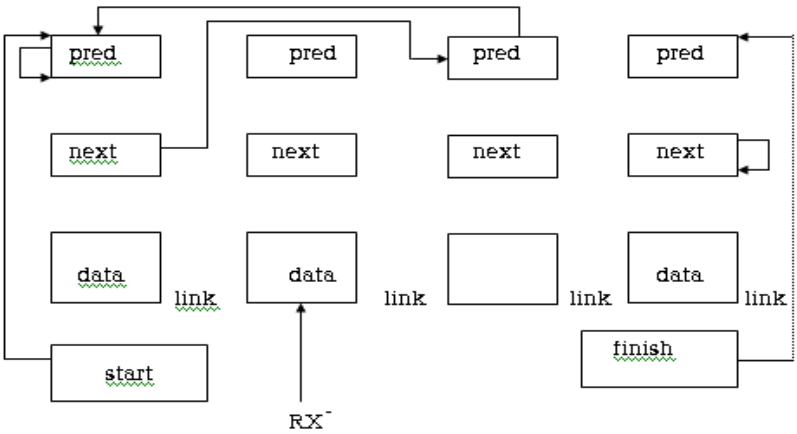
Halqaning har bir elementi *link* tipidagi yozuvdan iborat bo‘ladi. Zanjir qurilgandan keyin har bir halqaning *pred* maydoni avvalgi halqaning *pred* maydonini ko‘rsatadi, *next* esa navbatdagi halqaning *pred* maydonini ko‘rsatadi. Zanjirlar

massivlarga nisbatan murakkabroq bo‘lib, quyidagi afzalliklarga ega:
- oldindan ma’lumotlarning soni noaniq bo‘lsa. Masalan, katalogdagi fayllar ro‘yxatini EHM xotirasiga yozish uchun mo‘ljallangan programmada. Agar uning o‘rniga massivdan foydalanishga to‘g‘ri kelsa, uni iloji boricha kattaroq, o‘lchamda olish lozim bo‘ladi.



Masalan: *M: array[1.. 1000] of link;*
Zanjirni esa kerakli o‘lchamda qurib, ma’lumotlarni ozayishi yoki ko‘payishiga qarab, kengaytirish yoki qisqartirish mumkin. I va I+1 elementlar orasiga yangi halqani qo‘shishga yoki bir nechta halqani qisqartirishga to‘g‘ri kelsa-chi? Bunda yangi halqani qo‘shish orqali xotiraiipg zarur qismn band qilinadi yoki halqani uzib tashlab, xotiraning nokerak qismi tozalanadi.

21.2. NOKERAK HALQANI UZIB TASHLASH.



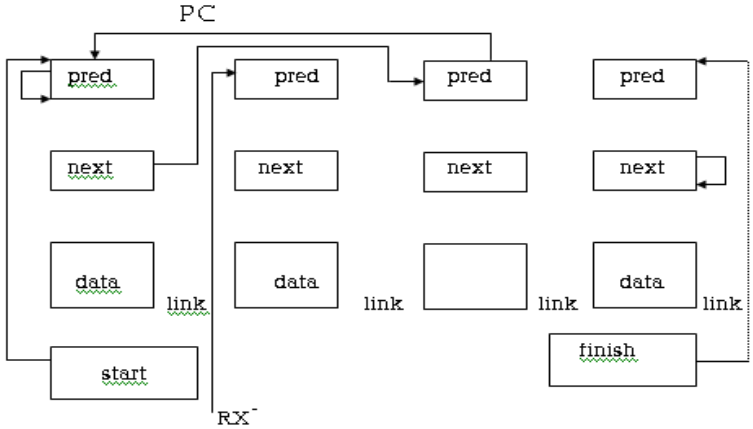
Faraz qilaylik, zanjirdan RX^ haqani uzib tashlashga to‘g‘ri kelgan bo‘lsin. Buniig uchun quyidagi amallarni bajarish lozim:

RX^.pred^.next:=RX^.next;RX^.next^.pred:=RX^.pred;

19.3. ZANJIRGA YANGI HALQANI QO‘SHISH

Faraz qilaylik, zanjirning RX^ ko‘rsatkich ko‘rsatayotgan halqasidan so‘ng

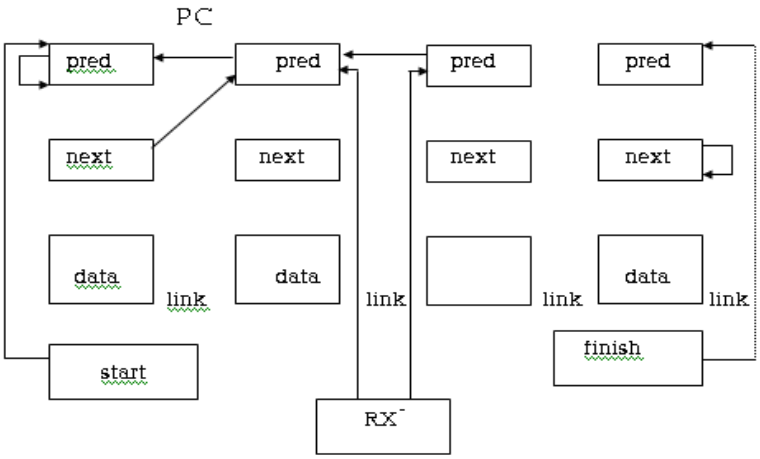
yangi halqani qo‘shishga to‘g‘ri kelgan va uning adresi RX bo‘lsin. Birinchi navbatda yangi halqani xosil qilamiz. *New(RX);*



So‘ngra, halqani zanjirga quyidagi programma yordamida qo‘shamiz:

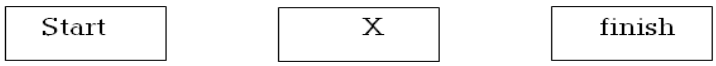
RX^.next:=PC^.next; RX=pred:=PC;

PC.next:=RX^; RX^=next.pred:=RX;



19.4. ZANJIRLARNI QURISH

Zanjirning boshlanishi *start* va oxiri *finish* bo‘lsin. Navbatda turgan halqaning adresi X^- ko‘rsatkich bo‘lsin.

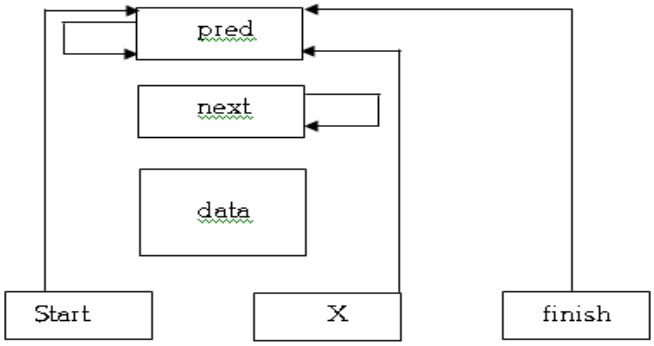


Dastlab X halqa uchun xotiradan joy ajratamiz.

NEW(X);

va uni *start* hamda *finish* ko‘rsatkichlar bilan bog‘laymiz:

X^.next:=Nil; X^.pred:=Nil; start:=X; finish:=X;

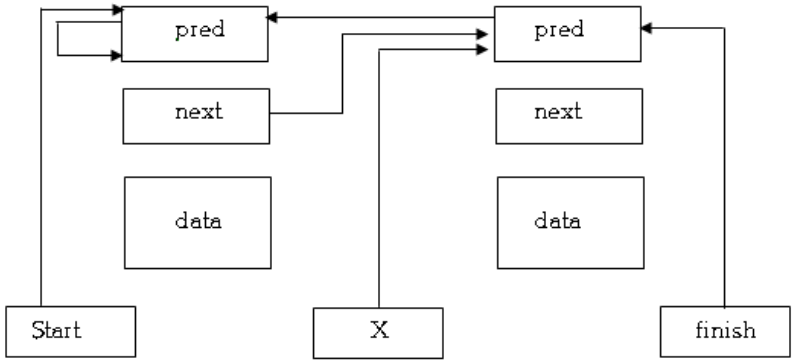


Endi navbatdagi halqa qurib, zanjir oxiriga qo‘shiladi.

X^.next:=Nil; finish^.next:=X; X^.pred:=finish;

va shundan keyin *finish* ko‘rsatkichi yangi halqaga bog‘lanadi.

Finish:=X;



Bu jarayon hamma halqalarni qo‘shib olinguncha davom ettiriladi.

MUNDARIJA