O’ZBEKISTON RESPUBLIKASI

OLIY VA O’RTA MAXSUS TA`LIM VAZIRLIGI

NAMANGAN MUHANDISLIK-PEDAGOGIKA INSTITUTI

«Informatika va AT»
kafedrasi

ALGORITMIK TILLAR

fanidan ma’ruzalar matni

Namangan 2010

«Algoritmik tillar va dasturlash» fanidan tayyorlangan ushbu ma`ruzalar matni 5140900 «Informatika va axborot texnologiyalari» kasbiy ta`lim yo'nalishlari bo'yicha ta`lim olayotgan kunduzgi, sirtqi va maxsus sirtqi bo'lim talabalari uchun mo'ljallangan bo'lib, NamMPI ilmiy kengashida tasdiqlangan na`munaviy o'quv dasturi asosida tuzildi. Ma`ruzalar matnida matematik modellashtirish, masalani EHMda yechish bosqichlari, algoritmlar nazariyasi, algoritmik tillarning umumiy tavsifi, Turbo-Paskal tilining asosiy tushunchalari, operatorlari, Turbo-Paskal muxiti bilan ishlash va uning qo'shimcha imkoniyatlar beruvchi modullari haqida ma`lumotlar keltirilgan.

Ma`ruzalar matnidan Paskal tilini mustaqil o'rganuvchilar, kasb-xunar kollejlari va akademik-litseylarning talabalari hamda o'qituvchilari foydalanishlari mumkin.

Mualliflar: - «Informatika va AT» kafedrasining mudiri,
dotsent S. S. Irisqulov;
- «Informatika va AT» kafedrasining assistenti
D.Bekmirzayev

Taqrizchi: - P. Karimov - NamMPI «Informatika va AT» kafedrasini dotsenti

Taklif va muloxazalarni hisobga olib, ma`ruzalar matnining qayta ishlanib to'ldirilgan xoli «Informatika va AT» kafedrasining _____ yil «___» _____ № ___-sonli yig'ilishida muhokama qilingan va ma`qullangan.

Mundarija

«Jahon sivilizasiyasiga daxldor bo'lgan eng zamonaviy ilmlarni egallamay turib, mamalakat taraqqiyotini ta`minlash qiyin»

Islom Karimov

So'z boshi

Insoniyat tarixining ko'p asrlik tajribasi ezgu g'oyalardan va sog'lom mafkuradan mahrum biron-bir jamiyatning uzoqqa bora olmasligini ko'rsatdi. Shu bois, mustaqillik tufayli mamlakatimiz o'z oldiga ozod va obod Vatan, erkin va farovon hayot barpo etish, rivojlangan mamlakatlar qatoridan o'rin olish, demokratik jamiyat qurish kabi ezgu maqsadlarni qo'ydi.

Bu esa kelajagimizni yaqqol tasavvur etish, jamiyatimizning ijtimoiy-ma`naviy poydevorini mustahkamlash ehtiyojini tug'diradi. Demak, galdagi eng asosiy vazifa: yosh avlodni Vatan ravnaqi, yurt tinchligi, xalq farovonligi kabi olijanob tuyg'ular ruhida tarbiyalash, yuksak fazilatlarga ega, ezgu g'oyalar bilan qurollangan komil insonlarni voyaga yetkazish, jahon andozalariga mos, kuchli bilimli, raqobatbardosh kadrlar tayyorlashdir.

«Jahon sivilizasiyasiga daxldor bo'lgan eng zamonaviy ilmlarni egallamay turib, mamalakat taraqqiyotini ta`minlash qiyin», - degan edilar prezidentimiz I. Karimov. O'zbekistonning iqtisodiy va ijtimoiy sohalarda yuqori natijalarga erishishi, jahon iqtisodiy tizimida to'laqonli sheriklik o'rnini egallay borishi, inson faoliyatining barcha jabhalarida zamonaviy axborot texnologiyalaridan yuqori darajada foydalanishning ko'lamlari qanday bo'lishiga hamda bu texnologiyalar ijtimoiy mehnat samaradorligini oshishida qanday rol o'ynashiga bog'liq.

Prezidentimiz Islom Karimovning ko'p yillik izlanishlari, asarlaridagi fikr-mulohazalariga tayanib yaratilgan «Milliy istiqlol g'oyasi: asosiy tushuncha va tamoyillar» nomli risolada ta`lim-tarbiya jarayonining ixtiyoriy bosqichida amal qilish lozim bo'lgan quyidagi mezon va talablar keltirilgan:

- o'quv mashg'ulotlarini olib borishda talabalarning yoshi, tafakkuri, dunyoqarashi va qiziqishlarini hisobga olish;
- ta`lim-tarbiyaning ilg'or, ta`sirchan vositalaridan, zamonaviy o'qitish texnologiyasi imkoniyatlaridan keng foydalanish;
- ayrim tushunchalarni haddan ziyod soddalashtirish, ta`limning eskicha uslub va tamoyillarini qo'llash natijasida fanning qadrsizlanishiga yo'l qo'ymaslik;
- ta`lim jarayonida tazyiq o'tkazmasdan ma`rifiy asosda ish tutish, yoshlarning mustaqil va erkin fikrlash, bahs-munozara yuritish ko'nikmalarini oshirishga e`tibor qaratish;
- o'qituvchi va tinglovchilar orasida o'zaro hamfikrlik va hamkorlik muhitini shakllantirish, mavzuning tushuncha va tamoyillarini sharhlashda hayotiy misollar, bugungi dunyoda ro'y berayotgan voqealar tahlilidan, matbuot materiallaridan keng foydalanish;
- yoshlarda g'oyalar, o'z ma`no-mohiyatiga ko'ra bunyodkor yoki vayronkor bo'lishi haqidagi hayotiy va haqqoniy tasavvurlarni shakllantirish;
- milliy istiqlol g'oyasining insonparvarlik mohiyatini ko'rsatish asosida mustaqillik biz uchun eng oliy qadriyat, uni asrab-avaylash esa har birimizning muqaddas burchimiz ekanini talabalarning qalbi va ongiga singdirish.

Yuqorida aytilgan mezon va talablarga rioya qilgan xolda Respublikamizda, zamonaviy hisoblash texnikasi vositalaridan samarali foydalanishni uddalay oladigan, zamonaviy komp yuterlardan amaliy ish faoliyatida keng foydalana oladigan yetuk kadrlar tayyorlash dolzarb vazifalardan hisoblanadi. Shuning uchun, kadrlar tayyorlash milliy dasturining ikkinchi bosqichida yuqori malaka, raqobatbardosh kadrlar tayyorlash uchun

sifatli, jahon andozalariga mos darsliklar, o’quv qo’llanmalari va ma`ruza matnlarini tayyorlab, chop ettirish masalasiga juda katta e`tibor berilgan.

Mazkur ma`ruzalar matni ham davlatimiz chaqirig’iga munosib javob sifatida va uzoq yillik pedagogik faoliyatimizning maxsuli tarzida yaratildi.

Bu ma`ruzalar matnidan 5140900 kasb ta`limi («Informatika va axborot texnologiyalari») ta`lim yo’nalishida tahsil olayotgan talabalar, paskal tilini mustaqil o’rganuvchilar, fan o’qituvchilari, kasb-xunar kollejlari va akademik-litseylarning talabalari foydalanishlari mumkin.

Ma`ruzalar matni 36 soatlik ma`ruza darslariga mo’ljallangan bo’lib, har bir ma`ruza bo’yicha reja, nazariy va amaliy ma`lumotlar hamda nazariy savollar va tayanch iboralar berilgan.

Mualliflar

1-ma`ruza: Matematik modellashtirishning asosiy tushunchalari, masalalarni EHMda yechish bosqichlari va algoritmlar nazariyasi.

Reja:

1. *Kirish;*
2. *Matematik model tushunchasi;*
3. *Statsion va nostatsion modellar;*
4. *Parametrlari to'plangan va tarqoq modellar;*
5. *Masalani EHMda yechish bosqichlari;*
6. *Algoritm tushunchasi va uning vazifasi;*
7. *Algoritmni ifodalash usullari, uning xossalari va unga qo'yiladigan talablar.*

1. Kirish

Qadim zamonlardan beri inson o'z imkoniyatlarini kengaytirishga harakat 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 hisobiga 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 uchrovchi masalalar yechimini qabul qilish 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 – komp yuterlashtirish asrida insoniyat faoliyatining barcha jabhalariga komp yuterlar jadal sur`atda kirib bormoqda.

Zamonaviy komp yuterlarning ko'payib borishi esa tabiiy ravishda undan foydalanuvchilarning safini ortib borishiga turtki bo'ladi. Odatda komp yuterdan foydalanuvchilar sinfi juda ham xilma-xildir. Lekin, umumiy qilib ularni komp yuterlardan o'z ishlarini bajarishda tayyor programma mahsuloti sifatida foydalanuvchi-operatorlar sinfi va ular uchun zarur bo'lgan programma ta`minotlarini yaratuvchi-dasturchilar sinfiga ajratish mumkin. Dasturchilar sinfini esa o'z navbatida shartli ravishda sistemali va amaliy dasturchilar guruxlariga ajratamiz.

Mazkur «Algritmik tillar va dasturlash» fanidan yozilgan ma`ruzalar matni amaliy dasturchilar guruxiga tegishli mutaxassislarni, institutning «Informatika va axborotlar texnologiyasi» kafedrasida «Informatika va AT» yo'nalishi bo'yicha ta`lim olayotgan kasbiy ta`lim bo'yicha muhandis-muallimlarni Turbo-Paskal algoritmik tiliga o'rgatish uchun mo'ljallangan.

2. Matematik model tushunchasi

Elektron hisoblash mashinalari bilan bevosita ishlashdan oldin qanday ishlarni bajarish kerakligini ko'rib chiqaylik. Istalgan hayotiy, matematik yoki fizik va hokazo

masala shartlarini ifoda qilish dastlabki ma`lumotlar va fikrlarni tasvirlashdan boshlanadi va ular qat`iy ta`riflangan matematik yoki fizik va hokazo tushunchalar tilida bayon qilinadi. So'ngra 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 hodisaning mohiyati, belgilari, ishlatiladigan ko'rsatkichlar so'zlar yordamida batafsil ifoda etiladi, so'ngra fizik qonunlar asosida kerakli matematik tenglamalar keltirilib chiqariladi. Bu tenglamalar o'rganilayotgan fizik jarayon yoki hodisalarning matematik modeli deb ataladi. Matematik modelni haqiqiy ob`ektga moslik darajasi amaliyotda tajriba orqali tekshiriladi. Odatda, matematik model qaralayotgan ob`ektning xususiyatlarini aynan, to'la o'zida mujassam qilmaydi. U har xil faraz va cheklanishlar asosida tuzilgani uchun taqribiylik xarakteriga ega, tabiiyki uning asosida olinayotgan natijalar ham 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 baholash masalasi modellashtirishning asosiy masalalaridan biridir.

Matematik modellarni shartli ravishda quyidagi turlarga ajratish mumkin.

3. Stasionar modellar va nostasionar 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 ham 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 hosilalari vaqtga bog'liq funksiya qatnashgan differensial tenglama yoki shunday tenglamalar sistemasi, xususiy hosilali differensial tenglamalar yordamida yoziladi.

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 bo'yicha hosilalar qatnashuvchi oddiy differensial tenglamalar yoki shunday tenglamalar sistemasi kabi tenglamalar bilan ifodalanadi.

Bunday modellarda umuman olganda qaralayotgan jarayon ko'rsatkichlari ham vaqtga, ham fazoviy o'lchovlarga bog'liq bo'ladi. Modellar asosan xususiy hosilali differensial tenglamalar yordamida ifodalanadi. Xususiy 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 ham berilishi mumkin. Masalan, ularni chiziqli va chiziqli bo'lmagan, bir o'lchamli va ko'p o'lchamli kabi guruhlarga ajratish mumkin.

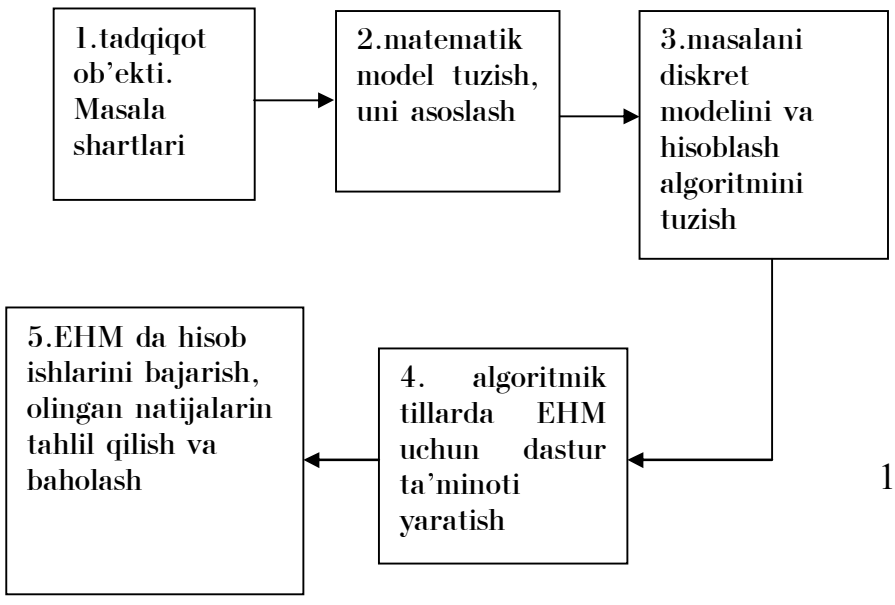
Shuni ham ta`kidlash lozimki, har doim ham qo'yilgan masalaning matematik modelini yaratib bo'lavermaydi.

5. Masalalarni EHMda yechish bosqichlari

Matematik model har xil vositalar yordamida berilishi mumkin. Bu vositalar fizik qonuniyatlar hamda funktsional analiz elementlarini ishlatib differensial va integral tenglamalar tuzishdan to hisoblash algoritmi va EHM 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 EHMda bajariladigan hisoblashlar asosida tahlil qilish - hisoblash tajribasi deyiladi. Hisoblash tajribasining umumiy sxemasi 1-rasmda ko'rsatilgan.

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



1-rasm

Ikkinchi bosqichda fizik, mexanik, kimyoviy va boshqa qonuniyatlar asosida matematik model tuziladi. U asosan algebraik, differensial, integral, integro-differensial va boshqa 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 model juda murakkablashib ketadi. Shuning uchun, model tuzishda qaraliyotgan 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 yechilishi va kerakli ko'rsatkichlar aniqlanishi lozim. Ayrim xollarda 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 EHMda yechishning uchinchi bosqichini tashkil qiladi.

Navbatdagi bosqichda, ya`ni, to'rtinchi bosqichda, masalani EHMdan foydalanib yechish uchun uning yechish algoritmi ishlab chiqiladi, hamda shu algoritm asosida biror-bir zamonaviy algoritmik tilda EHMda 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 yetarlicha 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 EHMga kiritiladi va sozlanadi hamda olingan natijalar chuqur tahlil qilinib, baholanadi. Natijalarni tahlil qilish, zarur bo'lgan hollarda algoritmni, yechish usulini va modelni aniqlashtirishga yordam beradi, hattoki masalani noto'g'ri qo'yilganligini ham baholab berishi mumkin.

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

6. Algoritm tushunchasi va uning vazifasi

Algoritm so'zi o'rta asrlarda paydo bo'lib, buyuk o'zbek mutafakkiri Al-Xorazmiyning (783-855) ishlari bilan yevropaliklarning birinchi bor tanishishi bilan bog'liqdir. Bu ilmiy ishlar ularda juda chuqur taasurot qoldirib algoritm (*algoritmi*) so'zining kelib chiqishiga sabab bo'ldiki, u Al-Xorazmiy ismining lotincha aytilishidir.

Algoritm deganda, berilgan masalani yechish uchun ma`lum tartib bilan bajarilishi kerak bo'lgan chekli sondagi buyruqlar ketma-ketligini tushuniladi.

Biror masalani komp yuterda yechishda eng muxim va ma`suliyatli ishlardan biri qo'yilgan masalani yechish algoritmini yaratish bo'lib, bu jarayonda bajarilishi kerak bo'lgan xamma bo'lajak buyruqlar ketma-ketligi aniqlanadi. Ma`lumki, komp yuterning o'zi xech qanday masalani yechmaydi, balki programma ko'rinishida yozilgan algoritmni bajaruvchi hisoblanadi xolos. Shuning uchun, algoritmda yo'l qo'yilgan xato

xisoblash jarayonining noto'g'ri bajarilishiga olib keladi, bu esa o'z navbatida yechilayotgan masalaning xato natijasiga olib keladi.

Biror soxaga tegishli masalani yechish algoritmini yaratish, algoritm tuzuvchidan shu soxani mukammal bilgan xolda, qo'yilgan masalani chuqur tahlil qilishni talab qiladi. Bunda masalani yechish uchun kerak bo'lgan ishlarning rejasini tuza bilish muxim axamiyatga ega. Shuningdek, masalani yechishda ishtirok yetadigan ob`ektlarning qaysilari boshlang'ich ma`lumot (masalani yechish uchun zarur bo'lgan ma`lumotlar) va qaysilari natijaligini aniqlash, ular o'rtasidagi o'zaro bog'lanishni aniq va to'la ko'rsata bilish lozim.

7. Algoritmni ifodalash usullari, uning xossalari va unga qo'yiladigan talablar

Masalani yechishning algoritmini turli usullar bilan ifodalash mumkin:

— so'z bilan;
— blok-sxemalar shaklida;
— formulalar orqali;
— algoritmik tillar orqali va x.z.

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

1. Algoritm xar doim bir qiymatlidir, ya`ni uni bir hil boshlang'ich qiymatlar bilan ko'p marta qo'llash har doim bir hil natija beradi.
2. Algoritm birgina masalani yechish qoidasi bo'lib qolmay, balki turli-tuman boshlang'ich shartlar asosida ma`lum turdagi masalalar to'plamini yechish yo'lidir.
3. Algoritmni qo'llash natijasida chekli qadamdan keyin natijaga erishamiz yoki natijaga erishish mumkin emasligi haqidagi ma`lumotga ega bo'lamiz.

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

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

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

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

- xuddi shuningdek, *a,b,c* larga doim bir hil qiymatlar bersak, algoritm har doim bir hil natija beradi, ya`ni to'liqdir;

- yaratilgan bu algoritm faqatgina bitta kvadrat tenglamaning yechish algoritmi bo'lib qolmay, balki u *a,b,c* larning mumkin bo'lgan barcha qiymatlari uchun natija xosil qiladi va shu turdagi barcha kvadrat tenglamalarning yechish algoritmidir;

- algoritmning oxirigi xossasi o'z-o'zidan bajariladi, ya`ni kvadrat tenglamani yechish albatta chekli qadamda amalga oshiriladi.

Dastur tuzuvchi uchun EHMning ikkita asosiy parametri eng muximdir: komp yuter xotirasining xajmi va tezkorligi. Shuningdek, algoritm tuzuvchidan ikki narsa talab qilinadi. Birinchidan, u tuzgan dastur komp yuter xotirasidan eng kam joy talab etisin, ikkinchidan, eng kam amallar bajarib masalaning natijasiga erishsin. Umuman olganda, bu ikki talab bir-biriga qarama-qarshidir, ya`ni algoritmning ishlash tezligini oshirish, algoritm uchun zarur xotirani oshirishga olib kelishi mumkin.

Algoritm tuzishda quyidagilarga amal qilinsa, qo'yilgan masalaning natijasini tez va to'g'ri olish mumkin:

— qo'yilgan masalani to'g'ri o'qish va tushinib olish, masalaning asosiy maqsadini ajrata bilish;
— ishga dahldor qiyinchiliklarni aniq ko'rish va ortiqcha, masala yechimiga katta ta`siri bo'lmagan parametrlarni yo'qota bilish;
— qo'yilgan masalani bir-biriga bog'liq bo'lmagan mustaqil bo'laklarga ajrata olish va ular orasidagi bog'liqlikni to'g'ri tashkil etish;
— qo'yilgan masalaning yechimini olishda har bir bo'lak yechimlarni to'plamini bir butun holga keltirish;
— masala yechimini sodda va tushunarli tilda bayon eta olish.

Nazariy savollar va tayanch iboralar:

1. Algoritm tushunchasi qanday paydo bo'lgan?
2. Al-Xorazmiyning hayoti va ijodi haqida nimalarni bilasiz?
3. Algoritm tushunchasiga ta`rif bering.
4. Masalani yechish algoritmini qanday usullar bilan ifodalash mumkin?
5. Algoritmga qo'yiladigan talablarni sanab o'ting.
6. Algoritmning qanday xossalarini bilasiz?
7. Masala yechimi algoritmini tuzishda qanday qoidalarga amal qilish lozim?
8. Chiziqli algoritm deb nimaga aytiladi?
9. Tarmoqlanuvchi algoritmni ta`riflang va unga hayotiy masalalar keltiring.
10. Takrorlanuvchi algoritmlarning vazifasini tushuntiring.
11. Algoritmlarni ifodalashda blok-sxemalardan foydalanishning afzalliklarini aytib bering.
12. Blok-sxema elementlarini chizib ko'rsating va ularning vazifalarini ayting.
13. Chiziqli tenglamalarni yechish algoritmini tuzing.
14. Kvadrat tenglama ildizlarini aniqlash algoritmini ishlab chiqing va uni blok-sxemalar orqali ifodalang.
15. Xalqa yuzini aniqlash algoritmini yarating.

16. Ismingizni yozishda qatnashgan xarflar sonini aniqlash algoritmini tuzing va ularni blok-sxemalar orqali ifodalang;
17. Masalani o'rganish nimadan boshlanadi?
18. Matematik model deb nimaga aytiladi?
19. Stasionar misollarga misollar keltiring.
20. Qanday modellar nostasionar modellar deb ataladi?
21. Faqat vaqt faktoriga bog'liq modellar qanday modellar deb ataladi?
22. Parametrlari tarqoq modellarni qanday tushunasiz?
23. Masalaning modeliini doim ham qurish mumkinmi?
24. Hisoblash tajribasi deb nimaga aytiladi?
25. Masalani EHM yordamida yechish necha bosqichda amalga oshiriladi?
26. Tadqiqot ob`ektini tanlash va masala shartlarini aniqlash bosqichining vazifasi nimadan iborat?
27. Matematik model tuzish va uni asoslash bosqining vazifalarini tushuntiring.
28. Diskret model va hisoblash algoritmi bosqichi qanday ishlarni o'z ichiga oladi.
29. Masalani yechishning dastur ta`minotini yaratish bosqichining vazifalarini tushuntiring.
30. Dastur natijalarini taxlil qilish va baxolash bosqichida bajariladigan ishlar ketma-ketligini ayting.
31. Qo'yilgan masalaning noto'g'ri qo'yilganligini qanday aniqlash mumkin?
32. Matematik modellarga doir misollar keltiring.

2-ma`ruza: Algoritmik tillarning umumiy tavsifi va metalingvistik formulalar tili.

Reja:

1. *EHMning «mashina tili» va uning imkoniyatlari;*
2. *Algoritmik tillar va ularning afzalliklari;*
3. *Algoritmik tillarning alfaviti, sintaksisi va semantikasi;*
4. *Algoritmik tillarga qiyosiy tavsifnoma;*
5. *Bekus-Naurning metalingvistik formulalar tili.*

1. EHMning «mashina tili» va uning imkoniyatlari

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

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

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

2. Algoritmik tillar va ularning afzalliklari

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

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

mashina tili alfavitidan algoritmik til alfavitining o’ta kengligi - tuzilgan programma matnining ko’rinish sifatini keskin oshiradi;

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

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

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

mashina uchun ko’zda tutilgan ma`lumot tiplaridan tashqari yangi tiplar kiritish imkoniyati yaratilganligi.

Shunday qilib, qaysidir ma`noda aytish mumkinki, algoritmik tillar mashina tiliga bog’liq emas.

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

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

A=n, B=m deylik

Agar *A=B* bo’lsa 5-punktga, aks holda 3-punktga o’t.

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

2-punktga o’t.

EKUB=A va hisobni to’xtat.

Ushbu algoritmni qisqaroq ko’rinishda quyidagicha ifodalashimiz ham mumkin:

A=n, B=m deylik;

Agar *A>B* bo’lsa *A=A-B* aks holda *B=B-A, A=B* bo’lguncha 2-punktni takrorla.

EKUB=A va hisobni to’xtat.

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

- algoritmni ortiqcha ko’p so’zli va uzun deyish mumkin;
- bir xil ma`nodagi ko’rsatmani turli xil uslublarda berish mumkinligi;
- bunday erkin ko’rinishda ifodalangan algoritmni EHM tiliga o’tkazish imkoniyati kamligi.

3. Algoritmik tillarning alfaviti, sintaksisi va semantikasi

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

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

Til sintaksisi alfavit harflaridan tashkil topib, mumkin bo’lgan konstruktsiyalarni aniqlovchi qoidalar sistemasidir. Mazkur tilda ifoda etilgan to’la algoritm va uning alohida hadlari shu konstruktsiyalar orqali ifoda qilinadi. Shunday qilib, belgilarning har qanday ketma-ketligini, mazkur tilning matni to’g’riligi yoki noto’g’riligini til sintaksisi orqali bilib olamiz.

Til semantikasi algoritmik tilning ayrim konstruktsiyalari uchun qoidalar sistemasini tushuntirishga xizmat qiladi.

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

4. Algoritmik tillarga qiyosiy tavsifnoma

Hozirda insoniyat faoliyatining barcha jabhalariga shaxsiy elektron hisoblash mashinalari (SHEHM) shaxdam qadamlar bilan kirib bormoqda. Asosan ShEHMlarga mo’ljallangan, hamda murakkab jarayonlarning hisob ishlarini bajarish va juda katta ma`lumotlar tizimi bilan ishlashni tashkil etuvchi yangi algoritmik tillar sinfi borgan sari kengayib bormoqda.

Bu tillar jumlasiga quyidagi ko’proq ishlatilayotgan tillarni kiritish mumkin:
Beysik tili;

Paskal tili;
Si tili va hakozo.

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

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

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

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

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

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

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

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

5. Bekus-Naurning metalingvistik formulalar tili

Algoritmik tilning sintaksis qoidalarini yozish va tushuntirish, hamda o'rgatish uchun oraliq til lozim bo'ladi. Bu til bilan programmalashning algoritmik tillarini almashtirib yubormaslik kerak. Kiritilishi kerak bo'lgan bu tilning o'zi, aslida programmalash tilini aniqlash uchun kerak bo'ladi. Tilni ifodalashda Bekus-Naurning metalingvistik formulalaridan (BNF) foydalaniladi.

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

Metaformulaning o'ng va chap qismlari "::q" belgisi bilan ajratiladi, uning ma`nosi "aniqlanishi bo'yicha shunday" jumlasiga yaqinroq (ruschasi "eto yest "). Bu belgining o'ng tomonida meta o'zgaruvchi, chap tomonida esa meta o'zgaruvchining qiymatlari to'plami yotadi. Tushunishga oson bo'lishi uchun meta o'zgaruvchilar, yoki ularning qiymatlari burchak qavs ("<" va ">") lar ichiga olib yoziladi. Masalan

<son>, <arifmetik ifoda>, <operator> va hakozo.

Meta o'zgaruvchilarning meta qiymatlari bir necha mumkin bo'lgan konstruktsiyalardan tashkil topishi mumkin. Bu holda konstruktsiyalar o'zaro tik chiziq (|) bilan ajratiladi. Bu belgi "yoki" so'ziga yaqinroq ma`noni anglatadi.

Metaformulalarga doir quyidagi masalalarni ko'rib chiqaylik:

1-misol:

<o'zgaruvchi>:: = A | B

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

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

A, B, A + A, A + B, B + A, B+ B, A - A, A - B, B - A, B - B.

2-misol:

<ikkilik raqam>::= 0|1

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

bu misoldagi metaformulaning o'ng tomonida aniqlanayotgan tushunchaning o'zi yotibdi, bu esa metaformulalarning rekursiv xossasiga ega ekanligini ko'rsatadi.

Meta formulalarni yozishda "{", "}" qavslari ham uchrab turadi. Bu qavs ichiga olib yozilgan konstruktsiya takrorlanuvchi konstruktsiya hisoblanadi. Yuqorida ko'rib chiqilgan 2-misolni quyidagicha yozsak ham bo'ldi:

<ikkilik raqam>::= 0|1

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

Nazariy savollar va tayanch iboralar:

1. «Mashina tili» atamasining mazmunini ayting.
2. Mashina kodi nimani ifodalaydi?
3. Algoritmik tillarning «Mashina til»laridan asosiy farqlarini sanab bering.
4. Algoritmik tillar «Mashina tili»ga bog'liqmi?
5. Translyator qanday vazifani bajaradi?
6. Algoritmning uchta o'zagini sanab bering va ularning ma`nosini tushuntiring.
7. Til alfaviti nima?
8. Tilning sintaksis qoidalari nimani ifodalaydi?
9. Til semantikasining vazifasi nimadan iborat?
10. Qanday algoritmik tillarni bilasiz?
11. Algoritmik tillarning paydo bo'lish tarixi va evolyutsiyasi haqida nimalarni bilasiz?
12. Algoritmik tillarda qiyosiy tavsifnoma bering.
13. Bekus-Naurning metalingvistik tili nima uchun kerak?
14. Metaformulalarga misollar keltira olasizmi.

3-ma`ruza: Tilning alfaviti va tilning asosiy tushunchalari. Programma matnini yozish qoidalari.

Reja:

1. *Paskal tilining alfaviti;*
2. *Operatorlar;*
3. *Ismlar va identifikatorlar;*
4. *E`lonlar;*
5. *O'zgaruvchilar;*
6. *Funktsiyalar va proseduralar;*
7. *Programma matnini yozish qoidalari.*

1. Paskal tilining alfaviti

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

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

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

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

Raqamlar sifatida oddiy arab raqamlari olingan:

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

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

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

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

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

Solishtirish amallarining belgilari, ularning matematik ifodasi va amallarning ma`nosi 1-jadvalda o’z ifodasini topgan. Bu yerda shu narsaga ahamiyat berish kerakki, ba`zi bir amallar ikkita belgi orqali ifodalangan.

1-jadval

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

Ajratgichlar guruhini quyidagi belgilar tashkil qiladi:

<ajratgich>::= . | , | : | ; | (|) | [|] | { | } | ' | :=

Ajratgichlarning vazifalarini tilni o’rganish davomida aniqlab boramiz.

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

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

2. Operatorlar

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

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

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

S; S; ...;S.

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

3. Ismlar va identifikatorlar

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

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

Agar quyidagi oraliq tushunchani kiritsak:

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

Yuqoridagi aniqlashni quyidagicha ham yozish mumkin:

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

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

Ismlarga misollar:

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

4. E`lonlar

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

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

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

5. O'zgaruvchilar

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

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

6. Funksiyalar va proseduralar

O'rta maktab kursidan funksiya tushunchasi bizga yaxshi ma`lum. Algoritmik tillarda, faqat qiymatini hisoblash algoritmlari ma`lum bo'lgangina funksiyalar ishlatiladi. Programma tuzuvchi programma uchun lozim bo'lgan keraklicha funksiyalarni o'z programmasiga kiritishi mumkin.

Xuddi funksiyalar kabi hal qilinayotgan masalaning ma`lum bir tugallangan bosqichlarini hisoblash vazifasini proseduralar zimmasiga yuklasa ham bo'ladi.

Fukntsiyani hisoblash natijasida faqat, yagona natijaviy qiymatga erishiladi, proseduradan foydalanganda esa, natijaviy qiymatlar soni yetarlicha ko'p bo'lishi mumkin.

Programmada aniqlangan funksiya va proseduralar o'zgaruvchilarning e`loni bo'limida e`lon qilinib qo'yilishi kerak. Bunda har bir funksiya va proseduraga, ularning bajaradigan vazifasiga mos ismlar berib qo'yiladi. Ularni aniqlashda formal parametrlardan foydalaniladi. O'z navbatida, bu parametrlarning tiplari funksiya va proseduraning ichida aniqlanilib, e`lon qilinadi.

Programmada aniqlangan funksiya va proseduralardan foydalanish uchun programma matnida ularning ismlari va formal parametrlarga mos, faktik parametrlari berilishi kerak.

Ma`lumki, matematika kursidagi elementar funksiyalardan programma tuzishda juda ko'p foydalanishga tug'ri keladi. Masalan $\sin x, \cos x, \ln x, e^x$ va hokazo. Bunday funksiyalarni boshqa tillardagi kabi standart funksiyalar deb ataladi va standart funksiyalarning ismlaridan boshqa maqsadda foydalanish maqsadga muvofiq emas.

7. Programma matnini yozish qoidalari

Har bir algoritmik tilning programma matnini yozish qoidalari turlicha bo'ladi. Programmalash tillaridan eng soddasi Beysik tilining ma`lum versiyalarida, programmaning har bir operatori qat`iy aniqlangan qator nomerlari orqali yoziladi. Paskal tilida esa, operatorlar ketma-ket yozilib, o'zaro ";" belgisi bilan ajratib boriladi. Bundan tashqari, yozilgan programmaning o'qishga oson va undan foydalanish qulay bo'lishligi uchun programmada "matnni ajratish" tushunchasidan foydalaniladi (bo'sh joy, qatorni tugashi va izohlar).

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

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

Izoxlar programmani o'qishga oson bo'lishi, uni qiynalmay tekshirib, yo'l qo'yilgan xatolarni to'g'rilash va programmada bajarilayotgan ishlarni tushuntirib borish uchun qo'yiladi. Izoxsiz yozilgan programmani hujjat sifatida qabul qilinmaydi. Muvaffaqiyatli qo'yilgan izoh programmaning va programma tuzuvchining katta yutug'i hisoblanadi. Izoxlar ixtiyoriy vaqtda programma matniga kiritilishi yoki olib tashlanishi mumkin. Bu bilan programmaning ishi o'zgarib qolmaydi. Izoxlarni "{" va "}" qavslari ichiga olinib yoziladi.

Programma "matn ajratgich"laridan foydalanishning quyidagi qoidalariga amal qilish lozim:

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

Quyida yuqoridagi qoiadalar asosida yozilgan programmaga doir misol keltirilgan.

Misol. Quyidagi berilgan funksiyalarning qiymatlarini [a,b] oralig'idagi *xqaQih,* $h=\frac{b-a}{n}$ lar uchun (*n*-berilgan son) hisoblash programmasini tuzing: $f_1(x)=x^2$,$f_2(x)=3-x$, $f_3(x)=0,5-\sin x$

```
Program P1;
 { f1(x)=x*x; f2(x)=3-x; f3(x)=0.5-sin(x) funksiyalar qiymatini [a,b]
 oralig'ida hisoblash programmasi }
  const
     n=10; {[a,b] oraliqni 10 ta bo'lakchalarga ajratdik}
  Var
     a,b:real;
     i:integer;
     x,h,y1,y2,y3:real;
   Begin
      read(a,b);                {[a,b] oraliqni chegaralarini ajratish}
      h:=(b-a)/n;  x:=a;  i:=0;  {Boshlang'ich ma`lumotlar hisoblandi}
      Repeat
          y1:=x*x;
          y2:=3-x;
          y3:=0.5-sin(x);
          Writeln (x,y1,y2,y3); {Funksiyalar hisoblanib, natijalar chop
                                 etilmoqda}
           x:=x+h;  i:=i+1;
       Until i=n+1
                                 {Hisob ishlari yakunlandi}
   end.
```

Nazariy savollar va tayanch iboralar:

1. Til alfaviti nima?
2. Til alfavitining BNF formulasini yozing.
3. Asosiy belgilarni sanab bering.
4. Hizmatchi so'zlarni yozing va ularning ma`nosini tushuntiring.

5. Til operatori qanday tushuncha?
6. Asosiy va tarkibiy operatorlarni ajratib sanab bering.
7. Operatorlar qanday tartibda bajariladi?
8. Programma operatorlariga ismlar qanday qo'yiladi va nima uchun qo'yiladi?
9. Ism(identifikator) ning BNF formulasini yozing.
10. E`lonlarning vazifasini tushuntiring.
11. O'zgaruvchi programmaning qanday ob`ekti hisoblanadi?
12. Funktsiya va proseduraning asosiy farqlarini sharqlang.
13. Programma matnini yozish qoidalarini yozish qoidalarini tushintirib bering.
14. Sodda paskal-programmalar tuzing va ularni sharxlang.

4-ma`ruza: Turbo-Paskal muhitini o'rnatish va muhitda ishlash.

Reja:

1. *Turbo-Paskal muhitini o'rnatish;*
2. *Turbo-Paskal muhitining asosiy fayllari;*
3. *Turbo-Paskal muhitida ishlash.*

1. Turbo-Paskal muhitini o'rnatish;

Turbo-Paskalning professional programmalash - 5.5 versiyasi (Turbo Pascal Professional) Borland International Inc firmasining bir nechta disketalarida beriladi. Shu disketalarning biri «INSTALL/COMPILER» deb nomlanadi va bu disketada INSTALL.EXE programmasi mavjud.

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

Faraz qilaylik, Turbo-Paskalni o'rnatishda 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/TVISION, D:/TP/UTILS

2. Turbo-Paskal muhitining asosiy fayllari

Turbo Paskalning asosiy fayllari D:/TP/ katalogida joylashgan bo'lib, ular quytdagilardir:

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

Turbo.hlp – tezkor yordam ma`lumotlarini saqlovchi fayl;

Turbo.tp – Turbo.exe programmasi foydalanadigan sistema konfiguratsiyasining fayli;

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

Tptour.exe – integrallashgan muhitda ishlashni tanishtiruvchi programma.

Bulardan tashqari, D:g'TPg'BGI katalogida Turbo Paskalning grafik rejim ishini ta`minlovchi fayllar mavjud:

Graph.tpu – Turbo Paskalning barcha grafik programmalarini ishlashi uchun zarur fayl;

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. Turbo-Paskal muhitida ishlash

Bugungi kunda kasb-hunar kollejlarida va oliy ta`lim muassasalarida Turbo Paskalning 5.5 – versiyasi keng tarqalganligini va undan foydalanish qulayligini hisobga olingan holda, quyida Turbo Paskalning 5.5 – versiyasi muhitida ishlash o'rgatiladi.

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 yuqori menyu hosil bo'lib, uni aktivlashtirish uchun, *F10* tugmachasidan foydalaniladi.

Yuqori menyu quyidagi bo'limlarni o'z ichiga oladi:

- ✓ *File* – *.pas kengaytmali fayllari va Turbo Paskalga tegishli bo'lgan kataloglar ustida turli amallar bajaradi;
- ✓ *Edit* – yuqori menyudan matnlarni tahrirlash oynasiga qaytish;
- ✓ *Run* – programmani turli ko'rinishda ishga yuklaydi va natijalar oynasiga o'tishni ta`minlaydi;
- ✓ *Compile* – programmani kompilyatsiya qilish va uni tekshirishni ta`minlaydi;
- ✓ *Options* – Turbo Paskal muhitini va undagi programmalarga tegishli xususiyatlarni taxrirlaydi;

- ✓ *Debug* – programmadagi proseduralarni izlash, hisoblash jarayonlari va o'zgaruvchilarni tekshirish, chaqirilgan stekni ko'rish va shunga o'xshash amallarni bajaradi;
- ✓ *Break/Watch* – agarda *Debug/Integrated Debugging* ustanovkasi *On* xolatda bo'lsa, u holda *Break/Watch* maenyusi *Watch window* (tekshirish oynasi) punktini oynaga o'rnatadi yoki olib tashlaydi;

Quyida yuqori menyuning mos ostki menyulariga qisqacha ta`riflar berilgan:

Files bo'limi:

- *Load (F3)* – fayl yoki dasturni chaqirish uchun hizmat qiladi;
- *Pick (ALT+F3)* – tanlashni belgilaydi;
- *New* – yangi dastur uchun oynani tozalaydi;
- *Save (F2)* – ekrandagi dasturni tashqi xotiraga joylaydi;
- *Write to* – ishchi faylni yangi nom bilan yozadi;
- *Change dir* – faol katalogni almashtiradi;
- *Os shell* – vaqtincha OS ga chiqishni tashkillaydi;
- *Quit (Alt+X)* – TURBO PASCAL muhitidan chiqish.

Run bo'limi:

- *Run (Ctrl+F9)* – yuklangan dasturni bajartiradi (ishga tushiradi);
- *Program reset (Ctrl+F2)* – oldindagi parametrlarni taxrirlash;
- *Goto cursor (F4)* – kursor turgan joyga qaytishni amalga oshirada;
- *Trace into (F7)* – dasturni qadamlab bajarish imkoniyatini hosil qiladi;
- *Setup over (F8)* – proseduraga kirishni ta`minlaydi;
- *User screen (Alt-F5)* – natijalar oynasiga o'tish.

Compile bo'limi:

- *Compile (Alt+F9)* – dasturni kompilyatsiya qiladi;
- *Make (F9)* – dasturni kompilyatsiya qilishga tayyorlaydi;
- *Build* – dasturni kompilyatsiya qilishga tayyorlaydi (*Make* dan farqi, dasturga tegishli barcha fayllarni tekshiradi);
- *Destination* – dasturni tezkor yoki tashqi xotiraga kompilyatsiya bo'lishini ta`minlaydi;

- *Find error* – dastur xatoligini izlaydi;
- *Primary files* – qaysi **.pas* fayl kompilyatsiya qilinishini aniqlaydi;
- *Get info* – ishchi oynadagi **.pas* fayli haqidagi ma`lumotlarni beradi.

Options bo'limi:

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

Debug bo'limi:

- *Evaluate* – har qanday o'zgaruvchining qiymatini ko'rish imkoniyatini yaratadi;
- *Call stack* – dastur bajarilish vaqtida joriy chaqiruvchi stekni ko'rsatadi;
- *Find procedure* – 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 xususiyatini o'rnatadi;
- *Refresh display* – ekran xususiyatini aktivlashtiradi.

Break/Watch bo'limi:

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

Yuqoridagilarga qo'shimcha qilgan holda dastur matnini taxrirlash uchun funktsional tugmachalardan ham foydalanish imkoniyati mavjud bo'lib, ular quyidagilardan iboratdir:

- *Ctrl+K+B* – blok boshini belgilash;
- *Ctrl+K+K* – blok ohirini 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.

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'piroq ishlatiladi?
4. Turbo.exe faylining vazifasini tushuntiring.
5. Turbo-Paskal muhitining yuqori 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 G' Watch bo'limining ostki menyulari vazifasini ayting.
12. Programma matnini taxrirlash uchun ishlatiladigan funktsional tugmachalar vazifalarini tushuntiring.
13. Ixcham programma tuzib, uni to'liq taxrirdan o'tkazing.

5-ma`ruza: Paskal tilining asosiy tiplari va Paskal programmaning strukturasi.

Reja:

1. *Butun sonli tiplar;*
2. *Haqiyqiy sonli tiplar;*
3. *Belgili va qatorli tiplar;*
4. *Mantiqiy tiplar;*
5. *Yangi tiplarni loyixalash;*
6. *Programmaning umumiy ko'rinishi;*
7. *Metka(tamg'a)lar bo'limi;*
8. *O'zgarmaslar bo'limi;*
9. *Tiplarni aniqlash bo'limi;*
10. *O'zgaruvchilar bo'limi;*

11. Protsedora va funksiyalar bo'limi;
12. Operatorlar bo'limi.

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

Butun qiymatli tipga tegishli songa misollar:

-1501, 0, 9999.

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

Haqiqiy qiymatli tipga tegishli sonlarga misollar:

25.0956, 6.75, -321.936, 1.2E+02, -3.57E-01

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

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

Belgilarning ixtiyoriy yig'ilmasi (ketma-ketligi) qatorlar deb ataladi. Misol 'Axmad', '$25', '_START'. Qator xatto bo'sh ham bo'lishi mumkin (' '). Bu tipdagi o'zgaruvchilarni e`lon qilish uchun String xizmatchi so'zidan foydalaniladi.

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

Ko'rsatgichlar ma`lumotlarning komp yuter xotirasidagi turar joyi (adresi)ni aniqlab beradi va ularni e`lon qilish uchun *Pointer* xizmatchi so'zidan foydalaniladi.

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

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

1. Butun sonlar

Butun qiymatli tiplarning barchasi quyidagi jadvalda keltirilgan:

Tip ko'rinishi	Mazkur tipli o'zgaruvchining qabul	O'zgaruvchining komp yuter

	qiladigan qiymatlar oralig'i	xotirasidan egallaydigan joyi
ShortInt	-128..127	8 bit
Integer	-32768..32767	16 bit
LongInt	-2147483648.. 2147483647	32 bit
Byte	0..255	8 bit
Word	0..65535	16 bit

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

Endi shu tipdan foydalanishga doir quyidagi misolni ko'rib chiqaylik:

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

```
{Programma sarlavhasini yozamiz}
Program Sonlar;
Var  {programmada foydalanish mumkin bo'lgan barcha o'zgaruvchilar shu
      var so'zidan so'ng e`lon qilinadi}
   m,n:integer;        {m va n o'zgaruvchilar o'rtacha kattalikdagi butun
                        sonlar}
  k1,k2,k3:integer;{k1=m+n, k2=m-n, k3=m*n – bajarilgan arifmetik
                        amallar natijasini xotirada saqlash uchun tanlangan
                        butun tipli o'zgaruvchilar}
begin
        {Paskal dasturi begin (boshlanmoq) so'zi bilan boshlanib,
         end.(tamom) so'zi bilan tamomlanadi}
   Readln(m,n);
            {m va n butun sonlarini kiritish so'ralyapti, agar kiritilayotgan
            son butun bo'lmasa "Error 106:Invalid numeric format." xatosi
            habar qilinadi va programma ishini to'xtatadi}
   k1:=m+n;
   k2:=m-n;
   k3:=m*n;                {so'ralgan amallar bajarildi}
   writeln (k1,k2,k3);  {hisoblangan k1=m+n, k2=m-n, k3=m*n natijaviy
                           qiymatlarni chop etish tashkil etildi}
end.                       {Programma tamom bo'ldi}
```

Bundan tashqari, Turbo-Paskalda o'n oltilik sanoq sistemasida yozilgan butun sonlardan foydalanishga ham ruhsat beriladi. O'n oltilik sanoq sistemasidagi butun sonni aniqlashda uning oldiga "$" (dollar) belgisi qo'yiladi.

Misol, $11 o'nli sanoq sistemasidagi 17 ga, $12 soni esa 18 ga teng.
Endi shu holatga doir quyidagi sodda programmani keltiramiz:

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

2. Haqiqiy sonlar

Haqiqiy sonlar matematika kursidan ma`lum bo'lgan oddiy o'nlik kasr sonlardir. Agar komp yuteringiz matematik soprotsessorli bo'lsa quyidagi jadvalda sanab o'tilgan barcha tiplar o'rinli bo'ladi:

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

Agar komp yuterda matematik soprotsessor bo'lmasa, faqat *Real* tipinigina ishlatish mumkin.

Haqiqiy sonlarga doir quyidagi misolni ko'raylik:

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

```
Program arifmetika;
VAR
```

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

3. Belgilar va qatorlar

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

Misol: 'A', yoki #60.

Qator – bu ′(apostrof) belgisi ichiga olib yozilgan belgilarning oddiy ketma-ketligidir: 'Ab21#9!cd', 'Toshbayev Dilmurod'.

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

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

Belgilar va qatorlarga doir quyidagi sodda programmani keltiramiz:

```
Program String;
Var
   ch: char; {ch o'zgaruvchi belgili qiymat qabul qiladi}
   qator1,qator2:String; {qator1 va qator2 o'zgaruvchilar uzunligi 255 dan
                          ortmagan qatorlarni o'zlashtirishi mumkin}
  N:String[5]; {N o'zgaruvchisi 5 ta belgidan tashkil topgan qatorlarni
                o'zlashtiradi}
BEGIN
   ch:='A'; {ch o'zgaruvchisi A belgini o'zlashtirdi}
   N:='Ascar'; {N o'zgaruvchisi 5 ta harfli Ascar so'zini o'zlashtirdi}
```

qator1:=ch+'li '+N; {qator1 o'zgaruvchisi natijaviy Ali Ascar so'zini o'zlashtirdi}

qator2:=''; {qator2 o'zgaruvchisi bo'sh qatorni ifodalayapti lekin, bu o'zgaruvchi uchun xotiradan 256 bayt joy ajratilgan}

end.

4. Mantiqiy tiplari

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

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

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

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

Ixtiyoriy, qiymatlarni solishtirish amali ham mantiqiy qiymatni beradi:

Misol: 3>2 natijasi ***true***

0<-1 natijasi ***false***.

5. Yangi tiplarni loyihalash

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

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

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

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

6. Programmaning umumiy ko'rinishi

Paskal - programmasi – programma sarlavhasi va nuqta bilan tugovchi programma tanasidan tashkil topgan. Programma sarlavhasi va programma tanasini ; (nuqta vergul) belgisi bilan ajratiladi:

<Paskal programmasi>::=<programma sarlavhasi>;< programma tanasi>.

Programma sarlavhasi ***program*** xizmatchi so'zidan boshlanadi va undan so'ng programmaga foydalanuvchi bergan nom yoziladi:

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

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

<blok>::= <metkalar bo'limi>
<ûzgarmaslar bo'limi>
<yangi tiplar bo'limi>
<ûzgaruvchilar bo'limi>
<prosedura (qism programma) va funksiyalar bo'limi>
<operatorlar bo'limi>

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

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

Program < programma ismi>;
label
<metkalar ro'yxati>;
const
<o'zgarmaslar va ularning qiymatlari>;
type
<ma`lumotlarning yangi, nostandart tiplarini aniqlash>;
var
<o'zgaruvchilarni, proseduralar va
funksiyalarni e`lon qilish>;
begin
<operatorlar bo'limi>

end.

7. Metkalar bo'limi

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

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

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

Misol: ***label*** L1, L2, A3;

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

```
Label  m10, m20, StopLabel, 1;
Var
      I:ShortInt;
Begin
      1:
         If  i<10 Then Goto m10 Else Goto m20;
   m10:  Writeln('i  kichik 10 dan');
         Goto StopLabel;
   M20:   i:=i-1;
         Goto 1;
   StopLabel:
End.
```

8. O'zgarmaslar bo'limi

O'zgarmas - bu programmani ishlashi davomida o'zgarmay qoladigan miqdordir. Agar miqdor programmada ko'p marta ishlatilsa, uni programma matnida qayta - qayta yozgandan ko'ra, bu miqdorni o'zgarmas deb aniqlab olib, programmadagi miqdorni o'rniga o'zgarmasni ismini yozish qulay bo'ladi. Masalan, hammaga ma`lum π (π=3,1415926535…) soni. Bu sonni bir necha marta takroran programmada yozish noqulay, shuning uchun, uni o'zgarmas sifatida aniqlab olish maqsadga muvofiqdir.

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

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

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

<ûzgarmaslar bo'limi>::=<bo'sh>| ***const*** <ûzgarmasni aniqlash>;

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

9. Tiplarini aniqlash bo'limi

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

Tip e`lon qilish quyidagi metaformula asosida amalga oshiriladi:

<tip e`loni>::=<tip ismi>=<tip>;
<tip>::=<tip ismi>|<tip vazifasi>;

Tiplarni aniqlash bo'limi esa quyidagicha aniqlanadi:

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

Misol:
```
type
Mantiq=boolean;
B=integer;
Hafta=(dush, sesh, chor, pay, ju, shan, yaksh);
Ish_Kuniqdush..ju;
```

10. O’zgaruvchilar bo’limi

Har qanday programmada o’zgaruvchilar deb ataluvchi programma ob`ektlaridan foydalaniladi. O’zgaruvchi - qiymat qabul qiluvchi ob`ektdir. Ularga qiymat programmani bajarilishi davomida beriladi. Har bir o’zgaruvchiga uni qabul qiladigan qiymati va vazifasiga qarab ismlar beriladi. Shunday qilib, o’zgaruvchi o’zining nomi va qabul qiladigan qiymati bilan xarakterlanadi.

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

<o’zgaruvchilar bo’limi>::=<bo’sh> | ***Var*** <o’zgaruvchilar e`loni>;
<ûzgaruvchilar e`loni>::=<o’zgaruvchi ismi>:<tip>

Misol:

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

Programmada ishlatilgan o’zgaruvchilar faqat bir martagina e`lon qilinishi kerak.

11. Prosedura va funksiyalar bo’limi

Boshqa tillardagi kabi, Paskal tilida ham programmani yozuvchi o’zi uchun qulay va zarur bo’lgan prosedura va funksiyalarni muomalaga kiritishi mumkin. Tabiiyki bu prosedura va funksiyalar ham xuddi o’zgaruvchilar kabi e`lon qilinib qo’yilishi kerak. Hamma ishlatiluvchi prosedura va funksiyalarga ism berilib, shu ismi bilan ular chaqirilib programmaning zarur joyida ishlatiladi. Ularga murojaat xuddi oddiy standart funksiyalarga murojaat kabi amalga oshiriladi.

Prosedura va funksiyalar bo’limi o’zgaruvchilar bo’limining davomi bo’lib ***procedure*** yoki ***function*** xizmatchi so’zlari bilan boshlanib, ixtiyoriy ketma-ketlikda e`lon qilinadi.

12. Operatorlar bo’limi

Bu bo’lim programmaning eng asosiy bo’limi bo’lib, programma bo’yicha hisoblanuvchi barcha ishlar shu bo’limda bajariladi. Bo’limning metaformulasi quyidagicha yoziladi:

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

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

Nazariy savollar va tayanch iboralar:

1. Paskal programmaning metaformulasini yozing.
2. Programma sarlavxasining vazifasini tushuntiring.
3. Programmaning tanasi (blok) qanday bo'limlardan tashkil topgan?
4. Programma strukturasini yozing.
5. Metkalar bo'limining vazifasi nimadan iborat?
6. Metkalar bo'limining metaformulasini yozing.
7. O'zgarmaslar bo'limining metaformulasini yozing.
8. O'zgarmaslar bo'limidan qachon foydalanish mumkin?
9. Yangi, nostandart tiplarni qanday aniqlanadi?
10. Prosedura va funksiyalar qanday e`lon qilinadi?
11. Operatorlar bo'limining vazifalarini tushuntiring.
12. Asosiy va xosilaviy tiplarning farqini tushuntiring.
13. Butun qiymatli tiplarni aytib bering va ularni tavsiflang.
14. Haqiyqiy qiymatli tiplarni sanab bering va ularni tushuntiring.
15. Belgili tiplarni qanday aniqlanadi?
16. Qatorlar, ya`ni belgilar ketma-ketligini qanday tushunasiz?
17. Mantiqiy tiplar qanday aniqlanadi?
18. Mantiqiy amallarni bajarish jadvalini tuzing.
19. Yangi tiplar qanday kiritiladi?

6-ma`ruza: Paskal tilining operatorlari. O'zlashtirish operatorlari.

Reja:

1. *Paskal tilining operatorlari;*
2. *O'zlashtirish operatorlari;*
3. *Arifmetik o'zlashtirish operatori;*
4. *Mantiqiy o'zlashtirish operatori;*
5. *Belgili o'zlashtirish operatori;*

1. Paskal tilining operatorlari

Programmaning asosiy vazifasi boshlang'ich ma`lumotlarni qayta ishlab, qo'yilgan masalaning natijasini beruvchi amallarni bajarishdan iborat. Algoritmik tillarda biror-bir masalani yechishda ma`lumotlar ustidagi amallarni bajarish operatorlar

zimmasiga yuklatiladi. Programmalash tillaridagi har bir operator, ma`lumotlarni qayta ishlash jarayonining mustaqil bosqichi bo'lib, mantiqan yakunlangan hisoblanadi. Programmada yozilgan operatorlarni to'g'ri talqin qilish uchun ularni yozish qoidalari (operatorning sintaksisi) qat`iy aniqlangan bo'lishi shart.

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

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

2. O'zlashtirish operatori

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

<o'zlashtirish operatori>::q<ûzgaruvchi>:q<ifoda>;

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

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

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

3. Arifmetik o'zlashtirish operatori

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

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

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

7 div 2 = 3
5 div 3 = 1
-7 div 2 = -3
-7 div –2 = 3
2 div 5 =0
3 div 4 = 0

Mod – butun sonlar bo'linmasining qoldig'ini aniqlaydi. m mod n qiymat faqat n>0 dagina aniqlangan. Agar m≥0 bo'lsa m mod nqm-((m div n)*n), m<0 bo'lsa m mod nqm-((m div n)*n)Qn, m mod n ning natijasi doim musbat sondir. Misol:

7 mod 2 = 1
3 mod 5 = 3
(-14) mod 3 = 1
(-10) mod 5 = 0

Paskal tilida ham boshqa algoritmik tillar kabi arifmetik standart funksiyalari mavjud. Bu funksiyalarni matematik yozilishi va Paskal tilida ifodalanishi quyidagi jadvalda keltirilgan:

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

Arifmetik ifodaga doir misollar :

2*5 - 4*3,
9 div 4/2,
45/5/3,
a + b/2*7.2 - sqrt(7),

exp(2 - a)*9.7 - 6.1*6.1

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

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

$lny=lna^n$, logarifm xossasiga ko'ra

$lny=nlna$, bu tenglikdan "u" ni aniqlaymiz,

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

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

Matematik yozuvi	**Paskal tilidagi yozuvi**
$\frac{A+b}{C+d}$	(a+b)/(c+d)
$\frac{a\cdot(a+b)}{bc}$	A*(a+b)/(b*c)
$\frac{1}{1-\frac{1}{1-\frac{1}{x}}}$	1/(1-1/(1-1/x))
$2-(x-b)^2-e^{ax}+\sin CX$	2-sqr(x-b)-exp(a*x)+sin(c*x)
$x\cdot\frac{e^{x^2+y^2}-1}{\sqrt{\lvert x^2+y^2\rvert}}$	x*(exp(x*x+y*y)-1)/sqrt(abs(x*x+y*y))

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

```
x:=0;
c:= sqrt(a*a+b*b);
y:= 2*pi*r; i:= i+1; i:=5/4; x:= a – b/2;
```

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

To'g'ri tuzilgan dastur

```
Program Misol;
Var
   a,x,y:Real;
Begin
   a:=2.3;
```

Noto'g'ri tuzilgan dastur

```
Program Misol;
Var
   a,x,y:Real;
Begin
   a:=2.3;
```

x:=3.1;
y:=a*x;
Writeln('y=',y);
End.

y:=a*x;
{o'zlashtirish operatorining o'ng tomonidagi "X" o'zgaruvchining qiymati aniqlanmagan}
Writeln('y=',y);
End.

4. Mantiqiy o'zlashtirish operatori

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

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

Mantiqiy ifodada amallarning bajarilish tartibi quyidagicha:

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

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

Mantiqiy ifodaga doir misollar:

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

Mantiqiy o'zlashtirish operatoriga doir misollar:

d:qtrue;
b:= (x>y) and (k=0);
c:=D or B and true;

VAR ***Global Flag:Boolean;***

FUNCTION GETSQR(x:real);

```
Const  SQRMAX=100;
Begin
    X:=x*x;
    GlobalFlag:=( x>SQRMAX );
    If  GlobalFlag  then  x:=SQRMAX;
    GetS+R:=x;
End;
```

5. Belgili o'zlashtirish operatori

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

Belgili o'zlashtirish operatoriga misollar:

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

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

Natija:

oliy ta`lim

Nazariy savollar va tayanch iboralar:

1. Programma qanday ob`ektlar to'plami?
2. O'zlashtirish operatorining metaformulasi qanday yoziladi?
3. O'zlashtirish operatori necha xil bo'ladi?

4. Arifmetik ifoda deb nimaga aytiladi?
5. Arifmetik-standart funksiyalarni sanab bering.
6. Darajaga ko'tarish amali qanday bajariladi?
7. Mantiqiy ifoda deb nimaga aytiladi?
8. Mantiqiy ifoda amallarining bajarilish tartibi qanday bo'ladi?
9. Mantiqiy o'zlashtirish operatori deb nimaga aytiladi?
10. Belgili o'zlashtirish operatoridan qachon foydalanish mumkin?
11. Belgili ifodalar ustida qanday amallar bajarish mumkin?

7-ma`ruza: Tashkiliy, o'tish va shartli operatorlar.

Reja:

1. *Tashkiliy operator va uning fazifasi;*
2. *O'tish operatori;*
3. *Chala va to'liq shartli operatorlar.*

1. Tashkiliy operator va uning fazifasi

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

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

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

Tashkiliy operatorga doir misollar :

1. Begin k:= 5 end

*2. Begin y:=x/7*exp(x+5); z:= ln(abs(y)) end*

3. Begin
```
    k:=0;
    begin
      i:=0;
        z:=i*(i+k);
    end;
    k:=2*k;
end
```

*4. if x>0 then begin a:=5; c:=a*Sin(a) end*

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

2. O’tish operatori

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

<o’tish operatori>::= ***goto*** <metka>

bu yerda *goto* - … ga o’tmoq. Bu operator yordamida boshqarish ko’rsatilgan metkali operatorga uzatiladi. Yuqorida aytganimizdek, programmada qatnashgan barcha metkalar, programmaning metkalar bo’limida e`lon qilinishi kerak.

O’tish operatoriga doir misollar:

1) *a:= 5.75;*
b:= sqr(a); goto L5;
c:= 9.76;
L5: d:= a+b;

2) L: a:=5; goto L;

*3) 1: x:=0; d:=x*x; goto 1; y:=x;*

1- programmada *S:=9.76* operatoridan boshqa barcha operatorlar bajariladi;
2- programma *a:=5* qiymatni tinimsiz hisoblaydi;
3- programma ham *x:=0* va *d:=x*x* ifodani qayta-qayta hisoblab, *y:=x* ifodani hisoblashga navbat kelmaydi.

Umuman olganda, programma tuzuvchi iloji boricha o’tish operatorida foydalanmaslikka harakat qilgani ma`quldir. Chunki o’tish operatoridan foydalanish, programmaning o’qilishini qiyinlashtirib,uning sifatini keskin pasaytiradi.

O’tish operatoridan foydalanishga doir quyidagi to’liq programmani ko’rib chiqaylik:

```
Program m1;
Label 1;
Var  a,y:real;
Begin
      1: Readln (a);
          If  a<=0  then  goto  1;{a ning qiymati a>0 shartini
                                   qanoatlantirmaguncha  qayta-dan
                                   kiritilmoqda}
     y:=ln(a)
     Writeln(‘y= ’,y);
```

end.

3. Shartli operator

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

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

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

Tarmoqlanuvchi jarayonni hisoblash yo'li ma`lum bir shartni bajarilishi yoki bajarilmasligiga qarab tanlanadi. Tarmoqlanuvchi jarayonlarni hisoblash uchun shartli operatordan foydalaniladi. Shartli operatori ikki xil ko'rinishda bo'ladi:

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

To'liq shartli operator quyidagi formada yoziladi:

<to'liq shartli operator>::= *if* <mantiqiy ifoda>
then <operator> *else* <operator>

bu yerda *if* (agar), *then* (u holda), *else* (aks holda) - xizmatchi so'zlar.

Shunday qilib, to'liq shartli operatorni soddaroq quyidagicha yozish mumkin:

if S ***then*** S1 ***else*** S2;

bu yerda S – mantiqiy ifoda;
S1 – S mantiqiy ifoda rost qiymat qabul qilganda ishlovchi operator;
S2 – S mantiqiy ifoda yolgon qiymat qabul qilganda ishlovchi operator.

Shartli operatorning bajarilishi unda yozilgan S1 yoki S2 operatorlaridan faqat birini bajarilishiga olib keladi, ya`ni agar S mantiqiy ifoda bajarilishidan so'ng ***true*** (rost) qiymati hosil bo'lsa S1 operatori, aks holda esa S2 operatori bajariladi.

To'liq shartli operatorga doir misollar:

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

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

if S ***then*** S1;

bu yerda S - mantiqiy ifoda, S1 - operator.

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

Yuqorida aniqlangan shartli operatorlardan bir xil maqsadda bemalol foydalanish mumkin.

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

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

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

"y" tarmoq funksiyasini hisoblash programmasini tuzish talab etilsin.

1. To'liq shartli operatordan foydalanib tuzilgan programma:

```
program misol1;
var  x, y, a, b, c, d: real;
begin
    readln (x); { x nig qiymatini klaviaturadan kiritish
                  so'ralmoqda}
    a:=1.5; b:=4; c:=3.7; d:=-4.2;
    if x>0 then y:= a*x+b else y:=c*x+d;
   writeln (y);
end.
```

2. Chala shartli operatordan foydalanib tuzilgan programma:

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

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

Misol:

if *a>b* **then**

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

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

***if** B1 **then if** B2 **then** S1 **else** S2*

bu operator quyidagicha tushunilishi mumkin:

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

Nazariy savollar va tayanch iboralar:

1. Tashkiliy operator nima uchun lozim bo'ladi?
2. Shartsiz o'tish operatorining metaformulasini yozing.
3. Necha xil shartli operator mavjud?
4. To'liq shartli operatorning metaformulasini yozing.
5. Shartli aperatorning ishlash printspini tushuntiring.
6. Chala shartli aperatorning ishlash printspini tushuntiring.
7. Shartli operatorning rekursivlik xossasini tushuntiring.

8-ma`ruza. Takrorlovchi va bo'sh operatorlar.

Reja:

1. *Parametrli takrorlash operatori;*
2. *repeat takrorlash operatori;*

3. *while takrorlash operatori;*
4. *Bo'sh operator.*

1. Parametrli takrorlash operatori

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

Takrorlash operatorlarining 3 xil turi mavjud:

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

Echilayotgan masalaning mohiyatiga qarab, programma yozuvchi o'zi uchun qulay bo'lgan takrorlash operatorini tanlab olishi mumkin.

Operatorning quyidagi ko'rinishdagi holi amalda ko'proq ishlatiladi:

for k:= k1 ***to*** k2 ***do*** S;

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

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

Operatorning ishlash prinsipi:

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

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

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

```
Program  sum1;
var
        S: real;
       i,n: byte; {i va n o'zgaruvchilar 255 dan katta bo'lmagan,
                   butun, natural sonlar}
begin
        readln (n);  S:= 0;
        for  i:q1  to   n  do
                S:=  S+ 1/i;
        writeln (S);
end.
```

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

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

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

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

Misol: yuqorida ko'rsatilgan misolni programmasini qaytadan tuzaylik.

Bu holda programmadagi sikl operatorigina o'zgaradi xolos:

for i:= n ***downto*** 1 ***do***

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

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

To'g'ri tuzilgan programma qismi

```
for i:=1 to 10 do
        Begin
                s:=i*i;
                writeln(s);
        end;
```

Noto'g'ri tuzilgan programma qismi

```
for i:=1 to 10 do
        Begin
                s:=i*i;
                writeln(s);
                i:=i+3
        end;
```

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

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

Funktsiyani necha marta hisoblash kerakligini $N=\frac{2-(-2)}{0,5}=\frac{4}{\frac{1}{2}}=8$

formula bilan aniqlaymiz.

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

2. Repeat takrorlash operatori

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

Repeat operatorining yozilish formasi quyidagicha bo'ladi:

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

bu yerda ***repeat*** (takrorlamoq), ***until*** (gacha) - xizmatchi so'zlar;
S1, S2, ..., SN lar esa sikl tanasini tashkil etuvchi operatorlar;
B - sikldan chiqish sharti (mantiqiy ifoda).

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

```
Program  Sum2;
var      i, n: Byte;
  S: real;
begin
    readln(n);
    S: =0;  i:=1;
    repeat
      S:q S+1/i;
      i:=i+1;
   until  i>n;
   writeln (S)
        end.
```

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

E`tiboringizga ya`na bir, ismni qidirib topish programmasini xavola qilamiz:

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

3. While takrorlash operatori

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

while B ***do*** S;

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

B - sikldan chiqishni ifodalovchi mantiqiy ifoda;

S - siklning tanasini tashkil etuvchi operator.

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

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

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

4. Bo'sh operator

Bu operator o'zidan keyingi operatorni aniqlab beradi xolos. Operatorlar ketma-ketligi orasida boshqa operatorlardan ";" belgisi bilan ajratilib turiladi. Bundan tashqari, bo'sh operator metka bilan jixozlangan ham bo'lishi mumkin.

Misol:

1. **begin** L1:; k:=5; M:=k+6; **end**.
2. **begin** M:=5; k:=M-2.7; L4: **end**.

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

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

Nazariy savollar va tayanch iboralar:

1. Takrorlash operatorlarining necha xili mavjud?

2. Parametrli takrorlash operatorining ishlash prinsipini tushuntiring;
3. Kamayish tartibida ishlovchi parametrli takrorlash operatorini yozing va uni ishlash prinsipini tushuntiring;
4. Chekli yig'indini hisoblash dasturni tuzing;
5. Qanday xollarda Repeat va While operatorlaridan foydalanish maqsadga muvofiq?
6. Repeat operatorining ishlash prinsipini tushuntiring;
7. While operatorining ishlash prinsipini tushuntiring;
8. Chekli va cheksiz yig'indini Repeat va While operatorlari yordamida hisoblash dasturini tuzing;
9. Bo'sh operatorning vazifasini tushuntiring.

9-ma`ruza: Qiymatlarning skalyar tiplari.

Reja:

1. *Sanalma tiplar;*
2. *Variant tanlash operatori;*
3. *Cheklangan tiplar.*

1. Sanalma tiplar

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

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

if k = 7 ***then***

programma qatorini o'qib, gapni nima haqida ketayotganiligini dabdurustdan anglash qiyin. Ehtimol, gap bu yerda 7 - oy haqidadir, balki "k" o'zgaruvchini 7 butun soni bilan tekshirilayotgandir. Shunday qilib, "7" soni ostida nima yashiringanini bilish juda qiyin. Lekin, programmaning bu qatori

if k = iyul ***then***

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

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

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

Sanalma qiymat tipini quyidagicha aniqlanadi:

<sanalma tipi>::=(<ism>,<ism>,...) yoki<sanalma tipi>::=(<ism> {,<ism>})

bu yerda kichik qavs ichidagi o'zaro vergul bilan ajratilgan <ism>lar aniqlangan tipning o'zgarmaslari hisoblanadi, ularning qavs ichiga olib yozilgan birikmasi esa, shu tipning qiymatlar to'plami hisoblanadi. Sanalma tip qiymatlari qat`iy noldan boshlab nomerlangan.

Masalan, (dushanba, seshanba, chorshanba, payshanba, juma, shanba, yakshanba) sanalma tipi 7 ta haddan iborat bo'lib, bu yerda quyidagi hol o'rinlidir:
dushanba < seshanba < chorshanba < payshanba < juma < shanba < yakshanba ya`ni dushanba 0 - tartib raqamiga , seshanba 1-tartib raqamiga ega va h.k.

Bu tip programmaning yangi tiplar bo'limida aniqlanadi.

Sanalma tipni aniqlashga doir misollar:

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

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

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

*Zirovor = (**zira, qalampir, olma**)*

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

Programmaning ***type*** bo'limida aniqlab qo'yilgan tiplardan o'zgaruvchilarni tiplarini e`lon qilish bo'limida xuddi standart tiplar kabi foydalanilsa bo'ladi:

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

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

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

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

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

Faraz qilaylik, xq sesh qiymatli bo'lsin.

succ (x) = chor (x dan keyingi qiymat**),**
pred (x) = dush (x dan oldingi qiymat**),**
org (x) = 1 (x qiymatning tartib raqami**),**

for x = ***sesh*** *to* ***yaksh*** *do S;*

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

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

2. Variant tanlash operatori

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

Variant tanlash operatorini sintaksis aniqlanmasi quyidagicha:

<variant tanlash operatori>::= ***case*** <operator selektori> ***of***
<variant ro'yxatining hadlari>
end;

bu yerda <operator selektori>::= <ifoda>;
<variant ro'yxatining hadi>::q<variantlar metkalarining ro'yxati>:<operator>;
<variantlar metkalarining ro'yxati>::=<variant metkasi>{,< variant metkasi>};
<variant metkasi>::=<o'zgarmas>.

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

Misollar:

1. Case ***i mod 3*** *of*

```
            0:  m: =  0;
            1:  m: = -1;
            2:  m: =  1
    end.
```

```
2. Case  sum  of
                  '=' :  k:=  1;
   '*',  '+',  '/',  '-' :  ;
                 '!' :  k:=  2;
            ':',  ';' :  k: = 3
   end.
```

```
3. Case  kun  of
            dush, sesh, chor, pay, jum:  writeln('ish kuni');
            shan, yaksh: writeln('dam olish kuni')
    end.
```

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

Endi shartli operatorni, variant tanlash operatori orqali ifodasini ko'rib chiqaylik:

Quyidagi shartli va variant tanlash operatorlari bir-biriga mos keladi

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

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

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

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

```
type
        haftaq(Du,Se,Cho,Pa,Jy,Sha,Jak);
  var
        kun:hafta;
```

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

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

3. Cheklangan tiplar

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

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

Cheklangan tiplar quyidagicha aniqlanadi:

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

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

Cheklangan tiplarga doir misollar:

1. 1..20 (integer tipidagi cheklanma);
2. dush..jum (sanalma tipli cheklanma);
3. 'A'..'Z' (char tipidagi cheklanma).

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

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

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

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

Nazariy savollar va tayanch iboralar:

1. Standart skalyar va yangi tiplar farqini tushuntiring;
2. Sanalma tipi qanday aniqlanadi;
3. Sanalma o'zgaruvchilarni e`lon qilish yo'lari;
4. Sanalma tipli argument uchun standart funksiyalar;
5. Variant tanlash operatori nima uchun kerak?
6. Variant tanlash operatorininng sintaksis aniqlanmasi qanday bo'ladi?
7. Selektor qiymati nima uchun hisoblanadi?
8. Variant tanlash operatorining tana qismiga kirish qanday amalga oshiriladi?
9. Shartli operator bilan variant tanlash operatorlarining qanday o'xshashligi bor?
10. Cheklangan tiplar qanday aniqlanadi?

10-ma`ruza: Kombinatsiyali (yozuvlar) va to'plamli tiplar.

Reja:

1. *Yozuvlar haqida umumiy ma`lumotlar;*
2. *Oddiy kombinatsiyali tiplar (yozuvlar);*
3. *Ierarxik yozuvlar;*
4. *Bog'lama operatori;*
5. *Paskal tilida to'plamlarni belgilash;*
6. *To'plamlar ustida amallar;*
7. *To'plamli tiplarni o'rnatish va to'plamli o'zgaruvchilar.*

1. Yozuvlar haqida umumiy ma`lumotlar

E`tiboringizga yana bir yangi, boshqa algoritmik tillarda mavjud bo'lmagan, Paskal tilining hosilaviy tiplaridan birini - kombinatsiyali tipni havola qilamiz. Kombinatsiyali tipning qiymati ham xuddi massivlarniki kabi (massivlar haqida boshlang'ich ma`lumotlarga egasiz deb o'ylaymiz) bir nechta haddan tashkil topadi, lekin massivdan farqli o'laroq, uning har bir hadi turlicha tipli bo'lishi mumkin. Bu tipning hadlariga, ularning joylashgan tartib raqamlari bilan emas, balki ismlari orqali murojaat qilinadi.

Kombinatsiyali tipni odatda soddagina qilib - yozuvlar deb ataladi.

Kombinatsiyali tip qiymatlari asosan murakkab, bir jinssiz tashkil etuvchilarga ega bo'lgan ob`ektlarni ifodalashga bag'ishlangan. Bu tipdan amalda turli xil ma`lumotlar jamg'armasini yaratishda keng foydalaniladi. Masalan, biror tashkilotda ishlovchi xodimlar haqidagi ma`lumotlar jamg'armasi - xodimlarning turli xil anketali habarlarini o'zida jamlaydi:

familiyasi,
ismi-sharifi,
tug'ilgan yili-oyi-kuni,
uy adresi ishchi va uy telefon raqamlari,
ma`lumoti, mutaxassisligi,
oilaviy ahvoli,
harbiyga aloqadorligi va h.k.

Sanab o'tilgan va bitta shaxsga tegishli bo'lgan ushbu ma`lumotlarning turli xil tipga tegishligiga ahamiyat bering:

telefonlar,tug'ilgan yili-oyi-kunlari – butun sonlardan, boshqa ma`lumotlar esa belgili qatorlardan tashkil topgan.

Shunday qilib, kombinatsiyali tip qiymati – maydonlar deb atalmish chekli sondagi hadlardan tashkil topgan ma`lumotlar strukturasidir. Yozuvning har bir maydoniga o'ziga xos ism beriladi va bu maydon qiymatining tipi ko'rsatiladi. Bunda, maydon qiymatining tipiga hech qanday cheklashlar qo'yilmaydi. Shuning uchun, yozuv hadlari o'z navbatida yana yozuv bo'lishi mumkin. Demak, yozuv aniq

ifodalangan ierarxik strukturaga ega bo'lishi mumkin. Bir xil darajada turgan, bitta yozuvning barcha ismlari turli xil bo'lishi lozim, xuddi shuningdek, turli yozuvlar bir xil ismli maydonlarni o'z ichiga olishi mumkin. Bunda bu maydonlarga murojaat qilishda hech qanday anglashilmovchilik bo'lmaydi, chunki murojaat tashqi yozuv orqali amalga oshiriladi.

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

2. Oddiy kombinatsiyali tiplar

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

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

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

<maydonlar ro'yxati>::=<yozuv sektsiyasi>{;<yozuv sektsiyasi>}
<yozuv sektsiyasi>::=<maydon ismi>{,<maydon ismi>}:<tip>

Shu qoidaga asoslanib, matematika fanidan yaxshi tanish bo'lgan kompleks sonli tip kiritaylik (a + bi -kompleks son, a, b - haqiqiy sonlar, $i^2=-1$) va tip nomini ***complex*** deb ataylik:

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

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

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

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

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

```
var
```

x, y, z: ***CompLex;***

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

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

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

y := x;

Yozuvlarga doir quyidagi misol ustida, ular bilan ishlash malakamizni oshiraylik:

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

Masalani yechish algoritmi:

Agar *x= Re x + i Im x, y= Re y+ i Im y* bo'lsa, ular ustida sanab o'tilgan amallarni bajarish algoritmi quyidagicha bo'ladi:

u = x + y, Re u = Re x + Re y, Im u = Im x+Im y;
v = x - y , Re v = Re x - Re y, Im u= Im x - Im y;
*w = x * y, Re w = Re x * Re y - Im x * Im y,*
*Im w = Re y * Im x + Re x * Im y*

Endi mazkur algoritmni programmada ifoda etamiz:

```
Program L1;
type
        comp = record
                re, im: real
        end;
var
        x, y, u, v, w: comp;
begin {x va u mavhum sonlarning haqiqiy (Re x, Re y) va
        mavhum (Im x, Im y) qismlarini kiritish}
        readln (x.re, x.im, y.re, y.im);
                { u=x+y}
        u.re:= x.re + y.re;  u.im := x.im + y.im;
                {v=x-y}
        v.re :=x.re - y.re;   v.im := x.im - y.im;
                {w=x*y}
        w.re:= x.re * y.re - x.im * y.im;
```

```
        w.im := x.re * y.im+ x.im * y.re;
        writeln ( 'x + y =',u.re,'+',u.im,'*i');
        writeln ( 'x - y = ',v.re,'+',v.im,'*i');
        writeln ( 'x * y =',w.re,'+',w.im,'*i');
    end.
```

Quyida yana bir misolni ko'rishimiz mumkin, bu misol yozuvlar ustida amal bajarishni ba`zi bir operatorlarini namoyish qiladi:

```
  program RecordExample;
type
  check =record
     number: integer;
     amount: real;
     date: array [1..8] of char;  { '99/99/99' }
     payee: string[40];
  end;
var
   MyCheck : check;
   YourCheck: check;
begin
   MyCheck.number := 753;    { Yangi yil sovgalari }
   MyCheck.amount := 87000000.00;
   MyCheck.date := '12/25/88';
   MyCheck.payee := ' O'zing uchun ';
  YourCheck := MyCheck; { xamma maydon avtomatik tarzda  kabul kilinadi }
  YourCheck.number := YourCheck.number + 1;
  with YourCheck do
   begin
     Writeln('Naqtiga: ',amount);
     Writeln('Kun:        ',date);
   end;
  end.
```

3. Ierarxik yozuvlar

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

Yozuv maydonining tipi ikki xil usul bilan aniqlanishi mumkin:

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

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

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

```
type rab = packed array [1..15] of  char;
        data = record
          kun : 1..31;
          oy : (yanv, fev, mart, apr,may, iyun, iyul, avg, setnt, oct, noy, dek);
          yil: integer;
         end;
Ilmiy_xodim q record
        fish : record  {fish – familiyasi, ismi, sharifi}
                    fam, ism, shar: rab
              end;
        tug : data;
        jins: (erkak, ajl);
        malum: (boch, orta, ortmax, olij);
        maosh: integer;
        ildar: (darajasiz, fan_nom, doktor); {ildar – ilmiy darajasi}
        ilunvon: (unvonsiz, dos,  kat_ul_xodim, prof); {Ilunvon – ilmiy unvoni}
        uyadr: record
                    ind: integer;
                    shah, kucha: rab;
                    uy, kvar: integer;
              end;
        tel: record
              uy, hizmat: integer
            end;
end;
```

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

```
var
    shaxs: Ilmiy_xodim;
```

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

```
shaxs.fish.ism:= 'Ahmad';
shaxs.fish.fam:='Po'latov';
shaxs.fish.Shar:='Anvarovich';
shaxs.tug.kun:='15';
shaxs.tug.oy:=maj;
shaxs.tug.yil:='1953';
shaxs.jins:=erkak;
shaxs.malum:=olij;
```

```
shaxs.maosh:='3000';
shaxs.ildar:=fan_nom;
shaxs.ilunvon:=dos;
shaxs.uyadr.ind:='717325';
shaxs.uyadr.shah:='Namangan';
shaxs.uyadr.kucha:='Navoiy';
shaxs.uyadr.uy:='20';
shaxs.uyadr.kv:='5';
shaxs.tel.uy:='45215';
shaxs.tel.hiz:='40625';
```

4. Bog'lama operatori

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

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

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

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

Endi bu operatorlarni amalda ishlatilishini ko'rib chiqamiz:

with R ***do*** S

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

R - kombinatsiyali tipga tegishli o'zgaruvchi;

S - ixtiyoriy operator.

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

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

```
fish.ism :='Axmad';
```

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

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

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

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

U holda bog’lama operatorining kiritilishi:

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

quyidagi tashkiliy operatorga teng kuchlidir:

```
begin
      R2.A:=1;  R2.B:=2; R2.C:=3; R3.D:=4;  R3.B.E:=5
End.
```

5. Paskal tilida to’plamlarni belgilash

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

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

Paskal tilida yozilgan to’plamlarga misollar:

1. [ ] - bo’sh to’plam;
2. [2,3, 5, 7, 11] - 2, 3, 5, 7, 11 butun sonlaridan tuzilgan to’plam;
3. [1..10] - 1 dan 10 gacha bo’lgan butun sonlar

to’plami;

4. [‘a’, ‘b’, ‘d’] - a, b, d harf (belgi)lardan tuzilgan to’plam;
5. [oq,qora, sariq] - 3 ta hadli, sanalma tipdagi qiymatlardan hosil qilingan to’plam;
6. [‘d’..’a’] - bo’sh to’plam;
7. [‘a’..’d’,’f’..’h’, ‘k’] - a, b, c, d, f, g, h, k xarflaridan tashkil topgan, belgili tipli to’plam;
8. [1..1,5..1] - bir qiymatli, bitta hadli, sonli to’plam, ya`ni [1] ga teng kuchli.

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

1) [true, false] va [false, true] to’plamlari ekvivalentdir;
2) [1,2,3,2..5,6,4,3] to’plami [1,2,3,4,5,6] to’plamiga ekvivalent, u esa o’z navbatida [1..6] to’plamiga ekvivalent.

Noto’g’ri yozilgan to’plamlardan namunalar:

1) [5.3, 2.5] – to’plam hadlari Real tipidagi son bo’lishi mumkin emas;
2) [9, 7, “P”,0] – to’plam hadlari bir xil tipli bo’lishi lozim edi;
3) [‘abc’,’Axad’] – to’plam hadlari hosilaviy tiplarga tegishli bo’lishi mumkin emas.

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

Paskaldagi yozilishi	Matematik yozilishi	Natijaviy qiymat	
		True	False
A=B	A=B	A va B to’plamlari mos tushadi	aks holda
A.<>B	A≠B	A va B to’plamlari mos tushmaydi	aks holda
A<=B	A⊆B	A to’plamining barcha hadlari B to’plamga tegishli	aks holda
A>=B	A⊇B	B to’plamining barcha hadlari A to’plamga tegishli	aks holda
x in A	X∈A	x hadi A to’plamga tegishli	aks holda

6. To’plamlar ustida amallar

1. Ikkita A va B to'plamlari teng (bir xil) deyiladi, agarda ularning barcha hadlari o'zaro bir xil bo'lsa (A=B).

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

2. A to'plam B to'plamining to'plam ostisi deyiladi, agarda A ning barcha hadlari B ning ham hadlari bo'lsa (A<B).

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

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

Misol: [1,2,3,4,5] * [2,5,6,7,8] =[2,5]. Agar A va B to'plamlari bir xil hadlarga ega bo'lmasa, S = A*B natija – bo'sh to'plam bo'ladi S= [ ].

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

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

5. A to'plamdan B to'plamni ayirmasi deb, shunday to'plamga aytiladiki, natijaviy to'plamning hadlari A to'plamga tegishli lekin B to'plamga tegishli bo'lmaydi (S= A - B). Misol: [1,2,3,4,5] - [2,5,6,7,8] = [1,3,4].

6. x qiymatni A to'plamiga tegishliligini aniqlash uchun «***in***» amalidan foydalaniladi. x ***in*** A = ***true*** , agar x qiymat A to'plamiga tegishli bo'lsa; x in A q=***false*** , agar x qiymat A to'plamiga tegishli bo'lmasa.

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

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

To'plamli tiplarni aniqlash quyidagicha amalga oshiriladi:

Set of <to'plam tipi>;

bu yerda ***Set*** va ***of*** - Paskal tilining xizmatchi so'zlari.

Misol:

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

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

1. ***Set of boolean***;

bunday aniqlangan to'plamning qiymatlari to'plami:

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

3. type

oy = (yanv, fev, ..., dek);

```
        yiloyi = Set of oy
        bool = Set of boolean;
var
        bolalar: Set of (ugil, kiz);
        oylar: yiloyi;
        mantiq: bool;
```

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

B:=S;

bu yerda B - to’plam tipiga tegishli o’zgaruvchi;
S - to’plamli ifoda.

To’plamli ifodaga misollar:

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

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

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

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

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

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

Nazariy savollar va tayanch iboralar:

1. Kombinatsiyali tip (yozuv) deb nimaga aytiladi?
2. Yozuvlarning qiymatlari nimadan iborat?
3. Oddiy yozuvlar qanday aniqlanadi?
4. Kompleks sonni aniqlash yozuvini yarating;
5. Kompleks sonlar ustida yozuvlardan foydalanib amallar bajaring;
6. Ierarxik yozuvlarni oddiy yozuvlardan farqi nima?
7. O'zingiz xaqingizdagi ma`lumotlarni ierarxik yozuv ko'rinishida rasmiylashtiring.

8. To’plam deb nimaga aytiladi?
9. To’plam hadlari soni qancha bo’lishi mumkin;
10. To’plam qanday tipli qiymatlar qabul qilishi mumkin?
11. To’plamlar ustida qanday munosabat amallarni bajarish mumkin?
12. To’plamlar ustida qanday amallarni bajarish mumkin?
13. To’plam tiplari qanday aniqlanadi?
14. To’plamli ifodalardan misollar keltiring.

11-ma`ruza: Massivlar (jadval kattaliklar).

Reja:

1. Massivlar haqida umumiy ma`lumotlar;
2. Bir o’lchamli massivlar va ular ustida amallar;
3. Ko’p o’lchamli massivlar va ular ustida amallar.

1. Massivlar haqida umumiy ma`lumotlar

Biz shu paytgacha qiymatlarning oddiy (skalyar) tiplaridan foydalanib, turli xil programmalar yaratishni o’rgandik. Skalyar tipga tegishli har bir qiymat yagona ma`lumot hisoblanib, trivial strukturaga egadir.

Amalda esa, turli xil hosilaviy tiplar bilan ishlashga, ulardan foydalanib murakkab programmalar yaratishga to’g’ri keladi. Bu tiplarga tegishli qiymatlarning har biri trivial bo’lmagan strukturaga ega, ya`ni bu qiymatlar o’z navbatida yana bir nechta qiymatlardan tashkil topadi.

Endi shunday tiplardan biri bo’lgan, programmalashda eng ko’p qo’llaniladigan programma ob`ekti – massivlar bilan tanishib chiqamiz.

2. Bir o’lchamli massivlar va ular ustida amallar

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

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

<massiv nomi> [<indeks>]

bu yerda <indeks> - massiv hadining joylashgan o’rnini anglatuvchi tartib qiymati.

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

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

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

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

array [1..100] of real;

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

1. array [1000..5000] of integer;
2. array [-754..-1] of byte;
3. array [0..100] of real;
4. array [0..10] of boolean;
5. array [10..25] of char;

6.
```
type
    chegara   =1..100;
    vektor    = array [chegara] of real;
    massiv1 =array [115..130] of integer;
    massiv2 = array [-754..-1] of integer;
var
    A,B: vektor;
    c,d : massiv1;
    e:    massiv2;
```

7.
```
var
    r,  t:  array [chegara] of real;
    s, q:  array [115..130] of integer;
    p:     array [-754..-1] of integer;
    k, m: array [1..50] of (shar, kub, doira);
```

8.
```
type kv1 = (yanvar, fevral, mart);
var t, r: array [kv1] of real;
```

9.
```
type
    belgi = array [boolean] of integer;
    belgi_kodi = array [char] of integer;
```

```
var
    k : belgi;
    p : belgi_kodi;
```

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

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

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

2. Bir o'lchamli, n ta hadli (n_q30) massiv hadlarining eng kattasini topish va uning joylashgan joyini aniqlash.

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

```
        writeln ('max(x) ning o'rni', k)
end.
```

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

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

3. Ko'p o'lchamli massivlar va ular ustida amallar

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

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

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

Ikki o'lchamli massivlarning tiplarini bir necha xil yo'lda aniqlashni quyidagi misol ustida ko'rib chiqaylik (A matritsa 10 ta satr va 20 ta ustundan iborat bo'lib, uning xadlari haqiyqiy tipga tegishli bo'lsin):

1. var

```
    A: array [1..10]  of  array [1..20]  of  real;
```

```
2. type  matr = array [1..10]  of  array [1..20]  of  real;
  var
    A: matr;

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

4. var  A: array [1..10, 1..20]  of  real;
```

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

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

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

1. Matritsalarni qo'shish.

```
Program  L2;
const  n = 3;  m = 4;
       { n -  matritsa satrlari soni,
         m -  ustunlar soni}
var    i, j: integer;
       A, B, C: array [1..n,1..m] of  real;

begin {A, B matritsa hadlarini kiritish}
       for i : = 1  to n  do
        for j: =1  to  m  do
          readln (A[i,j], B[i,j]);
       for i : = 1  to  n  do
       for  j: = 1  to m  do
         begin
           C[i,j] := A[i,j] + B[i,j];
           writeln (C[i,j])
       end
end.
```

2. Matritsani vektorga ko'paytirish.

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

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

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

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

```
    end;
 writeln ('max A=', R);
 writeln ('satr=',L,'ustun =',K);
end.
```

Nazariy savollar va tayanch iboralar:

1. Massiv deb nimaga aytiladi?
2. Massiv hadiga qanday murojaat qilinadi?
3. Bir o'lchovli massivlar qanday e`lon qilinadi?
4. Bir o'lchovli massivning eng katta va eng kichik hadlarini topish dasturini tuzing;
5. Vektorning skalyar ko'paytmasini hisoblash dasturini tuzing?
6. Ko'p o'lchamli massivlarga ta`rif bering;
7. Ko'p o'lchamli massivlarni e`lon qilish usullarini ko'rsating;
8. Turli o'lchamli massivlarning hadlarini kiritish va chop etish qanday amalga oshiriladi?
9. Matritsalarni qo'shish dasturini tuzing;
10. Matritsani vektorga ko'paytirish dasturini tuzing;
11. Matritsani matritsaga ko'paytirish dasturini tuzing;
12. Teskari matritsani hisoblash dasturini tuzing.

12-ma`ruza: Prosedura - operatorlar

Reja:

1. *Prosedura va funksiya haqida umumiy ma`lumotlar;*
2. *Parametrsiz proseduralar;*
3. *Parametrli proseduralar*

1. Prosedura va funksiya haqida umumiy ma`lumotlar

Programma tuzish jarayonida, uning turli joylarida ma`nosiga ko'ra bir xil, mustaqil xarakterga ega bo'lgan va yechilayotgan asosiy masalaning biror qismini hal qilishni o'z bo'yniga olgan murakkab algoritmdan bir necha marotaba foydalanishga to'g'ri keladi. Masalan, matritsalarni ko'paytirish, matritsani vektorga ko'paytirish, chiziqsiz tenglamani yechish, chiziqli algebraik tenglamalar sistemasini yechish, faktorial hisoblash, yig'indi hisoblash va hokazo kabi masalalarni hal qilish algoritlari juda ham ko'p masalalarni yechishning bosh algoritmlarida qayta-qayta, turli boshlang'ich ma`lumotlar bilan qatnashishi mumkin. Bunday hollarda, malakali dasturchi programma matnini ixchamlashtirish, programmaning ishonchlilik darajasini oshirish, programmani taxrirlash (otladka) ni tezlashtirish va programmaning umumiyligini (universalligini) ta`minlash uchun prosedura va funksiyalardan kengroq foydalanib, mukammal programma yaratishga harakat qiladi.

Prosedura va funksiyalar mustaqil programmali ob`ektlar hisoblanadi. Bu mustaqil programmali ob`ektni dasturchi o'z hoxishiga va undan olinadigan natijalariga

ko’ra prosedura yoki funksiya ko’rinishida aniqlashi mumkin. Odatda olinadigan natija yagona qiymatli bo’lsa funksiyadan, olinadigan natijalar soni bir nechta bo’lsa proseduradan foydalanish maqsadga muvofiqdir.

Prosedurani yozish strukturasi xuddi asosiy programma strukturasi kabi bo’lib, faqat sarlavhalari bilangina farq qiladi xolos:

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

Proseduralar va funksiyalarni aniqlash asosiy programmaning ***var*** (o’zgaruvchilarning tiplarini e`lon qilish) bo’limida bajariladi. Proseduradan programmada foydalanish uchun uning ismi va faktik parametrlar ro’yxati yoziladi. Shunda prosedura o’ziga belgilangan ishni bajarib, o’zining faktik parametrlari orqali asosiy programmaga o’z natijasini beradi.

Proseduraning e`loni va unga murojaat qilishni keyinchalik ko’riladigan misollar orqali o’zlashtirib olamiz.

2. Parametrsiz proseduralar

Yuqorida aytib o’tganimizdek, prosedura hisoblab bergan natijalar uning faktik parametrlari orqali asosiy programmaga uzatiladi. Lekin, ayrim paytlarda prosedura parametrsiz ham bo’lishi mumkin. Bu holda asosiy programmaning barcha parametrlari prosedura parametrlari rolini bajaradi. Parametrsiz prosedurada ham proseduraning barcha bo’limlari saqlanib qoladi, faqat parametrlar ro’yxatigina qatnashmaydi.

Proseduralarni aniqlash va ulardan foydalanishni quyidagi misol ustida ko’rib chiqaylik:

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

bu yerda x, u - qiymatlari kiritiladigan haqiqiy tipli o’zgaruvchilar.

1. Masalani yechish programmasining proseduradan foydalanmay tuzilgan holi:

```
Program  max;
var
        x, y, u, v: real;
        a, b, s: real;
begin
```

```
        {x, u - miqdorlarni kiritish};
        readln (x,y);
        a:= x +y;  b:= x*y;
        if  a > b  then S:= a else S:qb;
        u := S;
        a:= 0.5; b:=u;
        if  a > b  then  S:= a   else  S:=b;
        v:=S;
        {olingan natijalar};
        writeln (u, v)
end.
```

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

2. Masalani yechish programmasini parametrsiz proseduradan foydalanib tuzilgan holi (endi yuqoridagi programmada yo'l qo'yilgan kamchilikni proseduralar orqali tuzatishga harakat qilamiz):

```
Program max;
var x, y, u, v: real; a, b, S: real;
procedure  max1;
begin
        if  a>b  then  S:=a  else  S:qb;
end;

begin
        readln (x, y);
        a:= x + y;  b:=x * y;
        max1; {max1 prosedurasiga 1-marta murojaat    qilinmoqda}
         u:=S;
        a:= 0.5;   b:= u;
        max1;  {max1 prosedurasiga 2-marta murojaat qilinmoqda}
        v:=S;
        writeln (u,v);
end.
```

Asosiy programmaning operatorlar qismida ikki marta yozilgan max1 parametrsiz prosedurasiga murojaat, e`lon qilingan prosedurani ikki marta asosiy programmaga olib kelib ishlatishni tashkil qiladi. Ahamiyat berilsa, ikkinchi programma birinchi prosedurasiz tuzilgan programmaga ko'ra ixchamroq va soddaroqdir. Biz kiritgan prosedura hozircha faqat ikkita haqiqiy son ichidan kattasini aniqlab berdi xolos, shuning uchun programma matning xajmini kamaytirishdan erishgan yutuq salmoqli bo'lmadi. Lekin, proseduralar asosan ko'p xajmli matndagi amallarni, vazifalarni bajarishga mo'ljallanadi va bu holda erishilgan yutuq salmog'i ancha yuqori bo'ladi.

Parametrsiz proseduraning asosiy kamchiligi, uning asosiy programmaga va undagi ma`lum parametrlarga bog'lanib qolganligidir.

3. Parametrli proseduralar

Prosedura bilan asosiy programmani bog'laydigan asosiy faktor bu – prosedura parametrlaridir. Parametrlarni ikkita tipga ajratiladi: qiymatli parametrlar (parametr-qiymat), o'zgaruvchili parametrlar (para-metr - o'zgaruvchi).

Parametr - qiymat bu prosedurani ishlash jarayonini ta`minlovchi parametrlar hisoblanadi, ya`ni asosiy programma qiymatlarini proseduraga uzatadigan parametrlardir.

Endi, yuqorida ko'rib chiqilgan sonlarni eng kattasini topish algoritmining programmacini qiymatli parametr bilan yozilgan proseduralar orqali amalga oshiraylik:

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

bu yerda a, b - proseduraning qiymatli formal parametrlari.

Proseduraga murojaat qilishda formal va faktik parametrlarning tiplari o'zaro mos kelishi kerak, aks holda programma xato tuzilgan hisoblanadi. Yuqoridagi programmadan ko'rinib turibdiki, a va b formal parametrlar o'rniga natijaviy qiymatlari ma`lum ifodalar qo'yildi. Demak, qiymatli faktik parametrlar o'rniga, shu tipli natijaga erishuvchi ifoda yozilishi mumkin. Bundan tashqari, prosedurada kiritilgan a va b parametrlari faqat proseduraning ichidagina ma`noga ega, tashqarida, misol uchun asosiy programmada ular tushunarsiz, qiymatlari aniqlanmagan miqdorlardir. Shuning uchun, qiymatli parametrlarga prosedura natijalarini o'zlashtirib, asosiy programmaga uzatib bo'lmaydi.

Yuqorida tuzilgan programmaning asosiy kamchiligi, topilgan katta son doim S o'zgaruvchisiga o'zlashtirilayapti. Misolimiz shartiga ko'ra, natijalar u va v o'zgaruvchilariga o'zlashtirilishi kerak. Shuning uchun, programmada ikki marta qo'shimcha *u:qS* va *v:qS* o'zlashtirish operatorlari yozildi.

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

Parametr-o'zgaruvchini parametr-qiymatdan farq qilish uchun prosedurani aniqlashdagi parametrlar ro'yxatida o'zgaruvchi oldidan ***var*** xizmatchi so'zi yoziladi. Parametr - o'zgaruvchidan so'ng albatta, uning tipi ko'rsatib qo'yiladi. Yuqorida aytganimizdek, formal parametr - qiymat o'rniga proseduraga murojaat vaqtida shu tipli ifoda yozish mumkin bo'lsa, parametr - o'zgaruvchi uchun bu hol mutlaqo mumkin emas.

Prosedurani mukammallashtirib borish dinamikasini his etish uchun yana, yuqorida ko'rilgan maksimum topish misolining programmasini parametr - o'zgaruvchi ishlatgan holda ko'rib chiqamiz:

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

Shunday qilib, bitta programmani proseduraning uch xil varianti uchun tuzib chiqib, natijada ixcham va sodda programmaga ega bo'ldik.

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

Biz yuqorida eslatib o'tdikki, faktik parametr formal parametr - qiymatga mos tipli ixtiyoriy ifoda bo'lishi mumkin. Lekin, Paskal tilida ixtiyoriy tipli qiymatlar uchun shu tipdagi natija beruvchi hech qanday amal ko'zda tutilmagan. Shuning uchun, bu tiplar uchun faktik parametrlar faqat shu tipga mos o'zgaruvchilar bo'lishi mumkin xolos. Bunday hol, xususiy holda massivlar uchun ham o'rinlidir.

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

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

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

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

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

Bu proseduraga asosiy programmada murojaat

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

ko'rinishida amalga oshiriladi.

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

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

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

Prosedurani aniqlash (e`lon qilish) quyidagicha amalga oshiriladi:

<prosedurani aniqlash>::=<prosedura sarlavhasi>;<blok>

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

Endi esa <prosedura sarlavxasi>ga ta`rif beramiz:

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

Prosedura ismi dasturchi tomonidan tanlanadigan oddiy identifikator hisoblanadi.

Formal parametrlar ro'yxati quyidagicha aniqlanadi:

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

Formal parametrlar sektsiyasi deganda prosedura parametrlarining parametr-qiymat va parametr-o'zgaruvchi lardan iborat bo'lishligi tushuniladi:

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

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

Endi yuqoridagi aniqlashlarga tushuntirishlar berib o'tsak.

Yuqoridagi Bekus-Naur formulalaridan ko'rinib turibdiki, formal parametrlar ro'yxati (agar u mavjud bo'lsa) bitta yoki bir nechta o'zaro nuqta- vergul (;) belgisi bilan ajratilgan sektsiyalardan tashkil topgan. Har bir sektsiyada esa o'z navbatida, bitta yoki bir nechta o'zaro nuqta-vergul bilan ajratilgan formal parametrlar qatnashishi mumkin. Proseduradagi formal parametrlar sonini, dasturchining o'zi prosedurani aniqlash moxiyatidan kelib chiqqan holda tanlaydi.

Misol:

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

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

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

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

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

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

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

bu sarlavha

Procedure P(Var X,Y:Real);

sarlavxasi bilan bir xil emas.

Aniqlangan proseduraga murojaat yoki prosedura operatoridan qanday foydalanishni aniqlashni ko'rib chiqaylik (yaratilgan prosedurani «Aktivlashtirish», ya`ni ishlatish):

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

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

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

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

Nazariy savollar va tayanch iboralar:

1. Prosedura va funksiya programmaning qanday ob`ekti hisoblanadi?
2. Prosedurani yozish strukturasini tushuntiring;
3. Parametrsiz proseduraning kamchiligi va yutuqlarini sanab bering;
4. Parametr-qiymat va parametr-o'zgaruvchining vazifalarini tushuntiring;
5. Formal va faktik parametrlarning ma`nolarini tushuntiring;
6. Proseduraga qanday murojaat qilinadi?
7. Prosedurani e`lon qilish deganda nimani tushunasiz?
8. Kvadrat tenglama ildizini aniqlab beruvchi prosedura yarating va uni faollashtiring.

13-ma`ruza: Porotsedura-funksiyalar va lokallashtirish prinsipi.

Reja:

1. *Prosedura-funksiyaning vazifasi va uning strukturasi;*
2. *Rekursiv funksiyalar;*
3. *Parametrlarni lokallashtirish prnsipi.*

1. Prosedura-funksiyaning vazifasi va uning strukturasi

Hajmi katta va murakkab programmalarni ishlab chiqishda, tabiiyki katta qiyinchiliklarga duch kelinadi. Katta, kompleks programmalarni zarur muddatda yaratishga bitta dasturchining esa vaqti yetmaydi. Bunday hollarda, ya`ni muhim ahamiyatga ega bo'lgan va qisqa muddatlarda yaratilish kerak bo'lgan programmalarni ishlab chiqish uchun dasturchilarning katta guruhini jalb etishga to'g'ri keladi. Bunday, yagona programmani yaratishdagi paralel ish olib borishda prosedura va funksiyalarning roli juda katta bo'ladi. Bajarilishi kerak bo'lgan ishni mustaqil bo'limlarga ajratilib, har bir mustaqil ish alohida programmalanib, keyinchalik ular yagona - asosiy programmaga birlashtiriladi.

Asosiy programmada ishlatiluvchi o'zgaruvchilar va prosedura parametrlarini qanday tanlab olish kerak degan muammo, bajariladigan ishning eng og'ir qismlaridan biri bo'lib qoladi. Agar ularni bir-birlariga bog'lab yuborilsa u holda asosiy programmadagi biror o'zgaruvchiga kiritilgan o'zgartirish, prosedurada ishlatilgan va shu o'zgaruvchiga bog'liq barcha ishlarni qaytadan tahlil qilib, tekshirib chiqishga olib

keladi. Bunday chalkash va og'ir ishni bajarishning qiyinligi programma tuzishda parallel, bir nechta dasturchining ish olib borishiga halaqit beradi.

Shuning uchun, prosedura va funksiyalarni yozishda har bir programmaga o'zi yechayotgan masalaga muvofiq holda, turli xil ichki o'zgaruvchilar, programmali ob`ektlar o'zgaruvchilarining turli qiymatlarini tanlab olish huquqi beriladi. Xattoki, bitta o'zgaruvchini turli xil vazifalarda ishlatsa ham bo'ladi. Paskal tilida bunday masalani hal qilish uchun lokallashtirish prinsipi ishlab chiqilgan, ya`ni prosedura yoki funksiyada ishlatilgan o'zgaruvchi shu prosedura yoki funksiyaning ta`sir doirasida (ichida) gina o'z qiymatini saqlab qoladi. Prosedura va funksiyalarning ichida aniqlanib, qiymatlangan o'zgaruvchilarni lokal (ichki) o'zgaruvchilar deb ataladi. Tashqarida, ya`ni asosiy programmada kiritilgan o'zgaruvchilar esa umuman olganda programmaning ixtiyoriy joyida o'z qiymatini saqlab qola oladi. Bu o'zgaruvchilarni global (tashqi) o'zgaruvchilar deb ataladi.

Quyidagi misolda lokallashtirish prinsipi yaqqol ko'zga tashlanadi:

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

bu yerda t – asosiy programmaning global o'zgaruvchisi;
x, y – R proseduraning formal parametrlar;
n – P proseduradagi lokal o'zgaruvchi.

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

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

Funktsiya Paskal tilida quyidagi struktura bo'yicha aniqlanadi:

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

Yuqorida eslatib o'tganimizdek, funksiyalar ham proseduralar kabi mustaqil programmalar hisoblanib, asosiy programma orqali boshqariladi va xuddi asosiy programma va proseduraga o'xshash strukturada yoziladi.

Funktsiyani ham prosedura kabi programmaning ***var*** (o'zgaruvchilar tiplarini e`lon qilish) bo'limida aniqlab qo'yiladi. Prosedura uchun aytilgan gaplarning deyarli barchasi funksiya uchun ham o'rinlidir. Funktsiyaning proseduradan asosiy farqi quyidagilardir:

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

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

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

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

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

```
Program L1;
var
      k, m, i :integer; y: real;
function fact (n: integer): integer;
var
```

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

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

Misol:

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

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

2. Rekursiv funksiyalar

Paskal tilida prosedura – funksiyalar bilan ishlashda, funksiyalarning rekursivlik xossasidan foydalanish imkoniyati yaratilgan.

Rekursiya tushunchasiga misol qilib oddiy faktorial hisoblashni keltirish mumkin:

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

bu yerda ko'rinib turibdiki n! qiymati (n-1)! orqali aniqlanayapti, ya`ni rekursiya degani o'zi orqali o'zini aniqlash ma`nosini anglatadi.

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

Yuqoridagi faktorial hisoblashni rekursiv funksiyalar orqali amalga oshiraylik:

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

Funktsiyalarni rekursiv aniqlash qisqa va tushunarli tilda bo'ladi, rekursiv emas aniqlash esa uzoq va funksiyani ko'rinish effektini buzadi, lekin birinchi holda sarflangan EHM vaqti va xotira nisbatan ancha yuqoridir.

3. Parametrlarni lokallashtirish prinsipi

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

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

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

Bu yerda a, b, c, d - qiymatlari beriladigan o'zgaruvchilar;

$f(x) = e^{2x} + \sin 6x$, $g(x) = x^2 - 3x^3\cos x$

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

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

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

```
Program  L1;
var
        a, b, c, d, y, z: real;
function f(x: real) : real;
begin
        f:=exp(2*x)+sin(6*x)
end;
function g(x: real) : real;
begin
        g:=sqr(x) - 3*x*sqr(x)*cos(x)
end;
procedure  int(A, B: real; function F(x: real): real; var R: real);
const
        n=20;
var
        x, h: real; k: integer;
begin
        h:= (B - A)/n;  R:=(f(A)+f(B))/2;
        for k:=1  to  n-1  do
           R:= R+ f(a+ k * h);
        R:= R * h;
end;
{asosiy programmaning operatorlar bo'limi}
begin {1 - integral chegaralarini kiriting}
        read ( a, b);
        int (a, b, f, y);
        writeln( 'y=',y);
        {2-integral chegaralarini kiriting}
        read (c, d); int ( c, d, g, z);
        write ('z=', z);
end.
```

Nazariy savollar va tayanch iboralar:

1. Prosedura-funksiyaning asosiy farqini qanday izohlaysiz?
2. Funktsiya strukturasi qanday aniqlangan?
3. Funktsiyaga murojaat qanday amalga oshiriladi?
4. Rekursivlik xossasini qanday izohlaysiz?
5. Lokal va global o'zgaruvchilar qanday farqlanadi?
6. Parametrlarni lokallashtirish prinsipini izohlab bering;
7. Sinx, Cosx, e^x, $\sqrt{x}$ - kabi standart funksiyalarni xisoblash prosedura-funksiyalarini yarating.

14-ma`ruza: Turbo-Paskalning modullari va foydalanuvchi modulini yaratish. System modulining prosedura va funksiyalari.

Reja:

1. *Turbo-Paskalning asosiy modullari;*
2. *Foydalanuvchi modullarini yaratish;*
3. *Programma ishini bajaruvchi proseduralar;*
4. *Tiplarni almashtirish funksiyalari;*
5. *Sanalma tip funksiyalari va portseduralari;*
6. *Satrlar bilan ishlash prosedura va fugktsiyalari;*
7. *Parametrlar bilan ishlash funksiyalari;*
8. *Adreslar bilan ishlash funksiyalari;*
9. *Modulning boshqa prosedura va funksiyalari.*

1. Turbo-Paskalning asosiy modullari

Shaxsiy komp yuterlarning eng katta kamchiligi ularda amaliy programmalar kutubxonasining to'liq emasligidadir. Katta EHMlarda programma tuzuvchilar uchun juda katta programmalar kutubxonasi xizmat qilar va ulardan foydalanib tuzilgan programmalar o'zlarining ishonchlilik darajasi bilan yuqori turar edi. Shaxsiy kompyuterlarning bu kamchiligini yo'qotish uchun Turbo-Paskalda modullar tushunchasi kiritilgan. Umuman olganda, har bir malakali programma tuzuvchi o'z programmasini prosedura va funksiyalardan foydalanib tuzadi. Lekin, bu prosedura va funksiyalardan boshqa programmalarda foydalanish uchun ularning matnlarini qayta ko'chirib yozish lozim bo'ladi.

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

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

1. ***System*** *moduli* - standart prosedura va funksiyalarni o'z ichiga olib, avtomatik tarzda barcha programmalar uchun ochiqdir;

2. ***DOS*** *moduli* - MS DOS operatsion sistemasi bilan ishlashni tashkil qiluvchi funksiya va proseduralardan tashkil topgan;

3. ***Crt*** *moduli* – ekran, klaviatura va IBM rusumidagi komp yuterlarning tovushli dinamigi bilan ishlash proseduralarini o'z ichiga olgan;

4. ***Graph*** *moduli* - komp yuterning grafik imkoniyatlaridan foydalanib yaratilgan funksiya va proseduralarning katta to'plami;

5. ***Printer*** *moduli* - bu kichkinagina modul printer qurilmasi bilan ishlashni osonlashtiradi;

6. ***Overlay*** *moduli* – katta programmalarni bir nechta bo'laklarga ajratishning kuchli vositasi bo'lib, bir qancha proseduralar va funksiyalardan tashkil topgan.

2. Foydalanuvchi modullarini yaratish

Modullardan foydalanish uchun programma sarlavhasidan

Program <programma nomi>;

keyin quyidagi qator yozilishi kerak:

Uses <modul ismi>;

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

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

Turbo-Paskal bizga o'zimizning modullarimizni yaratib olish imkonini ham beradi. Foydalanuvchi modullari quyidagi strukturada bo'ladi:

Unit <modul ismi>;
Interface
...
{ ochiq e`lonlar bo'limi - interfeys sektsiyasi}
...
Implementation
...
{ yopiq e`lonlar bo'limi}
...
Begin
...
{ Initsializatsiya bo'limi}
...
End.

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

Uses <modullar ro'yxati>;

yoziladi.

Interfeysli bo'lim modulning bir qismi bo'lib, ***Interface*** va ***Implementation*** so'zlari orasida joylashadi. Bu bo'limda o'zgarmaslar, ma`lumotlar tipi, o'zgaruvchilar, prosedura va funksiyalarni aniqlash mumkin. Bu kiritilganlar mazkur modulda qatnashuvchi barcha programmalar va modullarda bemalol ishlatilishi mumkin. Bo'limda sanab o'tilgan prosedura va funksiyalarning tana qismlari ***Implementation*** so'zidan keyin aniqlanadi (ularning sarlavhalari aynan saqlanib qolishi kerak). Bu bo'limda ham, faqat shu bo'lim uchungina "Ko'rinadigan" (ishlatishi mumkin bo'lgan) e`lonlar bo'limi qatnashishi mumkin. Initsializatsiya sektsiyasi ***Begin*** va ***End*** so'zlari ichiga olib yoziladi. Agar ***Begin*** so'zi tushirib qoldirilgan bo'lsa, demak bu sektsiya yo'q hisoblanadi. Initsializatsiya sektsiyasida boshqarishni asosiy programmaga uzatgungacha bajariladigan operatorlar joylashgan bo'ladi. Bu operatorlar asosan programmani ishga tushirishga tayyorlab beradi.

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

```
Unit Stud;
Interface     {ochiq e`lonlar bo'limi – interfeys sektsiyasi}
    function min(x,y:integer):integer;
    function max(x,y:integer):integer;
Implementation    {yopiq e`lonlar bo'limi}
function min(x,y:integer):integer;
Begin
    if x<=y then min:qx else  min:qy;
End;
function max(x,y:integer):integer;
Begin
   if x>y    then max:qx else  max:qy;
End;
Begin
  {Initsializatsiya sektsiyasi yo'q}
End.
```

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

Bu moduldan foydalanish dasturi quyidagicha bo'lishi mumkin:

```
Uses Stud;
Var
   A,b,c,d:integer;
Begin
```

```
   Write('A va B larni kiriting >');
   Readln(a,b);
   C:=max(a,b);
   Writeln('Max= ',C);
    C:qmin(a,b);
    Writeln('Min= ',C);
    D:qmax(a,b)Qmin(a,b);
    Writeln('Max+Min=',D);
End.
```

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

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

```
    Readln(ch);
    If ch in ['M','m','M','m'] then SetMonoColors;
End.
```

3. Programma ishini bajaruvchi proseduralar

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

Quyida ushbu modulning ma`lum bir qism proseduralar va funksiyalari bilan qisqacha tanishib chiqamiz:

1) Programma ishini bajarish proseduralari.

Exit prosedurasi

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

Halt prosedurasi

Vazifasi: programma bajarilishini to'xtatadi va OS ga boshqarishni qaytaradi;
Aniqlanishi: **Halt[(ExitCode:Word)];**
bu yerda majburiy bo'lmagan (o'rta qavs majburiy emas belgisi) ExitCode parametri programmaning yakunlanish kodini beradi, agar bu parametr bo'lmasa bu kod nolga teng bo'ladi.

RunError prosedurasi

Vazifasi: programmaning bajarilishini to'xtatadi va bajarilish vaqtidagi xatolarni aniqlaydi;
Aniqlanishi: RunError[(ErrorCode:Word)];
bu yerda ErrorCode parametrining xatolik nomeri haqidagi ma`lumoti ekranga chop etiladi.

4. Tiplarni almashtirish funksiyalari

Chr funksiyasi

Vazifasi: ASCII jadvalidagi tartib raqami berilgan butun songa mos bo'lgan belgini aniqlaydi;
Aniqlanishi: Chr(N:Byte):Char;
bu yerda N belgining ASCII jadvalidagi tartib raqamini ifodalovchi butun, musbat son.

Ord funksiyasi

Vazifasi: sanalma tipli qiymatlar bo'yicha uning tartib sonini aniqlash;

Aniqlanishi: Ord(X):LongInt;
bu yerda X – sanalma tipli qiymat.

Round funksiyasi

Vazifasi: haqiqiy tipli qiymatni yaxlitlab, katta butun, son hosil qiladi;
Aniqlanishi: Round(X:Real):LongInt;

Trunc funksiyasi

Vazifasi: haqiqiy tipli qiymatning kasr qismini tashlab yuborib, butun son hosil qiladi;
Aniqlanishi: Trunc(X:Real):LongInt;

Arifmetik funksiyalar.

bu funksiyalarning yozilishi va ulardan foydalanish qoidalari 3-bobdagi o'zlashtirish operatori mavzusida to'liq berib o'tilganligi uchun, ularga to'xtalib o'tirmaymiz.

5. Sanalma tip funksiyalari va proseduralari

Dec prosedurasi

Vazifasi: o'zgaruvchi qiymatini kamaytiradi;
Aniqlanishi: Dec(Var X[;n]:LongInt);
bu yerda yozilishi majburiy bo'lmagan "n" o'zgaruvchi argumentning qiymatini qanchaga kamaytirish lozimligini ko'rsatadi. Bu o'zgaruvchi yozilmasa argument qiymati bir soniga kamayadi.

Inc prosedurasi

Vazifasi: o'zgaruvchi qiymatini orttiradi;
Aniqlanishi: Inc(Var X[;n]: Integer);
bu yerda Dec prosedurasiga teskari ish bajariladi.

Odd funksiyasi

Vazifasi: argumentning toq yoki juft sonligini tekshiradi;
Aniqlanishi: Odd(X:LongInt):Boolean;
bu yerda natijaviy qiymat rost (True) bo'lsa son toq, aks holda son juft.

Pred prosedurasi

Vazifasi: argumetni oldingi qiymatini aniqlash;
Aniqlanishi: Pred(X);

Succ prosedurasi

Vazifasi: argumetni keyingi qiymatini aniqlash;
Aniqlanishi: Succ(X);

6. Satrlar bilan ishlash funksiyalari va proseduralari.

ConCat funksiyasi

Vazifasi: satrlarni ketma-ket ulaydi;
Aniqlanishi: ConCat(S1[,S2,...,SN]:String):String;
bu erda S1 qatorni keyingi sanab o'tilgan qatorlar bilan ularni yozilish tartibida ulaydi.

Copy funksiyasi

Vazifasi: satr ichidan yangi satr hosil qilish;
Aniqlanishi: Copy(S:String;Index,Count:Ineger):String;
bu erda
S – berilgan satr;
Index – S satrining nechanchi belgisidan boshlab, yangi satr hosil qilish kerakligini aniqlaydi;
Count – yangi satrdagi belgilar soni.

Delete prosedurasi

Vazifasi: berilgan satr ichidan satr ostini olib tashlaydi;
Aniqlanishi: Delete(Var S:String;Index:Integer;Count:Integer);
bu erda
S – berilgan satr;
Index – shu tartib raqamli belgidan boshlab, S satrdan satr osti olib tashlanadi;
Count – olib tashlanadigan satrdagi belgilar soni.

Insert prosedurasi

Vazifasi: berilgan satrga yangi satr qo'shadi;
Aniqlanishi: Insert(S1:String; Var S2:String; Index:Integer);
bu erda
S2 – berilgan satr;
S1 – qo'shiladigan satr;
Index – S2 satrning qaysi tartib raqamli hadidan boshlab yangi satr qo'shilishini anglatadi.

Length funksiyasi

Vazifasi: satr uzunligini aniqlaydi;
Aniqlanishi: Length(S:String):Integer;
bu yerda S satridagi belgilarning sonini aniqlanadi:

Pos funksiyasi

Vazifasi: satrdan satr ostini qidiradi;
Aniqlanishi: Pos(SubStr,S:String):Byte;
bu erda SubStr – S satrda qidirilayotgan satr osti.
Agar SubStr satr osti S satrida topilsa, Pos funksiyasi mos kelgan birinchi belgining tartib raqamini beradi, agar bu satr osti S satrda bo'lmasa, funksiya nul qiymat beradi.

Str prosedurasi

Vazifasi: sonli qiymatni uning satrli ko'rinishiga o'tkazadi;
Aniqlanishi: Str(X[:Width[:decmals]];Var S:String);
bu yerda yozilishi shart bo'lmagan Width va decmals parametrlari mos ravishda, S satrining hadlar sonini va haqiqiy sonning verguldan keyingi xadlar sonini ifodalaydi.

Val prosedurasi

Vazifasi: satrli qiymatni uning sonli ko'rinishiga o'tkazadi;
Aniqlanishi: Val(S:String;Var BVar Code:Integer);
bu yerda
S – berilgan satr;
B– S satrning uning sonli ko'rinishiga o'tkazganidan keyin hosil bo'lgan sonni saqlash joyi;
Code – butun tipli o'zgaruvchi.
Agar S satrni songa aylantirib bo'lmasa Val prosedurasi bajarilganidan so'ng Code o'zgaruvchisi nul qiymatni qabul qiladi.

7. Parametrlar bilan ishlash funksiyalari

Programmani ishga tushirishda unga uzatiladigan qiymatlarni parametrlar deb hisoblaymiz.

Misol: Sistemali programmalashning quyidagi Copy buyrug'ini ko'raylik:

Copy file.dat c:g'progg'file2.dat

ya`ni, aktivlashgan katalogdagi file.dat faylining nusxasini «S» diskning prog katalogiga file2.dat nomi bilan ko'chirish. Bu yerda Copy buyrug'iga parametr sifatida "file1.dat"va "file2.dat" qatori uzatilmoqda.

Turbo-Paskalda parametrlar bilan ishlash uchun ParamCount va ParamStr funksiyalari mavjud.

ParamCount funksiyasi

Vazifasi: buyruqli qatordan programmaga uzatilgan parametrlar sonini aniqlaydi;
Aniqlanishi: ParamCount:Word;

ParamStr funksiyasi

Vazifasi: ko'rsatilgan nomerdagi parametrni aniqlaydi;
Aniqlanishi: ParamStr(N:Integer):String;

8. Adreslar bilan ishlash funksiyasi

Addr funksiyasi

Vazifasi: ko'rsatilgan ob`ektning adresini aniqlaydi;
Aniqlanishi: Addr(X):Pointer;

bu yerda X – ixtiyoriy tipli o'zgaruvchi yoki programmada e`lon qilingan prosedura yoki funksiya ismi.

Seg funksiyasi

Vazifasi: ko'rsatilgan o'zgaruvchi adresining segment qiymatini aniqlaydi;
Aniqlanishi: Seg(X):Word;
bu yerda X – ixtiyoriy tipli o'zgaruvchi yoki programmada e`lon qilingan prosedura yoki funksiya ismi.

9. Modulning boshqa prosedura va funksiyalari

FillChar prosedurasi

Vazifasi: ko'rsatilgan qiymat bilan ketma-ket kelgan va chekli sondagi baytlarni to'ldiradi;
Aniqlanishi: FillChar(Var X;Count:Word;Value);

Hi funksiyasi

Vazifasi: argumentning katta baytini aniqlaydi;
Aniqlanishi: Hi(X):Byte;

Lo funksiyasi

Vazifasi: argumentning kichik baytini aniqlaydi;
Aniqlanishi: Lo(X):Byte;

Move prosedurasi

Vazifasi: ko'rsatilgan miqdordagi baytlardan tezkor (operativ) xotiraning bir qismidan ikkinchi qismiga nusxa ko'chiradi;
Aniqlanishi: Move(Var Cource,Dest;Count:Word);

Random funksiyasi

Vazifasi: tasodifiy sonni aniqlaydi;
Aniqlanishi: Random([Range:Word]);
bu yerda aniqlanadigan tasodifiy son 0 va Range sonlari oralig'ida yotadi.

Randomize prosedurasi

Vazifasi: tasodifiy sonni hosil qiluvchi generatorni ishga tushiradi;
Aniqlanishi: Randomize;

UpCase funksiyasi

Vazifasi: kichik lotin harflarni katta harflarga o'tkazadi;
Aniqlanishi: UpCase(Ch:Char):Char;
funksiya rus alfaviti harflarini ham katta harflarga o'tkazadi.

Nazariy savollar va tayanch iboralar:

1. Dasturlar kutubxonasini so'zining ma`nosi nima?
2. System moduli qanday vazifani bajaradi?
3. DOS modulining imkoniyatlarini baholang;

4. Graph moduli haqida umumiy ma`lumotlar bering;
5. Printer moduli qanday qurilma bilan ishlashni ta`minlaydi?
6. Overlay moduli programmalar ustida qanday ishlar bajaradi?
7. Crt modulining imkoniyatlarini sanab bering;
8. Modullar programmaga qanday bog'lanadi?
9. Foydalanuvchi moduli deganda nimani tushunasiz?
10. Foydalanuvchi modulining strukturasi qanday bo'ladi?
11. Foydalanuvchi modullari qanday yaratiladi.
12. System moduli qanday o'rnatiladi;
13. Programma ishini bajarish proseduralarini vazifalarini tushintiring;
14. O'zgaruvchilarning tiplarini almashtirish funksiyalarini ayting va ularni ishlash prinsiplarini tushuntiring;
15. Sanalma tiplar bilan ishlash funksiyalari va proseduralarini sanab bering va ularni qanday ishlashini tushuntiring;
16. Satrlar bilan ishlashni tashkil qilib beruvchi funksiya va proseduralarni sanab bering va ularning qanday ishlashini tushuntiring;
17. Programmani ishga tushirishda uzatiladigan parametrlar bilan ishlash funksiyalarini tushuntiring;
18. Adreslar bilan ishlash funksiyalarining vazifalarini tushuntiring;
19. Modulning boshqa prosedura va funksiyalarini ishlash prinsiplarini tushuntiring.

15-Ma`ruza: Crt, MS DOS, Printer va Overlay modullarining prosedura va funksiyalari.

Reja:

1. *Crt modulining prosedura va funksiyalari;*
2. *MS DOS modulining prosedura va funksiyalari;*
3. *Printer moduli;*
4. *Dasturni overleyli qurish;*
5. *Overleyli dasturni rasmiylashtirish qoidalari;*
6. *Overleyni ishga sozlash;*
7. *Overley buferini boshqarish*.

1. Crt modulining prosedura va funksiyalari

Yuqorida qisqacha aytib o'tganimizdek, Crt moduli matnli ekran, klaviatura va tovush dinamigi bilan ishlashni tashkil etuvchi programmalar bilan jixozlangan.

Matnli rejimdagi ekran odatda 25 ta qator va har bir qatorda 80 ta belgi sig'dirish imkoniyatiga ega. Ekranda hosil qilish mumkin bo'lgan ranglar soni esa 16 ta (oq-qora rangli ekran rangini ko'zda tutmayapmiz).

Ekranda kerakli belgini zarur joyda chiqarish, matn va fon rangini o'zgartirish va ekranning ishini to'liq boshqara olish Crt modulining vazifalaridan hisoblanadi:

- **GotoXY(I,J:Byte)** prosedurasi ekranning I ustun va J satriga kursorni ko'chirib berdi;
- **Write(S)** prosedurasi kursor turgan joydan S qatorni chop etadi;

- **TextColor(Color:Byte)** prosedurasi ekranga chiqariladigan matn rangini o'rnatadi;
- **TextBackGround(Color:Byte)** prosedurasi chop etiluvchi matn uchun ekran fonining rangini o'rnatadi;
- **ClrScr** prosedurasi TextBackGround prosedurasi o'rnatgan rang bilan ekranni tozalaydi, agar rang o'rnatilmagan bo'lsa ekran qora rang bilan tozalanadi, kursor ekranning chap yuqori qismiga chiqariladi;
- **AssignCrt(Var F:Text)** prosedurasi Crt modulini o'rnatish uchun matnli faylni ta`minlaydi;
- **ClrEol** prosedurasi kursor turgan joydan boshlab ekranni tozalaydi. Agar Window prosedurasi bilan oyna ochilgan bo'lsa tozalov shu oyna ichida bo'ladi. Kursor o'z o'rnini o'zgartirmaydi;
- **Delay(N:Word)** prosedurasi N milli sekund davomida to'xtovni tashkil etib beradi;
- **DelLine** prosedurasi kursor turgan qatorni o'chiradi. Qolgan qatorlar bir satrga yuqoriga siljiydi va ekranning oxirgi qatori tozalanadi;
- **HighVideo** prosedurasi chop etilayotgan belgilar uchun yuqori yorug'lik o'rnatadi;
- **InsLine** prosedurasi kursor turgan joyga bo'sh qator o'rnatadi;
- **KeyPressed:Boolean** funksiyasi agar klavish bosilsa True, klavish bosilmagan bo'lsa False qiymatni qaytaradi;
- **LowVideo** prosedurasi chop etilayotgan belgilar uchun past yorug'lik o'rnatadi;
- **NormVideo** prosedurasi chop etilayotgan belgilar uchun o'rtacha yorug'lik o'rnatadi;
- **NoSound** prosedurasi tovush dinamigini o'chiradi;
- **ReadKey:Char** funksiyasi klaviaturadan belgini o'qiydi;
- **Sound(X:Word)** prosedurasi tovush dinamigini ulaydi va uni berilgan X chastotada tovush chiqarishga majbur qiladi;
- **TextMode(N:Integer)** prosedurasi kerakli N-matnli rejimni o'rnatadi;
- **WhereX:Byte** funksiyasi kursor turgan joyning ustun nomerini (qaralayotgan oynada), ya`ni X koordinatasini aniqlaydi;
- **WhereY:Byte** funksiyasi qaralayotgan oynaga nisbatan kursor joyining qator nomerini, ya`ni Y koordinatasini aniqlaydi;
- **Window(X1,Y1,X2,Y2:Byte)** prosedurasi ekranda matnli oyna hosil qiladi. Bu oyna o'zini xuddi to'liq ekrandek his etadi.

Ranglar bilan ishlashni qulaylashtirish maqsadida Crt modulida quyidagi o'zgarmaslar kiritilgan:

```
Const
Black=;{Qora}
Blue=;{Ko'k}
Green=;{Yashil}
Cyan=;{Zangori rang}
```

Red=;{Qizil}
Magenta=;{Pushtirang}
Brown=;{Jigarrang}
LightGray=;{Och kulrang}
DarkGray=;{To'q kulrang}
LightBlue=;{Och havorang}
LightGreen=0;{Och ko'k}
LightCyan=1;{Och zangorirang}
LightRed=2;{Och qizil}
LigthMagenta=3;{Och pushtirang}
Yellow=4;{Sariq}
White=5;{Oq}

2. MSDOS modulining prosedura va funksiyalari

DOSning sistemali moduli 6 kilobayt atrofidagi joy egallab, MSDOS muhiti bilan ishlashga va uning imkoniyatlaridan foydalanishga mo'ljallangan proseduralar va funksiyalarga ega. Aslida bu proseduralar va funksiyalar Paskal tilining sintaktik qoidalariga moslashtirilgan MSDOSni chaqirish funksiyalaridir.

DOS modulidagi prosedura va funksiyalarni bajaradigan ishlarining ma`nolariga ko'ra oltita funktsional guruhga ajratish mumkin:

- MSDOS parametrlarini so'rovi va ularni o'rnatish;
- ShEHMda kalendar va soatlar bilan ishlash;
- Disk resurslarining analizi;
- Katalog va fayllar bilan ishlash;
- MSDOS uzilishlari(**prero'vanie**) bilan ishlash;
- Rezident programmalar va subprotsesslar bilan ishlash.

1) MSDOS parametrlarini so'rovi va ularni o'rnatishning proseduralari va funksiyalari:

- DOSVersion:Word funksiyasi MSDOS versiyasining kodlashtirilgan nomerini aniqlaydi;
- **GetCbreak (Var B: Boolean)**-prosedurasi Break parametrining qiymatini o'qiydi;
- **SetCbreak(Var B: Boolean)** prosedurasi Break qiymatini o'rnatadi;
- **GetVerify(Var V:Boolean)** prosedurasi Verify parametrining qiymatini o'qiydi;
- **SetVerify(V:Boolean)** prosedurasi Verify qiymatini o'rnatadi;
- EnvCount:Integer funksiyasi MSDOSning sistemali o'zgaruvchilari sonini aniqlaydi;
- **EnvString(N:Integer):String** funksiyasi N nomerli MSDOS o'zgaruvchisining to'liq vazifa qatorini beradi;
- **GetEnv(E:String):String** funksiyasi E sistemali o'zgaruvchining qiymatini aniqlaydi.

2) SHEHMda kalendar va soatlar bilan ishlash prosedura va funksiyalari:

DOS modulida komp yuter soati va kalendar bilan ishlash hamda fayllarni tashkil qilish kuni va vaqtini o'rnatish proseduralari mavjud:

- **GetDate(Var Year,Month,Day,Dw:Word)** ShEHM soatidan yil, oy, kun va hafta kunini o'qiydi;
- **SetDate(Year,Month,Day:Word)** prosedurasi ShEHMning soatiga yil, oy va kunni o'rnatadi;
- **GetTime(Var Hour,Min,Sec,Sec100:Word)** prosedurasi ShEHM soatidan hozirgi vaqtni aniqlaydi;
- **SetTime(Var Hour,Min,Sec,Sec100:Word)** prosedurasi ShEHM soatiga yangi vaqtni o'rnatadi;
- **PackTime(Var Dt:Datetime; Var T:Longint)** prosedurasi kun,oy, yil va vaqt ma`lumotlarini faylga yozish uchun ixcham holda tayyorlaydi.
- **UnPackTime(Var T:Longint;Var Dt:Datetime)** prosedurasi fayldan o'qilgan kun, oy, yil va vaqt yozuvini ochib beradi;
- **GetFtime(Var F; Var T:Longint)** prosedurasi ochiq F fayli uchun kun, oy, yil va vaqtning ixcham yozuvini o'qib beradi;
- **SetFtime(Var F; Var T:Longint)** prosedurasi ochiq F fayli uchun yil, oy, kun va vaqtning ixcham yozuvini yozib qo'yadi.

3) Disk resurslari tahlilining funksiyalari:

DOS moduli o'z ichiga disk holatini tahlil qilishning ikkita funksiyasini oladi:

- **DiskFree(D:Word):Longint** funksiyasi diskdagi bo'sh joy o'lchovini aniqlaydi;
- **DiskSize(D:Word):Longint** funksiyasi diskning to'liq xajmini baytlarda aniqlab beradi. Bu yerda D butun qiymatli parametr yoki butun son bo'lib, uning qiymati aniq bir diskni ko'rsatadi:

- Agar D=0 bo'lsa ishlatilayotgan joriy disk, Dq1 bo'lsa A disk, Dq2 bo'lsa B disk va hokazolar tahlil qilinadi;
- Agar mazkur sistema Dning kiritilgan qiymatiga mos diskni topa olmasa funksiya -1ga teng qiymat beradi;

4) Kataloglar va fayllar bilan ishlash funksiyalari va proseduralari.

Paskal tilining tashqi fayllar bilan ishlash imkoniyatlari juda cheklangan bo'lib, ular asosan quyidagilardan tashkil topgan: faylni ochish, yopish, qayta nom berish va o'chirish. Bu kamchiliklarni bartaraf qilish uchun DOS modulida bir qancha funksiyalar va proseduralar ko'zda tutilgan:

- **FindFirst(Path:String;Attr:Word;Var Sr:SearchRec)** prosedurasi Path so'rovi bo'yicha Attr atributli birinchi mos ismni topadi;
- **FindNext(Var Sr:SearchRec)** prosedurasi FindFirst prosedurasidan keyin chaqirilib, keyingi mos ismlarni topish uchun ishlatiladi;
- **FSearch(Path:PathStr; DirList:String):PathStr** funksiyasi DirList kataloglar ro'yxatidan Path ismli faylni qidiradi;

- **GetFattr(Var F:File; var Fa:Word)** prosedurasi F fayli bilan bog'liqli diskdagi faylning Fa atributini o'qiydi;
- **SetFattr(Var F:File; var Fa:Word)** prosedurasi F fayli bilan bog'liqli diskdagi faylga atribut o'rnatadi;
- **Fsplit(Path:PathStr; Var Dir:DirStr; Var Name:NameStr; Var Ext:ExtStr)** prosedurasi Path faylining to'liq ismini uni tashkil etuvchilariga ajratadi: Dir-yo'li, Name-ismi, Ext-kengaytmasi.

5) MSDOS uzilishlari bilan ishlash.

MSDOS ning sistemali funksiyalarini chaqirish uzilishlar ko'rinishida hal qilinadi. Har bir uzilish aktivlashtirilgandan keyin turli xil funksiyalar to'plamiga murojaat qilish imkoniyatini yaratadi. Masalan, 16H nomerli uzilish, operatsion sistemasi darajasidagi klaviatura bilan muloqotni tashkil etuvchi funksiyalarga yo'l ochib beradi, 25H nomerli uzilish esa diskni o'qish ishlarini boshqarishni o'z zimmasiga oladi va hokazo. Juda katta vazifalar 21H uzilishi zimmasidadir, chunki u operatsion sistema (OS)ni tashkil etuvchi o'nlab funksiyalardan foydalanishni tashkil qilib beradi.

Quyida e`tiboringizga Paskal - programmasidan turib MSDOSning funksiyalariga murojaat qilishni tashkil qilib beruvchi DOS moduli proseduralarining vazifalari va aniqlanishlarini havola qilamiz:

- **GetIntVec(N:byte; Var Adress:pointer)** prosedurasi Adress ko'rsatkichiga berilgan N nomerli uzilish qism dasturining adresini aniqlab beradi;
- **SetIntVec(N:byte; Var Adress:pointer)** prosedurasi N-nomerli uzilishning yangi qism dasturini DOS ga o'rnatib, adresning eski qiymatini Adress ko'rstkichiga joylaydi;
- **Intr(N:byte; Var R:Register)** prosedurasi N programmali uzilishni aktivlashtirib, funksiya nomeri va parametrlarini R o'zgaruvchisiga uzatadi;
- **MsDos(Var R:Register)** prosedurasi 21H nomerli uzilishni maxsus chaqirish vazifasini bajaradi.

Oxirgi prosedurada yangi Register tipi kiritildi. Bu tip DOS modulida aniqlangan quyidagi yozuvdan iborat:

```
Type
    registerqRecord
        case Integer of
                0:(AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags:Word);
                1:(AL,AH,BL,BH,CL,Ch,DL,DH:Byte);
        end;
```

Bu tipdagi o'zgaruvchilar mikroprotsessor registrlarga Intr va MsDos larni chaqirishda yo'l ochish uchun ishlatiladi:

0-variantida-16 razryadli registrlarga 1-variantda esa 8 razryadli protsessor yacheykalariga murojaat qilinadi.

6) Rezident programmalar va subprotsess (subjarayon)larni tashkil kilish.

MS-DOS operatsion sistemasi subprotsesslarini tashkil qilish bo'yicha katta imkoniyatlarga ega. Subprotsessda bir programma boshqa bir programmani ishga tushirsa, boshka bir programma esa o'z navbatida keyingi programmani ishga tushiradi va hakozo. Bu holda ishga tushiruvchi programma (protsess) subprotsessni ishini tashkil qilib o'zi ishdan to'xtaydi, lekin ShEXM xotirasida saqlanib turadi. Subprotsess esa xotiraning qolgan qismida ixtiyoriy programma kabi bajariladi. U o'z ishini yakunlagach keyingi jarayon programmasi ish boshlaydi.

MS-DOS ning bu xossasi integrator-programmalarni yaratish imkoniyatini hosil qiladi. Hammaga tanish bo'lgan Norton Commander programmasi subprotsesslardan aktiv foydalanuvchi, rezident bo'lmagan programma hisoblanadi.

Oddiy programmalar o'z ishini yakunlagach ShEHM xotirasini bo'shatib qo'yadi. Subprotsesslar esa ishini yakunlagach, boshqarishni o'zlarini chaqirgan programmalarga uzatadi, ular esa o'z navbatida keyingisi va jarayon oxirida boshqarish OS (operatsion sistema)ga uzatiladi.

Rezident programmalar esa boshqacha ishlaydi. Ular ishga tushgach darrov qandaydir amallarni bajarishi mumkin, yoki OS uzilishlari bilan turli xil tushunarsiz ishlarni amalga oshirishi mumkin. Ularni asosiy hislati shuki, o'z ishini yakunlagach ShEHM xotirasida saqlanib qoladi va biror bir shartning bajarilishiga qarab qayta "tirilishi" ya`ni yana ishlay boshlashi mumkin. Rezident programmalarga misol qilib quyidagi programmalarni ko'rsatish mumkin:

Side Kick sistemasi, ALFA va BETA kirill drayverlari, antivirus programmalari, virus programmalari va hakozo. Rezident programmalarning barchasini bir xil narsa birlashtiradi, bu ham bo'lsa ularni ichki(tezkor) xotirada (OZU) joylashishi va maxsus shartlarda (ALFA, BETA klavishlarini bosishda, antivirus programmalarga murojaatda, qo'yilgan vaqtni yetib kelishida va hakozo) qayta "tirilishi". Passiv holatlarda rezident programmalar faqat ortiqcha joy ushlab turishadi, boshqa oddiy programmalar ishiga halaqit bermaydi.

ShEHM xotirasidan unumli foydalangan holda subprotsesslarni tashkil qilish va rezident programmalar bilan ishlashning maxsus proseduralaridan foydalanish mumkin:

- **SwapVectors** protsedrasi ishga tushirilayotgan subprotsesslar uchun sistemali yoki vaqtinchalik uzilishlar vektorlarini tiklaydi;
- **Exec (ExeFile, ComLine: String)** prosedurasi ComLine satrli parametr bilan ExeFile (subprotsess) bajariluvchi faylini ishga tushiradi;
- **DosExitCode: Word** funksiyasi subprotsess yakunlanganligi haqidagi kodni aniqlaydi;
- **Keep(ExitCode:Word)** prosedurasi ShEHM xotirasidan o'chmagan holda programma ishini yakunlaydi (rezident kodini tashkil qiladi).

3. Printer moduli

Paskal tilining kutubxonasida sanab o'tilgan modullar qatorida Printer moduli ham o'zining munosib joyini egallagan, chunki olingan natijalarni printer orqali

qog'ozga chop etish muhim ishdir. Bu modulning asosiy vazifasi programma va chop etuvchi qurilma (odatda printer) o'rtasida aloqa o'rnatishdir.

Bu kichik modul bor-yo'g'i bitta o'zgaruvchi va bajariluvchi matnning ikki qatoridan tashkil topgan. Bu modul Lst o'zgaruvchisi va LPT1 (birinchi parallel port) sistemali qurilma mosligini o'rnatadi. Modulning to'liq matni qo'yidagicha:

```
Unit Printer;
Interface
Var
  Lst:Text;
Implementation
Begin
  Assign(Lst,'LPT1');
  Rewrite(Lst)
End.
```

Printer modulini ulangandan so'ng ***Write(Lst, ...)*** **va** ***WriteLn(Lst, ...)*** operatorlari natijaviy ma`lumotlarni to'g'ridan-to'g'ri printerga chiqaradi.

Agar printer boshqa qurilma orqali ulangan bo'lsa, masalan ikkinchi port **LPT2** yoki ketma-ketli SOM1 porti orqali u holda dasturchi o'zining Printer2 nomli modulini yaratib olishi mumkin. U holda yuqoridagi modul programmasi matnidagi LPT1 yozuvini boshqa ismga (LPT2 yoki SOM1) almashtirilib translyatsiya qilib olinadi.

Natijalarni yoki maxsus buyruqli kodlarni printerga Lst fayli (yoki shunga o'xshagan) orqali chiqarish, chop etishni boshqarishning yagona yo'li emas. BSVV17N uzilishi orqali printerga belgilar yoki ularning ketma-ketligini uzatish mumkin. Bundan tashqari mazkur uzilishdan foydalanish, chop etish qurilmasining holatini so'rash imkoniyatini yaratadi.

17N uzilishi printerga xizmat ko'rsatishning 3 xil funksiyasidan foydalanishga ruxsat beradi:

- belgilarni printerda chop etish (0-nomerli funksiya);
- printer portini ishga sozlash (1-nomerli funksiya);
- printer holatini o'qish (2-nomerli funksiya).

4. Dasturni overleyli qurilish.

Juda ko'p holatlarda, dastur kompilyatsiya qilingandan so'ng ma`lum bo'ladiki dastur uchun ShEXM ning ichki tezkor xotirasi yetishmay qoladi, ya`ni dastur talab qilgan xotira joyining miqdori ShEXM ning xotirasidan katta bo'ladi. Bu holatda ekranda Not enough memory («xotira yetishmayapti») ma`nosidagi xato ko'rsatilgan ma`lumot hosil bo'ladi Bunday holatlarda dasturchiga Overlay moduli yordamga oshiqadi.

Overlay moduli dasturni alohida bo'laklarga ajratishning kuchli vositasi hisoblanadi. bunday bo'laklar soni ixtiyoriy sonda bo'lishi va bu bo'laklarga zarur bo'ladigan joy miqdori ShEXM imkoniyatidan ancha yuqori bo'lishi mumkin. Overleyli

tarzda tuzilgan dastur bitta . yeXE kengaytmali fayldan va shunday nomli lekin .OVR kengaytmali yana bitta fayldan tashkil topgan bo'ladi. Bunda yeXE – fayl dasturning doimiy qismini, OVR – fayli esa zarur paytda xotiraga yuklanadigan kodlarni saqlab turadi. Overleyli tarzda tuzilgan dasturda xotirada faqat xozirdagina zarur bo'ladigan overleyli prosedura va funksiyalargina saqlanadi, ular kerakli paytdagina o'zlarining joylarini boshqa prosedura va funksiyalarga bo'shatib beradi. Bu holda qo'shimcha xotira joyi talab qilinmaydi chunki dasturning overleyli qismlari xotiraning faqat bir qisminigina navbat bilan ishlatadi xolos. Xotiraning mazkur qismi overleyli bufer deb ataladi. Overleylarni xotiraga yuklash va xotiradan bo'shatish kabi ichki amallarning barchasini overley administratsiyasi avtomatik tarzda bajaradi. Dasturchidan esa faqat dasturning overleyli parchalarini e`lon qilish va uning administratsiyasini ishga solish talab qilinadi xolos.

5. Overleyli dasturlarni rasmiylashtirish qoidalari

Turbo Paskalda overleylar modullar darajasidagina yaratilishi mumkin. Overleylar xuddi oddiy modullar kabi e`lon bo'limi (INTERFACE) va xisoblash bo'limi (IMPLEMENTATION), hamda uni faollashtiradigan qismlardan tashkil topadi. Shu bilan bir qatorda quyidagi shartlarni bajarilishi ham talab qilinadi.

- xar bir overley modul uchun unga oldindan overleylilik huquqini beruvchi {$FQ} kompilyatsiya rejimi o'rnatilishi lozim;
- overleyli qism-dasturlardan bevosita va bilvosita foydalanuvchi barcha prosedura va funksiyalar {$FQ} kompilyatsiya qilinishi lozim.

Bu shartlarni bajarish uchun overleyli modulning bosh qismiga {$FQ, $OQ} direktivalarini o'rnatib, asosiy dasturning birinchi qatorlariga {$FQ}, kompilyatsiya rejimining kaliti yozib qo'yiladi.

Bundan tashqari, asosiy dastur overlay moduliga ulangan bo'lishi va uses direktivasidagi qaysi modullar overleyli ekanligi ko'rsatib qo'yiladi.

Misol sifatida quyidagi dasturning bosh qismini ko'rsatib o'tamiz:

```
Program MultiCalc;
{$FQ}
uses crt,dos,Overlay, MultiCalc, divCalc AddCalc, Subcalc;
{ $O MultiCalc}          {  MultiCalc moduli –overley }
{ $O DivCalc}            { DivCalc moduli –overley }
{ $O AddCalc}            { AddCalc moduli –overley }
{ $O SubCalc}            { SubCalc moduli –overley }
.    .    .    .    .    .    .    .    .    .
```

Ahamiyat bergan bo'lsangiz { $O <modul ismi >} direktivasi overleyli modullarni ko'rsatish uchun yozilishga majburdir, bundan tashqari, uses direktivasining modullari ro'yxatida overlay moduli overleyli modullar ro'yxatining boshida kelishi shart.

Overleyli dasturlar xotirada kompilyatsiya qilinmaydi, endi ular faqat disklardagina hosil qilinadi. Dastur kompilyatsiyasidan so'ng (bu dasturni MULTICALC.PAS nomi bilan ataylik) MULTICALC.EXE bajaruvchi fayli va overleyli deb e`lon qilingan barcha modullardagi proseduralarning kodlarini saqlovchi yo'ldosh fayli hosil qilinadi.

6. Overley ishga sozlash

Overley administratorini yuklash dasturni ishlashi davomida faqatgina bir marta bajarilishi shart. Bu ish quyidagi overlay modulida e`lon qilingan proseduraga murojaat orqali amalga oshiriladi:

OvrInit (Ovr File Name: string)

Bu prosedura overley administratorini ishga yuklaydi va ovr overleyli faylini ochadi.

2. Overleyni yuklash natijalarining tahlili uchun Overlay modulidagi oldindan aniqlab qo'yilgan butun tipli OvrResult o'zgaruvchisi mavjud bo'lib, u mazkur modul prosedura va funksiyalarining, shu jumladan OvrInit prosedurasining ishlarini yakunlash kodini o'zida saqlaydi. OvrResult o'zgaruvchisi yetti hil qiymat qabul qilishi mumkin va bu qiymatlarning har biriga mos ravishda o'zgarmaslar biriktirib quyilgan.

Bu ma`lumotlar quyidagi jadvalda keltirilgan

O'zgarmaslar	O'zgarmasning ma`nosi
OvrOk*0	to'g'ri yakunlanishi
OvrEror *-1	Overlay ni boshqarish xatosi
OvrNotFound*-2	.Ovr fayli topilmadi
Ovr NoMemory*-3	Bufer uchun xotira yetishmayapti
OvrIOEror*-4	overleyli faylni o'qishda buzilish ro'y yuergan
OvrNoEMSDriver*-5	EMS draveyri o'rnatilmagan
OvrNoEMSMemory*-6	EMS – xotira hajmi yetarli emas

Endi sanab o'tilgan xatoliklar haqida to'liqroq to'xtalib o'taylik:

- OvrEror xatosi - overleyli bo'lmagan faylni yuklashga xarakat qilinganda ro'y beradi ;
- OvrNotFound xatosi - overleyli faylni diskka noto'g'ri joylash oqibatida yuzaga kelishi mumkin ;
- OvrNoMemory xatosi - ShEXM ning bo'sh xotirasi yetishmasligidan ro'y berishi mumkin ;
- OvrIOEror xatosini - kelib chiqishiga ko'proq tashqi faktorlar sababchi bo'ladi (faylni shikastlanishi,MS –DOS ichidagi buzilishlar) ;

- OvrEMSDriver va OvrNoEMSMemory xatoliklari OvrInit proseduraSining emas, balki OvrInitEMS qo'shimcha proseduraSining ishidagi xatoliklaridan kelib chiqadi.

7. Overley buferini boshqarish

Overley buferining o'lchovlarini boshqarish va uni tozalash uchun Overlay modulida mahsus prosedura va funksiyalar kiritilgan:

OvrGetBuf funksiyasi

Vazifasi : ishchi buferning baytlardagi o'lchovlarini aniqlaydi ;

Aniqlanishi : **OvrGetBuf :Longint** ;

bu yerda funksiyaning qiymati ba`zi hollarda 64 kb dan oshishi mumkin, shuning uchun, funksiya qiymatini longint deb e`lon qilinadi.

OvrSetBuf prosedurasi

Vazifasi :bufer o'lchovini moslashtirib turadi;

Aniqlanishi : **OvrSetBuf (Size: longint);**

bu prosedura OvrInit va OvrInitEMS proseduralari yordamida overleylar administratsiyasi yuklangandan so'ng ishga tushishi lozim. Buferning talab qilinayotgan baytlardagi miqdori Size parametri orqali beriladi.

OvrClearBuf - prosedurasi

Vazifasi : overley buferini majburiy tarzda tozalash uchun ishlatiladi.

Aniqlanishi : **OvrClearBuf ;**

bu prosedura buferdagi turgan barcha overleyli dasturlarni chiqarib tashlash vazifasini bajaradi. Overleylar administratorining o'zi, bu proseduraNi chaqirmaydi.

Nazariy savol va tayanch iboralar:

1. Matnli rejimda ekranning holati haqida qanday ma`lumotlar bilasiz?
2. Ekranda kerakli belgini zarur joyda chiqarish, matn va fon rangini o'zgartirish, hamda ekran bilan ishlashni tashkil qilish funksiyasi va proseduralarning yozilishi va ulardan foydalanish imkoniyatlarini aytib bering;
3. Ranglarni aniqlovchi o'zgarmaslar qiymatlari haqida ma`lumot bering;
4. MS DOS moduli qancha joy egallaydi?
5. MS DOS modulining vazifasi va imkoniyatlari;
6. MS DOS parametrlarini o'rnatish funksiya va proseduralarini vazifasini tushuntiring;
7. Kalendar va soatlar o'rnatish funksiya va proseduralari haqida ma`lumot bering;
8. Disk resurslarini aniqlash funksiyalarini yozing va ulardan foydalanishni tashkil qiling;
9. Kataloglar va fayllar bilan ishlash proseduralari va funksiyalarini ishlash prinsipini tushuntiring;
10. MS DOS da uzilishlar bilan ishlash imkoniyatlarini yaratish funksiya va rotseduralari bilan tanishtiring;
11. Rezident va programlari va sub jarayonlarni tashkil qilish yo'llarini tushuntiring.
12. Printer moduli qanday ishni bajaradi?
13. Chop etuvchi qurilma xolatini qanday aniqlanadi?
14. Overlay modulidan qanday vaziyatlarda foydalanish lozim?
15. Dasturni overlayli qurish deganda nimani tushunasiz?
16. Overleyli dasturlarni rasmiylashtirish shartlarini sanab bering.
17. Overleyni initsalizatsiya qilish qay tarzda kechadi?
18. OverResult o'zgaruvchisining qabul qilish mumkin bo'lgan o'zgarmaslarni ma`nosini tushuntiring.
19. Overley buferi ishini boshqaruvchi funksiya va proseduralarni izohlab bering.

16-ma`ruza: Grafika rajimida ekran xolatlari hamda turli xil figuralar qurish, matnlar yozish va xatoliklar taxlili.

Reja:

1. *Turbo-Paskalning grafik rejimida ekran xolati;*
2. *Displeyni grafik rejimga o'tkazish;*
3. *Nuqta chiziq va ranglar;*
4. *Turli xil figuralar;*
5. *Grafik rejimdagi matnlar;*
6. *Ekran soxalari;*
7. *Xatolar taxlili;*
8. *Graph moduolining prosedura va funksiyalari.*

Barcha programmalash tillari kabi Paskal tili ham, dasturchiga faqat matnli axborotlar bilangina emas, balki grafik ma`lumotlar bilan ham ishlash imkoniyatini yaratadi. Graph moduli turli xil displey adapterlari (CGA,EGA,VGA,MCGA,Hercules,PC3270,AT&T6300 va IBM8514)ning grafik

rejimlarini to’liq boshqarish imkoniyatini yaratuvchi saksondan ortiq prosedura va funksiyalarning kutubxonasini ifoda etadi.

1. Turbo-paskalning grafika rejimida ekran xolati

Ma`lumki, ekran to’rtburchak maydon bo’lib, juda ko’p nuqta(Pixel)lardan tashkil topgan. Grafik ekran ishchi maydon va bordyur (ekranning chetki qismi)dan iborat bo’ladi.

Grafik rejimda ekrandagi barcha nuqtalarning ranglarini o’zgartirib chiqish mumkin (eslatib o’tamiz, matnli rejimda bu mumkin emas). Turli rangga bo’yalgan nuqtalar yordamida chiziqlar, matnlar va boshqa xar hil tasvirlar hosil qilinadi. Ekran turiga qarab ranglar soni turli hil bo’lishi mumkin, eng kami bilan esa 2 hil rang bo’ladi.

Komp yuter ekrani yoki matnli rejim yoki grafik rejim holatida bo’ladi. Bir vaqtning o’zida ekranning bir qismi grafik rejimda, bir qismi matnli rejimda bo’lishi mumkin emas. Chunki ekranda ko’rinayotgan barcha tasvirlar video xotiradagi ma`lumotlarning aksi - tasviridir.

Ekranning grafik yoki matnli rejimlariga qarab, videoxotiradagi ma`lumotlar turlicha bo’ladi.

Videoxotira, yorug’lik trubkasining kontrolleri, kirish-chiqish portlari va h.k. lar bitta platada joylashadi va displey adapteri deb ataladi. Amalda bir necha xil displey adapterlari mavjud bo’lib, ular quyidagi ko’satkichlari bilan bir-biridan farq qiladi :

- ekranning tasvirni ko’rsatish sifati;
- ekranda bir vaqtda ko’rsatish mumkin bo’lgan ranglar soni.

Quyida amalda keng tarqalgan videoadapterlar sanab o’tilgan:

- **CGA(Color Graphics Adapter) ;**
- **MCGA(Multi Color Graphics Array);**
- **EGA(Enhanced Graphic Adapter);**
- **VGA(Video Graphics Array).**

Komp yuterda qanday adapterning o’rnatilishiga qaramay Turbo-Paskalning prosedura va funksiyalaridan to’liq foydalanish mumkin. Ularni o’zaro sozlashuvi avtomatik tarzda kechadi. Bu sozlashlarni grafik drayverlar deb ataluvchi maxsus programmalar bajaradi. Drayverlar *.BGI kengaytmasi bor bo’lgan fayllarda joylashgan.

Misol uchun, EGA va VGA adapterlari bilan ishlash uchun zarur drayverlar EGAVGA.BGI faylida,CGA va MCGA adapterlariga mos drayverlar esa CGA.BGI faylida joylashgan.

Turli xil drayverlarni ko’rsatish uchun ***Graph*** modulida quyidagi o’zgarmaslar aniqlangan:

```
const
Detect=;{Drayverni avtomatik tarzda aniqlash}
CGA=;
MCGA=;
EGA=;
EGA64=4;
EGAMONO=5;
IBM8514=6;
HERCMONO=7;
ATT400=8;
VGA=9;
PC327=10.
```

Graph modulida grafik rejimlarni ko'rsatish uchun esa quyidagi o'zgarmaslar aniqlangan (CGA,EGA va VGA adapterlari uchun);

```
const
CGAC0=0;        {320x200 ta nuqta, 4 xil rang}
CGAC1=1;        {320x200 ta nuqta, 4 xil rang}
CGAC2=2;        {320x200 ta nuqta, 4 xil rang}
CGAC3=3;        {320x200 ta nuqta, 4 xil rang}
CGAHi=4;        {640x200 ta nuqta, 4 xil rang}
EGALo=0;        {640x200 ta nuqta, 16 xil rang, 4 ta varoq}
EGAHi=1;        {640x350 ta nuqta, 16 xil rang, 2 ta varoq}
EGA64Lo=0;      {640x200 ta nuqta, 16 xil rang,1 ta varoq}
EGA64Hi=1;      {640x350 ta nuqta, 4 xil rang, 4 ta varoq}
VGALo=0;        {640x200 ta nuqta, 16 xil rang, 4 ta varoq}
VGAMed=1;       {640x350 ta nuqta, 16 xil rang, 2 varoq}
VGAHi=2.        {640x480 ta nuqta, 16 xil rang, 1 varoq}
```

2. Displeyni grafik rejimga o'tkazish

Displeyning doimiy rejimi, matnli rejim hisoblanadi. Displeyni grafik rejimiga o'tkazish uchun *Graph* modulining *InitGraph* proseduralidan foydalaniladi:

InitGraph(GD,GM,Path)
bu yerda *GD*-drayver nomeri ;
GM-rejim nomeri ;
Path-kerakli drayver joylashgan fayl yo'li.

GD va *GM* nomerlarni qanday aniqlashni oldingi mavzuda ko'rib chiqdik. Agar *Path* o'zgaruvchisi bo'sh satr qatorini tashkil qilsa (*Path=' '*) drayverni ishchi katalogdan qidiriladi.

GD va *GM* o'zgaruvchi parametrlar hisoblanadi. Agar *InitGraph* prosedurasini ishga tushirishda ***GD=0*** bo'lsa, zarur drayver va bu drayver uchun eng yaxshi rejim avtomatik tarzda aniqlanadi.

Graph modulida qulaylik uchun ***Detect*** o'zgarmasi (*Detect=0*) kiritilgan. Misollarda undan qanday foydalanganligiga ahamiyat bering.

InitGraph prosedurasiga simmetrik prosedura ***CloseGraph*** hisoblanadi. Bu prosedura drayverni xotiradan chiqarib tashlaydi va oldingi videorejimni tiklaydi.

Quyidagi programma displeyni grafik rejimga o'tkazadi va shu zahotiyoq uni asli holiga qaytaradi:

```
uses  Graph;
var
Gd,Gm:Integer;
begin
GD:=detect;{Drayverni avtomatik tarzda aniqlaydi, chunki
            Detectq0}
InitGraph(GD,GM,'d:g'tp');  {Grafik rejimni o'rnatish}
ReadLn;              {Enter bosqichini bosilishini kutish}
CloseGraph;
end.
```

Ba`zi hollarda, masalan .BGI kengaytmali fayl yo'lini xato ko'rsatilsa yoki grafik rejimni o'rnatish uchun tezkor xotira yetishmasa InitGraph prosedurasi o'z ishini muvaffaqiyatli yakunlay olmaydi. Bunday hollarda yo'l qo'yilgan xatoliklarni aniqlash uchun ***GraphResult*** funksiyasidan foydalanish mumkin. Bu funksiya grafik rejimni ishga tushirishga harakat qilganda quyidagi qiymatlardan birini beradi:

 0 - xatolar yo'q;
-2 - grafik plata o'rnatilmagan;
-3 - drayver fayli topilmadi;
-4 – noto'g'ri drayver o'rnatilgan;
-5 - grafik rejimni o'rnatish uchun tezkor xotira yetishmagan;

GraphResult funksiyasidan foydalanishga doir quyidagi programmani ko'rib chiqaylik:

```
uses Graph;
var
GD,GM,ErrCode:integer;
begin
GD:=Detect;       {Drayverni  avtomatik tarzda aniqlash}
InitGraph(GD,GM,'     ');         {Grafik rejimni ishga tushirish}
ErrCode:=GraphResult;
if ErrCode<>0 then WriteLn(ErrCode,'raqamli  xato')
      {Xatoga yo'l qo'yilgan}
```

```
else CloseGraph;
        {Xato topilmagan va grafik rejimni ishi yakunlandi}
end.
```

3. Nuqta, chiziq va ranglar

Biz ekranda grafik rejimni qanday o'rnatishni ko'rib chiqdik. Endi ekranda turli xil nuqtalar, chiziqlar va shakllar chizishni tashkil qilaylik. ***Graph*** modulida 80 taga yakin funksiya va proseduralar mavjud bo'lib, ulardan foydalanib quyidagi ishlarni qilish mumkin:

- nuqtalar qurish;
- kesmalar chizish;
- ellips va aylanalar chizish;
- to'g'ri to'rtburchaklar va ko'pburchaklar chizish;
- yopiq shaklarni turli ranglarga bo'yash, hamda bir necha hil standart va keragicha nostandart usullar bilan sohalarni shtrixlash;
- ekranga turli shriftdagi matnlarni chiqarish;
- ekran sohalarini eslab qolish va ularni surish.

Grafik rejimda ishlash xuddi matematika fanidagi Dekart koordinatalar sistemasida nuqtalar orqali turli hil shakllar yasashga o'xshab ketadi. Ekrandagi har bir nuqta o'zining koordinatalariga ega. Ekranning chapdan eng tepadagi nuqtasining koordinatasi (0,0)ga teng. X koordinatasi chapdan o'ngga o'sib borsa, Y koordinatasi yuqoridan pastga qarab o'sib boradi. Nuqta koordinatasini aniqlovchi (x,y) juftlikdagi birinchi qiymat OX o'qidan, ikkinchi qiymat esa OY o'qidan aniqlanadi. Ekranning o'ngdan eng pastidagi nuqtasi oxirgi nuqta hisoblanadi va uning koordinatasi grafik rejimning turiga bog'liq. Masalan,VGAHi rejimida ekrandagi nuqtalar soni 640x480ga teng. O'ngdan eng pastki nuqta koordinatasi esa (639,479) ga teng bo'ladi.

Endi ***Graph*** modulining eng ko'p ishlatiladigan funksiya va proseduralari bilan tanishib chiqaylik.

1. ***PutPixel***(x,y,color) prosedurasi -(x,y) koordinatali nuqtani ***Color*** parametri bilan aniqlangan rangga bo'yab beradi;

Misol: **PutPixel**(100,120,Red) - ekranda (100,120) koordinatali qizil nuqta paydo bo'ladi.

2. ***GetPixel***(x,y) funksiyasi - (x,y) koordinatali nuqtaning rangini aniqlab beradi;

Misol: Color:qGetPixel(100,120); (100,120) koordinatadagi nuqtaga qo'yilgan rang qiymatini aniqlaydi.

3. ***Line***(x1,y1,x2,y2) prosedurasi -(x1,u1) va (x2,u2) koordinatali nuqtalarni tutashtiruvchi kesma chizadi;

4. ***Circle***(x,y,Radius) prosedurasi - markazi (x,u) nuqtada joylashgan radiusi Radius ga teng aylana chizadi;

5. ***Rectangle***(x1,y1,x2,y2) prosedurasi-chap yuqoridagi burchagi (x1,u1) va o'ngdan pastki burchagi (x2,u2) koordinatali nuqtalar orqali o'tkazilagan to'g'ri to'rtburchak chizadi;

6. ***SetColor***(Color) prosedurasi - chizish rangini o'rnatadi;

Grafik rejimda ranglarni belgilash uchun xuddi matnli rejimdagi kabi quyidagi o'zgarmaslar ishlatiladi:

```
const
Black=0;{Qora}
Blue=1;{Ko'k}
Green=2;{Yashil}
Cyan=3;{Zangori rang}
Red=4;{Qizil}
Magenta=5;{Pushtirang}
Brown=6;{Jigarrang}
LightGray=7;{Och kulrang}
DarkGray=8;{To'q kulrang}
LightBlue=9;{Och havorang}
LightGreen=10;{Och ko'k}
LightCyan=11;{Och zangorirang}
LightRed=12;{Och qizil}
LigthMagenta=13;{Och pushtirang}
Yellow=14;{Sariq}
White=15;{Oq}
```

Misol:

```
uses Graph;
var
GD,GM:integer;
Rang,Radius:word;
begin
GD:=Detect;                {Drayverni  avtomatik tarzda aniqlash}
InitGraph(GD,GM,'  ');     {Grafik rejimni ishga tushirish}
for Rang:=15 downto 0 do
begin
   setcolor(Rang); {Chizish rangini o'rnatish}
   Radius:=Rang*10;
```

```
        Circle(GetMaxX div 2, GetMaxY div 2,Radius);
        {Markazi ekran markazida joylashgan aylana chizish}
end;
Readln;
CloseGraph;
end.
```

4. Turli xil figuralar

Graph modulining prosedura va funksiyalari.Yuqoridagi mavzuda ko'rib chiqilgan funksiya va proseduralar yordamida faqat chiziqlar chizish mumkin. Endi boshqa turli xil ranglar bilan to'ldirilgan figuralar chizishni tashkil etishga yordam beruvchi yana bir nechta prosedura va funksiyalar bilan tanishib chiqamiz.

1.**SetFillStyle**(Style,Color) prosedurasi - Color rangi bilan sohalarni to'ldirish va ularni ko'rsatilgan uslubda to'ldirish (shtrixovka qilish) uchun ishlatiladi;
Sohani turli ranglar bilan to'ldirish o'zgarmaslari:

```
const
EmptyFill=0;{sohani ekran fonining rangiga bo'yaydi}
SolidFill=1;{sohani belgilangan rangda uzluksiz to'ldirish}
LineFill=2;{sohani qalin gorizontal (-----) chiziqlar bilan  to'ldiradi}
LtSlashFill=3;{sohani ingichka"/// " belgilari bilan to'ldiradi}
SlashFill=4;{sohani qalin "/// " belgilari bilan to'ldiradi}
BkSlashFill=5;{sohani qalin "g'g'g'"  belgilari bilan to'ldiradi}
LtBkSlashFill=6;{ sohani "---'"  qatlami bilan to'ldiradi}
HatChFill=7;{sohani to'r bilan to'ldiradi}
XHatChFill=8;{ sohani egri to'r bilan to'ldiradi}
InterLeaveFill=9;{ sohani zich egri shtrixovka bilan to'ldiradi}
WideDotFill=10;{sohani kam uchrovchi nuqtalar bilan to'ldiradi}
CloseDotFill=11;{sohani zich nuqtalar bilan to'ldiradi}
UserFill=12;{sohani dasturchi aniqlagan shtrixovka bilan
to'ldiradi}
```

2.**Bar**(x1,y1,x2,y2) prosedurasi ekrandagi rang va shtrixovka ustiga to'g'ri to'rt burchak quradi;

3.**Bar3D**(x1,y1,x2,y2,Depth,Top) prosedurasi ham shunday rang va shtrixovka bilan to'ldirilgan parallelepiped chizadi. **Depth** o'zgaruvchisi parallelepiped balandligini anglatadi. **Top** mantiqiy o'zgaruvchisi **true** qiymatli bo'lsa parallelipipedning yuqori asosi chiziladi aks xolda, chizilmay ochiq qoladi;

4. **FillEllipse**(x,y,XRadius,YRadius) prosedurasi oldin o'rnatilgan rangga to'ldirilgan ellips chizadi. Ellips o'qlari koordinata o'qlariga paralel deb olinadi. **XRadius** - ellips eni, **YRadius** - ellips balandligi.

5. Grafik rejimdagi matnlar

Grafik rejimda matnlarni yozish uchun ikki xil tipdagi shriftdan foydalanish mumkin: nuqtalar matritsasi va simvolni tashkil etuvchi vektorlar qatori orqali.

Shriftlar fayllari .ChR kengaytmasiga ega bo'ladi va shriftni ishga sozlaganda kerakli fayllar ishchi katalogda yoki .BGI grafik drayveri joylashgan katalogda bo'lishi kerak.

Shriftni tanlash va masshtab o'rnatish **SetTextStyle** prosedurasi yordamida amalga oshiriladi:

SetTextStyle(Font, Direction, Size) - kerakli shriftni o'rnatadi, matnni chiqarish yo'nalishini aniqlaydi va belgilar o'lchovini belgilab beradi. **Font**- shriftni aniqlovchi o'zgaruvchi, **Direction**-matnni chop etish yo'nalishini ko'rsatuvchi o'zgaruvchi (chapdan o'ngga yoki pastdan yukoriga), **Size**-shrift o'lchovini aniqlovchi o'zgaruvchi. Matritsali shriftda o'lchov Sizeq1, vektor shriftida esa Sizeq4 qiymatlarida erishiladi.

Turli xil shriftlarni ko'rsatish va matnlarni chop etish yo'nalishlarini tanlash uchun quyidagi o'zgarmaslar aniqlangan:

```
const
{shriftlar}
DefaultFont=0;{8x8 nuqtali standart matritsali shrift}
TriplexFont=1;{vektorli shrift}
SmallFont=2;{vektorli shrift}
SansSerifFont=3;{vektorli shrift}
GothicFont=4;{vektorli shrift}
{matn yo'nalishi}
HorizDir=0;{chapdan o'ngga}
VertDir=1;{pastdan yuqoriga}
```

OutTextXY(x,y,TextString) prosedurasi - oldindan aniqlangan shriftda, yo'nalishda va belgi o'lchovida ***TextString*** qatorini (x,u) nuqtadan boshlab chop etadi.

SetTextJustify(Horiz,Vert) prosedurasi *OutTextXY* prosedurasi chop etadigan matnni avtomatik tarzda tekislab beradi. ***Horiz*** - gorizontal, ***Vert*** - vertikal tekislashlar.

Matnlarni tekislash uchun quyidagi o'zgarmaslar aniqlangan:

```
const
{gorizontal tekislash uchun}
LeftText=0;{chap tomonga nisbatan tekislash}
CenterText=1;{markazga nisbatan tekislash }
RightText=2;{o'ng tomonga nisbatan tekislash}
```

```
{vertikal tekislash uchun}
BottomText=0;{pastgi tomonga nisbatan tekislash}
CenterText=1;{markazga  nisbatan tekislash}
TopText=2;{yuqori tomonga nisbatan tekislash}
```

6. Ekran sohalari

GetImage, ***PutImage*** proseduralari va ***ImageSize*** funksiyasi yordamida tasvirlarning to'g'ri to'rtburchakli sohalarini hotirada eslab qolish va ularni ekranga chiqarishimiz mumkin.

1.***ImageSize***(x1,y1,x2,y2) funksiyasi – ekranning to'g'ri to'rtburchak-li sohasini saqlash uchun zarur bo'lgan xotira o'lchovini (baytlarda) beradi.(x1,y1) to'g'ri to'rtburchakli ko'rinishning chapdan yuqoridagi, (x2,y2) - esa pastdan o'ngdagi burchak nuqtalari uchun koordinatalar.

2.***GetImage***(x1,y1,x2,y2,Area) prosedurasi xotiraning ***Area*** sohasida to'g'ri to'rtburchakli ekran tasvirini saqlaydi. (x1,y1) va (x2,y2) lar yuqoridagi ma`noda qayta ishlatilmoqda.

3.***PutImage***(x,y,Area,Mode) prosedurasi ekranning ko'rsatilgan joyiga tasvir ko'rinishini chop etadi. (x,y) – xotiraning ***Area*** sohasidagi tasvir ko'rinishi nusxasini chop etiladigan, ekranning chapdan yuqoridagi nuqtasining koordinatasi. ***Mode***-tasvirni ekranga chiqarish rejimi.

Tasvirlarni ekranga chiqarish rejimini aniqlash uchun foydalani-ladigan o'zgarmaslar:

```
const
{PutImage prosedurasi uchun o'zgarmaslar}
NormalPut=0;{ mavjud tasvirni almashtirish}
XorPut=1;{XOR mantiqiy amali}
OrPut=2;{OR mantiqiy amali}
AndPut=3;{AND mantiqiy ko'paytirish amali}
NotPut=4;{NOT mantiqiy rad etmoq amali}
```

7. Xatolar tahlili

Grafik rejimni o'rnatish mavzusida yo'l qo'yilgan xatolar diagnostikasi uchun ***GraphResult*** funksiyasidan foydalangan edik. Hozir shu funksiya beradigan xatolar kodi bilan to'liqroq tanishib chiqaylik.

Quyida ***GraphResult*** funksiyasi beradigan kodlarga mos o'zgarmaslar ro'yhati keltirilgan:

```
const
```

grOk=0;{xatolar yo'q}
grNoInitGraph=-1;{Grafik rejim o'rnatilmagan(InitGraph prosedura-sini ishga tushiring)}
grNotDetected=-2; {Grafik plata o'rnatilmagan}

-6 - sohalarni ko'chirishda xotira chegarasidan chiqish;
-7 - zarur sohani bo'yash paytida xotira chegarasidan chiqish;
-8 - shrift fayli topilmagan;
-9 - shrift faylini ishga tushirish uchun xotira yetishmayapti;
-10 - tanlangan drayver uchun noto'g'ri grafik rejim.

8. Graph modulining prosedura va funksiyalari

1. ***Arc*** prosedurasi - aylana yoyini chizadi.
Aniqlanishi : Arc(x,y : integer; StAng, EndAng, Radius: Word);
x,y - aylana markazining koordinatasi;
StAng, EndAng - mos ravishda yoyning boshlang'ich va oxirgi burchaklari;
Radius-aylana radiusi.

2.***Bar*** prosedurasi - rangga bo'yalgan to'g'ri to'rtburchak chizadi.
Aniqlanish: Bar(x1,y1,x2,y2:integer);
(x1,u1) va (x2,u2) mos ravishda to'g'ri to'rtburchakning chetki nuqtalari koordinatalari.

3.***Bar3D*** prosedurasi rangga bo'yalgan parallelipiped chizadi.
Aniqlanishi :Bar3D(x1,y1,x2,y2:integer;Depth:word;Top:boolean);
(x1,y1) va (x2,y2) asosni tashkil etuvchi to'g'ri to'rtburchak uchlarining koordinatalari;
Depth -parallelipiped chuqurligi;
Top- mantiqiy o'zgaruvchi.

4.***Circle*** prosedurasi - aylana chizadi;
Aniqlanishi: Circle(x,y:integer;Radius:word);
(x,y) aylana markazining koordinatasi;
Radius-aylana radiusi.

5. ***CloseGraph*** prosedurasi grafik rejimini uzadi.
Aniqlanishi :Closegraph;(parametrsiz prosedura)

6.***DrawPoly*** prosedurasi - ko'p burchak chizadi.

Aniqlanishi :DrawPoly(NumPoints:word; var PolyPoints);
NumPoints - ko'pburchak tomonlari soni;
PolyPoints - ko'pburchak uchlarining koordinatalaridan tuzilgan massiv.

7.***Ellipse*** prosedurasi - ellips yoyini chizadi.
Aniqlanishi: Ellipse(x,y:integer;StAng,EndAng:word;XRadius,YRadius:word);
(x,y) – ellips markazning koordinatasi;
StAng va EndAng - yoyning boshlang'ich va oxirgi burchaklari;
Xradiues va Yradius mos ravishda ellips balandligi va eni.

8.***FillPoly*** prosedurasi - rangli ko'pburchak chizadi.
Aniqlanishi: FillPoly(NumPoints:word; var PolyPoints);
NumPoints - ko'pburchakning uchlari soni;
PolyPoints - ko'pburchak uchlari koordinatalaridan tuzilgan massiv.

9.***GetArcCoords*** prosedurasi - oxirgi marta ishlatilgan ***Arc*** prosedurasining koordinatalarini aniqlaydi.
Aniqlanishi: GetArcCoords(var ArcCoords:ArcCoords Type);

10.***GetColor*** funksiyasi - ekran rangini aniqlaydi.
Aniqlanishi: GetColor:word;

11.***GetGraphMode*** funksiyasi - grafik ekranni qaytaradi.
Aniqlanishi: GetGraphMode:integer;

12.***GetImage*** prosedurasi - ekranning berilgan sohasini Area da saqlaydi.
Aniqlanishi: GetImage(x1,y1,x2,y2:integer;var Area);

13.***GetMaxColor*** funksiyasi - rangning eng katta qiymatini hisoblaydi.
Aniqlanishi: GetMaxColor:word;

14.***GetPixel*** funksiyasi - berilgan nuqta rangini aniqlaydi.
Aniqlanishi: GetPixel(x,y:integer):word;

15. ***GraphErrorMsg*** funksiyasi - berilgan kod bo'yicha xato haqida satr ma`lumot beradi.
Aniqlanishi: GraphErrorMsg(Code:integer):string;

16. ***LineTo*** prosedurasi - oldingi aniqlangan nuqtadan berilgan nuqtagacha kesma chizadi.
Aniqlanishi: LineTo(x,y:integer);

17. ***PieSlice*** prosedurasi sektor chizadi.

Aniqlanishi: PieSlice(x,y:integer;StAng,EndAng,Raduis:word);

Nazariy savollar va tayanch iboralar:

1. Grafik rejimda ekran xolatini to'liq ifodalang;
2. Graph modulida drayverlarni aniqlash o'zgarmaslarini sanab bering;
3. Grafik rejimni o'rnatish o'zgarmaslarini sanab bering;
4. Grafik rejimda o'tish proseduraining ishlash printspini tushuntiring;
5. Grafik rejimda yo'l qo'yilgan xatoliklarni qaysi funksiya yordamida aniqlash mumkin?
6. Graph modulining nechta prosedura va funksiyalari mavjud?
7. Ekranda nuqta va to'qri chiziq yasash, ularga rang tanlash proseduralari va funksiyalarini aytib bering;
8. Ranglarni aniqlash o'zgarmaslarini sanab bering;
9. Ekranda nuqtadan va kesmadan tashkil topgan tasvirlar yasang.
10. Turli xil ranglar bilan to'ldirilgan figuralar chizish uchun ishlatiladigan prosedura va funksiyalarni sanab o'ting, ularni vazifalarini tushuntirib bering;
11. Grafik rejimda matnlarni yozish uchun nacha xil tipdagi shiriftdan foydalanish mumkin?
12. Shiriftni tanlash va masshtab o'rnatish qaysi prosedura yordamida amalga oshiriladi?
13. Shiriftlarni ko'rsatish va matnlarni chop etish uchun yo'nlaishlar tanlashda ishlatiladigan o'zgarmaslarni sanab bering;
14. Matnlarni tekslash uchun ishlatiladigan o'zgarmaslarni sanab bering;
15. Tasvirlarning to'g'ri to'rtburchakli sohalarini hotirada eslab qolish va ularni ekranga chiqarish prosedura va funksiyalarining vazifalarini aniqlab bering;
16. Tasvirlarni ekranga chiqarish rejimini aniqlash uchun ishlatiladigan o'zgarmaslarni ko'rsatib bering;
17. GraphResult funksiyasi beradigan kodlarga mos o'zgarmaslarni sanab o'ting va hatolarni taxlil qilib bering;
18. Graph modulining asosiy prosedura va funksiyalarini aytib bering.

17-ma`ruza. Faylli tiplar.

Reja:

1. *Faylli tiplarni hosil qilish;*
2. *Faylli tiplar bilan ishlash proseduralari va ularning vazifalari.*

1. Faylli tiplarni hosil qilish

Faylli tipdagi o'zgaruvchilarni diskdan ma`lumot o'qib oluvchi yoki diskka ma`lumot yozib qo'yuvchi programmalarda ishlatish mumkin. Faylli tipdagi o'zgaruvchilarni e`lon qilishda ***file*** va ***text*** xizmatchi so'zlari ishlatiladi:

```
var   mfile 1, mfile 2: file;
      afile: file;
      Prima: text;
```

text xizmatchi so'zi faylning matnli ekanligini anglatadi. Matnli fayllar maxsus belgilar bilan ajratilgan, uzunligi noma`lum bo'lgan qatorlardan tashkil topadi.

Ayrim paytlarda fayllarni bir xil tipli hadlar ketma-ketligi ko'rinishida qarash qulayrok bo'ladi. Bu ketma-ketlik qatorlar, butun sonlar yoki yozuvlardan tashkil topishi ham mumkin:

```
var   A1: file  of  byte;
      {A1 fayli baytlar ketma - ketligidan tashkil topgan}
      A2: file  of  integer;
      {A2 fayli butun sonlar ketma-ketligidan tashkil topgan}
      A3: file  of  string;
      {A3 fayli katorlar ketma-ketligidan tashkil topgan}
      A4: file of  string[20];
      {A4 fayli 20ta belgili qatorlarning ketma-ketligidan tashkil topgan}
      A5: text;
      {A5 fayli matnli fayl hisoblanadi}
```

Agar faylning hadlari uchun tip aniqlangan bo'lsa, bunday fayllarni tiplashtirilgan, aks holda tiplashtirilmagan deb ataladi:

```
var   A: file ; { tiplashtirilmagan fayl}
      B: file of char; { tiplashtirilgan fayl}
```

Fayllar bilan ishlaydigan quyidagi programmani ko'rib chiqaylik.

```
Var
      mydata: file of integer;
      i, j, sum: integer;
begin
      assign (mudata, 'd:g'tpg'myfile.dat');
                  {mydata fayl o'zgaruvchisi bilan faylning
                    ismini myfile.dat va uning aniq yo'li
                   aniqlanmoqda}
      rewrite (mydata);              {fayl yozuv uchun ochiq}
      writeln ('Salom noma`lum o'rtoq...');
      writeln ('Birinchi sonni kiriting');
      readln (i);
      writeln ('Kiritilgan sonni diskdagi myfile.dat
                  fayliga yozilmokda');
      write (mydata, i);   {bu operator yordamida diskdagi myfile.dat
                           fayliga i sonini yoziladi}
      writeln ('Ikkinchi sonni kiriting');
      readln (j);
      writeln ('Kiritilgan ikkinchi sonni diskdagi myfile.dat
                  fayliga yozilmoqda');
      write (mydata, j);             {Diskka yozish bajarilmoqda}
```

```
        sum := i + j;
        writeln ('Yig'indi =', sum);
        writeln ('Yig'indi diskdagi myfile.dat
                    fayliga yozilmokda');
        write (mydata, sum);          {Diskka yozish bajarilmoqda}
        close (mydata);                        {mydata fayli yopildi}
        writeln ('Xayr noma`lum o'rtoq...');
end.
```

2. Faylli tiplar bilan ishlash proseduralari va ularning vazifalari

E`tiboringizga havola etilgan programmada ***Assign, Rewrite, Write*** va ***Close*** proseduralaridan foydalanildi. Endi shu proseduralarning va keyingi programmada ishlatiluvchi ***Reset*** va ***Read*** proseduralarning vazifalari va qanday aniqlanganligi haqida qisqacha ma`lumot berib o'taylik:

Assign prosedurasi.

Vazifasi: Faylli o'zgaruvchiga tashqi fayl ismini o'zlashtiradi.

Aniqlanishi: Assign (f; name: string);

bu yerda f - ixtiyoriy tipli faylli o'zgaruvchi;

name - qatorli tipdagi ifoda yoki qator, fayl ismi (agar faylning to'liq yo'li ko'rsatilmagan bo'lsa fayl ishlanayotgan katalogda joylashgan bo'ladi).

Close prosedurasi.

Vazifasi: ochiq faylni yopadi.

Aniqlanishi: Close (f);

bu yerda f - oldindan ochilgan faylga mos keluvchi faylli o'zgaruvchi.

Read prosedurasi.

Vazifasi: fayl hadini o'zgaruvchiga o'qiydi.

Aniqlanishi: Read (f, v);

bu yerda f - faylning ixtiyoriy tipiga mos faylli o'zgaruvchi (faqat matnli tipli emas);

v - fayl hadining tipi bilan bir xil tipli o'zgaruvchi.

Reset prosedurasi.

Vazifasi: mavjud faylni ochadi.

Aniqlanishi: Reset (f: file);

bu yerda f – faylning ixtiyoriy tipiga mos faylli o'zgaruvchi va bu o'zgaruvchi fayl bilan ***Assign*** prosedurasi orqali bog'langan bo'lishi kerak. ***Reset*** prosedurasi mazkur faylni ochadi.

Rewrite prosedurasi.

Vazifasi: yangi faylni yaratadi va ochadi.

Aniqlanishi: Rewrite (f: file);
bu yerda f – ixtiyoriy faylli tipdagi faylli o'zgaruvchi. ***Rewrite*** prosedurasini ishlatishdan oldin f o'zgaruvchi ***Assign*** prosedurasi yordamida diskdagi fayl bilan bog'lanishi kerak. ***Rewrite*** prosedurasi yangi fayl tashkil qiladi.

Write prosedurasi.

Vazifasi: fayl hadiga o'zgaruvchini yozib qo'yadi.
Aniqlanishi: Write (f, v);
bu yerda f – faylli o'zgaruvchi;
v - f faylining hadi bilan bir xil tipli o'zgaruvchi.

Oldingi tuzgan programmamiz «d:» diskdagi tp katalogida myfile.dat faylini tashkil qildi. Endi shu fayldan qanday qilib ma`lumotlarni o'qishni ko'rib chiqaylik.

```
Var
    mydata: file of integer;
    i, j, sum: integer;
begin
    assign (mydata, 'd:g'tpg'myfile.dat');
    reset (mydata);                    {fayl o'qish uchun ochilmoqda}
    writeln ('Salom noma`lum o'rtoq...');
    read (mydata, i);
    writeln ('myfile.dat faylidan birinchi son o'qildi');
    read (mydata, j);
    writeln ('diskdagi myfile.dat faylidan
                ikkinchi son o'qildi');
    read (mydata, sum);
    writeln ('myfile.dat faylidan uchinchi son o'qildi');
    close (mydata);          {mydata fayli yopiladi}
    writeln ('Xayr noma`lum o'rtoq...');
end.
```

Text standart faylli tip matnli fayllarni aniqlaydi. Matnli fayllar o'zaro yangi qatorga o'tish belgilari bilan ajratilgan qatorlardan tashkil topadi.

Matnli fayllar bilan ishlash uchun maxsus kiritish (***Readln***) chop etish (***Writeln***) proseduralari ko'zda tutilgan. Bu proseduralar uzunligi noma`lum katorlarni fayllardan o'qish va fayllarga yozish uchun ishlatiladi.

Endi matnli fayllar bilan ishlashga doir quyidagi programma bilan tanishib chiqaylik:

```
var
    mytext: text;
    s: string;
begin
    assign (mytext, 'd:g'tpg'mytext.txt');
```

```
                    {mytext faylli o'zgaruvchi orkali fayl
                          ismi va yo'li aniqlanmoqda}
        rewrite (mytext);
                    {fayl yozish uchun ochiq}
        writeln ('Sizning ismingiz?');
        readln (s);
        writeln ('Ismingizni diskdagi mytext.txt fayliga yozilmoqda');
        writeln (mytext, s);
              {s - qatori mytext.txt fayliga yozilmoqda}
        close ( mytext);
              {mytext fayli yopildi}
end.
```

Nazariy savollar va tayanch iboarlar:

1. Faylli tipdagi o'zgaruvchilarni e`lon qilishda qaysi hizmatchi so'zlardan foydalaniladi?
2. Tiplashtirilgan fayllarga ta`rif bering;
3. Tiplashtirilmagan fayllarga ta`rif bering;
4. Fayllar bilan ishlash programmalaridan na`munalar keltiring;
5. Faylli tiplar bilan ishlovchi proseduralarning vazifalarini tushuntiring;
6. Faylni tashkil etish va undan ma`lumot o'qish programmalarini tuzing;
7. Matnli fayllar bilan ishlash programmalaridan na`munalar keltiring.

18-ma`ruza: Xotiraning dinamik sohasi va ko'rsatkichlar.

Reja:

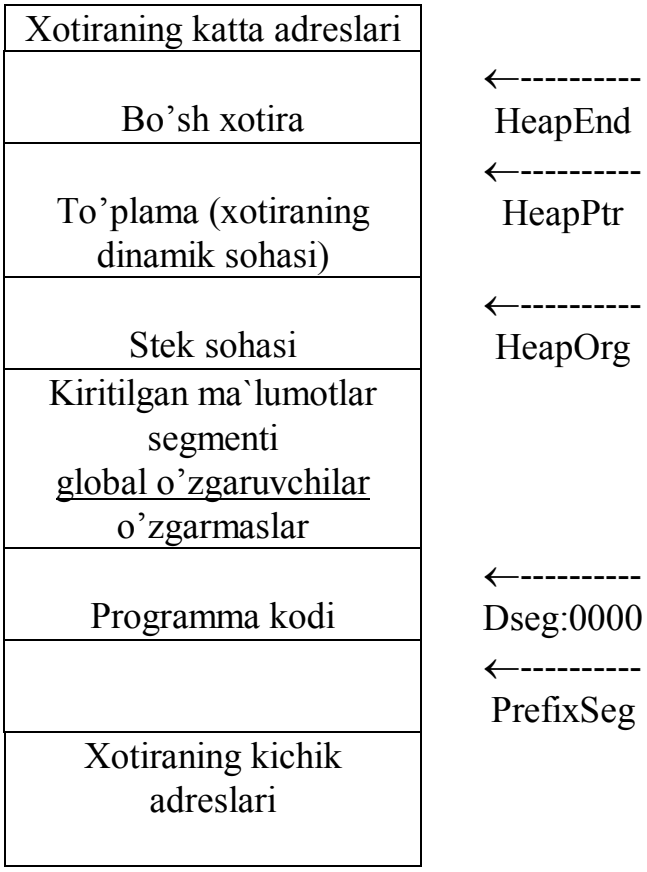
1. *Xotiraning dinamik soxasi;*
2. *Ko'rsatkichlar xaqida boshlang'ich ma`lumotlar.*

1. Xotiraning dinamik soxasi

Programmaning (tashqi) o'zgaruvchilari komp yuter xotirasining va o'zgarmaslar ma`lumotlar segmentida joylashgan bo'ladi. Segment uzunligi esa 64 Kb ni tashkil qiladi. Demak, shunday hollar bo'lishi mumkinki, programmaning barcha o'zgaruvchilarini ma`lumotlar segmentiga joylab bo'lmaydi, ya`ni ma`lumotlar segmentga sig'maydi. Masalan, kuyidagi programmani kompilyatsiya qilishga urinish natijasiz tugaydi, ya`ni ekranda "***Error 49: Data Segment too large***" (berilgan ma`lumotlar segmenti juda katta) yo'l qo'yilgan xato haqida ma`lumot chiqadi:

```
var
        A: array [1..40000] of  byte;
        B: array [1..25534]  of  byte;
begin
end.
```

Shuning uchun, katta xajmli ma`lumotlar bilan ishlashga to'g'ri kelganda bu ma`lumotlarni xotiraning dinamik sohasiga yozish maqsadga muvofiqdir. Xotiraning dinamik sohasi (to'plama deb ham atashadi) programma egallagan deyarli barcha xotira hisoblanadi. Faqat bu sohaga ma`lumotlar kodlari, stek va ma`lumotlar segmenti xotiralari kirmaydi. Odatda programma egallagan xotira ko'rinishi quyidagicha bo'ladi:



To'plamaning boshi va oxirini ***System*** modulining ***Heaporg*** va ***HeapEnd*** maxsus o'zgaruvchilari ko'rsatib turadi. Zarur paytdagi to'plama sohasining tugash joyini ***HeapPtr*** o'zgaruvchisi aniqlab beradi. To'plamaning o'lchovi kiritilgan ma`lumotlar segmentining o'lchovidan bir necha marta katta bo'lib, taxminan 500 kb (agar komp yuterning umumiy xotirasi 640 Kb bo'lsa) joy egallashi mumkin.

Ma`lumotlar segmentida joylashgan ma`lumotlar statik ma`lumotlar deb ataladi. Bunga asosiy sabab, ular uchun xotira programmani kompilyatsiya qilish davomida ajratib qo'yiladi. Programmani ishlashi davomida esa bu xotira o'zgarmay koladi. To'plamadagi xotira esa programmani ishlashi davomida to'ldirib boriladi va zarur bo'lgan paytda bu xotira bo'shatib qo'yilishi mumkin. Shuning uchun, to'plamada joylashgan o'zgaruvchilarni dinamik o'zgaruvchilar deb ataladi. To'plamadan xotira ajratishni ***GetMem*** va ***New*** proseduralari orqali amalga oshiriladi, uni bo'shatib qo'yishni esa ***FreeMem*** va ***Dispose*** proseduralari bajaradi.

2. Ko'rsatkichlar xaqida boshlang'ich ma`lumotlar

Ko'rsatkich shunday o'zgaruvchiki, u ma`lumotning qiymatini emas, balki ularning xotiradagi joylashgan adresini o'zida saqlaydi. Ko'rsatkichlarning ikki xil tipi mavjud: tiplangan va tiplanmagan. Tiplanmagan ko'rsatkichlarning e`lon qilish uchun ***pointer*** xizmatchi so'zidan foydalaniladi, tiplangan ko'rsatkichlarni e`lon qilish uchun esa ko'rsatilgan tip oldidan **"^"** belgisi qo'yiladi.

Misol:

```
var
      p: pointer; {tiplanmagan ko'rsatkich}
      pint: ^ integer ; {pint - tiplangan, butun tipli ko'rsatkich}
      x, y: ^ byte; {x, y - bayt tipidagi ko'rsatkich}
      pst: ^ string ; {pst - qator tipidagi ko'rsatkich}
```

Adresi ko'rsatkichda joylashgan zarur ma`lumotga ko'rsatkich qilish uchun ko'rsatkich ismidan keyin "^" belgisini yozish kerak:

```
GetImage (0, 0, 100, 100, p^); {grafik ekran sohasining
                                ma`lumotlari r da ko'rsatilgan
                               adresdagi xotira sohasiga yoziladi}
pint^ :=2; {pint  adresi bo'yicha 2 butun soni yoziladi}
x^ := # 12; y^ :=$2;      {x^ baytiga # 12 qiymatini o'zlashtiri-ladi,  y^
                           baytiga esa $ 2}
pst^ := ' Axmad'; {pst  qatoriga 'Ahmad' so'zini o'zlashtiriladi}
```

Ko'rsatkichni ishlatish uchun uni faqat e`lon qilibgina qolmay, uning qanday joyni ko'rsatayotganligiga ham katta ahamiyat berish zarur. Ya`ni, ko'rsatkich operativ xotiraning kerakli ma`lumotlar joylashgan adresini aniq ko'rsatishi kerak. Ko'rsatkich adresini xato ko'rsatish kutilmagan natijalar olishga yoki butunlay programma ishini chippakka chiqarishga olib kelishi mumkin. Shuning uchun, Paskal tilida maxsus proseduralar mavjudki, ular yordamida xotira adresi aniq hisoblanadi.

1. ***GetMem*** prosedurasi (tiplanmagan kursatkichlar uchun) - tezkor xotiradan zarur miqdorda joy ajratadi va uning adresini ko'rsatkichga joylashtiradi.

Aniqlanish: **GetMem (var p: Pointer; size : word);**

p - ko'rsatkich, size - ajratilgan xotiraning bayt o'lchovidagi miqdori.

Misol:

```
var
      p: pointer;
begin
    GetMem (p, 1000); {1000 bayt uzunlikdagi xotira
```

```
                                    sohasi ajratildi}
                                    {p^  sohadan foydalanish bloki}
                FreeMem (p, 1000); {r adresdagi 1000 bayt uzunlikli
                                    xotira sohasi bo'shatib qo'yildi}
        end.
```

2. ***New*** prosedurasi (tiplangan ko'rsatkichlar uchun) - xotiradan zarur joyni band qilib, uning adresini berilgan ko'rsatkichga jo'natadi.

Aniqlanishi: New (var p: pointer);

r - berilgan ko'rsatkich.

New va *GetMem* proseduralari nisbatan juda katta hisoblanmish xotiraning dinamik sohasidan zarur miqdordagi xotira so'rab oladi. Shu xotirani band qilib, uning adresini mos ko'rsatkichga jo'natadi. Shunday hollar ham bo'ladiki, xotiraning dinamik sohasida bo'sh joy qolmaydi, u holda shu xotiraga yuqoridagi proseduralar bilan qilingan ko'rsatkich quyidagi ma`lumotli xatoni yuzaga keltiradi:

Runtime Error 203 ...

Shuning uchun, xotiraning dinamik sohasini vaqti-vaqti bilan bo'shatib turish kerak. Bu ishlarni ***Dispose*** va ***FreeMem*** proseduralari bajaradi.

1. ***Dispose*** prosedurasi - tiplangan ko'rsatkichdagi adresda joylashgan xotirani bo'shatadi.

   Aniqlanishi: Dispose (var p: pointer);

2. ***FreeMem*** prosedurasi - tiplanmagan ko'rsatkichda ko'rsatilgan adresdagi xotirani bo'shatadi.

   Aniqlanishi: FreeMem (var p: pointer; size: word);

Ko'rsatkichni vaqtincha ishlamay turishini bildirish uchun ***Nil*** xizmatchi so'zidan foydalanish mumkin.

Bu proseduralar bilan bir qatorda quyidagi funksiyalar bilan ùam tanishib chiqaylik:

1. ***MemAvail*** funksiyasi - xotiraning dinamik sohasidagi bo'sh sohalarning umumiy xajmini aniqlaydi.

   Aniqlanishi: MemAvail : LongInt;

2. ***MaxAvail*** funksiyasi - xotiraning dinamik sohasidagi eng katta soha o'lchovini aniqlaydi.

   Aniqlanishi: MaxAvail: LongInt;

Misol sifatida GetMem va FreeMem proseduralaridan foydalanilgan quyidagi programmani ko'rib chiqaylik:

```
Var
        P: Pointer;
Begin
        GetMem(P,1024);  {Xotiradan 1024 bayt joy ajratildi,
        .....                    R shu sohani ko'rsatmoqda}
        FreeMem(P,1024); {R sohadagi  1024 bayt joy bo'shatildi}
End.
```

New va Dispose proseduralari esa quyidagicha ishlatiladi:

```
Var
        PS:^String;
Begin
        New(PS);  {Xotiradan 256 bayt joy ajratildi }
        PS^:='Dinamik xotira';
        Writeln(PS^);
        Dispose(PS); {RS sohasigi  tegishli bo'lgan joy bo'shatildi}
End.
```

Nazariy va amaliy savollar:

1. Kiritilgan ma`lumotlar xotiraning qaysi qismiga yoziladi?
2. Programma egallaydigan hotira ko'rinishini sxematik tarzda ifodalang;
3. To'plama o'lchovi komp yuter umumiy hotirasining qancha qismini tashkil qiladi?
4. Statik ma`lumotlar qaerda joylashadi?
5. Dinamik o'zgaruvchilar qaerda joylashadi?
6. Ko'rsatkichlar nimani anglatadi?
7. Ko'rsatkichlarni e`lon qilish uchun qaysi hizmatchi so'zdan foydalaniladi?
8. Ko'rsatkichli o'zgaruvchidan keyin qanday belgi qo'yiladi?
9. Qanday prosedura yordamida hotira adresi aniqlanadi?
10. Dispose va FreeMem proseduralar qanday vazifani bajaradi?
11. Ko'rsatkichni vaqtincha ishlatmay turish uchun qaysi hizmatchi so'zdan foydalaniladi?

Foydalanilgan adabiyotlar ro’yxati.

1. Вирт Н. Алгоритмы структуры данных программы. - М.: Мир, 1985 г.
2. Абрамов В., Трифонов Н., Трифонова Г . Введение в язык Паскал. - М.: Наука, 1988 г.
3. Файсман А. Профессиональное программирование на Турбо Паскале. - Т.: "Инфомэкс Корпорейшн", 1992 г.
4. Findlay W., Watt D.A., Paskal. An introduction to methodical programmin. Third edition. – London: Pitman, 1985 г.
5. Поляков Д. Б., Круглов И. Ю. Программирование в среде Турбо-Паскаль (версия 5.5): Справ.-метод. пособие. – М.: Изд-во МАИ. 1992 г.
6. Епанешников А., Епанешников В. Программирование в среде Turbo Pascal 7.0 М., Диалог-наука, 1993 г.
7. Епанешников А., Епанешников В. Delphi 5. Язик Object Pascal. – М.: Диалог-МИФИ, 2000 г.
8. Культин Н. Программирование в Turbo Pascal 7.0 и Delphi, 2-е издание, С. Петербург, БХВ-С. Петербург, 1999 г.
9. Миллий исти=лол \ояси: асосий тушунча ва тамойиллар, Т. – Янги аср авлоди, 2001 й.
10. www.dialog@bitex.ru, www.frolov.ru