M. ASHUROV
N. XAYTULLAYEVA
SH. SATTOROVA

DASTURLASH TILLARI

O’quv qo’llanma

MUNDARIJA

1-bob. OB’EKTGA YO‘NALTIRILGAN DASTURLASH TILLARI

1.1. “DASTURLASH TILLARI” FANIGA KIRISH.

Reja:

1. Dasturlash tillari va ularning klassifikasiyasi.
2. Mashinaga mo‘ljallangan va proseduraga mo‘ljallangan dasturlash tillari.
3. Yuqori darjali dasturlash tillari.
4. Interpretatorlar va kompilyatorlar.
5. Dasturlarni translyasiyalash.
6. Muyyan dasturlash tilining alifbosi, buruqlar tizimi va operatorlari.

Tayanch iboralar: Dasturlash; dasturchi; Dasturlash tillari turlari; assembler tili; fortran tili; beysik tili; quyi darajadagi DT; o’rta darajadagi DT; yuqori darajadagi DT;Kompilyatsiya va interpretatsiya qilinuvchi tillar; Komp’yuterda masala echish bosqichlari.

Tezkor elektron xisoblash mashinalarining paydo bo’lishi ***dasturlash tili*** deb ataluvchi turli-tuman belgilar sistemalarining paydo bo’lishiga olib keldi. SHunday qilib, hisoblash mashinalarida bajarilishi kerak bo’lgan jarayonlarni tavsiflash uchun qo’llaniladigan belgilar (simvollar) sistemasini ***dasturlash tili*** deb yuritamiz.

EHMlar uchun dastur tuzish va uni ishlatish juda murakkab va ko’p mehnat talab etadigan jarayon bo’lib, uning uchun ko’p aqliy mexnat va vaqt talab qilinadi. Shuning uchun yangi algoritmik tillarni yaratuvchilar dasturlashni sodda va xayotimizning turli sohalarida mexnat qiluvchilar uchun tushunarli bo’lishiga xarakat qilishadi. Hozirgi kunda dasturlash tillarining bir necha turi mavjud:

Bundan ko’rinadiki dasturlash tillarining soni bir necha yuzdan ortiq ekanligi.

Beysik dasturlash tili 1964 y. AQSh ning Dortmunt kollejining ilmiy xodimlari Jon Kemeni va Tomes Kurts tomonidan yaratilgan. Bu til turli hisoblashlarga doir masalalarni kompyuter bilan muloqat holda xal qilish uchun yaratildi.

Basic so’zi Beginners Allpurpose Simbolic Instruction Code –degan kengaytmasidan iborat bo’lib, ya’ni boshlovchilar uchun mo’ljallangan ko’p maqsadli, belgili ko’rsatmalar tili degan ma’noni bildiradi. Ushbu til kompyuter xotirasiga

qo’yiladigan talablarning juda kamligi sababli, ShK larda ishlatiladigan til bo’lib qoldi. Bu tilning bir necha ko’rinishi mavjud bo’lib, maqsad foydalanuvchilarning muloqotida osonlik tug’dirish.

Assembler tili inglizchasiga Assembler- bu EHM uchun mashina kodlarida dastur yozish ishini yengillashtirish maqsadida 40 yillar ohiri 50 yillarning boshida yaratilgan.

Fortran tili –inglizcha Formula Translator – formulani translyatsiya qilish so’zlaridan qisqartirib olingan bo’lib, formula tarjimoni degan ma’noni bildiradi va bu til muhandislik, ilmiy-texnik masalalarni yechishga mo’ljallangan til hisoblanadi. Ushbu tilning qator variantlari yaratilgan, ulardan eng mashhurlari Fortran II va Fortran IV. Fortran tilining asosiy g’oyalari keyingi Algol-60, 68, PL-1 va boshqa tillarda rivojlantirilgan. Bu til 1954 yil yaratilgan.

Shunday maqsadni 1970 yili Shvesiyalik Oliy texnika o’quv yurtining professori Niklaus Virt o’z oldiga qo’ydi va u tomonidan tavsiya etilgan algoritmik tilni yaratdi va ulug’ fransuz olimi B. Paskal (1623-1662) nomi bilan atadi.

Dasturlash tillarining sintaktik jihatdan turlari

Dasturlash tillarining sintaktik jihatdan turlari 3 turga bo‘linadi:

DTning sintaktik jihatdan turlari		
Quyi	O'rta	Yuqori

Quyi darajadagi dasturlash tili “Mashina tili” deb ham ataladi. Ushbu tilda dasturlar to‘g‘ridan-to‘g‘ri Operativ Xotira(OX) katakchalari va protsessor reyestrlari bilan ishlab tuziladi. Ushbu tildagi buyruqlar Markaziy Protsessor(MP)ning operatsiyalariga to‘g‘ri keladi. Buyruqlar ikkilik kodda yozilgan:

```
1: 10110000 01100001
2: (AL reyestriga 16-lik 61 sonini joylash)
```

Bir paytlar perfokartalar yordamida aynan mashina tilida dasturlar yozilgan.

O‘rta daraja dasturlash tillarida protsessor buyruqlarini mnemonik kodlarga(buyruqqa mos qisqartirilgan so‘zlar) almashtirilgan. Assembler tili bunga misoldir. O‘rta darajadagi dasturlash tillarida ham bir protsessor operatsiya deyarli bir

buyruqqa mos keladi. Masalan, yuqoridagi mashina kodi Assemblerda quyidagicha yoziladi:

```
1: MOV AL, 61h
2: (Yuqoridagidan ko‘ra tushunarliroq)
```

Ko‘rib turganingizdek, Assembler tili mashina kodidan bir pog‘ona yuqorida turadi xolos.

ESDA TUTING! Keyingi ikki tilda tuzilgan dasturlar ham har doim birinchi ko`rinishga (obyekt kodi) keltiriladi.

Yuqori darajadagi dasturlash tillari esa, asosan, dasturlash jarayonini tezlashtirish uchun yaratilgan. Shuni eslatib o‘tish lozimki, har qanday dastur bajarilishidan oldin mashina kodiga o‘tkaziladi. Ushbu darajadagi dasturlash tillarida yozilgan dastur ma’lum ma’noli so‘zlardan(odatda ingliz tilidagi) tashkil topadi.

Masalan:

```
1: if (var == 6) {
2: print "Salom";
3: }
```

Ko‘rib turibsizki, dastur qismi ingliz tilidagi ma’noli so‘zlardan tashkil topgan. Hozirgi zamonaviy tillarning barchasi yuqori darajaga mansub.

Kompyuter dunyosida ko‘plab dasturlash tillari mavjud bo‘lib, dasturlash va unga qiziquvchilar soni ortib bormoqda. Bir xil turdagi ishni bajaradigan dasturlarni **Basic**, **Pascal**, **Ci** va boshqa tillarda yozish mumkin. **Pascal, Fortran** va **Kobol** tillari universal tillar hisoblanadi, **Ci** va **Assembler** tillari mashina tiliga ancha yaqin tillar bo‘lib, quyi yoki o'rta darajali tillardir. Algoritmik til inson tillariga qanchalik yaqin bo‘lsa, u tilga yuqori darajali til deyiladi. Mashina tili esa eng pastki darajali tildir. Mashina tili bu sonlardan iboratdir, Masalan: 010110100010101

Quyi darajali dasturlash tili ancha murakkab bo'lib ular juda maxsus sohalarda ishlatiladi va ularning mutaxassislari ham juda kam. Chunki quyi dasturlash tillari (masalan: assembler) ko'pincha miktoprotsessorlar bilan ishlashda kerak bo'lishi mumkin.

Odatda turli dasturlash ishlari uchun yuqori darajali dasturlash tilidan keng foydalaniladi.

EHM (Elektron Hisoblash Mashinasi) endi yuzaga kelgan paytda dastur tuzishda, faqat mashina tillarida, ya’ni sonlar yordamida EHM bajarishi kerak bo‘lgan amallarning kodlarida kiritilgan. Bu holda mashina uchun tushinarli sanoq, sistemasi sifatida 2 lik, 6 lik, 8 lik sanoq sistemalari bo‘lgan. Dastur mazkur sanoq sistemasidagi sonlar vositasida kiritilgan. Yuqori darajali dasturlashda, mashina tillariga qaraganda mashinaga moslashgan (yo‘naltirilgan) belgili kodlardagi tillar hisoblanadi. Belgilar kodlashtirilgan tillarning asosiy tamoyillari shundaki, unda mashina kodlari ularga mos belgilar bilan belgilanadi, hamda xotirani avtomatik taqsimlash va xatolarni tashhis qilish kiritilgan. Bunday mashina moslashgan til - **ASSEMBLER** tili nomini oldi.

Odatda dasturlash yuqori saviyali dasturlash tillari (**Delphi, Java, C++, Python**) vositasida amalga oshiriladi. Bu dasturlash tillarining semantikasi odam tiliga yaqinligi tufayli dastur tuzish jarayoni ancha oson kechadi. Ko'p ishlatiladigan dasturlash tillari. Biz hozir biladigan va ishlatadigan tillarning barchasi shu guruhga mansub. Ular insonga "tushunarli" tilda yoziladi. Ingliz tilini yaxshi biluvchilar dastur kodini qiynalmasdan tushunishlari mumkin. Bu guruhga **Fortran, Algol, C, Pascal, Cobol** va h. k. tillar kiradi(ko`pchiligi hozirda deyarli qo`llanilmaydi). Eng birinchi paydo bo`lgan tillardan to hozirgi zamonaviy tillargacha ishlatish mumkin. Lekin, hozirgi web texnologiya orqali ishlaydigan tillarda(**PHP, ASP. NET, JSP**) bunday dasturlar tuzilmaydi. Chunki bunday dasturlarning ishlashi uchun yana bir amaliy dastur ishlab turishi kerak. Hozirda, amaliy dasturlar, asosan, **Visual C++, C#, Borland Delphi, Borland C++, Java, Phyhon** kabi tillarda tuziladi. O`zbekistonda ko`pchilik Delphi dan foydalanadi. Buning asosiy sababi: soddaligi, komponentlarning ko`pligi, interfeysining tushunarliligi va h. k. Delphida birinchi ishlagan odam ham qanaqadir dastur tuzishi oson kechadi. Lekin, Windows da dasturning asosiy ishlash mohiyatini ancha keyin biladi(komponentlarning ko`pligi va API funksiyalari dasturda ko`rsatilmasligi uchun). Yana bir tarafi, Delphi(Pascal) operativ xotirani tejashga kelganda ancha oqsaydi. Unda o`zgaruvchilarni oldindan e'lon qilib qo`yish evaziga ishlatilmaydigan o`zgaruvchilar va massivlar ham joy olib turadi. Eng keng tarqalgan

dasturlash tili(Windows OS ida) **Microsoft Visual C++** tilidir. Ko`pchilik dasturlar hozirda shu tilda tuziladi. Umuman olganda, C ga o`xshash(C-подобный) tillar hozirda dasturlashda yetakchi. Deyarli hamma zamonaviy tillarning asosida C yotadi. Bundan tashqari, Turli komputer o'yinlari tuzishda yoki kichik hajmdagi dasturlar tayyorlashda **LUA script** yoki **javascript** tillari ham keng ishlatilmoqda.

Biz sizga xozirgi kunda keng tarqalgan **desktop** dasturlashda ishlatiladigan dasturlash tillaridan bazilari haqida aytib o'tamiz:

Delphi (talaff. délfi) — dasturlash tillaridan biri. Borland firmasi tomonidan ishlab chiqarilgan. Delphi dasturlash tili ishlatiladi va avvaldan Borland Delphi paketi tarkibiga kiritilgan. Shu bilan bir qatorda 2003-yildan hozirgacha qo‘llanilayotgan shu nomga ega bo'lgan. Object Pascal — Pascal tilidan bir qancha kengaytirishlar va to‘ldirishlar orqali kelib chiqqan bo‘lib, u ob’yektga yo‘naltirilgan dasturlash tili hisoblanadi. Avvaldan ushbu dasturlash muhiti faqatgina Microsoft Windows amaliyot tizimi uchun dasturlar yaratishga mo'ljallangan, keyinchalik esa **GNU/Linux** hamda **Kylix** tizimlari uchun moslashtirildi, lekin 2002-yilgi **Kylix 3** sonidan so'ng ishlab chiqarish to'xtatildi, ko'p o'tmay esa MicrosoftNET tizimini qo'llab quvvatlashi to'g'risida e'lon qilindi. Lazarus proekti amaliyotidagi (Free Pascal) dasturlash tili Delphi dasturlash muhitida **GNU/Linux, Mac OS X** va **Windows CE** platformalari uchun dasturlar yaratishga imkoniyat beradi.

Visual Basic (talaffuzi: "Vijual Beysik") – Microsoft korporatsiyasidan dasturlash tili va uning uchun dasturlash muhitdir. U BASICdan ko`p tushunchalar oldi va tez rasmli interfeys bilan dasturlar taraqqiyot ta`minlaydi. Oxirgi versiya 6. 0 1998 yilda reliz kelishdi. Microsoftdan voris Visual BasicNET 2002 yilda paydo bo`ldi.

Java dasturlash tili - eng yaxshi dasturlash tillaridan biri bo'lib unda korporativ darajadagi mahsulotlarni(dasturlarni) yaratish mumkin. Bu dasturlash tili **Oak** dasturlash tili asosida paydo bo'ldi. Oak dasturlash tili 90-yillarning boshida Sun Microsystems tomonidan platformaga(Operatsion tizimga) bog'liq bo'lmagan holda ishlovchi yangi avlod aqlli qurilmalarini yaratishni maqsad qilib harakat boshlagan edi. Bunga erishish uchun Sun hodimlari **C++** ni ishlatishni rejalashtirdilar, lekin ba'zi sabablarga ko'ra bu fikridan voz kechishdi. Oak muvofaqiyatsiz chiqdi va

1995-yilda Sun uning nomini **Java** ga almashtirdi, va uni WWW rivojlanishiga hizmat qilishi uchun ma'lum o'zgarishlar qilishdi. Java Obyektga Yo'naltirilgan Dasturlash(**OOP-object oriented programming**) tili va u C++ ga ancha o'xshash. Eng ko'p yo'l qo'yiladigan xatolarga sabab bo'luvchi qismlari olib tashlanib, Java dasturlash tili ancha soddalashtirildi. Java kod yozilgan fayllar(*. java bilan nihoyalanuvchi) kompilatsiyadan keyin bayt kod(bytecode) ga o'tadi va bu bayt kod interpretator tomonidan o'qib yurgizdiriladi.

C++ (talaffuzi: si plyus plyus) — turli maqsadlar uchun mo‘ljallangan dasturlash tili. 1979-yili Bell Labsda Biyarne Stroustrup tomonidan C dasturlash tilining imkoniyatlarini kengaytirish va OOP(object Oriented Programming) xususiyatini kiritish maqsadida ishlab chiqarilgan. Boshida „**C with Classes**“ deb atalgan, 1983-yili hozirgi nom bilan ya’ni C++ deb o‘zgartirilgan. C++ C da yozilgan dasturlarni kompilyatsiya qila oladi, ammo C kompilyatori bu xususiyatga ega emas. C++ tili operatsiyon tizimlarga aloqador qisimlarni, klient-server dasturlarni, EHM o‘yinlarini, kundalik ehtiyojda qo‘llaniladigan dasturlarni va shu kabi turli maqsadlarda ishlatiladigan dasturlarni ishlab chiqarishda qo‘llaniladi.

Quyidagi jadvalda dasturlash tillari haqida ma’lumotlar keltirilgan.

Til	Yaratilgan yili	Mualliflar	Tashkilot, firma
Ada	1979-80	Jean Ichbian	Cii-Honeywell (Fransiya)
Algol	1960		International Commitee
ARL	1961-1962	Kenneth Iverson, Adin Falkoff	IBM
DELPHI	1995	Borland	VASIS,
Beysik	1964-1965	JohnKemeny, Thomas Kurtz	Dartmouth Colleje
C	1972-1973	Dennis Ritchie	Bell Laboratories
C++	1980	Bjarne Strostrup	Bell Laboratories
Kobol	1959-1961	Grace Murray	Hopper
Fort	1971	Charles H. Moore	
FORTRAN	1950-1958	John Backus	IBM
HTML	1989	im Berners-Li CERN,	Jeneva LISP,
LISP	1956-1960	John MC	Carthy
LOGO	1968-70	Seymour Papert	Massachusetts Institute of Techn.

Pascal	1967-1971	Niklaus Wirth	Federal Institute of Technology (SHveysariya)
PL1	1964-1966		
PROLOG	1978	Alan Kalmeroe	
SIMULA	1967	Ole-Yoxan Dal, Kristen Nigaard	Norvegiya XM
Java	1995	Djeyms Gosling Sun	Microsystems

Dasturlash tillari komputerda bajarilishiga qarab kompilyatsiya qilinuvchi va interpretatsiya qilinuvchi tillarga bo‘linadi.

Kompilyatsiya qilinuvchi dasturlash tillarida dastur kodi kompilyator tomonidan mashina kodiga o‘tkaziladi. Operatsion tizim(OT) esa, shu kodni to‘g‘ridan-to‘g‘ri ishlataveradi. Kompilyatsiya jarayoni komputer protsessori va OT talablariga mos ravishda amalga oshiriladi. Shuning uchun, bir OT uchun kompilyatsiya qilingan dasturning mashina kodi ikkinchi OT da ishlamaydi. Ushbu turdagi tillarga quyidagilarni misol qilib keltirishimiz mumkin: C, C++, Pascal va h. k.

Microsoft Windows OTlarida kompilyatsiya qilingan dastur nomi *. exe ko‘rinishidagi fayl bo‘ladi. Linux, Unix(va shularning davomchilari) kabi OT larda esa fayl kengaytmasining ahamiyati yo‘q.

Kompilyatsiya qilinuvchi dasturlash tillarining asosiy yutuqlaridan biri — u OT dan boshqa biror dastur yoki kutubxona(Library, mas. DLL) o‘rnatishni talab qilmaydi. Bundan tashqari, interpretatsiya qilinuvchi tillarga nisbatan ancha tez ishlaydi.

Interpretatsiya qilinuvchi dasturlash tillarida tuzilgan dastur kodi kompilyatsiya qilinmaydi. Ushbu turdagi dasturni ishlatishdan oldin dastur kodi interpretatsiya qilinadi. Interpretatsiya qilinuvchi dasturlash tillarida tuzilgan dastur mos interpretator o‘rnatilgan komputerlardagina ishlaydi. Ushbu turdagi dasturlash tillariga PHP, Python, Ruby kabi tillar kiradi.

Interpretatsiya qilinuvchi dasturlash tillari kompilyatsiya qilinuvchilaridan, asosan, yozilgan dasturning deyarli hamma platformalarda ishlashi bilan ajralib turadi. Dastur biror turdagi OT yoki protsessor uchun yozilmaydi — faqat interpretatorgina turli platformalar uchun yoziladi.

Interpretatsiya qilinuvchi dastur kodi bajarilishidan oldin interpretator tomonidan oraliq kodga “kompilyatsiya” qilinadi. Shu oraliq kod interpretator tomonidan bajariladi. Python kabi tillar oraliq kodni saqlab qo‘yadi, dastur kodi o‘zgarmaguncha shu oraliq kodni ishlatadi.

Dastur biror masalani echishda elektron hisoblash mashinalari bajarishi lozim bo’lgan amallarning izchil tartibidan iborat. EHM uchun dastur tuzish jarayoni dasturlash deyiladi. Dasturlash echilishi kerak bo’lgan masala algoritmini EHM tiliga, ya’ni «mashina tili»ga o’tkazishdir. EHM uchun dastur tuzish – masalani echish usulini mashina buyruqlarining shunday majmui (dasturi)ga, keltirish demakki, bu buyruqlar xotiraga joylashib, tartib bilan amalga oshadi va tegishli hisoblashlarni bajaradi.

DASTUR biror masalani echishda elektron hisoblash mashinalari bajarishi lozim bo’lgan amallarning izchil tartibidan iborat.

EHM uchun dastur tuzish jarayoni **DASTURLASH** deyiladi.

KOMPYUTERDA MASALA YECHISH BOSQICHLARI (KMEB).

1. Masalaning qo’yilishi- bunda berilgan masalaning to’g’ri qo’yilganligi va uni yechish uchun qanday ma’lumotlar kerakligi va qanday natija olinishini bilish kerak.
2. Masalaning matematik modelini tuzish-bu bosqichda matematik formula tanlanadi
   matematik modeli tuziladi.
3. Masalani yechishni sonli usuli-bu bosqichda Hosil qilingan mat-k masalaning yechish usuli Tanlanadi va buning uchun sonli usuldan foydalaniladi.
4. Algoritmini tuzish-bunda mashina tushunadigan biror bir algoritmik tilda dG’tuzish.
5. Dasturni kiritish va chiqarish. Mavjud muhitdan foydalangan xolda.
6. Kiritilgan dastur natijaga erishishi.
7. Natijani tahlil qilish.

Mavzuga oid savollar

1. Dastur nima?
2. Dasturlash tillarining qanday turlarini bilasiz?
3. Dasturlash tillari va ularning klassifikasiyasi.
4. Mashinaga mo‘ljallangan va proseduraga mo‘ljallangan dasturlash tillari haqida ma’lumot bering.
5. Yuqori darjali dasturlash tillariga misol keltiring
6. Interpretatorlar va kompilyatorlar nima?
7. Dasturlarni translyasiyalash deganda nimani tushunasiz?

1.2. OB’EKTGA YO‘NALTIRILGAN DASTURLASH TILLARI.

Reja:

1. Ob’ektga yo‘naltirilgan dasturlash tillari tarixi.
2. Proseduraviy, strukturaviy va obyektlarga mo’ljallangan dasturlash.

Tayanch iboralar: Ob’ektga yo‘naltirilgan dasturlash tillari. Dasturlashning ob’ektga yo‘naltirilgan paradigmasi.

Ob’ektga yo‘naltirilgan dasturlash tillari tarixi

Obyektga mo’ljallangan yondoshuv (OMYo) bir kunda o’ylab topilgan emas. Uning paydo bo’lishi dasturiy ta’minotning tabiiy rivojidagi navbatdagi pog’ona xolos. Vaqt o’tishi bilan qaysi uslublar ishlash uchun qulay-u, kaysinisi noqulay ekanini aniqlash oson bo’lib bordi. OMYo eng muvaffaqiyatli, vaqt sinovidan o’tgan uslublarni o’zida samarali mujassam etadi.

Dastlab dasturlash anchayin boshqotirma ixtiro bo’lib, u dasturchilarga dasturlarni kommutasiya bloki orqali kompyuterning asosiy xotirasiga to’g’ridan-to’g’ri kiritish imkonini berdi. Dasturlar mashina tillarida ikkilik kurinishda yozilar edi. Dasturlarni mashina tilida yozishda tez-tez xatolarga yo’l qo’yilar edi, eng ustiga ularni tuzilmalashtirish imkoni bo’lmagani tufayli, kodni kuzatib borish amalda deyarli

mumkin bo’lmagan hol edi. Bundan tashqari, mashina kodlaridagi dastur tushunish uchun g’oyat murakkab edi.

Vaqt o’tishi bilan kompyuterlar tobora kengroq qo’llana boshladi hamda yuqoriroq darajadagi prosedura tillari paydo bo’ldi. Bularning dastlabkisi FORTRAN tili edi. Biroq obyektga mo’ljallangan yondoshuv rivojiga asosiy ta’sirni keyinroq paydo bo’lgan, masalan, ALGOL kabi prosedura tillari ko’rsatdi. Prosedura tillari dasturchiga axborotga ishlov berish dasturini pastroq darajadagi bir nechta proseduraga bo’lib tashlash imkonini beradi. Pastroq darajadagi bunday proseduralar dasturning umumiy tuzilmasini belgilab beradi. Ushbu proseduralarga izchil murojaatlar proseduralardan tashkil topgan dasturlarning bajarilishini boshqaradi.

Dasturlashning bu yangi paradigmasi mashina tilida dasturlash paradigmasiga nisbatan ancha ilg’or bo’lib, unga tuzilmalashtirishning asosiy vositasi bo’lgan proseduralar qo’shilgan edi, Maydaroq funksiyalarni nafaqat tushunish, balki sozlash ham osonroq kechadi. Biroq, boshqa tomondan, prosedurali dasturlash koddan takroran foydalanish imkonini cheklab qo’yyadi. Buning ustiga dasturchilar tez-tez «makaron» dasturlar ham yozib turishganki, bu dasturlarni bajarish likopdagi spagetti uyumini ajratishga o’xshab ketar edi. Va, nihoyat, shu narsa aniq bo’ldiki, prosedurali dasturlash usullari bilan dasturlarni ishlab chiqishda diqqatni ma’lumotlarga qaratishning o’zi muammolarni keltirib chiqarar ekan. Chunki ma’lumotlar va prosedura ajralgan, ma’lumotlar inkapsullanmagan. Bu nimaga olib keladi? Shunga olib keladiki, har bir prosedura ma’lumotlarni nima qilish kerakligini va ular qayerda joylashganini bilmog’i lozim bo’ladi. Agar prosedura o’zini yomon tusa-yu, ma’lumotlar ustidan noto’g’ri amallarni bajarsa, u ma’lumotlarni buzib qo’yishi mumkin. Har bir prosedura ma’lumotlarga kirish usullarini dasturlashi lozim bo’lganligi tufayli, ma’lumotlar taqdimotining o’zgarishi dasturning ushbu kirish amalga oshirilayotgan barcha o’rinlarining o’zgarishiga olib kelar edi. Shunday qilib, xatto eng kichik to’g’rilash ham butun dasturda qator o’zgarishlar sodir bo’lishiga olib kelar edi.

Modulli dasturlashda, masalan, Modula2 kabi tilda prosedurali dasturlashda topilgan ayrim kamchiliklarni bartaraf etishga urinib ko’rildi. Modulli dasturlash

dasturni bir necha tarkibiy bo’laklarga, yoki, boshqacha qilib aytganda, modullarga bo’lib tashdlaydi. Agar prosedurali dasturlash ma’lumotlar va prosedsralarni bo’lib tashlasa, modulli dasturlash, undan farqli o’laroq, ularni birlashtiradi. Modul ma’lumotlarning o’zidan hamda ma’lumotlarga ishlov beradigan proseduralardan iborat. Dasturning boshqa qismlariga moduldan foydalanish kerak bo’lib qolsa, ular modul interfeysiga murojaat etib qo’yaqoladi. Modullar barcha ichki axborotni dasturning boshqa qismlarida yashiradi.

Biroq modulli dasturlash ham kamchiliklardan holi emas. Modullar kengaymas bo’ladi, bu degani kodga bevosita kirishsiz hamda uni to’g’ridan-to’g’ri o’zgartirmay turib modulni qadamma-qadam uzgartirish mumkin emas. Bundan tashqari, bitta modulni ishlab chiqishda, uning funksiyalarini boshqasiga o’tkazmay (delegat qilmay) turib boshqasidan foydalanib bo’lmaydi. Yana garchi modulda turni belgilab bo’lsa-da, bir modul boshqasida belgilangan turdan foydalana olmaydi.

Modulli va prosedurali dasturlash tillarida tuzilmalashtirilgan va tuzilmalashtirilmagan ma’lumotlar o’z «tur»iga ega. Biroq turni kengaytirish usuli, agar «agregatlash» deb ataluvchi usul yordamida boshqa turlarni yaratishni hisobga olmaganda, mavjud emas.

Va, nihoyat, modulli dasturlash – bu yana proseduraga mo’ljallangan gibridli sxema bo’lib, unga amal qilishda dastur bir necha proseduralarga bo’linadi. Biroq endilikda proseduralar ishlov berilmagan ma’lumotlar ustida amallarni bajarmaydi, balki modullarni boshqaradi.

Obyektga mo’ljallangan dasturlash (OMD) modulli dasturlashdan keyingi mantiqiy pog’onani egallaydi, u modulga nasldan-naslga o’tishni va polimorfizmni qo’shadi. OMD dan foydalanr ekan, dasturchi dasturni bir qator oliy darajali obyektlarga bo’lish yo’li bilan tizimlashtiradi. Har bir obyekt hal qilinayotgan muammoning ma’lum bir tomonini modellashtiradi. OMD endilikda dasturni bajarish jarayonini boshqarish uchun dasturchi diqqatini proseduralarni ketma-ketlikda chaqirib olish ro’yxatini tuzib o’tirishga qaratmaydi. Buning o’rniga obyektlar o’zaro aloqada bo’ladi. OMYo yordamida ishlab chiqilgan dastur hal qilinayotgan muammoning amaldagi modeli bo’lib xizmat qiladi.

Dasturga obyektlar atamalari bilan ta’rif berish dasturiy ta’minotni ishlab chiqishning eng tushunarli usulidir. Obyektlar hamma narsani obyekt nima qilayotgani nuqtai nazaridan idrok etishga, ya’ni uning hatti-xarakatlarini hayolan modellashtirishga majbur qiladi. Shu tufayli obyektga yondoshishda u dasturning bajarilishi jarayonida qanday ishlatiladi degan nuqtai nazardan biroz e’tiborni chalg’itish mumkin. Shunday qilib, dasturni yozish jarayonida haqiqiy dunyoning tabiiy atamalaridan foydalanish mumkin. Dasturni alohida proseduralar va ma’lumotlar shaklida (kompyuter dunyosi atamalarida) qurish o’rniga, obyektlardan iborat dastur qurish mumkin. Obyektlar otlar, fe’llar va sifatlar yorlamida haqiqiy dunyoni dasturda modellashtirishga imkon beradi. Joriy qilish (realizasiya) hatti-xarakatlar qanday bajarilayotganini belgilaydi. Dasturlash atamalarida joriy qilish – bu dasturiy kod.

Yechilayotgan masala atamalari bilan fikrlab, joriy qilishning mayda-chuyda detallarida o’ralashib qolish havfidan qochish mumkin. Albatta, ayrim oliy darajadagi obyektlar kompyuter bilan aloqa qilishda past darajadagi, mashinaga mo’ljallangan usullardan foydalanishi lozim. Biroq obyekt bu aloqani tizimning boshqa qismlaridan izolyasiya qiladi.

Obyekt dastur konsturksiyasi bo’lib, unda holat va hatti-xarakat inkapsulalangan bo’ladi. Obyekt holati bu ichki obyekt o’zgaruvchanlari qiymatlarining yig’indisidir.

Ichki o’zgaruvchan deb obyekt ichida saqlanadigan qiymatga aytiladi.

Mohiyat e’tibori bilan, obyekt bu sinfning ekzemplyari (nushalaridan biri)dir.

OMD, haqiqiy dunyo kabi, obyektlardan tashkil topadi. Obyektga mo’ljallangan sof dasturlash tilida, eng dastlabki, bazaviy, butun, mantiqiy turlardan tortib, to sinflarning murakkabroq nushalarigacha, barchasi obyekt hisoblanadi. Biroq obyektga mo’ljallangan tillarning hammasi ham bu darajada chuqurlashib ketmagan. Ayrim tillarda (masalan, Java kabi) int va float ga o’xshash oddiy primitivlar obyekt sifatida olib qaralmaydi.

OMD obyektlari, haqiqiy olam obyektlari kabi, o’z xususiyatlari va xatti-harakatlari bo’yicha tasniflanadi.

Biologiyada itlar, mushuklar, fillar va odamlar sut emizuvchilarga kiradi. Bu turli xildagi jonivorlarni umumiy xususiyatlar birlashtirib turadi. Xuddi shunday, dasturiy ta’minot olamida ham obyektlar bitta yoki bir nechta sinflarga mansub bo’ladi.

Bitta sinfga mansub obyektlarga umumiy xususiyatlar xos bo’ladi. Boshqacha qilib aytganda, sinf obyektni tavsiflaydigan xususiyatlar va xulq-atvorlarni, shuningdek, va bu eng muhimidir, obyekt javob beradigan xabarlarni belgilab beradi. Biron bir obyekt boshqa obyektning xulq-atvoriga ta’sir ko’rsatgan vaqtda, u bu ta’sirni bevosita ko’rsatmaydi, balki undan qandaydir bir qo’shimcha axborotdan foydalangan holda o’zini-o’zi o’zgartirishni iltimos qiladi. Odatda bu «xabarni jo’natish» deb ataladi.

Sinf umumiy xususiyatlar va xulq-atvorga ega bo’lgan obyektlarni birlashtiradi. Bitta sinfga mansub obyektlar bir xil xususiyatlarga ega bo’lib, bir xil xatti-harakat namoyon etadi.

Sinflar shablon (qolip)ga o’xshaydi: ular obyektlarning ekzemplyarlari (nushalari)ni tayyorlash uchun qo’llanadi.

Belgilar – sinfning tashqaridan ko’rinib turgan xususiyatlari.

Obyekt ichki o’zgaruvchiga bevosita kirishni taqdim etganda yoki usul yordamida qiymatni qaytargandagina, o’z belgilarini namoyon qilishi mumkin.

Xulq-atvor – xabarga yoki holatning o’zgarishiga javoban obyekt tomnidan bajariladigan xatti-harakatlar. U obyekt nima qilayotganini bildiradi.

Bir obyekt ikkinchi obyekt ustida xatti-harakatlar bajarib, uning xulq-atvoriga ta’sir ko’rsatishi mumkin. «Xatti-harakat» atamasi o’rniga «usulni chaqiirish», «funksiyani chaqirish» yoki «xabarni uzatish» atamalari qo’llanadi. Muhimi bu atamalarning qaysi biri qo’llanayotganida emas, albatta, muhimi bu xatti-harakatlar obyekt xulq-atvorini namoyon qilishga da’vat etishidadir.

Obyektlar o’rtasida aloqa obyektga mo’ljallangan dasturlashning muhim tarkibiy qismidir. Obyektlar o’zaro aloqasining ikkita asosiy usuli mavjuddir.

Birinchi usul: obyektlar biri ikkinchisidan mustaqil ravishda mavjud bo’ladi. Agar alohida obyektlarga o’zaro aloqa kerak bo’lib qolsa, ular bir-birlariga xabar jo’natadi.

Obyektlar bir-birlari bilan xabarlar yordamida aloqa qiladi. Xabar olgan obyekt ma’lum xatti-harakatlarni bajaradi.

Xabar uzatish bu obyekt holatini o’zgartirish maqsadida uslubni chaqirib olish yoki xulq-atvor modellaridan birini qo’llashning o’zginasidir.

Ikkinchi usul: obyekt tarkibida boshqa obyektlar bo’lishi mumkin. Xuddi OMD da bo’lganidek, dastur obyektlardan tashkil topganidek, obyektlar ham, o’z navbatida, agregasiya yordamida boshqa obyektlardan jamlanishi mumkin. Ushbu obyektlarning har bittasida uslub va belgilarga ega bo’lgan interfeys mavjud bo’ladi.

Xabar – obyektga mo’ljallangan yondoshuvning muhim tushinchasi. Xabarlar mexanizmi tufayli obyektlar o’z mustaqilligini saqlab qolishi mumkin. Boshqa biron obyektga xabar jo’natayotgan obyekt uchun xabar olgan obyekt talabdagi xatti-harakatni qanday bajarishi unchalik muhim emas. Unga xatti-harakat bajarilganligining o’zi muhimdir.

Proseduraviy, strukturaviy va obyektlarga mo’ljallangan dasturlash

Shu vaqtgacha dasturlar berilgan ma’lumotlar ustida biror bir amal bajaruvchi proseduralar ketma-ketligidan iborat edi. Prosedura yoki funksiya ham o’zida aniqlangan ketma-ket bajariluvchi komandalar to’plamidan iboratdir. Bunda berilgan ma’lumotlarga murojaatlar proseduralarga ajratilgan holda amalga oshiriladi.

Strukturaviy dasturlashning asosiy g’oyasi «bo’lakla va hukmronlik qil» prinsipiga butunlay mos keladi. Kompyuter dasturini masalalar to’plamidan iborat deb qaraymiz. Oddiy tavsiflash uchun murakkab bo’lgan ixtiyoriy masalani bir nechta nisbatan kichikroq bo’lgan tarkibiy masalalarga ajratamiz va bo’linishni toki masalalar tushunish uchun yetarli darajada oddiy bo’lguncha davom ettiramiz.

Misol sifatida kompaniya xizmatchilarining o’rtacha ish haqini hisoblashni olamiz. Bu masala sodda emas. Uni qator qism masalalarga bo’lamiz:

1. Har bir xizmatchining oylik maoshi qanchaligini aniqlaymiz.
2. Kompaniyaning xodimlari sonini aniqlaymiz.
3. Barcha ish, haqlarini yig’amiz.
4. Hosil bo’lgan yig’indini kompaniya xodimlari soniga bo’lamiz.

Xodimlarning oylik maoshlari yig’indisini hisoblash jarayonini ham bir necha bosqichlarga ajratish mumkin.

1. Har bir xodim, haqidagi yozuvni o’qiymiz.
2. Ish xaqi to’g’risidagi ma’lumotni olamiz.
3. Ish haqi qiymatini yig’indiga qo’shamiz.
4. Keyingi xodim, haqidagi yozuvni o’qiymiz.

O’z navbatida, har bir xodim, haqidagi yozuvni o’qish jarayonini ham nisbatan kichikroq qism operasiyalarga ajratish mumkin:

1. Xizmatchi faylini ochamiz.
2. Kerakli yozuvga o’tamiz.
3. Ma’lumotlarni diskdan o’qiymiz.

Strukturaviy dasturlash murakkab masalalarni yechishda yetarlicha muvofaqqiyatli uslub bo’lib qoldi. Lekin, 1980 – yillar oxirlarida Strukturaviy dasturlashning ham ayrim kamchiliklari ko’zga tashlandi.

Birinchidan, berilgan ma’lumotlar (masalan, xodimlar, haqidagi yozuv) va ular ustidagi amallar (izlash, tahrirlash) bajarilishini bir butun tarzda tashkil etilishidek tabiiy jarayon realizasiya qilinmagan edi. Aksincha, proseduraviy dasturlash berilganlar strukturasini bu ma’lumotlar ustida amallar bajaradigan funksiyalarga ajratgan edi.

Ikkinchidan, dasturchilar doimiy tarzda eski muammolarning yangi yechimlarini ixtiro qilar edilar. Bu situasiya ko’pincha velosipedni qaytam ixtiro qilish ham deb aytiladi. Ko’plab dasturlarda takrorlanuvchi bloklarni ko’p martalab qo’llash imkoniyatiga bo’lgan hohish tabiiydir. Buni radio ishlab chiqaruvchi tomonidan priyomnikni yig’ishga o’xshatish mumkin. Konstruktor har safar diod va tranzistorni ixtiro qilmaydi. U oddiygina – oldin tayyorlangan radio detallaridan foydalanadi xolos. Dasturiy ta’minotni ishlab chiquvchilar uchun esa bunday imkoniyat ko’p yillar mobaynida yo’q edi.

Amaliyotga do’stona foydalanuvchi interfeyslari, ramkali oyna, menyu va ekranlarni tadbiq etilishi dasturlashda yangi uslubni keltirib chiqardi. Dasturlarni ketma-ket boshidan oxirigacha emas, balki uning alohida bloklari bajarilishi talab

qilinadigan bo’ldi. Biror bir aniqlangan hodisa yuz berganda dastur unga mos shaklda ta’sir ko’rsatishi lozim. Masalan, bir knopka bosilganda faqatgina unga biriktirilgan amallar bajariladi. Bunday uslubda dasturlar ancha interaktiv bo’lishi lozim. Buni ularni ishlab chiqishda hisobga olish lozim.

Obyektga mo’ljallangan dasturlash bu talablarga to’la javob beradi. Bunda dasturiy komponentlarni ko’p martalab qo’llash va berilganlarni manipulyasiya qiluvchi metodlar bilan birlashtirish imkoniyati mavjud.

Obyektga mo’ljallangan dasturlashning asosiy maqsadi berilganlar va ular ustida amal bajaruvchi proseduralarni yagona obyekt deb qarashdan iboratdir.

1.3. OB’EKTGA YO‘NALTIRILGAN LOYIHALASH.

Reja:

1. Ob‘ektlarni loyihalash: satrlar, steklar, ro’yxatlar, navbatlar, daraxtlar.
2. Matematik ob’ektlar: rasional va kompleks sonlar, vektorlar, matrisalar.
3. Ob’ektlar kutubxonasi.
4. Interfeys ob’ektlari: boshqarish elementlari, oynalar, dialoglar.

Tayanch iboralar: Ob‘ektlarni loyihalash: satrlar, steklar, ro’yxatlar, navbatlar, daraxtlar. Matematik ob’ektlar: rasional va kompleks sonlar, vektorlar, matrisalar. Ob’ektlar kutubxonasi. Interfeys ob’ektlari: boshqarish elementlari, oynalar, dialoglar.

*Obyektga mo’ljallangan tahlil va loyihalash (object-oriented analysis and design)*ning bosh g’oyasi predmetga oid sohani va masalaning mantiqiy yechimini obyektlar (tushunchalar va mohiyatlar) nuqtai nazaridan ko’rib chiqishdan iborat. Obyektga mo’ljallangan tahlil jarayonida asosiy diqqat-e’tibor obyektlar (yoki tushunchalar)ning predmetga oid soha atamalarida ta’riflash va tavsiflashga qaratiladi. Obyektga mo’ljallangan tahlil jarayonida obyektga mo’ljallangan dasturlash tili vositalari bilan joriy qilinadigan mantiqiy dasturiy obyektlar aniqlanadi. Bu dasturiy

obyektlar atributlar va metodlarni o’z ichiga oladi. Va, nihoyat, *konstruksiyalash (construction) yoki obyektga mo’ljallangan dasturlash (object-oriented programming)* jarayonida ishlab chiqilgan komponentlar va sinflarning joriy qilinishi ta’minlanadi.

Obyektga mo’ljallangan tahlil va loyihalashning ayrim asosiy tamoyillarini qisqacha ko’rib chiqamiz. Keyinchalik biz barcha ko’rib chiqilgan atamalarning yanada aniqroq ta’rifini va yanada to’liqroq shifr yechimini beramiz.

– Avval *talablar tahlili (requiremens analysis)* amalga oshiriladi hamda bu paytda modellashtirilayotgan tizimda sodir bo’layotgan asosiy jarayonlar va ularning presedentlar ko’rinishidagi ta’rifi ajratiladi. *Presedent (precedent)* – predmetga oid sohada sodir bo’layotgan jarayonlarning matniy tavsifi.

– Ikkinchi qadam. *Predmetga oid sohaning obyektga mo’ljallangan tahlili (object-oriented domain analysis)*. Bu qadamning vazifasi jarayon ishtirokchilarining faoliyat turlarini aniqlash hamda predmetga oid soha elementlarining turli kategoriyalarini aks ettiruvchi *konseptual model (conceptual model)* ni tuzishdan iborat.

– Uchinchi qadam. Bunda kim nima bilan shug’ullanayotganini ko’rib chiqamiz. Aynan shu faoliyat *obyektga mo’ljallangan loyihalash (object-oriented design)* deb ataladi va bunda asosiy diqqat-e’tibor majburiyatlarni taqismlashga qaratiladi. *Majburiyatlarni taqsimlash (responsibility assignment)* da ilovadagi turli dasturiy obyektlarning vazifa va majburiyatlari ajratib ko’rsatiladi.

Obyektga mo’ljallangan tahlil va loyihalashning muhim momenti shundan iboratki, bunda dasturiy tizim komponentlari o’rtasida majburiyatlar taqsimotini malakali o’tkazish ko’zda tutiladi. Gap shundaki, obyektga mo’ljallangan tahlil va loyihalashsiz ish ko’rib bo’lmaydi. Buning ustiga u robastlik, masshtablashish, kengayishlik va takroran qo’llash imkoniyatiga hal qiluvchi ta’sir ko’rsatadi. Obyektlarning majburiyatlari va o’zaro aloqalari *sinflar diagrammasi (design class diagram)* va *o’zaro aloqalar diagrammasi (collaboration diagram)* qo’llangan holda ifodalanadi.

Obyektga mo’ljallangan yondoshuvning afzalliklari va maqsadlari

OMYo dasturiy ta’minotni ishlab chiqishda oltida asosiy maqsadni ko’zlaydi. OMYo paradigmasiga muvofiq ishlab chiqilgan dasturiy ta’minot quyidagi xususiyatlarga ega bo’lmog’i lozim:

1. tabiiylik;
2. ishonchlilik;
3. qayta qo’llanish imkoniyati;
4. kuzatib borishda qulaylik;
5. takomillashishga qodirlik;
6. yangi versiyalarni davriy chiqarishning qulayligi.

Tabiiylik

OMYo yordamida tabiiy dasturiy ta’minot yaratiladi. Tabiiy dasturlar tushunarliroq bo’ladi. Dasturlashda «massiv» yoki «xotira sohasi» kabi atamalardan foydalanish o’rniga, yechilayotgan masala mansub bo’lgan soha atamalaridan foydalanish mumkin. Ishlab chiqilayotgan dasturni kompyuter tiliga moslash o’rniga, OMYo aniq bir sohaning atamalaridan foydalanish imkonini beradi.

Ishonchlilik

Yaxshi dasturiy ta’minot boshqa har qanday mahsulotlar, masalan, muzlatgich yoki televizorlar kabi ishonchli bo’lmog’i lozim.

Puxta ishlab chiqilgan va tartib bilan yozilgan obyektga mo’ljallangan dastur ishonchli bo’ladi. Obyektlarning modulli tabiati dastur qismlaridan birida, uning boshqa qismlariga tegmagan holda, o’zgartishlar amalga oshirish imkonini beradi. Obyekt tushunchasi tufayli, axborotga ushbu axborot kerak bo’lgan shaxslar egalik qiladi, mas’uliyat esa berilgan funksiyalarni bajaruvchilar zimmasiga yuklatiladi.

Qayta qo’llanish imkoniyati

Quruvchi uy qurishga kirishar eakan, har gal g’ishtlarning yangi turini ixtiro qilmaydi. Radiomuxandis yangi sxemani yaratishda, har gal rezistorlarning yangi turini o’ylab topmaydi. Unda nima uchun dasturchi «G’ildirak ixtiro qilaverishi kerak»? Masala o’z yechimini topgan ekan, bu yechimdan ko’p martalab foydalanish lozim.

Malakali ishlab chiqilgan obyektga mo'ljallangan sinflarni bemalol takroran ishlatish mumkin. Xuddi modullar kabi, obyektlarni ham turli dasturlarda takroran qo'llash mumkin. Modulli dasturlashdan farqli o'laroq, OMYo mavjud obyektlarni kengaytirish uchun vorislikdan, sozlanayotgan kodni yozish uchun esa polimorfizmdan foydalanish imkonini beradi.

Kuzatib borishda qulaylik

Dasturiy mahsulotning ish berish davri uning ishlab chiqilishi bilan tugamaydi. Dasturni ishlatish jarayonida *kuzatib borish* deb nomlanuvchi tirgak kerak. Dasturga sarflangan 60 foizdan 80 foizgacha vaqt kuzatib borishga ketadi. Ishlab chiqish esa ish berish siklining 20 foizinigina tashkil etadi.

Puxta ishlangan obyektga mo'ljallangan dastur ishlatishda qulay bo'ladi. Xatoni bartaraf etish uchun, faqat bitta o'ringa to'g'rilash kiritish kifoya qiladi. Chunki ishlatishdagi o'zgarishlar tiniq, boshqa barcha obyektlar takomillashtirish afzalliklaridan avtomatik ravishda foydalana boshlaydi. O'zining tabiiyligi tufayli dastur matni boshqa ishlab chiquvchilar uchun tushunarli bo'lmog'i lozim.

Kengayishga qodirlik

Foydalanuvchilar dasturni kuzatib borish paytida tez-tez tizimga yangi funksiyalarni qo'shishni iltimos qiladilar. Obyektlar kutubxonasini tuzishning o'zida ham ushbu obyektlarning funksiyalarini kengaytirishga to'g'ri keladi.

Dasturiy ta'minot statik (qotib qolgan) emas. Dasturiy ta'minot foydali bo'lib qolishi uchun, uning imkoniyatlarini muttasil kengaytirib borish lozim. OMYo da dasturni kengaytirish usullari ko'p. Vorislik, polimorfizm, qayta aniqlash, vakillik hamda ishlab chiqish jarayonida foydalanish mumkin bo'lgan ko'plab boshqa shablonlar shular jUMLasidandir.

Yangi versiyalarning davriy chiqarilishi

Zamonaviy dasturiy mahsulotning ish berish davri ko'p hollarda haftalar bilan o'lchanadi. OMYo tufayli dasturlarni ishlab chiqish davrini qisqartirishga erishildi, chunki dasturlar ancha ishonchli bo'lib bormoqda, kengayishi osonroq hamda takroran qo'llanishi mumkin.

Dasturiy ta’minotning tabiiyligi murakkab tizimlarning ishlab chiqilishini osonlashtiradi. Har qanday ishlanma hafsala bilan yondoshuvni talab qiladi, shuning uchun tabiiylik dasturiy ta’minotning ishlab chiqish davrlarini qisqartirish imkonini beradi, chunki butun diqqat-e’tiborni yechilayotgan masalaga jalb qildiradi.

Dastur qator obyektlarga bo’lingach, har bir alohida dastur qismini boshqalari bilan parallel ravishda ishlab chiqish mumkin bo’ladi. Bir nechta ishlab chiquvchi sinflarni bir-birlaridan mustaqil ravishda ishlab chiqishi mumkin bo’ladi. Ishlab chiqishdagi bunday parallellik ishlab chiqish vaqtini qisqartiradi.

Obyektga mo’ljallangan yondoshuvning uchta tamoyili

Obyektga mo’ljallangan yondoshuv (OMYo) ni tushunib yetish hamda undan foydalanishni uzlashtirib olish uchun, avvalam bor puxta bazaviy bilimlarni egallab olish lozim. Bazaviy tushunchalarni puxta anglab yetibgina, dasturlarni yaratishda OMYo ni qo’llash mumkin. Inkapsulyalash, vorislik va polimorfizm obyektga mo’ljallangan dasturlash (OMD) ninguchta bazaviy iushunchasi hisoblanadi.

Inkapsulyalash

Inkapsulyalash dasturni qandaydir monolit, bo’linmas narsa sifatida olib qaramay, ko’plab mustaqil elementlarga bo’lish imkonini beradi. Har bir element o’z funksiyalarini boshqa elementlardan mustaqil ravishda bajara oladigan alohida modul sifatida olib qaraladi. Aynan inkapsulyalash tufayli mustaqillik darajasi ortadi, chunki ichki detallar interfeys ortida yashiringan bo’ladi.

Inkapsulyalash modullikning obyektga mo’ljallangan tavsifidir. Inkapsulyalash yordamida dasturiy ta’minotni ma’lum funksiyalarni bajaruvchi modullarga bo’lib tashlash mumkin. Bu funksiyalarni amalga oshirish detallari esa tashqi olamdan yashirin holda bo’ladi.

Mohiyatan *inkapsulyalash* atamasi «germetik berkitilgan; tashqi ta’sirlardan himoyalangan dastur qismi» degan ma’noni bildiradi.

Agar biron-bir dasturiy obyektga inkapsulyalash qo’llangan bo’lsa, u holda bu obyekt qora quti sifatida olib qaraladi. Siz qora quti nima qilayotganini uning tashqi interfeysini ko’rib turganingiz uchungina bilishingiz mumkin. Qora quti biron narsa qilishga majburlash uchun, unga xabar yuborish kerak. Qora quti ichida nima sodir

bo’layotgani ahamiyatli emas, qora quti yuborilgan xabarga adekvat (mos ravishda) munosabatda bo’lishi muhimroqdir.

Interfeys tashqi olam bilan tuzilgan o’ziga xos bitim bo’lib, unda tashqi obyektlar ushbu obyektga qanday talablar yuborishi mumkinligi ko’rsatilgan bo’ladi. Interfeys – obyektni boshqarish pulti.

Ommaviy, xususiy va himoyalangan kirish.

Qandaydir bir elementni ommaviy interfeysga kiritish yoki, aksincha, undan chiqarish uchun, kalit so’zdan foydalanish kerak. OMD ning har bir tilida kalit so’zlar to’plami belgilangan, biroq bu so’zlar asosan bir xil funksiyalarni bajaradi.

Obyektga mo’ljallangan tillarning ko’pchiligida kirishning uchta darajasi mavjud.

-Ommaviy (public) – barcha obyektlar uchun kirish uchun ruxsat bor.

-Himoyalangan (protected) – faqat ushbu ekzemplyarga va har qanday tarmoq sinflarga kirishga ruxsat bor.

-Xususiy (private) – faqat ushbu ekzemplyarga kirishga ruxsat bor.

Loyihada kirish darajasini to’g’ri tanlab olish g’oyat muhimdir. Ko’rinadigan qilinishi lozim bo’lgan barcha narsa ommaviy bo’lmog’i lozim. Berkitilishi lozim bo’lgan har qanday narsa himoyalangan yoki xususiy kirishga ega bo’lmog’i kerak.

Inkapsulyalash nima uchun kerak?

Inkapsulyalashdan to’g’ri foydalanish tufayli obyektlar bilan o’zgartiriladigan komponentlar (tarkibiy qismlar) dek muomala qilish mumkin. Boshqa obyekt sizning obyektingizdan foydalana olishi uchun, u sizning obyektingizning ommaviy interfeysidan qanday fodalanish kerakligini bilishi kifoya. Bunday mustaqillik uchta muhim afzallikka ega.

– Mustaqillik tufayli, obyektdan takroran foydalanish mumkin. Inkapsulyalash puxta amalga oshirilgan bo’lsa, obyektlar ma’lum bir dasturga bog’lanib qolgan bo’lmaydi. Ulardan imkoni bo’lgan hamma yerda foydalanish mumkin bo’ladi. Obyektdan boshqa biron o’rinda foydalanish uchun, uning interfeysidan foydalanib qo’ya qolish kifoya.

– Inkapsulyalash tufayli, obyektda boshqa obyektlar uchun ko’rinmas bo’lgan o’zgarishlarni amalga oshirish mumkin. Agar interfeys o’zgartirilmasa, barcha

o’zgarishlar obyektdan foydalanayotganlar uchun ko’rinmas bo’ladi. Inkapsulyalash komponentni yaxshilash, amalga oshirish samaradorligini ta’minlash, xatolarni bartaraf etish imkonini beradi, yana bularning hammasi dasturning boshqa obyektlariga ta’sir ko’rsatmaydi. Obyektdan foydalanuvchilar ularda amalga oshirilayotgan barcha o’zgarishlardan avtomatik tarzda yutadilar.

– Himoyalangan obyektdan foydalanishda obyekt va dasturning boshqa qismi o’rtasida biron-bir ko’zda tutilmagan o’zaro aloqalar bo’lishi mumkin emas. Agar obyekt boshqalardan ajratilgan bo’lsa, bu holda u dasturning boshqa qismi bilan faqat o’z interfeysi orqali aloqaga kirishishi mumkin.

Shunday qilib, Inkapsulyalash yordamida modulli dasturlarni yaratish mumkin. Samarali Inkapsulyalashning uchta o’ziga xos belgisi qo’yidagicha:

-abstraksiya;

-joriy qilishning berkitilganligi;

-mas’uliyatning bo’linganligi.

Mavzuga oid savollar

1. Ob‘ektlarni loyihalash deganda nimani tushunasi?
2. Satrlar, steklar, ro’yxatlar, navbatlar, daraxtlarni o’zaro farqini ayting.
3. Matematik ob’ektlarni sanab bering.
4. Ob’ektlar kutubxonasi nima?
5. Interfeys ob’ektlari: boshqarish elementlari, oynalar, dialoglar.
6. Inkapsulyatsiya nima?

2-BOB. DELPHI DASTURLASH TILI

2.1. DELPHI DASTURLASH TILI ISHCHI MUHITI.

Reja:

1. Delphi dasturlash tilining ishchi muhiti, undagi oynalar, u o‘rnatilishi zarur bo‘lgan kompyuterga qo‘yiladigan texnik talablar va instrumental tugmalar.
2. Komponentlar palitrasi.
3. Palitra bo‘limlari va ayrim komponentlar xossalari bilan tanishish.

Tayanch iboralar: Delphi dasturlash tilining ishchi muhiti, undagi oynalar (Ob’ektlarning daraxtsimon ko‘rinish oynasi, ob’ektlar inspektori oynasi, kod brauzeri oynasi, asosiy oyna, forma oynasi, dastur kodi oynasi), u o‘rnatilishi zarur bo‘lgan kompyuterga qo‘yiladigan texnik talablar va instrumental tugmalar. Komponentlar palitrasi. Palitra bo‘limlari va ayrim komponentlar xossalari bilan tanishish.

So‘nggi vaqtlarda dasturlashga bo‘lgan qiziqish kuchayb ketdi. Bu holat kundalik xayotimizga informatsion-kommunikatsion texnologiyalarning kirib kelishi va rivojlanishi bilan bog‘liqdir. Inson kompyuterda ishlar ekan, ertami-kech unda dasturlash hoxishi uyg‘onadi, ba’zan esa zaruriyat tug‘iladi.

SHK foydalanuvchilari asosan Windows OT ishlashga va dastur tuzishda esa aynan shu tizimda ishlay oladigan dastur tuzishga intilishadi. Avvallari bunday dastur tuzish uchun tajribaga ega bo’lgan professional dasturchilarga mo‘ljallangan Borland C++ for Windows vositasidan foydalanib kelingan. Hisoblash texnikasining rivojlanishi «tez dasturlash» deb yuritiluvchi Borland Delphi va Microsoft Visual Basic kabi dasturlash tizimlarinini yaratilishiga olib keldi. Dasturlarni tez qayta ishlash muxiti (RAD-tizimlari, Rapid Application Development — dasturlarni tez qayta ishlash muxiti) ning asosi - vizual loyihalash va xodisaviy dasturlashdir. Uning mohiyati shundauki, mazkur muhit asosiy ishlarni o‘z zimmasiga oladi va dasturchiga

muloqatli oynali loyihalash va xodisani qayta ishlashni yuklaydi. RAD-tizimlarini foydalangan dasturchi kata natijalarga oson va tez erishishi mumkin.

Delphi - dasturlarni tez qayta ishlash muxiti bo‘lib, dasturlash tili sifatida Delphi tili qo‘llaniladi. Delphi tili – dasturchilarga tanish bo‘lgan Object Pascal asosida ishlaydigan qat’iy tiplashtirilgan obyektli – oriyentirlangan tildir.

Uning birqancha versiyalari mavjud bo‘lib, dasturchilar xozirgi kunda navbatdagi Borland Delphi 7 Studio paketi versiyasidan keng foydalanmoqda. Bu versiya paketi avvalgi versiyalari kabi turli xil dasturlarni: oddiy bitta amaliy oynali dasturdan tortib murakkab bazalar bilan ishlay oladigan dasturlarni tuzish imkonini beradi. Paketga MO, ma’lumotlar tizimi orqali hosil qilingan XML-xujjatlar bilan ishlash va boshqa turli masalalarni xal qilish imkonini yaratuvchi turli utilitlar kiritilgan. 7-versiyaning asosiy farqli tomoni -. NET texnologiya bilan bog‘liqligidir.

Borland Delphi 7 Studio Windows OT da ishlay oladi. Bu paketdan foydalanish uchun kompyuter resurslariga quyidagi talablar qo‘yiladi:

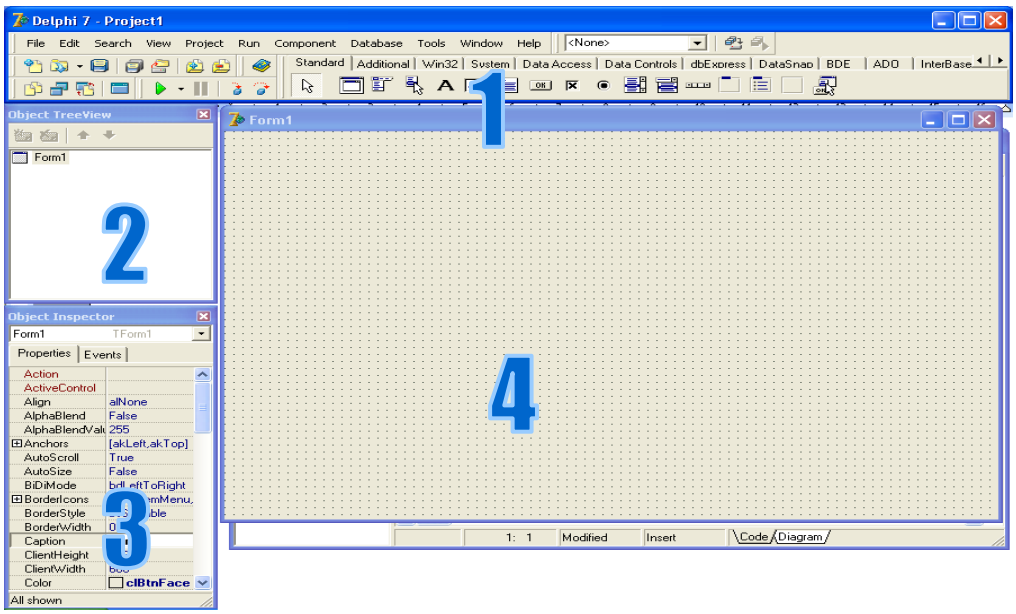
- ❖ OT Windows 98 dan to Windows 8 gacha
- ❖ Protsessor taktli chastotasi 166 dan kam bo‘lmagan Pentium yoki Celeron
- ❖ Operativ xotirasi – 128 Mbayt (256 Mbayt tavsiya etiladi) dan yuqori
- ❖ Qattiq diskda paketni o‘rnatish uchun zarur miqdorda joy (tahminan 475 Mbayt).

Delphi o‘zining ikki xossasi bilan tez dasturlash imkonini beradi:

1. Formani vizual jihozlash imkoni.
2. Tayyor kutubxondan foydalanish imkoni.

Delphi 7 dasturini ishga tushirish uchun quyidagi amallar ketma-ketligini bajarish lozim:

Pusk →Vse programmi → Borland Dephi 7 → Dephi 7

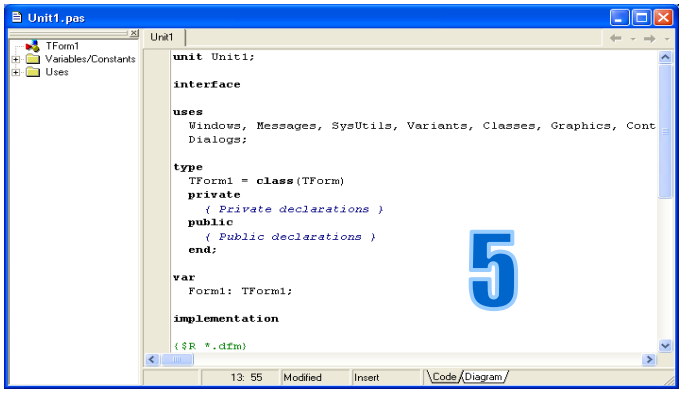


1. 1-rasm.

Ochilgan oyna asosan 5 qismdan iborat:

1. Asosiy oyna (**Delphi 7 Project1**);
2. Komponentlar daraxti (**Object TreeView**);
3. Obyektlar inspektori (**Object Inspector**);
4. Forma (**Form1**);
5. Kod kiritish oynasi (**Unit1. pas** - bu oyna asosiy forma aktiv holda F12 tugmasini bosish orqali aktivlashtiriladi)

Boshlang‘ich shakl oynasi taxrirlash oynasini deyarli to‘liq qamrab olgan.

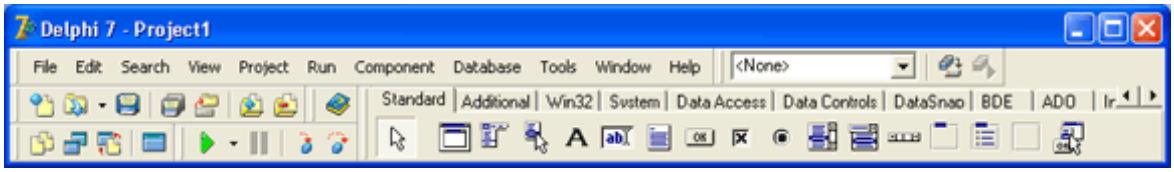


1. 2-rasm.

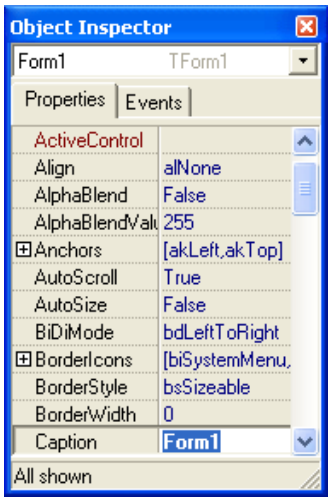
1. **Asosiy oyna.** Bu yerda menyular satri va komponetlar paneli joylashgan. Asosiy oyna tuzilayotgan dastur loyixasining asosiy ishini bajaradi. Bu oyna ekranda xar doim yuqori qismda joylashadi, o‘lchami xam deyarli o‘zgarmaydi. Asosiy oynani buyruqlar paneliga yig‘ib qo‘yish, ekrandagi Delphi ning qolgan

oynalarini xam yig‘ib qo‘yadi, asosiy oynaning yopilishi esa Delphi ishini yakunlashga olib keladi. Uskunalar panelini shakllantirish uchun uskunalar paneli ustida sichqonchaning o‘ng tugmachasini bosish va tegishli gurux nomiga «bayroqcha» o‘rnatish mumkin, Customize … (Nastroyka…) buyruqlari orqali muloqatli oyna hosil qilinib, oynaning Command nomli sahifasi orqali ma’lum buyruq tugmachasini o‘rnatish yoki yo‘qotish mumkin.

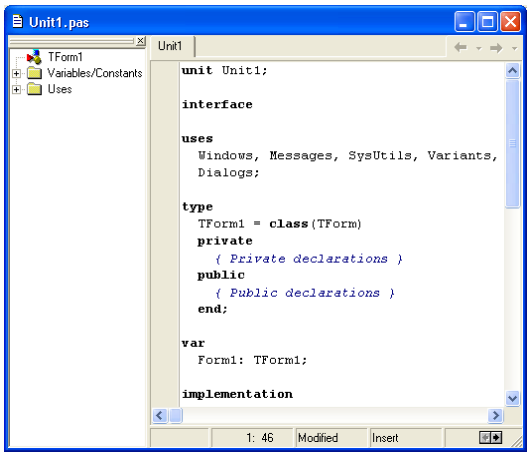
2. **Komponentlar daraxti** formaning vizual ko‘rinishini yaratish davomida ishlatilgan obyektlarning ro‘yxaini daraxt ko‘rinishida beradi. Komponentov palitrasi – bu Delphi ning asosiy boyligidir. U asosiy oynaning o‘ng qismini egallaydi va birqancha sahifalardan iarkib topadi. Komponent deyilganda dasturchi tomonidan forma tarkibiga kiritilishi mumkin bo’lgan, o‘zining xususiyatlariga ega bo‘lgan biror funksional element tushiniladi. Ular orqali dasturning ekranda ko‘rinuvchi elementlari, ya’ni oynalari, tugmachalari, ro‘yxati va hokazo. Komponentlarni xam komponentlar ustida sichqonchaning o‘ng tugmachasi orqali hosil qilingan ro‘yxat orqali shakllantirish mumkin.



3. **Obyektlar inspektori** obyektlarni boshqarishga mo‘ljallangan bo‘lib, ikki vkladkadan iborat:
    - **Properties** – xossalar. Bu vkladkada belgilangan obyektning xossalari belgilanadi. Obyektlar panelning yuqori qismidagi tanlash maydonidan tanlanishi mumkin.
    - **Events –** xodisalar. Bu yerda belgilangan obyektning biror xodisaga nisbatan reaksiyasi belgilanadi.



4. **Forma** yaratish oynasida bo‘lajak dasturning ko‘rinishi yaratiladi. Forma oynasi tuziluvchi dasturning Windows-oynasi loyihasini ifodalaydi. Daslab bu oyna bo‘sh bo‘ladi, ya’ni faqat sarlavha satridan va bo‘sh maydondan tarkib topadi. Dasturchi ma’lum vaqtni formaga interfeys elementlarini, ya’ni komponentlarni joylashtirishga sarflaydi. Aynan shu jarayonda vizual dasturlash asosi yaratiladi. Dasturchi ixtiyoriy vaqtda oynadagi elementlarga o‘zgartirish kiritishi mumkin.
5. **Kod kiritish oynasi**da buyruqlar kiritiladi. Dastur kodlari oynasi dastur matnini kiritish va taxrirlash uchun mo‘ljallangan. Bu dastur matni Paskal dasturlash tilining kengaytirilgan va takomillashgan versiyasi Object Pascal dasturlash tili asosida yoziladi. Bu til dastlab 1970 y. Shveysariya olimi N. Virt va Borland korporatsiyasi xodimlari tomonidan tavsiya etilgan. Vizual muxit Delphi o‘z zimmasiga dasturlashning ko‘p aspektlarini olishiga qaramay, bu muxitda ishlash uchun dasturlash tilni bilish asosiy shartlaridan biri hisoblanadi. Dastlab kodlar oynasida forma funksiyasini belgilovchi dasturning minimal boshlang‘ich matni beriladi. Sodda dastur yaratish uchun xam unga o‘zgartirish kiritish lozim bo’ladi. Formaga o‘rnatilgan element ustida sichqonchaning ikki marta bosilishi, aynan shu element funksiyasini belgilash uchun dasturni kirtish imkonini beradi. (Formaga qaytish – F12, Inspektorga o‘tish – F11)



Menyular satri.

File menyusi.

- **New** – yangi proyekt, forma yoki shablon yaratish. Bu buyruq ustida kursor bir oz to‘xtatib turilsa, eng ko‘p ishlatiladigan forma va modullar ro‘yxati berilgan ost menyu ochiladi. **Other** ost menyusi yaratilishi mumkin bo‘lgan barcha modullar ro‘yxatini beruvchi muloqot oynasini ochadi;
- **Open** – mavjud fayllarni ochish;
- **Open Project** – mavjud proyektni ochadi. Proyekt bir nechta modullardan tashkil topganligi sababli yaratilayotgan dastur ustida ishlash uchun qandaydir fayl emas, proyektning o‘zi ochilishi kerak. Aks holda yozgan dasturimizni kompilyatsiya (Dasturash tili muxiti yozilgan kodning xatoliklarin tekshirib chiqadi va uni bir kengaytma bilan saqlaydi. Odatda **exe** sifatida) qilishning iloji bo‘lmaydi;
- **Reopen** – avval ko‘rilgan proyektni qayta ochish;
- **Save** – aktiv modulni saqlash;
- **Save As** – aktiv modulni yangi nom bilan saqlash;
- **Save Project As** – proyektni yangi nom bilan saqlash;
- **Save All** – barcha o‘zgatirishlarni saqlash;
- **Close** – aktiv modulni yopish;
- **Close All** – barchasini yopish;
- **Use Unit** – moduldan foydalanish;
- **Print** – modulni chop etish;
- **Exit** – chiqish.

Modul – bu dastur kodini butunligicha yoki bir qismini o‘zida saqlovchi fayldir. U. **pas** kengaytmasi bilan saqlanadi. Xozirgi kunda modul sifatida dasturning vizual qismini ham tushunilmoqda. Chunki dastur kodi va vizual qismi alohida fayllarda saqlansa ham ular bir-biri bilan chambarchas bog‘liq.

Proyekt – bir yoki bir necha modullarni o‘z ichiga olgan loyiha.

Endi **Edit** menyusi bilan tanishamiz. Bu menyu buyruqlari boshqa matn yoki grafik muxarrirlardagi kabi Delphi 7 da ham matnlarni taxrirlashda, bundan tashqari vizual obyektlar bilan ishlashda qo‘llaniladi.

View menyusi dasturlash muhitidagi ma’lum panellarni o‘chirish yoki yoqish imkonini beradi.

- **Project Manager** – buyrug‘i yordami proyekt tarkibiga kiradigan obyektlarni ko‘rish, yangi hosil qilish va ularni o‘chirish mumkin bo‘lgan muloqot oynasi ochiladi;
- **Object Inspector** – Obyektlar inspektori oynasini ochish/yopish vazifasini bajaradi;
- **Object TreeView** – komponentlar daraxti oynasini ochish/yopish vazifasini bajaradi;
- **Browser** – klass, komponent, modul, obyekt, o‘garuvchi va boshqalarning shajarasini ko‘rsatuvchi oynani hosil qiladi. Shajara tushunchasini keyinroq tushuntirib o‘taman;
- **Code explorer** – aktiv formaning kodi yozilgan oynani aktivlashtiradi.

View menyusining asosiy buyruqlari shulardan iborat edi. Endi eng muhim menyulardan biri **Project** bilan tanishamiz.

- **Add to project** – proyektga mavjud fayllarni qo‘shish;
- **Remove from project** – proyektdan modulni o‘chirish;
- **Remove from project** – modul shablon sifatida saqlab qo‘yiladi va uning asosida yangi forma yaratish mumkin bo‘ladi;
- **View source –** proyektning kodi. Bu kod qandaydir biz yaratgan qandaydir formaning emas, balki aynan proyektning dasturlash muhiti tomonidan avtomatik ravishda yaratilgan kod hisoblanadi;

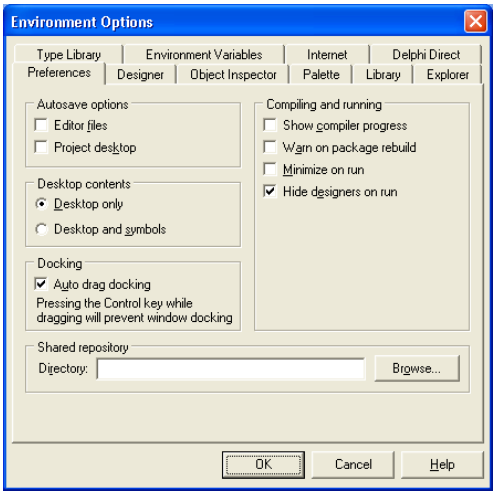
- **Compile ProyektNomi** – ProyektNomi nomli proyektni kompilyatsiya qilish. Bu yerda ProyektNomi so‘zi o‘rnida sizning proyektingiz nomi bo‘ladi. Komilyatsiya jarayonini tezkor amalga oshirish uchun Ctrl+F9 tugmalaridan foydalanish mumkin;
- **Build ProyektNomi** - ProyektNomi nomli proyektni qayta qurish. Qayta qurishning kompilyatsiyadan farqi shundaki, kompilyatsiya jarayonida Delphi proyektni ishlatish uchun kerak bo‘ladigan maxsus fayllarni yaratib oladi. Navbatdagi kompilyatsiyada o‘zgartirish kiritilmagan modullar komplyatsiya qilinmaydi, balki avval yaratilgan fayllardan foydalaniladi. Qayta qurishda esa barchasi yangidan kompilyatsiya qilinadi va u o‘zgartirish kiritilgandan so‘ng proyekt kompilyatsiya qila olmayottgan holda kerak bo‘ladi;
- **Options** – proyektning xossalari.

Run menyusining **Run** buyrug‘i yordamida dasturni ishga tushirib ko‘rish, mumkin.

Component bo‘limining buyruqlari yordamida mavjud komponentlarni o‘rnatish yoki yangi komponent yaratish mumkin.

Delphi 7 dasturlash tili muhitida turli sozlashlar mavjud bo‘lib, ularning barchasini ko‘rib chiqishga juda ko‘p vaqt kerak bo‘ladi. Buning ustiga ularning hammasi ham doim ishlatilaverilmaydi.

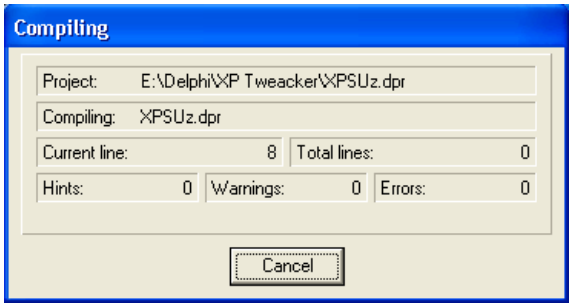
Asosiy sozlashlarni amalga oshirish uchun **Tools** menyusining **Environment Options** bo‘limi tanlanadi. Natijada quyidagi oyna ochiladi:



1. 3-rasm.

Oyna bir nechta vkladkalardan tashkil topgan bo‘lib, **Preferences** vkladkasida quyidagi sozlashlarni amalga oshirish mumkin:

- **Autosave options** – avtomatik ravishda saqlash xossalari. Bu yerda ikkita punkt mavjud:
  - **Editor Files –** taxrirlanayotgan modulni saqlash. Ushbu punkt belgilangan xolda modul avtomatik ravishda saqlab boriladi;
  - **Project desktop –** agar ushbu puknt belgilangan bo‘lsa, ishchi oynadagi barcha o‘zgarishlar avtomatik ravishda saqlab qo‘yiladi.
- **Compiling and Running** – kompilyatsiya va tayyor dasturni ishga tushirish jarayonini sozlash. Bu yerda bir necha bo‘limlar mavjud bo‘lib, ulardan asosiylari quyidagilar:
  - **Show compiler progress –** kompilyatsiya jarayonini ko‘rsatib turuvchi oynani ochishga xizmat qiladi. Bu oynada kompilyatsiya jarayoni va uning natijasi haqida ma’lumot berib boriladi. Oynaning nuqsonli tomoni kompilyatsya jarayonini uzaytirib yuborishidir. Oynaning ko‘rinishi quyidagicha:



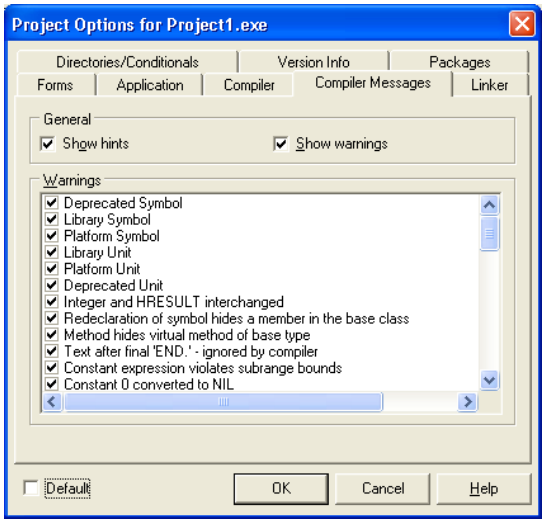
1. 4-rasm.

Uning eng pastki qatoridagi uchta maydon asosiy maydon hisoblanadi:

1. **Hints –** xabar. Bu yerda dastur kodinining optimallashtirilishi mumkin bo‘lgan qismi haqida ma’lumot beriladi. Masalan, biror o‘zgaruvchi e’lon qilingani holda ishlatilmasa, shu haqida ma’lumot beriladi;

2. **Warning** – ogohlantirish. Bu maydonda dastur kodini yozish davomida yo‘l qo‘yilgan elementar xatolar ko‘rsatiladi. Misol tariqasida boshlang‘ich qiymati berilmasdan ishlatishga urinilgan o‘zgaruvchilarni ko‘rsatish mumkin;
3. **Errors –** dastur kodini yozishda yo‘l qo‘yilgan va kompilyatsiya jarayonining mamalga oshirilishiga to‘sqinlik qiluvchi xatolar ko‘rsatiladigan maydon.
   - **Minimize on run –** dasturni tekshirish uchun ishga tushirilganda, Delphi oynasi pastga olib qo‘yiladi;

Yuqoridagi sozlashlar Delphining barcha proyektlari uchun mo‘ljallangan bo‘lib, xar bir aktiv proyektni aloxida sozlash ham mumkin. Buning uchun **Environment Options** oynasidan chiqib, **Project** menyusining **Options** bo‘limi tanlanadi. Xozircha uning **Compiler Messages** vkladkasini ko‘rib chiqamiz (1. 5-rasm):



1. 5-rasm.

General bo‘limi ikki punktdan iborat.

- **Show warnings –** kompilyatsiya jarayonida ogohlantirishlarni ko‘rsatish;
- **Show hints** – kompilyatsiya davomida xabarlarni chiqarish.

Yuqoridagi ikki punkt odatda belgilangan holatda bo‘ladi. Men Sizga ularni o‘chirib qo‘yishni tavsiya qilmayman. Chunki ikkisi ham ish davomida tez-tez asqotib qolishi mumkin. Pastroqda esa kompilyatsiya jarayonida o‘chirib/yoqib qo‘yish foydaliroq bo‘lgan xabarlar ro‘yxati bo‘lib, ular bilan tanishib chiqamiz:

– **Platform symbol** (platformaga bog‘liq belgi) – ushbu xabar faqat mazkur platformadagina mavjud bo‘lgan belgi ishlatilganda beriladi. Delphida yozilgan

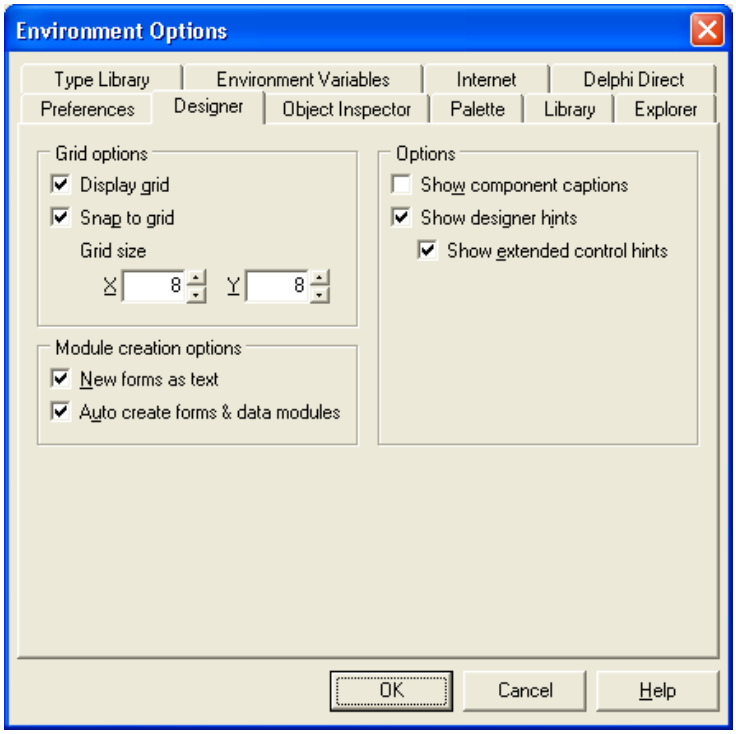
dasturlar Linux operatsion sistemasi uchun Borland Kylix yordamida qayta kompilyatsiya qilinishi mumkin. Delphida ishlayotgan vaqtingizda Borland Kylixda mavjud bo‘lmagan belgidan foydalansangiz, ushbu xabar beriladi. Delphida yozilgan dasturlar kamdan-kam xollarda Borland Kylixda qayta kompilyatsiya qilinganligi uchun bu xabarni o‘chirib qo‘yish ziyon keltirmaydi;

– **Platform unit** (platformaga bog‘liq modul) – bu punkt ham yuqoridagi kabi bo‘lib, farqi belgi emas modul uchun ekanligida xolos;

– **Unsafe type, Unsafe code i Unsafe typecast** punktlari Delphi 7 da paydo bo‘lgan bo‘lib, bu xabarlar xatolikka olib keluvchi o‘zgaruvchi, kod yoki yozuvlar kiritilganda beriladi. Katta xajmli dasturlar yozilayotganda bunday xabarlar soni juda ko‘p bo‘lishi mumkin. Ular orasidan aynan keraklisini qidirib topish oson bo‘lishi uchun bu uch punktning belgilanmagan xolatda bo‘lishi maqsadga muvofiq.

Endi yana **Environment Options** oynasiga qaytamiz va **Designer** vkladkasini faollashtiramiz (1. 6-rasm). Bu vkladkada bizni qiziqtirishi extimoli ko‘proq bo‘lgan ikki punkt bor. Ular:

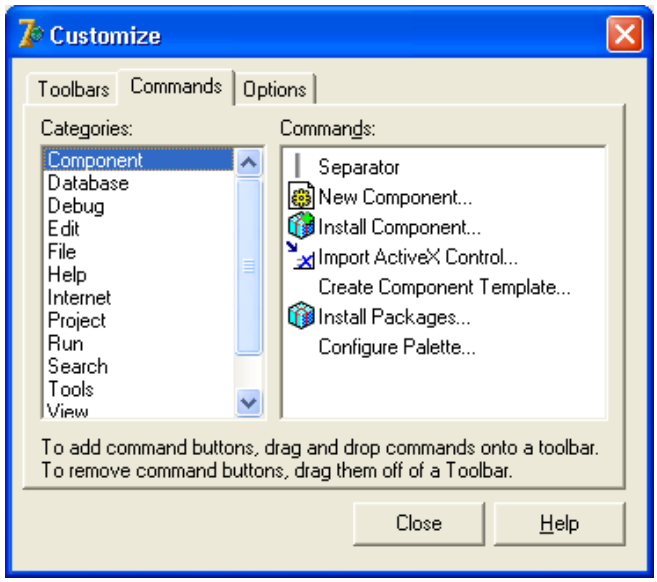
- **Display grid –** setkani ko‘rsatish;
- **Snap to grid –** obyektlarni setka yordamida ko‘chirish;



1. 6-rasm.

Formadagi setka odatda ko‘ringan holatda bo‘ladi va men sizga uni o‘chirib qo‘yishni maslahat bermayman. Chunki bo‘lajak dasturingizning interfeysini yaratayotgan vaqtingizda uning chiroqli ko‘rinishi uchun setka katta yordam beradi. Setkaning kattaligi odatda 8x8 pikseldan iborat bo‘ladi. Uni o‘zgartirish mumkin, lekin buni ham maslahat bermayman. Agar juda zarur bo‘lmasa, yaxshisi barchasi o‘z xolicha qolgani ma’qul.

Ko‘rib chiqishimiz kerak bo‘lgan navbatdagi va so‘nggi sozlash oynasi uskunalar panelida sichqonchaning o‘ng tugmasini bosib, ochilgan kontekst menyudan **Properties** (Xossalar) buyrug‘ini tanlash orqali chaqiriladi (1. 7-rasm).



Bu oyna uchta vkladkadan iborat bo‘lib, birinchi vkladkada panellar ro‘yxati beriladi. Bu yerda faqat tez-tez ishlatadigan panellarimizni qoldirib, qolganlarini o‘chirib qo‘yishimiz mumkin. Ikkinchi vkladkada “sudrab o‘tish” yo‘li bilan uskular paneliga qo‘shish mumkin bo‘lgan barcha tugmalar ro‘yxati beriladi.

Mavzuga oid savollar

1. Delphi dasturini ishga tushirish yo’llarini aytib bering
2. Delphi dasturining oynalari nomlari qanday?
3. Delphi DTda kodlar oynasi nima uchun ishlatiladi?

2.2. STANDARD KOMPONENTLAR PALITRASI.

Reja:

1. Label, Edit, Memo matn komponentlari va Button tugmachasi
2. Boshlang'ich forma ilovasini yaratish
3. Tanlash tugmalarini o'rnatish
4. ListBox va ComboBox komponentalari

Tayanch iboralar: Frame komponenti va uning xossalari. MainMenu, PopupMenu komponentlari. Label, Edit, Button, Memo, Panel komponentlari va ularning xossalari. CheckBox, ListBox, ComboBox, ListBox, RadioGroup, RadioButton, ScrollBar komponentlari.

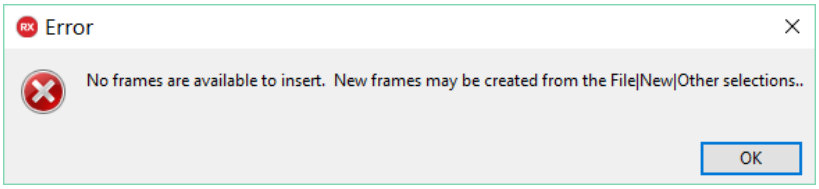
Standard ilovasida joylashgan komponentlar Windows tizimida ishlaydigan ilovaning interfeys elmentlarini yartish uchun mo'ljallangan bo'lib ular dasturlar yaratishda muhim hisoblanadilar. Ularning ba'zilari haqida ma'lumotlar keltiramiz.

Frames	Frame. Forma oynasi kabi boshqa komponenetlarni o'zida qamrab oluvchi konteyner vazifasini bajaradi va formadan farqli holda tanlangan ilovada joylasha oladi.
TLabel	Label. Bu komponenti o'zida qisqa hajmdagi matnli ma'lumotlarni o'zida aks ettirishga mo'ljallangan.
TMainMenu	MainMenu. Dasturning bosh menyusini yaratish uchun qo'llaniladi. Bu menu murakkab iyerarxik ko'rinishda ham bo'lishi mumkin.
TPopupMenu	PopupMenu. Dastur va uning ob'ektlari uchun kontekstli menu yartishga mo'ljallangan
TEdit	Foydalanuvchi tomonidan bir qatorli ma'lumotlarni kiritish uchun foydalaniladi. Ayrimhollardakeraklimatnni turli maqsadlarda aks ettirishi ham mumkin.
TMemo	Ko'p qatorli matnlarni aks ettiruvchi, kirituvchi va tahrirlovchi matnli maydon komponenti.

TCheckBox	Foydalanuvchiga turli holatlarni tanlash imkoniyatini beradi.
TRadioButton	Foydalanuvchiga bir necha varianlardan birini tanlash imkoniyatini beradsi. Forma yoki boshqa konteynerga birnecht ao'rnatilishi mumkin.
TButton	Tugma sifatida turli vazifalarni bajaruvchi komponent.
TListBox	Maxsus ro'yxat sifatida shakllnuvchi va foydalanuvchi undan ro'yxatning bo'limlari sifatida foydalanishi mumkin.
TComboBox	ComboBox - kiritish qatoriga ega (комбированній) ro'yxatdan tanlash. Ro 'yxatdan kombinatsiya qilib tanlash.
TScrollBar	ScrollBar - yo'lchali boshqarish. Windows oynasi chetlaridan gorizontal yoki yertikal yo'lcha tashkil etadi.
TRadioGroup	RadioGroup - bog'liq guruhlangan tanlash tugmalari (o'chirib yoquychi tugmalar). Bir necha bog'liq tanlash tugmalari xossalarini saqlaydi.
TPanel	Ви komponenta xuddi GroupBoxga o'xshab bir necha komponentalarni birlashtirish uchun xizmat qiladi.
TActionList	Actionlist - ta'sir qilish ro'yxatlari. Foydalanuvchi dasturga markazlashgan holda ta'sir qilishi uchun ishlatiladi.
TGroupBox	GroupBox - elementlar guruhi. Ma'по bo'yicha bir necha bog'liq komponentalarni guruhlashda ishlatiladi.

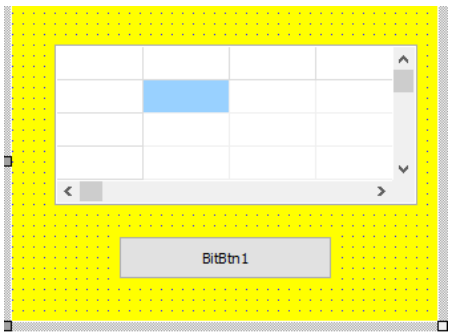
Frame komponenti.

Frame komponenti dastavval yaratilishi kerak va shu holatda undan foydalanish mumkin. Uni yaratmasdan foydalanishga harakat qilinsa quyidagi mazmundagi xatolik xaqida xabar beriladi.

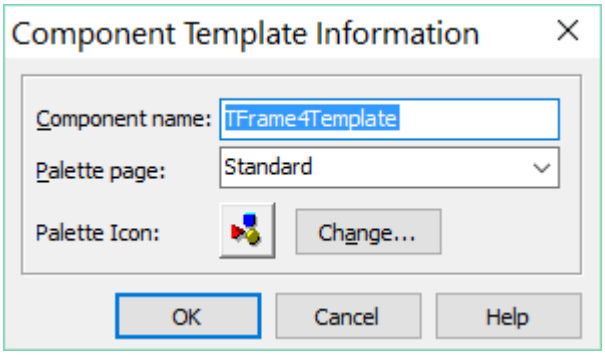


Uni yaratish tartibi qyuidagicha:

1. File->new->Other-> Delphi Files-> Vcl Frame->Ok buyruqlar tizimidan foydalanib frame oynasini yaratib olamiz va unda kerakli komponentlar joylashtiriladi va zarur hollarda ularning xossalari belgilanadi. Masalan, quyidagi frame oynasiga StringGrid va BitBtn komponentlari joylashtirilgan.

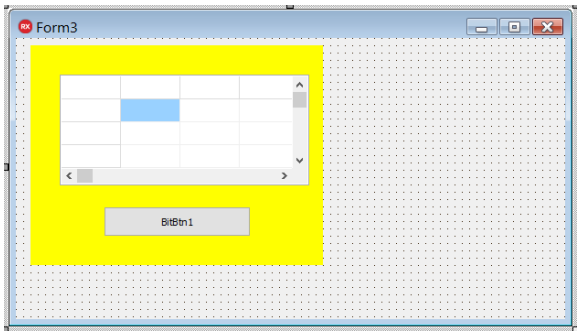


2. Kerakli komponentlar joylashtirilb o’zgarishlar kiritilgach frame saqlanadi va buning uchun uning ustida sichqonchaning o’ng tugmasi bosilib ochilgan kontekstli menudan “Add to Palette” buyrug’i beriladi va quidagi muloqot oynasi yordamida nomi, ikonkasi va joylashadigan joyi aniqlanadi.



3. “Ok” buyrug’i beriladi.

Endi biz bemalol belgilangan ilovada yarilgan frameni ko’ra olamiz va uni xoxlagancha formaga joylashrib bilamiz. Xulosa sifatida biz shablon yaratganimizni eslatin o’tmoqdamiz. Masalan, quyida forma3 oynasida shu holat o’z aksini topgan.



TMainMenu komponenti dasturning bosh menyusini aniqlaydi. Formaga bir necha kompenentni joylashtirish mumkin ammo ulardan faqat formaning “menu” xossasida ko’rsatilgani dastur uchun bosh menyu vazifasini bajaradi. Formada loyihalashni bajarayotgan vaqtda bu komponent ko’zga tashlansada dasur bajarilish

vaqtida u “ko’rinmas” holda bo’ladi. Komponent formaga joylashtirilgach unung uchun menyular yartiladi.

Buning uchun quyidagi varianlardan birini tanlash kifoya:

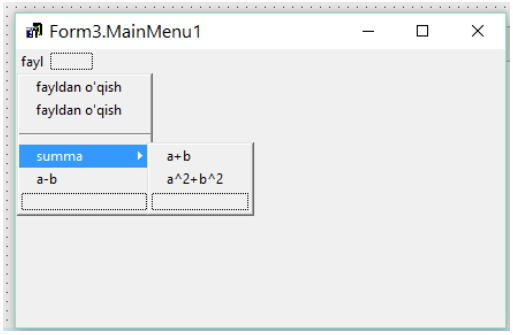
- komponent satxida sichqonchaning chap tugmasini ikki marta bosish;
- komponent satxida o’ng tugma yordamida kontekstli menu ochib undan “Menu Designer” bandini tanlash;
- komponentni tanlab ob’ektlar inspektorining “items” bandida sichqonchaning chap tugmasini ikki marta bosish;

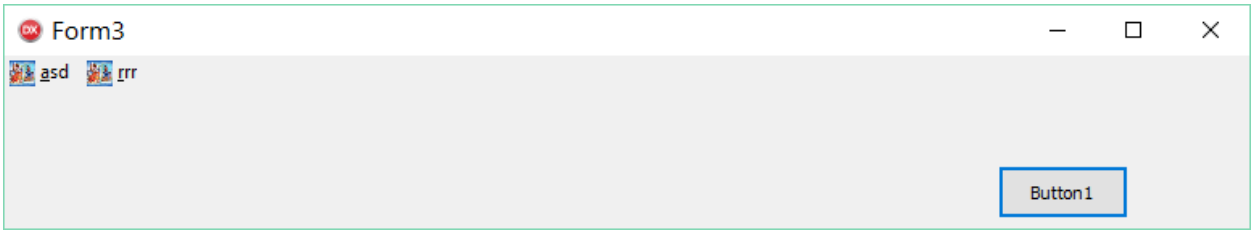
qaysi holat tanlanishidan qat’iy nazar bizga quyidagi oyna tavsiya etiladi:

Endi menyu yaratishning jihatlariga to’xtalib o’tamiz:

1. Menyu bandiga nom berish ucun ob’ektlar inspektorining “Caption” bandiga nom yozilib “Enter” tugmasi bosiladi. (masalan, “faylga yozish”)
2. Har bir menyuning tarkibida ichki menyular(yoki ostki menyular) bo’lishi mumkin(ichki menu yaratish uchun kerarki menyuni tanlab “Ctrl+->” tugmalarini bosish yetarli.
3. Menyuni qismlarga bo’lish uchun “Caption” xossasida “-“ belgisini joylashtirish zarur.
4. Menyuda tez ishga tushirish tugmasi bo’lishi uchun nom berish jarayonida kerakli simvoldan avval “&” belgisini joylashtiramiz. (masalan, “cho&p etish”)
5. Menyuga kichik hajmdagi ikonka ko’rinishdagi rasmlarni ham joylastirish mumkin. Buni amalga oshirishning yo’llaridan biri bu menyu yaratish jarayonida ob’ektlar inspektorining “bitmap” bandidan “Picture Editor” yordamchi oynasini ochib “Load” buyrug’i orqali kerakli rasmni tanlashdir.

6. Menu to’liq fa’oliyat ko’rsatishi uchun kerakli bandlar uchun ularga tegishli buyruqlar tizimi (kodlar) beriladi.





Label komponenti. Komponentning Caption xossasida turli maqsadlarda foydalanadigan matnli ma’lumotlar joylashishi mumkin.

Quyidagi jadvalda komponentning ayrim xossalarini keltiramiz:

Xossa	Xossa tavsifi
FocusControl	Tez ishga tushirish tugmasi bilan bog’liq bo’lgan komponentning nomi ko’rsatiladi. Masalan, Label komponentning Caption xossasida &Label yozuvi joylashtirilsa, va FocusControl xossasida Edit1 komponentining nomi ko’rsatilsa edit1 ga kiritish fokuci berilishi aniqlanadi. Misol keltirib o’tamiz: Label1. Caption:='&Label';Label1. FocusControl:=Edit1; Izoh: bunga faqat komponentning ShowAccelChar xossasining qiymati “True” bo’lganda erishish mumkin.
Color	Komponent rangini aniqlaydi. (misol: Label1. Color:=clyellow;)
Transparent	True va false qiymatlarida komponentning shaffofligi yoki berilgan rang bilan bo’yalishini ta’minlaydi. Bundan rasm ustiga matn joylashtirishda ustalik bilan foydalanish mumkin.
…	

Edit komponenti. Asosiy xossasi text bo’lib unda ma’lumot joylastiriladi. Masalan, Edit1. Text:=’Delphi dasturlash tili’;

Quyidagi jadvalda komponentning ayrim xossalarini keltiramiz:

Xossa	Xossa tavsifi
AutoSelect	Kiritish fokusu komponent uchun aniqlangan bo’lsa, komponentda joylashgan matnni to’liq belgilanishini ta’minlaydi. Loyihalashtirish jarayonida xossa uchun “true” yoki “false” qiymat belgilanishi mumkin. Dastur bajarilishi jarayonida esa Edit1. AutoSize:=true; (Edit1. AutoSize:=false;) buyruqlari yordamida bunga erishish mumkin.
AutoSize	Uning qiymati “True” va BorderStyle qiymati bsSingle bo’lsa, Font. Size (shrift parametrlari) xossasi o’zgarganda komponentning balandligi avtomatik o’zgaradi.
Color	Komponent rangini aniqlaydi. (misol: Edit1. color:=clYellow;)
CharCase	Matn holati ecNormal, ecLowerCase va ecUpperCase qiymatlardan biri xossada belgilansa, matn mos ravishda normal, kichik va bosh shriftlar bilan yoziladi.
MaxLength	Matn maydoni uzunligi, ya’ni nechta belgidan iborat bo’lishligi integer tipi orqali berilisi mumkin. (misol: Edit1. MaxLength:=10;)
Hint	Eslatmamatni. Loyihalash va dastur bajarilishi jarayonida belgilanishi mumkin. (misol: edit1. Hint:='dastur natijasi'; edit1. showhint:=true;edit1. Font. Color:=clred;)
Visible	True qiymatida forma satxida ko’zga tashlanadi, false holatida “yashirin” holatda bo’ladi.
PasswordChar	Matndagi barcha simvollar o’rniga bu xossada belgilangan simvolni aks ettirishga xizmat qiladi
…	

Memo komponenti. Qatorlari nomerlanlgan (nomerlash 0 dan boshlanadi) ko'p qatorli matnlarni aks ettiruvchi, kirituvchi va tahrirlovchi matnli maydon komponenti. Asosiy xossasi – Lines.

Memo komponenti orqali ma'lumotlarni kiritishga mo'ljallangan kodni mustaqil tahlil qilib ko'rib kerakli xulosa chiqaring:

```
procedure TForm5. Button1Click(Sender: TObject);
var ss:string;
begin ss:=Memo1. Lines[1]; ShowMessage(ss); end;
end.
```

Memo komponentida matnli ma'lumotlarni akslantirishga doir dasturni mustaqil tahlil qilib ko'rib kerakli xulosa chiqaring:

```
procedure TForm5. Button1Click(Sender: TObject);
var ss:string; a:char;
begin
 ss:='';
 for a:='a' to 'z' do
 ss:=ss+a;
 memo1. Lines. Add(ss);
end;
end.
```

Memo komponentida "surgich yo'lagini" joylashtirish uning ScrollBars xossasiga maxsus qiymatlardan birini (sshorizontal, ssNone, ssVertical,ssBoth) berish bilan amalga oshiriladi. Masalan, Memo1. ScrollBars:=sshorizontal;

TButton komponenti. Bu component asosan dasturda turli boshqarish yoki hodisalarni amalga oshirish maqsadida ishlatiladi. Hodisalar ko'pincha OnClick buyrug'i yordamida tashkil etiladi. Buning uchun formaga joylasttirilgan komponent uchun sichqonchadan foydalanib quyidagi protsedurani yaratish lozim:

```
procedure TForm5. Button3Click(Sender: TObject);
begin
. . .
```

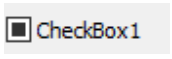
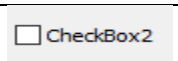
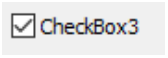
end;

end.

Albatta bu component ham turli xossalarga ega. Biz bu yerda ulardan ayrimlarini eslatib o’tamiz.

1. Cancel xossasi “True” qiymatiga ega bo’lsa, Onclick hodisasi “Esc” tugmasini bosganda amalga oshadi.
2. Default xossasi “True” qiymatiga ega bo’lsa, Onclick hodisasi “Esc” tugmasini bosganda amalga oshadi.
   Izoh: har ikki xossa “True” qiymatiga ega bo’lsa hodisa ikki holatda ham amalga oshadi.
3. TButton tugmasi Windows operatsion tizimi elementi bo’lib uning rangi yoki unga mos shrift rangini o’zgartirish faqat Windows palitrasi yordamida o’zgartirish mumkin.

TCheckBox komponenti. Bu komponent foydalanuvchiga kerakli varianlardan birini tanlash imkoniyatini beradi. Holatga ko’ra “bayraqcha” o’rnatish yoki uni olib tashlash yo’li bilan dastur bajarilshi kerakli yo’nalishga “yo’naltiriladi”. Asosiy xossalari “Checked” va “Stsate” xisoblanadi.

	State xossasi holati	**Komponentning ko’rinishi**	**Unga nisbatan bajaraladigan amal**
1	cbGrayed		Bayraqcha o’rnatilmasligi, o’rnatilishi va umuman o’zgarishsiz foydalanishi mumkin. (3 holatdan biri orqali)
2	cbUnchecked		Bayraqcha o’rnatilmasligi yoki o’rnatilishi orqali foydalanishi mumkin. (2 holatdan biri orqali)
3	cbChecked		

```
procedure TForm5. Button2Click(Sender: TObject);
begin
case CheckBox1. state of
cbChecked: Memo1. lines. Add('a');
```

cbUnchecked: Memo1. lines. Add('b');

cbGrayed: Memo1. lines. Add('c');

END;

end;

TRadioButtonkomponenti. TCheckBox komponenti kabi qo'llanuvchi bo'lib, undan farqli formaga joylashtirilgan bir necha komponentdan faqat biri tanlanishi mumkin. Formaga 3-ta RadioButton,1-ta Button va Memo komponentlarini joylashtirib Button uchun quyidagi kodni yarating va TRadioButton komponentini vazifasi to'g'risida kerakli xulosa chiqring.

procedure TForm5. Button1Click(Sender: TObject);

begin

Memo1. Clear;

if RadioButton1. Checked then

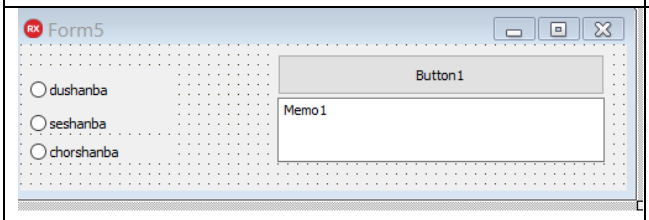
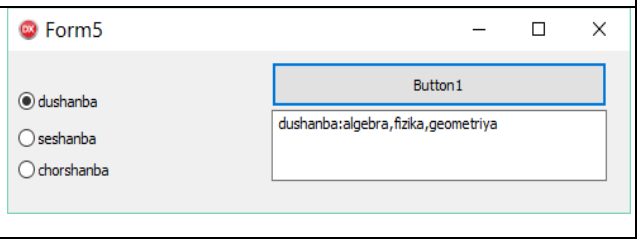
begin Memo1. Lines. Add('dushanba:algebra,fizika,geometriya');end;

if RadioButton2. Checked then

begin Memo1. Lines. Add('seshanba:biologiya,informatika,tarix');end;

end;

end.

Forma oynasi	RadioButton1 tanlangandagi natija:
	

ListBox komponenti. Maxsus ro'yxat sifatidashakllnuvchikomponent bo'lib, foydalanuvchiundanro'yxatningbo'limlarisifatidafoydalanishimumkin. Bo'limlar matn yoki rasmlardan iborat bo'lishi mumkin.

Komponentning ayrim xossalarini jadvalda va ayrimlariga misollar keltiramiz:

AutoCompete	True qiymatida ro'yxatdan element tanlanishi foydalanuvchi tomonidan kiritilgan birinchi simvol bo'yicha amalga oshiriladi.

BorderStyle	bsNone –qiymatida komponent chegara chizig’isiz, bsSingle qiymatida chiziq qalinligi 1 ga teng.
Canvas	Cizish maydoni.
Columns	Ro’yxatdagi ustunlar soni
ItemIndex	Kiritish fokusiga ega element indexi
Items	Ro’yxatdagi elementlar to’plami
MultiSelect	Ro’yxatning bir necha bandini tanlashga ruxsat berish yoki bermaslik

Loyihalash jarayonida ro’yxatni komponentning items xossasi bandidan (…) belgi yordamida maxsus oyna ochib shakllantirish mumkin. Dastur bajarilishida maxsus buyruqlar yordamida (masalan, ListBox1. Items. **Add**('Alisher'); ListBox1. Items. **Add**(Edit1. Text);) bu vazifani bajarish mumkin.

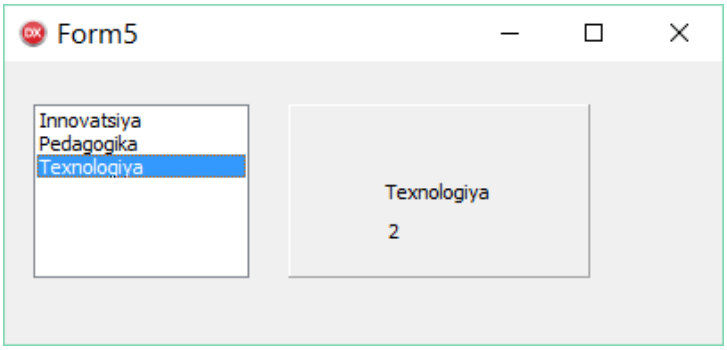
ListBox1. **Clear**; kopmponentdagi ro’yxatni to’la tozalashda, ListBox1. **DeleteSelected esa belgilangan satrni o’chirishda foydalanishi mumkin.**

ListBox1. Sorted:=true; hodisasi ro’yxatni alfavit tartibida joylashtirish imkoniyatini beradi.

Quyida keltirilgan dasturni tahlil qilib **ItemAtPos xossasi vazifasini aniqlang.**

```
procedure TForm5. ListBox1MouseUp(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);
var Point : TPoint; Nomer : Integer;
begin
 Point. X := X; Point. Y := Y;
 Nomer := ListBox1. ItemAtPos(Point, True);
 Label1. Caption := IntToStr(Nomer);
 Panel1. Caption := ListBox1. Items[ListBox1. ItemIndex];
end;
```

Olinadigan natija:



listBox komponenti bilan bog'liq hodisalarga misollar keltirgan holda Delphi dasturi imkoniyatlari bilan tanishtirishga harakat qilamiz:

- ***OnMouseUp hodisasi sichqoncha ko'rsatgichi komponent sohasida "qo'yib" yuborilganda yuz beradi.***

procedure TForm5. ListBox1MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

BeginShowMessage('dasturlash olami juda ajoyib olam. . . ');end;

end.

- ***OnMouseEnter hodisasi sichqoncha ko'rsatgichi komponent sohasiga "kirishi" bilan yuz beradi.***

procedure TForm5. ListBox1MouseEnter(Sender: TObject);

begin

ShowMessage('dasturlash olami bilan qiziqan shaxs uni tark eta olmaydi. . . ');

end;

- ***OnMouseUp hodisasi sichqoncha ko'rsatgichi komponent sohasini "tark" etganida yuz beradi.***

procedure TForm5. ListBox1MouseLeave(Sender: TObject);

begin

ShowMessage('dasturlash olami bilimli shaxslar uchun yaratilgan olam. . . ');

end;

ComboBox komponenti. ComboBox komponentasi Combination Boxes so'zidan qisqartirilgan bo'lib ListBox komponentasiga o'xshashdir. Lekin kombinasiyalashgan darcha kam joyni talab qiladi va bir vaqtning o'zida bitta

elementni ko’rsatadi. Demak, ListBox komponentasi bilan ComboBox komponentasining farqi bir vaqtning o’zida qancha element ko’rsata olishda ekan. Joydan tejash maqsadida ComboBoxdan foydalanish qulayroq bo’lsa, ko’proq elementni ko’rsatish uchun ListBoxdan foydalanish qulay. Bu komponentda ma’lumotlar, ochiluvchi ro’yxat ko’rinishida turadi. Listbox komponentidan farqi bu komponentda ro’yxatning faqat bitta elementi ko’rinib turadi. Joyni tejash maqsadida bu komponentdan foydalangan ma’qul. Quyidagi jadvalda uning asosiy xususiyatlari keltirilgan.

Color	Komponent ran[H1]gi
Font	Ro’yxatdagi ma’lumotlar shrift parametrlari
Items	Ro’yxatga ma’lumotlar kiritish va undan foydalanish
ItemIndex	Ro’yxatga chiqadigan element indeksi, agar -1 bo’lsa bo’sh turadi
Text	ItemIndex=-1 bo’lsa Text da kiritilgan ma’lumot chiqadi
Sorted	Ro’yxatga ma’lumotlarni saralab joylashtirish

procedure TForm5. Button1Click(Sender: TObject);

begin ComboBox1. Sorted:=True; end;

Quyida ayrijm komponentlar uchun umumiy bo’lgan xossalardan ayrimlarini keltirib o’tamiz:

Xossa	**Qiymat**
Height, Width	Komponentning pikseldagi balandlik va qalinlik qiymatlari
Left	Komponentning shaklga nisbatan chap tomonda joylashishi
Top	Komponentning shaklga nisbatan yuqorida joylashishi
Align	Komponentning shaklga nisbatan: alTop (yuqorida), alBotton (pastda), alLeft (chapda), alRight (o’ngda), alNone (Left va Top hossalar bilan aniqlanadi) joylashishi
Ctl3D	Agar True bo’lgan holda, komponent hajmli ko’rinishga ega bo’ladi, aks holda - yassi
BevelEdges BevelKind BevelInner BevelOuter	Komponent oynasining chegarasini belgilaydi

Color	Komponening asosiy rangi
Font	Tashqi ko’rinish (o’lcham, rang va h. k.) yozuv uchun shriftlar
Caption	Matn satrini boshqaruv elimenti bilan bog’laydi: Tugmadagi yozuv, belgi matni, forma uchun oyna sarlavhasi
Text	Formani ochish vaqtida ko’rinuvchi matn oynasi maydonining tarkibi
Visible	Agar bu hossa qiymati True bo’lsa, komponentni formada ko’rish mumkin, agar qiymat False bo’lsa, komponent ko’rinmaydi
Enabled	Agar hossa False qiymatiga ega bo’lsa, komponent noaktiv holda bo’ladi, ya’ni bu tugmada sichqoncha tugmasini bosish hech qanday natija bermaydi. Bu holda tugmadagi yozuv kulrang bo’ladi. Agar hossa True qiymatiga ega bo’lsa, komponentdan dasturni ishlatish jarayonida foydalanish mumkin.
AutoSize	Agar hossa True qiymatiga ega bo’lsa, u holda komponent oynasining o’lchami shrift o’lchami o’zgarganda yoki boshqa komponentlar qo’shilganda, avtomatik holda o’zgaradi.
Hint ShowHint	Agar ShowHint hossasi True qiymatiga ega bo’lsa, Hint ihossasida mavjud bo’lgan izoh paydo bo’ladi
Focused	Agar hossa True qiymatiga ega bo’lsa, ushbu komponentga diqqat qaratilgan bo’ladi. Formadagi elimentlar soni bir nechta bo’lgan holatda, faqat bitta eliment bu hossani olishi mumkin.
TabStop	Agar hossa True qiymatiga ega bo’lsa, bu komponentga Tab tugmasi orqali diqqat qaratiladi
TabOrder	Tab tugmasi komponentini tanlashning tartib raqami

Xossalardan dasturda foydalanishga misollar.

Dasturda hossalarni o’rnatish	**Natija:**
Edit1. AutoSize:=True;	Matn oynasi o’lchamini avtomatik tarzda o’zgartirish;
Edit1. Text:=FloatToStr(a);	Xaqiqiy o’zgaruvchi a ning qiymatini matn oynasida ko’rinishi
Button1. TabOrder:=3;	Tab tugmasini uch marta bosgandan so’ng, Button1 ga diqqat qaratiladi.

Mavzuga oid savollar

1. Label, Edit, Memo matn komponentlari va Button tugmachasininh qanday umumiy xossalari bor?
2. Boshlang'ich forma ilovasini qanday yaratiladi?
3. Tanlash tugmalari qanday o'rnatiladi?
4. ListBox va ComboBox komponentalarining vazifalarini ayting.

2.3. ADDITIONAL KOMPONENTLAR PALITRASI.

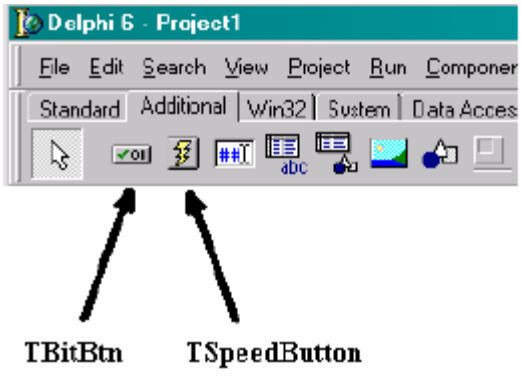
Reja:

1. TSpeedButton va TBitBtn tugmalari.
2. Markirovka qilingan ro'yxat (TCheckListBox)
3. TscrollBox komponenti.

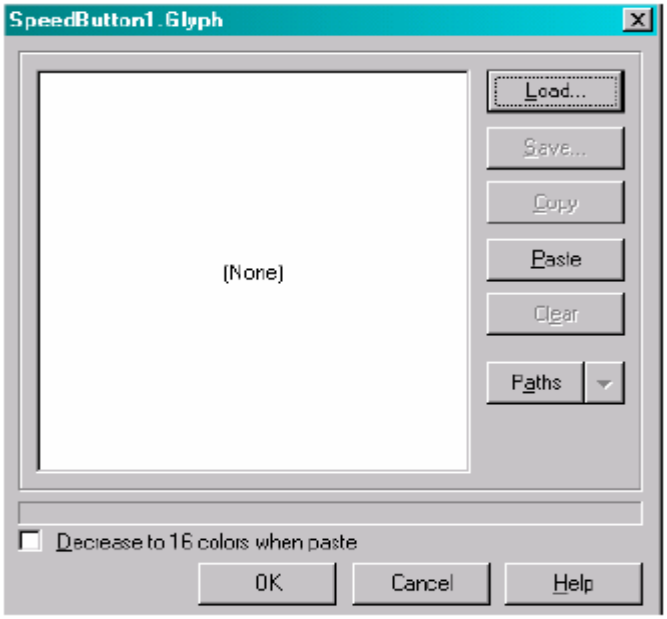
Tayanch iboralar: BitBtn, MaskEdit, StringGrid, MaskEdit, CheckListBox, DrawGrid, Image, Shape va boshqa komponentlaridan foydalanish. Komponentlar xossalalri.

TSpeedButton va TBitBtn tugmalari.

Bu tugmalar ***TButton*** vazifalarini bajaradi. YAgona farqi matndan tashqari racmlarni ham aks ettiradi. ***TSpeedButton*** tugmasi fokus olmaydi. Bu shuni bildiradiki, agar matn qatorida satr terib, bu tugma bosilsa, shu hodisa qayta ishlangandan so'ng fokus yana matn qatoriga qaytib keladi. TAB tugmasi bilan bu tugmani ajratib bo'lmaydi.

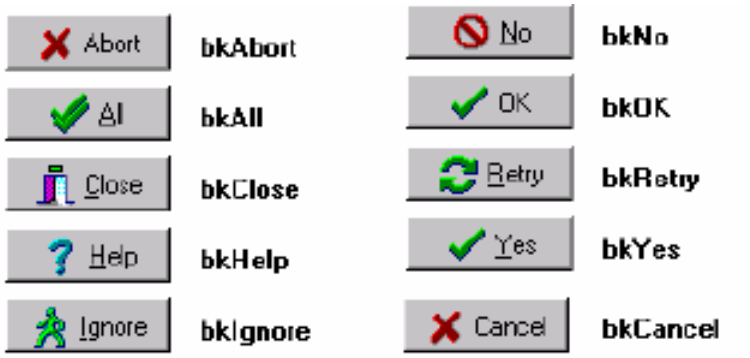


Tugmaga rasm o'rnatish uchun ikki marta *Glyph* xossasi qatorida chertish lozim. Natijada rasm paydo bo'lgan yuklash oynasida *Load* tugmasini bosish lozim. Ko'p rasmlar Program Files\Common Files\Borland Shared\Images \Buttons katalogida joylashgandir.



TBitBtn va ***TSpeedButton*** tugmalari deyarli bir xil xossalarga egadir. Ular uchun umumiy *Layout*, xossasi rasm va matn o'zaro joylashuvini o'zgartirishga imkon beradi. Quyidagi rasmda har xil qiymatlar mos variantlari ko'rsatilgan

TBitBtn tugmasining yana bir xossasi *Kind* bo'lib oldindan tayyorlangan standart tugmalarni tanlash imkonini beradi. Quyidagi rasmda standart tugmalar va ularga mos qiymatlarni ko'rish mumkin.

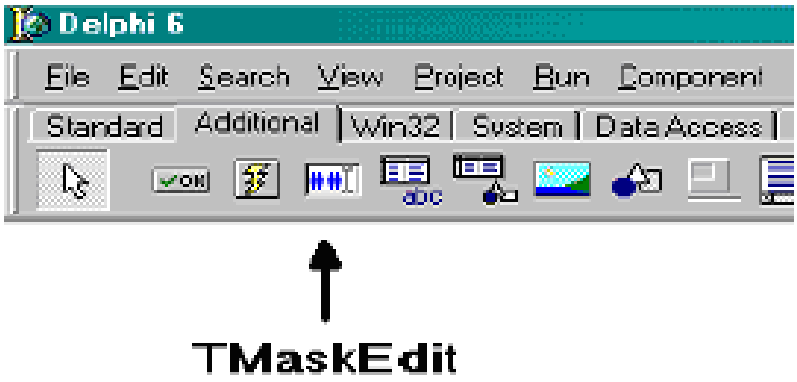


Yana bir xossa ModalResult – dialog oynasi uchun tugma qaytaradigan natijani tanlashga imkon beradi.

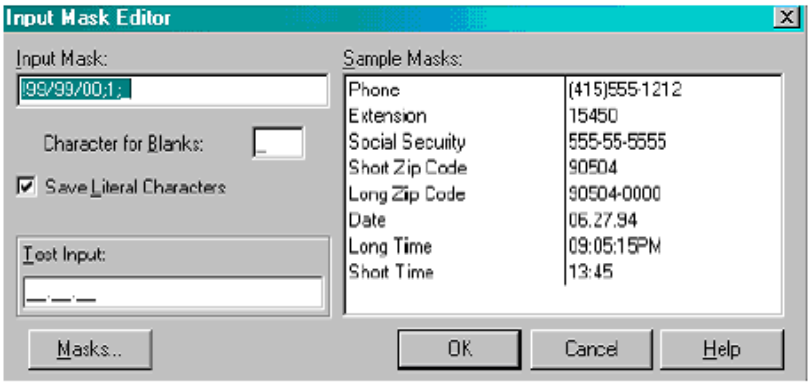
TSpeedButton tugmasining *GroupIndex* xossasi tugmalarni guruhlashga imkon beradi. Buning uchun bir guruhga tegishli tugmalarning *GroupIndex* xossasi bir xil qiymatga masalan 1 ga teng bo'lishi kerak. Guruhlangan tugmalarning biri bosilsa, qolganlaridan ajralib qoladi. Buning uchun *Down* xossasi qiymati *true* ga teng bo'lishi kerak.

Maskalangan kiritish katori (TMaskEdit)

Bu komponent ma'lum formatga mos satr kiritishga imkon beradi.



Asosiy xossasi *EditMask* bo'lib, shu xossa qatorida ikki marta chertilsa kiritish muharriri ochiladi.



Input Mask qatorida maska kiritish mumkin. *Test Input*, qatorida maskani testlash mumkin.

Maska terish osondir. Agar qator to'rt raqamli son, tere va uch raqamli sondan iborat bo'lishi kerak bo'lsa, *Input Mask* qatoriga 9999-999 kiritish mumkin.

Siljitish yo'lchasiga ega panelь (TScrollBox)

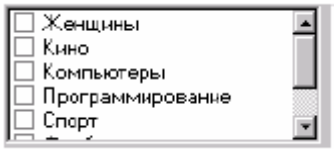
TScrollBox komponentasining oddiy Panel komponentasidan farqi siljitish yo'lchasiga ega bo'lishi mumkinligidir.

 - TScrollBox

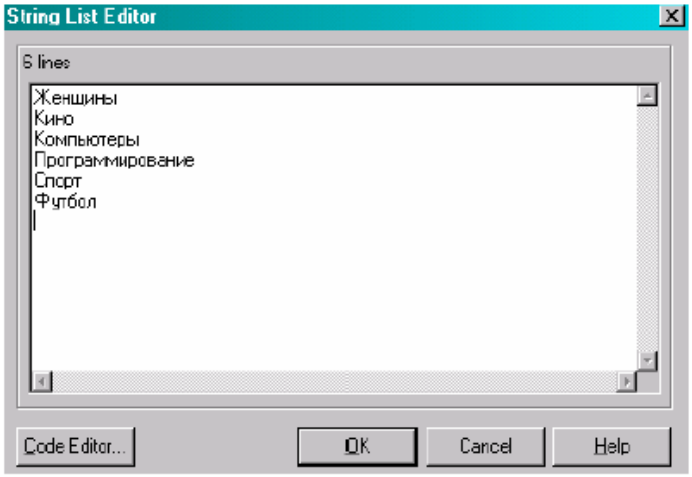
Formaga ***TScrollBox*** komponentasini o'rnatib, uning ichiga (***TImage***). komponentasini o'rnating. Endi *Image1* ga katta rasm joylab, *AutoSize* xossasiga *true* qiymatini bering. Agar *Image1* komponentasida rasm kattaligini olib *ScrollBox* chegarasiga sig'may qolsa, siljitish yo'lchalari paydo bo'ladi.

Markirovka qilingan ro'yxat (TCheckListBox)

TCheckListBox komponentasi ***TListBox***, komponentasiga juda o'xshash, faqat har bir ro'yxat yonida ***TCheckBox*** komponentasidagi kabi ajratish to'rtburchagi mavjuddir.

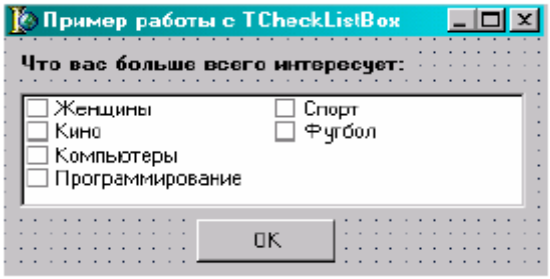


Ro'yxat kiritish uchun *Items* xossasi qatorida ikki marta chertish lozim.



TCheckListBox yana bir xossasi – *columns*, ya'ni ustunlar sonidir. Agar bu xossa qiymati birdan katta bo'lsa va ro'yxat bir ustunga sig'masa, ko'rsatilgan sonli ustunlarga ajratiladi.

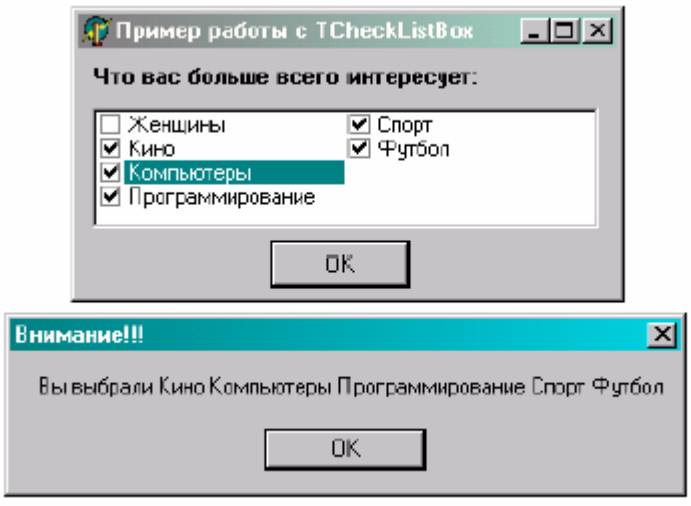
Quyida shu elementdan foydalanilgan dastur formasini keltiramiz.



OK tugmasining *OnClick* hodisasi uchun quyidagi protsedurani kiritamiz:

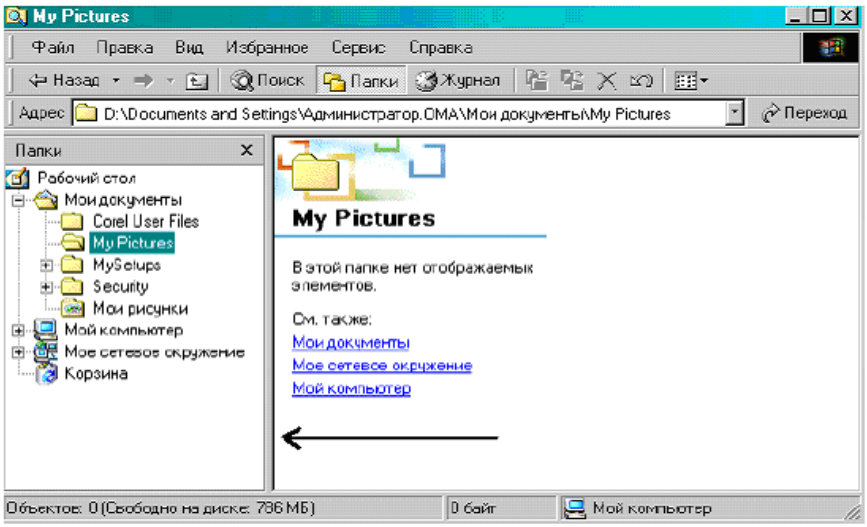
```
procedure TForm1. OKButtonClick(Sender: TObject);
var
i:Integer;
Str:String;
begin
Str:='Siz tanladingiz ';
for i:=0 to CheckListBox1. Items. Count-1 do
if CheckListBox1. Checked[i] then
Str:=Str+CheckListBox1. Items[i]+' ';
Application. MessageBox(PChar(Str), 'Dikkat!!!');
end;
```

Dastur ishlash natijasiga misol.

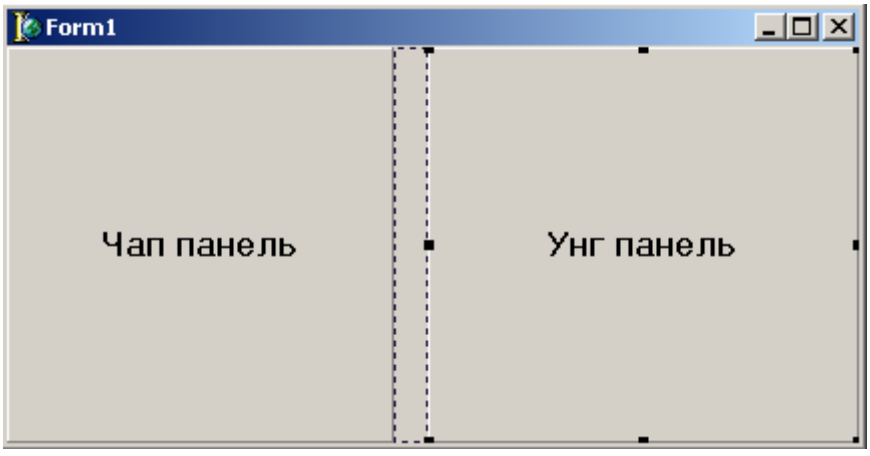


Ajratish yo'lchasi (TSplitter) 

Agar Windows Explorerni ochib ko'rsak oynasi ikkiga ajralgan bo'lib, o'rtasida siljitish mumkin bo'lgan yo'lchani ko'ramiz. Mana shu effektni ***TSplitter*** komponentasi yaratishga imkon beradi.



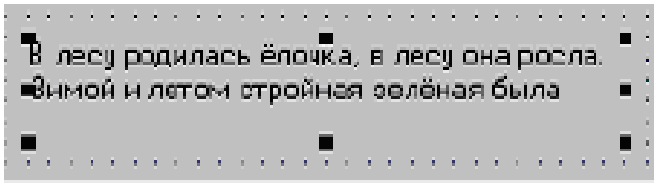
TSplitter komponentasidan foydalanishga misol. Formaga panel (***TPanel***) komponentasini joylashtirib *Align* xossasiga *alLeft* qiymat beramiz va *Caption* xossasiga «CHap panelь » qatorini kiritamiz. Formaga ***TSplitter*** joylashtirib *Align* xossasiga *alLeft* qiymat beramiz. YAna bir panelь joylashtirib *Align* xossasiga *alClient* qiymat beramiz va *Caption* xossasiga «Ung panelь » qatorini kiritamiz. Natija rasmda ko'rsatilgan



Dasturni ishga tushirib, ajratish yo'lchasi sichqoncha Bilan harakatlantirilsa panellar kattaligi o'zgaradi.

Ko'p qatorli matn (TStaticText)

Ko’pincha dasturda bir necha qatorli matn chiqarishga to’g’ri keladi. Buning uchun formaga bir necha ***TLabel*** komponentasini o’rnatish mumkin. Lekin osonroq ***TStaticText*** komponentasini o’rnatib *AutoSize* xossasiga false qiymatini berishdir. Rasmda shu komponentadan foydalanishga misol keltirilgan.



Parametrlar muharriri (TValueListEditor)

Bu komponenta ob’ektlar inspektoridagi kabi xossalar muharririni yaratishga imkon beradi.

Asosiy xossalari:

• *DefaultColumnWidth* – ustunlar ko’zda tutilgan kengligi;

• *DefaultColumnHeight* – ustunlar ko’zda tutilgan balandligi;

• *DisplayOption* – komponentani akslantirish optsiyalari;

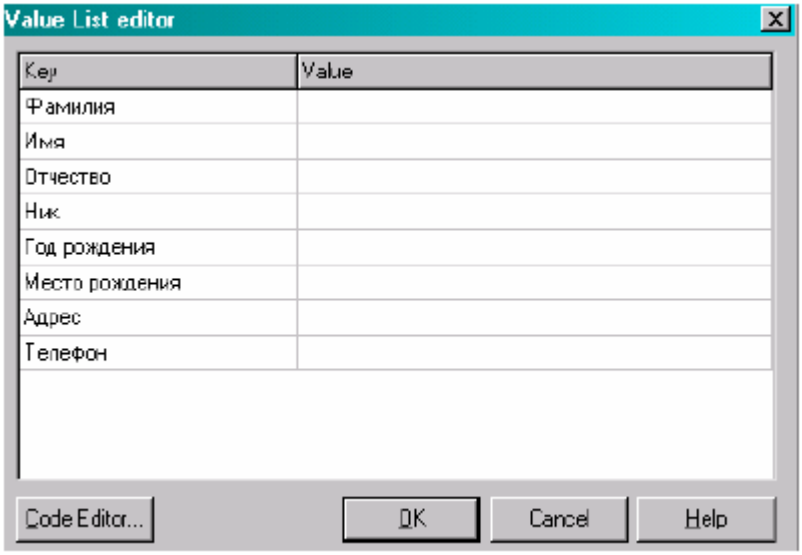
• *TitleCaptions* – sarlavhalar nomlari. Agar ikki marta chertilsa oddiy matn muharriri chiqadi.

• *FixedColor* – fiksirlangan ustun rangi.

• *FixedCols* – fiksirlangan ustun indeksi.

• *KeyOption* – kalit maydon optsiyalari

• *Strings* – xossalar nomlari. SHu qatorga ikki marta chertilsa xossalar muharriri chiqadi



Formaning OnShow hodisasi uchun quyidagi protsedura yaratamiz:

procedure TForm1. FormShow(Sender: TObject);

begin

ValueListEditor1. ItemProps[6]. EditStyle:=esPickList;

ValueListEditor1. ItemProps[6]. PickList. Add('Moskva');

ValueListEditor1. ItemProps[6]. PickList. Add('Piter');

ValueListEditor1. ItemProps[6]. PickList. Add('Rostov-na-Donu');

ValueListEditor1. ItemProps[4]. EditMask:='99/99/9999';

end;

ItemProps. Xossasida ro'yxat elementlari xossasi joylashgan. Agar 3- element xossasini o'zgartirish lozim bo'lsa *ValueListEditor1. ItemProps[2]* yozish kerak.

EditStyle – xossasi tahrirlash uslubini o'rnatadi. (*ValueListEditor1. ItemProps[6] EditStyle*):= *esPickList* instruktsiyasi qator tugma bosilganda chiquvchi qatorga aylantiradi.

ValueListEditor1. ItemProps[6]. PickList. Add (tekst elementa) buyrug'i oltinchi qatorga satr qo'shadi.

EditMask – xossasi kiritish maskasini yaratishga imkon beradi.

Mavzuga oid savollar

1. TSpeedButton va TBitBtn tugmalarining xossalalrini sanab bering.

2. Markirovka qilingan ro’yxat qaysi komponent orqali amalga oshiriladi?
3. TscrollBox komponentining vazifalarini aytib bering.

3-BOB. DELPHIDA DASTURLASH

3.1. DELPHI DASTURLARI STRUKTURASI. LOYIHA VA MODUL.

Reja:

1. Delphi dasturlari strukturasi.
2. Loyiha va modul.

Tayanch iboralar: Bo‘sh forma va uning modifikasiyasi. Delphida nomlanishlar, forma xossalarini o‘zgartirish. Formaga yangi komponent joylashtirish va unda komponent xossalaridan foydalanish, Xodisa tushunchasi. Loyiha va modul strukturasi. Dastur elementlari (alfavit, identifikatorlar, doimiyliklar, ifodalar va amallar).

Modul quyidagi strukturaga ega:

unit <nom>

interface

<Interfeys qismi>

implementation

<Bajariluvchi qism>

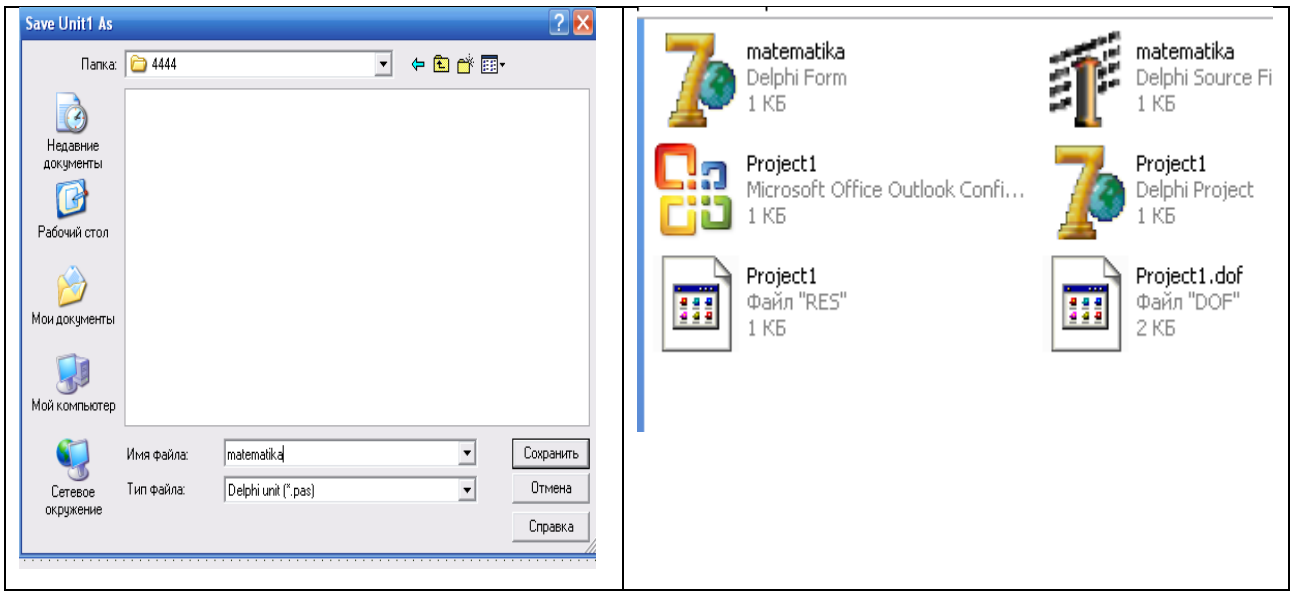
Initialization

<initsializatsiya qismi>

end.

Bu erda unit modul boshlanishini bildirsa <nom> uning nomini anglatadi.

Aniqroq bo’lishiuchun alihida ishchi papka yarating va Delphi dasturini ishga tushiring, so’ngra SAVE AS, yoki SAVE UNOT1 AS, buyrug’i yordamida loyixani saqlang papkada xosil bo’lgan fayllarga e’tibor bering.



Hajm jihatidan kata bo’lgan dasturlarda ishni osonlashtirish maqsadida nom diskdagi fayl nomi bilan bir xil bo’lishi lozim.

Masalan, modul nomi

unit matematika;

bo’lsa, modulning matni shu nom bilan pas kengaytmali faylda joylashadi. Nom boshqa modullar bilan bog’lanishi va asosiy dasturda foydalanishi uchun beriladi. Bu aloqa

uses <modullar ro’yxati> jumlasi bilan o’rnatiladi.

Masalan,

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, matematika;

Modullar bir-biridan foydalana oladilar. Uses jumlasi modullarda interface xizmatchi so’zidan keyin, yoki implementation xizmatchi so’zidan keyin yoki ikkalasidan keyin ham yozilishi mumkin. (Demak, modulda ikkita Uses jumlasi qatnashishi ham mumkin.)

Interfeys qismi interface xizmatchi so’zidan keyin beriladi. Bu qismda modulning barcha global ob’ektlari(tiplar, doimiyliklar, o’zgaruvchilar, funktsiya va protseduralar) beriladi.

Masalan:

```
Unit Cmplx;
Interface
 Type
Complex=record
                Re,im:real
          end;
  function AddC(x,y:compex):complex;
  function MulC(x,y:compex):complex;
```

Kelajakda boshqa modullarda uses cmplx jumlasini yozilishi bu modullarda Complex tipi va AddC hamda MulC protseduralaridan foydalanish imkoniyatini beradi. Bajariluvchi qism Implementation xizmatchi so’zidan keyin beriladi.

1. Delphi dasturlash tili qanday kengaytmali fayllar ishlatiladi?
2. Delphi oynasi qanday elementlardan tashkil topgan?
3. Tayyor dastur nechta asosiy etapdan o`tiladi?
4. Delphi tizimida ishga tushiriladigan modul strukturasi qanday ko`rinishda bo`ladi?
5. Oynaning bo’limlarini tushuntiring.

3.2. DELPHIDA TIPLAR, O‘ZGARMASLAR, O‘ZGARUVCHILAR VA STANDART FUNKSIYALAR.

Reja:

1. Delphida tiplar, ularning ahamiyati.
2. Butun tiplar: sodda (tartib va xaqiqiy) tiplar, mantiqiy va simvolli tiplar, tip-diapazon, vaqt-sana tipi.
3. Delphida simvolli va satriy tiplar. Simvolli va satriy tiplarning berilishi, ular bilan bajariladigan amallar. Simvolli va satriy kattaliklar.
4. Delphi dasturlash tilida o‘zgarmaslar, o‘zgaruvchilar va standart funksiyalar

Tayanch iboralar: tiplar, butun tiplar, sodda (tartib va xaqiqiy) tiplar, mantiqiy va simvolli tiplar, tip-diapazon, vaqt-sana tipi, simvolli va satriy tiplar, simvolli va satriy tiplarning berilishi, simvolli va satriy kattaliklar, o‘zgarmaslar, o‘zgaruvchilar va standart funksiyalar.

Delphida tiplar, ularning ahamiyati.

Delphi tilida dastip ishlashi mobaynida qiymati o’zgarmaydigan identifikatorlar **o’zgarmaslar** deyiladi va ular dastipning bosh qismida **Const** so’zi bilan e’lon qilinib, unga aniq qiymat tenglashtiriladi.

Misol. Const aa1=2. 27;

Pi=3. 14;

radius=14;

Dastip ishlashi mobaynida qiymatlari o’zgarishi mumkin bo’lgan identifikatorga **o’zgaruvchilar** deyiladi va ular dastip bosh qismida **Var** so’zi bilan e’lon qilinadi. O’zgaruvchilar nomi keltirilib, ularning tiplari beriladi. O’zgaruvchilarning eng ko’p ishlatiladigan tiplari **butun, haqiqiy, belgili, qator** va **mantiqiy**dir. Ular mos

ravishda butun - **Integer**, haqiqiy - **Real**, belgili - **Char**, qator (matn) - **String** va mantiqiy - **Boolean** deb yoziladi.

Masalan:
```
Var   a, d1, alfa : Integer;
      c121, df : Real;
      Etx, xx : Char;
          St,Sw: String;
      fl : Boolean;
```

Mantiqiy o'zgaruvchilar faqat ikkita qiymat qabul qiladi: "True" (chin) va "False" (yolg'on).

Standart va nostandart matematik funktsiyalar

Funktsiya nomi	Tilda yozilishi	Ma'nosi
Sinx	SIN(x)	x ning sinusi
Cosx	COS(x)	x ning kosinusi
Lnx	Ln(X)	x ning natipal logarifmi
e^x	EXP(x)	Eksponenta
$\sqrt{x}$	SQRT(x)	Kvadrat ildiz
Arctgx	ARCTAN(x)	x ning arktangensi
|x|	ABS(x)	x ning moduli
x^2	SQR(x)	x ning kvadrati
a^b	EXP(b*LN(a))	a ning b chi darajasi

Nostandart matematik funktsiyalar.

$$1.\ Secx = \frac{1}{Sinx};\quad 2.\ Co\sec x = \frac{1}{Cosx};\quad 3.\ Tgx = \frac{Sinx}{Cosx};\quad 4.\ Arcctgx = Arctg\frac{1}{x};$$

$$5.\ Arc\sin x = Arctg\frac{x}{\sqrt{1-x^2}};\quad 6.\ Arc\cos x = Arctg\frac{\sqrt{1-x^2}}{x};\quad 7.\ Arc\sec x = Arctg\frac{1}{\sqrt{1-x^2}};$$

$$8.\ Arc\cos ecx = Arctg\sqrt{1-x^2};\quad 9.\ Log_a b = \frac{Lnb}{Lna};\quad 10.\ Padian = \frac{Gradius\cdot\pi}{180}$$

O'zgartirish funktsiyalari

Funktsiya	Qiymati
Chr(n)	Kodi n ga teng simvol
IntToStr(k)	Butun k ni tasvirlovchi satr
FloatToStr (n)	Haqiqiy n tasvirlovi satr
FloatToStrF(n, f, k,m)	Haqiqiy n tasvirlovi satr. Bunda: f - format; k - aniqlik; m - kasr qismidagi raqamlar soni
StrToInt (s)	Satrni butun songa o'tkazish
StrToFloat (s)	Satrni haqiqiy songa o'tkazish
Round (n)	Haqiqiy sonni yaxlitlash

Trunc (n)	Haqiqiy son kasr qismini olib tashlash
Frac(n)	Kasrli sonning kasr qismi
Int (n)	Kasr sonning butun qismi

Dastipda arifmetik va mantiqiy ifodalar o'zgaruvchi, o'zgarmas, standart funktsiyalar, qavslar va amal belgilari orqali tashkil qilinadi.

Ifodalarda hisoblashlar tartibi qavslar ichidagi ifodalar bajarilgandan keyin quyidagi tartibda bajariladi:

1. NOT amali;
2. *, /, DIV, MOD, AND;
3. +, -, OR;
4. taqqoslash belgilari: <, >, <=, >=, <>, =, IN.

Ifodadagi amal natijasi qanday tipda bo'lishi amallarda qatnashayotgan o'zgaruvchilarning tiplariga bog'liq. Agar ikkita o'zgaruchining tipi Integer yoki Real bo'lsi, amal natijasi ham Integer yoki Real bo'ladi. Agar biri Integer ikkinchisi Real bo'lsa natija Real bo'ladi. NOT, OR, AND va taqqoslash amallarining natijalari esa Boolean tipida bo'ladi.

Kompьyuter foydalanuvchi tomonidan qo'yilgan masalani aniq va tushunarli ko'rsatmalar berilgandagina bajara oladi. Bu ko'rsatmalar ma'lum bir ma'noni anglatuvchi so'zlardan iborat bo'lib, kompьyuterga qanday operatsiyani bajarish lozimligini bildiradi va bu ko'rsatmalarga **operatorlar** deyiladi. Operatorlar dastip ishlaganda ketma-ket ravishda bajariladi. Delphi tilida bir satrga bir necha operatorlarni yozish mumkin.

Ma'lumotlar tiplarini Delphi tilida umumiy holda ikkiga ajratish mumkin:

standart tiplar. Bu tiplar oldindan Delphi tili tomonidan aniqlangan bo'ladi;

dastipchi tomonidan kiritiladigan (aniqlanadigan) tiplar.

Standart tiplar tarkibiga quyidagilar kiradi: butun, haqiqiy, belgili (simvol), qator (strok), mantiqiy, ko'rsatgichli va variant.

Dastipchi tiplarni dastipning **Var** bo'limida o'zgaruvchilarni tavsiflashda aniqlaydi yoki maxsus tiplarni aniqlash uchun bo'lim bo'lgan -tiplarni tavsiflash **Type** bo'limida aniqlaydi.

Bu bo’lim umumiy holda quyidagicha bo’ladi.

Type

<tip nomi>=<tipning tavsifi>;

Misol:

```
Type
    TColor=(Red,Blue,Black);
Var Color1,Color2,Color3: TColor;
```

Type bo’limida dastipchi tomonidan yangi Tcolor nomli tip kiritilmoqda va u Red,Blue,Black mumkin bo’lgan qiymatlarni qabul qilishi mumkin.

Var bo’limida dastipchi tomonidan tipi aniqlangan uchta Color1,Color2,Color3 o’zgaruvchilar tavsiflanmoqda.

Bu o’zgaruvchilarni to’g’ridan to’g’ri quyidagicha ham tavsiflash mumkin.

```
Var Color1,Color2,Color3: (Red,Blue,Black);
```

Standart tiplarni Type bo’limida tavsiflash shart emas, ularni to’g’ridan to’g’ri Var bo’limida tavsiflash mumkin.

Delphida standart tiplarni quyidagicha klassifikatsiya qilish mumkin.

Oddiy

Tartibli

Butun

Belgi

Mantiqiy

Sanoqli

Chegaralangan

Haqiqiy

Qator

Struktipa

To’plam

Massiv

Yozuv

Fayl

Klass

Interfeys

Ko’rsatgichli

Protsedurali

Variant

Oddiy tiplarga tartiblashgan va haqiqiy tiplar kiradi. Tartiblashgan tiplar shu bilan xarakterlanadiki uning har bir qiymati o’zining tartiblangan nomeriga ega. Haqiqiy tip qiymatlari kasr qismidan iborat bo’lgan sonlardan iboratdir.

Tartiblashgan tiplarga butun, belgili, mantiqiy, sanoqli va chegaralangan tiplar kiradi.

Butun tiplar. Butun tiplar butun sonlarni tasvirlash uchun ishlatiladi. Jadvalda Delphida ishlatiladigan butun tiplar ro’yxati keltirilgan. Bu tiplar bir-biridan qiymatlar diapazoni va xotirada egallangan xotira hajmi bilan farqlanadi.

Tip formati	O’zgarish diapazoni	O’lcham (baytda)
Integer	-2147483648. . 2147483647	4
Cardinal	0. . 4294967295	4
Shjrtint	-128. . 127	1
Smallint	-32768. . 32767	2
Longint	-2147483648. . 2147483647	4
Int64	-263. . 263-1	8
Byte	0. . 255	1
Word	0. . 65535	2
LongWord	0. . 4294967295	4

Butun tiplar bilan quyidagi amallar bajarilishi mumkin:

- **+-qo’shish;**
- **- ayirish;**
- *** ko'paytirish;**
- **div-butun sonli bo’lish;**
- **mod-bo’lishdagi qoldiq.**

Butun sonlarga qo’llanadigan funktsiya va protseduralarning jadvalini ketirib o’tamiz:

Murojaat	Natija tipi	Foydalanish natijasi
Abs(x)	x	x-ning absolyut qiymati

Chr(b)	char	Simvol kodiga ko’ra simvolni qaytaradi
Dec (V [x,i]	-	Vx –ning qiymati i-ga, i qiymati ko’rsatilmaganda 1 ga kamaytiraladi
Inc (V [x,i]	-	Vx –ning qiymati i-ga, i qiymati ko’rsatilmaganda 1 ga orttiraladi
Hi(w)	byte	Argumentning eng katta baytini beradi
Hi(i)	byte	Argumentning uchinchi baytini beradi
Lo(l)	byte	Argumentning eng kichik baytini beradi
Lo(w)	byte	Argumentning eng kichik baytini beradi
Odd(l)	boolean	i-toq bo’lsa true qiymatini beradi
Random(w)	Parametrga mos	0. . w-1 orailiqdagi tasodufiy sonni beradi
Sqr(x)	x	Argumentning kvadratini beradi

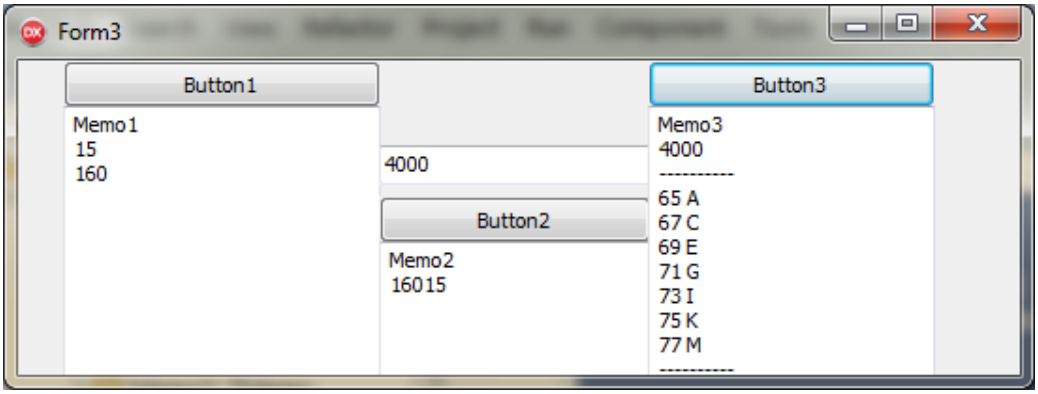
izoh: jadvalda b,w,I,l bilan mos ravishda Byte, Word, Integer, LongInt tipli (x-ko’rsatilgan barcha tiplar) ifodalar, vb,vw,vi,vl billan mos tipdagi o’zgaruvchilar ifodalangan.

Quyida ushbu standart funktsiya va protseduralarning ayrimlaridan foydalanishga misollar keltiramiz:

```
…. . .
Var d,i:word; x:integer;
implementation
{$R *. dfm}
procedure TForm3. Button1Click(Sender: TObject);
begin
 d:=strtoint(edit1. text); Memo1. Lines. Add(IntToStr(hi(d))) ;
 Memo1. Lines. Add(IntToStr(Lo(d))) ;
end;
procedure TForm3. Button2Click(Sender: TObject);
 var m:byte; ss:string;
begin
ss:=''; d:=strtoint(edit1. text);
 while d>256 do
 begin m:=d mod 256; ss:=ss+IntToStr(m); d:= d div 256; end;
 ss:=ss+IntToStr(d); Memo2. Lines. Add(ss);
end;
procedure TForm3. Button3Click(Sender: TObject);
 var b:byte;
begin x:=StrToInt(edit1. Text); memo3. Lines. Add(IntToStr(abs(x)));
memo3. Lines. Add('----------');
 b:=65;
 while( b<=77) do
```

```
begin memo3. Lines. Add(IntToStr(b)+' '+(chr(b))); inc(b,2); end;
memo3. Lines. Add('----------');
end;
end.
```

Olinadigan natija:



Mantiqiy tiplar. Mantiqiy tiplar chin (True) yoki yolg'on (False) qiymatning birini qabul qiladi. Jadvalda Delphida ishlatiladigan mantiqiy tiplar ro'yxati keltirilgan. Pascal dasturlash tilida faqat Boolean tipi mavjud edi,qolgan tiplar Windows bilan hamkorlik uchun kiritilgan. Bu tiplar tartibli tiplar qatoriga mansub.

Tip	O'lcham (baytda)
Boolean	1
ByteBool	1
WordBool	2
LongBool	4

Ular uchun quyidagi qoidalar o'rinlidir:

Ord(False)=0;

Ord(True)<>0;

Succ(False)=True;

Pred(True)=False;

Bu tiplar bilan quyidagi amallar bajarilishi mumkin:

- not (mantiqiy inkor);
- and (mantiqiy va);
- or (mantiqiy yoki);
- xor ((mantiqiy yokini inkori);

Butun (mantiqiy) tiplar bilan bajaraladigan mantiqiy amallar.

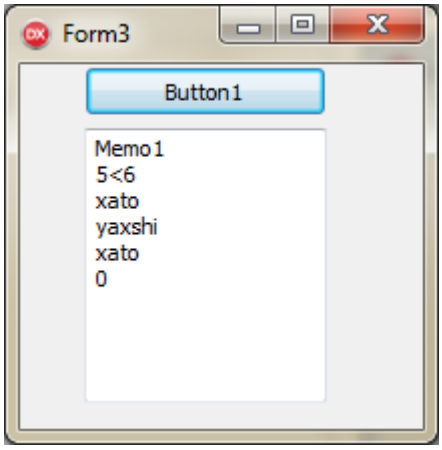
1-chi operand	2-chi operand	not	and	or	xor
1(true)	-	0(false)	-	-	-

0(false)	-	1(true)	-	-	-
0(false)	0(false)	-	0(false)	0(false)	0(false)
0(false)	1(true)	-	0(false)	1(true)	1(true)
1(true)	0(false)	-	0(false)	1(true)	1(true)
1(true)	1(true)	-	1(true)	1(true)	0(false)

Mantiqiy tiplardan foydalanishga misol keltitamiz:

```
procedure TForm3. Button1Click(Sender: TObject);
 var aa:boolean; a,b,c:Integer;
begin
 a:=5; b:=6; c:=7; aa:=a<b;
 if aa=true then memo1. Lines. Add('5<6'); aa:=b<a;
 if aa=false then memo1. Lines. Add('xato');
 if (b>a) or (c>b)then memo1. Lines. Add('yaxshi');
 if (b>a) xor (c>b)then memo1. Lines. Add('567') else
 memo1. Lines. Add('xato'); aa:=a>b;
 memo1. Lines. Add(BoolToStr(aa));
end;
end.
```

Olinadigan natija:



Haqiqiy tiplar. Haqiqiy tiplar haqiqiy sonlarni tasvirlash uchun ishlatiladi. Jadvalda Delphi 7 da ishlatiladigan haqiqiy tiplar ro'yxati keltirilgan.

Tip	O'zgrish diapazoni	O'lcham (baytda)
Real	$5.0*10^{-324}..1.7*10^{308}$	8
Real48	$2.9*10^{-39}..1.7*10^{38}$	6
Single	$1.5*10^{-45}..3.4*10^{38}$	4
Double	$5.0*10^{-324}..1.7*10^{308}$	8
Extended	$3.6*10^{-4951}..1.1*10^{4932}$	10
Comp	$-2^{63}+1..2^{63}-1$	8

Belgili tiplar. Bu tipga kompyuter klaviaturasidagi barcha belgilar kiradi va har bir belgiga 0. . 255 diapazondagi butun son mos qo'yiladi. Bu son belgidan foydalanish va uni tasvirlashda ishlatiladi. Bu sonni ord funktsiyasi qaytaradi.

Windowsda kodlash uchun ANSI (American National Standard Institute) kod tizimidan foydalaniladi. Ma'lumotlarning belgili tiplari faqat bitta belgini saqlash uchun xizmat qiladi. Jadvalda Delphida ishlatiladigan belgili tiplar ro'yxati keltirilgan. 0. . 31 kodli belgilar xizmatchi kodlar hisoblanadilar. Char tipida munosabat amallari va quyidagi funktsiyalardan foydalanishi mumkin:

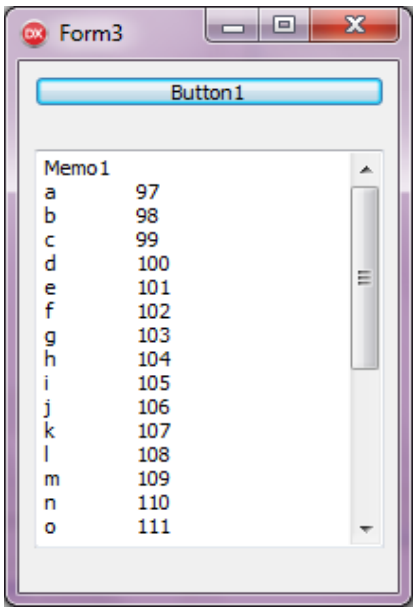
- chr(b)-byte tipidagi ifoda qiymatiga mos belgini beradi;

Tip	O'lcham (baytda)
Char	1
ANSChar	1
WideChar	2

Ushbu holatni quyidagi misol yordamida ko'rib chiqamiz:

```
procedure TForm3. Button1Click(Sender: TObject);
var b:char;
begin
for b := 'a' to 'z' do Memo1. Lines. Add(b+#9+IntToStr(ord(b)));
end;
end.
```

Olinadigan natija:



Delphi tili dastipchiga uzining tiplarini kiritishga imkon beradi.

Bu tiplar standart tiplarga eki avval kiritilgan tiplarga asoslangan bulib kuyidagi tiplarga tegishli bulishi mumkin:

sanovchi;

interval;

murakkab tip (yozuv).

Sanoqli tiplar. Sanoqli tiplar tartiblangan qiymatlar to'plamini ishlatadi.

Tip =(1 Kiymat, 2 Kiymat,. . . ,I Kiymat)

Masalan:

```
Type
 Color=(black,green,ellow blue,red,white);
 Fam=(Petrov,Sidorov,Raximov,Sobirov);
 DayOfWeek=(mon,tue,wed,thu,fri,sat,sun);
```

Bu erda

Color sanoq tipi beshta ranglar ketma ketligini aniqlaydi.

Fam sanoq tipi to'rtta familiyani aniqlaydi.

DayOfWeek sanoq tipi hafta nomlarini aniqlaydi.

Odatda Delphi tilida tiplar nomlari T xarfidan boshlanadi (Tipe — tip so'zidan).

Yangi tip ta'riflangandan sung shu tipga tegishli o'zgaruvchini ta'riflash mumkin, masalan:

```
type
TDayOfWeek = (MON,TUE,WED,THU, FRI,SAT,SUN) ;
var
ThisDay, LastDay: TDayOfWeek;
```

Sanovchi tip ta'rifi qiymatlar o'zaro munosabatini ko'rsatadi. Eng chap element minimal, eng ung element maksimal hisoblanadi. Yuqorida kiritilgan DayOfWeek tipi elementlari uchun quyidagi munosabat o'rinli:

MON < TUE < WED < THU < FRI < SAT < SUN

Sanovchi tip elementlari orasidagi munosabat o'zgarukvchilarni boshqaruvchi instruktsiyalarda qo'llashga imkon beradi, masalan:

```
if (Day = SAT) OR (Day = SUN) then
begin
{ agar kun shanba yoki yakshanba bo'lsa bajarilsin }
end;
```

Bu instruktsiyani quyidagicha yozish mumkin:
if Day > FRI then begin
{ agar kun shanba eki yakshanba bo'lsa bajarilsin }
end;

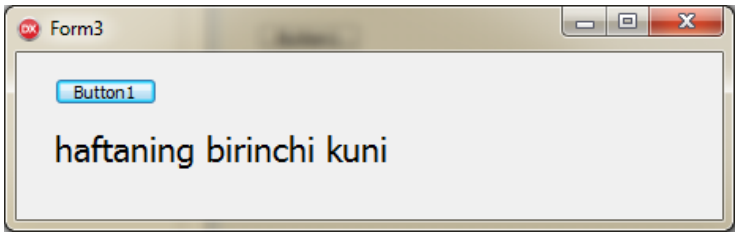
Sanovchi tip ta'rifi nomlangan konstantalar kiritishning qisqartirilgan shakli deb qarash mumkin. Misol uchun TDayOfWeek tipining ta'rifi quyidagi ta'riflarga tengdir:

const
MON=0; TUE=1; WED=2; THU=3; FRI=4; SAT=5; SUN=6;
end. end.

Izoh: sanoq tiplarini to'g'ridan-to'g'ri o'zgaruvcilar bo'limida ham berish mumkin.

Misol:

```
procedure TForm3. Button1Click(Sender: TObject);
var kun:(dushanba,seshanba,chorshanba,payshanba,juma,shanba,yakshanba);
begin
 kun:=dushanba;
Label1. Caption:='haftaning birinchi kuni';
end;
```



Interval (diapazon) tipi.

Interval (diapazon) tipi beriladigan qiymatga chegara qo'yadi.

Tip-diapazon asosiy tipning qism to'plami hisoblanadi. Asosiy tip sifatida har qanday tartib tiplariidan(tip-diapazonning o'zi bundan mustasno) foydalanish mumkin. Bu tip minimal va maksimal qiymatlar orqali beriladi.

```
 Type
 <tip nomi>=<minimal>. . <maksimal>;
```

Masalan:

```
Type
 Color=red. . green; // rangga chegara
 Digit=0. . 9; //butun sonlarga chegara
 Symb='A'. . 'Z'; // harflarga chegara
```

Haqiqiy tiplarga chegara qo'yilmaydi.

Interval tip ta’rifida nomlangan konstantalardan foydalanish mumkin. Quyidagi misolda interval tip TIndex ta’rifida HBOUND nomlangan konstantadan foydalanilgan:

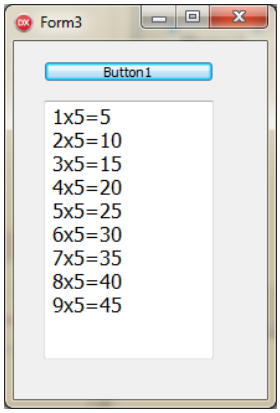
```
const
HBOUND=100;
type
TIndex=l. . HBOUND;
```

Interval tipdan massivlarni ta’riflashda qulaydir:

```
type
TIndex =1. . 100;
var
tab1 : array[TIndex] of integer; i:TIndex;
```

Butun son tipidan tashqari asos tip sifatida sanovchi tipdan foydalanish mumkin. Quyidagi dastip qismida TMonth sanovchi tip asosida interval tip TSammer ta’riflangan:

```
type
TMonth = (Jan, Feb, Mar, Apr, May, Jun,Jul, Aug, Sep, Oct, Nov, Dec);
TSammer = Jun. . Aug;
Misol:
procedure TForm3. Button1Click(Sender: TObject);
type dd=1. . 10; var i,j:dd;
begin
Memo1. Text:=''; j:=5;
 for i:=1 to 9 do
 Memo1. Lines. Add(IntToStr(i)+'x'+IntToStr(j)+'='+IntToStr(i*j));
 end;
end.
```



Izoh:

- tip-diapazon ham tiplar bo’limida berilishi shart emas. U ham o’zgaruvchilar bo’limida berilishi mumkin;

- Minimal chegara qiymati maksimal qiymatdan katta bo’lishi mumkin emas;
- .. belsi bir belgi sifatda ishlatiladi. Bu ikki nuqta orasiga “bo’shliq” belgisini qo’yish mumkin emas;

Delphining standart kutubxonasida ikkita funktsiya mavjud bo’lib ular tip-diapazonning maksimal va minimal qiymatlarini aniqlaydilar:

- High(x)-o’zgaruvchi x tegishli bo’lgan tip-diapazonning maksimal qiymatini beradi;
- Low(x)- o’zgaruvchi x tegishli bo’lgan tip-diapazonning minimal qiymatini beradi;

Yozuv

Dastiplash amaliyotida standart ma’lumotlardan tashkil topgan murakkab ma’lumotlar bilan ishlashga to’g’ri keladi. Misol uchun talaba to’g’risidagi ma’lumot ismi sharifi, tugilgan yili, adresi, kursi, guruhi va hokazolardan iborat bo’lishi mumkin. Bunday ma’lumotlarni ta’riflash uchun Delphi da yozuv (record) lardan foydalaniladi.

Yozuv bu - alohida nomlangan har xil tipli komponentalardan iborat murakkab tipdir.

Har qanday tip kabi, "yozuv" type bo’limida ta’riflanishi lozim. Bu ta’rif umumiy ko’rinishi:

```
Nom = record
1_ Maydon: 1_Tip; 2_ Maydon: 2_Tip;…; K_ Maydon: K_Tip; end;
```

Ta’riflarga misollar:

```
type
TPerson = record
f_name: string[20];
l_name: string[20];
day: integer;
month: integer;
ear: integer;
address: string[50]; end;
TDate = record
day: integer; month: integer; ear: integer;
end;
```

Yozuv tipidagi o’zgaruvchini quyidagicha ta’riflash mumkin:

```
var student : TPerson; birthday : TDate;
```

Yozuv elementiga (maydoniga) murojaat qilish uchun yozuv nomi va nuqtadan so’ng maydon nomini ko’rsatish kerak. Masalan:

Writeln('Imya: ', student. f_name + #13 + 'Adres: ', student. address);

Instruktsiya ekranga student o’zgaruvchi-yozuvning f_name (nom) va address (adres) maydonlarini chiqaradi.

Ba’zida o’zgaruvchi-yozuv tipi o’zgaruvchilar e’lon qilish bo’limida e’lon qilinadi. Bu holda yozuv tipi o’zgaruvchi nomidan so’ng ko’rsatiladi. Misol uchun student yozuvi var bo’limida quyidagicha ta’riflanishi mumkin:

```
student: record
f_name:string[20];
l_name:string[20];
day:integer;
month:integer;
ear:integer;
address:string[50];
end;
```

With instruktsiyasi

With instruktsiyasi dastipda maydonlar nomlarini o’zgaruvchi – yozuv nomini ko’rsatmasdan ishlatishga imkon beradi. Umumiy holda with instruktsiyasi quyidagi ko’rinishga ega:

```
with nom do
begin
( dastip instruktsiyasi} end;
```

Misol uchun dastipda quyidagi yozuv ta’riflangan bo’lsin

```
student: record
f_name: string[30];
l_name: string[20];
address: string[50];
end;
```

va studentlar to’g’risidagi ma’lumotlar E1, E2 va E3 o’zgaruvchilarda joylashgan bo’lsin. U holda

```
student. f_name := El;
student. l_name := E2;
student. address := E3;
```

instruktsiyalar o’rniga quyidagi instruktsiyani yozish mumkin:

```
with student do begin
f_name := E1; l_name := E2; address := E3;
```

end;

Butun tip

Tip	Diapazon
Shortint	-128-127
Smallint	-32 768 - 32 767
Longint	-2 147 483 648 - 2 147 483 647
Int64	-2^{63}- 2^{63} - 1
Byte	0-255
Word	0-65 535
Longword	0 - 4 294 967 295

Object Pascalda ko'proq universal Integer tipi (Longint ga ekvivalent) qo'llaniladi.

Xaqiqiy tip

Tip	Diapazon
Real48	2. 9x 10-39-1. 7x1038
Single	1. 5 x 10-45-3. 4x 1038
Double	5. 0x10-324 -1. 7x10308
Extended	3. 6x10-4951 -1. 1 x104932
Comp	263+1 - 263-1
Currency	-922 337 203 685 477. 5808 --922 337 203 685 477. 5807

Object Pascalda ko'proq universal Real tipi (Double ga ekvivalent) qo'llaniladi.

Simvolli tip

Char – ANSI kodlar jadvalidagi 0 dan 255 gacha bo'lgan raqamlarga mos keluvchi belgilar

Satriy tip

String – uzunligi 0 dan 255 gacha bo'lgan belgilar ketma – ketligi.

Mantiqiy tip

Boolean – True(rost) yoki False (yolg'on) qiymatlarning birini qabul qilishi mumkin.

1. Delphida tiplar, ularning ahamiyatini tushuntiring.
2. Butun tiplar haqida ma'lumot bering

3. Delphida simvolli va satriy tiplar.
4. Simvolli va satriy tiplarning berilishi, ular bilan bajariladigan amallar.
5. Simvolli va satriy kattaliklar.
6. Delphi dasturlash tilida o‘zgarmaslar, o‘zgaruvchilar va standart funksiyalar

4-BOB. DELPHI DASTURLASH TILI OPERATORLARI.

4.1. DELPHI DASTURLASH MUXITIDA TARMOQ OPERATORLARI.

Reja:

1. Tarkibiy va bo‘sh operatorlar.
2. Shart va mantiqiy ifodalar.
3. If…Then…else shartli operatori.
4. Tanlash (case) operatori.
5. Goto o‘tish operatori.
6. Label (belgilar) xizmatchi so‘zidan foydalanish qoidalari bilan tanishish.

Tayanch iboralar: Tarkibiy va bo‘sh operatorlar. Shart va mantiqiy ifodalar. If…Then…else shartli operatori. Tanlash (case) operatori. Goto o‘tish operatori. Label (belgilar) xizmatchi so‘zidan foydalanish qoidalari bilan tanishish.

Pascal tilida shart - bu mantiqiy turdagi ifoda bo’lib, u faqat «chin»(True) yoki «yolg’on»(False) qiymatni qabul qiladi.

Quyidagi mantiqiy belgilar ishlatiladi: >, <,<=,>=,<>,=. Bularga munosabat amallari deyiladi.

Quyidagi mantiqiy amallar ishlatiladi:

- NOT-«inkor»;
- AND-«mantiqiy va»;
- OR-«mantiqiy yoki».

Bu mantiqiy amallarning bajarilish natijalari quyidagicha:

Op1	Op2	Op1 AND Op2	Op1 OR Op2	NOT Op1
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Masalan: (5<6) AND (6<50) -mantiqiy ifoda rost (True),

(20>0) OR (20<0. 5) -mantiqiy ifoda rost (True),

(10<8) AND (10<15) -mantiqiy ifoda yolQon (False),

NOT(100>3) -mantiqiy ifoda yolQon (False).

Mantiqiy ifodalarni biror bir mantiqiy o'zgaruvchiga yuborish ham mumkin.

Masalan:

F:=(A<B) AND (A<C);

Bu yerda, agar ikkala shart bajarilgandagina F mantiqiy o'zgaruvchi "chin" (True) qiymatni qabul qiladi. Aks holda "yolg'on" (False) qiymatni qabul qiladi.

Pascal tilida shartli o'tish operatorining ikki xil ko'rinishi mavjud: to'liq va qisqa.

To'liq ko'rinish:

If <shart> then Begin

<shart rost bo'lganda bajariladigan operatorlar>

End

Else

Begin

<shart yolQon bo'lganda bajariladigan operatorlar>

End;

Qisqa ko'rinish:

If <shart> then Begin

<shart rost bo'lganda bajariladigan operatorlar>

End;

Bu yerda IF -agar; then -u holda; else -aks holda ma'nosini bildiruvchi xizmatchi (kalit) so'zlar.

Birinchi ko'rinishdagi shartli operatorda, agar shart bajarilsa birinchi Begin va end ichidagi operatorlar ketma–ket bajariladi, aks holda ikkinchi Begin va end ichidagi operatorlar ketma-ket bajariladi.

Ikkinchi ko’rinishdagi shartli operator quyidagicha ishlaydi. Agar berilgan shart bajarilsa Begin va end ichidagi operatorlar ketma-ket bajariladi, aks holda ular bajarilmaydi.

Agar bajariluvchi operatorlar soni bitta bo’lsa Begin va End so’zlarini yozish shart emas.

Misollar:

1) If A>0 Then Begin C:=1; B:=C+1; End

Else Begin C:=0; B:=4; End;

2) If D=A Then D:=A Else A:=D;

O’ar bir shartli o’tish operatori ichida boshqa ichki shartli operatorlar joylashishi ham mumkin, masalan.

If b1 then a1 else If b2 then a2 Else a3;

Misollar.

A:=0. 5; B:=-1. 7; IF A<B THEN A:=B ELSE B:=A;

Javob: 0. 5<-1. 7 yolg’on bo’lganligi sababli B:=A operator bajariladi, va bunda A=0,5 va B=0,5 ekenligi kelib chiqadi.

A:=0. 1; B:=0. 1; C:=0. 5; D:=0;

IF (A<B) OR (A>C) THEN D:=B+C ELSE

IF B=A THEN BEGIN D:=C; C:=A; END;

Javob: (0. 1<0. 1)yoki(0. 1>0. 5) bu mantiqiy ifoda yolg’on bo’lganligi sababli B=A shart tekshiriladi. Bu shart chin bo’lganligi sabab D=0,5 ga, S=0,1 qiymatlarga teng ekenligi kelib chiqadi.

Dasturda shunday holatlar bo’ladiki operatorlarning bajarilish shartiga qarab dasturning u yoki bu qismiga to’g’ridan-to’g’ri o’tishga to’g’ri keladi. Bunday holatlarda shartsiz o’tish operatoridan foydalanish mumkin.

Shartsiz o’tish operatorining ko’rinishi quyidagicha:

Goto n;

Bu yerda n -belgi(metka) bo’lib identifikator yoki butun son bo’lishi mumkin. Goto - o’tish ma’nosini bildiradi.

n- belgi dasturning bosh qismida Label so’zi yordamida e’lon qilingan bo’lishi shart. n boshqarilish uzatiladigan joyga n: shaklida qo’yiladi.

Misol:

. . . . . .

Goto L2;

. . . . . .

L2: C:=x*y;

. . . . . .

Ko’p hollarda baror bir parametrning qiymatiga qarab kerakli operatorlarni bajarishga to’g’ri keladi. Bunday hollarda tanlash operatorini ishlatgan qulay. Tanlash operatori ko’rinishi quyidagicha bo’ladi:

Case s of

1: A1;

2: A2;

. . . . . .

n: An;

Else Begin

<B1,B2,. . Bn>

End;

End;

Bu yerda Case -xizmatchi so’z bo’lib tanlash ma’nosini beradi; of -«dan» ma’nosini beradi; s-operator selektori; 1,2,. . n-operator belgilari; A1,A2,. . . An va B1,B2,. . . Bn-operatorlar.

Case operatori tarmoqlanish jarayonida berilgan bir necha operatoridan birini tanlash yo’li bilan amalga oshiradi. Opreatorlar ketma-ketligini tanlash operator selektorining qiymatiga qarab aniqlanadi. Operator selektori haqiqiy bo’lmagan o’zgaruvchi yoki ifoda bo’lishi mumkin. Agar operator selektori qiymati operator belgilari o’zgarmas qiymatiga teng bo’lmasa B1,B2,. . . Bn-operatorlari ketma-ket bajariladi.

Shartli o’tish operatorining quyidagi ko’rinishi

If B Then A1 Else A2;

tanlash operatorining quyidagi operatoriga ekvivalentdir.

```
Case B of
    True: A1;
    False: A2;
End;
```

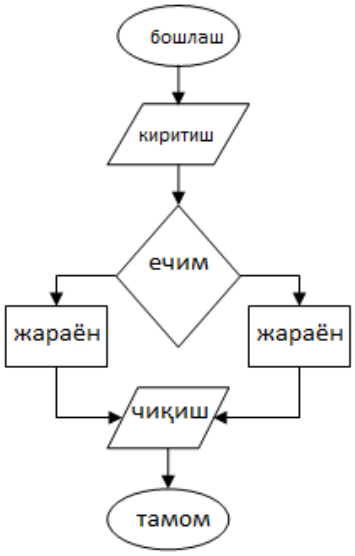
Misol

$ax^2+bx+c=0$ kvadrat tenglamaning ildizlarini topish dasturi tuzilsin.

```
PROGRAM Corni;
 Label 20;
 Var A, B,C, D, E, F, X, X1, X2, Z:real;
 BEGIN
      Writeln ('a, b, c koeffisientlar qiymatini kiriting:');
 Read (A,B,C);
 if A=0 THEN
 BEGIN
 x:=-B/C; writeln (x);
      goto 20
      end
 else
 BEGIN
 D:=B*B-4. 0*A*C;       Z:=2. 0*A; E:=-B/Z;
F:=sqrt(ABS(d)/Z;
 End;
      if D>=0 THEN
 BEGIN
      x1:=E+F; x2:=E-F; writeln (x1,x2);
 End
 else if D=0 then
      begin X:=E; writeln(x); end else writeln ('echim yo'q');
```

20: end.

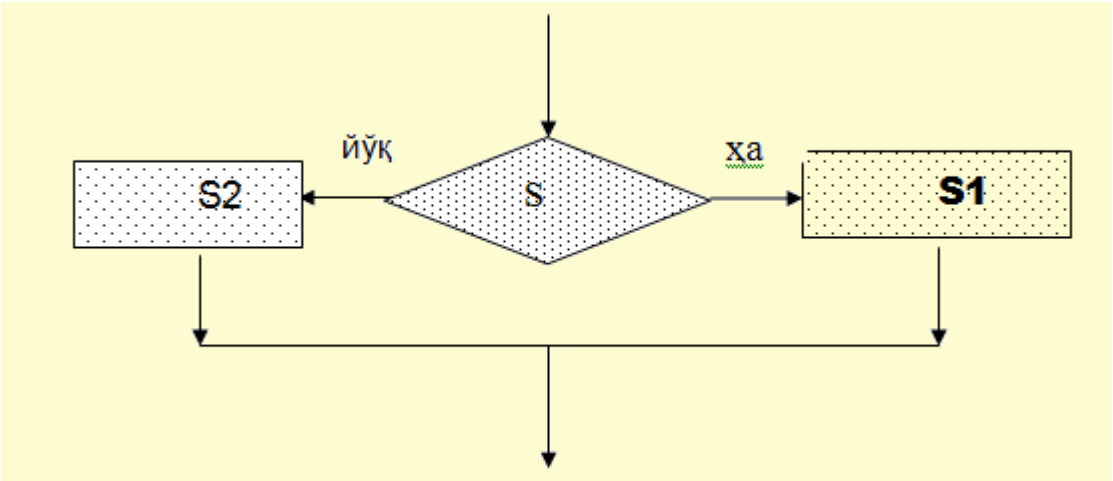
Шартга мувофиқ бажариладиган кўрсатмалар билан тузиладиган алгоритмлар ***тармоқланувчи алгоритмлар*** дейилади



Ma’lum bir shartni bajarilishi yoki bajarilmasligiga qarab, tarmoqlanuvchi jarayon holatlari aniqlanadi. Tarmoqlanuvchi jarayonlarni hisoblash uchun shartli operatordan foydalaniladi. SHartli operator ikki xil ko’rinishda bo’ladi:

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

To’la shartli operatorning algoritmik sxemasini quyidagi ko’rinishga ega:



To’liq shartli operator quyidagi formada yoziladi:

if <mantiqiy ifoda> *then* <operator> *else* <operator>

bu erda *if* (agar), *then* (u xolda), *else* (aks xolda) xizmatchi so’zlar.

SHunday kilib, to’liq shartli operatorni quyidagicha yozish mumkin:

if S then S1 else S2;

bu erda S - mantiqiy ifoda;

S1 – S mantiqiy ifoda rost qiymat qabul qilganda bajariluvchi operator;

S2 -S mantiqiy ifoda yolg'on qiymat qabul qilganda bajariluvchi operator.

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

To'liq shartli operatorga doir misollar:

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

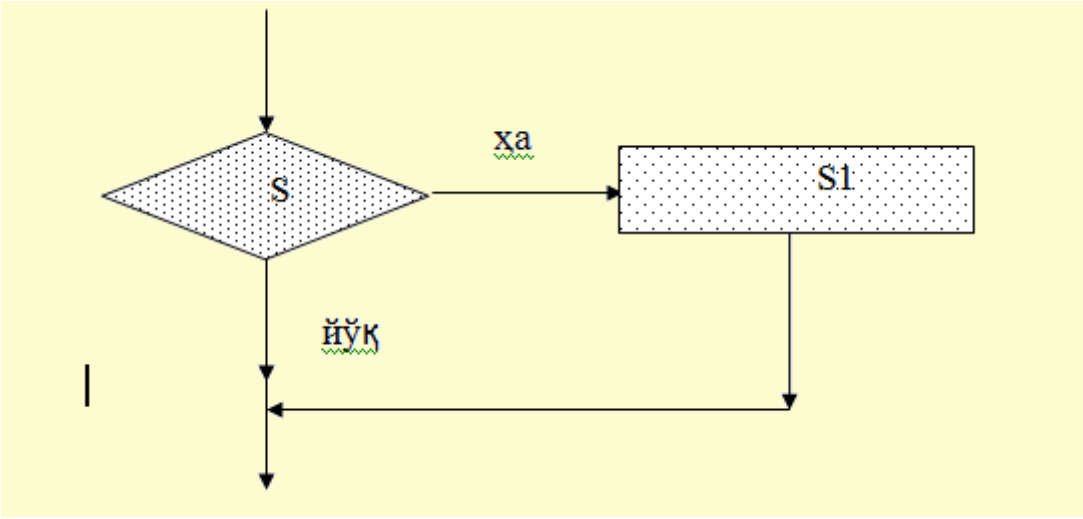
if (x<y) and (z>5) *then begin* y: = x * sin(x);

t: =x * cos(x) *end else begin* y: = 0; t: =1 *end*;

if x<0 or x =3 *then* y: = x*x+1 *else* if x<2

then y: = sqr(abs(x-1)) *else* y: = x*x;

Kisqa (to'liqmas) shartli operatorning algoritmik sxemasini quyidagi ko'rinishga ega:



CHala (to'liqmas) shartli operatorning yozilishini quyidagicha ifodalanadi:

if S then S1;

bu erda S - mantiqiy ifoda, S1 - operator.

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

SHartli operatordan foydalanishga misollar keltiramiz.

1-misol. Kiritilgan ixtiyoriy butun sonni juft yoki toqligini aniqlovchi dastur yarating.

```
program r1;
var x:integer; javob:string;
 begin
 readln(x);
 if x mod 2 =0 then javob:='juft' else javob:='to"q' ;
 writeln('kiritilgan son ',javob);
 readln;
 end.
```

2-misol. Kiritiladigan ixtiyoriy a,b,c sonlar uchun a+b>c, a+c>b, b+c>a tengsizliklarning barchasi bajarilganda «shartlar qanoatlantirilgan» deb javob beruvchi dastur yarating.

```
program r2;
uses crt;
var a,b,c,d:integer;
 javob:string;
 begin clrscr;
 readln(a,b,c);
 if (a+b>c) and (a+c>b) and (b+c>a) then
 writeln('shartlar qanoatlantirilgan');
 readln;
 end.
```

YUqorida keltirilgan ikki misoda masala shartiga ko’ra shartli operatorning to’liq va to’liq bo’lmagan holatlaridan foydalanildi.

Tarmoqlanish operatoridan foydalanishda quyidagi qoidalarga amal qilish shart:

IF operatoridan foydalanganda ELSE dan oldin «;» (nuqta-vergul) qo’yilmaydi.

SHartli operator Then va ELSE xizmatchi so’zlaridan keyin bir necha operator (amal yoki buyruq) ishlatilishi zarurati bo’lsa, u holda bu buyruqlar begin va end qavslari ichiga joylashtirishi shart.

(SHartsiz o’tish operatorini o’rganishda $ax^2+bx+c=0$ tenglamaning echimlarini aniqlovchi dastur keltirilgan, shu holatga e’tibor bering).

O’tish operatori (SHartsiz o’tish operatori).

SHartsiz o’tish operatori goto quyidagicha yoziladi:

goto belgi;

Bu erda goto xizmatchi so’z bo’lib, belgi operator boshqarishni uzatishi zarur bo’lgan (belilangan) «manzili» hisoblanadi. Belgi sifatida Turbo Paskal dasturlash tilida 0 dan 9999 gacha bo’lgan butun sonlardan va simvollar birikmasidan(xizmatchi so’zlardan tashqari) foydalanish mumkin. Belgilar dasturning tavsiflash qismining Label (nishonlar ro’yxati) bo’limida beriladi, masalan: Label 12, bel, r1;

YUqoridagi operatorlardan foydalanib, $ax^2+bx+c=0$ tenglamaning echimlarini aniqlovchi dastur yaratamiz:

```
uses crt;
label 12,13,aa,2;
var a,b,c:integer; x, d:real;
begin
 clrscr;
readln(a,b,c);
d:=sqr(b)-4*a*c;
 if d=0 then goto aa else goto 12;
aa:x:=-b/(2*a);
 writeln('x=',x:2:0); goto 13;
 12: if d>0 then
 begin
 writeln('x1=',(-b+sqrt(d))/2*a:2:0);
 writeln('x2=',(-b-sqrt(d))/2*a:2:0);
 end
```

```
else
writeln('tenglama haqiqiy echimlarga ega emas');
13:end.
```

4.2. DELPHI DASTURLASH MUXITIDA SIKLIK OPERATORLAR.

Reja:

1. Delphi dasturlash tilida sikllar.
2. For sikli.
3. While sikli.
4. Repeat sikli.
5. Murakkab sikllar.

Tayanch iboralar: sikl, murakkab sikl, for, while. . do, repeat. . until

Ayrim masalalarda bir yoki bir necha parametrlarning o'zgarishiga qarab ma'lum hisoblashlar bir necha marta takrorlanib bajarilishi mumkin. Masalan, y=ax+b funksiyani x ning bir necha qiymatida uning mos qiymatlarini hisoblash kerak deylik. Bunday hisoblashlarni kompyuterda dastur tuzib bajarish uchun siklik strukturali dasturlar tuzish kerak bo'ladi. Bu kabi dasturlarni shartli o'tish operatori yordamida ham tuzish mumkin. Lekin Paskal tilida siklik strukturali dastur tuzish uchun bir necha maxsus operatorlar mavjud.

For operatori takrorlanishlar soni aniq bo'lgan sikllik jarayonlar tashkil etishda ishlatiladi. Uning umumiy ko'rinishi quyidagicha:

For i:=m1 to m2 Do S;

Bu yerda i-sikl parametri; m1,m2 -i parametrining boshlanQich va oxirgi qiymati bo'lib, ular o'zgarmas son yoki ifoda bo'lishi mumkin; S-sikl tanasi bo'lib, bir necha operatorlardan tashkil topishi mumkin.

Agar sikl tanasi bir necha operatordan iborat bo'lsa ular **Begin** va **End** ichiga olinadi.

Misol. 1,2,. . . 10 conlar yiQindisini hisoblash dastursini tuzing.

```
Program S10;
 Const kn=10;
 Var i: Integer; S: Real;
 Begin
```

```
S:=0;
For i:=1 to kn do S:=S+i;
Write (‘S=’,S); Readln;
End.
```

Agar to so’zni **DoWnto** so’ziga almashtirilsa sikl parametri teskari bo’yicha o’zgaradi, ya’ni -1 qadam bilan. U holda sikl ko’rinishi quyidagicha bo’ladi.

For i:=m1 DoWnto m2 Do S;

Misol. 10 dan 1 gacha conlarni ekranga chiqarish dastursini tuzing.

```
Program SP;
Var i: Integer;
Begin
For i:=10 DoWnto 1 do Write (i); Readln;
End.
```

While sikl operatori takrorlanishlar soni oldindan aniq bo’lmagan hollarda takrorlanishni biror bir shart asosida bajaradi. Berilgan shart oldin tekshiriladi va keyin shartning bajarilishiga qarab kerakli operatorlar ketma-ketligi bajariladi. Bu operatorning umumiy ko’rinishi quyidagicha: **While B Do S;**

Bu yerda B -mantiqiy ifoda; S -sikl tanasi bo’lib, bir yoki bir necha operatorlar ketma-ketligidan iborat bo’lishi mumkin. Mantiqiy ifoda ‘True’ yoki ‘False’ qiymat qabul qiladi.

Agar mantiqiy ifoda ‘True’ qiymat qabul qilsa S operatorlari bajariladi, aks holda bajarilmaydi, ya’ni sikl ishlashdan to’xtaydi.

Misol. 1,2,. . . ,10 conlar yiQindisini hisoblash dastursini tuzing.

```
Program S10;
Const kn=10;
Var i: Integer; S: Real;
Begin
S:=0; i:=0;
While i<=kn do Begin i:=i+1; S:=S+i; End;
Write (‘S=’,S);
```

```
  Readln;
End.
```

Repeat sikl operatori ham takrorlanishlar soni oldindan aniq bo'lmagan hollarda takrorlanishni biror bir shart asosida bajaradi. Oldin sikl tanasidagi operatorlar ketma-ketligi bajariladi. Berilgan shart keyin tekshiriladi. Agar berilgan shart rost (True) bo'lsa, boshqaruv sikldan keyingi operatorni bajarishga o'tadi, aks holda sikl takrorlanadi. Bu operatorning umumiy ko'rinishi quyidagicha:

Repeat S

Until B

Bu yerda B -mantiqiy ifoda, 'True' yoki 'False' qiymat qabul qiladi; S -sikl tanasi bo'lib, bir yoki bir necha operatorlar ketma-ketligidan iborat bo'lishi mumkin.

Agar mantiqiy ifoda 'False' qiymat qabul qilsa siklda takrorlanish davom etadi, aks holda to'xtaydi.

Misol. 1,2,. . . ,10 conlar yig'indisini hisoblash dastursini tuzing.

```
Program S10;
  Const kn=10;
  Var i: Integer; S: Real;
  Begin
   S:=0; i:=0;
   Repeat
          i:=i+1; S:=S+i;;
   Until I>kn;
   Write ('S=',S);
   Readln;
End.
```

Odatda WHILE operatori REPEAT operatoriga nisbatan ko'p ishlatiladi. Bunga sabab ko'pchilik masalalarda sikl tugallanish sharti sikl boshlanmasdan tekshirish maqsadga muvofiqdar. Zarur bo'lsa siklni umuman bajarmasdan o'tish mumkin.

Ko’pchilik masalalarni yechishda tuzilgan dasturda ichma-ich joylashgan sikllar tashkil etishga to’g’ri keladi. Bunday sikllarga murakkab sikllar deyiladi. Murakkab sikllar tashkil etilganda quyidagi talablar bajarilishi zarur.

-ichki sikl tashqi sikl ichida to’liq yotishi kerak;

-sikllar bir-biri bilan kesishmasligi kerak;

-sikl ichiga tashqaridan to’g’ridan-to’g’ri kirish mumkin emas;

-sikl parametrlari boshqa-boshqa identifikatorlar bilan belgila-nishi kerak;

Misol. $S=\sum_{i=1}^{10}\prod_{j=1}^{5}\frac{i+j}{\sqrt{i\cdot j}}$ ifodani hisoblash dastursini tuzing.

Bu formulada agar yig’indini ochsak u quyidagi ko’rinishga keladi.

$$S=\sum_{i=1}^{10}\prod_{j=1}^{5}\frac{i+j}{\sqrt{i\cdot j}}=\prod_{j=1}^{5}\frac{1+j}{\sqrt{1\cdot j}}+\prod_{j=1}^{5}\frac{2i+j}{\sqrt{2\cdot j}}+...+\prod_{j=1}^{5}\frac{10+j}{\sqrt{10\cdot j}}$$

```
Program SP;
 Var
  i,j: Integer; S: Real;
 Begin
  S:=0;
  For i:=1 to 10 do
  Begin
        P:=1;
        For j:=1 to 5 do P:=P*(i+j)/Sqrt(i*j);
  S:=S+P;
 End; Write (‘S=’,S);
 End
```

Dasturlash jarayonida ayrim hollarda bir yoki bir necha amallarni bir necha marotaba takrorlab bajarish zarurati tug’iladi. Masalan, 1+2+…+2011 yig’indini hisoblashimiz hisoblash uchun quyidagi dastur tuzadigan bo’lsak,

```
. .
a:=1; S:=s+a;
a:=2; S:=s+a;
```

a:=3; S:=s+a;

a:=4; S:=s+a; va hokazo, ya’ni dasturimiz «uzundan-uzun» ko’rinishga ega bo’lar edi. E’tibor bilan qaraydigan bo’lsak a-o’zgaruvchi har safar 1-ga ortib borib, ega bo’lgan qiymati S-ga qo’shilmoqda. Aynan shunday hollar uchun parametrli tsikllardan foydalanish dasturchining ishini engillashtiradi.

Parametrik tsikllarning umumiy ko’rnishi quyidagicha:

For <TSikl parametri>:=<a> to <b> do

< operator yoki operatorlar>

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

1-misol:

```
For i:=1 to 23 do
    s:=s+1/I;
```

TSiklning bu holatida parametr i-ning qiymati dastlab 1-ga teng bo’lib, sungra tsiklning har bir qadamida ‘+1’-ga orta boradi va 2,3,…,23 ga teng bo’ladi. Zarur hollarda parametrning qiymatini ‘-1’ ortttirish mumkin bo’lib, bunda «to» o’rniga «downto» ishlatiladi.

1-misol:

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

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

Parametrik tsikllarning o’ziga xos xususiyatlari quyidagilardan iborat:

For tsiklidan takrorlanishlar soni aniq bo’lgan hollarda foydalanish maqsadga muvofiqdir.

TSikl parametri qiymati +1 yoki -1 ga avtomatik tarzda oshiriladi(«to» yoki «downto» ishlatilishiga ko’ra).

TSikl parametri sifatida butun, belgili, mantiqiy yoki sanoq tiplaridan foydalanish mumkin.

TSikl bir necha amalni bajarishga mo’ljallangan bo’lsa, tsikl tanasida bu amallar «begin» va «end» qavslari ichida berilishi shart(2-misolga qarang).

Parametrik takorlanishlar «ichma-ich» joylashishlari ham mumkin va bu holat juda ko’p masalalarni echishda qo’llaniladi.

Masalan:

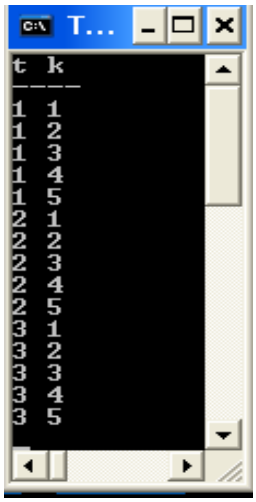
```
for t:=1 to 3 do
for k:=1 to 5 do
writeln(t,k);
```

Bu tsikllarni aniqroq tasavvur etish uchun quyidagi dastur va uning natijasini taqqoslaymiz:

t-parametrning qiymati 1-ga teng bo’lganda, k-parametr 1,2,3,4,5 qiymatlarni qabul qiladi

t-parametrning qiymati 2-ga teng bo’lganda, k-parametr yana 1,2,3,4,5 qiymatlarni qabul qiladi vahokazo.

```
uses crt;
var t,k:byte;
begin clrscr;
writeln('t',' ','k');
writeln('----');
for t:=1 to 3 do
for k:=1 to 5 do
writeln(t,' ',k);
end.
```



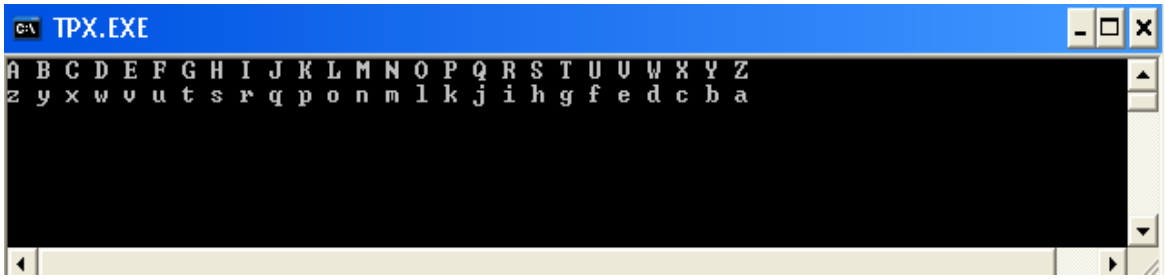
2-misol. 'A' dan 'Z'-gacha va 'z' dan 'a'-gacha bo’lgan barcha simvollarni chop etuvchi dastur tuzing.

Dastur ko’rinishi:

```
var i:char;
 begin
```

```
for i:='A' to 'Z' do
write(i,' ');
writeln;
for i:='z' downto 'a' do
write(i,' ');
readln;
end.
```

Dastur natijasi:



3-misol. Raqamlari yig'indisi 8-ga teng bo'lgan barcha ikki xonali sonlarni aniqlab, chop etuvchi dastur yarating.

```
uses crt;
var i:10. . 99 ;
a,b:0. . 9;
begin
clrscr;
for i:=10 to 99 do
begin
a:=i div 10;
b:=i mod 10;
if a+b=8 then write(i,' ');
end;
end.
```

Mavzuga oid savollar

1. Takrorlash operatorlarining necha xili mavjud?
2. Takrorlanuvchi algoritmiga misol keltiring.
3. Parametrli takrorlash operatorining ko'rinishi qanday?
4. Parametrli takrorlash operatori qachon qo'llaniladi?
5. Takrorlash parametri qanday qiymatlarni qabul qiladi?
6. Takrorlash parametrining qiymatlari chegaralanganmi?
7. Takrorlash operatorini ishlashini tushintiring.
8. Qanday holda Do yoki Downto xizmatchi so'zlar ishlatiladi?

4.3. DELPHIDA MASSIVLAR.

Reja:

1. Delphi dasturlash tilida massivlar.
2. Massivlarni tavsiflash, e'lon qilish.
3. Massivlarni berilish usullari.
4. Massiv elementlarini kiritish va chiqarish.
5. Random (max) funksiyasi bilan tanishtirish.
6. Ko'p o'lchovli massivlar.
7. Massiv elementlarini saralash va tartiblash.

Tayanch iboralar: Delphi dasturlash tilida massivlar. Massivlarni tavsiflash, e'lon qilish. Massivlarni berilish usullari. Massiv elementlarini kiritish va chiqarish. Random (max) funksiyasi bilan tanishtirish. Ko'p o'lchovli massivlar. Massiv elementlarini saralash va tartiblash.

Massiv - bu bir xil tipli, chekli qiymatlarning tartiblangan to'plamidir. Massivlarga misol sifatida matematika kursidan ma'lum bo'lgan vektorlar yoki matritsalarni ko'rsatish mumkin. Dasturda ishlatiluvchi barcha massivlarga o'ziga xos ism berish kerak. Massivning har bir hadiga murojaat esa uning nomi va o'rta qavs ichiga olib yozilgan tartib hadi orqali amalga oshiriladi.

Massiv boshqa o'zgaruvchilar kabi o'zgaruvchilar bo'limida e'lon qilinadi va uning e'lon qilinishi umumiy holda quyidagicha:

<Nom>: array [quyi_indeks. . yuqori_indeks] of <tip>;

Bu yerda :

<Nom>- massiv nomi;

Array- Object Pascal ning massivni ifodalash uchun ishlatadigan xizmatchi sozi;

quyi_indeks va yuqori_indeks butun qiymatli doimiyliklar (indekslar diapazoni);

of –xizmatchi so'z;

<tip>- massiv elementlarining tipi;

Misollar keltirib o’tamiz:

```
Stud :array [1. . 100] of string;
Qiymat: array [-3. . 33] of real;
Kunlar: array [1. . 365] of integer;
```

Massivni e’lon qilishda dasturning const (doimiyliklar) bo’limida qiymatlari berilgan kattalillardan ham foydalanish mumkin. Masalan:

```
Const mm:=55;
Var aa:array[1. . mm] of real;
```

Agar massiv modulning lokal o’zgaruvchisi sifatida berilgan bo’lsa uning dastlabki qiymatlarini e’lon qilinishi jarayonida berish mumkin. Bu holatni quyida keltirilgan dastur tahlili yordamida qarab chiqing.

```
…
var
 Form5: TForm5;
 a:array[1. . 5] of Integer=(3,4,5,6,7);
implementation
{$R *. dfm}
```

```
end.
```

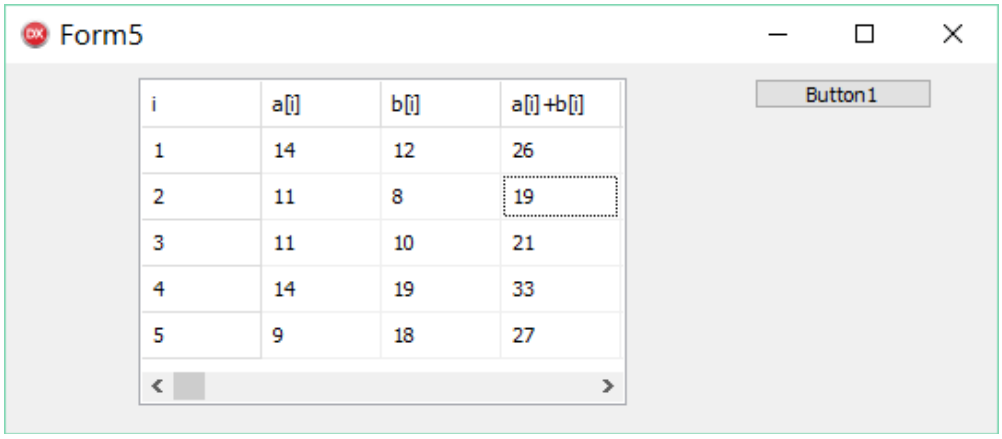
Massivlarni e’lon qilishni usullaridan biri bu uni tiplar bo’limida tavsiflashdir. Type bo’limida

type <nom> = array [quyi_indeks. . yuqori_indeks] of <tip>;

ko’rinishida berilishi mumki. Albatta bu tipdan foydalanish o’zgaruvchilar bo’limida tavsiflanishi shart. Bunga misol keltirib o’tamiz:

```
procedure TForm5. Button1Click(Sender: TObject);
type Name = array[1. . 5] of real; var a,b:name; i:integer;
begin
StringGrid1. cells[0,0]:='i'; StringGrid1. cells[1,0]:='a[i]';
StringGrid1. cells[2,0]:='b[i]'; StringGrid1. cells[3,0]:='a[i]+b[i]';
for I := 1 to 5 do
begin
a[i]:=random(20); b[i]:=random(20);
StringGrid1. Cells[0,i]:=IntToStr(i);
StringGrid1. Cells[1,i]:=FloatToStr(a[i]);
```

```
        StringGrid1. Cells[2,i]:=FloatToStr(b[i]);
        StringGrid1. Cells[3,i]:=FloatToStr(a[i]+b[i]);
        end;
        end;
end.
```



Massivning zarur hadiga murojaat quyidagicha amalga oshiriladi:

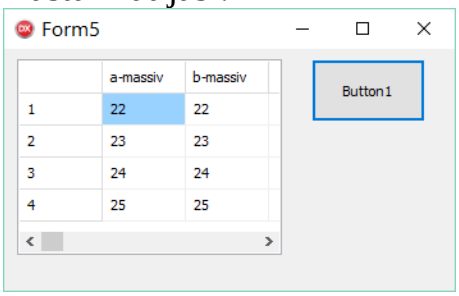
<massiv nomi>[<indeks>], bu yerda <indeks> massiv hadining joylashgan joyini anglatuvchi tartib qiymati.

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 aniqlash va ularning tartibini belgilashi kerak. Ammo shuni qayd qilish shart: indeks tipiining quvvati 2 Gbaytdan oshmasligi nazarda tutilgan.

Delphi dasturlash tilida qiymat berish operatori yordamida bir massiv elementlarini ikkinchisiga o'zlashtirish mumkin. Buni quyidagi dastur va uning keltirilgan natijasi tahlili yordamida kuztishingiz mumkin.

```
procedure TForm5. Button1Click(Sender: TObject);
var a,b:array [1. . 4] of integer;
 I: Integer;
begin
 a[1]:=22; a[2]:=23; a[3]:=24; a[4]:=25;
 b:=a;
 for I := 0 to 3 do
 begin
 StringGrid1. Cells[0,i+1]:=IntToStr(i+1);
 StringGrid1. Cells[1,i+1]:=IntToStr(a[i+1]);
 StringGrid1. Cells[2,i+1]:=IntToStr(b[i+1]);
 end;
end;
procedure TForm5. FormCreate(Sender: TObject);
begin
StringGrid1. Cells[1,0]:='a-massiv';
StringGrid1. Cells[2,0]:='b-massiv';
end;
end.
```

Dastur natijasi:



Bu o’rindan yana quyidagi holatga ham e’tibor berish shart, agar
var a,b:array [1. . 4] of integer; e’loni a:array [1. . 4] of integer;
b:array [1. . 4] of integer; e’lonlari bilan almashtirilsa xatolik albatta kuzatiladi.

```
b:=a;
```

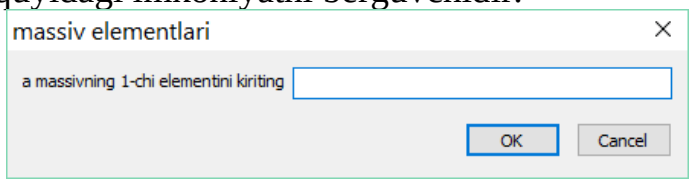
Massivlar ustida munosabat amallarini bajarib bo’lmaygi, ya’ni quyidagi kabi buyruqni berib natija olish mumkin emas.

```
if a=b then ShowMessage('massvlar teng');
```

Ammo massivlar elementlari tengligini hadlarga nisbatan tekshirish mumkin.
Masala mohiyatiga ko’ra massiv elemenlari turlicha usullar yordamida kiritilishi mumkin. Masalan random funktsiyasi, o’zlashtirish operatori va boshqa usullar. Quyida qiymatlar kiritishning ajoyib usullaridan birini keltirib o’tamiz:

```
procedure TForm5. Button1Click(Sender: TObject);
var a,b:array [1. . 4] of integer;
I: Integer; s1,s2:string;
begin
for I := 1 to 3 do
 begin
 s1:='a massivning '+inttostr(i)+'-chi elementini kiriting';
 s2:=inputbox('massiv elementlari',s1,'');
 a[i]:=StrToInt(s2);
 end;
end;
end.
```

Dastur foydalanuvchiga aniq yordam berishga mo’ljallangan. S1 va s2 uchun aniqlangan buyruq quyidagi imkoniyatni berguvchidir.



Endi massiv elementlari yig’ndisini xisoblovchi dastur bilan tanishib chiqaylik. Bir o’lchamli n ta hadli (n=30) massiv hadlarini yig’indisini aniqlash.

```
const n = 30;
var
i: integer; x: array [1. . n] of real; S: real;St: string;
begin
 for I := 1 to n do
 begin
```

```
 st:=InputBox(’’, ’’, ’’);
 x[i] := StrToFloat(st); { massiv xadlarini kiritish}
 end;
 S := 0;
 for I := 1 to n do S := S + x[i];
 ShowMessage(‘natijaq’, S)
end;
```

Ko’p o’lchamli massivlar

Bir o’lchamli massivlarning hadlari skalyar miqdorlar bo’lgan edi. Umumiy holda esa massiv hadlari o’z navbatida yana massivlar bo’lishi mumkin, agar bu massivlar skalyar miqdorlar bo’lsa natijada ikki o’lchamli 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 massivlar tiplarini aniqlashni bir necha xil yo’lda quyidagi misol ustida ko’rib chiqaylik: (10 ta satr va 20 ta ustundan iborat matritsa tipini aniqlash, massiv hadlari real tipida bo’lsin)

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

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

```
const n = 24;
type
 hafson = (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 ustida bir nechta tugallangan dasturlar bilan tanishib chiqaylik.

1. Matritsalarni qo’shish:

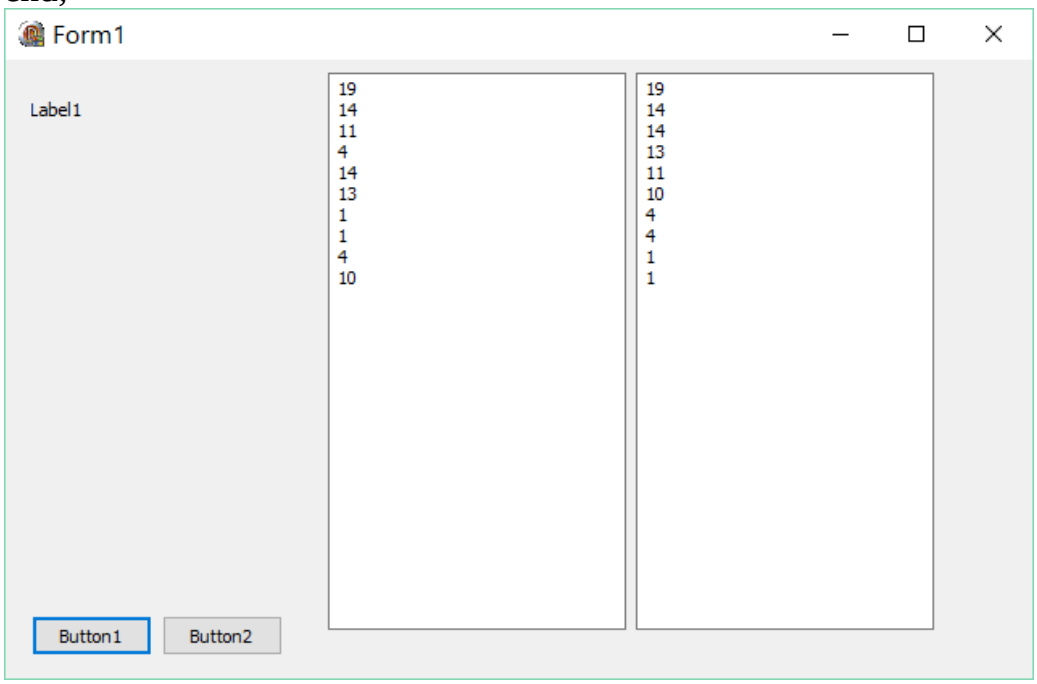
```
const n = 3; m = 4;
    { n - matritsa satrlari soni,
     m - ustunlar soni}
var
 i, j: integer;
 A, B, C: array[1. . n, 1. . m] of real;
 St: strin;
begin
 {A, V matritsa xadlarini kiritish}
 for i := 1 to n do
 for j := 1 to m do
 begin
 st:=InputBox(’A massiv elementlari’, ’’, ’’);
 A[i,j] := StrToFloat(st); { massiv xadlarini kiritish}
 st:=InputBox(’B massiv elementlari’, ’’, ’’);
 B[i,j] := StrToFloat(st); { massiv xadlarini kiritish}
 end;
 Label1. Caption := ‘’;
 for i := 1 to n do
 begin
 for j := 1 to m do
 begin
 C[i,j] := A[i,j] + B[i,j];
 Label1. Caption := Label1. Caption + FloatToStr(C[i,j])
 end;
 Label1. Caption := Label1. Caption + #10#13;
 end; end;
```

Massiv elementlarini saralash usullari juda ko’p bo’lib ulardan ayrimlarini keltiramiz:

1-usul:

```
procedure TForm1. Button1Click(Sender: TObject);
var a:array [byte] of integer; i,j,k,m,z,d:integer;
begin k:=10; memo1. Clear; memo2. Clear;
for i := 1 to k do
begin a[i]:=random(20); memo1. Lines. Add(inttostr(a[i])); end;
for j :=1 to k do
begin
m:=a[j];
for i := j to k do
begin
 if m<a[i] then
 begin m:=a[i]; d:=i; z:=a[j]; a[j]:=m; a[d]:=z;
 end ;
```

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



2-usul:

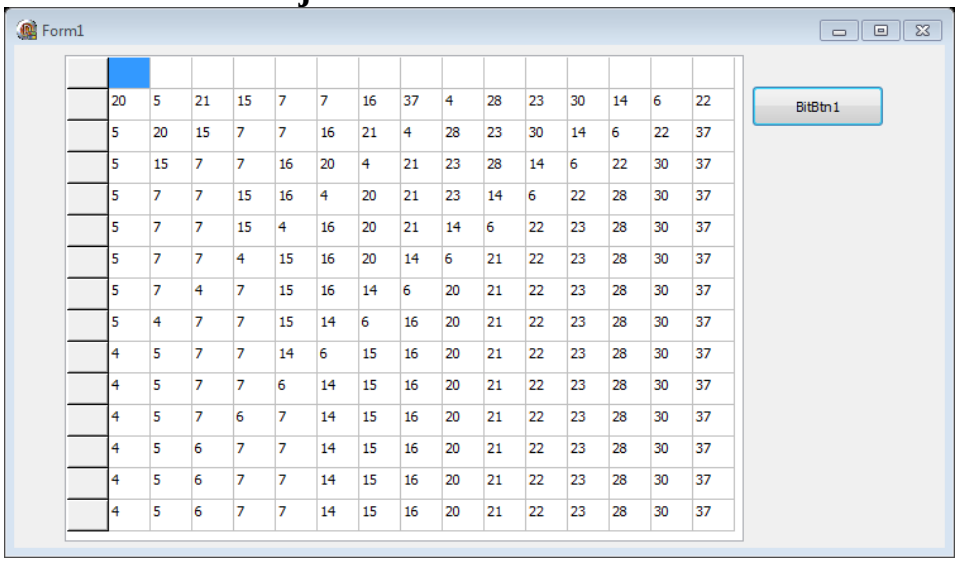
Pufakcha (qalqib chiqish) usuli. A[0], A[1],. . , A[N] massivning elementlari berilgan bo'lsin. Ketma-ket ravishda A[0], va A[1], A[1], va A[2] elementlar o'zaro taqqoslanib agar A[i]>a[i+1] bo'lsa ular o'zaro o'rin almashadilar. Ikkinchi qadamda shu holat A[N-1] gacha davom ettiriladi va hokazo. Bu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

Ushbu algoritmni Delphi dasturlash tilida keltirib o'tamiz.

```
procedure TForm1. BitBtn1Click(Sender: TObject);
 var A:ARRAY[1. . 15] OF INTEGER; I,D,K,Z,QAT:INTEGER;
begin
RANDOMIZE; QAT:=2; StringGrid1. RowCount:=1;
 for I := 1 to 15 do
 BEGIN
 A[I]:=RANDOM(39); StringGrid1. Cells[I,1]:=FloatToStr(A[I]);
 END;
 K:=14;
 while (K>=1) do
 BEGIN
 I:=1;
 while (I<=K) do
 BEGIN
 if A[I]>A[I+1] then
 BEGIN D:=A[I]; A[I]:=A[I+1]; A[I+1]:=D; END;
 I:= I+1;
 END;
 K:=K-1;
```

```
for Z := 1 to 15 do
BEGIN StringGrid1. Cells[Z,QAT]:=FloatToStr(A[Z]); END;
QAT:=QAT+1; StringGrid1. RowCount:=StringGrid1. RowCount+1;
END;
end;
end.
```

Dastur natijasi:



20	5	21	15	7	7	16	37	4	28	23	30	14	6	22	
5	20	15	7	7	16	21	4	28	23	30	14	6	22	37	
5	15	7	7	16	20	4	21	23	28	14	6	22	30	37	
5	7	7	15	16	4	20	21	23	14	6	22	28	30	37	
5	7	7	15	4	16	20	21	14	6	22	23	28	30	37	
5	7	7	4	15	16	20	14	6	21	22	23	28	30	37	
5	7	4	7	15	16	14	6	20	21	22	23	28	30	37	
5	4	7	7	15	14	6	16	20	21	22	23	28	30	37	
4	5	7	7	14	6	15	16	20	21	22	23	28	30	37	
4	5	7	7	6	14	15	16	20	21	22	23	28	30	37	
4	5	7	6	7	14	15	16	20	21	22	23	28	30	37	
4	5	6	7	7	14	15	16	20	21	22	23	28	30	37	
4	5	6	7	7	14	15	16	20	21	22	23	28	30	37	
4	5	6	7	7	14	15	16	20	21	22	23	28	30	37	

Mavzuga doir savollar:

1. Massiv tarifi va uning berilish usullari.
2. Massivlarni tiplar bo’limida e’lon qilishning o’ziga xos jihatlari.
3. Stud :array [1. . 100,1. . 6] of string; massiv haqida fikr bildiring.
4. Massiv elementlarini kirtish yo’llari.
5. Massivlardan foydalanishga misolllar keltiring.
6. Massiv elementlarini saralash yo’llari.

6.1. PROSEDURA VA FUNKSIYALAR.

Reja:

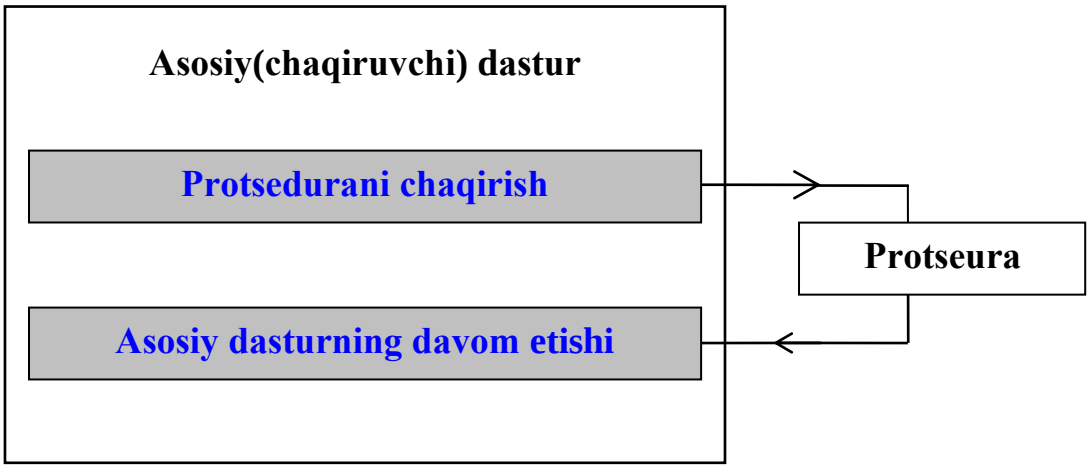
1. Prosedura va funksiyalar Delphi dasturlash tilining muhim instrumenti sifatida.
2. Prosedura tarifi, uning nomi, undan foydalanish yo’llari.
3. Funksiya tarifi, uning nomlanishi, undan dasturda foydalanish.

Tayanch iboralar: Prosedura va funksiyalar Delphi dasturlash tilining muhim instrumenti sifatida. Prosedura tarifi, uning nomi, undan foydalanish yo’llari. Funksiya tarifi, uning nomlanishi, undan dasturda foydalanish.

Dasturlash jarayonida shunday holatlar bo’ladiki, bir xil operatorlar ketma-ketligini dasturning bir necha joylarida takroran yozishga to’g’ri keladi. Bunday takrorlanishni yo’qotish maqsadida dasturlashning ko’pgina tillarida qism dastur

tushunchasi kiritilgan. Takrorlanadigan operatorlar ketma-ketligini mustaqil dastur bo'lagi -qism dastur ko'rinishida bir marotaba yoziladi va bu dastur bo'lagi kerak bo'lgan joylarda esa, unga murojaat qilinadi xalos. Paskal tilida qism dastur prosedura yoki funksiya ko'rinishida beriladi

Dasturlash jarayonida dasturning ayrim qismlari ko'p marotaba takrorlanishi kuzatiladi va to'g'ridan-to'g'ri bunday dasturlash qator noqulayliklarni vujudga keltiradi. Bu xolatdan qutulishning yo'li dasturda protsedura va funktsiyalardan foydalanishdir. Protsedura deb o'zining maxsus nomiga ega bo'lgan dasturning qismiga aytiladi va dasturda bu nomni joylashtirish protseduraning faollashtirishiga ya'ni undan foydalanishga olib keladi. Qism dastur faollashi Bilan undagi operatorlar ketma-ket bajariladi va oxirgi operator bajarilgach, boshqaruv asosiy dasturdagi shu qism dasturni chaqiruvchi operatordan keyingi operatorga o'tadi. Asosiy va qism dasturning o'zaro aloqasini sxematik tarzda quyidagicha ifodalash mumkin:



Ayrim masalalarni yechishda ma'lum parametrlarning har xil qiymatlarida bir xil hisoblashlarni bajarishga to'g'ri keladi. Bunday hollarda dastur hajmini kichiraytirish maqsadida prosedura yoki funksiyalar tashkil qilish zarur. Prosedura yoki funksiyaga murojat qilish dasturda uning nomini ko'rsatish orqali amalga oshiriladi. Kerakli parametrlar shu nomdan keyin beriladi. Prosedura yoki funksiyalar tashkil qilinganda ular dasturning bosh qismida beriladi. Ularga murojaat qilish esa dasturning asosiy qismining kerakli joyida beriladi. Asosiy dastur bilan prosedura orasida o'zgaruvchilar qiymat almashuvi formal va faktik parametrlar yordamida amalga oshiriladi. Prosedura yoki funksiyaga murojaat qilinganda boshqarilish qaerdan uzatilsa yana shu joyga qaytib keladi. Prosedura ichida yana bir necha prosedura yoki funksiya ishlatilishi

mumkin. Dasturda e’lon qilingan o’zgaruvchilar, shu dasturdagi prosedura va funksiyalarga nisbatan global deyiladi. Prosedura va funksilar ichida e’lon qilingan o’zgaruvchilar lokal deyiladi. Ularning ta’sir doirasi shu prosedura va funksiyalarning ichida bo’ladi xalos.

Proseduralarni e’lon qilish dasturning bosh qismida keltiriladi va u quyidagicha boshlanadi.

Procedure <pros. nomi> (<formal parametrlar>);

M: **Procedure AB (x,y);**

Formal parametrlarni shu prosedura bosh qismida yoki sarlavhada e’lon qilish mumkin.

M. **Procedure AB (x,y: Real);**

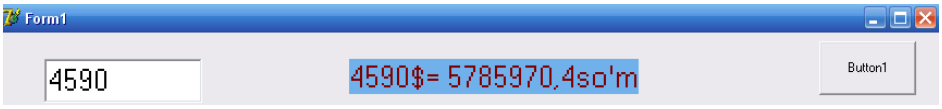
Har qanday prosedurani kichik bir dastur deb qarash mumkin. Prosedura ham dasturga o’xshab bosh va asosiy qismlardan toshkil topadi. Bosh qismda prosedura nomi va uning parametrlari e’lon qilinadi. Asosiy qism operatorlar ketma-ketligidan tashkil topgan bo’lib, ular Begin - End ichiga olinadi. Prosedura nomi foydalanuvchi tamonidan beriladi.

Funktsiyaning protseduradan farqi shunda kim undan foydalanishda natija funktsiya qiymati sifatida xosil bo’ladi, ya’ni funktsiyadan boshqa operandlar kabi ifodalarda foydalanish mumkin. Protsedura va funktsiyalardan biz avval ham foydalangan edik. Masalan, Exit, ShowMessage protseduralari, StrToint, FloatTostr, Random va matematik funktsiyalar bo’lib bular standart funktsiyalar toifasiga mansub. (Delphi dasturining asosiy boyligi xisoblanadi). Quyida standart va nostandart qism dasturlarga misol ketiramiz:

```
implementation
{$R *. dfm}
Procedure doltosum;
var dol,sum:real;
begin
dol:=StrToFloat(form1. Edit1. Text);
sum:=dol*1260. 56;
```

```
Form1. Label1. Caption:=Form1. Edit1. Text+'$= '+ FloatTostr(sum)+'so"m';
end;
procedure TForm1. Button1Click(Sender: TObject);
begin
doltosum;
end;
procedure TForm1. edit1keypress(Sender: TObject; var Key: Char);
begin
 if key=char(vk_return) then doltosum;
end;
end.
```

Dastur natijasi:



Umumiy holda kism dastur parametrga ega bo'ladi. Bu parametrlar haqiqiy va formal nom bilan yuritiladi. Funktsiyani berilishida ko'rsatiladigan parametrlar formal parametrlar, undan foydalanishda ko'rsatiladigan parametrlar esa haqiqiy parametrlar deb yuritiladi. Parametrlardan kattalikni qism dasturga uzatish va qism dasturdan natija olish uchun foydalaniladi. Kerakli hollarda haqiqiy parametr sifatida ifodadan ham foydalanish mumkin va bu holda uning tipi formal parametr tipi bilan mos kelishi lozim.

Funktsiyaga murojaat qilish instruktsiyasi quyidagicha:

O'zgaruvchi:=Funktsiya(parametrlar);

Bu erda:

- o'zgaruvchi-funktsiya qiymati o'zlashtiriladigan o'zgaruvchining nomi;
- funktsiya-funktsiya nomi;
- parametrlar-formal parametrlar ro'yxati(parametr sifatida o'zgaruvchilar yoki doimiyliklar beriladi);

Funktsiyani e'lon qilishning umumiy ko'rinishi quyidagicha:

Function *nom (parametr1:tip1,..., parametr k :tip K): Tip;*

Var

// lokal parametrlar e’lon qilinadi

Begin

// funktsiya instruktsiyalari

Nom:= ifoda;

End;

Bu erda:

- Function-Delphidagi xizmatchi so’zlardan bo’lib, dasturchining funktsiyasini bildiradi;
- *Nom*-funktsiya nomini anglatadi va asosiy dasturdan funktsiya instruktsiyalariga o’tish uchun foydalaniladi;
- *Parametr*- funktsiya qiymatini xisoblash uchun ishlatiladigan o’zgaruvchi va uning odatiy o’zgaruvchidan farqi var so’zidan keyin berilmasdan funktsiya sarlavxasida beriladi;

Izox: kerakli natija oxirida albatta funktsiya nomiga o’zlashtirilishi shart.

1-misol.

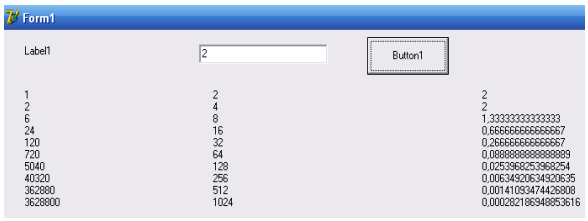
Quyidagi ketma-ketlik dastlabki 10 ta hadlarining surati va maxrajini alohida-alohida va ularning nisbatini chop etuvchi dastur tuzing.

$$\frac{x}{1!}, \frac{x^2}{2!}, \frac{x^3}{3!}, \cdots$$

Javob:

```
implementation
{$R *. dfm}
procedure TForm1. Button1Click(Sender: TObject);
var k:integer;x:real;
function fact(N:integer):currency;
var i,p:integer;
begin
p:=1;
for i:=1 to n do
```

```
p:=p*i;fact:=p;
end;
function dar(x1:real;N:integer):real;
var i:integer;p1:real;
begin
p1:=1;
for i:=1 to n do
p1:=p1*x1;dar:=p1;
end;
begin
x:=strtofloat(edit1. Text);
for k:=1 to 10 do
begin
label3. Caption:=label3. Caption+floattostr(dar(x,k))+#13;
 label2. Caption:=label2. Caption+floattostr(fact(k))+#13;
 label4.  Caption:=label4.  Caption+floattostr(dar(x,k)/fact(k))+#13;end;  end;  end.
```



Natijalar to’g’riligini Excel dan foydalanib ko’rishimiz mumkin, ya’ni

	A	B	C	D
1	1	1	2	2
2	2	2	4	2
3	3	6	8	1,333333
4	4	24	16	0,666667
5	5	120	32	0,266667
6	6	720	64	0,088889
7	7	5040	128	0,025397
8	8	40320	256	0,006349
9	9	362880	512	0,001411
10	10	3628800	1024	0,000282

2-misol.

Elementlari soni 10 ta tasodufiy massiv yaratuvchi va uning maksimal elementini aniqlovchi dastur tuzing.

Javob:

```
implementation
```

```
{$R *. dfm}
procedure TForm1. Button1Click(Sender: TObject);
var tasoduf:array [1. . 10] of integer;
i,maxmas:integer;
function max(a,b:integer):integer;
begin
if a>=b then max:=a else max:=b;
end;
begin
label1. Caption:='';
 for i:=1 to 10 do
 begin
 tasoduf[i]:=random(2007);
 label1. Caption:=label1. Caption+inttostr(tasoduf[i])+#13;
 end;
 maxmas:=tasoduf[2];
 for i:=1 to 10 do
 maxmas:=max(maxmas,tasoduf[i]);
 label2. Caption:=inttostr(maxmas);
end;end.
```

Dastur natijasi:

Protsedura bu qism dasturning bir ko’rinishi bo’lib, undan quyidagi ikki holatda foydalaniladi:

- agar qism dastur asosiy dasturga biror kattalik qiymatini xisoblab bermasa. Masalan, dialog oynasi uchun faqat grafik chizib bersa;
- agar qism dastur asosiy dasturga bir necha kattalik qiymatini xisoblab berish zarurati bo'lsa;

Protsedurani e'lon qilishning umumiy ko'rinishi quyidagicha:

Procedure *nom (var:parametr1:tip1,…, parametr k :tip K);*

Var

// lokal parametrlar e'lon qilinadi

Begin

// Protsedura instruktsiyalari

End;

Bu erda:

- Procedure -Delphidagi xizmatchi so'zlardan bo'lib, dasturda protsedura e'lon qilinganligini bildiradi;
- *Nom- protsedura* nomini anglatadi va asosiy dasturdan protsedura instruktsiyalariga o'tish uchun foydalaniladi;

Parametr- formal parametr (o'zgaruvchi) bo'lib, undan protsedura instruktsiyalarida foydalaniladi, *var* so'zi parametrlardan oldin berilishi shart emas va agar berilgan bo'lsa protseduradan foydalanishda uning formal parametri o'zgaruvchidan iborat bo'lishi lozim. o'zlashtirilishi shart.

Quyidagi ketma-ketlik dastlabki 10 ta hadlarining surati va maxrajini alohida-alohida va ularning nisbatini chop etuvchi dastur tuzing.

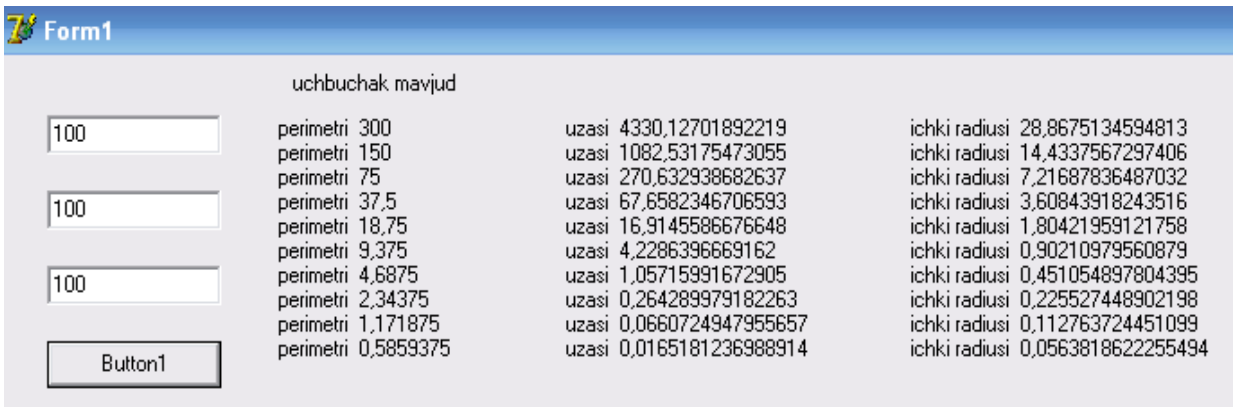
Misol:

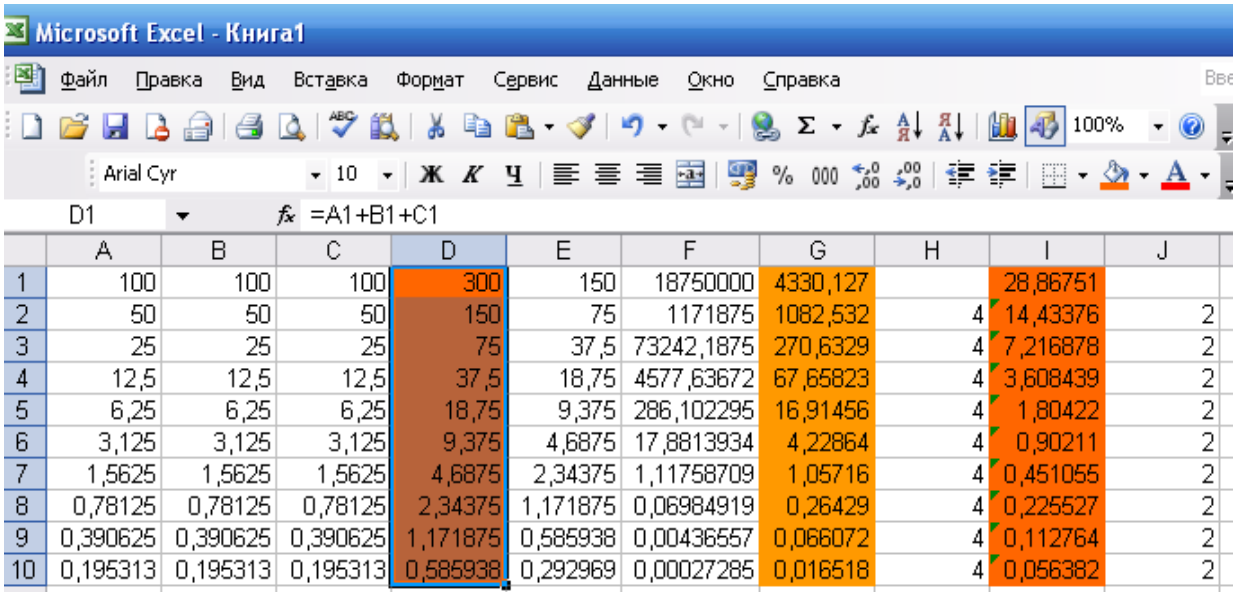
Dastlab uchburchakning uch tomoni berilgan va u birinchi uchburchak tomonlari xisoblanadi. Keyingi uchburchak tomonlari mos ravishda avvalgi uchburchak tomonlarining ikkiga bulingan qiymatiga teng. SHu qonuniyat asosida hosil bo'gan 10 ta uchburchaklar uchun perimetr, yuza, ichki chizilgan radiuslar qiymatini xisoblovchi dastur tuzing. (Natijani Excel dasturida xam xosil qiling)

Javob:

implementation

```
{$R *. dfm}
procedure TForm1. Button1Click(Sender: TObject);
var a,b,c,ri,rt:real;
T:INTEGER;
procedure aniq(x,y,z:real);
label 10,11,12;var javob:string;p,s:real;
begin
if (x+y>z) and (x+z>y) and (y+z>x) then goto 10 else goto 11;
10: label1. Caption:='uchbuchak mavjud'; p:= x+y+z;
label2. Caption:=LABEL2. CAPTION+'perimetri '+floattostr(p)+#13;
p:=p/2; s:=sqrt(p*(p-a)*(p-b)*(p-c)); ri:=s/p;
label3. Caption:=LABEL3. CAPTION+'uzasi '+floattostr(s)+#13;
 label4. Caption:=LABEL4. CAPTION+'ichki radiusi '+floattostr(ri)+#13;
goto 12;
11: label1. Caption:='uchburchak mavjud emas';
label2. Caption:='';
12: end;
begin
a:=strtofloat(edit1. Text);b:=strtofloat(edit2. Text);c:=strtofloat(edit3. Text);
A:=2*A;B:=2*B;C:=2*C; label2. Caption:='';
label3. Caption:=''; label4. Caption:='';
 FOR T:=1 TO 10 DO
 BEGIN
 A:=A/2;B:=B/2;C:=C/2; aniq(a,b,c);
 END;
end; end.
```





	A	B	C	D	E	F	G	H	I	J
1	100	100	100	300	150	18750000	4330,127		28,86751	
2	50	50	50	150	75	1171875	1082,532	4	14,43376	2
3	25	25	25	75	37,5	73242,1875	270,6329	4	7,216878	2
4	12,5	12,5	12,5	37,5	18,75	4577,63672	67,65823	4	3,608439	2
5	6,25	6,25	6,25	18,75	9,375	286,102295	16,91456	4	1,80422	2
6	3,125	3,125	3,125	9,375	4,6875	17,8813934	4,22864	4	0,90211	2
7	1,5625	1,5625	1,5625	4,6875	2,34375	1,11758709	1,05716	4	0,451055	2
8	0,78125	0,78125	0,78125	2,34375	1,171875	0,06984919	0,26429	4	0,225527	2
9	0,390625	0,390625	0,390625	1,171875	0,585938	0,00436557	0,066072	4	0,112764	2
10	0,195313	0,195313	0,195313	0,585938	0,292969	0,00027285	0,016518	4	0,056382	2

Misol.

```
Procedure Dr(Var x,h1,h2,z1,z2 : Real);
        Var h,z: Real;
        Begin
            h:=h1/z1+h2/z2;
            z:=z1/z2;
            x:=(h+z)/2;
        End;
```

Bu prosedurada h1,z1,h2,z2 parametrlar qiymati proseduraga murojat qilinganda aniqlangan bo'lishi kerak. Natijani esa x- parametr uzatadi. h va z o'zgaruvchilar ichki o'zgaruvchilardir. Bu proseduraga dasturdan quyidagicha murojaat qilinadi.

Dr(x,h1,h2,z1,z2);

Proseduraga murojaat qilinganda mos parametrlar qiymati bir biriga uzatiladi. Beriladigan formal va faktik parametrlar soni teng va ular turlari bir xil bo'lishi shart. Lekin parametrlar nomlari har xil bo'lishi mumkin.

Funksiyalardan foydalanish va ularni tashkil qilish xuddi prosedura kabi bo'lib, ularni e'lon qilish dasturning bosh qismida keltiriladi va u quyidagicha boshlanadi:

Function <f-ya nomi>(<formal parametrlar>):<f-ya turi>;

M. Function Min (x,y:Real): Real;

Funksiya nomi foydalanuvchi tamonidan beriladi. Funksiyaga murojaat qilish uning nomi orqali beriladi.

Funksiya ham proseduraga o'xshab bosh va asosiy qismlardan toshkil topadi. Funksiyaning proseduradan farqi, unga murojat qilinganda natija faqat bitta bo'lib, u shu funksiya nomiga uzatiladi.

Misol 1. Juyidagi hisoblashni funksiyani ishlatgan holda dastursini tuzing.

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

```
Program Kol;
 Var ncm,n,m,l: Integer;
 Function Fact (k: Integer): Integer;
   Var P,i: Integer;
 Begin
   P:=1;
   For i:=1 to k do P:=P*I;
   Fact:=P;
 End;
 Begin
   Read(n,m); l:=n-m;
   ncm:=Fact(n)/Fact(m)/Fact(l);
   Write('ncm=',ncm);
 End.
```

Misol 2. Quyidagi hisoblashni prosedurani ishlatgan holda dastursini tuzing.

$$z = \frac{tha - th^2(a-b)}{\sqrt{th(a^2 - b^2)}}, \quad thx = \frac{e^{2x} - 1}{e^{2x} + 1}.$$

```
Program Fun1;
 Var a,b,z,c,d,t1,t2,t3: Real;
 Procedure Th(Var x,r: Real);
   Var c: Real;
 Begin
   c:=exp(2. 0*x);
   r:=(c-1)/(c+1);
 End;
 Begin
   Read(a,b); th(a,t1);
   c:=a-b; th(c,t2);
   d:=Sqr(a)-Sqr(b); th(d,t3);
   z:=(t1+Sqr(t2))/Sqr(t3);
   Write('z=',z:10:3);
 End.
```

Oldindan e'lon qilish

Bizga ikki prosedura A va B berilgan bo'lib A prosedura B proseduraga murojaat qilsin va A ta'rifi B ta'rifidan oldin kelsin. Masalan:

```
Procedure A (i : integer);
begin
V (i);
Writeln(i);
end;
Procedure V (var j : integer) ;
begin
j:=j*2;
end;
```

Bunday ta’rif xatolikka olib keladi. Chunki A prosedura xali ta’riflanmagan proseduraga murojaat qilmokda. Bu holda B proseduranі quyidagicha oldindan e’lon kilish lozim:

```
Procedure V (var j : integer); Forward;
Procedure A (i : integer);
begin
V (i);
Writeln(i);
end;
Procedure V (var j : integer) ;
begin
j:=j*2;
end;
```

Oldindan e’lon qilishda prosedura tanasi standart direktiva Forward bilan almashtiriladi.

Mavzuga oid savollar

1. Qism dastur nima?
2. Funksiyaga ta’rif bering.
3. Protseduralar nima uchun kerak?
4. Formal parametrlar qayerda qo’llaniladi?

6.2. DELPHI DASTURLASH TILINING GRAFIK VOSITALARI.

Reja:

1. Delphi dasturlash tilining grafik imkoniyatlari.
2. Delphidagi maxsus TCanvas, TFont, TPen, Tbrush klasslari.
3. TFont klassi xossalari: Color, Name, Size, Style.
4. TPen klassi xossalari: Color, Mode, Width, Style.
5. TBrush klassi xossalari: Bitmap, Color, Style.

Tayanch iboralar: Canvas xossasi, chizish sohasi, TRen obyekti, TextOut uslubi, grafik primitivlarni chizish usullari, Image, Shape, PaintBox komponentalari. Delphidagi maxsus TCanvas, TFont, TPen, Tbrush klasslari. TFont klassi xossalari: Color, Name, Size, Style. TPen klassi xossalari: Color, Mode, Width, Style. TBrush klassi xossalari: Bitmap, Color, Style.

Delphi dasturchiga grafik dasturlar sxema, chertej, illyustrasiyalar yaratishga imkon beradi. Dastur grafikani obyekt (**forma** yoki **Image** komponentasi) yuzasiga chiqaradi. Obyekt yuzasiga **canvas** xossasi moc keladi. Obyekt yuzasiga grafik element (**to'g'ri chiziq, aylana, turtburchak** va hokazo), chiqarish uchun bu obyektning **Canvas** xossasiga mos usul qo'llash lozim. Misol uchun **Form1. Canvas. Rectangle (10,10,100,100)** instruksiyasi dastur oynasida turtburchak chizadi.

Chizish sohasi va uning o'lchovlari. Yukorida ko'rilgan **Canvas** xossasi - **TCanvas** tipidagi obyektdir. Grafik primitivlarni chiqarish usullari **Canvas** xossasini abstrakt chizish sohasi deb qaraydi. Chizish sohasi alohida nuqtalar - piksellardan iborat. Piksel holati uning gorizontal (X) va vertikal (Y) koordinatalari bilan aniqlanadi. Chap yuqori piksel koordinatalari (0,0). Koordinatalar yuqoridan pastga va chapdan o'ngga qarab o'sib boradi.

Soha o'lchovlarini Image komponentasining Height i **Width** xossalari va **formaning ClientHeight** va **Clientwidth** xossalari orqali aniqlash mumkin.

Qalam geometrik figuralarni chizish uchun ishlatiladi. Chiziq ko'rinishi **TRen** obyektining quyidagi jadvalda ko'rsatilgan xossalari orqali aniqlanadi.

TRen (qalam) xossalari.

Xossa	***Ta'rifi***
Color	Chiziq rangi
Width	Chiziq kalinligi
Style	Chiziq ko'rinishi

Mode	Akslantirish rejimi

Quyidagi jadvalda **Color** xossasi qiymati sifatida beriluvchi nomlangan konstantalar sanab o’tilgan.

Color xossasi qiymatlari**.**

Konstanta	**Rang**	**Konstanta**	**Rang**
clBlack	Qora	**clSilver**	Kumush rang
clMaroon	Kashtan rang	**clRed**	Qizil
clGreen	Yashil	**clLime**	Och yashil
clOlive	Och jigar rang	**clBlue**	Ko’k (zangori)
clNavy	Tim-ko’k	**clFuchsia**	Alvon
clPurple	Och qizil	**clAqua**	Moviy
clTeal	Pushti	**clWhite**	Oq
clGray	Kul rang		

Chiziq qalinligi **Width** xossasi orqali piksellarda beriladi.

Chiziq turini **Style** xossasi belgilaydi. Quyidagi jadvalda chiziq turini belgilovchi nomlangan konstantalar sanab o’tilgan.

Style xossasi qiymatlari

Konstanta	**Chiziq ko’rinishi**
psSolid	Uzluksiz chiziq
psDash	Punktir chiziq, uzun shtrixlar
psDot	Punktir chiziq, qisqa shtrixlar
psDashDot	Punktir chiziq, uzun va qisqa shtrixlar ketma ketligi
psDashDotDot	Punktir chiziq, bitta uzun va ikkita qisqa shtrixlar ketma ketligi
psClear	Chiziq aks etmaydi

Mode xossasi chiziq rangining fon rangiga munosabatini ko’rsatadi. Odatda chiziq rangi **Pen. Color** xossasi qiymati bilan belgilanadi.

Dasturchi chiziq uchun fon rangiga nisbatan invers rang berishi mumkin. Bu holda hatto chiziq va fon rangi bir xil berilgan bo’lsa ham chiziq ajralib turadi.

Quyidagi jadvalda **Mode** xossasi qiymati sifatida ishlatish mumkin bo’lgan konstantalar berilgan. Mode xossasi qiymatlari

Konstanta	**Chiziq rangi**
pmBlack	Qora, **Pen. Color** xossasi qiymatiga bogliq emas
pmWhite	Oq, **Pen. Color** xossasi qiymatiga bog’liq emas
pmCopy	Chiziq rangi **Pen. Color** xossasi qiymatiga bog’liq
pmNotCopy	Chiziq rangi **Pen. Color** xossasi qiymatiga invers
pmNot	Chiziq rangi sohaning mos nuqtasi rangiga invers

Muyqalam (**Canvas. Brush**) yopiq sohalarni chizish va soha ichini bo’yash uchun mo’ljallangan usullardan foydalaniladi. Muyqalam obyekt jadvalda ko’rsatilgan ikki xossaga ega.

TBrush (muyqalam) xossalari.

Xossa	**Ta’rifi**
Color	Yepiq sohani bo’yash rangi
Style	Sohani to’ldirish uslubi

Kontur ichidagi soha bo’yalishi yoki shtrixlanishi mumkin. Sohani to’ldirish usulini belgilovchi konstantalar quyidagi jadvalda berilgan.

Brush. style xossasi qiymatlari

Konstanta	**Soha bo’yash uslubi**
bsSolid	Uzluksiz bo’yash
bsClear	Soha bo’yalmaydi
bsHorizontal	Gorizontal shtrixlash
bsVertical	Vertikal shtrixlash
bsFDiagonal	Diagonal shtrixlash, oldinga og’ish

bsBDiagonal	Diagonal shtrixlash, orqaga og'ish
bsCross	Katakli gorizontal-vertikal shtrixlash
bsDiagCross	Katakli diagonal shtrixlash

Grafik obyekt yuzasiga matn chiqarish uchun **TextOut** usuli qo'llaniladi. Bu usulni chaqirish instruksiyasi quyidagi ko'rinishga ega:

Obyekt. Canvas. TextOut(x, u, Tekst)

Matn shrifti Font xossasi qiymati bilan aniqlanadi. **Font** xossasi **TFont** tipidagi obyekdir. Quyidagi jadvalda **TFont** obyekti xossalari keltirilgan.

TFont obyekti xossalari

Xossa	**Ta'rifi**
Name	Shrift nomi, masalan Arial
Size Style	Shrift punktlarda kattaligi Simvollar chiqarish uslubi. Quyidagi konstantalar orqali beriladi: **fsBold** (qoraroq), **fsltalic** (qiya), **fsUnderline** (tagiga chizilgan) **fsStrikeOut** (ustiga chizilgan).
Font Color	Bu xossa bir necha uslublarni kombinasiyasini olishga imkon beradi. Masalan: **Obyekt. Canvas. Font:=[fsBold, fsItalic]** Simvollar rangi.

Matn chiqarish sohasi muyqalam joriy rangiga buyaladi. Shuning uchun matn chiqarishdan oldin **Brush. Color** xossasiga **bsClear** qiymatini yoki soha rangiga mos qiymatni berish lozim.

TextOut uslubi orqali matn ekranga chiqarilgandan so'ng qalam matn chiqarish sohasining yuqori o'ng burchagiga keltiriladi.

Grafik primitivlarni chizish usullari To'g'ri chiziq **LinyeTo** usuli orqali amalga oshiriladi: **Komponent. Canvas. LineTo(x,u),**

LinyeTo usuli qalam joriy pozisiyasidan berilgan koordinatali nuqtagacha to'g'ri chiziq chizadi. Boshlangich nuqtani kerakli nuqtaga ko'chirish uchun **MoveTo** usulidan foydalanish mumkin.

Aylana yoki ellips chizish uchun Ellipse usuli chaqiriladi. Usulni chaqirish instruksiyasi umumiy ko’rinishi:

Obyekt. Canvas. Ellipse(x1,y1, x2,u2).

Bu yerda x1, y1, x2, u2 – ellipsni o’z ichiga olgan minimal turtburchak koordinatalari. Agar turtburchak kvadrat bo’lsa aylana chiziladi.

Yoyni chizish uchun Arc usuli qo’llaniladi va u quyidagi umumiy ko’rinishga ega:

Obyekt. Canvas. Arc(x1,y1,x2,y2,x3,u3,x4,u4)

Bu yerda:

 x1, y1, x2, u2 - yoyga tegishli bo’lgan ellips yoki aylana parametrlari;

 x3, u3 - yoy boshlang’ich nuktasi parametrlari;

 x4, u4 - so’ngi nuqtasi parametrlari. Yoy soat miliga teskari tartibda chiziladi.

To’rtburchak **Rectangle** usuli bilan chizilib, bu usulni chaqirish instruksiyasi umumiy ko’rinishi quyidagicha:

Obyekt. Canvas. Rectangle(x1, y1,x2, y2)

Bu yerda x1, y1 va x2, u2 — chapgi yuqori va o’nggi pastgi burchaklar koordinatalari.

RoundRec usuli burchaklari yumaloq to’rtburchak chizishga imkon beradi. **RoundRec** usulini chakirish instruksiyasi kuyidagi kurinishga ega: **Obyekt. Canvas. RoundRec(x1,y1,x2, u2, x3, u3)**

Bu yerda:

 x1, y1, x2, u2 – turtburchak parametrlari;

 x3 i u3 — chorak kismi yumalok burchak chizish uchun ishlatildadigan ellips kattaligi.

Ellips yoki aylana sektori **Pie** usuli bilan chizilib, chaqirish instruksiyasi quyidagi umumiy ko’rinishga ega:

Obyekt. Canvas. Pie(x1,y1,x2,y2,x3,u3,x4,u4)

Bu yerda:

 x1, y1, x2, u2 - ellips yoki aylana parametrlari;

 x3, u3, x4, u4 - sektor chegarasini tashkil qiluvchi to’g’ri chiziqlar oxirgi nuktalari koordinatalari.

Nuqta. **Canvas** obyektining **Pixels** xossasi tipidagi ikki o’lchovli massiv bo’lib har bir soha nuqtasining rangi haqidagi ma’lumotni o’z ichiga oladi.

Pixels xossasidan foydalanib ixtiyoriy nuqta rangini o’zgartirish, ya’ni nuqta chizish mumkin. Misol uchun

Form1. Canvas. Pixels[10,10]:=clRed

Instruksiyasi soha nuqtasini qizil ranga bo’yaydi.

Grafik komponentalar

Image komponentasi formaga rasmlarni joylashtirish uchun ishlatiladi. Joylashtirilishi lozim bo’lgan rasmlar bitli fayllar (kengaytmalari-. **bmp**), piktogrammali (kengaytmalari-**. ico)**, metafayllar (kengaytmalari-. **wmf**) bo’lishi kerak.

Image komponentasi **Additional** palitrasida joylashgan bo’lib, u  ko’rinishdagi piktogrammaga ega. Bu tugmachani bosib formadan rasm uchun joy ajratiladi va keyin esa xossalar bo’limidan **Picture** xossasi tanlanib, u yerdan uch nuqtali tugmacha bosiladi. Natijada ekranda rasmni aniqlash va joylash uchun

muloqat darchasi ochiladi. Muloqat darchasi quyidagi tugmachalarga ega:

Load -fayldan rasmni chaqirish;

Save -rasmni faylga saqlash;

Clear -tanlangan rasmni olib tashlash;

Ok -tanlangan rasmni ajratilgan joyga yozish;

Cancel -qilingan o’zgartirishlarni bekor qilish.

Shape komponentasi formaga aylana, to’rtburchak, ellips va boshqa shakllarni joylashtirish uchun ishlatiladi. Uning quyidagi xossalari mavjud:

Brush -shaklni bo’yash uchun cho’tkacha; **Pen** -shakl chetini chizish uchun qalam; **Shape** -ekranga chiqadigan shaklni aniqlaydi:

StRectangle -to’rtburchak;

StSquare -kvadrat;

StRoundRect -chetlari aylana to’rtburchak; **StRoundSquare** -chetlari aylana kvadrat; **StEllipse** -ellips;

StCircle -aylana.

Shape komponentasi ham **Additional** palitrasida joylashgan bo’lib, u  ko’rinishdagi piktogrammaga ega. Bu tugmachani bosib formadan shakl uchun joy ajratiladi va keyin esa xossalar bo’limidan Shape xossasiga kirilib kerakli shakl tanlanadi.

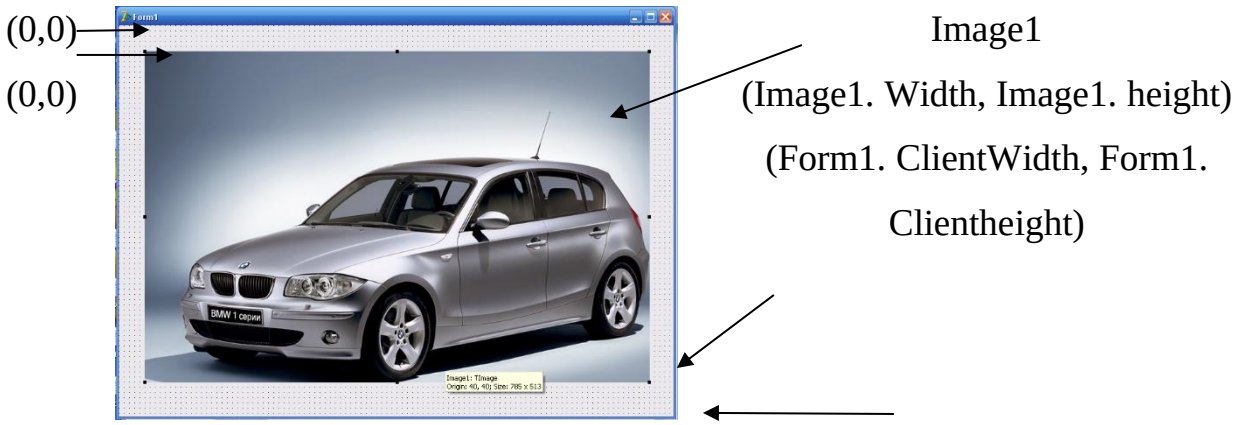
PaintBox komponentaci formaga chegaralangan maydonda shakllarni chizish imkonini beradi.

PaintBox komponentasi **System** palitrasida joylashgan bo’lib, u  ko’rinishdagi piktogrammaga ega. Bu tugmachani bosib formadan shakl uchun joy ajratiladi va keyin esa xossalar bo’limidan **Shape** xossasiga kirilib kerakli shakl tanlanadi.

Delphi dastur yaratuvchilarga qator grafikali elementlar bilan ishlash imkoniyatlarini beradi. Dasturchi o’zining dasturida sxema, chizmalar va illyustratsiyalarni joylashtira oladi. Grafik odatda ob’ektning (Forma yoki Im age komponenti) satxiga chiziladi va bu satxga Canvas xossasi mos qo’yiladi. Odatda ob’ektning satxiga chizish uchun Canvas xossasiga mos metod qo’llaniladi.

Masalan, canvas. Pen. Color:=clred;

Umuman, chiziladigan sath-maydon alohida nuqtalar-piksellardan tashkil topgan. Nuqtaning pozitsiyasi uning gorizontal(X) va vertikal(U) koordinatalari bilan belgilanadi. YUqori chap nuqtaning koordinatalari (0,0) dan iborat. Koordinatalar qiymatlari tepadan pastga va chapdan o’ngga ortib boradi.



Rassom o’z faoliyatida qalam va mo’yqalamdan foydalanganidek, Delphi ham o’zining qalami (Pen) va mo’yqalamidan(Brush) o’zining metodlarida foydalanadi. SHriftdan foydalanish uchun Delphida maxsus imkoniyatlarga ega Tfont klassi mavjud.

TFont klasi.

Bu klass yordamida har qanday grafik qurilma uchun (ekran, printer, plotter va boshqlar) uchun ob’ekt-shrift yaratiladi. Klassning xossalarini keltiramiz:

Xossa	Xossaning tavsifi(misol)
property Charset: TFontCharset;	Simvollar to’plami(canvas. Font. Charset:= default_charset;)
property Color: TColor;	SHriftning rangi(canvas. Font. Color:=clLime;)
property Name: TFontName;	SHriftning nomi(canvas. Font. Name:='Balica Tad';)
property Height: Integer;	SHriftning balandligi(sanvas. Font. height:=84;)
property Size: Integer;	SHriftning o’lchami(sanvas. Font. size:=22;)

property Style: TFontStyles;	SHriftning ko’rinishi quyidagi alomatlar kombinatsiyasi bo’lishi mumkin :fsbold, fsitalic, fsunderline, fsstriceout (canvas. Font. Style:=[fsitalic,fsbold];)

TPen klasi.

Bu klass yordamida ob’ekt-pero uchun yaratiladi. Uning yordamida chiziqlar chiziladi va u quyidagi xossalarga ega:

Xossa	Xossaning tavsifi(misol)
property Mode: TpenMode;	CHiziqning fon rang bilan o’zaro moslashishi
property Color: TColor;	CHiziqning rangi(canvas. Pen. Color:=clLime;)
property Style: TPenstyle;	CHiziqning ko’rinishi(faqat qalinligi 1 pikselь bo’lgan chiziqlarga qo’llaniladi, boshqa qalinlikdagi chiziqlar uchun psSolid aniqlanadi)
property Width: Integer;	CHiziqning qalinligi (pikselda)

Mode xossasing ayrim qiymatlarini ketiramiz:

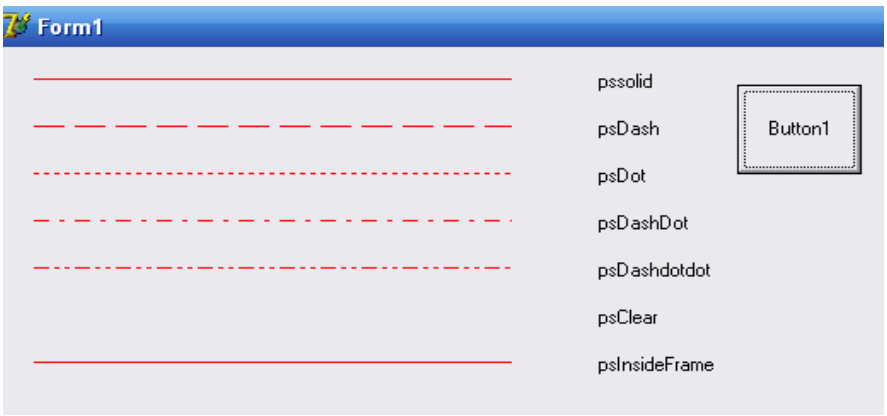
- pm Black-chiziqlarning rangi qora, Color va Style xossalari bekor qilinadi;
- pmWhite - chiziqlarning rangi oq, Color va Style xossalari bekor qilinadi;
- pm Nop –fon rangi o’zgarmaydi (chiziqlar ko’rinmas holda);
- pm Not - fon rangi inversiyasi, Color va Style xossalari bekor qilinadi;
- pm Copy - chiziqlarning peroning Color xossasi bilan aniqlanadir;
- pm NotCopy – pero rangining inversiyasi, Style xossasi bekor qilinadi;
- pm mergePenNot – pero va fonning inversiya ranglari kombinatsiyasi;
- pm MaskPenNot- pero va fonning inversiya ranglari kombinatsiyasi, Style xossasi bekor qilinadi;

(Qolgan xossalar bilan turli adabiyotlar orqali tanishiladi)

Quyida keltirilgan dasturda siz bu klasslarning xosalaridan foydalanish yo’llari va psStyle xoccacining imkoniyatlari Bilan tanishasiz:

```
procedure TForm1. Button1Click(Sender: TObject);
begin
with canvas do
begin Pen. Color:=clred; pen. Width:=1;
moveto(20,20); pen. Style:=pssolid; lineto(300,20); TextOut(350,15,'pssolid');
moveto(20,50); pen. Style:=psDash; lineto(300,50); TextOut(350,45,'psDash');
moveto(20,80); pen. Style:=psDot; lineto(300,80); TextOut(350,75,'psDot');
moveto(20,110); pen. Style:=psDashDot; lineto(300,110);
TextOut(350,105,'psDashDot');
moveto(20,140); pen. Style:=psDashdotdot; lineto(300,140);
TextOut(350,135,'psDashdotdot');
moveto(20,170); pen. Style:=psClear; lineto(300,170);
TextOut(350,165,'psClear');
moveto(20,200); pen.
Style:=psInsideFrame;lineto(300,200);TextOut(350,195,'psInsideFrame');
end; end;
end.
```

Dasturning natijasi:



TBrush klasi(Muykalam).

Bu klass xossalaridan yopik soxalarni chizuvchi va bu soxalarni buyash uchun ishlatiladigan metodlarda foydalanish mumkin. Muykalam ob’ekt sifatida ikki xossaga ega:

Xossalar	Xossaning tavsifi
Color	yopik soxani buyash rangi
Style	yopiq soxani to’ldirish stili

Grafik ob’ekt sathida matnni aks ettirish

Grafik ob’ekt sathida matnni aks ettirish uchun TextOut metodidan foydalanamiz va uning ko’rinishi quyidagicha:

Ob’ekt. Sanvas. TextOut(x,y,tekst);

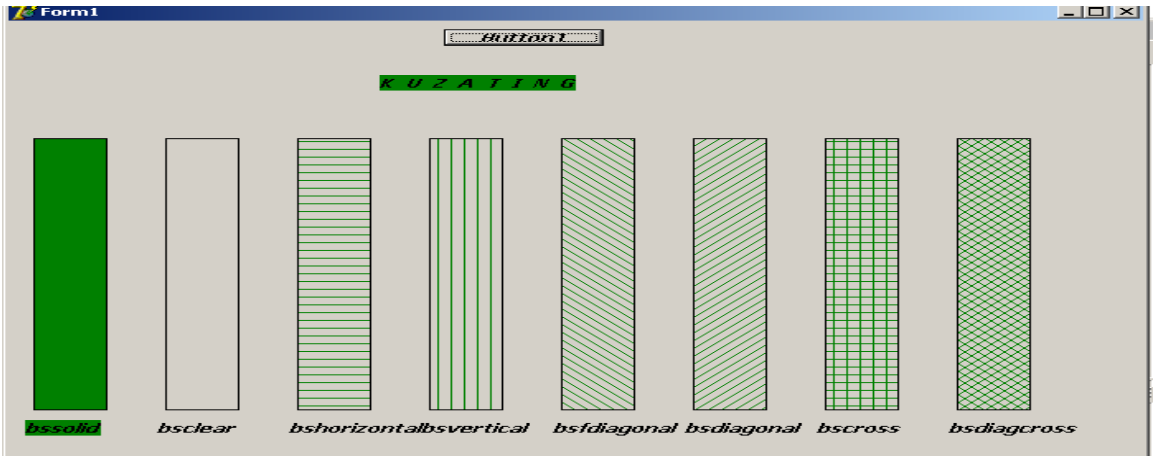
Bu erda:

- Ob’ekt-ustida matn yoziluvchi ob’ektning nomi;
- x,y- matn yozilish koordinatasi;
- tekst-satriy tipdagi doimiylik yoki o’zgaruvchi;

Quyida yuqoridagi xossalarni aks ettiruvchi dastur va dastur natijasini keltiramiz:

```
procedure TForm1. Button1Click(Sender: TObject);
const
bsname: array[1. . 8] of
string=('bssolid','bsclear','bshorizontal','bsvertical','bsfdiagonal','bsdiagonal','bscross','
bsdiagcross');
var x:integer; bs:TBrushstyle; i,j,k:integer;
 begin
 Font. Name:='Tahoma'; Font. Size:=10; font. Style:=[fsBold,fsitalic]; Brush.
Style:=bsclear;
 x:=0;
 for k:=1 to 8 do
 begin
 case k of
 1:bs:=bssolid;
 2:bs:=bsClear;
 3:bs:=bsHorizontal;
 4:bs:=bsVertical;
```

```
 5:bs:=bsFDiagonal;
 6:bs:=bsBDiagonal;
 7:bs:=bsCross;
 8:bs:=bsDiagCross;
 end;
 canvas. Brush. Color:=ClGreen; canvas. Brush. Style:=bs; canvas.
rectangle(15+x,120,60+x,400);
 canvas. TextOut(x+10,410,bsname[k]); x:=x+80; canvas. TextOut(225,55,'K U Z A
T I N G');
 end; end;
end.
```



Grafik primitivlarni aks ettirish

Har qanday rasm oddiy grafik primitivlar to’plamidan(nuqta, to’g’ri chiziq, aylanalar va xk) iborat. Buning uchun turli metodlardan foydalanamiz.

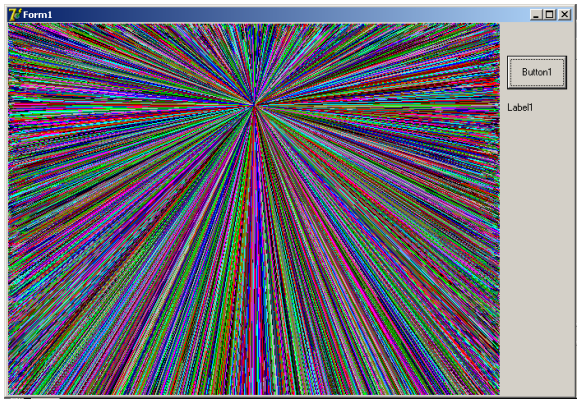
To’g’ri chiziq chizish uchun Lineto metodi ishlatiladi. Uning umumiy ko’rinishi quyidagicha:

canvas. LineTo(x,y);

Bu holda metod «joriy» nuqtadan (x,y) nuqtagacha Pen ob’ekti xossalari Bilan to’g’ri chiziq chizadi va bu holatlarni quyidagi dastur yordamida kuzatish mumkin.

```
procedure TForm1. Button1Click(Sender: TObject);
 var x,y,r,t:integer;
 function tch(k:integer):tcolor;
```

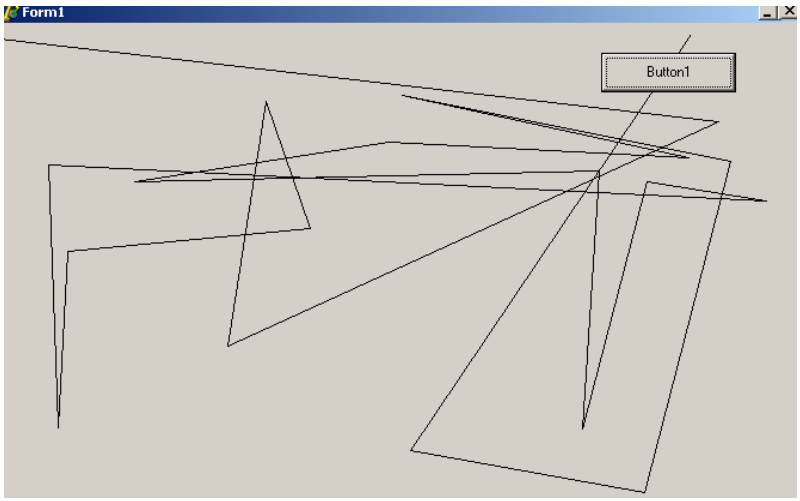
```
 var bs:tcolor;
 begin
 case k of
 0:bs:=clblack;
 1:bs:=clMaroon;
 2:bs:=clGreen;
 3:bs:=clOlive;
 4:bs:=clNavy;
 5:bs:=clPurple;
 6:bs:=clTeal;
 7:bs:=clGray;
 8:bs:=clSilver;
 9:bs:=clRed;
 10:bs:=clLime;
 11:bs:=clBlue;
 12:bs:=clFuchsia;
 13:bs:=clAqua;
 14:bs:=clWhite;
 end;
 tch:=bs;
 end;
begin
FOR t:=1 TO 1000000 do
BEGIN
randomize;canvas. MoveTo(300,100); r:=random(14);
x:=random(600);Y:=RANDOM(600);canvas. pen. Color:=tch(r); canvas.
LineTo(x,y);
end;end;end.
```



Siniq chiziq

Ciniq chiziqni chizish uchun Polyline metodidan foydalaniladi. Bu metodning parametri sifatida TPoint massivi ishlatiladi. Polyline metodi massivning birinchi elementini ikkinchisi bilan, ikkinchi elementini uchinchisi bilan va hk ulab siniq chiziqni chizib beradi.

```
procedure TForm1. Button1Click(Sender: TObject);
var nuqtalar:array [1. . 20] of Tpoint;
 i:integer;
 begin
 for i:=1 to 20 do
 begin
 nuqtalar[i]. x:=random(700); nuqtalar[i]. y:=random(400);
 end; canvas. Polyline(nuqtalar);end;
end.
```



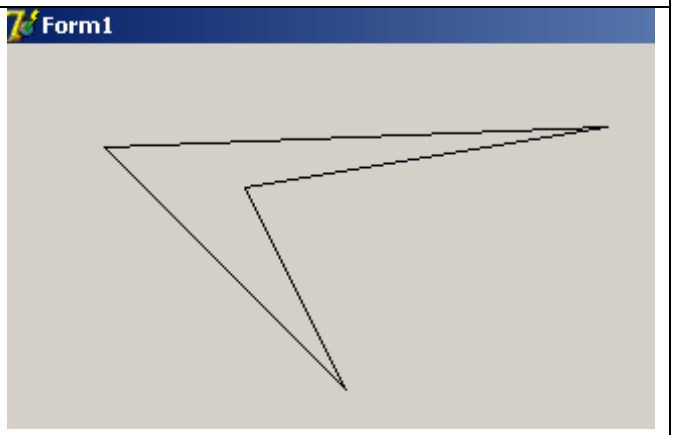
Polyline metodidan foydalanib yopiq kontur chizishda birinchi va oxirgi nuqtalar koordinatalari mos kelishi shart. Bunga quyidagi misolni keltiramiz.

```
procedure TForm1. Button1Click(Sender: TObject);
var
 nuqtalar:array [1. . 5] of Tpoint;
 begin
 nuqtalar[1]. x:=50; nuqtalar[1]. y:=50; nuqtalar[2]. x:=170; nuqtalar[2]. y:=170;
 nuqtalar[3]. x:=120; nuqtalar[3]. y:=70; nuqtalar[4]. x:=300; nuqtalar[4]. y:=40;
 nuqtalar[5]. x:=50; nuqtalar[5]. y:=50;
canvas. Polyline(nuqtalar);
end; end.
```

Ko’pburchak

Ko’pburchak shaklini chizish uchun Polygon metodidan foydalaniladi. Bu metodning parametri sifatida TPoint massivi(nuqtalar koordinatlari to’plami) ishlatiladi. Polyline metodi massivning birinchi elementini ikkinchisi bilan, ikkinchi elementini uchinchisi bilan va hk ulanib oxirida birinchi va oxirgi nuqtalar ulanib ko’pburchak xosil qilinadi.

Dastur	Natija

procedure TForm1. Button1Click(Sender: TObject);
var
nuqtalar:array [1. . 4] of Tpoint;
begin
nuqtalar[1]. x:=50;nuqtalar[1]. y:=50;
nuqtalar[2]. x:=170; nuqtalar[2]. y:=170;
nuqtalar[3]. x:=120;nuqtalar[3]. y:=70; nuqtalar[4]. x:=300; nuqtalar[4]. y:=40;
canvas. Polygon(nuqtalar);
end; end.
	

Aylana, ellips va yoy

Ellipse metodi ellips va aylana chizishda qo’llaniladi. Uning umumiy ko’rinishi quyidagicha:

Bu erda:

- x1,y1,x2,y2-to’g’ri to’rtburchak koordinatalari(uning biror diagonallari uchlari koordinatasi)

Izoh: agar to’g’ri to’rtburchak kvadrat bo’lsa, aylana aksincha ellips chiziladi.

YOy chizish uchun Arc metodidan foydalanamiz. Uning umumiy ko’rinishi quyidagicha:

Ob’ekt. Canvas. Arc(x1,y1,x2,y2,x3,y3,x4,y4)

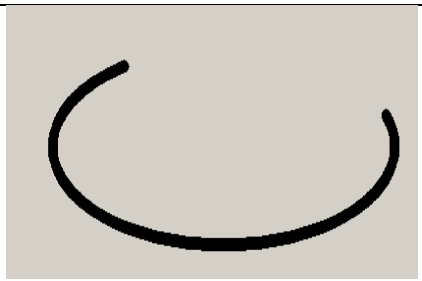
Bu erda:

- x1,y1,x2,y2-to’g’ri to’rtburchak koordinatalari(uning biror diagonallari uchlari koordinatasi)
- x3,y3- yoyning boshlang’ich nuqtasi koordinatalari
- x4,y4 -yoyning oxirgi nuqtasi koordinatalari

YOy boshlang’ich nuqtadan yoyning oxirgi nuqtasiga soat mili harakatiga nisbatan qarshi yo’nalishda chiziladi. CHiziqning ko’rinishi Pen xossalari bilan aniqlanadi.

Misol:

Dastur	Natija

procedure TForm1. Button1Click(Sender: TObject);
begin
canvas. Pen. Width:=10;
canvas.
Arc(250,250,600,400,300,250,600,300);
end;
end.
	

Quyida turli ranglardagi kontsentrik aylanalar chizuvchi dastur keltirilgan.

```
procedure TForm1. Button1Click(Sender: TObject);
var a,b,d:string; xm,ym,xb1,yb1,xb2,yb2,t,r:integer;
 function tch(k:integer):tcolor;
 var bs:tcolor;
 begin
 case k of
 0:bs:=clblack; 1:bs:=clMaroon; 2:bs:=clGreen; 3:bs:=clOlive;
 4:bs:=clNavy; 5:bs:=clPurple;
 end;
 tch:=bs;
 end;
begin
a:=inputbox('aylana markazi absissasi','kiriting','');
b:=inputbox('aylana markazi ordinatasi','kiriting','');
d:=inputbox('raduslar orttirmasi', 'kiriting','');
 xm:=strtoint(a); ym:=strtoint(b);
 for t:= 120 downto 1 do
 begin
 xb1:=xm-t*strtoint(d); xb2:=xm+t*strtoint(d);
 yb1:=ym-t*strtoint(d); yb2:=ym+t*strtoint(d);
 canvas. Ellipse(xb1,yb1,xb2,yb2);
 r:=random(5); canvas. Pen. Color:=tch(r);
end;end; end.
```

To’g’ri to’rtburchak

To’g’ri to’rtburchak Rectangle metodi yordamida chiziladi. Uning umumiy ko’rinishi quyidagicha:

Ob’ekt. Canvas. Rectangle(x1,y1,x2,y2)

Bu erda:

- x1,y1,x2,y2-to’g’ri to’rtburchak koordinatalari(uning biror diagonallari uchlari koordinatasi)

RoundRec metodi uchlari ovalsimon to’g’ri to’rtburchak chizadi. Uning umumiy ko’rinishi quyidagicha:

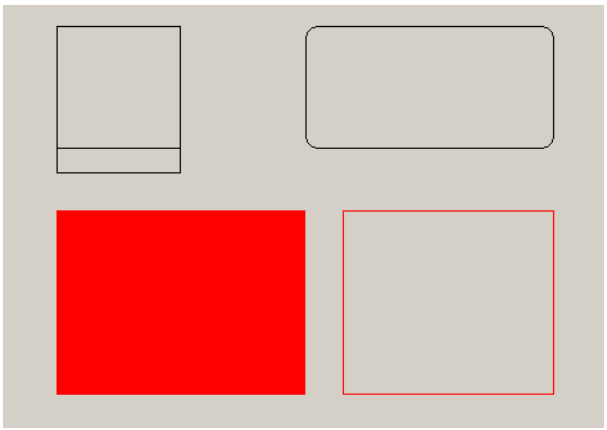
Ob’ekt. Canvas. RoundRec(x1,y1,x2,y2,x3,y3). Bu erda:

- x1,y1,x2,y2-to’g’ri to’rtburchak koordinatalari(uning biror diagonallari uchlari koordinatasi)
- x3,y3- oval burchak yaratish uchun foydalanadigan ellipsning to’rtdan bir qismini ifodalovchi andoza parametrlari

CHiziqning ko’rinishini (rang, qalinligi, stili) Pen xossalari bilan, to’g’ri to’rtburchak ichki sohasini bo’yash uchun Brush xossalaridan foydalaniladi.

To’g’ri to’rtburchak chizish uchun yana faqat instrument sifatida Brushdan foydalanuvchi ikki metod mavjud. FillRect metodi ichi bo’yalgan to’g’ri to’rtburchak chizsa, FrameRect metodi uning faqat konturini chizadi. Bu ikki metodning yagona Trect tipidagi parametri mavjud.

Dastur	Natija

procedure TForm1.
Button1Click(Sender: TObject);
var r1,r2:Trect;
begin
sanvas. Rectangle(200,100,100,220);
canvas. Rectangle(100,200,200,100);
canvas.
RoundRect(300,100,500,200,20,20)
r1:=rect(100,250,300,400);
r2:=rect(330,250,500,400);
canvas. brush. Color:=clred; form1.
canvas. fillrect(r1);
canvas. FrameRect(r2);
end;end.
	

Nuqta

Canvas ob’ektining Pixcels xossasi yordamida nuqtani tasvirlash mumkin. Bunda nuqtaning koordinatalari va uning rangi beriladi. Masalan, canvas.

Pixels[150,250]:=clblack;

Quyida nuqtalar yordamida hosil qilingan tasvirga e’tibor bering.

Dastur	Natija

procedure TForm1.
Button1Click(Sender: TObject);
var a,b,t,x,y:integer;
function tch(k:integer):tcolor;
var bs:tcolor;
begin
case k of
0:bs:=clblack; 1:bs:=clMaroon;
2:bs:=clGreen; 3:bs:=clOlive;
end; tch:=bs;
end;
begin
for t:=1 to 200000 do
begin
a:=random(800); b:=random(700);
canvas. Pixels[a,b]:=tch(random(14));
end;
canvas. Pen. Width:=10; canvas. Pen. Color:=clblue;
canvas. RoundRect(100,100,600,300,110,110);
canvas. font. Size:=25;
for t:=1 to 50 do
begin
canvas. Font. Height:=t;
	

canvas. Textout(150,150,'fizika-matematika fakulteti'); end;end; end. 2	

```
procedure TForm1. Formmousedown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);
begin
 If button=mbleft
 Then Form1. canvas. pen. color:=clred else form1. canvas. Pen. Color:=clgreen;
 canvas. Ellipse(20,200,120,340);
end;
```

Mavzuga oid savollar

1. Soha o’lchovlari qaysi xossalar yordamida aniqlanadi?
2. Matn shrifti qaysi xossalar orqali aniqlanadi?
3. Grafik obyekt yuzasiga matn chiqarish uchun qaysi usul qo’llaniladi va bu usulni chaqirish instruksiyasi ko’rinishini keltiring.
4. To’g’ri chiziq qaysi usul orqali amalga oshiriladi va bu usulni chaqirish instruksiyasi ko’rinishini keltiring.
5. Aylana yoki ellips chizish uchun qaysi usul chaqiriladi va bu usulni chaqirish instruksiyasi ko’rinishini keltiring.
6. Yoyni chizish uchun qaysi usul qo’llaniladi va u qanday umumiy ko’rinishga ega.
7. To’rtburchak chizish uchun qaysi usul qo’llaniladi va u qanday umumiy ko’rinishga ega.
8. Ko’pburchak chizish uchun qaysi usul qo’llaniladi va u qanday umumiy ko’rinishga ega.
9. Ellips yoki aylana sektorini chizish uchun qaysi usul qo’llaniladi va u qanday umumiy ko’rinishga ega.

10. Image komponentasi formaga nimani joylashtirishda qo’llaniladi?

11. Shape komponentasi formaga nimalarni joylashtirishda qo’llaniladi.

6.3. DELPHIDA FAYLLAR BILAN ISHLASH.

Reja:

1. Faylning vazifasi va turlari
2. Fayllar bilan ishlovchi funksiyalar
3. Fayl tashkil qilish. Fayllar bilan ishlash
4. Muloqot oynalarini yaratish

Tayanch iboralar: matnli fayl, tiplashgan fayl, tiplashmagan fayl, faylli o’zgaruvchi, standart muloqot oynasi, tugmalar.

Fayl – bu tashqi xotiradagi ma’lumotlarning nomlangan sohasi. Object Pascal da ularni tashkillashtirish va elementlariga murojaat qilish uslublariga ko’ra fayllarning uch xil ko’rinishi farqlanadi: matnli, tiplashgan va tiplashmagan.

Fayl deb kompyuer tashqi xotarasining nomlangan soxasiga aytiladi va uning uchta xarakterli xususiyatlari mavjud. Birinchidan, uning nomi mavjud bulib, bu nom dasturda foydalaniladi. Ikkinchidan uning kompponentlari bir tipga mansub va bu fayl tipidan boshqa barcha tiplar bulishi mumkin. Uchinchidan yangi yaratiluvchi faylning uzunligi haqida uning e’lon qilish vaqtida “fikr yuritilmaydi” va bu faqat tashqi xotira elementining xajmiga bog’liq

Matnli fayl – qatorlardan iborat fayl. Matnli faylga DELPHI dagi (kengaytmasi *. pas) dasturning dastlabki matn fayli misol bo’ladi. Matnli fayl bilan ishlash uchun mos faylli o’zgaruvchi tavsiflanishi zarur: ***Var F: TextFile;***

Tiplashgan fayllar barcha elementlari fiksirlangan va bir xil o’lchovli ega bo’lgan, bu esa ularning har biriga to’g’ridan-to’g’ri murojaat qilishni tashkillashtirish imkonini beradi (elementga murojaat uning tartib nomeri orqali amalga oshiriladi). Odatda, bunday fayllarning elementlari yozuv bo’ladi. Faylli o’zgaruvchi tavsiflanishida uning tipi ko’rsatiladi: ***Var F: TextFile;***

Tiplashmagan fayl – bu ma’lumotlari ma’lum tipga ega bo’lmagan va baytlar ketma-ketli sifatida qaraladigan fayl. Faylli o’zgaruvchi e’lon qilinadi: ***Var F: File;***

Fayllar bilan ishlash tartibi quyidagicha:

. . .

AssignFile(F, ‘Filename. txt’);

// “Filename. txt” fayl bilan

// F faylli o’zgaruvchini bog’lash

Rewrite(F);

// yangi yaratish yoki (**Reset**(F);) mavjud

// faylni ochish

Read(F, Stud);

// ma’lumotni fayldan o’qish yoki (**Write**(F, Stud)) faylga yozish

. . .

CloseFile(F);

// faylni yopish

Fayllar bilan ishlovchi qism dasturlar:

AssignFile(var F; FileName: string) – **F** faylli o’zgaruvchi va **FileName** nomli faylni bog’lash.

Reset(var F[: File; RecSize: word]) – mavjud faylni ochadi. Tiplashmagan faylni ochishda **RecSize** fayl elementining o’lchovi beriladi.

Rewrite(var F[: File; RecSize: word]) – yangi fayl yaratiladi va ochiladi.

Append(var F: TextFile) – fayl oxiriga matn yozish uchun matnli fayl ochiladi.

Read(F,v1[,v2,…vn]) – tiplashgan fayldan joriy ko’rsatgichdan boshlab qiymatlarni va matnli fayldan satrni o’qish.

Write(F,v1[,v2,…vn]) – tiplashgan fayldan joriy ko’rsatgichdan boshlab qiymatlarni va matnli fayldan satrni yozish.

CloseFile(F) – oldin ochilgan fayl yopiladi.

Rename(var F; NewName: string) – ixtiyoriy tipdagi ochilmagan faylni qayta nomlaydi.

Erase(var F) – ixtiyoriy tipdagi ochilmagan faylni o’chiradi.

Seek(var F; NumRec: Longint) – matnli bo’lmagan fayllar uchun

NumRec nomerli elementga ko’rsatgichni o’rnatadi.

SetTextBuf(var F: TextFile; var Buf[;Size: word]) – matnli fayl uchun **Size** hajmdagi kiritish-chiqarish yangi buferini aniqlaydi.

Flush(var F: TextFile) – kiritish-chiqarish buferi qiymatini faylga yozish.

Truncate(var F) – joriy pozisiyadan boshlab faylning oxirigi elementigacha o’chiradi.

LoResult: integer - oxirgi kiritish-chiqarish amalining natijasining kodi.

FilePos(var F):longint – matnli bo’lmagan fayllar uchun joriy pozisiya nomeri aniqlaydi. Hisoblash noldan boshlanadi.

FileSize(var F):longint – matnli bo’lmagan fayllar uchun fayldagi elementlar sonini aniqlaydi.

Eoln(var F: TextFile): boolean – kursor satr oxiriga o’rnatilsa True qiymat beradi.

Eof(var F)): boolean - kursor fayl oxiriga o’rnatilsa True qiymat beradi.

SeekEoln(var F: TextFile): boolean – probel yoki tabulyasiya belgisidan farqli satr yoki fayldagi oxirgi simvolga kelganda **True** qiymatga ega bo’ladi (matnli fayllar uchun).

SeekEof(var F: TextFile): boolean – SeekEoln kabi ish bajaradi va barcha fayllar ishlatiladi.

BlockRead(var F: File; var Buf; Count: word[; Result: word]), BlockWrite(var F: File; var Buf; Count: word[; Result: word]) - **Count** miqdordagi bloklarning Buf o’zgaruvchini mos o’qish va yozish prosedurasi.

Fayl - axborotlarning nomlangan strukturasi bo’lib, axborotning barcha elementlari bir tipga tegishli bo’ladi. Undagi elementlar miqdori amaliy jixatdan chegaralanmagan. Odatda fayl ham o’zgaruvchilarni e’lon bo’limida e’lon qilinishi kerak. Faylni e’lon qilishda fayl elementlarining tipi ko’rsatiladi.

E’lon umumiy ko’rinishda quyidagicha yoziladi:

Var faylli o’zgaruvchi: file of tip;

Masalan:

Var f : file of integer;

g : file of string[20];

Bu yerda faqat butun sonlar uchun mo'ljallangan ***f*** hamda har bir elementi 20 tagacha belgidan iborat bo'lgan matnli ma'lumotlarning ***g*** – faylli o'zgaruvchilari e'lon qilinmoqda.

Faylli o'zgaruvchini e'lon qilish faqat fayl komponentining tipini beradi, xolos. Dastur faylga ma'lumotlarni yozishi yoki fayldan o'qishi uchun aniq bir fayl ko'rsatilishi kerak, ya'ni, faylli o'zgaruvchini aniq bir fayl bilan bog'lash kerak (fayl nomini berish). Fayl nomi **AssignFile** proseduraga murojaat qilish bilan beriladi. **AssignFile** proseduraning ko'rinishi quyidagicha bo'ladi:

Bu prosedura umumiy ko'rinishda

AssignFile(faylli o'zgaruvchi,'fayl manzili va nomi');

tarzida yoziladi. Masalan:

assignfile(f, 'D:\DELPHI\alomat. pas');

Bunday e'londan keyin, kompyuter ***f*** fayli deganda D diskda joylashgan DELPHI papkasidagi alomat. pas faylini tushunadi, ya'ni har gal ***f*** fayli ustida bajarilishi lozim bo'lgan amallarni D:\DELPHI\alomat. pas fayli uchun bajaradi.

Fayllar ustida biror amalni bajarish uchun avval uni ochish talab qilinadi. Fayllarni ochish esa uch xil maqsaddan faqat bittasi uchun amalga oshiriladi. Fayllarni ochish masalasi maqsadga ko'ra quyidagi metodlar yordamida hal qilinadi:

Rewrite(f). - ***f*** faylidagi ma'lumotlarni o'qish;

Reset(f) - f faylidagi ma'lumotlarni o'qish uchun ochish;

Append(f) - f faylining oxiriga yangi ma'lumotlar qo'shish uchun ochish. Dastur matnining oxirida yoki zarurat paydo bo'lganda ochilgan fayllarni

albatta yopish talab qilinadi. Buning uchun ***Close(f) - f*** faylini yopish metodidan foydalaniladi.

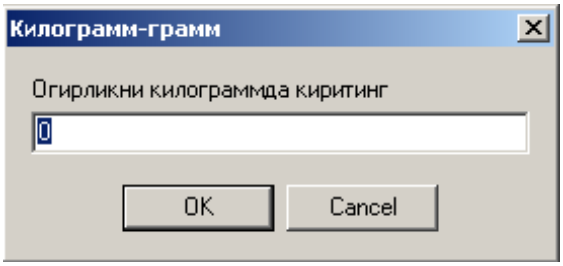
Windows operasion tizimi bir qancha standart muloqot oynalariga ega. Bu oynalar misoliga fayllarni ochish va saqlash, shriftlarni tanlash va to'g'rilash, rang berish,

printerni boshqarishlarni keltirish mumkin. Delphi sistemasi ham bu muloqot oynalarini qo'llaydi.

Kiritish oynasi -standart dialog oynasi bo’lib **InputBox** funksiyasini chaqirish natijasida ekranga chiqariladi. InputBox funksiyasi qiymati - foydalanuvchi kiritgan qatordir.

Umumiy holda **InputBox** funksiyasini chaqirish quyidagi ko’rinishga ega:

O’zgaruvchi := InputBox(Sarlavha, Izoh, Qiymat);



Rasmda dialog oynasining ko’rinishi keltirilgan. Bu kiritish oynasi dasturda quyidagi instruksiya orqali chiqarilishi mumkin:

s:=InputBox('Kilogramm-gramm','Og’irlikni kilogrammda kiriting','0');

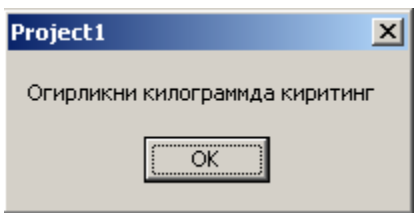
Agar dastur bajarilishi davomida foydalanuvchi qator kiritib **OK**, tugmasini bossa **InputBox** funksiyasi qiymati kiritilgan katorga teng bo’ladi. Agar **Cancel**, tugmasi bosilsa funksiya qiymati, funksiyaga parametr sifatida berilgan satrga teng bo’ladi

Shuni ta’kidlash, lozimki **InputBox** funksiyasi qiymati satr (**string**) turiga tegishli. Shuning uchun dasturga son qaytarish lozim bo’lsa, mos o’zgartirish funksiyasidan foydalanish lozim.

Ekranga ma’lumot oynasini chiqarish uchun **ShowMessage** prosedurasidan yoki **MessageDlg** funksiyasidan foydalanish lozimdir.

ShowMessage prosedurasi ekranga matnli hamda **OK** buyruq tugmasiga ega bo’lgan ma’lumot oynasini chiqaradi.

ShowMessage prosedurasini chaqirish instruksiyasi quyidagi ko’rinishga ega: **ShowMessage(Ma’lumot);**



Rasmda quyida keltirilgan instruksiyani bajarish natijasida ekranda aks etuvchi ma’lumot oynasi ko’rsatilgan:

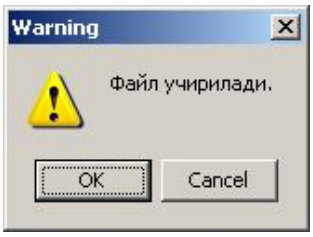
ShowMessage('Og’irlikni kilogrammda kiriting');

etadi.

Ma’lumot oynasining sarlavxasida Project Options oynasining Application bo’limida ko’rsatilgan ilova nomi aks etadi. Agar ilova nomi berilmagan bo’lsa sarlavhada bajarilayotgan fayl nomi aks

MessageDig funksiyasi universal xarakterga egadir. Bu funksiya ma’lumotli oynaga standart belgilardan birini, masalan "Vnimaniye", buyruq tugmalarining sonini va turini berishga, hamda foydalanuvchi qaysi tugmani bosganligini aniqlashga imkon beradi. Rasmda quyidagi instruksiya bajarilish natijasi keltirilgan.

r:=MessageDlg('Fayl uchiriladi. ', mtWarning, [mbOk,mbCancel],0); MessageDlg funksiyasining qiymati qaysi buyruq tugmasi bosilgan-ligini aniklashga imkon beruvchi sondir.



MessageDlg funksiyasiga murojaat umumiy ko’rinishi quyidagichadir:

Tanlov: = MessageDlg (Ma’lumot, Tur, Tugmalar, KontekstSpravkalar)

Bu

yerda:

 Ma’lumot - ma’lumot matni;

 Tur - ma’lumot turi. Ma’lumot informasion, ogohlantiruvchi yoki kritik xato haqidagi ma’lumot bo’lishi mumkin. Har bir ma’lumot turiga ma’lum belgi mos keladi. Ma’lumot turi nomlangan konstanta bilan beriladi.

MessageDlg funksiyasi

konstantalari:

Konstanta	Ma’lumot tipi	Belgi

mtWarning	Diqqat	
mtError	Xato	
mt Information	Ma’lumot	
mtConfirmation	Tasdiqlash	
mtCustom	Oddiy	Belgisiz

Tugmalar — ma’lumot oynasida aks etuvchi tugmalar ro’yxati. Ruyxat nomlangan konstantalardan iborat bo’ladi

MessageDlg funksiyasi konstantalari:

Konstanta	**Tugma**	**Konstanta**	**Tugma**
mbYes	Yes	Mb Abort	Abort
mbNo	No	mbRetry	Retry
mbOK	OK	Mblgnore	Ignore
mbCancel	Cancel	mbAll	All
mbHelp	Help		

Masalan, ma’lumot oynasida OK va Cancel, tugmalari paydo bo’lishi uchun tugmalar ro’yxati quyidagicha berilishi lozim: [mbOK,mbCancel]

Keltirilgan tugmalardan konstantalardan tashqari eng ko’p qo’llanadigan konstantalar: mbOkCancel, mbYesNoCancel va mbAbortRetryIgnore.

- kontekst Spravkalar — foydalanuvchi <F1> tugmasini bosganda ekranda paydo bo’luvchi spravka tizimining bo’limidir. Agar bu parametr qiymati nolga teng bo’lsa spravka ekranga chiqarilmaydi.

Quyidagi jadvalda **MessageDlg** qaytarishi mumkin bo’lgan qiymatlar va ularga mos buyruq tugmalari berilgan.

MessageDlg funksiyasi konstantalari:

MessageDlg konstantalari	**Bosilgan tugma**

mrAbort	Abort
mrYes	Yes
mrOk	Ok
mrRetry	Retry
mrNo	No
mrCancel	Cancel
mrIgnore	Ignore
mrAll	All

Delphi tizimida muloqot oynalarini qo'llash uchun maxsus **Dialogs** nomli komponentalar palitrasi mavjud bo'lib, u o'z ichiga bir necha vizual bo'lmagan komponentalarni oladi. Ulardan **OpenDialog, SaveDialog** va **FontDialog** komponentalarini ko'rib chiqamiz.

OpenDialog komponentasi kompyuter fayl tizimini ko'rish va undan kerakli fayl nomini tanlash imkonini beradi. Bu komponenta piktogrammasi  ko'rinishga ega. U vizual bo'lmagan komponenta bo'lib, uni formaga sichqonchada bir marta bosib qo'yiladi va keyin uning xossalari o'rnatiladi.

Uning asosiy xossalarini ko'rib chiqamiz: DefaultExt -faylning kengaytma nomini saqlaydi. FileName - tanlangan fayl nomini saqlaydi.

Filter -fayl nomlarini muloqot darchasiga ko'rsatilgan kengaytma nom bo'yicha filtrlab chiqaradi. Masalan, agar. pas ko'rsatilgan bo'lsa, muloqot oynasida faqat. pas kengaytmali fayllar chiqadi.

Filter xossasiga o'tilib uch nuqtali tugmacha bosilsa Filter Editor muloqot oynasi chiqadi. U ikki qismdan iborat bo'lib, birinchi qismda filtr matni ikkinchi qismida esa filtrning o'zi beriladi.

Masa

lan:

filtr matni nomlari:

Faylы moduley Delphi (*. pas) Tekstovыye dokumentы (*. txt,*. doc) Vse faylы (*. *)

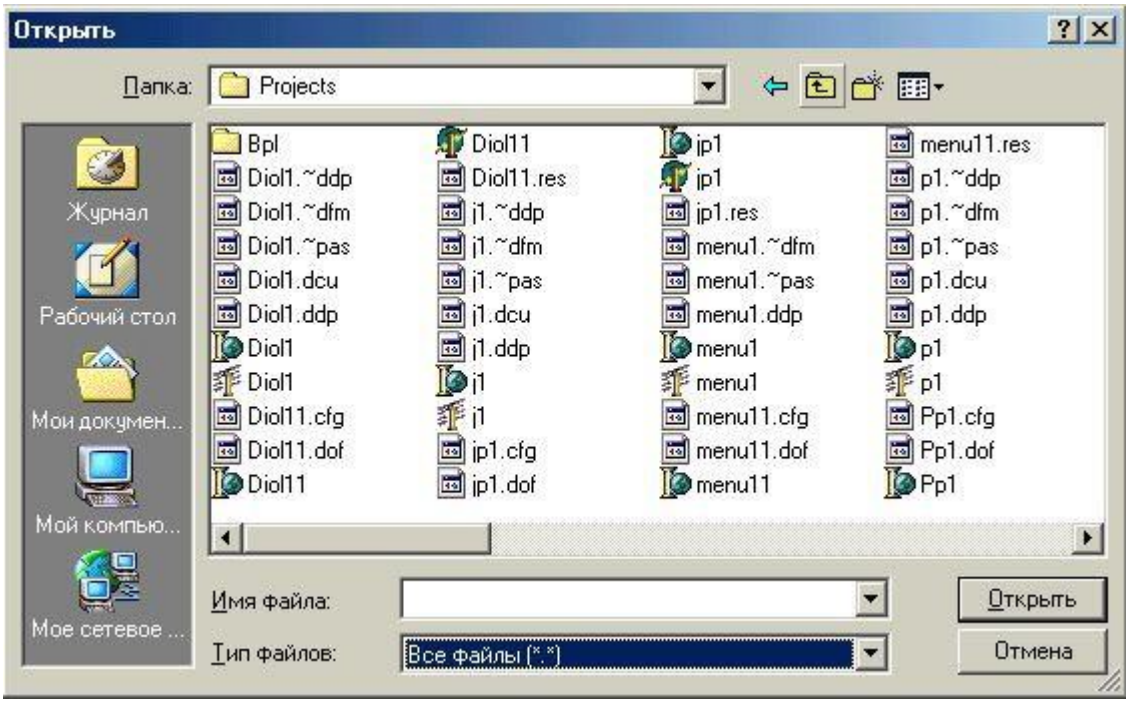
va boshqa mos filtrlar:

*. pas

*. txt; *. doc

*. *

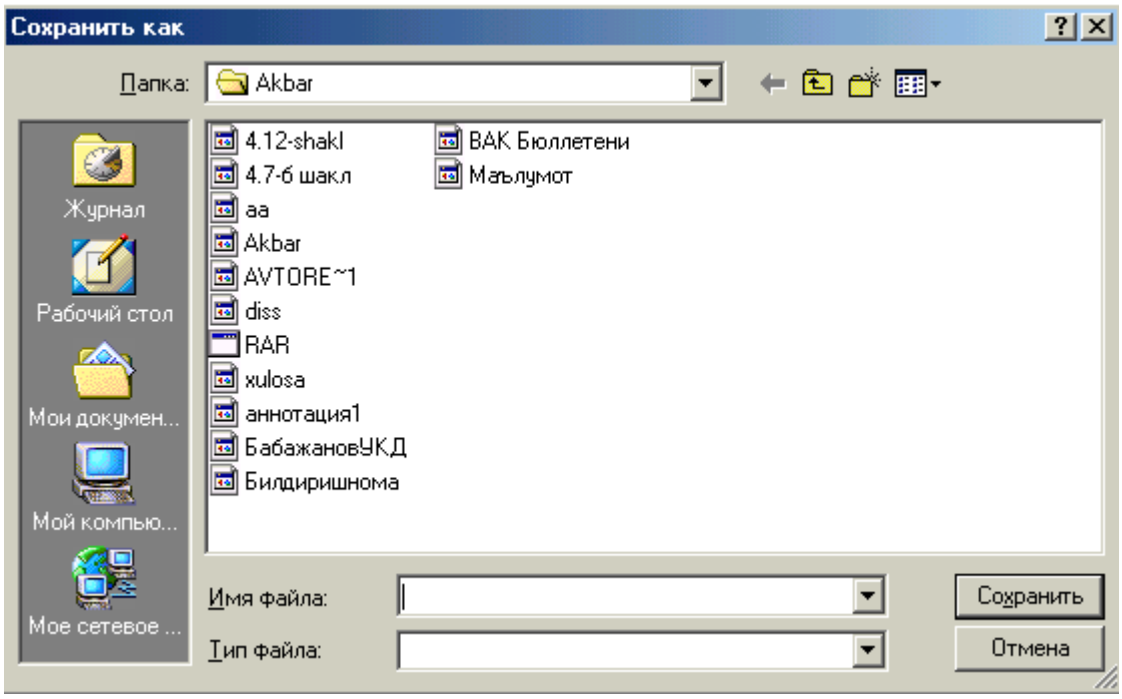
Fayl ochish muloqot oynasining ko’rinishi.



SaveDialog komponentasi kompyuter xotirasiga fayllarni saqlash imkonini beradi. Bu komponenta piktogrammasi  ko’rinishga ega. U vizual bo’lmagan

komponenta bo’lib, uni formaga sichqonchada bir marta bosib qo’yiladi va keyin uning xossalari o’rnatiladi. Agar uning DefaultExt xossasi qiymati. txt qilib tenglashtirilsa, faylni saqlashda avtomatik ravishda uning kengaytmasi. txt qilinib saqlanadi.

Faylni saqlash muloqot oynasining ko’rinishi.

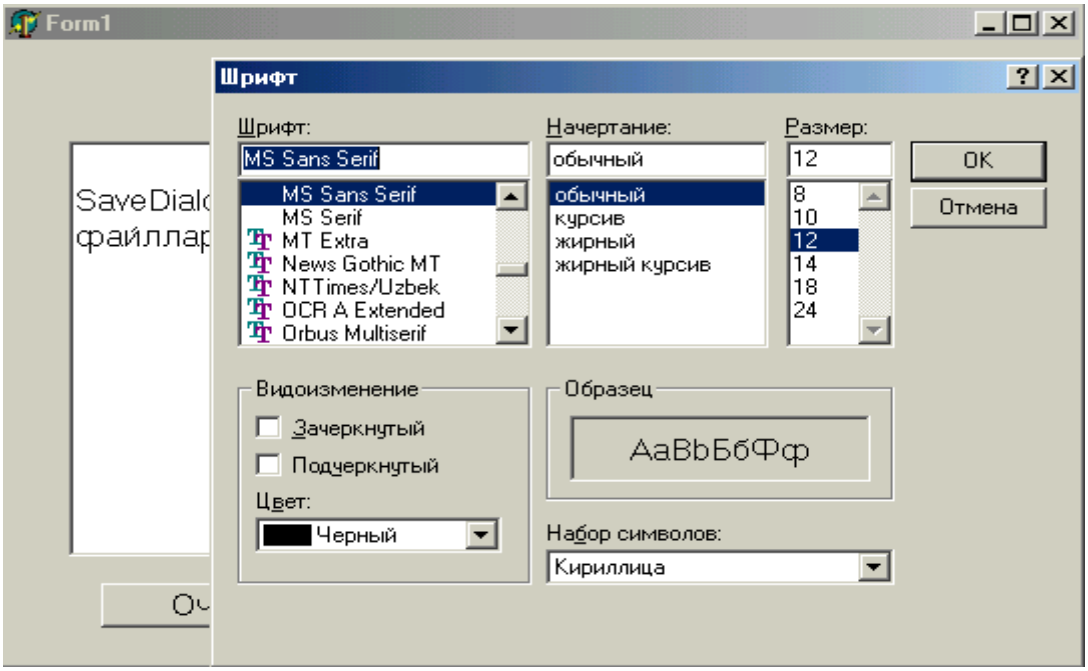


FontDialog komponentasi foydalanuvchiga shriftlarni tanlaydi va uning

xarakteristikasini belgilaydi. Bu komponenta piktogrammasi  ko'rinishga ega. U vizual bo'lmagan komponenta bo'lib, uni formaga sichqonchada bir marta bosib

qo'yiladi va keyin uning xossalari o'rnatiladi. Uning Font xossasi shrift xarakteristikasini beradi.

Shriftni tanlash muloqot oynasining ko'rinishi.



Fayl tipii quyidagi uch yullarning biri bilan yaratiladi:

<nom>=file of<tip>;

<nom>=TextFile;

<nom>=File;

Bu erda <nom>- fayl tipining nomi (tug'ri nomlangan identifikator), file, of xizmatchi suzlar, <tip> bu fayl tipidan boshqa barcha tiplar

Misol:

```
Type
  Product=record
  Name:String;
 Code:Word;
End;
Text80=file of String [80];
Var
  F1: File of Char;    F2: TextFile; F3: File;     F4: Text80; F5: File of Product;
```

Fayllarni e'lon qilish usullariga kura, ularni uch turga ajratish mumkin:

- tiplashtirilgan fayllar(File of … bilan beriladi, yuqorida misolda, F1, F4, F5);
- matnli fayllar(TextFle tipi bilan aniqlanadi, yuqorida misolda, F2);
- tiplashtirilmagan fayllar(File tipi bilan beriladi, yuqorida misolda, F3);

Faylning turi uning saqlanish usulini aniqlaydi va umuman Delphida oldindan yaratilgan faylni nazorat qilish vositalari mavjud emas va bu vazifani dasturchi uz zimmasiga olishi lozim. Fayllar bilan ishlay bilish faqat faylni ochish protsedurasi bajarilgandan sung bajarilishi mumkin. Bu odindan e'lon qilingan fayl ´zgaruvchisini yaratilgan yoki yaratilishi lozim bulgan fayl nomi btlan bog'lash protsedurasi bulib, undan su´ng fayldan ´qsh yoki unga yozish yunalishi beriladi.

AssignFile (var F, FileName: String) protsedurasi fayl uzgaruvchisi F-ni FileName –fayl nomi bilan boglaydi.

AssignFile (<fayl uzgaruvchisi >,< fayl nomi >);- bu protseduraning umumiy kurinishi bulib, bu erda fayl uzgaruvchisi - dasturda e'lon qilingan fayl tipidagi uzgaruvchi, fayl nomi esa, fayl nomini yoki ungacha bulgan yulni ifodalovchi matn.

Fayl initsializatsiyasi deb, bu faylga ma’lumotlarni junatish yoki undan olish yunalishiga aytiladi.

Faylni ukish uchun fayl Reset protsedurasi yordamida initsializatsiya qilinadi va bu protseduraning kurinishi kuyidagicha:

Reset (<fayl uzgaruvchisi >);

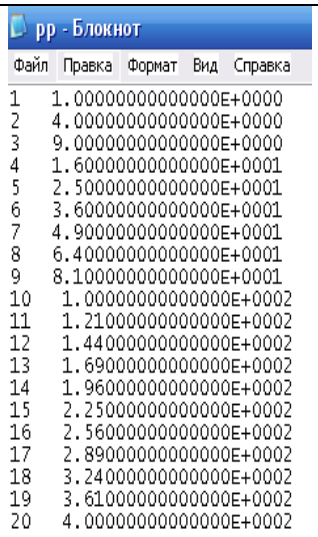
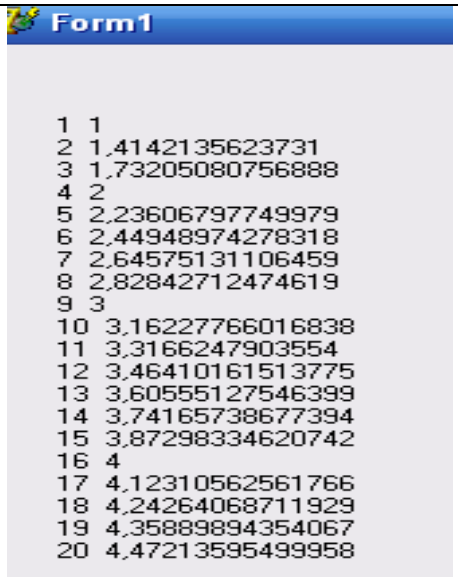
Izox: fayl uzgaruvchisi –avval AssignFile protsedurasi yordamida mavjud fayl bilan bog’langan bulishi lozim.

Bu protsedura bajarilishi natijasida fayl ukish uchun tayyorlanadi va natijada maxsus kursatgich bu faylni boshiga, ya’ni 0-tartib nomerli komponentni kursatib turadi.

Delphi dasturlash tilida Reset protsedurasi yordamida ochilgan tiplashtirilgan fayllarga Write protsedurasi bilan murojaat kilish mumkin. Bu avval yaratilgan tiplashtirilgan fayllarni kengaytirish va yangilash imkoniyatini beradi. Reset protsedurasi yordamida ochilgan matnli fayllar uchun Write yoki Writeln protseduralaridan foydalanib bulmaydi.

Rewrite < fayl uzgaruvchisi > protsedurasi fayl uzgaruvchisi bilan boglangan faylga yozish uchun beriladi.

```
procedure TForm1. Button1Click(Sender: TObject);
const N=1000;
 var
 F,f1:textfile; M:array [1. . N] of real; i:integer;
begin
 AssignFile(F,'prog. txt'); AssignFile(f1,'pp. txt'); reset(F); Append(f1);
 i:=1; label1. Caption:='';
 while not EOF(f) and (i<=N) do
 begin
 Read(F,M[i]); writeln(F1,i,' ',sqr(m[i]));
 label1. Caption:=label1. Caption+inttostr(i)+ ' '+floattostr(sqrt(m[i]))+#13;
 inc(i); end; CloseFile(F); closefile(f1); end; end.
```

pp. txt –faylidagi yozuv	Ekrandagi natija
	

Fayllar bilan ishlash uchun qo’llanaladigan qism dasturlar.

1. function EOF (var F):Boolean;

Fayl kursatgichi faylning oxirida turgan bulsa TRUE, aksincha FALSE kiymatini beradi. Masalan, yukorida keltirilgan protseduradagi

while not EOF(f) and (i<=N) do

va yoki

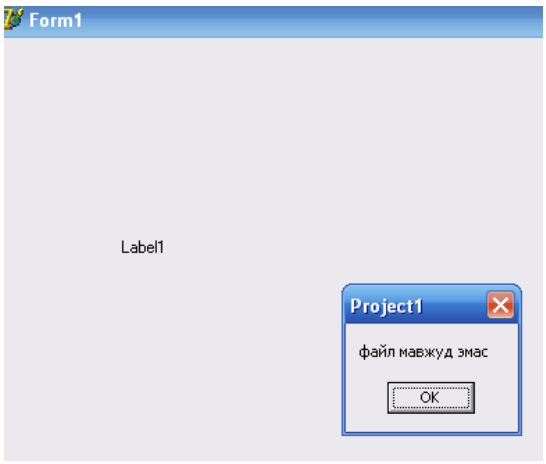
while EOF(f)=false and (i<=N) do buyruglari yordamida f-fayl uzgaruvchisidan ('prog. txt') matnni oxirigacha ukish uchun foydalanish mumkin.

2. function FileExists (const Filename:string):boolean;

FileName dagi fayl (xatto faylgacha bulgan yul) mavjud bulsa, True aksincha False kiymatini beradi.

Masalan, kuyida keltirilgan dasturning kismida ('prog. txt') faylining majudligi tekshiriladi va u mavjud bulmagan xolda” fayl mavjud emas” yozuvi ekranga chikadi.

```
AssignFile(F,'prog. txt');
AssignFile(f1,'pp. txt');
if fileExists('prog. txt')=true then goto 10 else showMessage(' fayl mavjud emas');
10:reset(F);
```



3. function FindFirst(const Path: String; Attr:integer; var F: TSearchRec):integer;

Katalogda mavjud bulgan birinchi faylning atributlarini beradi: Path-kidiruv yunalishi va fayllarni tanlash shabloni, Attr-fayl atributlari, F- TSearchRec tipidagi uzgaruvchi, fayl mavjud xolda 0 kiymatni beradi.

4. function findClose(var F: TSearchRec);

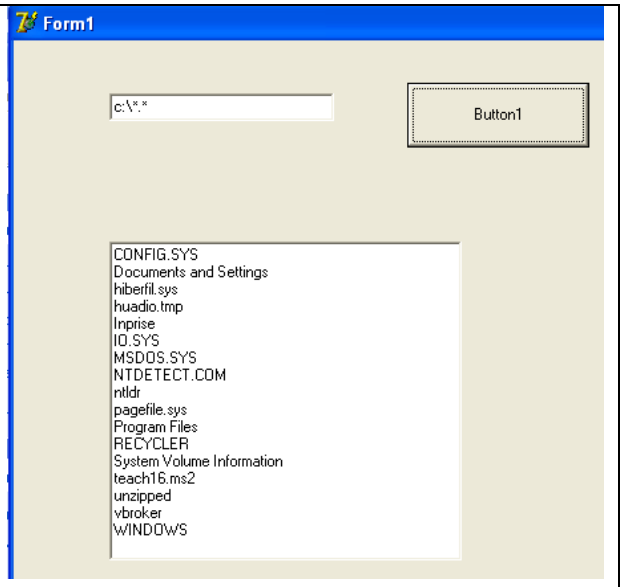
FindFirst va FindNext funktsiyalari orkali fayllarni kidirish uchun ajratilgan xotiraning band kismini ozod kiladi.

5. function FindNext(var F: TSearchRec):integer;

Katalogdan kidirilayotgan navbatdagi faylning nomini F- uzgaruvchiga uzatadi.

YUkoridagi funktsiyalar uchun kuyidagi dastur va dasturning natijasini keltiramiz.

```
var mask:string;
sr:TsearchRec;
begin
 mask:=edit1. Text;
 if mask=" hen mask:='*. *';
 memo1. LineS. Clear;
 If
FindFirst(mask,FaAnyFile,sr)=0
then
 repeat
 memo1. LineS. Add(sr. Name);
```



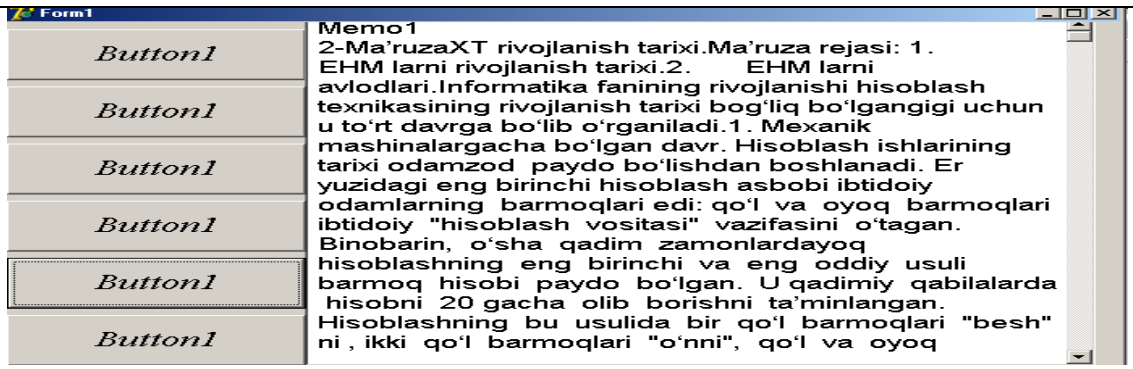
```
until FindNext(sr)<>0;
findclose(sr);
end;
end.
```

Kuyidagi dastur bajarilgach, Memo maydonida ko’rsatilgan faylni o’qish uchun «tavsiya» etadi va bunda o’z aksini topgan matnni o’zgartirish imkoniyatiga ega emas.

Dastur

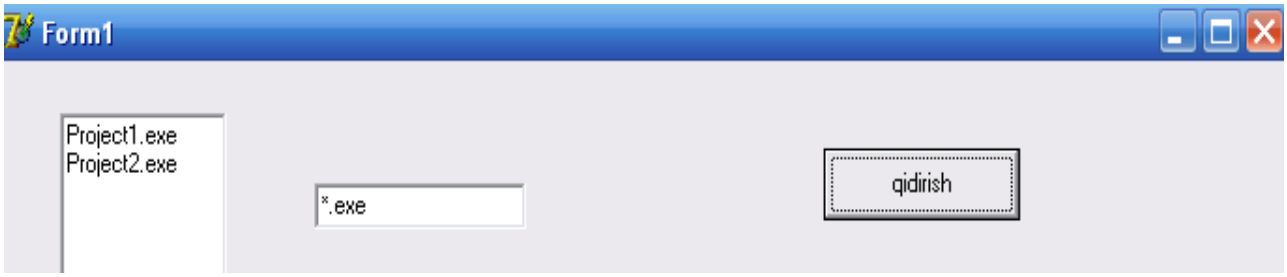
```
procedure TForm1. Button4Click(Sender: TObject);
var f1:textfile; ff:string; h:integer;
begin
AssignFile(f1,'ww. txt'); reset(F1);
form1. Memo1. ReadOnly:=true;
while not eof(f1) do
begin Readln(f1,ff); Memo1. Text:=ff; end;
CloseFile(f1);
End;end.
```

Natija




```
procedure TForm1. bbrunClick(Sender: TObject);
var mask:string;
Sr:TSearchRec;
begin
mask:=edit1. Text;
```

```
 if mask='' then mask:='*. *'; memo1. Lines. Clear;
 if FindFirst(mask,faANYFile,sr)=0 then
 repeat memo1. Lines. Add(sr. Name); until findNext(sr)<>0;
 findClose(sr);end;end.
procedure TForm1. bbrunClick(Sender: TObject);
var mask:string;
Sr:TSearchRec;
begin
 mask:=edit1. Text;
 if mask='' then mask:='c:\*. *';
 memo1. Lines. Clear;
 if FindFirst(mask,faANYFile,sr)=0 then
 repeat
 memo1. Lines. Add(sr. Name);
 until findNext(sr)<>0;
 findClose(sr);yo
end;
```

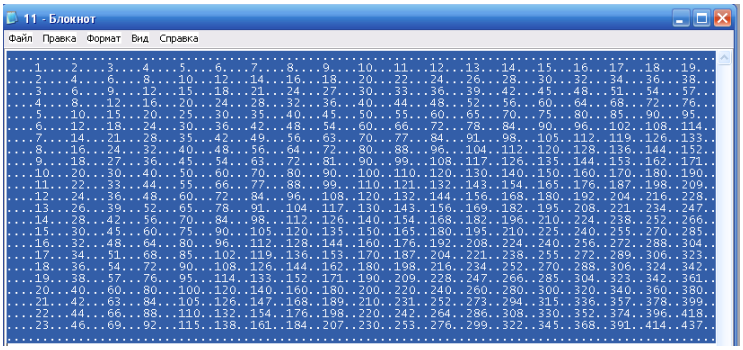



```
procedure TForm1. FormCreate(Sender: TObject);
var i,j: byte; y: word; a:string;
 begin
 Stringgrid1. GridLineWidth:=2;
 form1. StringGrid1. MouseCoord(i,j);
 stringgrid1. DefaultColWidth:=40;
 stringgrid1. defaultrowheight:=20;
 stringgrid1. Colcount:=25;
 stringgrid1. rowCount:=40;
```

```
 for i:=1 to 23 do
 for j:=1 to 35 do
 stringgrid1. Cells[i,j]:=inttostr(i*j);
 a:=stringgrid1. Cells[stringgrid1. Col,stringgrid1. Row];
 label1. Caption:=(a); label2. Caption:='';
end;
procedure TForm1. Image2Click(Sender: TObject);
 var a:string; B:REAL;
begin
a:=stringgrid1. Cells[stringgrid1. Col,stringgrid1. Row];
label1. Caption:= label1. Caption+inttostr(stringgrid1. Col)+' * '+inttostr(stringgrid1.
row)+' = '+a+#13;
label2. Caption:= label2. Caption+inttostr(stringgrid1. row)+' * '+inttostr(stringgrid1.
col)+' = '+a+#13;
end;
procedure TForm1. Image4Click(Sender: TObject);
begin
//Showmessage('Jadvaldan biror sonni tanlang va Kuzatish tugmasini bosing!');
 form2. show;
end;
procedure TForm1. Image3Click(Sender: TObject);
var a:string; B:REAL;
begin
a:=stringgrid1. Cells[stringgrid1. Col,stringgrid1. Row];
label1. Caption:= label1. Caption+inttostr(stringgrid1. Col)+' * '+inttostr(stringgrid1.
row)+' = '+a+#13;
label2. Caption:= label2. Caption+inttostr(stringgrid1. row)+' * '+inttostr(stringgrid1.
col)+' = '+a+#13;
end;
procedure TForm1. Image5Click(Sender: TObject);
```

```
begin
form3. show;
end;
procedure TForm1. Button1Click(Sender: TObject);
var f:textfile;
x,y:integer;
begin
assignfile (f,'11. txt');
rewrite (f);
write (f,stringgrid1. colcount);
write (f,stringgrid1. rowcount);
For X:=0 to stringgrid1. colcount-1 do
begin
For y:=0 to stringgrid1. rowcount-1 do
write (F, stringgrid1. cells[x,y]);
 writeln (F, '');
 end;
closefile (f);
end;
procedure TForm1. Image6Click(Sender: TObject);
var f:textfile;
x,y:integer;
probel,probel1,probel2,probel3:string;
begin
probel1:='. . . . ';
probel2:='. . . ';
probel3:='. . ' ;
assignfile (f,'11. txt');
rewrite (f);
For X:=0 to stringgrid1. colcount-1 do
```

```
begin
For y:=0 to stringgrid1. rowcount-1 do
begin
if length(stringgrid1. cells[x,y])<2 then probel:=probel1 else
if (length(stringgrid1. cells[x,y])>=2 )and (length(stringgrid1. cells[x,y])<3) then
probel:=probel2 else probel:=probel3 ;
write (F,stringgrid1. cells[x,y],probel);
 end; writeln (F, ''); end;
closefile (f);
append(f);
end; end.
```



Fayl joyni ko’rsatish

```
procedure TForm1. Button1Click(Sender: TObject);
const
 N=1000;
 var
 F:textfile;
 M:array [1. . N] of real;
 i:integer; d:byte;
 s:string;
begin
 AssignFile(F,'prog. txt');
 reset(F);
```

```
i:=1;
while not EOF(f) and (i<=N) do
begin
Read(F,M[i]);
inc(i)
end;
CloseFile(F);
label1. Caption:=floattostr(m[1]);
// MkDir('kk');
getdir(d,s);
label2. Caption:=s;
end;
end.
```

Mavzuga oid savollar

1. Fizik fayl deb qanday fayllarga aytiladi?
2. Mantiqiy fayl deb qanday fayllarga aytiladi?
3. Fizik va mantiqiy fayllar qanday bog’laniladi?
4. Fayllar nima uchun kerak va qanday ochiladi?
5. Fayl qanday yopiladi?
6. Fayl ko’rsatkichi deb nimaga aytiladi?
7. Fayllar necha xil xolatda bo’lishi mumkin?
8. Muloqot oynalarini yaratish qanday amalga oshiriladi?
9. OpenDialog, SaveDialog va FontDialog komponentalari qanday funksiyalarni bajaradi va ularning qanday xossalarini bilasiz?

5-BOB. C++ DASTURLASH TILIGA KIRISH

7.1. C++ TILINING LEKSIK ASOSLARI.

Reja:

1. C++ dasturlash tiliga kirish. C++ dasturlash tili alifbosi va xizmatchi so’zlari.
2. Amallar. Izohlar satrini tavsiflash.
3. C++ tilida operatorlar.
4. Standart funksiyalar va ularning yozilishi.
5. Konsol orqali muloqot qilish.
6. Chiqarish operatori. Kiritish operatori

Tayanch iboralar: kommunikatsiya, dasturiy ta’minot, tashxis, teskari aloqa, loyihalash, foydalanuvchi interfeysi, foydalanuvchi, aniqlik, Stereotip, buyurtmachi, dasturchi, samaradorlik.

Amallar. Izohlar satrini tavsiflash. C++ tilida operatorlar. Standart funksiyalar va ularning yozilishi. Konsol orqali muloqot qilish. Chiqarish operatori. Kiritish operatori

C++ dasturlash tiliga kirish. C++ dasturlash tili alifbosi va xizmatchi so’zlari.

C++ tili Byarn Straustrup tomonidan 1980 yil boshlarida ishlab chiqilgan. C++ tilida yaxshi dastur tuzish uchun “aql, farosat va sabr” kerak bo’ladi. Bu til asosan tizim sathida dasturlovchilar uchun yaratilgan.

C++ algoritmik tilining alifbosi:

1. 26 ta lotin va 32 ta kirill harflari (katta va kichik);
2. 0 dan 9 gacha bo’lgan arab raqamlari;
3. Maxsus belgilar: - + * / : ;. , % ? ! = “” № < > { } [ ] ( ) $ # & ^ va h. k.

Amallar. Izohlar satrini tavsiflash.

Berilganlarni qayta ishlash uchun C++ tilida amallarning juda keng majmuasi aniqlangan. Amal - bu qandaydir harakat bo‘lib, u bitta (unar) yoki ikkita (binar) operandlar ustida bajariladi, hisob natijasi uning qaytaruvchi qiymati hisoblanadi.

Tayanch arifmetik amallarga qo‘shish (+), ayirish (-), ko‘paytirish (*), bo‘lish (/) va bo‘lish qoldig‘ini olish (%) amallarini keltirish mumkin.

Amallar qaytaradigan qiymatlarni o‘zlashtirish uchun qiymat berish amali (=) va uning turli modifikatsiyalari ishlatiladi: qo‘shish, qiymat berish bilan (+=); ayirish, qiymat berish bilan (-=); ko‘paytirish, qiymat berish bilan (*=); bo‘lish, qiymat berish

bilan (/=); bo‘lish qoldig‘ini olish, qiymat berish bilan (%=) va boshqalar. Bu holatlarning umumiy ko‘rinishi:

Dastur bajarilishi natijasida ekranga quyidagi sonlar satri paydo bo‘ladi:

4 188 2 12 9 1 482 2

Ifoda tushunchasi: C++ tilida ifoda - amallar, operandlar va punktatsiya belgilarining ketma-ketligi bo‘lib, kompilyator tomonidan berilganlar

ustida ma’lum bir amallarni bajarishga ko‘rsatma hisoblanadi. Har qanday ';' belgi bilan tugaydigan ifodaga til ko‘rsatmasi deyiladi.

C++ tilidagi ifodaga (til ko‘rsatmasiga) misol:

x=3*(y-2. 45);

u=summa(a, 9,c) ;

Inkrement va dekrement amallari: C++ tilida operand qiymatini birga oshirish va kamaytirishning samarali vositalari mavjud. Bular inkrement (++) va dekrement) unar amallardir.

Operandga nisbatan bu amallarning prefiks va postfiks ko‘ri|nishlari bo‘ladi. Prefiks ko‘rinishda amal til ko‘rsatmasi bo‘yicha ish bajarilishidan oldin operandga qo‘llaniladi. Postfiks holatda esa amal til ko‘rsatmasi bo‘yicha ish bajarilgandan keyin operandga qo‘llaniladi.

Prefiks yoki postfiks amal tushunchasi faqat qiymat berish bilan bog’liq ifodalarda o‘rinli: ;

x=u++; // postfiks

index =--i; // prefiks

count:++; // unar amal, "++count; " bilan ekvivalent

abc-- ; // unar amal, "--abc; " bilan ekvivalent

Bu yerda u o‘zgaruvchining qiymatini x o‘zgaruvchisiga o‘zlashtiriladi va keyin bittaga oshiriladi, i o‘zgaruvchining qiymati bittaga kamaytirib, index o‘zgaruvchisiga o‘zlashtiriladi.

sizeof amali: Har xil turdagi o‘zgaruvchilar kompyuter xotirasida turli sondagi baytlarni egallaydi. Bunda, hattoki bir turdagi o‘zgaruvchilar ham kaysi kompyuterda

yoki qaysi operatsion sistemada amal qilinishiga qarab turli o‘lchamdagi xotirani band qilishi mumkin.

C++ tilida ixtiyoriy (tayanch va hosilaviy) turdagi o‘zgaruvchilarning o‘lchamini sizeof amali yordamida aniqlanadi. Bu amalni o’zgarmasga, turga va o‘zgaruvchiga qo‘llanishi mumkin.

C++ tilida operatorlar.

Standart funksiyalar va ularning yozilishi.

Konsol orqali muloqot qilish.

Chiqarish operatori. Kiritish operatori

Dastur bajarilishi jarayonida o’z qiymatini o’zgartira oladigan kattaliklar o’zgaruvchilar deyiladi. O’zgaruvchilarning nomlari harfdan boshlanuvchi xarf va raqamlardan iborat bo’lishi mumkin. O’zguruvchilarni belgilashda katta va kichik harflarning farqlari bor. (A va a harflari 2 ta o’zgaruvchini bildiradi) Har bir o’zgaruvchi o’z nomiga, toifasiga, xotiradan egallagan joyiga va son qiymatiga ega bo’lishi kerak. O’zgaruvchiga murojaat qilish uning ismi orqali bo’ladi. O’zgaruvchi uchun xotiradan ajratilgan joyning tartib raqami uning adresi hisoblanadi. O’zgaruvchi ishlatilishidan oldin u aniqlangan bo’lishi lozim.

O’zgaruvchilarning son qiymatlari quyidagi ko’rinishda yoziladi:

- Butun toifali o’nlik sanoq tizimsida: ular faqat butun sondan iborat bo’ladilar. Masalan: 5; 76; -674 va h. k.
- Sakkizlik sanoq tizimsidagi sonlar: 0 (nol) dan boshlanib, 0 dan 7 gacha bo’lgan raqamlardan tashkil topadi. Masalan: *x*=0453217; *s*=077;
- O’n oltilik sanoq tizimsidagi sonlar: 0 (nol) dan boshlanadi va undan keyin x yoki X harfi keladi, so’ngra 0-9 raqamlari va a-f yoki A-F harflaridan iborat ketma-ketliklar bo’ladi. Masalan: 10 s. s. dagi 22 soni 8 s. s. da 026, 16 s. s. da 0x16 shaklida bo’ladi.
- Haqiqiy toifali sonlar: ular butun va kasr qismlardan iborat bo’ladilar. Masalan: 8,1; -12,59 va x. k. Haqiqiy toifali sonlarning bu ko’rinishi oddiy ko’rinish deyiladi. Juda katta yoki juda kichik haqiqiy toifali sonlarni darajali

(eksponensional) formada yozish qulay. Masalan: $7{,}204*10^{12}$ yoki $3{,}567*10^{-11}$ kabi sonlar 7. 204e+12 va 3. 567e-11 ko'rinishda yoziladi.

- Simvolli konstantalar. Ular qatoriga dastur bajarilishi ' ' ichida qabul qilinadigan simvollar kiradi.

C++ tilida har qanday o'zgaruvchi ishlatilishidan oldin e'lon qilinishi kerak. E'lon qilish degani ularning toifalarini aniqlab qoyish demakdir.

C++ tilida quyidagi toifali o'zgaruvchilar ishlatiladi:

- Butun toifali kichik sonlar yoki simvollar uchun: char uning o'zgarish intervali -128 dan +127 gacha yoki apostrof ichidagi ixtiyoriy 1ta simvol. Xotiradan 1 bayt joy oladi. Simvollar ASCII kodlariga mos keladi. (ASCII – American Standart Code for Information Interchange)
- Butun toifali o'zgaruvchilar: int. Masalan: int a, i, j ; Bu yerda dasturda ishlatilayotgan a, i, j o'zgaruvchilarining toifasi butun ekanligi ko'rsatildi. Bu toifadagi o'zgaruvchilar 2 bayt joy egallaydi. Ularning o'zgarish intervali: -32768 dan +32767 gacha; (Hozirgi 32 razryadli kompyuterlarda 4 bayt joy oladi va oralig'i 2 marta oshgan).
- Butun toifali katta (uzun) o'zgaruvchilar: long. Masalan: long s, s2, aa34; Bu toifadagi o'zgaruvchilar 4 bayt joy egallaydi. Ular –2147483648 dan +2147483647 oraliqdagi sonlarni qabul qilishi mumkin.
- Ishorasiz butun o'zgaruvchilar: unsigned short – 2 bayt joy oladi, o'zgarish intervali 0 dan 65535 gacha; unsigned long – 4 bayt joy oladi, o'zgarish intervali: 0 dan 4294967295 gacha; unsigned char – 1 bayt joy oladi, o'zgarish chegarasi 0 dan 255 gacha.
- Haqiqiy toifadagi o'zgaruvchilar: float. Masalan: float a, b: Bu yerda dasturda ishlatilayotgan a, b o'zgaruvchilarining toifasi haqiqiy ekanligi ko'rsatilgan. Bu toifadagi o'zgaruvchilar 4 bayt joy egallaydi va qabul qilish chegarasi 10^{-38} dan 10^{+38} gacha.
- Katta yoki kichik qiymatli o'zgaruvchilarni ifoda etishda double toifasi ishlatiladi. Ular uchun 8 bayt joy ajratiladi va qabul qilish chegarasi 10^{-304} dan 10^{+304} gacha.

- Juda katta yoki juda kichik qiymatli o’zgaruvchilar uchun long double toifasi ishlatiladi, u 10 bayt joy oladi va qabul qilish chegarasi 3. $4*10^{-4932}$ dan 1. $1*10^{-4932}$ gacha.
- Qator toifasidagi o’zgaruvchilar uchun ham char toifasi belgilangan. Ular ham 1 bayt joy oladi va 0 dan 256 tagacha bo’lgan simvollar ketma-ketligidan iborat bo’lishi mumkin. Satr toifasidagi o’zgaruvchilar qo’shtirnoq (“) ichida yoziladi.

C++ tilida o’zgaruvchilarni inisializasiya qilish degan tushuncha ham mavjud. Inisializasiya qilish degani o’zgaruvchini e’lon qilish barobarida unga boshlang’ich qiymatini ham berish demakdir. Masalan: int a=5, b, s=-100; - a, b, s o’zgaruvchilari butun toifali ekanligi ko’rsatildi va a o’zgaruvchisiga 5 (a=5), s o’zgaruvchisiga esa –100 (s=-100) boshlang’ich qiymatlar berildi.

Dastur bajarilishi jarayonida o’z qiymatini o’zgartira olmaydigan kattaliklar o’zgarmaslar deyiladi. Masalan: x=1; bo’lsa keyinchalik x=x+5 deb yozib bo’lmaydi. O’zgarmaslarni const so’zi bilan ko’rsatiladi. Maslan: const int x=95; float y=9. 17; (const lar simvol yoki nol (NULL) bo’lishi xam mumkin.)

C++ tilida standart funksiyalarning yozilishi

Funksiya	Ifodalanishi	Funksiya	Ifodalanishi
Sin x	sin(x)	$\sqrt{x}$	sqrt(x); pow(x,1/2.)
Cos x	cos(x)	$\lvert x\rvert$	abs(x) yoki fabs(x)
Tg x	tan(x)	Arctan x	atan(x)
e^x	exp(x)	Arcsin x	asin(x) ?
Ln x	log(x)	Arccos x	acos(x) ?
Lg x	log10(x)	$\sqrt[3]{x^2}$	pow(x,2/3.)
x^a	pow(x,a)	Log_2x	log(x)/log(2)

Masalan: $\frac{-b+\sqrt{b^2-4ac}}{2a}$ → (-b + sqrt(b*b-4*a*c)/(2*a); yoki

(-b+pow(b*b-4*a*c,1/2.)/(2*a);

$e^{\sin x} + tg^2(x+3) \rightarrow$ exp(sin(x)) + pow(tan(x+3),2);

k=(m*5)+((7 % n) / (9+x));

C++ tilidagi dastur quyidagi tarkibdan tashkil topadi:

1. Direktivalar – # include <file. h> direktiva – instruksiya degan ma'noni beradi. C++ tilida dasturning tuzilishiga, ya'ni ehtiyojiga qarab, kerakli direktivalar ishlatiladi. Ular < > belgisi orasida keltiriladi. Umuman olganda quyidagi direktivalar mavjud (jami 32 ta):
   - #include <stdio. h> - S da oddiy kiritish/chiqarish dasturi uchun. Bu yerda std - standart, i – input, o - output degani.
   - #include <iostream. h> - C++ da kiritish/chiqarish uchun, oddiy amallar bajarilsa.
   - #include <math. h> - standart funksiyalarni ishlatish uchun.
   - #include <conio. h> - dasturning tashqi ko'rinishini shakllantirish uchun.
   - #include <string. h> - satr toifasidagi o'zgaruvchilar ustida amallar bajarish uchun.
   - #include <stdlib. h> - standart kutubxona fayllarini chaqirish uchun.
   - #include <time. h> - kompyuter ichidagi soat qiymatlaridan foydalanish uchun.
   - #include <graphics. h> - C++ tilining grafik imkoniyatlaridan foydalanish uchun.

Bu fayllar maxsus kutubxona e'lon fayllari hisoblanadilar va ular aloxida INCLUDE deb nomlanadigan papkada saqlanadiar. Hozirda C++ kutubxonasini yangilandi va undagi fayllarning nomlaridan. h (head – bosh ma'nosida) kengaytmasi olib tashlandi va oldiga c harfi qo'shildi (C dan qolgan 18 tasiga). Bu fayllarda funksiya prototoifalari, toifalari, o'zgaruvchilar, o'zgarmaslar ta'riflari yozilgan bo'ladi.

Direktivalar dasturni uni kompilyasiya qilinishidan oldin tekshirib chiqadi.

2. Makroslar - # define makro qiymati. Masalan:

#define y sin(x+25) – u = sin(x+25) qiymati berildi;

#define pi 3. 1415 - pi = 3. 1415

#define s(x) x*x - s(x) = x*x (; belgisi qoyilmaydi)

Global o’zgaruvchilarni e’lon qilish. Asosiy funksiya ichida e’lon qilingan o’zgaruvchilar lokal, funksiyadan tashqarida e’lon qilinganlari esa global o’zgaruvchilar deyiladi. Global o’zgaruvchilar dastur davomida ishlaydi va xotiradan ma’lum joyni egallaydi. O’zgaruvchini bevosita ishlatishdan oldin e’lon qilsa ham bo’ladi, u holda o’z lokal bo’ladi. Global o’zgaruvchilar nomi lokal o’zgaruvchilar nomi bilan bir xil bo’lishi ham mumkin. Bunday holatda lokal o’zgaruvchining qiymati joriy funksiya ichidagini qiymatini o’zgartiradi, funksiyadan chiqishi bilan global o’zgaruvchilar ishlaydi.

Asosiy funksiya - main ( ) hisoblanadi. Bu funksiya dasturda bo’lishi shart. Umuman olganda C++ dagi dastur funksiyalardan iborat deb qaraladi. main ( ) funksiyasi { boshlanadi va dastur oxirida berkitilishi shart }. main – asosiy degan ma’noni beradi. Bu funksiya oldida uning toifasi ko’rsatiladi. Agar main ( ) funksiyasi beradigan (qaytaradigan) javob oddiy so’z yoki gaplardan iborat bo’lsa, hech qanday natija qaytarmasa, void so’zi keltiriladi. main ( ) funksiyasi dastur tomonidan emas, balki OS tomonidan chaqiriladi. OSga qiymat qaytarish shart emas, chunki u bu qiymatdan foydalanmaydi. Shuning uchun main ( ) funksiyasining turini *void* deb ko’rsatganimiz ma’qul. Har bir funksiyaning o’z argumenti bo’ladi, shuning uchun main funksiya ( ) lari ichiga uning parametri keltiriladi. Ba’zan u bo’sh bo’lishi ham mumkin. Bu funksiyadan chiqish uchun odatda *return* operatori ishlatiladi. 0 (nol) qiymatining qaytarilishi operasion tizimga ushbu dastur normal bajarilib turganini bildiradi. return orqali qaytadigan qiymat toifasi funksiya e’lonidagi qaytish toifasi bilan bir xil bo’lishi kerak. Masalan int main ( ) va 0 (nol) qiymat butun toifalidir. Bu funksiyadan so’ng lokal o’zgaruvchilar, qism dasturlar, ularning haqiqiy parametrlar e’lon qilinadi. So’ngra dasturning asosiy operatorlari (kiritish/chiqarish, hisoblash va h. k.) yoziladi. Agar bu operatorlar murakkab toifali bo’lsalar, ularni alohida {} qavslarga olinadi. C++ tilida dastur kichik harflarda yoziladi. Ba’zi operatorlar katta harflar bilan kelishi mumkin, bunday xollarda ular alohida aytib o’tiladi. Operatorlar oxiriga ; belgisi qoyiladi. Operatorlar bir qatorga ketma-ket yozilishi mumkin.

Dasturda izohlar xam kelishi mumkin, ular /*. . . . */ belgisi orasiga olinadi. Agar izoh bir qatorda tugasa, uni // belgisidan keyin yoziladi. Masalan:

main ( ) // C++ tilining asosiy funksiyasi

Tilda quyidagi amallardan foydalanish mumkin:

Arifmetik amallar: +, -, /, *, %. Barcha amallar odatdagidek bajariladi, faqat bo'lish amali butunga bo'lish bajariladi, ya'ni agar butun sonlar ustida bajarilayotgan bo'lsa, natija doim butun bo'ladi, ya'ni kasr qism tashlab yuboriladi (9/5=1; vaxolanki 1,8 bo'lishi kerak). Shuning uchun surat yoki maxrajiga nuqta (.) qoyilsa, natija ham xaqiqiy bo'ladi (9. /5=1. 8). % belgisi (modul operatori) esa butun sonni butun songa bo'lgandan hosil bo'ladigan qoldiqni bildiradi.

Masalan: 9 % 5=4

Taqqoslash amallari: = = (tengmi?); != (teng emas); < ; > ; >=; <=

Mantiqiy amallar: && (and) mantiqiy ko'paytirish; || (or) mantiqiy qo'shish; ! (not) mantiqiy inkor. Mantiqiy amallarni ixtiyoriy sonlar ustida bajarish mumkin. Agar javob rost bo'lsa, natija 1 bo'ladi, agar javob yolg'on bo'lsa, natija 0 bo'ladi. Umuman olganda 0 (nol) dan farqli javob rost deb qabul qilinadi.

Masalan: i>50 && j==24 yoki s1 < s2 && (s3>50 || s4<=20);

Yoki 6 ≤ x ≤ 10 yozuvini x>=6 && x<=10 deb yoziladi

Qiymat berish amallari:

a=5; b = 2*c; x = y = z =1; a = (b = c)*d // 3=5 deb yozib bo'lmaydi

qabul qildim va almashtirdim deb nomalandigan amallar:

+ = : a+=b → a = a + b;

- = : a-=b → a = a - b;

* = : a*=b → a = a * b;

/ = : a/=b → a = a / b;

% = : a%=b → a = a % b;

- inkrement operatsiyasi (++) ikki ma'noda ishlatiladi: o'zgaruvchiga murojaat qilinganidan keyin uning qiymati 1 ga oshadi (a++ postfiks ko'rinishi) va o'zgaruvchining qiymati uning murojaat qilishdan oldin 1 ga oshadi (++a prefix ko'rinishi);

- dekrement operatsiyasi (--), xuddi inkrement operatsiyasi kabi, faqat kamaytirish uchun ishlatiladi. Masalan: s = a + b++ (a ga b ni qo'shib keyin b ning qiymatini 1 ga oshiradi); s = a+(--b) (b ning qiymatini 1 ga kamaytirib, keyin a ga qo'shadi).

Yuqoridagi standart funksiyalardan tashqari yana quyidagi funksiyalar ham ishlatiladi:

- ceil (x) - x ni x dan katta yoki unga teng bo'lgan eng kichik butun songacha yaxlitlash. Masalan: ceil (12. 6) = 13. 0; ceil (-2. 4) = -2. 0;
- floor (x) - x ni x dan kichik bo'lgan eng katta butun songacha yaxlitlash. Masalan: floor (4. 8) = 4. 0; floor (-15. 9) = -16. 0; floor(12. 1) = 12; floor(-12. 1)=-13;
- fmod (x,y) – x / y ning qoldig'ini kasr son ko'rinishida berish. Masalan: fmod(7. 3, 1. 7) = 0. 5;

7.2. O'ZGARUVCHI VA O'ZGARMAS TIPLI KATTALIKLAR.

Reja:

1. O'zgaruvchilarni ta'riflash.
2. O'zgaruvchilar turlari
3. O'zgarmaslar va ularning turlari.
4. Mantiqiy va satrli konstanta
5. Nomlangan va to'plam o'zgarmaslari

Tayanch iboralar: O'zgaruvchilar. Identifikatorlar. Ma'lumotlar tipi. O'zgaruvchilarni e'lon qilish. O'zgaruvchilarni initsializatsiya qilish. Belgili o'zgarmaslar. Maxsus belgilar. Ma'lumotlarning butun son turi. Ma'lumotlarning xaqiqiy son turi. Mantiqiy konstanta. Satrli konstanta. Nomlangan o'zgarmaslar. To'plam o'zgarmaslari

O'zgaruvchilarni ta'riflash. C++ tilida o'zgaruvchini aniqlash uchun kompyuterga uning tipi (masalan, int, char yoki float) hamda ismi xaqida haqida ma'lumot beriladi. Bu axborot asosida kompilyatorga o'zgaruvchi uchun qancha joy ajratish lozim va bu o'zgaruvchida qanday turdagi qiymat saqlanishi mumkinligi haqida ma'lumot aniq bo'ladi. O'zgaruvchi nomi identifikator bo'lib, xizmatchi so'zlardan farqli bo'lishi kerak.

Har bir yacheyka bir bayt o‘lchovga ega. Agar o‘zgaruvchi uchun ko‘rsatilgan tip 4 baytni talab qilsa, uning uchun to‘rtta yacheyka ajratiladi. Aynan o‘zgaruvchini tipiga muvofiq ravishda kompilyator bu o‘zgaruvchi uchun qancha joy ajratish kerakligini aniqlaydi.

Kompyuterda qiymatlarni ifodalash uchun bitlar va baytlar qo‘llaniladi va xotira baytlarda hisoblanadi.

O‘zgaruvchilar

O‘zgaruvchilar turlari. Dastur o‘zi ishlatadigan ma’lumotlarni saqlash imkoniyatiga ega bo‘lishi lozim. Buning uchun o‘zgaruvchilar va o‘zgarmaslardan foydalaniladi.

C++ tilida o‘zgaruvchilar ma’lumotni saqlash uchun qo‘llaniladi. O‘zgaruvchining dasturda foydalanish mumkin bo‘lgan qandaydir qiymatlarni saqlaydigan komyuter xotirasidagi yacheyka ko‘rinishda ifodalash mumkin.

Kompyuter xotirasini yacheykalardan iborat qator sifatida qarash mumkin. Barcha yacheykalar ketma – ket nomerlangan. Bu nomerlar yacheykaning adresi deb ataladi. O‘zgaruvchilar biror – bir qiymatni saqlash uchun bir yoki bir nechta yacheykalarni band qiladi.

O‘zgaruvchining nomini (masalan, MyVariable) xotira yacheykasi adresi yozilgan yozuv deb qarash mumkin.

Masalan MyVariable o‘zgaruvchisi 102 – adresdagi yacheykadan boshlab saqlanadi. O‘zining o‘lchoviga muvofiq MyVariable o‘zgaruvchisi xotiradan bir yoki bir necha yacheykani band qilishi mumkin.

O‘zgaruvchilarning quyidagi tiplari mavjuddir:

bool – mantiqiy;

char – bitta simvol;

long char – uzun simvol;

int – butun son;

short yoki **short int** – kiska butun son

long yoki **long int** – uzun butun son

float xaqiqiy son;

long float yoki **double** – ikkilangan xaqiqiy son

long double – uzun ikkilangan xaqiqiy son

Butun sonlar o‘lchami. Bir xil tipdagi o‘zgaruvchilar uchun turli kompyuterlarda xotiradan turli hajmdagi joy ajratilishi mumkin. Lekin, bitta kompyuterda bir xil tipdagi ikkita o‘zgaruvchi bir xil miqdorda joy egallaydi.

char tipli o‘zgaruvchi bir bayt hajmni egallaydi. Ko‘pgina kompyuterlarda short int (qisqa butun) tipi ikki bayt, long int tipi esa 4 bayt joy egallaydi. Butun qiymatlar o‘lchovini kompyuter sistemasi va ishlatiladigan kompilyator aniqlaydi. 32 – razryadli kompyuterlarda butun o‘zgaruvchilar 4 bayt joy egallaydi. Quyidagi dastur sizning kompyuteringizdagi tiplarning o‘lchovini aniqlab beradi.

Tayanch tiplar uchun kompyuter xotirasidan ajratiladigan baytlarni aniqlash.

```
#include <iostream>
using namespace std;
int main()
{
cout << "int tipining o`lchami: \t"
<<sizeof(int)<<" bayt. "<<endl;
cout<< "short int tipining o`lchami:\t"
<< sizeof(short)<< "bayt. "<<endl;
cout<<"long int tipining o`lchami:\t"
<< sizeof(long)<<"bayt. "<<endl;
cout<< "char tipining o`lchami:\t"
<< sizeof(char)<<"bayt. "<<endl;
return 0;
};
```

Natija:

```
int tipining o`lchami: 4 bayt.
short int tipining o`lchami:2 bayt.
long int tipining o`lchami: 4 bayt. ;
char tipining o`lchami:1 bayt;
```

O‘zgaruvchiga qiymat berish. O‘zgaruvchilarni dasturning ixtiyoriy qismida ta’riflash yoki qayta ta’riflash mumkin.

Misol uchun:

```
int a, b1, ac; yoki
int a;
int b1;
 int ac;
```

O‘zgaruvchilar ta’riflanganda ularning qiymatlari aniqlanmagan bo‘ladi. Lekin o‘zgaruvchilarni ta’riflashda initsializatsiya ya’ni boshlang‘ich qiymatlarini ko‘rsatish mumkin.

Misol uchun:

```
int i=0;
char c='k';
```

O‘zgaruvchilarga qiymat berish uchun o‘zlashtirish operatori qo‘llaniladi. Masalan, Width o‘zgaruvchisiga 5 qiymatni berish uchun quyidagilarni yozish lozim:

```
unsigned short Width;
Width = 5;
```

Bu ikkala satrni Width o‘zgaruvchisini aniqlash jarayonida birgalikda yozish mumkin.

```
unsigned short Wigth = 5;
```

Bir necha o‘zgaruvchilarni aniqlash vaqtida ham ularga qiymat berish mumkin:

```
long width = 5, length = 7;
```

Bu misolda long tipidagi width o‘zgaruvchisi 5 qiymatni, shu tipdagi length o‘zgaruvchisi esa 7 qiymatni qabul qildi. Quyidagi dasturda o‘zgaruvchilarni aniqlashga oid misolni qaraymiz.

O‘zgaruvchilarning qo‘llanishi.

```
#include <iostream>
using namespace std;
int main()
{
```

```
int Buyi=5, Eni=10,Yuzasi;
cout << "Bo'yi:" << Buyi << " \n";
cout << "Eni:" << Eni << endl;
Yuzasi= Buyi*Eni;
cout << "Yuzasi:" << Yuzasi << endl;
return 0;
}
```

Natija:

Bo`yi: 5

Eni: 10

Yuzasi: 50

Ishorali va ishorasiz tiplar. Dasturda qo‘llaniladigan butun sonli tiplar ishorali va ishorasiz bo‘lishi mumkin. Ba’zan o‘zgaruvchi uchun faqatgina musbat sonni qo‘llash foydali bo‘ladi. Unsingned kalitli so‘ziciz keltirilgan butun sonli tiplar (short va long) ishorali hisoblanadi. Ishorali butun sonlar manfiy va musbat bo‘lishi mumkin. Ishorasiz sonlar esa doimo musbat bo‘ladi. Ishorasiz butun sonlar ustida amallar mod 2^n arifmetikasiga asoslangandir. Bu yerda n soni int tipi xotirada egallovchi razryadlar sonidir. Agar ishoraciz k soni uzunligi int soni razryadlar sonidan uzun bulsa, bu son qiymati k mod 2^n ga teng bo‘ladi. Ishorasiz k son uchun –k amali 2^n – k formula asosida hisoblanadi. Ishorali ya’ni signed tipidagi sonlarning eng katta razryadi son ishorasini ko‘rsatish uchun ishlatilsa unsigned (ishorasiz) tipdagi sonlarda bu razryad sonni tasvirlash uchun ishlatiladi.

Ishorasiz butun sonlarni ayirishda, agarda natija manfiy son bo‘lsa g‘ayrioddiy natija beradi. Buni quyida ko‘rishimiz mumkin.

Ayirish natijasida butun sonni to‘lib qolishiga misol

```
#include <iostream>
using namespace std;
int main()
{
```

```
unsigned int ayirma;
unsigned int kattaSon = 100;
unsigned int kichikSon = 50;
ayirma = kattaSon - kichikSon;
cout << "Ayirma:"<< ayirma<< " ga teng\n";
ayirma = kichikSon - kattaSon;
cout << "Ayirma:"<< ayirma<< " ga teng\n";
return 0;
}
```

Hatija:

Ayirma: 50 ga teng

Ayirma: 4294967246 ga teng

typedef kalitli so‘zi. unsigned short int kabi kalit so‘zlarni ko‘p martalab dasturda yozilishi zerikarli va diqqatvozlik talab qilganligi uchun C++ tilida bunday tiplarni typedef kalitli so‘zi yordamida psevdonimini (taxallusini) tuzish imkoniyati berilgan. typedef so‘zi tipni aniqlash ma’nosini bildiradi.

Psevdonim tuzishda tipning nomi yangi tuziladigan tip nomidan farqli bo‘lishi lozim. Bunda birinchi typedef kalitli so‘zi, keyin mavjud tip nomi, undan so‘ng esa yangi nom yoziladi. Masalan:

```
typedef unsigned short int ushort
```

Bu satrdan so‘ng ushort nomli yangi tip hosil bo‘ladi va u qayerda unsigned short int tipidagi o‘zgaruvchini aniqlash lozim bo‘lsa, shu joyda ishlatiladi.

Misol uchun yangi COD tipini kiritish:

```
typedef unsigned char COD;
COD simbol;
```

typedef operatori orqali tiplarning aniqlanishi

```
#include <iostream>
using namespace std;
typedef unsigned short int ushort;
int main()
```

```
{
ushort Buyi = 5;
ushort Eni = 10;
ushort Yuzasi = Buyi* Eni;
cout << "Yuzasi:" << Yuzasi << endl;
}
```

Natija:

Yuzasi: 50

Tiplar bilan ishlovchi amallar. Tiplarni o‘zgartirish amali quyidagi ko‘rinishga ega:

(tip_nomi) operand;

Bu amal operandlar qiymatini ko‘rsatilgan tipga keltirish uchun ishlatiladi. Operand sifatida kostanta, o‘zgaruvchi yoki qavslarga olinga ifoda kelishi mumkin. Misol uchun (long)6 amali konstanta qiymatini o‘zgartirmagan holda operativ xotirada egallagan baytlar sonini oshiradi. Bu misolda konstanta tipi o‘zgarmagan bo‘lsa, (double)6 yoki (float)6 amali konstanta ichki ko‘rinishini ham o‘zgartiradi. Katta butun sonlar xaqiqiy tipga keltirilganda sonning aniqligi yo‘qolishi mumkin.

sizeof amali operand sifatida ko‘rsatilgan obektning baytlarda xotiradagi xajmini hisoblash uchun ishlatiladi. Bu amalning ikki ko‘rinishi mavjud:

sizeof ifoda

sizeof (tip)

Misol uchun:

sizeof 3. 14=8

sizeof 3. 14f=4

sizeof 3. 14L=10

sizeof(char)=1

sizeof(double)=8.

Tiplarni keltirish. Binar arifmetik amallar bajarilganda tiplarni keltirish quyidagi qoidalar asosida amalga oshiriladi:

short va char tiplari int tipiga keltiriladi;

Agar operandlar biri long tipiga tegishli bo‘lsa ikkinchi operand ham long tipiga keltiriladi va natija ham long tipiga tegishli bo‘ladi;

Agar operandlar biri float tipiga tegishli bo‘lsa ikkinchi operand ham float tipiga keltiriladi va natija ham float tipiga tegishli bo‘ladi;

Agar operandlar biri double tipiga tegishli bo‘lsa ikkinchi operand ham double tipiga keltiriladi va natija ham double tipiga tegishli bo‘ladi;

Agar operandlar biri long double tipiga tegishli bo‘lsa ikkinchi operand ham long double tipiga keltiriladi va natija ham long double tipiga tegishli bo‘ladi.

O‘zgarmaslar

O‘zgarmaslar turlari. O‘zgaruvchilar kabi o‘zgarmaslar ham ma’lumotlarni saqlash uchun mo‘ljallangan xotira yacheykalarini o‘zida ifodalaydi. O‘zgaruvchilardan farqli ravishda ular dasturni bajarilishi jarayonida qiymati o‘zgarmaydi. O‘zgarmas e’lon qilinishi bilan unga qiymat berish lozim, keyinchalik bu qiymatni o‘zgartirib bo‘lmaydi.

C++ tilida ikki turdagi, literal va nomlangan o‘zgarmaslar aniqlangan.

Literalli o‘zgarmaslar to‘g‘ridan-to‘g‘ri dasturga kiritiladi. Masalan:

```
int myage =39;
```

Bu ifodada Myage int tipidagi o‘zgaruvchi, 39 soni esa literal o‘zgarmasdir.

Belgili o‘zgarmaslar. Belgili o‘zgarmaslar odatda bir bayt joyni egallaydi va bu 256 xil belgini saqlash uchun yetarlidir. Char tipi qiymatlarini 0. . 255 sonlar to‘plamiga yoki ASCII belgilar to‘plamiga interpretatsiya qilish mumkin.

ASCII belgilari deganda kompyuterlarda qo‘llaniladigan standart belgilar to‘plami tushuniladi. ASCII - bu American Standard Code for Information Interchange (Amerikaning axborot almashinishi uchun standart kodi) degan ma’noni anglatadi.

Maxsus belgilar. C++ kompilyatorida tekstlarni formatlovchi bir nechta maxsus belgilardan foydalaniladi. (Ulardan eng ko‘p tarqalgani jadvalda keltirilgan). Bu belgilarni dasturda ishlatishda «teskari slesh»dan foydalanamiz. Teskari sleshdan keyin boshqaruvchi belgi yoziladi. Masalan, tabulyatsiya belgisini dasturga qo‘yish uchun quyidagicha yozuvni yozish kerak.

char tab ='\t';

Bu misoldagi char tipidagi o‘zgaruvchi \t qiymatini qabul qiladi. Maxsus belgilar axborotlarni ekranga, faylga va boshqa chiqarish qurilmalariga chiqarishda formatlash uchun qo‘llaniladi.

Maxsus '\' simvolidan boshlangan simvollar eskeyp simvollar deyiladi. Simvolli konstanta qiymati simvolning kompyuterda qabul qilingan sonli kodiga tengdir.

ESC (eskeyp) simvollar jadvali:

Yezilishi	Ichki kodi	Simvoli(nomi)	Ma’nosi
\a	0x07	bel (audible bell)	Tovush signali
\b	0x08	bs (bascspase)	Bir kadam kaytish
\f	0x0C	ff (form feed)	Saxifani utkazish
\n	0x0A	lf (line feed)	Katorni utkazish
\r	0x0D	cr (carriage return)	Karetkani qaytarish
\t	0x09	ht (horizontal tab)	Gorizontal tabulyatsiya
\v	0x0B	vt (vertical tab)	Vertikal tabulyatsiya
\\	0x5C	\ (bacslash)	Teskari chizik
\'	0x27	‘ (single out)	Apostrif (oddiy qavs)
\"	0x22	" (double quote)	Ikkilik qavs
\?	0x3F	? (question mark)	Savol belgisi
\000	000	ixtiyoriy (octal number)	Simvol sakkizlik kodi

\xhh	0xhh	ixtiyoriy (hex number)	Simvol unoltilik kodi

Ma’lumotlarning butun son turi. Butun sonlar o‘nlik, sakkizlik yoki o‘n oltilik sanoq sistemalarida berilishi mumkin.

O‘nlik sanoq sistemasida butun sonlar 0-9 raqamlari ketma ketligidan iborat bo‘lib, birinchi rakami 0 bo‘lishi kerak emas.

Sakkizlik sanoq sistemasida butun sonlar 0 bilan boshlanuvchi 0-7 raqamlaridan iborat ketma ketlikdir.

O‘n oltilik sanoq sistemasida butun son 0x yoki 0X bilan boshlanuvchi 0-9 rakamlari va a-f yoki A-F harflaridan iborat ketma ketlikdir.

Masalan 15 va 22 o‘nlik sonlari sakkizlikda 017 va 026, o‘n oltilikda 0xF va 0x16 shaklda tasvirlanadi.

Ma’lumolarning uzun butun son turi.

Oxiriga l yoki L harflari qo‘yilgan o‘nlik, sakkizlik yoki o‘n oltilik butun son.

Ma’lumotlarning ishorasiz (unsigned) butun son turi:

Oxiriga u yoki U harflari qo‘yilgan o‘nlik, sakkizlik yoki o‘n oltilik oddiy yoki uzun butun son.

Ma’lumotlarning xaqiqiy son turi. Ma’lumotlarning xaqiqiy son turi olti qismdan iborat bo‘lishi mumkin: butun qism, nuqta, kasr qism, e yoki E belgisi, o‘nlik daraja, F yoki f suffikslari.

Masalan : 66. . 0. 12 3. 14F 1. 12e-12

Ma’lumolarning uzun xaqiqiy son turi :

Oxiriga L yoki l suffikslari kuyilgan xaqiqiy son.

Masalan: 2E+6L;

Mantiqiy konstanta. Mantiqiy konstantalar true(rost) va false(yolg‘on) qiymatlardan iborat. Ichki ko‘rinishi false – 0, ixtiyoriy boshqa qiymat true deb qaraladi.

Satrli konstanta. Satrli konstantalar C++ tili konstantalariga kirmaydi, balki leksemalari aloxida tipi hisoblanadi. Shuning uchun adabiyotda satrli konstantalar satrli leksemalar deb ham ataladi. .

Satrli konstanta bu ikkilik qavslarga olingan ixtiyoriy simvollar ketma ketligidir. Misol uchun "Men satrli konstantaman".

Satrlar orasiga eskeyp simvollar ham kirishi mumkin. Bu simvollar oldiga \ belgisi qo‘yiladi. Misol uchun:

"\n Bu satr \n uch katorga \n joylashadi".

Satr simvollari xotirada ketma ket joylashtiriladi va har bir satrli konstanta oxiriga avtomatik ravishda kompilyator tomonidan '\0' simvoli qo‘shiladi. Shunday satrning xotiradagi xajmi simvollar soni+1 baytga tengdir.

Ketma-ket kelgan va bo‘shliq, tabulyatsiya yoki satr oxiri belgisi bilan ajratilgan satrlar kompilyatsiya davrida bitta satrga aylantiriladi. Misol uchun:

"Salom" "Toshkent "

satrlari bitta satr deb qaraladi.

"Salom Toshkent"

Bu qoidaga bir necha katorga yozilgan satrlar ham bo‘ysinadi. Misol uchun :

"O‘zbekistonga "

"baxor "

"keldi"

qatorlari bitta qatorga mos:

"O‘zbekistonga baxor keldi"

Agar satrda '\' belgisi uchrasa va bu belgidan so‘ng to '\n' satr oxiri belgisigacha bo‘shlik belgisi kelsa bu bo‘shlik belgilari '\' va '\n' belgisi bilan birga satrdan o‘chiriladi. Satrning o‘zi keyingi satrda kelgan satr bilan qo‘shiladi.

"O‘zbekistonga \ "

" baxor\

" keldi"

qatorlari bitta qatorga mos:

"Uzbekistonga baxor keldi"

Nomlangan o‘zgarmaslar. Belgili o‘zgarmas – bu nomga ega bo‘lgan o‘zgarmasdir. C++ tilida belgili o‘zgarmasni aniqlashning ikki usuli mavjud:

1. #define direktivasi yordamida o‘zgarmasni aniqlash.
2. const kalitli so‘zi orqali o‘zgarmasni aniqlash.

An’anaviy usul hisoblangan #define direktivasi orqali o‘zgarmasni aniqlashni quyidagi misolda ko‘rishimiz mumkin.

#define StudentsPerClass 15

Bu holda StudentsPerClass o‘zgarmas hech qanday tipga tegishli bo‘lmaydi.

Preprotsessor StudentsPerClass so‘ziga duch kelganida uni 15 literaliga almashtiradi.

C++ tilida #define direktivasidan tashqari o‘zgarmasni aniqlashning nisbatan qulayroq bo‘lgan yangi usuli ham mavjud:

const unsigned short int StudentsPerClass=15

Bu misolda ham belgili konstanta StudentsPerClass nomi bilan aniqlanayapti va unga unsigned short int tipi berilyapti. Bu usul bir qancha imkoniyatlarga ega bo‘lib u sizning dasturingizni keyingi himoyasini yengillashtiradi. Bu o‘zgarmasni oldingisidan eng muhim afzalligi uning tipga egaligidir.

Bu konstantalar qiymatlarini dasturda o‘zgartirish mumkin emas. Konstantalar nomlari dasturchi tomonidan kiritilgan va xizmatchi so‘zlardan farqli bo‘lgan identifikatorlar bo‘lishi mumkin. Odatda nom sifatida katta lotin harflari va ostiga chizish belgilari kombinatsiyasidan iborat identifikatorlar ishlatiladi. Nomlangan konstantalar quyidagi shaklda kiritiladi:

sonst tip konstanta_nomi=konstanta_qiymati.

Misol uchun:

const double EULER=2. 718282;

const long M=99999999;

const R=765;

Oxirgi misolda konstanta tipi ko‘rsatilmagan, bu konstanta int tipiga tegishli deb hisoblanadi.

Belgili o‘zgarmaslarni literal o‘zgarmaslarga nisbatan ishlatish qulayroqdir. Chunki agarda bir xil nomli literalli o‘zgaruvchini qiymatini o‘zgartirmoqchi bo‘lsangiz butun dastur bo‘yicha uni o‘zgartirishga to‘g‘ri keladi, belgili o‘zgarmaslarni esa faqatgina birining qiymatini o‘zgartirish yetarli.

To‘plam o‘zgarmaslari. Bunday o‘zgarmaslarni hosil qilish uchun yangi berilgan ma’lumotlar tiplari tuziladi va undan so‘ng bu tipga tegishli o‘zgarmasli qiymatlar to‘plami bilan chegaralangan o‘zgaruvchilar aniqlanadi.

Sanovchi konstantalar enum xizmatchi so‘zi yordamida kiritilib, int tipidagi sonlarga qulay so‘zlarni mos qo‘yish uchun ishlatiladi.

Misol uchun:

enum{one=1,two=2,three=3};

Agar son qiymatlari ko‘rsatilmagan bo‘lsa eng chapki so‘zga 0 qiymati berilib kolganlariga tartib bo‘yicha o‘suvchi sonlar mos qo‘yiladi:

enum{zero,one,two};

Bu misolda avtomatik ravishda konstantalar quyidagi qiymatlarni qabul qiladi:

zero=0, one=1, two=2;

Konstantalar aralash ko‘rinishda kiritilishi ham mumkin:

enum(zero,one,for=4,five,seeks}.

Bu misolda avtomatik ravishda konstantalar quyidagi qiymatlarni qabul kiladi:

zero=0, one=1, for=4;five=5,seeks=6;

Sanoqli tiplarni hosil qilish uchun enum kalitli so‘zi va undan keyin tip nomi hamda figurali qavs ichida vergullar bilan ajratilgan o‘zgarmas qiymatlari ro‘yxati ishlatiladi. Masalan, RANG nomli sanoqli tip deb e’lon qilaylik va uning uchun 5 ta QIZIL, KUK, YASHIL, OQ, QORA qiymatlarini aniqlaylik.

enum RANG { QIZIL, KUK, YASHIL, OQ, QORA };

Bunda ifoda ikkita ishni bajaradi:

1. RANG nomli yangi sanoqli tip hosil qiladi;
2. Quyidagi belgili o‘zgarmaslarni aniqlaydi.

0 qiymat bilan QIZIL;

1 qiymat bilan KUK;

2 qiymat bilan YASHIL va hokazo;

Har bir sanoqli o‘zgarmas biror bir aniqlangan butun qiymatga mos keladi.

Boshlang‘ich holatda o‘zgarmaslarga 0 dan boshlab qiymat beriladi. Lekin, ixtiyoriy o‘zgarmasga boshqa qiymatni o‘zlashtirish ham mumkin. Bunda ularga qiymat berish o‘sish tartibida bo‘lishi lozim. Masalan,

enum RANG{QIZIL=100,KUK=200,YASHIL=300,OQ,QORA=500};

ko‘rinishda sanoqli tipni aniqlasak QIZIL o‘zgarmasi 100 ga, KUK – 200 ga, Yashil – 300 ga, OQ –301 ga, QORA – 500 ga teng bo‘ladi.

Yana bir misol:

enum BOOLEAN {NO, YES};

Konstantalar qiymatlari:

NO=0, YES=1;

7.3. DASTURLASH OPERATORLARI.

Reja:

1. C++ dasturlash tilidagi o’zlashtirish amali.
2. Qiymatni bir birlikka o’zgartiruvchi operatorlar.
3. Amallar ustivorligi
4. Murakkab qiymat berish amali
5. Taqqoslash operatorlari.

Tayanch iboralar: C++ dasturlash tilidagi operatsiyalar. Arifmetik operatorlar. Qiymatni bir birlikka o’zgartiruvchi operatorlar. Taqqoslash operatorlari.

Amallar

O‘zlashtirish amali. O‘zlashtirish amali (=) o‘zidan chap tomonda turgan operand qiymatini tenglik belgisidan o‘ng tomondagilarni hisoblangan qiymatiga almashtiradi. Masalan,

x = a+b;

ifodasi x operandga a va v o‘zgaruvchilarni qiymatlarini qo‘shishdan hosil bo‘lgan natijani o‘zlashtiradi.

O‘zlashtirish amaliidan chapda joylashgan operand adresli operand yoki l–qiymat deyiladi. O‘zlashtirish amalidan o‘ngda joylashgan operand operatsion operand yoki r–qiymat deyiladi.

O‘zgarmaslar faqatgina r–qiymat bo‘lishi mumkin va hech qachon adresli operand bo‘la olmaydi, chunki dasturning bajarilishi jarayonida o‘zgarmas qiymatini o‘zgartirib bo‘lmaydi.

35 = x // noto‘g‘ri!

l–qiymat esa r–qiymat bo‘lishi mumkin.

Qiymat berish amali = binar amal bo‘lib chap operandi odatda o‘zgaruvchi o‘ng operandi odatda ifodaga teng bo‘ladi. Misol uchun

z=4. 7+3. 34

Bu qiymati 8. 04 ga teng ifodadir. Bu qiymat Z o‘zgaruvchiga ham beriladi.

Bu ifoda oxiriga nuqta vergul; belgisi qo‘yilganda operatorga aylanadi.

z=4. 7+3. 34

Bitta ifodada bir necha qiymat berish amallari qo‘llanilishi mumkin. Misol uchun:

c=y=f=4. 2+2. 8;

Matematik amallar. C++ tilida 5 ta asosiy matematik amallar qo‘llaniladi: qo‘shish (+), ayirish (-), ko‘paytirish (*), butun songa bo‘lish (/) va modul bo‘yicha bo‘lish (%)(qoldiqni olish). Amallar odatda unar ya’ni bitta operandga qo‘llaniladigan amallarga va binar ya’ni ikki operandga qo‘llaniladigan amallarga ajratiladi.

Binar amallar additiv ya’ni + qo‘shish va – ayirish amallariga, hamda multiplikativ ya’ni * ko‘paytirish, / bo‘lish va % modul olish amallariga ajratiladi.

Butun songa bo‘lish odatdagi bo‘lishdan farq qiladi. Butun songa bo‘lishdan hosil bo‘lgan bo‘linmaning faqatgina butun qismi olinadi.

Butun sonni butun songa bo‘lganda natija butun songacha yaxlitlanadi. Misol uchun 20/3=6; (-20)/3=-6; 20/(-3)=-6.

Modul amali butun sonni butun songa bo‘lishdan xosil bo‘ladigan qoldiqqa tengdir. Agar modul amali musbat operandlarga qo‘llanilsa, natija ham musbat bo‘ladi, aks holda natija ishorasi kompilyatorga bog‘liqdir.

Masalan, 21 sonini 4 ga bo‘lsak 5 soni va 1 qoldiq hosil bo‘ladi. 5 butun songa bo‘lishni qiymati, 1 esa qoldiqni olish qiymati hisoblanadi.

Inkrement va dekrement. Dasturlarda o‘zgaruvchiga 1 ni qo‘shish va ayirish amallari juda ko‘p hollarda uchraydi. C++ tilida qiymatni 1 ga oshirish inkrement, 1

ga kamaytirish esa dekrement deyiladi. Bu amallar uchun maxsus operatorlar mavjuddir.

Inkrement operatori (++) o‘zgaruvchi qiymatini 1 ga oshiradi, dekrement operatori (—) esa o‘zgaruvchi qiymatini 1 ga kamaytiradi. Masalan, s o‘zgaruvchisiga 1 qiymatni qo‘shmoqchi bo‘lsak quyidagi ifodani yozishimiz lozim.

C++ //s o‘zgaruvchi qiymatini 1 ga oshirdik.

Bu ifodani quyidagicha yozishimiz mumkin edi.

s=s+1;

Bu ifoda o‘z navbatida quyidagi ifodaga teng kuchli:

s+=1;

Prefiks va postfiks. Inkrement operatori ham, dekrement operatori ham ikki variantda ishlaydi: prefiksli va postfiksli. Prefiksli variantda ular o‘zgaruvchidan oldin (++Age), postfiksli variantda esa o‘zgaruvchidan keyin (Age++) yoziladi.

Oddiy ifodalarda bu variantlarni qo‘llanishida farq katta emas, lekin bir o‘zgaruvchiga boshqa o‘zgaruvchining qiymatini o‘zlashtirishda ularning qo‘llanilishi boshqacha harakterga ega. Prefiksli operator qiymat o‘zlashtirilguncha, postfiksli operator esa qiymat o‘zlashtirilgandan keyin bajariladi. Misol uchun i qiymati 2 ga teng bo‘lsin, u holda 3+(++i) ifoda qiymati 6 ga, 3+i++ ifoda qiymati 5 ga teng bo‘ladi. Ikkala holda ham i qiymati 3 ga teng bo‘ladi.

Amallar ustivorligi. Murakkab ifodalarda qaysi amal birinchi navbatda bajariladi, qo‘shishmi yoki ko‘paytirishmi? Masalan:

x=5+3*8;

ifodada agarda birinchi qo‘shish bajarilsa natija 64 ga, agarda ko‘paytirish birinchi bajarilsa natija 29 ga teng bo‘ladi.

Har bir operator prioritet qiymatiga ega. Ko‘paytirish qo‘shishga nisbatan yuqoriroq prioritetga ega. Shuning uchun bu ifoda qiymati 29 ga teng bo‘ladi.

Agarda ikkita matematik ifodaning prioriteti teng bo‘lsa, ular chapdan o‘ngga qarab ketma – ket bajariladi.

Demak

x=5+3+8*9+6*4

ifodada birinchi ko‘paytirish amallari chapdan o‘ngga qarab bajariladi 8*9=72 va 6*4=24. Keyin bu ifoda soddaroq ko‘rinish hosil qiladi.

x=5+3+72+24

Endi qo‘shishni ham xuddi shunday chapdan unga qarab bajaramiz:

5+3=8; 8+72= 80; 80+24=104;

Lekin, barcha amallar ham bu tartibga amal qilmaydi. Masalan, o‘zlashtirish amali o‘ngdan chapga qarab bajariladi.

Additiv amallarining ustivorligi multiplikativ amallarining ustivorligidan pastrokdir.

Unar amallarning ustivorligi binar amallardan yuqoridir.

Murakkab qiymat berish amali. C++ tilida murakkab qiymat berish amali mavjud bo‘lib, umumiy ko‘rinishi quyidagichadir:

O‘zgaruvchi_nomi amal= ifoda;

Bu yerda amal quyidagi amallardan biri *, /, %, +, -, &,^, |, <<, >>.

Misol uchun:

x+=4 ifoda x=x+4 ifodaga ekvivalentdir;

x*=a ifoda x=x*a ifodaga ekvivalentdir;

x/=a+b ifoda x=x/(a+b) ifodaga ekvivalentdir;

x>>=4 ifoda x=x>>4 ifodaga ekvivalentdir;

Imlo belgilari amal sifatida. C++ tilida ba’zi bir imlo belgilari ham amal sifatida ishlatilishi mumkin. Bu belgilardan oddiy () va kvadrat [] qavslardir. Oddiy qavslar binar amal deb qaralib ifodalarda yoki funksiyaga murojaat qilishda foydalaniladi. Funksiyaga murojaat qilish quyidagi shaklda amlga oshiriladi:

<funksiya nomi> (<argumentlar ro‘yxati>). Misol uchun sin(x) yoki max(a,b).

Murakkab ifodalarni tuzishda ichki qavslardan foydalaniladi. Masalan, sizga sekundlarning umumiy soni keyin esa barcha qaralayotgan odamlar soni, undan keyin esa ularning ko‘paytmasini hisoblash kerak bo‘lsin.

TotalPersonSeconds=((NumMinutesToThink+ NumMinutesToType)*60*(PeopleInTheOffice+ PeopleOnVocation))

Bu ifoda quyidagicha bajariladi. Oldin NumMinutesToThink o‘zgaruvchisining qiymati NumMinutesToType o‘zgaruvchisi qiymatiga qo‘shiladi. Keyin esa hosil qilingan yig‘indi 60 ga ko‘paytiriladi. Bundan keyin PeopleInTheOffice o‘zgaruvchi qiymati PeopleOnVocation qiymatiga qo‘shiladi. Keyin esa sekundlar soni kishilar soniga ko‘paytiriladi.

Kvadrat qavslardan massivlarga murojaat kilishda foydalaniladi. Bu murojaat quyidagicha amalga oshiriladi:

<massiv nomi>[<indeks>]. Misol uchun a[5] yoki b[n][m].

Vergul simvolini ajratuvchi belgi deb ham karash mumkin amal sifatida ham karash mumkin. Vergul bilan ajratilgan amallar ketma-ketligi bir amal deb qaralib, chapdan o‘ngga hisoblanadi va oxirgi ifoda qiymati natija deb karaladi. Misol uchun:

d=4,d+2 amali natijasi 8 ga teng.

Shartli amal. Shartli amal ternar amal deyiladi va uchta operanddan iborat bo‘ladi:

<1-ifoda>?<2-ifoda>:<3-ifoda>

Shartli amal bajarilganda avval 1- ifoda hisoblanadi. Agar 1-ifoda qiymati 0 dan farqli bo‘lsa 2- ifoda hisoblanadi va qiymati natija sifatida qabul qilinadi, aks holda 3-ifoda hisoblanadi va qiymati natija sifatida qabul qilinadi.

Misol uchun modulni hisoblash: x<0?-x:x yoki ikkita son kichigini hisoblash

a<b?a:b.

Shuni aytish lozimki shartli ifodadan har qanday ifoda sifatida foydalanish mumkin. Agar F FLOAT tipga,a N – INT tipga tegishli bo‘lsa,

(N > 0) ? F : N

ifoda N musbat yoki manfiyligidan qat’i nazar DOUBLE tipiga tegishli bo‘ladi.

Shartli ifodada birinchi ifodani qavsga olish shart emas.

Mantiqiy amallar. Dasturlashda bir emas balki bir nechta shartli ifodalarni tekshirish zaruriyati juda ko‘p uchraydi. Masalan, x o‘zgaruvchisi y o‘zgaruvchisidan, y esa o‘z navbatida z o‘zgaruvchisidan kattami sharti bunga misol bo‘la oladi. Bizning

dasturimiz mos amalni bajarishdan oldin bu ikkala shart rost yoki yolg‘onligini tekshirishi lozim.

Quyidagi mantiq asosida yuqori darajada tashkil qilingan signalizatsiya sistemasini tasavvur qiling. Agarda eshikda signalizatsiya o‘rnatilgan bo‘lsa VA kun vaqti kech soat olti VA bugun bayram YOKI dam olish kuni BO‘LMASA politsiya chaqirilsin. Barcha shartlarni tekshirish uchun C++ tilining uchta mantiqiy operatori ishlatiladi. Ular jadvalda keltirilgan

Amal	Belgi	Misol
VA	&&	1ifoda && 2ifoda
YOKI	||	1ifoda||2ifoda
INKOR	!	!ifoda

Mantiqiy amallar || (dizyunksiya); && (konyunksiya); !(inkor) amallari deb ataladi. Mantiqiy amallarni butun sonlarga qo‘llash mumkin. Bu amallarning natijalari quyidagicha aniqlanadi:

x||y amali 1 ga teng agar x>0 yoki y>0 bo‘lsa, aksincha 0 ga teng

x&&y amali 1 ga teng agar x>0 va y>0 bo‘lsa, aksincha 0 ga teng

!x amali 1 ga teng agar x>0 bo‘lsa, aksincha 0 ga teng

Bu misollarda amallar ustivorligi oshib borish tartibida berilgandir.

Inkor ! amali unar qolganlari binar amallardir.

Mantiqiy ko‘paytirish amali. Mantiqiy ko‘paytirish amali ikkita ifodani hisoblaydi, agar ikkala ifoda true qiymat qaytarsa VA operatori ham true qiymat qaytardi. Agarda sizning qorningiz ochligi rost bo‘lsa VA sizda pul borligi ham rost bo‘lsa siz supermarketga borishingiz va u yerdan o‘zingizga tushlik qilish uchun biror bir narsa harid qilishingiz mumkin. Yoki yana bir misol, masalan,

(x==5)&&(y==5)

mantiqiy ifodasi agarda x va u o‘zgaruvchilarini ikkalasining ham qiymatlari 5 ga teng bo‘lsagina true qiymat qaytaradi. Bu ifoda agarda o‘zgaruvchilardan birortasi 5 ga teng bo‘lmagan qiymat qabul qilsa false qiymatini qaytaradi. Mantiqiy ko‘paytirish operatori faqatgina o‘zining ikkala ifodasi ham rost bo‘lsagina true qiymat qaytaradi.

Mantiqiy ko‘paytirish amali && belgi orqali belgilanadi.

Mantiqiy qo‘shish amali. Mantiqiy qo‘shish amali ham ikkita ifoda orqali hisoblanadi. Agarda ulardan birortasi rost bo‘lsa mantiqiy qo‘shish amali true qiymat qaytaradi. Agarda sizda pul YOKI kredit kartochkasi bo‘lsa, siz schyotni to‘lay olasiz. Bu holda ikkita shartning birdaniga bajarilishi: pulga ham va kredit kartochkasiga ham ega bo‘lishingiz shart emas. Sizga ulardan birini bajarilishi yetarli. Bu operatorga oid yana bir misolni qaraymiz. Masalan,

(x==5)||(y==5)

ifodasi yoki x o‘zgaruvchi qiymati, yoki u o‘zgaruvchi qiymati, yoki ikkala o‘zgaruvchining qiymati ham 5 ga teng bo‘lsa rost qiymat qaytaradi.

Mantiqiy inkor amali. Mantiqiy inkor operatori tekshirilayotgan ifoda yolg‘on bo‘lsa true qiymat qaytaradi. Agarda tekshirilayotgan ifoda rost bo‘lsa inkor operatori false qiymat qaytaradi. Masalan, (!(x==5))

ifodasining qiymati, agarda x o‘zgaruvchisi 5 ga teng bo‘lmasa true qiymat qaytaradi. Bu ifodani boshqacha ham yozish mumkin:

(x!=5)

Munosabat amallari. Bunday amallar ikkita qiymatni teng yoki teng emasligini aniqlash uchun ishlatiladi. Taqqoslash ifodasi doimo true (rost) yoki false (yolg‘on) qiymat qaytaradi. Munosabat amallari arifmetik tipdagi operandlarga qo‘llanilsa qiymatlari 1 ga teng agar nisbat bajarilsa va aksincha 0 ga tengdir.

Munosabat amallarining qo‘llanilishiga oid misol jadvalda keltirilgan.

Nomi	Amal	Misol	Qaytaradigan qiymat
Tenglik	==	100==50 50==50	false true
Teng emas	!=	100!=50 50!=50	true false
Kata	>	100>50 50>50	true false
Katta yoki teng	>=	100>=50 50>=50	true true
Kichik	<	100<50	true

		50<50	false
Kichik yoki teng	<=	100<=50 50<=50	true true

Katta >, kichik <, katta yoki teng >=, kichik yoki teng <= amallarining ustivorligi bir xildir.

Teng == va teng emas != amallarining ustivorligi o‘zaro teng va kolgan nisbat amallaridan pastdir.

Razryadli amallar. Razryadli amallar natijasi butun sonlarni ikkilik ko‘rinishlarining har bir razryadiga mos mantiqiy amallarni qo‘llashdan xosil bo‘ladi. Masalan 5 kodi 101 ga teng va 6 kodi 110 ga teng.

6&5 qiymati 4 ga ya’ni 100 ga teng.

6|5 qiymati 7 ga ya’ni 111 ga teng.

6^5 qiymati 3 ga ya’ni 011 ga teng.

~6 qiymati 4 ga ya’ni 010 ga teng.

Bu misollarda amallar ustivorligi oshib borishi tartibida berilgandir.

Bu amallardan tashqari M<<N chapga razryadli siljitish va M>>N o‘ngga razryadli siljitish amallari qo‘llaniladi. Siljitish M butun sonning razryadli ko‘rinishiga qo‘llaniladi. N nechta pozitsiyaga siljitish kerakligini ko‘rsatadi.

Chapga N pozitsiyaga surish bu operand qiymatini ikkining N chi darajasiga ko‘paytirishga mos keladi. Misol uchun 5<<2=20. Bu operatorning bitli ko‘rinishi: 101<<2=10100.

Agar operand musbat bo‘lsa N pozitsiyaga ungga surish chap operandni ikkining N chi darajasiga bo‘lib kasr qismini tashlab yuborishga mosdir. Misol uchun 5>>2=1. Bu operatorning bitli ko‘rinishi 101>>2=001=1. Agarda operand qiymati manfiy bo‘lsa ikki variant mavjuddir: arifmetik siljitishda bo‘shatilayotgan razryadlar ishora razryadi qiymati bilan to‘ldiriladi, mantiqiy siljitishda bo‘shatilayotgan razryadlar nullar bilan to‘ldiriladi.

Razryadli inkor amali unar qolgan amallar binar amallarga kiradi.

Razryadli surish amallarining ustivorligi o‘zaro teng, razryadli inkor amalidan past, kolgan razryadli amallardan yuqoridir.

Amallar ustivorligi jadvali

Rang	Amallar	Yo‘nalish
1	() [] -> ::.	Chapdan o‘ngga
2	! ~ + - ++ -- & * (tip) sizeof new delete tip()	O‘ngdan chapga
3	. * ->*	Chapdan o‘ngga
4	* / % (multiplikativ binar amallar)	Chapdan o‘ngga
5	+ - (additiv binar amallar)	Chapdan o‘ngga
6	<< >>	Chapdan o‘ngga
7	< <= >= >	Chapdan o‘ngga
8	= !=	Chapdan o‘ngga
9	&	Chapdan o‘ngga
10	^	Chapdan o‘ngga
11	|	Chapdan o‘ngga
12	&&	Chapdan o‘ngga
13	||	Chapdan o‘ngga

14	?:(shartli amal)	O‘ngdan chapga
15	= *= /= %= += -= &= ^= |= <<= >>=	O‘ngdan chapga
16	, (vergul amali)	Chapdan o‘ngga

Mavzuga oid savollar

1. Nima uchun literalli o‘zgarmasga nisbatan belgili o‘zgarmasni ishlatish yaxshiroq?
2. const kalitli so‘zini #define direktivasi o‘rniga qo‘llashni afzalligi nimada?
3. #include direktivasi qanday vazifani bajaradi.
4. main() funksiyasining o‘ziga xos xususiyati nimadan iborat?
5. Qanday izoh turlarini bilasiz ? Ular nima bilan farq qiladi?
6. Izohlar bir necha qatorda yozilishi mumkinmi?
7. x=5+7 yozuvi ifoda bo‘la oladimi? Uning qiymati nechaga teng?
8. 201/4 ifodaning qiymati nechaga teng?
9. 201%4 ifoda qiymati nechaga teng?
10. x = 3 va x = =3 ifodalarning farqi nimadan iborat?

7.4. SHARTLI OPERATORLAR.

Reja:

1. if operatori.
2. else kalit so‘zi
3. if operatori orqali murakkab konstruksiyalarni hosil qilish
4. Tanlash operatori
5. Shartsiz o‘tish operatori

Tayanch iboralar: C++ dasturlash tilida o’tish operatori. if operatori. else kalit so‘zi. if operatori orqali murakkab konstruksiyalarni hosil qilish Shartli operatorning qisqa ko’rinishi. Shartli operatorning uzun ko’rinishi. Tanlash operatorlari. Shartsiz o‘tish operatori. Ko’p tarmoqlanishlar va variant tanlash operatorlari.

if operatori. Odatda dastur satrma–satr tartib bilan bajariladi. If operatori shartni tekshirish (masalan, ikki o‘zgaruvchi tengmi) va uning natijasiga bog‘liq ravishda dasturni bajarilish tartibini o‘zgartirish imkonini beradi. If operatorining oddiy shakli quyidagi ko‘rinishdadir:

```
if (shart)
ifoda;
```

Qavs ichidagi shart ixtiyoriy ifoda bo‘lishi mumkin. Agarda bu ifoda false qiymatini qaytarsa undan keyingi ifoda yoki blok tushirib qoldiriladi. Agarda shart true qiymat qaytarsa navbatdagi ifoda bajariladi. Quyidagi misolni qaraymiz:

```
If (kattaSon>kichikSon)
kattaSon=kichikSon;
```

Bu yerda kattaSon va kichikSon o‘zgaruvchilari taqqoslanayapti. Agarda kattaSon o‘zgaruvchisi qiymati katta bo‘lsa, bu navbatdagi qatorda unga qiymat sifatida kichikSon o‘zgaruvchisining qiymati o‘zlashtiriladi.

if operatorida figurali qavs ichiga olingan ifodalar blokini ham ishlatish mumkin.

```
if (shart)
{
    1  - ifoda
    2  - ifoda
    3  - ifoda
}
```

Quyida ifodalar blokining qo‘llanilishiga oid misol keltirilgan

```
if(kattaSon>kichikSon)
{ kattaSon=kichikSon
 cout<<"kattaSon:"<<kattaSon << "/n";
 cout<<"kichikSon:"<<kichikSon<< "/n";
}
```

Bu holda kattaSon o‘zgaruvchisiga nafaqat kichikSon o‘zgaruvchisi o‘zlashtirilayapti, balki ekranga bu haqida axborot ham chiqarilayapti.

Munosabat operatorining qo‘llanilishi orqali tarmoqlanishga misol

```
#include <iostream>
using namespace std;
int main( )
{
int BuxoroGol, PaxtakorGol;
 cout<<"Buxoro komandasi kiritgan to`plar"<< "sonini yozing:";
 cin >> BuxoroGol;
 cout<<"Paxtakor komandasi kiritgan";
 cout<< "to`plar sonini yozing:";
 cin >> PaxtakorGol;
cout << "\n";
if ( BuxoroGol>PaxtakorGol)
cout << "Yashasin Buxoro!\n";
if (BuxoroGol < PaxtakorGol)
{
 cout << "Yashasin PaxtakorGol \n";
 cout << "Bugun Toshkentda bayram!\n";
};
if (BuxoroGol==PaxtakorGol)
{
 cout << "Durrangmi? Yo-oq? Bo`lishi"<<
 cout<<" mumkin emas \n";
 cout <<"Paxtakorning kiritgan to`plari";
 cout<< "haqida ma`lumotni qaytadan yozing\n";
};
cin >> PaxtakorGol;
if (BuxoroGol>PaxtakorGol)
{
cout<<"Buxoro yutishini oldindan bilgan";
```

```
cout<<" edim! Shuning uchun qayta so`radim\n";
cout<< "Yashasin Buxoro!";
}
if (BuxoroGol<PaxtakorGol)
{
cout<<"Paxtakor yutishini oldindan bilgan";
cout<<" edim! Shuning uchun qayta so`radim\n";
cout<< "Yashasin Paxtakor!";
cout << "Bugun Toshkentda bayram!\n";
}
if (BuxoroGol==Paxta
korGol)
cout<<"Qoyil! Haqiqatan ham during ekan\n";
cout<<"\n Ma`lumotingiz uchun rahmat\n";
return 0;
}
```

Natija:

```
Buxoro komandasi kiritgan to`plar sonini yozing:3
Paxtakor komandasi kiritgan to`plar sonini yozing:3
Durrangmi? Yo-oq? Bo`lishi mumkin emas
Paxtakorning kiritgan to`plari haqida ma`lumotni qaytadan yozing: 2
Buxoro yutishini oldindan bilgan edim! Shuning uchun qayta so`radim
Yashasin Buxoro!
```

else kalit so‘zi. Dasturlarda ko‘p hollarda biror bir shartning bajarilishiga (ya’ni bu shart true qiymat qaytarsa) bog‘liq ravishda bir blok, uning bajarilmasligiga asosan esa (ya’ni bu shart false qiymat qaytarsa) boshqa bir blokning bajarilishi talab qilinadi. Quyidagi – listingda birinchi tekshirish *(*BuxoroGol>PaxtakorGol*)* true qiymat qaytarsa ekranda bir xabar, false qiymatida esa boshqa bir xabar chiqariladi.

Bunday masalalarni yuqorida ko‘rsatilgan usul, ya’ni qator shartlarni tekshirish uchun bir nechta if operatorini qo‘llash orqali hal qilish mumkin, lekin bu tushunish

uchun biroz murakkabroqdir. Dasturning soddaligini ta’minlash uchun else kalitli so‘zidan foydalanish mumkin.

```
if (shart)
 Ifoda;
else
 Ifoda;
```

else kalitli so‘zining ishlatilishi.

```
#include <iostream>
using namespace std;
int main()
{
int BirinchiSon, IkkinchiSon;
cout << "Katta sonni kiriting:";
cin >> BirinchiSon;
cout<<"\n Kichik sonni kiriting:";
cin >> IkkinchiSon;
if (BirinchiSon > IkkinchiSon)
cout << "\n Rahmat! \n";
else
cout << "\n Ikkinchisi katta son-ku!";
return 0;
}
```

Natija:

```
Katta sonni kiriting: 10
Kichik sonni kiriting: 12
Ikkinchisi katta son - ku!
```

Sanoqli o‘zgarmasni qo‘llanishi

```
#include <iostream>
using namespace std;
int main( )
```

```
{
enum Kunlar{Dushanba, Seshanba, Chorshanba,
Payshanba, Juma, Shanba, Yakshanba};
int tanlash;
cout << "Kun nomerini kiriting (0-6):";
cin>>tanlash;
if (tanlash==Yakshanba || tanlash == Shanba)
cout<<"\nBugun siz uchun dam olish kuni!"
<<endl;
else
cout << "\n Bugun siz uchun ish kuni. \n";
return 0;
};
```

Hatija:

```
Kun nomerini kiriting(0-6): 6
Bugun siz uchun dam olish kuni!
```

if operatori orqali murakkab konstruksiyalarni hosil qilish. if–else konstruksiyasida ifodalar blokida ixtiyoriy operatorlarni ishlatishda hech qanday chegara yo‘q. Shu jumladan, ifodalar bloki ichida yana if–else operatorlarini ishlatish mumkin. Bu holda bir nechta if operatoridan iborat ichma – ich konstruksiya hosil bo‘ladi.

```
if (1–shart)
{
 if (2–shart)
 1–ifoda
 else
 {
 if (3–shart)
 2–ifoda
 else
```

```
3–ifoda
} }
else

4–ifoda;
```

Ushbu bir nechta if operatoridan tashkil topgan konstruksiya quyidagi tartibda ishlaydi: agarda 1–shart va 2–shart rost bo‘lsa 1–ifoda bajariladi. Agarda 1–shart rost va 2–shart yolg‘on natija qaytarsa, u holda 3–shart tekshiriladi va agarda bu shart rost bo‘lsa 2–ifoda, yolg‘on bo‘lsa esa 3–ifoda bajariladi. Va eng oxiri, agarda 1–shart yolg‘on bo‘lsa 4–ifoda bajariladi. Bunday murakkab konstruksiyaga misol quyida keltirilgan.

```
if operatori ichki bo‘lgan murakkab konstruksiya
#include <iostream>
using namespace std;
{
// Ikkita son kiritamiz. Ularni BirinchiSon
//va IkkinchiSon o‘zgaruvchilariga beramiz
//Agarda KattaSon qiymati KichikSon
// qiymatidan katta bo‘lsa katta son
//kichigiga koldiksiz bo‘linishini tekshira-
// miz. Agarda u koldiksiz bo‘linsa ular
//teng yoki teng emasligini tekshiramiz.
int BirinchiSon, IkkinchiSon;
cout<<"Ikkita son kiriting. \n Birinchisi: ";
cin >> BirinchiSon;
cout << "\n Ikkinchisi:";
cin >> IkkinchiSon;
cout << "\n\n";
if (BirinchiSon>=IkkinchiSon)
```

```
if ( (BirinchiSon%IkkinchiSon)==0)
{
if (BirinchiSon==IkkinchiSon)
cout<< "Ular bir – biriga teng!\n";
else
cout<< "Birinchi son ikkinchisiga"<<"karrali!\n";
}
else
cout<<"Ikkinchi son katta!\n";
return 0;
}
```

Natija:

Ikkita son kiriting
Birinchisi: 9
Ikkinchisi: 3
Birinchi son ikkinchisiga karrali!

Tanlash operatori

switch operatori. Oldingi mavzularda biz if va if/else operatorlari bilan tanishgan edik. Lekin ayrim masalalarni yechishda if operatori ichida ko‘p sondagi if operatorlarini qo‘llashga to‘g‘ri keladi. Bu esa dasturni yozishni ham, uni tushinishni ham murakkablashtirib yuboradi. Bunday muammoni yechish uchun C++ tilida switch operatori qo‘llaniladi. Bu operatorning if operatoridan asosiy farqi shuki, unda bir yo‘la bir nechta shart tekshiriladi.

Switch operatorining qo‘llanilishi. switch operatorining qo‘llanilish sintaksisi quyidagicha:

```
switch(ifoda)
{
case 1-nchi qiymat: ifoda;
case 2-nchi qiymat: ifoda;
…
```

```
case n-nchi qiymat: ifoda;
default : ifoda;
}
```

switch operatori orqali dasturning tarmoqlanishi bir necha mumkin bo‘lgan qiymatlarni qaytaruvchi ifodaning natijasi asosida tashkil etiladi. switch operatoridagi qavs ichida berilgan ifodaning qaytargan qiymati case operatoridan keyinda ko‘rsatilgan qiymat bilan solishtiriladi. Ifodaning qiymati bilan case operatoridan keyingi qiymat mos kelsa tanlangan case operatoridan keyingi barcha satrlar bajariladi. Bunda amallarni bajarilishi break operatorigacha davom etadi.

Agarda case operatorlari qiymatidan birortasi ham qaytarilgan qiymatga mos kelmasa default operatoridan keyingi dastur satrlari bajariladi. Agarda bu operator mavjud bo‘lmasa boshqaruv switch bloki tanasidan chiqadi va keyingi dastur satrlariga beriladi.

switch operatoridan keyingi qavs ichida tilning konstruksiyasi nuqtai–nazaridan to‘g‘ri bo‘lgan ixtiyoriy ifodani ishlatish mumkin. Operator identifikatori o‘rnida ham ixtiyoriy operator yoki ifoda, hamda operator va ifodalarning ketma-ketligini ishlatish mumkin. Lekin bu yerda mantiqiy operatsiyalar yoki taqqoslash ifodalarini ishlatish mumkin emas.

1-misol.

```
switch(choice)
{
case 0:
 cout<< "zero!"<< endl;
 break;
case 1:
 cout<< "one!"<< endl;
 break;
case 2:
 cout<< "two!" <<endl;
 break;
```

```
default:
 cout<< "default!"<<endl;
}
```

2-misol

```
switch (choice)
{
case 0:
case 1:
case 2:
 cout< "Less than 3! "<< endl;
 break;
case 3:
 cout<< "Equals 3! " << endl;
 break;
default:
 cout<< "Greater than3 ! " << endl;}
```

Operator yoki ifodalardan keyin break operatori qo‘llanilmasa joriy case operatoridan keyingi case blokidagi barcha ifodalar bajariladi. Ko‘p hollarda bunday situatsiyada xatolik ro‘y beradi. Shuning uchun, break operatorini tushirib qoldirsangiz bu amalni tavsiflovchi mos izohni(komentariyni) yozishni unutmang.

switch operatorining qo‘llanilishi

```
#include <iostream>
using namespace std;
int main()
{
unsigned short int number;
cout<< "Enter a number between 1 and 5:";
cin>>number;
switch (number)
{
```

```
case 0: cout << "Small";
        break;
case 5: cout<< "Good job! \n";
case 4: cout << "Nice Pick! \n";
case 3: cout<< "Excellent! \n";
case 2: cout << "Master full! \n";
case 1: cout << "Incredible! \n";
        break;
default: cout << "Big! \n";
        break;
}
cout<< "\n\n";return 0;
}
```

Hatija:

Enter a number between 1 and 5: 3

Excellent!

Masterful!

Ineredible!

Enter a number between 1 and 5: 8

Big!

TAHLIL

Dastur oldin son kiritishni so‘raydi. Keyin esa kiritilgan son switch operatori orqali tekshiriladi. Agarda 0 kiritilgan bo‘lsa unga muvofiq ravishda ekranga kichik son xabari chiqariladi. Va undan keyin yozilgan break operatori switch konstruksiyasini bajarilishini yakunlaydi. Agarda 5 soni kiritilsa muvofiq xabar chiqariladi. Undan keyingi toki break so‘zigacha barcha satrlar ketma – ket bajariladi.

Agarda dasturga 5 dan katta son kiritilsa default bajariladi, ya’ni ekranga Big xabari chiqariladi.

switch operatori yordamida menyu komandalarini bajarish. for(;;) operatori orqali cheksiz sikllarni tashkil qilinishini oldingi mavzularda ko‘rib chiqqan edik.

Agarda bunday sikllar bajarilishini break operatori orqali to‘xtatmasak u cheksiz marta ishlaydi. Bu turdagi sikllar menyu komandalari bilan ishlashda qulaylik tug‘diradi. Bunda foydalanuvchi taklif etilgan komandalardan birini tanlaydi, keyin esa aniqlangan amal bajariladi va yana menyuga qaytiladi.

Cheksiz sikllarda hech qanday shart mavjud emas. Shuning uchun bunday sikldan faqatgina break operatori orqali chiqiladi.

Cheksiz siklga misol

```
#include <iostream>
using namespace std;
void DoTaskOne()
{
cout<<"Salom"<<endl;
};
void DoTaskMany(int n)
{
for(int i=0;i<n;i++) cout<<"Salom"<<endl;
};
int main ( )
{
bool exit=false;
 for (; ;)
 {
 int choice;
 cin>>choice;
 switch (choice)
 {
case (0): exit=true;
 break;
case (1): DoTaskOne ( );
 break;
```

```
case (2): DoTaskMany (2);
 break;
case (3): DoTaskMany (3);
}
if (exit) break;
}
return 0;
}
```

Shartsiz o‘tish operatori

goto operatori tarixi. Dasturlashni ilk davrlarida kichikroq hajmdagi va yetarlicha sodda dasturlar ishlatilar edi. Bunday dasturlarda sikllar nishonlardan, operatorlar va komandalar ketma – ketligidan hamda o‘tish operatoridan iborat edi.

C++ tilida nishon deb orqasidan ikki nuqta (:) yoziladigan identifikatorga aytiladi. Nishon doimo boshqaruv o‘tishi lozim bo‘lgan operatordan oldin o‘rnatiladi. Kerakli nishonga o‘tish uchun goto operatori qo‘llaniladi.

Bunda kalit so‘zdan keyin nishon nomi yoziladi. O‘tish operatorining ko‘rinishi:

goto <identifikator>. Bu operator identifikator bilan belgilangan operatorga o‘tish kerakligini

ko‘rsatadi.

Misol uchun goto A1;…; A1:y=5;

goto operatori yordamida sikl tashkil yetish.

```
#include <iostream>
using namespace std;
 int main()
{
int counter=0; // cchyotchikni initsializatsiya kilish
loop:
counter ++; // siklni boshlanishi
cout<<"counter: " <<counter<< "\n";
if(counter <5) //qiymatni tekshirish
```

```
goto loop; //sikl boshiga kaytish
cout<<"Tsikl tugadi. counter:"<<counter<<endl;
return 0;
}
```

Natija:

```
counter : 1
counter : 2
counter : 3
counter : 4
counter : 5
Tsikl tugadi. Counter: 5.
```

Nima uchun goto operatorini ishlatmaslik kerak. goto operatori orqali dasturning ixtiyoriy nuqtasiga borish mumkin. Lekin goto operatorining tartibsiz qo‘llanilishi bu dasturni umuman tushunarsiz bo‘lishiga olib keladi. Shuning uchun oxirgi 20 yillikda butun jahon bo‘yicha dasturlashni o‘rganuvchilarga qo‘yidagi fikr ta’kidlanib kelinmokda “Hech qachon goto operatorini ishlatmang’’.

goto operatorining o‘rnini bir muncha mukammalroq strukturaga ega bo‘lgan konstruksiyalar egalladi. Bular for, while va do…while operatorlari bo‘lib, ular goto operatoriga nisbatan ko‘prok imkoniyatlarga egadir. Lekin dasturlashda har qanday instrument to‘g‘ri qo‘llanilgandagina foydali bo‘lishi hisobga olinib ANSI komiteti C++ tilida goto operatorini qoldirishga qaror qildi. Albatta, bu bilan quyidagi hazil fikr ham paydo bo‘ldi: “Bolalar! Bu operatordan uy sharoitida foydalanish zararsizdir’’.

Mavzuga oid savollar

1. Butun sonli va haqiqiy tiplarni qanday farqi bor?
2. Unsigned tipining xossalarini ko‘rsating.
3. Tiplarni keltirish qoidalari.
4. unsigned short int va long int tiplarining o‘zaro farqi nimada?
5. Dastur ishiga “yaxshi” va “yomon” nomlangan o‘zgaruvchi qanday ta’sir qiladi?

6. Switch operatorining umumiy ko‘rinishi.
7. Sirpanish deb nimaga aytiladi?
8. Continue operatoridan nima uchun foydalaniladi?
9. Break operatoridan nima uchun foydalaniladi?
10. Goto operatori boshqarishni qayerga uzatadi?

7.5. C++ DASTURLASH TILIDA TAKRORLANUVCHI JARAYONLAR.

Reja:

1. while operatori orqali sikllarni tashkil etish.
2. Keyingi shartli do…while konstruksiyasining qo‘llanilishi.
3. Parametrik sikl operatori.
4. Ichki sikllar.

Tayanch iboralar: Sikl operatorlari. Oldingi shartli While takrorlash operatori. Keyingi shartli do-while takrorlash operatori. For takrorlash operatori. Uzish break operatori. Umumiy takrorlanish algoritmlari va ichma-ich takrorlanishlar. Continue operatori.

Shartli sikllar

Sikllarni tashkil etish. Har qanday dasturning strukturasi tarmoqlanish va sikllar to‘plamining kombinatsiyasidan iborat bo‘ladi. Qator masalalarni yechish uchun ko‘pincha bitta amalni bir necha marotaba bajarish talab qilinadi. Amaliyotda bu rekursiyalar va iterativ algoritmlar yordamida amalga oshiriladi. Iterativ jarayonlar – bu operatsiyalar ketma-ketligini zaruriy sonda takrorlanishidir.

while operatori orqali sikllarni tashkil etish. while operatori yordamida sikllarni tashkil etishda operatsiyalar ketma-ketligi siklning davom etish sharti «to‘g‘ri» bo‘lsa yoki 0 dan farqli bo‘lsagina uning navbatdagi operatsiyalari amalga oshiriladi.

while operatori quyidagi umumiy ko‘rinishga egadir:

while(ifoda) Operator

Agar dasturda while (1); satr qo‘yilsa bu dastur xech qachon tugamaydi.

Quyidagi dasturda counter o‘zgaruvchisi qiymati toki 5 ga teng bo‘lgunga qadar oshib borar edi.

while operatori yordamida siklni tashkil etish

```
#include <iostream>
```

```
using namespace std;
int main()
{
int counter=0; //Birlamchi qiymatni o‘zlashtirish
while(counter<5)//Sikl shartini tekshirish
{
counter ++;
cout << "counter :"<< counter << ". \n" ;
}
cout<<"Tsikl tugadi. Counter:"<<counter<<". \n";
return 0;
}
```

Natija:

counter : 1
counter : 2
counter : 3
counter : 4
counter : 5
Tsikl tugadi. Counter: 5.

while operatori orqali murakkab konstruksiyalarni tuzish. while operatori shartida murakkab mantiqiy ifodalarni ham qo‘llash mumkin. Bunday ifodalarni qo‘llashda && (mantiqiy ko‘paytirish), || (mantiqiy qo‘shish), hamda !(mantiqiy INKOR) kabi operatsiyalardan foydalaniladi. Quyida while operatori konstruksiyasida murakkabroq shartlarni qo‘yilishiga misol keltirilgan.

while konstruksiyasidagi murakkab shartlar.

```
#include <iostream>
using namespace std;
int main()
{
```

```
unsigned short kichik;
 unsigned long katta;
 const unsigned short MaxKichik=65535;
 cout << "Kichik sonni kiriting:";
 cin >> kichik;
 cout << "Katta sonni kiriting:";
 cin >> katta;
 cout << "kichik son:" << kichik << "…";
 //Xar bir iteratsiyada uchta shart tekshiriladi.
 while (kichik<katta && katta>0 &&
 kichik< MaxKichik )
 {
 if(kichik%5000==0) //Xar 5000 satrdan
 //keyin nuqta chikariladi
 cout<<". " ;
kichik++;
katta-=2 ;
}
cout<<"\n kichik son:"<<kichik<<" katta son:"
<<katta << endl ;
return 0 ;
}
```

Natija:

Kichik sonni kirit : 2

Katta sonni kirit : 100000

Kichik son : 2 ……

Kichik son :33335 katta son : 33334

TAHLIL

Dastur quyidagi mantiqiy o‘yinni ifodalaydi. Oldin ikkita son – kichik va katta kiritiladi. Undan so‘ng toki ular bir biriga teng bo‘lmaguncha, ya’ni «uchrashmaguncha» kichik son birga oshiriladi, kattasi esa ikkiga kamaytiriladi. O‘yinni maqsadi qiymatlar «uchrashadigan» sonni topishdir.

Qiymatlar kiritilgandan so‘ng siklni davom ettirishning quyidagi uchta sharti tekshiriladi:

kichik o‘zgaruvchisi qiymati katta o‘zgaruvchisi qiymatidan oshmasligi.

katta o‘zgaruvchisi qiymati manfiy va nolga teng emasligi

kichik o‘zgaruvchisi qiymati MaxKichik qiymatidan oshib ketmasligi

So‘ngra kichik soni 5000 ga bo‘lingandagi qoldiq hisoblanadi. Agarda kichik 5000 ga qoldiqsiz bo‘linsa bu operatsiyaning bajarilishi natijasi 0 ga teng bo‘ladi. Bu holatda hisoblash jarayonini vizual ifodasi sifatida ekranga nuqta chiqariladi. Keyin esa kichik qiymati bittaga oshiriladi, katta qiymati esa 2 taga kamaytiriladi. Sikl agarda tekshirish sharti tarkibidagi birorta shart bajarilmasa to‘xtatiladi.

do…while konstruksiyasi yordamida sikl tashkil etish. Ayrim hollarda while operatori yordamida sikllarni tashkil etishda uning tanasidagi amallar umuman bajarilmasligi mumkin. Chunki siklni davom etish sharti har bir iteratsiyadan oldin tekshiriladi. Agarda boshlang‘ich berilgan shart to‘g‘ri bo‘lmasa sikl tanasining birorta operatori ham bajarilmaydi.

while sikli tanasidagi amallar bajarilmay qolishi

```
#include <iostream>
using namespace std;
int main()
{
int counter;
cout << "How manu hellos ?:";
cin >> counter;
while (counter>0 )
{
cout << "Hello ! \n";
```

```
counter-- ;
}
cout<<"Counter is OutPut ;" << counter ;
return 0;
}
```

Hatija:

```
How manu hellos ? : 2
Hello !
Hello !
counter is OutPut : 0
How manu hellos ? : 0
counter is OutPut : 0
```

do…while konstruksiyasining qo‘llanilishi. do-while operatori umumiy ko‘rinishi quyidagicha:

```
do
Operator
while(ifoda)
```

do…while konstruksiyasida sikl sharti uning tanasidagi operatsiyalar bir marta bajarilgandan so‘ng tekshiriladi. Bu sikl operatorlarini hech bo‘lmaganda bir marta bajarilishini kafolatlaydi. To sharti «yolg‘on» bo‘lmaguncha yoki ifoda qiymati 0 bo‘lmaguncha sikl qaytariladi.

Quyida oldingi dasturda keltirilgan variantning bir oz o‘zgartirilgan shakli, ya’ni while operatori o‘rniga do…while konstruksiyasi qo‘llangan shakli keltirilgan.

do. . . while konstruksiyasining qo‘llanilishi

```
#include <iostream>
using namespace std;
int main()
{
int counter;
cout<<"How manu hellos ?";
```

```
cin >>counter;
do
{
cout << "hello \n";
counter--;
}
while(counter>0);
cout << "Counter is :" << counter <<endl;
return 0 ;
}
```

Hatija:

how manu hellos ? 2

hello

hello

Sounter is : 0

How manu hellos ? 0

Hello

Counter is: - 1

3. 2. Parametrik sikl operatori

for operatori. for operatori umumiy ko‘rinishi quyidagicha:

for(1-ifoda;2- ifoda; 3-ifoda)

Operator

Bu operator quyidagi operatorga mosdir.

1-ifoda;

while(2-ifoda) {

operator

3-ifoda

}

while operatori yordamida sikllarni tashkil etishda 3 ta zaruriy amallar: sikl o‘zgaruvchisiga boshlang‘ich qiymat berish, har bir iteratsiyada siklni davom etish

sharti bajarilishini tekshirish va sikl o‘zgaruvchisi qiymatini o‘zgartirishni bajarishimiz kerak.

while operatorining ishlatilishiga yana bir misol.

```
#include <iostream>
using namespace std;
int main()
{
int counter=0;
while (counter <5)
{
counter++ ;
cout << "Looping!";
}
cout << "\n Counter:" << counter << "\n";
return 0;
}
```

Natija:

Looping! Looping! Looping! Looping! Looping!

Counter: 5

for operatori siklni ishlashi uchun zarur bo‘ladigan uchta operatsiyani o‘zida birlashtiradi. Bu operatsiyalarni qisqacha quyidagicha harakterlash mumkin: boshlang‘ich qiymatni o‘zlashtirish, shartni tekshirish, sikl schyotchigini qiymatini oshirish. for operatori ifodasidagi qavsning ichida shu uchchala operatsiyani amalga oshiruvchi ifodalar yoziladi. Qavs ichidagi ifodalar nuqtali vergul orqali ajratiladi.

for siklining birinchi ifodasi sikl schyotchigiga boshlang‘ich qiymatni o‘zlashtiradi. Schyotchik – to‘g‘ridan–to‘g‘ri for siklida e’lon qilinadigan va qiymat o‘zlashtiriladigan butun sonli o‘zgaruvchidir. C++ da bu o‘rinda schyotchikka qiymat beradigan ixtiyoriy ifoda yozilishiga imkon berilgan. for siklining ikkinchi parametrida siklni davom etish sharti aniqlanadi. Bu shart while konstruksiyasining sharti

bajaradigan vazifani amalga oshiradi. Uchinchi parametrda esa sikl schyotchigi qiymatini o‘zgartiruvchi (oshiruvchi yoki kamaytiruvchi) ifoda yoziladi.

for siklining qo‘llanilishiga misol.

```
#include <iostream>
using namespace std;
int main()
{
int counter;
for (counter=0 ; counter<5; counter++ )
cout<< "Looping!";
cout<< "\n Counter:" << counter<< ". \n";
return 0;
}
```

Natija:

Looping! Looping! Looping! Looping! Looping!

Counter: 5

for operatori uchun murakkab ifodalarni berilishi. for sikli dasturlashning kuchli va qulay instrumentidir. for operatorida siklni o‘zaro bog‘liq bo‘lmagan parametrlar (boshlang‘ich qiymat o‘zlashtirish, bajarilish sharti va qadam) ni qo‘llanilishi sikl ishini boshqarishda juda yaxshi imkoniyatlarni ochib beradi.

for sikli quyidagi ketma–ketlikda ishlaydi.

3. Sikl schetchigiga boshlang‘ich qiymat o‘zlashtiriladi.
4. Siklni davom etish shartidagi ifoda qiymati hisoblanadi.
5. Agarda shart ifodasi true qiymat qaytarsa oldin sikl tanasi bajariladi, keyin esa sikl schyotchigi ustida berilgan amallar bajariladi.

Har bir iteratsiyada 2 – va 3 – qadamlar takrorlanadi.

Siklda bir nechta schyotchikni qo‘llanilishi. for siklining sintaksisi unda bir nechta o‘zgaruvchi - schetchikni qo‘llanilishiga, siklni davom etishini murakkab shartlarini tekshirishga va sikl schyotchiklari ustida ketma-ket bir nechta operatsiyani bajarilishiga imkon beradi.

Agarda bir nechta schyotchikka qiymat o‘zlashtirilsa yoki ular o‘rtasida bir nechta operatsiya bajarilsa, bu ifodalar vergul bilan ajratilgan holda ketma – ket yoziladi.

for siklida bir nechta schyotchikni qo‘llanilishi

```
#include <iostream>
using namespace std;
int main()
{
for (int i=0, j=0; i<3; i++, j++)
cout<< "i:" <<i<< "j:" <<j<< endl;
return 0;
}
```

Hatija:

```
i: 0          j: 0
i: 1          j: 1
i: 2      j: 2
```

for siklida nol parametrlarni ishlatilishi. for siklining ixtiyoriy parametri tushirib qoldirilishi mumkin. Nol parametrlar for siklini boshqa parametrlaridan nuqtali vergul (;) bilan ajratiladi. Agarda for siklini 1 – va 3 – parametrlarini tushirib qoldirsak, u xuddi while operatoridek qo‘llaniladi.

for siklining nol parametrlari

```
#include <iostream>
using namespace std;
int main()
{
int counter=0;
for ( ; counter<5 ; )
{
counter++;
cout <<" Looping!";
```

```
}
cout<< "\n Counter:" << counter<< ". \n";
return 0;
}
```

Natija:

Looping! Looping! Looping! Looping! Looping!

Counter: 5

TAHLIL

Bu siklni bajarilishi while siklini bajarilishiga o‘xshash tarzda amalga oshiriladi. Avval sounter o‘zgaruvchisiga qiymat o‘zlashtirilayapti. for siklida esa parametr sifatida faqatgina siklni davom etish shartini tekshirish ifodasi ishlatilgan. Sikl o‘zgaruvchisi ustida operatsiya ham tushirib qoldirilgan. Bu holda ushbu siklni quyidagicha ifodalash mumkin:

```
while(counter<5)
```

for siklining tanasi bo‘sh bo‘lgan holda qo‘llanilishi. Siklda for operatori orqali uning tanasiga hech qanday operator yozmasdan turib ham biror bir amalni bemalol bajarish mumkin. Bunda sikl tanasi bo‘sh satrdan iborat bo‘ladi.

For siklining tanasi bo‘sh bo‘lgan holda qo‘llanilishi.

```
#include <iostream>
using namespace std;
int main()
{
for (int i=0; i<5; cout<< "i" <<i++<<endl);
return 0;
}
```

Hatija:

```
i:      0
i:      1
i:      2
i:      3
```

i: 4

Ichki sikllar. Boshqa siklning ichida tashkil etilgan sikl ichki sikl deb aytiladi. Bu holda ichki sikl tashqi siklni har bir iteratsiyasida to‘liq bajariladi. Quyida matritsa elementlarini ichki sikl orqali to‘ldirilishi namoyish qilingan.

Ichki sikllar.

```
#include <iostream>
using namespace std;
int main()
{
int rows, columns;
char theChar;
cout << "How many rovs?";
cin >> rows;
cout << "How many columns?";
cin >> columns;
cout << "What character?";
cin>>theChar;
for ( int i=0; i<rows; i++)
{
for (int j=0; j<columns; j++)
cout << theChar;
cout<< "\n";
}
return 0;
}
```

Hatija:

How many rows? 4

How many columns? 12

What character? x

x x x x x x x x x x x x

```
x x x x x x x x x x x
x x x x x x x x x x x
x x x x x x x x x x x
```

for sikli schyotchigining ko‘rinish sohasi. ANSI ning yangi standarti bo‘yicha siklda e’lon qilingan o‘zgaruvchining ko‘rinish sohasi faqat sikl ichidangina iborat. Lekin ko‘pgina kompilyatorlar eski standartlarni ham qo‘llab – kuvvatlaydilar. Quyida keltirilgan dastur kodini kiritib o‘zingizning kompilyatoringiz yangi standartga mos kelish- kelmasligini tekshirishingiz mumkin.

```
#include <iostream>
using namespace std;
int main()
 {
for( int i = 0; i<5; i++ )
{
cout << " i: " << i << endl;
}
i=7; // i ko‘rinish soxasi chegarasidan tashqarida
return 0;
}
```

Agarda bu dastur xatoliksiz kompilyatsiya qilinsa demak u ANSI ning yangi standartini qo‘llab - quvvatlamaydi. Yangi standartga muvofiq kompilyatorlar i=7 ifoda uchun xatolik xaqida xabar berishi kerak. Dastur kodiga ozgina o‘zgartirish kiritilganda so‘ng dastur barcha kompilyatorlar uchun xatoliksiz ishlaydi.

```
#include <iostream>
using namespace std;
int main ()
int i;
for ( int i = 0; i<5; i++ )
{
cout << " i: " << i << endl ;
```

```
}
i=7; //Endi i barcha kompilyatorlar tomonidan
//xatoliksiz qabul kilinishi mumkin.
return 0;
}
```

Mavzuga oid savollar

1. While operatorining umumiy ko‘rinishi.
2. While operatorida murakkab shart qanday beriladi?
3. Do while operatori While operatoridan qanday farq qiladi?
4. For operatorining umumiy ko‘rinishi.
5. Ichki sikllar qanday tashkil etiladi?

7.6. C++ DASTURLASH TILIDA FUNKSIYALAR.

Reja:

1. Funsiya tushunchasi va uning qo’llanilishi
2. Funksiyalarni ta’riflash va bajarish.
3. Global va lokal o‘zgaruvchilar
4. Funksiyadan foydalanish xususiyatlari
5. Funksiyaga parametrlar uzatish

Tayanch iboralar: Funsiyalar haqida tushuncha va ularni yaratish. Funksiyalarrning tuzilishi. Funksiya parametrlari. Lokal va global o’zgaruvchilar.

Funksiya tushunchasi. Obektlarga mo‘ljallangan dasturlashda asosiy e’tibor funksiyaga emas, balki obektga qaratilgan bo‘lsada dasturlarda funksiya markaziy komponentligicha qoldi. Funksiya bu ma’nosiga ko‘ra dastur osti bo‘lib, u ma’lumotlarni o‘zgartirishi va biror bir qiymat qaytarishi mumkin. C++ da har bir dastur hech bo‘lmaganda bitta main() funksiyasiga ega bo‘ladi. main() funksiyasi dastur ishga tushirilishi bilan operatsion sistema tomonidan avtomatik chaqiriladi. Boshqa funksiyalar esa u tomonidan chaqirilishi mumkin.

Har bir funksiya o‘zining nomiga egadir. Qachonki, dasturda bu nom uchrasa boshqaruv shu funksiya tanasiga o‘tadi. Bu jarayon funksiyani chaqirilishi (yoki

funksiyaga murojaat qilish) deb aytiladi. Funksiya ishini tugatgandan so‘ng dastur o‘z ishini funksiya chaqirilgan qatorning keyingisidan boshlab davom ettiradi.

Qachonki, dastur biror bir xizmat ko‘rsatuvchi amallarni bajarilishiga ehtiyoj sezsa kerakli funksiyani chaqiradi. Bu operatsiya bajarilgandan keyin esa dastur o‘z ishini funksiya chaqirilgan joydan boshlab davom ettiradi. Bu g‘oya quyidagi dasturda namoyish etilgan.

Funksiyani chaqirilishiga misol

```
#include <iostream>
using namespace std;
//Namoyishfunktsiyasi funksiyasi ekranga
// axboriy ma’lumot chikaradi.
void Namoyishfunktsiyasi()
{
cout<< "Namoyishfunktsiyasi chaqirildi\n";
}
//main() funksiyasi oldin axborot chikaradi va
// Namoyishfunktsiyasi funksiyasini chakiradi
// Keyin yana namoyish funksiyasini chakiradi
int main()
{
 cout << "Bu main() funktsiyasi \n";
Namoyishfunktsiyasi();
cout << "main() funktsiyasiga qaytildi\n";
return 0;
}
```

Natija:

```
Bu main() funktsiyasi
 Namoyish funktsiyasi chaqirildi
   main() funktsiyasiga qaytildi
```

Funksiyalarning qo‘llanilishi. Funksiyalar yo void tipidagi, yo boshqa biror bir tipdagi qiymat qaytaradi. Ikkita butun sonni qo‘shib, ularning yig‘indisini qaytaradigan funksiya butun qiymat qaytaruvchi deyiladi. Faqatgina qandaydir amallarni bajarib, hech qanday qiymat qaytarmaydigan funksiyaning qaytaruvchi tipi void deb e’lon qilinadi.

Funksiya sarlavha va tanadan iboratdir. Funksiya sarlavhasida uning qaytaradigan tipi, nomi va parametrlari aniqlanadi. Parametrlar funksiyaga qiymat uzatish uchun ishlatiladi. Masalan, agar funksiya ikki sonni qo‘shishga mo‘ljallangan bo‘lsa u holda bu sonlarni funksiyaga parametr qilib berish kerak. Bunday funksiyaning sarlavhasi quyidagicha bo‘ladi:

int Qushish(int a,int b)

Parametr – bu funksiyaga uzatiladigan qiymat tipini e’lon qilishdir. Funksiya chaqirilganda unga uzatiladigan qiymat argument deb aytiladi. Ko‘pchilik dasturchilar bu ikkala tushunchani sinonim sifatida qarashadi. Ba’zilar esa bu terminlarni aralashtirishni noprofessionallik deb hisoblaydi. Mavzularda ikkala terminni bir xil ma’noda kelgan.

Funksiya tanasi ochiluvchi figurali qavs bilan boshlanadi va u bir necha qatordan iborat bo‘lishi mumkin. (Funksiya tanasida hech qanday satr bo‘lmasligi ham mumkin). Satrlardan keyin esa yopiluvchi figurali qavs keladi. Funksiyaning vazifasi uning satrlarida berilgan dasturiy kodlar bilan aniqlanadi. Funksiya dasturga return operatori orqali qiymat qaytaradi. Bu operator funksiyadan chiqish ma’nosini ham anglatadi. Agarda funksiyaga chiqish operatorini (return) qo‘ymasak funksiya satrlarini tugashi bilan u avtomatik void tipidagi qiymatni qaytaradi. Funksiya qaytaradigan tip uning sarlavhasida ko‘rsatilgan tip bilan bir xil bo‘lishi lozim.

Quyidagi dsturda ikkita butun sonli parametrni qabul qiluvchi va butun sonli qiymat qaytaruvchi funksiya namoyish qilingan.

Oddiy funksiyani ishlatishga misol

```
#include <iostream>
using namespace std;
```

```
int Qushish(int x,int y)
{
 cout<<x<<" va "<<y<<"sonlarini Qushish()"
 << "funktsiyasi orqali qushdik\n";
 return (x + y) ;
}
 int main()
{
 cout <<"Biz main() funktsiyasidamiz!";
 int a, b, c;
 cout << "Ikkita son kiriting:";
 cin >> a;
 cin >> b;
cout <<"\nQushish()funktsiyasi chaqirildi"<<endl;
 c = Qushish(a,b);
 cout <<"\n main() funktsiyasiga qaytildi. "<<endl;
 cout << "c ning qiymati " <<c<<" ga teng" << endl;
 cout << " \n Ishni tugatish. . . \n";
 return 0;
}
```

Natija:

```
 Biz main() funktsiyasidamiz!
 Ikkita son kiriting: 5 3
 Qushish()funktsiyasi chaqirildi
 5 va 3 sonlarini Qushish()funktsiyasi orqali qushdik
 main() funktsiyasiga qaytildi.
 c ning qiymati 8 ga teng
 Ishni tugatish …
```

TAHLIL

Qushish() funksiyasi ikkita butun sonli parametrni qabul qiladi va butun sonli qiymat qaytaradi. Dasturning ekranga birinchi xabarni chiqaradi. Keyin foydalanuvchiga ikkita sonni kiritish haqida xabar beriladi. Foydalanuvchi bo‘shlik belgisi (probel) orqali ajratgan holda ikkita son kiritadi. Keyin esa Enter tugmasini bosadi. Bosh main() funksiyasi Qushish()funksiyasiga foydalanuvchi tomonidan kiritilgan ikkita sonni parametr sifatida uzatadi.

Dastur boshqaruvi Qushish() funksiyasiga o‘tadi. a va b parametrlar ekranga chiqariladi va qo‘shiladi. Funksiya natijasini qaytaradi va o‘z ishini yakunlaydi.

Xotiraning taqsimlanishi. Dastur ishlay boshlashi bilan operatsion sistema (Dos yoki Microsoft Windows) kompilyatorning talabiga muvofiq xotira sohasidan joy ajratadi. C++ dasturchisi global nomlar fazosi, erkin taqsimlanuvchi xotira, registr, segmentli xotira va stek tushunchalarini bilishi lozim.

Global nomlar fazosida global o‘zgaruvchilar saqlanadi. Global nomlar fazosi va erkin taqsimlanuvchi xotira haqida keyingi mavzularda batafsilroq to‘xtaladi.

Registr xotiraning maxsus sohasi bo‘lib, uning asosiy vazifasi ichki yordamchi funksiyalarni ishlashini tashkil etishdir.

Dasturning o‘zi dastur operatorlari ikkilik formatda saqlanishi uchun ajratilgan kompyuter xotirasida saqlanadi.

Stek – bu dasturda funksiya chaqirilganda undagi ma’lumotlar uchun talab qilinadigan xotiraning maxsus sohasidir. Uning stek deb atalishiga sabab «oxirgi kelgan – birinchi ketadi» prinsipi asosida ishlashidir. Haqiqatan ham, funksiyalarni chaqirilishi xuddi shu prinsip asosida amalga oshiriladi. Agarda bir funksiya ikkinchisini chaqirsa, ikkinchi funksiya o‘z ishini tugatgandan so‘ng birinchi funksiya o‘z ishini yakunlaydi, ya’ni oxirgi chaqirilgan funksiya birinchi ishini tugatadi.

Funksiyalarni ta’riflash va bajarish

Funksiyaning ta’rifi. Funksiya ta’rifida funksiya nomi, tipi va formal parametrlar ro‘yxati ko‘rsatiladi. Formal parametrlar nomlaridan tashqari tiplari ham ko‘rsatilishi shart. Formal parametrlar ro‘yxati funksiya signaturasi deb ham ataladi.

Funksiya ta’rifi umumiy ko‘rinishi quyidagichadir:

Funksiya tipi funksiya nomi(formal_parametrlar_ta’rifi)

Formal parametrlarga ta’rif berilganda ularninga boshlang‘ich qiymatlari ham ko‘rsatilishi mumkin.

Funksiya qaytaruvchi ifoda qiymati funksiya tanasida **return** <ifoda> ; operatori orkali ko‘rsatiladi.

```
 int Yuza int uzunlik,int kenglik)
 { - ochiluvchi figurali kavs.
// funktsiya tanasi
 return (uzunlik*kenglik);
} - yopiluvchi figurali kavs.
```

Funksiyaning bajarilishi. Funksiya chaqirilganda unda ko‘rsatilgan amallar ochiluvchi figurali qavsdan ({) keyingi birinchi ifodadan boshlab bajariladi. Funksiya tanasida if shartli operatoridan foydalanib tarmoqlanishni ham amalga oshirish mumkin.

Funksiya o‘z tanasida boshqa funksiyalarni va hatto o‘z – o‘zini ham chaqirishi mumkin.

Qaytariladigan qiymatlar, parametrlar va argumentlar. Funksiya biror bir qiymat qaytarishi mumkin. Funksiyaga murojaat qilingandan so‘ng u qandaydir amallarni bajaradi, keyin esa u o‘z ishining natijasi sifatida biror bir qiymat qaytaradi. Bu qaytariladigan qiymat deb ataladi va bu qiymatning tipi oldindan e’lon qilinishi lozim. Quyidagi yozuvda myFunction funksiyasi butun sonli qiymat qaytaradi.

```
int myFunction()
```

Funksiyaga ham o‘z navbatida biror bir qiymat uzatish mumkin. Uzatiladigan qiymatlar funksiyaning parametrlari deb aytiladi.

```
int myFunction (int Par, float ParFloat);
```

Bu funksiya nafaqat butun son qaytaradi, balki parametr sifatida butun va haqiqiy sonli qiymatlarni qabul qiladi.

Parametrda funksiya chaqirilganda unga uzatiladigan qiymat tipi aniqlanishi lozim. Funksiyaga uzatiladigan haqiqiy qiymatlar argumentlar deb aytiladi.

```
int theValueReturned=myFunction(5,6,7);
```

Bu yerda theValueReturned nomli butun sonli o‘zgaruvchiga argument sifatida 5, 6 va 7 qiymatlar berilgan myFunction funksiyasining qaytaradigan qiymati o‘zlashtirilayapti. Argument tiplari e’lon qilingan parametr tiplari bilan mos kelishi lozim.

```
#include <iostream>
using namespace std;
float min(float a=0. 0, float b)
{
if (a<b) return a;
return b;
}
int main()
{
cout<<min(5,-6);
}
```

Funksiya ta’rifida formal parametrlar initsializatsiya kilinishi, ya’ni boshlang‘ich qiymatlar ko‘rsatilishi mumkin.

Funksiyaga murojaat qilinganda biror xaqiqiy parametr ko‘rsatilmasa, uning o‘rniga mos formal parametr ta’rifida ko‘rsatilgan boshlang‘ich qiymat ishlatiladi.

Misol uchun:

```
include <iostream. h>
float min(float a=0. 0, float b)
{
if (a<b) return a;
return b;
}
int main()
{
int y=6,z; z=min(y);
cout<<z;
```

```
}
```

Agar funksiya xech kanday qiymat qaytarmasa uning tipi void deb ko‘rsatiladi.

Misol uchun:

```
void print()
{
cout<<"\n Salom! ";
};
```

Bu funksiyaga print(); shaklida murojjat kilish ekranga Salom! yozilishiga olib keladi.

Funksiya prototipi

Funksiyani e’lon qilish va aniqlanishi. Dasturda funksiyani qo‘llash uchun, oldin uni e’lon qilish, keyin esa aniqlash lozim. Funksiyani e’lon qilishda kompilyatorga uning nomi, qaytaradigan qiymatlari va parametrlari haqida xabar beriladi. Funksiyani aniqlanishidan kompilyator uning qanday ishlashi haqida ma’lumot oladi. Dasturdagi biror funksiyani oldindan e’lon qilmasdan turib chaqirish mumkin emas. Funksiyani e’lon qilinishi uning prototipini (timsolini) hosil qilish deb ataladi.

Funksiyani e’lon qilish. Funksiyani e’lon qilishning uch xil usuli mavjud:

- Funksiya prototipi faylga yoziladi, keyin esa u #include ifodasi qo‘llanilib kerakli dasturga qo‘shib qo‘yiladi.
- Funksiya ishlatiladigan faylga uning prototiplari yoziladi.
- Funksiya uni chaqiruvchi ixtiyoriy funksiyadan oldin yoziladi va bu holda funksiya e’lon qilinishi bilan bir vaqtda aniqlanadi.

Funksiyani prototipini tuzmasdan turib ham uni ishlatishdan oldin e’lon kilish mumkin. Lekin, dasturlashning bunday uslubi quyidagi uchta sababga ko‘ra yaxshi hisoblanmaydi.

Birinchidan, funksiyani faylda ko‘rsatilgan tartibda yozish, uni dastur ishlatilishida o‘zgartirish jarayonini murakkablashtiradi.

Ikkinchidan, quyidagi ko‘p uchraydigan holatni amalga oshirish imkoniyati mavjud emas.

A() funksiya V() funksiyani chaqirsin. Xuddi shuningdek, dasturning biror bir qismida V() funksiya A() funksiyani chaqirsin. U holda biz A() funksiyani V() funksiya aniqlanmasdan turib ishlata olmaymiz.

Bu holda hech bo‘lmaganda bitta funksiya oldindan e’lon qilinishi lozim.

Uchinchidan, funksiyaning prototiplari dasturni tekshirish jarayonida juda yaxshi ishlatiladi. Agarda funksiya prototipi aniqlangan bo‘lsa unga muvofiq funksiya aniqlangan parametrini qabul qiladi yoki aniqlangan biror bir qiymat qaytaradi. Dasturda e’lon qilingan prototipga muvofiq bo‘lmagan funksiyani ishlatishga urinsak kompilyator bu xatolikni kompilyatsiya jarayonini o‘zidayoq aniqlaydi va dastur ishlashida turli noxush xatoliklarni ro‘y berishining oldini oladi.

Prototip ta’rifi. Ko‘pgina ichki qurilgan funksiyalarning prototiplari dasturga #include kalit so‘zi yordamida qo‘shiladigan fayl-sarlavhasida yoziladi. Foydalanuvchi tomonidan tuziladigan funksiyalar uchun esa ularning mos prototiplarini dasturga qo‘shish dasturchi tomonidan bajarilishi lozim.

Funksiyaning prototipi nuqtali vergul orqali tugaydigan funksiyani qaytaradigan qiymati va signaturasidan iboratdir. Funksiyani signaturasi deb uning nomi va parametrlar ro‘yxati tushiniladi.

Formal parametrlar ro‘yxati barcha parametrlar va ularning tiplarini ifodalaydi.

Funksiyaning prototipi hamda aniqlanishidagi uning qaytaradigan qiymati tipi va signaturasi mos bo‘lishi lozim. Agarda bunday mutanosiblik bo‘lmasa kompilyator xatolik haqida xabar beradi. Funksiya prototipida parametr nomlarisiz tiplarni ko‘rsatilishi yetarlidir. Masalan, quyida keltirilgan misol to‘g‘ridir:

```
long Area(int, int)
```

Bu prototip ikkita butun sonli parametrni qabul qilib, long tipidagi qiymat qaytaradigan Area() nomli funksiyani e’lon qiladi. Prototipning bunday yozilishi unchalik yaxshi variant emas. Prototipga parametrlarning nomlarini qo‘shilishi uni tushunarliroq bo‘lishini ta’minlaydi.

Har bir funksiyaning qaytaradigan qiymati tipi aniqlangan bo‘ladi. Agarda u ochiq aniqlanmagan bo‘lsa avtomatik ravishda int tipini qabul qiladi.

Agar dasturda funksiya ta’rifi murojaatdan keyin berilsa, yoki funksiya boshqa faylda joylashgan bo‘lsa, murojjatdan oldin shu funksiyaning prototipi joylashgan bo‘lishi kerak.

Funksiyani e’lon qilinishi, aniqlanishi va ishlatilishi

```
#include <iostream>
using namespace std;
//funksiya prototipi
int Yuza(int uzunlik, int kenglik);
int main()
{
int YerUzunligi;
int YerKengligi;
int YerMaydoni;
cout<< "\n Yerning uzunligi necha metr? ";
cin >> YerUzunligi;
cout<< "\n Yerning kengligi necha metr? ";
cin >> YerKengligi;
YerMaydoni=Yuza(YerUzunligi,YerKengligi);
cout << "\n Yer maydoni yuzasi ";
cout<<YerMaydoni;
cout << " kvadrat metr \n";
return 0;
}
int Yuza(int YerUzunligi,int YerKengligi)
{
return YerUzunligi* YerKengligi;
}
```

Natija:

```
Yerning uzunligi necha metr? 200
Yerning kengligi necha metr? 100
```

Yer maydoni yuzasi 20000 kvadrat metr

Funksiyalar va o‘zgaruvchilar

Lokal o‘zgaruvchilar. Funksiyaga qiymatlar uzatish bilan birga uning tanasida o‘zgaruvchilarni e’lon qilish ham mumkin. Bu lokal o‘zgaruvchilar orqali amalga oshiriladi. Qachonki dasturni bajarilishi funksiyadan asosiy qismga qaytsa, bu funksiyadagi lokal o‘zgaruvchilar xotiradan o‘chiriladi.

Lokal o‘zgaruvchilar xuddi boshqa o‘zgaruvchilar kabi aniqlanadi. Funksiyaga beriladigan parametrlarni ham lokal o‘zgaruvchilar deb atash mumkin va ularni funksiya tanasida aniqlangan o‘zgaruvchilar kabi ishlatish mumkin.

Funksiya lokal o‘zgaruvchilari va parametrlarining qo‘llanilishi

```
#include <iostream>
using namespace std;
float Almashtirish(float);
int main()
{
 float TempFer;
float TempCel;
cout <<" Feringait bo`yicha temperaturani";
cout<< "kiriting:";
cin >> TempFer;
TempCel = Almashtirish(TempFer);
cout << "\n Bu temperatura selziy shkalasi";
cout<< "bo`yicha: ";
cout << TempCel << endl;
return 0 ;
}
float Almashtirish(float TempFer)
{
float TempCel;
TempCel=((TempFer-32)*5)/9;
```

```
return TempCel;
};
```

Natija:

Feringait bo`yicha temperaturani kiriting: 50

Bu temperatura selziy shkalasi bo`yicha: 10

Global o‘zgaruvchilar. main() funksiyasida aniqlangan o‘zgaruvchilar dasturdagi barcha funksiyalar uchun murojaat qilishga imkonli va ko‘rinish sohasiga ega hisoblanadi. Bunday o‘zgaruvchilar dasturdagi funksiyalar uchun global o‘zgaruvchilar deyiladi.

Global o‘zgaruvchi nomi bilan funksiya ichida nomlari ustma-ust tushadigan lokal o‘zgaruvchilar faqatgina joriy funksiyaning ichidagina global o‘zgaruvchining qiymatini o‘zgartiradi. Lekin global o‘zgaruvchi funksiya o‘z ishini tugatgach u chaqirilishidan oldingi qiymatini saqlab qoladi, ya’ni funksiya tanasida e’lon qilingan lokal o‘zgaruvchi funksiyaning ichida global o‘zgaruvchini yashiradi xolos. Bunda lokal o‘zgaruvchi alohida hosil qilinadi va funksiya ishlash vaqtida global va lokal o‘zgaruvchilarning nomlari bir xil bo‘lsa faqatgina lokal o‘zgaruvchi ustida amallar bajariladi. Global o‘zgaruvchi esa funksiyaning bajarilishi davomida oldingi qiymatini saqlab turadi.

Global va lokal o‘zgaruvchilarning qo‘llanishi

```
#include <iostream>
using namespace std;
void MeningFunktsiyam() ; // prototip
int x = 5, y = 7; // global o‘zgaruvchilar
int main()
{
cout << "main()dagi x ning qiymati:"<<x<<"\n";
cout<<"main()dagi y ning qiymati y:"<<y<<"\n";
MeningFunktsiyam();
cout << "MeningFunktsiyam()funktsiyasi"<<"ishini tugatdi! \n";
cout<<"main()dagi x ning qiymati:"<<x<<"\n";
```

```
cout<<"main()dagi y ning qiymati:"<<y<<"\n";
 return 0;
 }
void MeningFunktsiyam()
{
int y = 10;
cout<<"MeningFunktsiyam()dagi"<<"x:" << x <<"\n";
cout<<"MeningFunktsiyam()dagi"<<"y:"<<y<<"\n";
}
```

Hatija:

```
main()dagi x ning qiymati: 5
main()dagi y ning qiymati: 7
MeningFunktsiyam()dagi x: 5
MeningFunktsiyam()dagi y: 10
MeningFunktsiyam()funktsiyasiishini tugatdi!
main()dagi x ning qiymati: 5
main()dagi y ning qiymati: 7
```

Lokal o‘zgaruvchilar haqida batafsilroq ma’lumot. Funksiyani ichida aniqlangan o‘zgaruvchilar lokal ko‘rinish sohasiga ega deyiladi. Yuqorida aytib o‘tilganidek, bu o‘zgaruvchilarni faqatgina funksiyaning ichidagina qo‘llash mumkinligini anglatadi. C++da o‘zgaruvchilarni nafaqat dasturning boshida balki ixtiyoriy joyda aniqlash mumkin. Agarda o‘zgaruvchi funksiya tanasidagi biror bir blok ichida aniqlangan bo‘lsa, bu o‘zgaruvchi faqatgina shu blok ichidagina ta’sirga ega bo‘lib butun funksiyaning ichida ko‘rinish sohasiga ega bo‘lmaydi.

Lokal o‘zgaruvchini ko‘rinish sohasi

```
#include <iostream>
using namespace std;
void MeningFunktsiyam();
int main()
```

```
{
 int x=5;
cout<<"\n main()dagi x ning qiymati:"<<x;
MeningFunktsiyam();
cout<<"\n main()dagi x ning qiymati:"<< x;
 return 0;
}
void MeningFunktsiyam()
{
int x = 8;
cout<<"\n MeningFunktsiyam()dagi";
cout<<"local x ning qiymati: "<<x<<endl;
{
cout <<"\n MeningFunktsiyam() ";
cout<<"funktsiyasi blokidagi x ning qiymati";
cout<< " x:"<<x;
int x = 9;
cout<<"\n Blok ichida aniqlangan";
cout<< "x ning qiymati:"<<x;
}
cout<<"\n MeningFunktsiyam()dagi";
cout<< "blockdan tashqarisida x ning qiymati:";
cout<<x<<endl;
}
```

Hatija:

main()dagi x ning qiymati: 5

MeningFunktsiyam()dagi local x ning qiymati: *8*

MeningFunktsiyam() funktsiyasi blokidagi x ning qiymati: 8

Blok ichida aniqlangan x ning qiymati: 9

MeningFunktsiyam()dagi blockdan tashqarisida x ning qiymati: 8

main()dagi x ning qiymati: 5

Funksiyadan foydalanish xususiyatlari

Funksiyada ishlatiladigan operatorlar. Funksiyada ishlatiladigan operatorlar soni va turiga hech qanday chegara yo‘q. Funksiya ichida istalgancha boshqa funksiyalarni chaqirish mumkin bo‘lsada, lekin funksiyalarni aniqlash mumkin emas.

C++ tilida funksiyaning hajmiga hech qanday talab qo‘yilmasa ham uni kichikroq tarzda tuzgan ma’quldir. Har bir funksiya tushunish oson bo‘ladigan bitta masalani bajarishi lozim. Agarda funksiya hajmi kattalashayotganligini sezsangiz, bu funksiyani bir nechta kichikroq funksiyalarga ajratish haqida o‘ylashingiz kerak.

Funksiya argumentlari. Funksiyaning argumentlari turli tipda bo‘lishi mumkin. Shuningdek, argument sifatida C++ tilidagi biror bir qiymat qaytaradigan o‘zgarmaslar, matematik va mantiqiy ifodalar va boshqa ixtiyoriy funksiyalarni berish mumkin.

Misol sifatida bizda biror bir qiymat qaytaruvchi double(), triple(), square() va cube() funksiyalari berilgan bo‘lsin. Biz quyidagi ifodani yozishimiz mumkin:

Answer=(double(triple(square(my Value))))

Bu ifodada myValue o‘zgaruvchisini qabul qilinadi va u cube() funksiyasiga argument sifatida uzatiladi.

cube() funksiyasi qaytargan qiymat esa square() funksiyasiga argument cifatida uzatiladi. square() funksiyasi qiymat qaytargandan keyin, buning qiymati o‘z navbatida triple() fuknsiyasiga argument sifatida beriladi. triple() funksiyasining qiymati esa double() funksiyasiga argument qilib beriladi va u Answer o‘zgaruvchisiga o‘zlashtiriladi.

Parametrlar bu lokal o‘zgaruvchilardir. Funksiyaga uzatilgan har bir argument, bu funksiyaga nisbatan lokal munosabatda bo‘ladi. Funksiyani bajarilishi jarayonida argumentlar ustida bajarilgan o‘zgartirishlar funksiyaga qiymat sifatida berilgan o‘zgaruvchilarga ta’sir qilmaydi.

Funksiyaga parametrlarni qiymat sifatida uzatish

```
#include <iostream>
using namespace std;
```

```
void Almashtirish(int x, int y);
int main()
{
int x = 5, y =10;
cout << "Main(). Almashtirishdan oldin,x:";
cout<<x <<" y:" << y <<"\n";
Almashtirish(x,y);
cout<<"Main(). Almashtirishdan keyin, x:";
cout<<x<< "y:" <<y<< "\n";
return 0;
}
void Almashtirish(int x, int y)
{
int temp;
cout<<"Almashtirish(). Almashtirishdan ";
cout<<"oldin, x: "<<x<<" y: "<<y<< "\n";
temp = x;
x = y;
y = temp;
cout<<"Almashtirish(). Almashtirishdan ";
cout<<"keyin, x: "<<x<<" y: "<<y<< "\n";
}
```

Hatija:

Main(). Almashtirishdan oldin,x: 5 y:10

Almashtirish(). Almashtirishdan oldin, x:5 y:10

Almashtirish(). Almashtirishdan keyin, x:10 y:5

Main(). Almashtirishdan keyin, x:5 y:10 x:

Funksiyaning qaytaradigan qiymatlari. Funksiya yo biror bir real qiymatni, yo kompilyatorga hech qanday qiymat qaytarilmasligi haqida xabar beruvchi void tipidagi qiymatni qaytaradi.

Funksiyani qiymat qaytarishi uchun return kalitli so‘zidan foydalaniladi. Bunda oldin return kalitli so‘zi, keyin esa qaytariladigan qiymat yoziladi. Qiymat sifatida esa o‘zgarmaslar kabi butun bir ifodalarni ham berish mumkin. Masalan:

```
return 5 ;
return (x > 5) ;
return (MyFunction()) ;
```

MyFunction() funksiyasi biror bir qiymat qaytarishidan kelib chiqsak, yuqoridagi barcha ifodalar to‘g‘ri keltirilgan. return(x>5) ifodasi esa x 5 dan katta bo‘lsa true, kichik yoki teng bo‘lsa false mantiqiy qiymatini qaytaradi.

Agarda funksiyada return kalit so‘zi uchrasa undan keyingi ifoda bajariladi va uning natijasi funksiya chaqirilgan joyga uzatiladi. return operatori bajarilgandan keyin dastur funksiya chaqirilgan satrdan keyingi ifodaga o‘tadi. return kalitli so‘zidan keyingi funksiya tanasidagi operatorlar bajarilmaydi.

Funksiya bir nechta return operatorlarini o‘zida saqlashi mumkin.

Bir nechta return operatorini qo‘llanilishi

```
#include <iostream>
using namespace std;
int IkkigaKupaytirish(int KupaytSon);
int main()
{
int natija=0;
int input;
cout << "Ikkiga ko`paytiriladigan sonni";
cout<< "kiriting(0 dan 10000 gacha):";
cin >> input;
cout << "\n IkkigaKupaytirish() funktsiyasi";
cout<< "chaqirilishidan oldin\n";
cout<<"Kiritilgan qiymat:" <<input;
cout<<"Ikkilangani:"<<natija<< "\n";
natija = IkkigaKupaytirish(input);
```

```
cout<<"\nIkkigaKupaytirish() funktsiyasidan";
cout<<"qaytgandan so`ng. . . \n";
cout<<"Kiritilgan qiymat:" <<input;
cout<<"Ikkilangani:"<<natija<< "\n";
return 0;
}
int IkkigaKupaytirish(int original)
{
if (original <= 10000)
return original*2;
else
return -1;
cout<< "Siz bu satrga o`ta olmaysiz!\n";
}
```

Natija:

```
Ikkiga ko`paytiriladigan sonni kiriting (0 dan 10000 gacha ): 9000
IkkigaKupaytirish() funktsiyasi chaqirilishidan oldin
Kiritilgan qiymat: 9000 Ikkilangani: 0
Ikkiga_kupaytirish() funktsiyasidan qaytgandan so`ng
Kiritilgan qiymat:9000 Ikkilangani: 18000
Ikkiga ko`paytiriladigan sonni kiriting (0 dan 10000 gacha ): 11000
IkkigaKupaytirish() funktsiyasi chaqirilishidan oldin
Kiritilgan qiymat: 11000 Ikkilangani: 0
Ikkiga_kupaytirish() funktsiyasidan qaytgandan so`ng
Kiritilgan qiymat:11000 Ikkilangani: -1
```

Inlayn funksiyalar. Dasturda funksiya ta’riflanganda kompilyator funksiya kodini bir marta mashina kodiga o‘tkazadi va dasturga ma’lumotlarni stekka joylovchi instruksiyalar qo‘shadi. Argumentlarni stekka qo‘shish, funksiyaga o‘tish va qaytish mashina vaqtini oladi.

C++ kompilyatori *inline,* so‘zini uchratsa bajariluvchi faylga har bir funksiyaga murojaat o‘rniga funksiya operatorlarini qo‘yadi. Shunday qilib dastur effektivligini oshirish mumkindir.

Misol:

```
#include <iostream>
using namespace std;
inline int min(float a, float b)
{
if(a<b) return a; else return b;
}
int main()
{
int x=5,y=6,z;
z=min(x,y);
cout<<"z="<<z;
return 0;
}
```

Natija

Z=5

BC++ kompilyatorida inlayn funksiya dasturga joylashtirilishi uchun rekursiv bulmasligi kerak va for, while, do,switch, goto operatorlari ishlatilmagan bo‘lishi kerak.

Funksiyaga parametrlar uzatish

Qiymat bo‘yicha uzatish. Parametrlar qiymatini funksiyaga uzatishning ikkita usuli mavjud: adres bo‘yicha va qiymati bo‘yicha.

C++ tilida chaqirilgan funksiya chaqiruvchi funksiyadagi o‘zgaruvchi qiymatini o‘zgartira olmaydi. U faqat o‘zining vaqtinchalik nusxasini o‘zgartirishi mumkin xolos.

Chaqirayotgan va chaqirilayotgan funksiyalar parametrlar orqali axborot almashadi

Qiymati bo‘yicha uzatishda quyidagi amallar bajariladi:

1. faktik parametrlar o‘rnida turgan ifodalar qiymatlari hisoblanadi;
2. funksiyaning formal parametrlari uchun stekda xotira ajratiladi;
3. hisoblangan qiymatlar formal parametr uchun ajratilgan adres bo‘yicha yoziladi, bunda turlarning o‘zaro muvofiqligi tekshiriladi hamda, zarurat tug‘ilganda, ular qayta o‘zgartiriladi.

Misol:

```
double square(double a, double b, double c);
{
//funktsiya a, b, c uzunlikdagi tomonlarga ega bo‘lgan uchburchak maydonini qaytarib beradi.
double s, r=(a+b+c)/2;
return s=sqtr(p*(p-a)*(p-b)*(p-c));//Geron formulasi
}
```

Shunday qilib, stekka faktik parametrlarning nusxalari kiritiladi va funksiya operatorlari ushbu nusxalar bilan ish olib boradi. Faktik parametrlarning o‘ziga funksiyaning kirish huquqi yo‘q, demak ularni o‘zgartirish imkoni ham yo‘q.

Qiymat bo‘yicha chaqirish kulaylik tug‘diradi. Chunki funksiyalarda kamroq o‘zgaruvchilarni ishlatishga imkon beradi. Misol uchun shu xususiyatni aks ettiruvchi POWER funksiyasi variantini keltiramiz:

```
int power(x,n)
{
int x,n;
int p;
for (p = 1; n > 0; --n)
p = p * x;
return (p);
}
```

Argument N vaqtinchalik o‘zgaruvchi sifatida ishlatiladi. Undan to qiymati 0 bulmaguncha bir ayriladi. N funksiya ichida o‘zgarishi funksiyaga murojjat qilingan boshlang‘ich qiymatiga ta’sir qilmaydi.

Adres bo‘yicha uzatish. Adres bo‘yicha uzatishda stekka parametrlar adreslarining nusxalari kiritiladi, demakki, funksiyada faktik parametr joylashtirilgan xotira uyasiga kirish huquqi paydo bo‘ladi va funksiya bu parametrni o‘zgartirishi mumkin.

Adres bo‘yicha uzatish uchun ilovalar qo‘llanishi mumkin. Ilova bo‘yicha uzatishda funksiyaga chaqirish paytida ko‘rsatilgan parametr adresi uzatiladi, funksiya ichida esa parametrga barcha murojaatlarning sezilmagan holda nomlari bekor qilinadi.

Misol uchun turtburchak yuzi va perimetrini berilgan tomonlari bo‘yicha hisoblash funksiyasini quyidagicha tasvirlash mumkin.

```
void pr(float a,float b, float& s, float& p)
{
p=2(a+b);
s= a*b;
}
```

Bu funksiyaga quyidagicha murojaat kilinishi mumkin pr(a,b,&p,&s). Funksiyaga p va s o‘zgaruvchilarning adreslari uzatiladi. Funksiya tanasida shu adreslar bo‘yicha 2*(a+b) va a*b qiymatlar yoziladi.

Keyingi misolda berilgan ikki o‘zgaruvchining qiymatlarini o‘zaro almashtirish funksiyasidan foydalaniladi:

```
#include <iostream>
using namespace std;
  void change(int &a,int &b)
  {
  int r=a; a=b;b=r;
  };
  int main()
  {
```

```
  int x=1,y=5;
  change(x,y);
  cout<<"x="<<x<<" y="<<y;
return 0;
}
```

Natija

x=5 y=1

Funksiyalarni qo‘shimcha yuklash

Bir xil nomli har xil funksiyalar. C++ tilida bir nomdagi bir nechta funksiyalarni tuzish imkoniyati mavjuddir. Bu *bir xil nomdagi har xil funksiyalar* deyiladi. Qayta yuklanuvchi funksiyalar bir – birlari bilan parametrlari ro‘yxati bilan farq qilishi lozim. Bunda parametrlar yoki turli sonda aniqlanishi, yoki tiplari bilan farqlanishi kerak. Quyidagi misollarni ko‘rib chiqamiz:

```
int myFunction( int, int )
int myFunction( long, long)
int myFunction( long )
```

myFunction() funksiyasi uchta turli parametrlar ro‘yxati bilan har xil nomda aniqlanyapti. Funksiyaning birinchi va ikkinchi versiyalari parametrlar tipi bilan, uchinchisi esa parametrlar soni bilan farq qilayapti.

Ushbu funksiyalarning qaytaradigan qiymatlarining tiplari bir xil yoki turlicha bo‘lishi mumkin. Lekin, parametrlari ro‘yxati bir xil bo‘lgan ikkita funksiya turli tipdagi qiymat qaytaradigan qilib aniqlansa dasturni kompilyatsiya qilish jarayonida xatolik yuz beradi.

Funksiyalarni bir xil nom bilan aniqlanishi ularning polimorfizmi deb ham ataladi. Polimorfizm so‘zi poli *(*gr. poly*)* - ko‘p, morfe *(*gr. morphyo*)* - shakl co‘zidan olingan bo‘lib, ko‘pshakllilik degan ma’noni anglatadi.

Funksiyaning polimorfizmi deganda dasturda bir nechta turli vazifalarni bajaruvchi bir xil nomdagi funksiyalar bo‘lishi tushuniladi. Parametrlari soni va tipini o‘zgartirish orqali bir nechta bir xil nomdagi funksiyalarni aniqlash mumkin. Bunday

holda funksiyani chaqirishda hech qanday noaniqlik bo‘lmaydi, kerakli funksiya parametriga muvofiq tarzda aniqlanadi.

Misol uchun har xil o‘zgaruvchilarni ko‘paytirish uchun quyidagi funksiyalar kiritilgan bo‘lsin:

```
Double multy(double x) {return x*x*x;}
Double multy(double x, double y) {return x*y*y;}
Double multy(double x,double y, double z) {return x*y*z;}
```

Quyidagi murojaatlarning har biri to‘g‘ri bajariladi:

```
multy(0. 4)
multy(4. 0,12. 3)
multy(0. 1,1. 2,6. 4);
```

Faraz qilamiz, siz ixtiyoriy berilgan qiymatni ikkiga ko‘paytiradigan funksiya yozmoqchisiz. Bu funksiyaga int, long, float yoki double tipidagi qiymatlarni uzatish imkoniyati bo‘lishini hohladingiz. Bir xil nomli har xil funksiyalarsiz buni bajarish uchun to‘rtta turli nomdagi funksiyalarni hosil qilishingiz kerak bo‘ladi:

```
int DoubleInt(int);
long DoubleLong(long);
fl
oat DoubleFloat(float);
double DoubleDouble(double);
```

Bir xil nomli funksiyalar yordamida esa ularni o‘rniga ushbu funksiyalarni aniqlashimiz mumkin.

```
int Double(int);
long Double(long);
float Double (float);
double Double(Double);
```

Qayta yuklangan funksiyalar chaqirilganda kompilyator aynan qaysi variantdagi funksiyani ishlatish kerakligini avtomatik tarzda uzatilayotgan parametrlarning tiplariga muvofiq aniqlaydi.

Funksiyalarning polimorfizmi

```
#include <iostream>
using namespace std;
int Doubled(int);
long Doubled(long);
float Doubled(float);
double Doubled(double);
int main()
{
int myInt = 6500;
long myLong = 65000;
float myFloat=6. 5;
double myDouble = 6. 5e20;
int doubledInt;
long doubledLong;
float doubledFloat;
double doubledDouble;
cout<<"myInt: "<< myInt<< "\n";
cout<<"myLong: "<< myLong<< "\n";
cout<<"myFloat: "<< myFloat<< "\n";
cout<<"myDouble: "<< myDouble<< "\n";
doubledInt=Doubled(myInt);
doubledLong=Doubled(myLong);
doubledFloat=Doubled(myFloat);
doubledDouble=Doubled(myDouble);
cout<<"doubledInt:"<<doubledInt<<"\n";
cout<<"doubledLong:"<<doubledLong<<"\n";
cout<<"doubledFloat:"<<doubledFloat<<"\n";
cout<<"doubledDouble:"<<doubledDouble<<"\n";
return 0;
}
```

```
int Doubled(int original)
{
return 2*original;
}
long Doubled(long original)
{
return 2*original;
}
float Doubled(float original)
{
return 2*original;
}
double Doubled(double original)
{
return 2*original;
}
```

Natija:

MyInt: 6500

MyLong: 65000

MyFloat: 6. 5

MyDouble: 6. 5 e+20

DoubledInt: 13000

DoubledLong: 130000

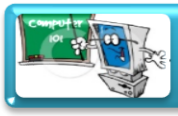
DoubledFloat: 13

DoubledDouble: 13e+21

Funksiyalarni qo‘shimcha yuklash qoidalari.

1) Funksiyalar bitta ko‘rinish soxasiga tegishli bo‘lishi lozim.
2) Funksiyalar parametrlari ko‘zda tutilgan qiymatlarga ega bo‘lishi mumkin, lekin har xil funksiyalardagi bir xil parametrlar bir xil qiymatga ega bo‘lishi kerak.

3) Agar funksiyalar parametrlari ta’rifi faqat const modifikatori yoki ilova mavjudligi bilan farq kilsa, bu fuknksiyalarni qo‘shimcha yuklash mumkin emas. Masalan kompilyator int&f1(int&,const int&){. . . } va
int f1(int,int){. . . } – funksiyalarni ajrata olmaydi.



YANGI PEDAGOGIK TEXNOLOGIYALAR
“Aqliy hujum” strategiyasi

1. Birinchi qaysi funksiya bajariladi?
2. Simvolli kiritish funksiyalari.
3. Shartli operator umumiy ko‘rinishi.
4. Nima uchun barcha o‘zgaruvchilarni global deb e’lon qilish maqsadga muvofiq emas.
5. Nima uchun funksiyaga argument sifatida uzatilgan o‘zgaruvchilar qiymati funksiya tanasida o‘zgartirilsa dasturning asosiy kodidagi shu o‘zgaruvchi qiymatiga akslantirilmaydi.
6. Funksiya prototipini e’lon qilish va funksiyani aniqlash o‘rtasida qanday farq bor?
7. Funksiya prototipini ko‘rsatishda, aniqlashda va chaqirishda uning parametrlari nomlari ustma – ust tushishi kerakmi?
8. Agarda funksiya hech qanday qiymat qaytarmasa uni qanday e’lon qilish kerak?
9. Agarda funksiyani e’lon qilishda qaytaradigan tipi aniqlanmasa, u boshlang‘ich ravishda qanday tip qaytaradi.
10. Lokal o‘zgaruvchi nima?
11. Ko‘rinish sohasi nima?
12. Bir xil nomli har xil funksiyalar qanday aniqlanadi?
13. Quyidagi ifodalar hisoblanshggandan so‘ng i va j o‘zgaruvchilarinning qiymati nechaga teng bo‘ladi: i=5,i+=6, j=i++;
14. Agarda Myage, a va v o‘zgaruvchilarining tiplari int bo‘lsa ularning qiymatlari quyidagi ifodalar bajarilgandan so‘ng nechaga teng bo‘ladi.
    a. Myage = 39;

b. a=Myage++

c. b=++Myage

7.7. C++ DASTURLASH TILIDA BIR O'LCHOVLI MASSIVLAR.

Reja:

1. Massivlar tushunchasi.
2. Massivlarni navlarga ajratish.
3. Bir o‘lchamli massivlarni funksiya parametrlari sifatida uzatish
4.

Tayanch iboralar: Massivlar haqida tushuncha. Massivlarni tavsiflash va ulardan foydalanish. Bir o’lchovli massivlar.

Bir o‘lchovli massivlar

Massiv tushunchasi. Massiv bu bir tipli nomerlangan ma’lumotlar jamlanmasidir. Massiv indeksli o‘zgaruvchi tushunchasiga mos keladi. Massiv ta’riflanganda tipi, nomi va indekslar chegarasi ko‘rsatiladi. Masalan type turidagi length ta elementdan iborat a nomli massiv shunday e’lon qilinadi:

type a[length];

Bu maxsus a[0], a[1],. . . , a[length -1] nomlarga ega bo‘lgan type turidagi o‘zgaruvchilarning e’lon qilinishiga to‘g‘ri keladi.

Massivning har bir elementi o‘z raqamiga - indeksga ega. Massivning x-nchi elementiga murojaat indekslash operatsiyasi yordamida amalga oshiriladi:

int x=. . . ; //butun sonli indeks

TYPE value=a[x]; //ch-nchi elementni o‘qish

a[x]=value; //x-yxb elementga yozish

Indeks sifatida butun tur qiymatini qaytaradigan har qanday ifoda qo‘llanishi mumkin: char, short, int, long. C++ da massiv elementlarining indekslari 0 dan boshlanadi (1 dan emas), length elementdan iborat bo‘lgan massivning oxirgi elementining indeksi esa - bu length -1 (length emas). Massivning int z[3] shakldagi ta’rifi, int tipiga tegishli z[0],z[1],z[2] elementlardan iborat massivni aniqlaydi. Massiv chegarasidan tashqariga chiqish (ya’ni mavjud bo‘lmagan elementni o‘qish/yozishga urinish) dastur bajarilishida kutilmagan natijalarga olib kelishi mumkin. Shuni ta’kidlab o‘tamizki, bu eng ko‘p tarqalgan xatolardan biridir.

Agar massiv initsializatsiya qilinganda elementlar chegarasi ko‘rsatilgan bo‘lsa, ro‘yxatdagi elementlar soni bu chegaradan kam bo‘lishi mumkin, lekin ortiq bo‘lishi mumkin emas.

Misol uchun int a[5]={2,-2}. Bu holda a[0] va a[1] qiymatlari aniqlangan bo‘lib, mos holda 2 va –2 ga teng. Agar massiv uzunligiga qaraganda kamroq element berilgan bo‘lsa, qolgan elementlar 0 hisoblanadi:

int a10[10]={1, 2, 3, 4}; //va 6 ta nol

Agar nomlangan massivning tavsifida uning o‘lchamlari ko‘rsatilmagan bo‘lsa, kompilyator tomonidan massiv chegarasi avtomatik aniqlanadi:

int a3[]={1, 2, 3};

Massivda musbat elemenlar soni va summasini hisoblash.

```
#include <iostream>
using namespace std;
int main()
int s=0,k=0;
int x[]={-1,2,5,-4,8,9};
for(int i=0; i<6; i++)
{
if (x[i]<=0) continue;
k++;
s+=x[i];
};
cout<<k<<endl;
cout<<s;
return 0;
};
```

Massivning eng katta, eng kichik elementi va o‘rta qiymatini aniqlash:

```
#include <iostream>
using namespace std;
```

```
int main()
{
int i,j,n;
int min,max,s;
float a,b,d,x[100];
while(1)
{
cout<<("\n n=");
cin>>n;
if ( n>0 && n<=100 ) break;
cout<<"\n Hato 0<n<101 bo'lishi kerak";
}
cout<<"\n elementlar qiymatlarini kiriting:\n";
for (i=0;i<n;i++)
{
cout<<"x["<<i<<"]=";
cin>>x[i];
}
max=x[0];
min=x[0];
for (s=0,i=0;i<n;i++)
{
s+=x[i];
if (max<x[i]) max=x[i];
if (min>x[i]) min=x[i];
};
s/=n;
cout<<"\n max="<<max;
cout<<"\n min="<<min;
cout<<"\n o'rta qiymat="<<s;
```

```
return 0;
}
```

Massivlarni navlarga ajratish. Navlarga ajratish - bu berilgan ko‘plab obektlarni biron-bir belgilangan tartibda qaytadan guruhlash jarayoni.

Massivlarning navlarga ajratilishi tez bajarilishiga ko‘ra farqlanadi. Navlarga ajratishning n*n ta qiyoslashni talab qilgan oddiy usuli va n*log(n) ta qiyoslashni talab qilgan tez usuli mavjud. Oddiy usullar navlarga ajratish tamoyillarini tushuntirishda qulay hisoblanadi, chunki sodda va kalta algoritmlarga ega. Murakkablashtirilgan usullar kamroq sonli operatsiyalarni talab qiladi, biroq operatsiyalarning o‘zi murakkabroq, shuning uchun uncha katta bo‘lmagan massivlar uchun oddiy usullar ko‘proq samara beradi.

Oddiy usullar uchta asosiy kategoriyaga bo‘linadi:

- oddiy kiritish usuli bilan navlarga ajratish;
- oddiy ajratish usuli bilan navlarga ajratish;
- oddiy almashtirish usuli bilan navlarga ajratish

Oddiy kiritish usuli bilan navlarga ajratish

Massiv elementlari avvaldan tayyor berilgan va dastlabki ketma-ketliklarga bo‘linadi. i=2 dan boshlab, har bir qadamda dastlabki ketma-ketlikdan i-nchi element chiqarib olinadi hamda tayyor ketma-ketlikning kerakli o‘rniga kiritib qo‘yiladi. Keyin i bittaga ko‘payadi va h. k.

Kerakli joyni izlash jarayonida, ko‘proq o‘ngdan bitta pozitsiyadan tanlab olingan elementni uzatish amalga oshiriladi, ya’ni tanlab olingan element, j:=i-1 dan boshlab, navlarga ajratib bo‘lingan qismning navbatdagi elementi bilan qiyoslanadi. Agar tanlab olingan element a[i] dan katta bo‘lsa, uni navlarga ajratish qismiga qo‘shadilar, aks holda a[j] bitta pozitsiyaga suriladi, tanlab olingan elementni esa navlarga ajratilgan ketma-ketlikning navbatdagi elementi bilan qiyoslaydilar. To‘g‘ri keladigan joyni qidirish jarayoni ikkita turlicha shart bilan tugallanadi:

- agar a[j]>a[i] elementi topilgan bo‘lsa;
- agar tayyor ketma-ketlikning chap uchiga yetilgan bo‘lsa.

```
int i, j, x;
for(i=1; i<n; i++)
{
x=[i];// kiritib qo‘ishimiz lozim bo‘lgan elementni esda saqlab qolamiz
j=i-1;
while(x<a[j]&&j>=0)//to‘g‘ri keladigan joyni qidirish
}
a[j+1]=a[j] /o‘nga surilish
j--;
}
a[j+1]=x;//elementni kiritish
}
```

Oddiy tanlash usuli bilan navlarga ajratish

Massivning minimal elementi tanlanadi hamda massivning birinchi elementi bilan joy almashtiriladi. Keyin jarayon qolgan elementlar bilan takrorlanadi va h. k.

```
int i,min,n_min,j;
for(i=0;i<n-1;i++)
{
min=a[i];n_min=i; //minimal qiymatni qidirish
for(j=i+1;j<n;j++)
if(a[j]<min){min=a[j];n+min=j;}
a[n_min]=a[i];//almashtirish
a[i]=min;
}
```

Oddiy almashtirish usuli bilan navlarga ajratish

Elementlar juftlari oxirgisidan boshlab qiyoslanadi va o‘rin almashinadi. Natijada massivning eng kichik elementi uning eng chapki elementiga aylanadi. Jarayon massivning qolgan elementlari bilan davom yettiriladi.

```
for(int i=1;i<n;i++)
for(int j=n-1;j>=i;j—
```

```
if(a[j]<a[j-1])
{int r=a[j];a[j]=a[j-1];a[j-1]=r;}
}
```

Bir o‘lchamli massivlarni funksiya parametrlari sifatida uzatish. Massivdan funksiya parametri sifatida foylalanganda, funksiyaning birinchi elementiga ko‘rsatkich uzatiladi, ya’ni massiv hamma vaqt adres bo‘yicha uzatiladi. Bunda massivdagi elementlarning miqdori haqidagi axborot yo‘qotiladi, shuning uchun massivning o‘lchamlari haqidagi ma’lumotni alohida parametr sifatida uzatish kerak.

Misol:

Massiv barcha elementlari chiqarilsin:

```
#include <iostream>
using namespace std;
int form(int a[100])
{
int n;
cout<<"\nEnter n:";
cin>>n;
for(int i=0;i<n;i++)
a[i]=rand()%100;
return n;
}
void print(int a[100],int n)
{
for(int i=0;i<n;i++)
cout<<a[i]<<" ";
cout<<"\n";
}
int main()
{
        int a[100];
```

```
	int n;
	n=form(a);
	print(a,n);
return 0;
}
```

Funksiyaga massiv boshlanishi uchun ko‘rsatkich uzatilgani tufayli (adres bo‘yicha uzatish), funksiya tanasining operatorlari hisobiga massiv o‘zgarishi mumkin. Funksiyalarda bir o‘lchovli sonli massivlar argument sifatida ishlatilganda ularning chegarasini ko‘rsatish shart emas.

Misol:

Massivdan barcha juft elementlar chiqarilsin:

```
#include <iostream>
using namespace std;
void form(int a[],int n)
{
 for(int i=0;i<n;i++)
	a[i]=rand()%100;
	}
void print(int a[],int n)
{
	for(int i=0;i<n;i++)
		cout<<a[i]<<" ";
	cout<<"\n";
}
int Dell(int a[],int n)
{
	int j=0,i,b[100];
	for(i=0;i<n;i++)
		if(a[i]%2!=0)
		{
```

```
                b[j]=a[i];j++;
            }
            for(i=0;i<n;i++)a[i]=b[i];
return j;
}
int main()
{
    int a[100];
    int n;
    cout<<"\nEnter n:";
    cin>>n;
    form(a,n);
    print(a,n);
    n=Dell(a,n);
    print(a,n);
return 0;
}
```

Funksiyalarda bir o‘lchovli sonli massivlar argument sifatida ishlatilganda ularning chegarasini ko‘rsatish shart emas. Misol tarikasida n o‘lchovli vektorlar bilan bog‘liq funksiyalarni ta’riflashni ko‘rib chiqamiz.

Vektorning uzunligini aniqlash funksiyasi:

```
float mod_vec(int n,float x[])
{
float a=0;
for (int i=0; i<n; i++)
 a+=x[i]*x[i];
return sqrt(double(a));
}
```

Funksiyalarda bir o‘lchovli massivlar qaytariluvchi qiymatlar sifatida ham kelishi mumkin. Misol uchun ikki vektor summasini hisoblovchi funksiya protsedurani ko‘ramiz:

```
void sum_vec(int n, float a,float b, float c)
{
for(int i=0;i<n;i++,c[i]=a[i]+b[i]);
}
```

Bu funksiyaga quyidagicha murojaat kilish mumkin:

```
float a[]={1,-1. 5,-2};
b[]={-5. 2,1. 3,-4},c[3];
sum_vec(3,a,b,c);
```

7.8. C++ DASTURLASH TILIDA IKKI O'LCHOVLI MASSIVLAR.

Reja:

1. Ko‘p o‘lchovli massivlar ta’rifi
2. Funksiyaga ko‘p o‘lchamli massivlarni uzatish

Tayanch iboralar: Ko’p o’lchovli (indeksli) massivlar. Massivlarni navlarga ajratish usullari.

Ko‘p o‘lchovli massivlar ta’rifi.

Ikki o‘lchovli massivlar matematikada matritsa yoki jadval tushunchasiga mos keladi. Jadvallarning insializatsiya qilish qoidasi, ikki o‘lchovli massivning elementlari massivlardan iborat bo‘lgan bir o‘lchovli massiv ta’rifiga asoslangandir.

Misol uchun ikki qator va uch ustundan iborat bo‘lgan xaqiqiy tipga tegishli d massiv boshlang‘ich qiymatlari quyidagicha ko‘rsatilishi mumkin:

```
float d[2][3]={(1,-2. 5,10),(-5. 3,2,14)};
```

Bu yozuv quyidagi qiymat berish operatorlariga mosdir:

```
d[0][0]=1;d[0][1]=-2. 5;d[0][2]=10;
d[1][0]=-5. 3;d[1][1]=2;d[1][2]=14;
```

Bu qiymatlarni bitta ro‘yxat bilan xosil qilish mumkin:

```
float d[2][3]={1,-2. 5,10,-5. 3,2,14};
```

Initsializatsiya yordamida boshlang‘ich qiymatlar aniqlanganda massivning hamma elementlariga qiymat berish shart emas.

Misol uchun: int x[3][3]={(1,-2,3),(1,2),(-4)}.

Bu yozuv quyidagi qiymat berish operatorlariga mosdir:

x[0][0]=1;x[0][1]=-2;x[0][2]=3;

x[1][0]=-1;x[1][1]=2;x[2][0]=-4;

Initsializatsiya yordamida boshlang‘ich qiymatlar aniqlanganda massivning birinchi indeksi chegarasi ko‘rsatilishi shart emas, lekin qolgan indekslar chegaralari ko‘rsatilishi shart.

Misol uchun:

double x[][2]={(1. 1,1. 5),(-1. 6,2. 5),(3,-4)}

Bu misolda avtomatik ravishda katorlar soni uchga teng deb olinadi.

Quyidagi ko‘radigan misolimizda jadval kiritilib har bir katorning maksimal elementi aniklanadi:

```
#include <iostream>
using namespace std;
int main()
{
double a[4][3];
double max;
int i,j;
for(i=0;i<4;i++)
{
for(j=0;j<3;j++)
{
cout<<"a["<<i<<"]["<<j<<"]=";
cin>>a[i][j];
}
cout<<'\n';
};
```

```
for(i=0;i<4;i++)
{
max=a[i][0];
for(j=0;j<3;j++)
if (max<a[i][j]) max=a[i][j];
cout<<max<<endl;
}
return 0;
}
```

Funksiyaga ko‘p o‘lchamli massivlarni uzatish. Ko‘p o‘lchamli massivlarni funksiyaga uzatishda barcha o‘lchamlar parametrlar sifatida uzatilishi kerak. C++ da ko‘p o‘lchamli massivlar aniqlanishi bo‘yicha mavjud emas. Agar biz bir nechta indeksga ega bo‘lgan massivni tavsiflasak (masalan, int mas [3][4]), bu degani, biz bir o‘lchamli mas massivini tavsifladik, bir o‘lchamli int [4] massivlar esa uning elementlaridir

Misol: Kvadrat matritsani uzatish (transportirovka qilish)

Agar void transp(int a[][],int n){. . . . . } funksiyasining sarlavhasini aniqlasak, bu holda biz funksiyaga noma’lum o‘lchamdagi massivni uzatishni xohlagan bo‘lib qolamiz. Aniqlanishiga ko‘ra massiv bir o‘lchamli bo‘lishi kerak, hamda uning elementlari bir xil uzo‘nlikda bo‘lishi kerak. Massivni uzatishda uning elementlarining o‘lchamlari haqida ham biron narsa deyilmagan, shuning uchun kompilyator xato chiqarib beradi.

Bu muammoning eng sodda yechimi funksiyani quyidagicha aniqlashdir:

void transp(int a[][4],int n){. . . . . }, bu holda har bir satr o‘lchami 4 bo‘ladi, massiv ko‘rsatkichlarining o‘lchami esa hisoblab chiqariladi.

```
#include <iostream>
using namespace std;
const int N=4;//global o‘zgaruvchi
void input_array(int a[][N],int n)
{
```

```
int i,j;
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
cout<<"a["<<i<<"]["<<j<<"]=";
cin>>a[i][j];
}
cout<<'\n';
}
}
void print_array(int a[][N],int n)
{
int i,j;
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
cout<<"a["<<i<<"]["<<j<<"]=";
cout<<a[i][j]<<" ";
}
cout<<'\n';
}
}
void transp(int a[][N],int n)
{
int r;
for(int i=0;i<n;i++)
for(int j=0;j<n;j++)
if(i<j)
```

```
{
r=a[i][j];a[i][j]=a[j][i];a[j][i]=r;
}
}
int main()
{
int mas[N][N];
int n;
cout<<'\n'<<" n="; cin>>n;
input_array(mas,n);
transp(mas,n);
print_array(mas,n);return 0;
}
```

Misol tarikasida uch o‘lchovli kvadrat matritsani uch o‘lchovli vektorga ko‘paytirish funksiyasini ko‘rib chiqamiz:

```
void umn_vec( float a[3][3],float b[3], float c[3])
{
for(int i=0; i<3; i++)
{
c[i]=0;
for(int j=0; j<3; j++)
c[i]+=a[i][j]*b[j];
};
}
```

Har xil chegarali jadvallar bilan funksiyalardan foydalanishning bir yuli bu oldindan kiritiluvchi konstantalardan foydalanishdir.

Mavzuga oid savollar

1. Satr simvolli massivdan qanday farq qiladi?

2. Bir o‘lchovli massivlarni initsializatsiya qilish usullarini ko‘rsating.
3. Ko‘p o‘lchovli massiv ta’rifi xususiyatlarini keltiring.
4. Ko‘p o‘lchovli masivlar initsializatsiyasi xususiyatlari.

7.9. C++ DA KO’RSATKICHLAR VA SATRLAR.

Reja:

1. Pointer (ko'rsatkich) va satrlar
2. Pointer operatorlari
3. Pointer argumentli funksiyalar
4. Const sifatli pointerlar
5. Pointer va oddiy o'zgaruvchilarning egallagan adres kattaligi.

Tayanch iboralar: Adres (manzil) operatori. Jo’natish operatori. Ko’rsatkich tipidagi o’zgaruvchilarni e’lon qilish. Ko‘rsatkichga boshlang‘ich qiymat berish. Ko‘rsatkich ustida amallar. Adresni olish amali. Ko‘rsatkichlar va adres oluvchi o‘zgaruvchilar funksiya parametri sifatida. Ko’rsatkichlar va massivlar.

POINTER (ko'rsatkich) VA SATRLAR

C++ da ikki ko'rinishdagi ko'rsatkichlar - &-ko'rsatkichlar va pointerlar mavjuddir. Aslida bularning farqi faqat qo'llanilishi va ko'rinishida desak ham bo'ladi. Bu qismda biz C dan meros qolgan pointerlar bilan yaqindan tanishamiz. &-ko'rsatkichlarni biz o'tgan qismda ko'rgan edik. Pointerlar C/C++ dasturlash tillarining eng kuchli qurollaridandir. Lekin pointer tushunchasini anglash ham oson emas. Pointerlar yordamida funksiyalarning ko'rsatkich bo'yicha chaqirish mehanizmini amalga oshirish mumkin. Undan tashqari pointerlar yordamida dinamik strukturalar – stek (stack), ro'yhat (list), navbat (queue) va darahtlar (tree) tuzish mumkin. Undan tashqari pointer, satr va massivlar orasida yaqin aloqa bordir. Satr va massivlarni pointerlar yordamida berish bizga C dan meros bo'lob qoldi. Keyingi boblarda biz satr va massivlarni to'la qonli ob'ekt sifatida qo'lga olamiz. Pointerlar qiymat sifatida hotira adreslarini oladilar. Oddiy o'zgaruvchilar ma'lum bir qiymatga ega bo'lgan bir paytda, pointerlar boshqa bir o'zgaruvchining adresini o'z ichlariga oladilar. Shunda o'zgaruvchi bevosita qiymatga ko'rsatib tursa, pointer qiymatga bilvosita ko'rsatadi. Pointer e'lonida, pointer ismidan oldin '*' (yulduzcha, ko'paytiruv) belgisi qo'yilishi kerak. Misolda ko'raylik:

```
char *charPtr, c = 8, *pc, ff[] = "IcyCool";
```

Bu erda charPtr va pc lar char tipidagi ko'rsatkichlardir. Yani, charPtr ni "char tipidagi oz'garuvchiga ga ko'rsatkich" deb o'qisak bo'ladi. Ko'rsatkich sifatida e'lon qilinayatgan har bir o'zgaruvchi ismi oldida '*' bo'lishi shartdir. Pointerlarni boshqa o'zgaruvchilar kabi e'lon davrida, yoki dastur ichida qiymat berish yordamida initsalizatsiya qilish mumkin. Pointerlar qiymat sifatida faqat o'zgaruvchi yoki ob'ektlarning adreslarini va NULL yoki 0 ni oladilar. NULL simvolik konstantasi <iostream. h> va boshqa bir necha standart e'lon fayllarida aniqlangan. Pointerga NULL qiymatini berish pointerni 0 ga tenglashtirish bilan tengdir, ammo C++ pointerga to'g'ridan-to'g'ri 0 qiymatini berish afzalroqdir, chunki agar NULL emas, 0 qiymati berilsa, ushbu qiymat avtomatik ravishda pointerning tipiga keltiriladi. Butun sonlardan faqat 0 qiymati pointerga keltirilishsiz berilishi mumkin. Agar ma'lum bir hotira adresini pointerga bermoqchi bo'lsak, quyidagicha yozishimiz mumkin:

```
int *iPtr, address = 0x45ad7 ;
iPtr = (int *) address; // C uslubida tiplarni keltirish
iPtr = static_cast<int *>(address); // C++ " " "
```

Lekin, albatta, yuqoridagi yozganimizni kamdam-kam qo'llashga to'g'ri kelsa kerak, chunki adres olishning soddaroq yo'llari bordir. Aslida, pointer e'lon qilinsa-yu lekin hali qo'llanilmayatgan bo'lsa, unga 0 qiymatini berish tavsiya etiladi. Chunki, agar biz bu 0 qiymatli pointerni bilmasdan qo'llamoqchi bo'lsak, kompilyator xato beradi, bunga sabab odatda operatsiyon sistemalar 0 adresli hotira maydoni bilan ishlashga ruhsat bermaydilar. Shu tariqa qiymatsiz pointer qo'llanilganidan ogoh bo'lamiz.

POINTER OPERATORLARI

O'zgaruvchilarning (yani harqanday ob'ektning) adresini olishda biz & operatorini - adres olish operatorini qo'llaymiz. Bu kontekstda & operatori bir dona argument oladi. Undan tashqari & ikkili operatori bitli qo'shishda qo'llaniladi. Adres olishga misol keltiraylik.

```
int *iPtr, var = 44;
```

```
iPtr = &var;
 double d = 77. 0, *dPtr = &d;
```

Bu erda bir narsani o'tib ketishimiz kerak. C++ da identefikatorlar (o'zgaruvchi va ob'ektlar) ikki turda bo'ladi. Birinchisi chap identefikatorlar (lvalue - left value: chap qiymat), ikkinchi tur esa o'ng identefikatorlardir (rvalue - right value: o'ng qiymat). Yani chap identefikatorlar '=' (qiymat berish operatori) belgisining chap argumenti sifatida qo'llanilishi mumkin. O'ng identifikatorlar esa '=' ning o'ngida qo'yilishlari kerak. Bunda o'ng identefikatorlar har doim ham chap identefikator o'rnida qo'llanila olomaydilar. Yani chap identefikatorlarning qiymatlari '=' operatori yordamida o'zgartirilishi mumkin. Agar o'zgaruvchi *const* sifati bilan e'lon qilingan bo'lsa, u normal sharoitda faqat o'ng identefikatordir. Bu ma'lumotlarni keltirganimizning sababi, & adres olish operatori faqat chap identefikator bo'la oladigan o'zgaruvchilarga nisbatan qo'llanilishi mumkin. Agar o'zgaruvchi const sifatli konstantalarga, register sifatli o'zgaruvchilarga va ko'rsatkich qaytarmaydigan (adres qaytarmaydigan) ob'ektlarga nisbatan qo'llanilishi ta'qiqlanadi. Faraz qilaylik, biz pointerimizga boshqa bir o'zgaruvchining adresini berdik. Endi bizning pointerimiz ikki qiymatni ifoda etmoqda, biri bu o'zining qiymati, yani boshqa o'zgaruvchining adresi, ikkinchi qiymat esa, bu boshqa o'zgaruvchining asl qiymatidir. Agar biz pointerning o'zi bilan ishlasak, biz hotiradagi adres bilan ishlagan bo'lamiz. Ko'p hollarda esa buning keragi yo'q. Pointer ko'rsatayotgan o'zgaruvchining qiymati bilan ushbu pointer yordamida ishlash uchun biz '*' belgisini, boshqacha qilib etganda, ko'rsatish operatorini (indirection, dereferencing operator) qo'llashimiz kerak bo'ladi. Bunga misol beraylik.

```
. . .
int k = 74;
int *kPtr = &k;
. . .
cout << k << " --- " << *kPtr << endl; // Ekranda:
 // 74 --- 74
cin >> *kPtr; // 290 qiymatini kiritaylik. . .
cout << k << " === " << *kPtr << endl; // Ekranda:
```

```
// 290 === 290
*kPtr = 555;
cout << k << ". . . "<< *Ptr << endl; // Ekranda:
// 555. . . 555
. . .
```

Ko'rib turganimizdek, biz kPtr ko'rsatkichi orqali k o'zgaruvchining ham qiymatlarini o'zgartira oldik. Demak *kPtr chap identefikatordir, chunki uning qiymatini qiymat berish operatori yordamida o'zgartira olar ekanmiz. Agar pointerimiz 0 ga teng bo'lsa, uni ko'rsatish operatori - '*' bilan ko'llash ko'p hollarda dastur ijrosi xatolariga olib keladi. Undan tashqari, boshlangich qiymatlari aniqlanmagan pointerni qiymatiga ko'rsatish eng kamida mantiqiy xatolarga olib keladi, bunung sababi, pointer ko'rsatayotgan hotira qismida oldingi ishlagan dasturlar kodlari qolgan bo'lishi mumkin. Bu esa bizga hech keragi yo'q. Pointer bo'lmagan o'zgaruvchilarga ko'rsatish operatorini qo'llash ta'qiqlanadi. Aslini olganda, & adres olish operatori va * ko'rsatish operatorlari, bir-birining teskarisidir. Bir misol kertiraylik.

```
// Adres olish va ko'rsatish operatorlari
# include <iostream. h>
char c = 44; // char tipidagi o'zgaruvchi
char *pc = &c; // char tipidagi pointer
int main ()
{
cout << "&*pc ning qiymati: " << &*pc << endl;
cout << "*&pc ning qiymati: " << *&pc << endl;
cout << "c ning hotiradagi adresi: " << &c << endl;
cout << "pc pointerning qiymati: " << pc << endl;
cout << "c ning qiymati: " << c << endl;
cout << "*pc ning qiymati: " << *pc << endl;
return (0);
}
```

Ekranda:

&*pc ning qiymati: 0x4573da55

*&pc ning qiymati: 0x4573da55

c ning hotiradagi adresi: 0x4573da55

pc pointerning qiymati: 0x4573da55

c ning qiymati: 44

*pc ning qiymati: 44

Demak, &*pc va *&pc ayni ishni bajarar ekan, yani * va & operatorlari bir-birining teskarisidir. Hotiradagi adres ekranga boshqa ko'rinishda chiqishi mumkin. Bu mashina va kompilyatorga bog'liqdir.

POINTER ARGUMENTLI FUNKSIYALAR

Funksiylar ikki argumentlariga qarab ikki turga bo'linadi degan edik. Argumentlar qiymat bo'yicha, yoki ko'rsatkich bo'yicha berilishi mumkin edi. Qiymat bo'yicha berilgan argumentning funksiya chaqirig'iga nushasi beriladi. Ko'rsatkich bo'yicha argument chaqirig'ida, funksiyaga kerakli argumentga ko'rsatkich beriladi. Ko'rsatkich bo'yicha chaqiriqni ikki usulda bajarish mumkin, birinchi usul &-ko'rsatkichlar orqali amalga oshiriladi. Ikkinchi usulda esa pointerlar qo'llaniladi. Pointerlar bilan chaqishning afzalligi (qiymat bo'yicha chaqiriq bilan solishtirganda) shundagi, agar ob'ektlar katta bo'lsa, ulardan nusha olishga vaqt ketqizilmaydi. Undan tashqari funksiya ob'ektning asl nushasi bilan ishlaydi, yani ob'ektni o'zgartura oladi. Funksiya faqat bitta ob'ektni yoki o'zgaruvchini return ifodasi yordamida qiytara olgani uchun, oddiy yol bilan, qiymat bo'yicha chaqiriqda funksiya faqat bitta o'zgaruvchining qiymatini o'zgartira oladi. Agar pointerlarni qo'llasak, bittadan ko'p ob'ektlarni o'zgartirishimiz mumkin, huddi &-ko'rsatkichli chaqiriqdagi kabi. Funksiya chaqirig'ida esa, biz o'zgaruvchilarning adresini qo'llashimiz kerak. Buni & adres olish operatori yordamida bajaramiz. Massivni berayatganda esa adresni olish kerak emas, chunki massivning ismining o'zi massiv birinchi elementiga pointerdir.

Pointerlarni qo'llab bir dastur yozaylik.

// Pointer argumentli funksiyalar

include <iostream. h>

```
int foo1(int k) {return (k * k);}
void foo2(int *iPtr) {*iPtr = (*iPtr) * (*iPtr);}
int main()
{
 int qiymat = 9;
 int javob = 0;
 javob = foo1(qiymat); // javob = 81
 cout << "javob = " << javob << endl;
 foo2(&qiymat); // qiymat = 81
 cout << "qiymat = " << qiymat << endl;
 return (0);
}
```

Ekranda:

```
javob = 81
qiymat = 81
```

Yuqoridagi dasturimizda foo2() funksiya chaqirig'ida qiymat nomli o'zgaruvchimizning adresini oldik (& operatori) va funksiya berdik. foo2() funksiyamiz iPtr pointer argumentining qiymatini * operatori yordamidao'zgartiryapti. Funksiya e'lonida pointer tipidagi parametrlardan keyin o'zgaruvchi ismlarini berish shart emas. Masalan:

```
int func(int *, char * ); // funksiya e'loni
int func(int *arg1, char *arg2); // funksiya e'loni
```

Yuqoridagi ikki e'lon aynidir. Aytib o'tkanimizdek, massivlarning ismlari birinchi elementlariga ko'rsatkichdir. Hatto, agar massiv bir indeksli bo'lsa, biz massivlar bilan ishlash uchun pointer sintaksisini qo'llashimiz mumkin. Kompilyator

```
foo(int m[]);
```

e'lonini

```
foo(int * const m);
```

e'loniga almashtiradi. Yuqoridagi m pointerini "int tipiga o'zgarmas pointer" deb o'qiymiz. const bilan pointerlarning qo'llanilishini alohida ko'rib chiqamiz.

const SIFATLI POINTERLAR

const ifodasi yordamida sifatlantirilgan o'zgaruvchining qiymatini normal sharoitda o'zgartira olmaymiz. const ni qo'llash dasturning xatolardan holi bo'lishiga yordam beradi. Aslida ko'p dasturchilar const ni qo'llashga o'rganishmagan. Shu sababli ular katta imkoniyatlarni boy beradilar. Bir qarashda const ning keragi yo'qdek tuyuladi. Chunki const ni qo'llash dasturning hech qaysi bir erida majburiy emas. Masalan konstantalarni belgilash uchun # define ifodasini qo'llasak bo'ladi, kiruvchi argumentlarni ham const sifatisiz e'lon qilsak, dastur mantig'i o'zgarishsiz qoladi. Lekin const kerak-kerakmas joyda o'zgaruvchi va ob'ektlarning holat-qiymatlarini o'zgartirilishidan himoyalaydi. Yani ob'ekt qiymatini faqat cheklangan funksiyalar va boshqa dastur bloklari o'zgartira oladilar. Bu kabi dasturlash uslubi esa, yani ma'lumotni berkitish va uni himoya

qilish ob'ektli dasturlash falsafasiga kiradi. Ko'rsatkich qo'llanilgan funksiyalarda, agar argumentlar funksiya tanasida o'zgartirilmasa, kirish parametrlari const deb e'lon qilinishlari kerak. Masalan, bir massiv elementlarini ekranga bosib chiqaradigan funksiya massiv elementlarini o'zgartirishiga hojat yo'q. Shu sababli argumentdagi massiv const sifatiga ega bo'ladi. Endi, agar dasturchi adashib, funksiya tanasida ushbu massivni o'zgartiradigan kod yozsa, kompilyator xato beradi.

Yani bizning o'zgaruvchimiz himoyalangan bo'ladi. Bu mulohazalar boshqa tipdagi const sifatli funksiya kirish parametrlariga ham tegishlidir. Pointerlar bilan const ni to'rt hil turli kombinatsiya qo'llashimiz mumkin.

1. Oddiy pointer va oddiy o'zgaruvchi (pointer ko'rsatayatgan o'zgaruvchi).
2. const pointer va oddiy o'zgaruvchi.
3. Oddiy pointer va const o'zgaruvchi.
4. const pointer va const o'zgaruvchi.

Yuqoridagilarni tushuntirib beraylik. Birinchi kombinatsiyada o'zgaruvchini hech bir narsa himoya qilmayapti. Ikkinchi holda esa o'zgaruchining qiymatini

o'zgartirsa bo'ladi, lekin pointer ko'rsatayotgan adresni o'zgartirish ta'qiqlanadi. Masalan massiv ismi ham const pointerdir. Va u ko'rsatayatgan massiv birinchi elementi-ni o'zgartirishimiz mumkin. Endi uchinchi holda pointerimiz oddiy, lekin u ko'rsatayatgan o'zgaruvchi himoyalan gandir. Va nihoyat, to'rtinchi variantda eng yuqori darajadagi o'zgaruvchi himoyasita'minlanadi. Yuqoridagi tushunchalarga misol berib o'taylik.

```
// const ifodasi va pointerlar
# include <iostream. h>
# include <ctype. h>
int countDigits(const char *); // oddiy pointer va const o'zgaruvchi
void changeToLowerCase(char *); // oddiy pointer va oddiy o'zgaruvchi
int main()
{
char m[] = "Sizni 2006 yil bilan tabriklaymiz!";
char n[] = "TOSHKENT SHAHRI. . . ";
cout << m << endl <<"Yuqoridagi satrimizda " << countDigits(m)
<< " dona son bor. " << endl << endl;
cout << n << endl << "Hammasi kichik harfda:" << endl;
changeToLowerCase(n);
cout << n << endl;
return (0);
}
int countDigits(const char * cpc) { // satrdagi sonlar (0. . 9) miqdorini
// hisoblaydi
int k = 0;
for ( ; *cpc != '\0' ; cpc++){ // satrlarni elementma-element
// ko'rib chiqishning birinchi yo'li.
if ( isdigit(*cpc) ) // <ctype. h> kutubhona funksiyasi
k++;
}
```

```
 return (k);
}
 void changeToLowerCase(char *pc) { // katta harflarni kichik harflarga
 // almashtiruvchi funksiya

 while( *pc != '\0'){ // satrlarni elementma-element
 // ko'rib chiqishning ikkinchi yo'li.
 *pc = tolower( *pc ); // <ctype. h> kutubhona funksiyasi
 ++pc; // pc keyingi harfga siljitildi
 }
 return;
}
```

Ekranda:

```
 Sizni 2006 yil bilan tabriklaymiz!
Yuqoridagi satrimizda 4 dona son bor.

TOSHKENT SHAHRI. . .
Hammasi kichik harfda:
toshkent shahri. . .
```

Yuqoridagi dasturda ikki funksiya aniqlangan. Change ToLowerCase()funksiyasining parametri juda oddiydir. Oddiy char tipidagi pointer. Ushbu pointer ko'rsatayotgan ma'lumot ham oddiydir. Ikkinchi funksiyamizda esa (countDigits()), pointerimiz oddiy, yani uning qiymati o'zgarishi mumkin, u hotiraning turli adreslariga ko'rsatishi mumkin, lekin u ko'rsatayotgan o'zgaruvchi const deb e'lon qilindi. Yani pointerimiz ko'rsatayotgan ma'lumot ayni ushbu pointer yordamida o'zgartirilishi ta'qiqlanadi. Bizda yana ikki hol qoldi, ular quyida berilgan:

const pointer va const o'zgaruvchi

const pointer va oddiy o'zgaruvchi

Birinchi holga misol beraylik.

```
. . .
int m = 88, j =77;
const int * const pi = &m; // const pointer e'lon paytida
 // initsalizatsiya qilinishi shartdir
. . .
m = 44; // To'g'ri amal
*pi = 33; // Xato! O'zgaruvchi const deb belgilandi; birinchi const
pi = &j; // Xato! Pointer const deb belgilandi; int * dan keyingi const
. . .
j = *pi; // To'g'ri. const o'zgaruvchilarning
 // qiymatlari ishlatilinishi mumkin.
. . .
```

Yuqoridagi parchada const pointer va const o'zgaruvchili kombinatsiyani ko'rib chiqdik. Eng asosiysi, const pointer e'lon qilinganda initsalizatsiya bo'lishi shart. Bu qonun boshqa tipdagi const o'zgaruvchilarga ham tegishli. Ko'rib turganimizdek,

```
*pi = 33;
```

ifodasi bilan m ning qiymatini o'zgartirishga intilish bo'ldi. Lekin bu xatodir. Chunki biz pi pointeri ko'rsatib turgan hotira adresini o'zgarmas deb pi ning e'lonida birinchi const so'zi bilan belgilagan edik. Lekin biz o'zgaruvchining haqiqiy nomi bilan - m bilan o'zgaruvchi qiymatini o'zgartira olamiz. Albatta, agar m ham o'z navbatida const sifatiga ega bo'lmasa. Yani hotira adresidagi qiymatga ikkita etishish yo'li mavjud bo'lsa, bular o'zgaruvchining asl nomi - m, va pi pointeri, bulardan biri orqali ushbu qiymatni o'zgartirsa bo'ladi, boshqasi o'rqali esa bu amal ta'qiqlanadi.

Keyin,

```
pi = &j;
```

ifoda bilan esa pi ga yangi adres bermoqchi bo'ldik. Lekin pointerimiz o'zgarmas bo'lgani uchun biz biz bu amalni bajara olmaymiz. Pointerlar va const ifodasining birga

qo'llanilishining to'rtinchi holida const pointer va oddiy hotira adresi birga ishlatilinadi. Bunga bir misol:

```
int j = 84, d = 0;
int * const Ptr = &j; // e'lon davrida initsalizatsiya shartdir
*Ptr = 100; // to'g'ri amal
Ptr = &d; // Xato! Ptr ko'rsatayatgan
 // hotira adresi o'zgartilishi ta'qiqlanadi
```

Yuqorida Ptr ko'ratayatgan adresni o'zgartirsak, kompilyator xato beradi. Aslida, massiv ismlari ham ayni shu holga misol bo'la oladilar. Massiv ismi massivning birinchi elementiga const pointerdir. Lekin u ko'rsatayotgan massiv birinchi elementning qiymati o'zgartilishi mumkin.

POINTER VA ODDIY O'ZGARUVCHILARNING EGALLAGAN ADRES KATTALIGI

Pointerlar istalgan ichki asos tipga (char, int, double. . .) yoki qollanuvchi belgilagan tipga (class, struct. . .) ko'rsatishi mumkin. Albatta, bu turli tiplar hotirada turlicha er egallaydilar. char bir bayt bo'lsa, double 8. Lekin bu tiplarga ko'rsatuvchi pointerlarning kattaligi bir hil 4 bayt. Bu kattalik turli adreslash turlariga qarab o'zgarishi mumkin, lekin bitta sistemada turli tipdagi ko'rsatkichlar bir hil kattalikga egadirlar. Buning sababi shuki, pointerlar faqat hotiraning adresini saqlaydilar. Hozirgi sistemalarda esa 32 bit bilan istalgan adresni aniqlash mumkin. Pointerlar oz'garuvchining faqat boshlangich baytiga ko'rsatadilar. Masalan, bizda double tipidagi o'zgaruvchi d bor bo'lsin. Va unga ko'rsatuchi pd pointerimiz ham e'lon qilingan bo'lsin. Pointerimiz d o'zgaruvchisining faqat birinchi baytiga ko'rsatadi. Lekin bizning d o'zgaruvchimizda pointerimiz ko'rsatayotgan birinchi baytidan tashqari yana 7 dona qo'shimcha bayti mavjud. Mana shunda pointerning tipi o'yinga kiradi. Kompilyator double tipi hotirada qancha joy egallishi bilgani uchun, pointer ko'rsatayotgan adresdan boshlab qancha baytni olishni biladi. Shuning uchun pointerlar

hotirada bir hil joy egallasa ham, biz ularga tip beramiz. void * tipidagi pointerni ham e'lon qilish mumkin. Bu pointer tipsizdir. Kompilyator bu pointer bilan * ko'rsatish operatori qo'llanilganda necha bayt joy bilan ishlashni bilmaydi. Shu sababli bu amal tayqiqlangandir. Lekin biz void * pointerini boshqa tipdagi pointerga keltirishimiz mumkin, va o'sha yangi tip bilan ishlay olamiz.

Masalan:

```
. . .
int i = 1024;
int *pi = &i, *iPtr;
void *pv;
pv = (void *) pi;
cout << *pv; // Xato!
iPtr = (int *) pv;
cout << *iPtr; // Ekranda: 1024
. . .
```

Tiplarning hotiradagi kattaligini kopsatadigan, bir parametr oladigan sizeof() (sizeof - ning kattaligi) operatori mavjuddir. Uning yordamida tiplarning, o'zgaruvchilarning yoki massivlarning kattaliklarini aniqlash mumkin. Agar o'zgaruvchi nomi berilsa, () qavslar berilishi shart emas, tip, massiv va pointer nomlari esa () qavslar ichida beriladi. Bir misol beraylik.

```
// sizeof() operatori

# include <iostream. h>
 int k;
int *pk;
 char ch;
char *pch;
 double dArray[20];
 int main()
```

```
{
 cout << sizeof (int) << " - " << sizeof k << " - " << sizeof (pk) << endl;
 // tip nomi o'zgaruvchi pointer
 cout <<sizeof (char) << " - " <<sizeof ch << " - " <<sizeof (pch) << endl;
 cout << "\nMassiv hotirada egallagan umumiy joy (baytlarda): "
 << sizeof (dArray) << endl;
 cout << "Massivning alohida elementi egallagan joy: "
 << sizeof (double) << endl;
 cout << "Massivdagi elementlar soni: "
 << sizeof (dArray) / sizeof (double) << endl;
 return (0);
}
```

Ekranda:

4 - 4 - 4

1 - 1 - 4

Massiv hotirada egallagan umumiy joy (baytlarda): 160

Massivning alohida elementi egallagan joy: 8

Mavzuga oid savollar

1. Ko‘rsatkich ta’rifini keltiring.
2. Ilova ko‘rsatkichdan qanday farq qiladi?
3. Ko‘rsatkichlar bilan bog‘liq amallarni keltiring

7.10. C++ DA STRUKTURALAR VA BIRLASHMALAR.

Reja:

1. C++ da strukturalar
2. C++ da birlashmalar

Tayanch iboralar: Strukturalar. Ma’lumot strukturalari. Struktura ko’rsatkichlari. Strukturalar bilan ko'rsatgich a'zolar. Birlashmalar va ular ustida amallar. Foydalanuvchi tomonidan aniqlangan berilganlar turi. Sinflar.

Structura nomi ko’cha manzili bu yolg’iz mohiyatga ikki ahamiyatni birlashtirish uchun tariflanishi mumkin. C++da, biz struct odamovi so’zli structurani tariflaymiz:

```
struct ko’cha manzili
{
 int uy_raqami
 string ko’cha_ismi;
};
```

Bu tariflash maydonlarisiz o’zgaruvchilarni elon qilish uchun foydalanadigan ko’cha manzilining yangi turi;

ko’cha manzili oq uy oq uy ozgaruvchisida azolar deb ataladigasan ikki qismi bor, uy_raqami va ko’cha_nomi. Siz bu nuqtalardan har bir azolarning kirishida foydalanasiz, quyidagiga o’xshagan:

oq_uy,uy_raqami = 1600; oq_uy,ko’cha_nomi = "Pennsylvania Avenue";

Structuralarga ko'rsatkichlar bu jo’shqin structura ahamiyatini ajratmasi uchun odatlangan,yangi operatordan foydalanish ko'rsatgich manzili*manzil_ko’rsatgichi yangi ko'rsatgich manzili faraz qiling manzil ko’rsatgich ko’rsatgichiga structuraning uy raqamini o’rnatishni xoxlaysiz manzil ko’rsatgich, uy raqami= 1600; // XATOLIK Baxtga qarshi,bu sintaksis xato. Bu nuqta operatori *operatoridan ko’ra ustunroq. Bu siz anglagan kompliyator oylari (manzil_ko’rsatkich,uy_raqami)=1600;//XATOLIK Manzil ko’rsatgichi ko’rsatgich bo’lishiga qaramay u structura emas. Siz operator ko’rsatgichiga va kompliyator xatolar haqida bayon qiladigan nuqtalarga etibor qila olmaysiz. O’rniga siz birinchi*operatoriga keyin nuqtaga aniq etibor berishingiz kerak (manzil_ko’rsatgich,uy_raqami)=1600;//XATOLIK. Chunki bu shunchalik odatiy holatki C+

+ning loyihachilari.

structuraning azosi ko'rsatgich bo'lishi mumkin. Bu holat odatda malumot structura qiymatlari orasida bolinganda vujudga keladi. Quyidagi misolni hisoblangTashkilotlarning ko'p sonli offislarida har bir ishchida ism va ish joylari bor.

```
struct Emploee
{
 string name;
 StreetAddress* office;
}
```

Bu erda biz ikki ishchining hisoblash idoralarini tariflaymiz:

```
StreetAddress accounting;
accounting. house_number = "1729";
accounting. street_name = "Park Avenue";
Emploee harry;
harry. name = "Smith, Harry";
harry. office = &accounting;
Emploee sally;
sally. name = "Lee, Sally";
sally. office = &accounting;
```

12-raqam structuralarning qanday aloqadorligini ko'rsatadi. Bu malumotlarni bo'lishishda muhim foyda bor. Faraz qiling hisoblash offisiga ko'cha boylab harakat qilinadi:

```
hisoblash. uy_raqami = 1720;
```

Hozir Sally va Harryning offis manzillari avtomatik ravishda malumotlardan xabar-dor qiladi.

Struktura

Ma’lumki, biror predmet sohasidagi masalani echishda undagi obektlar bir nechta, har xil turdagi parametrlar bilan aniqlanishi mumkin. Masalan, tekislikdagi nuqta haqiqiy turdagi x-absissa va y -ordinata juftligi - (x,y) ko‘rinishida beriladi. Talaba haqidagi ma’lumotlar – satr turidagi talaba familiyasi, ismi va sharifi, mutaxassislik yo‘nalish, talaba yashash adresi, butun turdagi tug‘ilgan yili, o‘quv

bosqichi, haqiqiy turdagi reyting bali, mantiqiy turdagi talaba jinsi haqidagi ma’lumot va boshqalardan shakllanadi.

C++ tilida bir yoki har xil turdagi berilganlarni jamlanmasi struktura deb nomlanadi. Struktura foydalanuvchi tomonidan aniqlangan berilganlarning yangi turi hisoblanadi. Struktura quyidagicha aniqlanadi:

```
struct <struktura nomi> // struktura nomi
 {
 <1-tur> <1-nom>; // 1-maydonni e’lon qilish
 <2-tur> <2-nom>; // 2-maydonni e’lon qilish
  . . .
 <n-tur> <n-nom>; // n-maydonni e’lon qilish
 };
```

Masalan,

```
struct Date
{
int ear;
char month, day;
};
```

Masala, Book nomli struktura yarating. Book o‘z ichiga quyidagi maydonlarni olsin:

- Muallif (*satr*);
- Nomi (*satr*);
- Nashr qilingan yili (*butun son*);
- Kitob beti (*butun son*).

Struktura ko‘rinishi quyidagicha bo‘ladi:

```
struct Book {
 char author[40]; // muallif, satrli
 char title[80]; // nomi, satrli
 int ear; // nashr qilingan yil, butun son
 int pages; // varaqlar soni, butun son
 };
```

Demak biz Book nomli struktura yaratdik. Demak, bu struktura asosida yangi o‘zgaruvchilarni e’lon qilish mumkin.

Book b; // bu erda xotira ajratiladi!

Book b1 = {“Y. Golosinskiy", “Ingliz tili … ", 2010, 576};

Bunda b va b1 o‘zgaruvchilar e’lon qilingan. Ularni tiplari esa Book. B ni hali qiymati ma’lum emas, lekin b1 ning qiymatlari berilgan.

Bundan tashqari struktura maydonlarini quyidagicha ham to‘ldirish mumkin:

Dastur orqali	Klaviatura orqali

strcpy (b. author, "Y. Golosinskiy");
strcpy (b. title, “Ingliz tili … ");
b. ear = 2010;
b. pages = 576;
	printf (“Muallif");
	gets (b. author);
	printf (“Kitob nomi");
	gets (b. title);
	printf (“Nashr qilingan yili, Betlar soni");
	scanf ("%d%d", &b. ear, &b. pages);

Demak, berilganlar asosida kutubxonadagi kitoblarni ma’lumotlarini saqlovchi kichkina dasturcha tuzamiz. Uning dasturi quyidagicha:

```
#include<iostream>
#include<conio. h>
using namespace std;
struct Book {
 char author[40]; // muallif, satrli
 char title[80]; // nomi, satrli
 int ear;          // nashr qilingan yil, butun son
 int pages;       // varaqlar soni, butun son
 };
void Kutubxona(Book); // Funksiyani e'lon qilish
int main() // asosiy dastur
{
```

```
 Book b;
 printf ("Muallifi " ); gets(b. author );
 printf ("Kitob nomi " ); gets(b. title );
 printf ("Nashr qilingan yili varaqlar soni");
 scanf ("%d%d", &b. ear, &b. pages);
 Kutubxona(b);
}
void Kutubxona(Book b) // Funksiya kodi
 {
 cout << "Kitob muallifi: " << b. author << endl;
 cout << "Kitob nomi: " << b. title << endl;
 cout << "Nashr qilingan yili: " << b. ear << endl;
 cout << "Betlar soni: " << b. pages ;
 getch();
 }
```

Struktura bilan ishlash

1. Ma’lumotlar bazasi uchun predmetli sohani tanlash va ma’lumotlar bazasining ayrim qaydlarini yozish uchun tuzilmalar taklif etish. Tanlangan tuzilma ikki yoki undan ortiq turdagi kamida beshta maydon (unsur)ga ega bo‘lishi lozim.

Masalan. ***“Davlat” tuzilmasi.***

Tuzilma element (unsur, maydon, komponent)lari:

- mamlakat nomi;
- poytaxt;
- davlat tili;
- aholi soni;
- er maydoni.

2. Standartga mos kirish oqimi (klaviatura)dan chiqadigan bir o‘lchovli tuzilmalar massivini shakllantirish uchun funksiya yozish. Tuzilmalar kiritilayotganda quyidagi mexanizmlardan birini qo‘llash mumkin:

- oldindan berilgan tuzilmalar miqdorini kiritish;

- berilgan belgili tuzilmalar paydo bo‘lgunga qadar kiritish;
- kiritish amalini davom ettirish zarurati haqida foydalanuvchi bilan dialog o‘rnatish;

3. Tuzilmalar massivi fayliga qayd etish uchun funksiya yozish.

4. Tuzilmalar massiviga fayldan o‘qish funksiyasini yozish.

5. Mavjud tuzilmalar massivini yangi tuzilmalar bilan to‘ldirish funksiyasini yozish.

6. Tanlangan unsur (element)ning berilgan belgi (znachenie)li tuzilmasini izlab topish funksiyasini qayd etish.

7. Tuzilmalar massivi tarkibini displey ekraniga sahifa ko‘rinishida chiqarish funksiyasini yozish.

8. Berilgan belgili tuzilmalarni izlab topish funksiyasini yozish (masalan, unsur qiymatlarining berilgan diapazon bo‘yicha tuzilmasini tanlash).

9. Berilgan maydon bo‘yicha tuzilmalar massivini tartibga solish funksiyasini yozish. Masalan, davlatlarni aholisi soni bo‘yicha yoki mamlakatlarni alfavit bo‘yicha tartibga keltirish.

10. Faylni to‘liq yangilash funksiyasini yozish, masalan, tuzilmalar massivi faylga tartibga keltirilgandan so‘ng qayta yoziladi.

Namoyish qilish:

- dastur tugagandan keyin ma’lumotlarni faylda saqlash (xotiraga);
- tuzilmalar majmuini turlicha tartibga keltirish;
- mos keladigan tuzilmalarni izlab topish (element belgisi, element belgisining diapazoni bo‘yicha).

3. 1-jadvalda talabalar uchun variantlar berilgan. Har bir variant uchun quyidagi amallar bajarilishi lozim:

- har bir strukturada kamida 10 tadan ma’lumot bo‘lsin;
- strukturaning ixtiyoriy maydoni bo‘yicha saralash amalga oshirilsin (satrli va sonli maydonlar uchun);
- struktura maydonlaridan ma’lumotlarni qidirsin (masalan, davlat uchun faqat osiyo qit’asidagi davlatlarni yoki ma’lum bir davlatni);

- struktura ma’lumotlarni fayldan o‘qib faylga yozsin;
- strukturani sonli maydonlari uchun ma’lum bir oraliqda saralasin (masalan, davlat strukturasida aholi soni 20 000 000 dan 50 000 000 gacha bo‘lgan davlatlarni chiqaring);
- satrli maydonlar uchun biror bir harf bilan boshlanadigan nomlarini chiqaring (masalan, davlat nomi maydonidan faqat “A” harfi bilan boshlanadigan davlatlarni).

Mavzuga oid savollar

1. Ko’rsatgich nima?
2. Adres olish amali nimaga kerak?
3. Funksiyaga ko’rsatgichlar qanday ishlatiladi?
4. Funksiya parametrlarida ko’rsatgichni ishlatish nimaga kerak?
5. Havola deb nimaga aytiladi?
6. Struktura nima?
7. Strukturaning qanday turlari bor?

7.11. C++ TILIDA GRAFIKA.

Reja:

1. Matnli rejimda ekran bilan ishlash
2. Grafik rejida ekran bilan ishlash
3. Vorislikka asoslanib grafik sinflarni yaratish

Tayanch iboralar: getch(arg), putch(arg), getchar(arg), putchar(arg), void clrscr(void) – ekranni tozalash, void gotoxy(int x, int y) – kursorni ko‘rsatilgan nuqtaga ko‘chirish, void textcolor(int c) – text rangini o‘rnatish, void textbackground (int c) – text foni rangini o‘rnatish

Matnli rejimda ekran bilan ishlash

Simvolli kiritish va chiqarish. Quyidagi funksiyalar dasturda simvollarni kiritish va chikarish uchun ishlatiladi.

getch(arg) – bitta simvol kiritilishini kutish. Kiritilayotgan simvol monitorda aks etmaydi. Bu funksiyani dastur oxirida argumentsiz ishlatilsa, monitorda ma’lumotlarni to klavisha bosilguncha o‘kish mumkin bo‘ladi.

putch(arg)- bitta simvolni standart okimga chiqarish uchun ishlatiladi. Simvol monitorda aks etmaydi.

getchar(arg) – bitta simvol kiritilishini kutish. Kiritilayotgan simvol monitorda aks etadi. Bu funksiyani dastur oxirida argumentsiz ishlatilsa, monitorda ma’lumotlarni to klavisha bosilguncha ukish mumkin bo‘ladi.

putchar(arg)- bitta simvolni standart okimga chikarish uchun ishlatiladi. Simvol monitorda aks etadi.

Bu funksiyalar **stdio. h** modulida joylashgandir.

Misol:

```
#include <stdio. h>
int main()
{
int c;
c=getchar();
putchar(c);
getch();
```

```
return 0;
}
```

Ekran bilan ishlovchi funksiyalar. Quyidagi funksiyalar matnli rejimda ekran bilan ishlashga mo‘ljallangan.

void clrscr(void) – ekranni tozalash

void gotoxy(int x, int y) – kursorni ko‘rsatilgan nuqtaga ko‘chirish

void textcolor(int c) – text rangini o‘rnatish

void textbackground (int c) – text foni rangini o‘rnatish

Bu funksiyalar **conio. h** modulida joylashgandir.

Ekran bilan ishlashga misol.

Do‘stona funksiyaga ega bo‘lgan sinfga misol:

```
#include <conio. h>
class charlocus
{
int x,y;
char cc;
friend void friend_put (charlocus p, char c);
public:
charlocus (int xi, int yi, char ci)
{
x=xi; y=yi; cc=ci;
}
void display (void)
{
gotoxy(x,y); putch(cc);
}
};
void friend_put(charlocus p, char c)
{
p. cc=c;
```

```
}
int main ()
{
charlocus D(20,4, 'd');
charlocus S(10,10, 's');
clrscr( );
D. display( ); getch( ); S. display( ); getch( );
friend_put(D, 'x'); D. display( ); getch( );
friend_put(S, '#'); S. display( ); getch( );
return 0;
}
```

Dasturda D va S obektlari yaratilib ular uchun ekranda kordinatalar va (d,s) simvollari aniqlanadi. So‘ng sinf funksiyasi charlocus : : display ( ) simvollarni ko‘rsatilgan pozitsiyaga chiqaradi. Global friend_put funksiyasi simvollarning o‘rnini almashtirib qo‘yadi.

Grafik rejida ekran bilan ishlash

Grafik biblioteka. Turbo C va Borland C++ kompilyatorlarida grafik biblioteka bilan bog‘lanish uchun graphic. h – sarlavxali fayl qo‘llaniladi.

Bu bibliotekaga kiruvchi ba’zi grafik funksiyalar:

void initgraph(int* graphdriver, int* graphmode, char* pathtodriver)- grafik rejimga o‘tkazish

void closegraph(void)-grafik rejimdan matnli rejimga o‘tkazish.

void putpixel(int x, int y, int color) - Ekranda color rangli(x,y) kordinatali nuqtani tasvirlaydi.

void line (int x1, int y1, int x2, int y2) - Ekranda chiziq chizadi chizadi.

void rectangle (int left, int top, int right, int bottom) - Ekranda to‘rtburchak chizadi.

void circle (int x, int y; int radius) - Ekranda aylana chizadi.

void ellipse (int x, int y, int stangle, int endangle, int xradius, int yradius) - Ekranda ellips chizadi.

void outtextxy (int x, int y, char* textstring) – Textni berilgan pozitsiyada chiqaradi.

void outtext (char* textstring) – Textni joriy pozitsiyada chiqaradi.

int getbcolor(void) - Fon rangini qaytaradi

int getcolor(void) - Tasvir rangini qaytaradi.

void getimage (int left, int top, int right, int bottom, void* bitmap) - ekran oynasini xotirada saqlash;

putimage (int left, int top, void* bitmap, int op)- xotirada saqlangan tasvirni ekranga joylash;

Grafik sinfga misol. Misol tariqasida nuqta tushunchasini aniqlovchi point sinfini yaratib point. cpp fayliga yozib qo‘yamiz:

```
#ifndef POINTH
#define POINTH 1
class point
{
protected:
int x,y;
public:
point ( int xi=0, int yi=0);
int& givex(void);
int& givey (void);
void show(void);
void move(int xn=0, int yn=0);
private:
void hide();
}
#endif
```

Kelgusida point sinfini boshqa sinflarga qo‘shish mumkin bo‘lgani uchun shartli protsessor direktivasi #ifndef POINTH ishlatilgan. Protsessorli identifikator POINTH #define POINT 1 direktivasi orqali kiritilgan.

Shuning uchun #include "point. h" direktivasi bir necha marta ishlatilganda ham POINT sinfi ta’rifi teksti faqat bir marta kompilyatsiya kilinayetgan faylda paydo bo‘ladi.

POINT sinfi komponenta funksiyalarini quyidagicha ta’riflaymiz:

```
ifdef POINTCPP
#define POINTCPP 1
#include <graphics. h>
#include "point. h"
point ::point(int xi=0, int yi=0)
{
x=xi; y=yi;
}
int &point::givex(void)
{
return x;
}
int &point::givey(void)
{
return y;
}
void point::show(void)
{
putpixel(x,y, getcolor());
}
void point::hide (void)
{
putpixel(x,y,getbcolor());
}
void point::move(int xn=0, int yn=0)
{
```

```
hide( );
x=xn; y=yn;
show( );
}
#endif
```

Kiritilgan point sinfi va komponenta funksiyalari bilan ishlovchi dasturni keltiramiz:

```
#include <graphics. h>
#include <conio. h>
#include "point. cpp"
int main()
{
point A(200,50);
point B;
point D(500,200);
int dr=DETECT, mod;
initgraph(&dr, &mod, "c:\\borlandc\\bgi");
A. show();getch();
B. show();getch();
D. show();getch();
A. move();getch();
B. move(50,60);getch();
closegraph();
return 0;
}
```

Qo‘shimcha yuklash. Misol tarikasida '+' amalini point sinfi bilan ta’riflanuvchi displeydagi nuqtalarga qo‘llaymiz. Soddalashtirish uchun point sinfida eng kerakli kompanentalarni qoldiramiz.

```
#include <graphics. h>
#include <conio. h>
```

```
class point
{
protected:
int x,y;
public:
point (int xi=0, int yi=0)
{
x=xi; y=yi;
}
void show (void)
{
putpixel (x,y,get color ( ) );
};
point operator+ (point p);
};
point point : : operator + (point &p)
{
point d;
d. x=this. x+p. x;
d. y=this. y+p. y;
return d;
}
int main()
{
int dr=DETECT, mod;
point A(200,50);
point B;
point D(50,120);
INITGRAPH (&dr, &mod, "C\\borlandc \\ bgi");
A. show ( );
```

```
getch ( );
B. show ( ); getch ( );
D. show ( ); getch ( );
B=A+D
;
B. show ( ); getch ( );
B=A. operator + (B);
B. show ( ); getch ( );
closegraph ( );
return 0;
}
```

Dastur bajarilishi natijasida displey ekraniga ketma-ket quyidagi nuqtalar qo‘yiladi: A(200,50); B(0,0); D(50,120); B(250,70), B(450,220)

Lokal sinf. Lokal sinflardan foydalanish xususiyatlarini tushuntirish uchun quyidagi masalani ko‘rib chiqamiz. «Kvadrat» sinfini aniqlash kerak bo‘lsin. Kvadrat tomonlarini koordinatalar o‘qiga parallel deb qaraymiz. Har bir kvadrat berilganligi sifatida markaz koordinatalari va tomon uzunligi olinadi. Kvadrat sinfi ichida «kesma lokal»sinfini aniqlaymiz. Har bir kesmani berilganlari sifatida uchlarining koordinatalarini olamiz.

Uchlari mos ravishda olingan to‘rtta kesma kvadratni xosil qiladi. Shu usulda kvadrat ekranda tasvirlanadi.

```
#include<conio. h>
#include "point. cpp"
class square
{class segment{
point pn, pk;
public:
segment(point pin=point(0,0), point pik=point(0,0))
{
pn. givex()=pin. givex();
```

```
pn. givey()=pin. givey();
pk. givex()=pik. givex();
pk. givey()=pik. givey();
}
point &beg(void)
{return pn;}
point &end (void)
{return pk;}
void showSeg()
{
line(pn. givex(), pn. givey(), pk. givex(), pk. givey());
};
};
segment ab,bc,cd,da;
public:
square(point ci=point(0,0), int di=0)
{
point a,b,c,d;
a. givex()=ci. givex()-di/2;
a. givey()=ci. givey()-di/2;
b. givex()=ci. givex()+di/2;
b. givey()=ci. givey()-di/2;
c. givex()=ci. givex()+di/2;
c. givey()=ci. givey()+di/2;
d givex()=ci. givex()-di/2;
d. givey()=ci. givey()+di/2;
ab. beg()=a; ab. end()=b;
bc. beg()=b; bc end()=c;
cd. beg()=c; cd. end()=d;
da. beg()=d; da end()=a;
```

```
}
void showsquare(void)
{
ab. showSeg();
bc. showSeg();
cd. showSeg();
da. showSeg();
}
};
int main()
{ int dr=DETECT,mod;
initgraph(&dr,&mod,"c:\\borlondc\\ bgi");
point p1(80,120);
point p2(250,240);
square A(p1,30);
square B(p2,140);
A. showsquare(); getch();
B. showsquare(); getch();
closegraph();return 0;}
```

Vorislikka asoslanib grafik sinflarni yaratish

Grafik sinflarda vorislikkka misol. Keyingi misolda " ekrandagi nuqta" asosida " ekrandagi darcha " sinfi yaratiladi.

Nasldan o‘tuvchi komponentalarga qo‘shimcha spot sinfiga quyidagi kompanentalarni kiritamiz: tasvir radiusi (rad); ekranda ko‘rinishi belgisi (vir=0 ekranda tasvir yuk; vi1==1 ekranda tasvir bor); tasvirni bitli matnda saqlash belgisi (tag=0 tasvir saqlanmaydi; tag==1 tasvir saqlanadi); tasvirni bitli matnda saqlash uchun ajratilgan xotira qismiga ko‘rsatgich pspot.

```
//Spot. cpp
#ifndef SPOT
#define SPOT 1
```

```
#include "point. cpp"
class spot: public point
{
int rad;
int vil;
int tag;
void * pspot;
public:
spot (int xi, int yi, int ri): point (xi, yi)}
{
int size;
vir =0; tag=0; rad=ri;
size=imagesize (xi-ri, yi-ri, xis- ri, yi- ri);
pspot=new char [size];
}
~ spot ()
{
hide();
tag =0;
delete[] pspot;
}
void show ()
{
if (tag==0)
{
circle (x, y, rad);
floodfill (x, y, getcolor ());
getimage (x-rad, y-rad, y+rad, pspot);
tag=1;
};
```

```
else
putimage (x-rad, y-rad, pspot, XOR_PUT);
vis=1;
}
void hide ()
{
if (vis==0) return;
putimage (x-rad,y-rad, pspot, XOR_PUT);
vis=0;
}
void move (int xn, int yn)
{
hide ();
x= xn; y=yn;
show ();
}
viod vary (float dr)
{
float a;
int size;
hide ();
tag=0;
delete pspot;
a=dr*rad;
if (a<=0) rad=0;
else rad= (int) a;
size=imagesize (x-rad; y-rad, x+rad, y+rad);
pspot=new char [size];
show ();
}
```

```
int& giver (void);
{
return rad;
}
};
# endif
```

Spot sinfida konstruktor, destruktor ~ spot () va beshta metod ko‘rsatilgan:

show ()-- ekranga doirani chizib, bitli tasvirni xotiraga olish;

hide ()-- ekrandan doira tasvirini uchirish;

move ()--tasvirni ekranning bitta joyiga kuchirish;

vary ()--ekrandagi tasvirni uzgartirish (kichkinalashtirish yoki kattalashtirish);

giver () --doira radiusiga murojatni ta’minlash;

point sinfidan spot sinfi naslga nuqta markazi (x,u) koordinatalarini va givex, givey metodlarni oladi, point : : show () va point : : move () metodini xuddi shu nomli yangi funksiyalar bilan almashtirilgan point :: hide() funksiyasi nomi o‘tmaydi, chunki point sinfida u xususiy (private) statusiga ega. spot() konstruktori uch parametrga ega -markaz koordinatalari (xi,yi) va doira radiusi (ri).

Avval point sinfi konstruktori chaqiriladi bu konstruktor xi,yi ga mos keluvchi xaqiqiy parametr asosida doira markazini aniklaydi. Asosiy sinf konstruktori har doim xosilaviy sinf konstruktoridan oldin chaakiriladi. Sungra spot() sinfi konstruktolari boshlanadi. Bu konstruktor vis, tag parametrlarining boshlang‘ich qiymatini aniklaydi va ri ga mos keluvchi xaqiqiy parametr qiymati asosida doira radiusi red aniklanadi. Standart funksiya imagesize yordamida doira joylashuvchi kvadratik operativ xotirada aniqlash uchun zarur bulgan xotira xajmi hisoblanadi. Kerakli xotira new standart operatsiya yordamida ajratib size elementidan iborat char massivlar yoziladi. Agar aytilgan xotira spot sinfida protected statutisiga ega bulgan pspot ko‘rsatkichiga ulanadi.

Vorislikda destruktorlar xossalari. Sinfning har bir obekti yaratilganda sinf konstruktori chaqirilib, obekt uchun kerakli xotira yaratish va initsializatsiya qilish vazifalarini bajaradi. Obekt yukotilganda yoki sinf ta’sir doirasidan tashqariga

chiqilganda teskari vazifalar bajarish kerak bo‘lib, bular ichida eng keraklisi xotirani ozod kilishdir. Bu vazifalarni boshkarish uchun sinfga maxsus funksiya destruktor kiritiladi. Destruktor quyidagi shaklga ega bo‘lgan anik nomga ega ~ sinf-nomi.

Destruktor xatto void tipidagi parametrlarga ega bo‘lmaydi va xatto void tipidagi qiymat qaytarmaydi. Destruktor statusi aloxida e’lon qilinmagan bo‘lsa umumiydir.

Sodda sinflarda destruktor avtomatik aniklanadi, misol uchun paint sinfida destruktor e’lon kilinmagan va kompilyator quyidagi destruktorni avtomatik chakiradi

~ point () {};

spot sinfida destruktor aniq ko‘rinishga ega;

~ spot () {hide (); tag=0;delete [] pspot;}

Bu destruktor vazifalari doira tasvirini spot::hide() funksiyasi orkali o‘chirish; tag belgisiga 0 qiymatini berish; obekt bitli tasvirni saklash uchun ajratilgan xotirani tozalash.

Destruktorlar naslga o‘tmaydi, shuning uchun xosilaviy sinfda destruktor mavjud bo‘lmasa asosiy sinfdagi destruktor chaqirilmaydi. Balki kompilyator tomonidan yaratiladi. Ko‘rilayotgan misolda quyidagicha:

public: ~spot () {~point ();}

Asosiy sinflar destruktorlar ro‘yxatda ko‘rsatilganidek teskari tartibda boshqariladi. Shunday qilib obektlarni o‘chirish tartibi yaratilish tartibiga teskaridir. Agar obekt yaratilganda dasturda xotira ajratilgan bulsa destruktor dasturda chaqirilishi lozim.

Spot sinfi obektlari bilan ishlovchi dasturni keltiramiz:

```
#include<graphics. h>
#include<conio. h >
#include "spot. cpp"
int main()
{
int dr=DETECT, mod;
initgraph (&dr, &mod);
{
```

```
spot A(200,50,20);
spot D(500,200,30);
A. show();
getch ();
D. show ();
getch();
A. move(50,60);
getch ();
}
closegraph();
return 0;
}
```

Grafik abstrakt sinf va uning vorislariga misol. Quyidagi misolda abstrakt sinflar umumiy tushunchalarni tavsiflash uchun ishlatiladi point sinfi vorisi sifatida figure abstrakt sinfi yaratiladi. Bu sinfda konstruktor, sof virtual funksiya show (), hamda hide() va move() metodlari aniqlangan. Dasturda figure sinfi asosida ikkita avlod sinf circle (aylana) va ellips (elips)sinflari aniqlanadi.

Ikkala sinfda point sinfidan shakllar markazlari koordinatalari naslga o‘tgan. Ikkala sinfda konkret show() metod aniqlangan va figure () abstrakt sinfidan move() va hide() funksiyalari naslga o‘tgan.

```
// figure. cpp abstrakt sinf
# include "point. cpp"
class figure: public point
{
public :
//figure sinf konstruktori
figure (point p) :point (p. give x(), p. give y()){}
// sof virtual funksiya
virtual void show() = 0;
// figurani o‘chirish funksiyasi
```

```
void hide()
{
int bk,cc;
bk = getbkcolor();
cc = getcolor();
setcolor(bk);
show();
setcolor(cc);
}
// figurani harakatlantirish funksiyasi
void move (point p)
{
hide ();
x=p. givex();y=p. givey();
show();
}
};
```

figure abstrakt sinf asosida konkret sinflar yaratamiz;

```
// ELLIPS CPP
class ellips: public figure
{
int rx, ry;
public:
// konstruktor
ellips(point d, int radx, int rady): figure (d)
{
rx = radx; ry = rady;
}
void show()
{
```

```
ellipse (x,y,0, 360, rx, ry);
return;
}
};
// circ. fig aylana sinfi
class circ: public figure
{
int radius;
public:
// konstruktor
circ (point e, int rad): figure (e)
{
radius=rad;
}
void show () {
circle(x,y,radius);
}
};
```

quyidagi dasturda uchta sinf hammasi ishlatilgan;

```
# include <graphics. h>
# include "figure. cpp"
# include "circ. fig"
# include "ellips. fig"
# include <conio. h>
int main ()
{
point A(100,80), B(300,200);
circ C(A,60);
ellips E(B,200,100);
{
```

```
int dir = DETECT, mod;
initgraph (&dir, &mod, "c\\borlandc\\ bgi");
A. show (); getch();
B. show (); getch();
C. show (); getch();
E. show (); getch();
C. move(B); getch();
E. hide(); getch();
C. hide(); getch();
}
closegraph();
return 0;
}
```

Mavzuga oid savollar

1. Grafik funksiyalar qaysi bibliotekada saqlanadi.
2. Grafik rejimga o‘tish komandasi.
3. Grafik rejimdan chiqish komandasi
4. Ekranda chiziq chizish funksiyasi
5. Ekranda aylana va ellips chizish funksiyasi.
6. Ekran qismini qanday qilib xotirada saqlash mumkin?

7.12. C++ DA FAYLLAR BILAN ISHLASH.

Reja:

1. Fayllar bilan ma’lumot almashish
2. O‘qish va yozish operatsiyalarining bajarilishi
3. Faylning kerak bo‘lmay qolganda berkitilishi
4. Fayl operatsiyalarini bajarishda xatolarni tekshiris
5. Fayl oxirini aniqlash
6. Fayllar bilan ishlash sinflari.

Tayanch iboralar: Matn fayllarini o'qish va yozish. Oqimni ochish. Fayldan o'qish. Faylga yozish. Binary fayllar bilan ishlash operatorlari. Matn va binar fayllar. O‘qish-yozish oqimlari. Standart oqimlar. Belgilarni o‘qish-yozish funksiyalari. Satrlarni o‘qish - yozish funksiyalari. Fayldan o‘qish-yozish funksiyalari. Formatli o‘qish va yozish funksiyalari. Fayl ko‘rsatkichini boshqarish funksiyalari.

Fayllar bilan ishlash sinflari. C++da fayllar bilan ishlash fstream kutubxonasidagi biron-bir sinflar yordamida amalga oshiriladi.

fstream kutubxonasi fayllarni o‘qib olish uchun javob beradigan ifstream sinfiga, hamda faylga axobotning yozib olinishiga javob beradigan ofstream sinfiga ega.

Biron-bir faylni yozish yoki o‘qish uchun ochish uchun, ofstream turdagi o‘zgaruvchini yaratib, initsiallashda fayl nomidan foydalanish lozim:

ofstream file_object("FILENAME. EXT");

Agar fayl mavjud bo‘lmasa,yangidan yaratiladi va oqimga ulanadi. Agar fayl mavjud bo‘lsa u o‘chiriladi, va bo‘sh fayl yangidan yaratiladi.

Agar fayl ham dasturning bajarilayotgan fayli joylashtirilgan papkada bo‘lsa, u holda faylning nomi to‘liq ko‘rsatilmasligi mumkin (faqat fayl nomi, unga borish yo‘lisiz). Bundan tashqari fayl nomini to‘g‘ridan-to‘g‘ri ko‘rsatish o‘rniga, uning nomidan iborat belgilar massivlarini ko‘rsatish mumkin.

```
char s[20]= "C:\text. txt";
ofstream file_object (s);
```

Quyidagi dasturda faylga uch qator ma’lumot yoziladi:

```
#include <fstream. h>
int main()
 {
    ofstream book_file("BOOKINFO. DAT");
```

```
        book_file <<"C++ tilida dasturlashni o’rganamiz" << endl;
        book_file << "Jamsa Press" << endl;
        book_file << "22. 95" << endl;
  return 0;
  }
```

Biron-bir faylni yozish yoki o‘qish uchun ochish uchun, ifstream turdagi o‘zgaruvchini yaratish kerak.

```
ifstream input_file("filename. EXT");
```

Bunday fayl mavjud bo‘lmasa oqim yaratilmaydi. Quyidagi dasturda fayldan uch qator ma’lumot o‘qiladi:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
        ifstream input_file("BOOKINFO. DAT") ;
        char one[64], two[64], three[64];
        input_file >> one;
        input_file >> two;
        input_file >> three;
        cout << one << endl;
        cout << two << endl;
        cout << three << endl;
  return 0;
}
```

Satr haqida gap ketganda, chiqarish satr oxiri belgisi, ya’ni \n paydo bo‘lishidan oldin amalga oshiriladi. Belgisiz turga ega bo‘lgan barcha o‘zgaruvchilar oldin belgilarga o‘zgartirib olinadi.

Axborotni fayldan o‘qib olish uchun >> operatoriga ekvivalent bo‘lgan get funksiyasi qo‘llanadi. Bu funksiya har qanday o‘zgaruvchilarning standart turlari yoki

belgilar massivlari bilan ishlay oladi. Shuningdek get ga har jihatdan ekvivalent bo‘lgan getline funksiyasi mavjud: farqi faqat shundaki, getline funksiyasi satr oxiridani oxirgi belgini qaytarmaydi.

Butun satrni fayldan o‘qib olish uchun getline usulidan foydalanish qulaydir:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
        ifstream input_file("BOOKINFO. DAT");
        char one[64], two[64], three[64];
        input_file. getline(one, sizeof(one));
        input_file. getline(two, sizeof(two));
        input_file. getline(three, sizeof(three));
        cout << one << endl;
        cout << two << endl;
        cout << three << endl;
return 0;
}
```

Fayl oxirini aniqlash. Fayl ichidagisini, fayl oxiri uchramaguncha, o‘kish dasturdagi oddiy fayl operatsiyasi hisoblanadi. Fayl oxirini aniqlash uchun, dasturlar oqim obektining eof funksiyasidan foydalanishlari mumkin. Agar fayl oxiri hali uchramagan bo‘lsa, bu funksiya 0 qiymatini qaytarib beradi, agar fayl oxiri uchrasa, 1 qimatini qaytaradi. While siklidan foydalanib, dasturlar, fayl oxirini topmagunlaricha, quyida ko‘rsatilganidek, uning ichidagilarini uzluksiz o‘qishlari mumkin:

```
while (! Input_file. eof())
{
 //Operatorlar
}
```

Ushbu holda dastur, eof funksiyasi yolg‘on (0) ni qaytarguncha, siklni bajarishda davom etadi. Navbatdagi dastur BOOKINFO. DAT fayli ichidagisini, fayl oxiriga yetmaguncha, o‘kish uchun eof funksiyasidan foydalanadi.

```
#include <iostream>
#include <fstream>
using namespace std;
int main ()
{
 ifstream input_file("BOOKINFO. DAT");
 char line[64];
 while (! input_file. eof())
 {
 input_file. getline(line, sizeof(line));
 cout << line << endl;
 }
return 0;
}
```

Xuddi shunday, keyingi dastur - fayl ichidagisini bitta so‘z bo‘yicha bir martada, fayl oxiri uchramaguncha, o‘qiydi:

```
#include <iostream. h>
#include <fstream. h>
int main ()
{
 ifstream input_file("BOOKINFO. DAT");
 char word[64];
 while (! input_file. eof())
 {
 input_file >> word;
 cout << word << endl;
 }
```

```
return 0;
}
```

Va, nihoyat, keyingi dastur - fayl ichidagisini bitta belgi bo‘yicha bir martada get funksiyasidan foydalanib, fayl oxiri uchramaguncha, o‘qiydi:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
 ifstream input_file("BOOKINFO. DAT");
 char letter;
 while (! input_file. eof())
 {
 letter = input_file. get();
 cout << letter;
 }
return 0;
}
```

Fayl operatsiyalarini bajarishda xatolarni tekshirish. Hozirgacha taqdim etilgan dasturlarda ko‘zlanganidek, fayl operatsiyalarini bajarishda xatolar sodir bo‘lmaydi. Afsuski, bunga hamma vaqt ham erishib bo‘lmaydi. Masalan, agar siz kiritish uchun fayl ochayotgan bo‘lsangiz, dasturlar ushbu fayl mavjudligini tekshirib ko‘rishi kerak. Xuddi shunday, agar dastur ma’lumotlarni faylga yozayotgan bo‘lsa, operatsiya muvaffakyatli o‘tganiga ishonch hosil qilish kerak (masalan, diskda bo‘sh joyning yo‘qligi ma’lumotlarning yozib olinishiga to‘sqinlik qiladi). Xatolarni kuzatib borishda dasturlarga yordam berish uchun, fayl obektining fail funksiyasidan foydalanish mumkin. Agar fayl operatsiyasi jarayonida xatolar bo‘lmagan bo‘lsa, funksiya yolg‘on (0) ni qaytaradi. Biroq, agar xato uchrasa, fail funksiyasi haqiqatni qaytaradi. Masalan, agar dastur fayl ochadigan bo‘lsa, u, xatoga yo‘l qo‘yilganini aniqlash uchun, fail funksiyasidan foydalanishi kerak. Bu quyida shunday ko‘rsatilgan:

```
ifstream input_file("FILENAME. DAT");
if (input_file. fail())
{
 cerr << "Ochilish xatosi FILENAME. EXT" << endl;
 exit(1);
}
```

Shunday qilib, dasturlar o‘qish va yozish operatsiyalari muvaffaqiyatli kechganiga ishonch hosil qilishlari kerak. Quyidagi dasturda turli xato vaziyatlarni tekshirish uchun fail funksiyasidan foydalanadi:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
 char line[256] ;
 ifstream input_file("BOOKINFO. DAT") ;
 if (input_file. fail()) cerr << "Ochilish xatosi BOOKINFO. DAT" << endl;
 else
 {
 while ((! input_file. eof()) && (! input_file. fail()))
 {
 input_file. getline(line, sizeof(line)) ;
 if (! input_file. fail()) cout << line << endl;
 }
 }
return 0;
}
```

Faylning kerak bo‘lmay qolganda berkitilishi. Dasturni tugallash uchun operatsiya tizimi o‘zi ochgan fayllarni berkitadi. Biroq, odatga ko‘ra, agar dasturga

fayl kerak bo‘lmay qolsa, uni berkitishi kerak. Faylni berkitish uchun dastur, quyida ko‘rsatilganidek, dastur close funksiyasidan foydalanishi kerak:

input_file. close ();

Faylni yopayotganingizda, dastur ushbu faylga yozib olgan barcha ma’lumotlar diskka tashlanadi va ushbu fayl uchun katalogdagi yozuv yangilanadi.

O‘qish va yozish operatsiyalarining bajarilishi. Hozirga qadar gap borayotgan dasturlar belgili satrlar ustida operatsiyalar bajarar edi. Dasturlaringiz murakkablashgan sari, ehtimol, sizga massivlar va tuzilmalarni o‘qish va yozish kerak bo‘lib qolar. Buning uchun dasturlar read va write funksiyalaridan foydalanishlari mumkin. read va write funksiyalaridan foydalanishda ma’lumotlar o‘qiladigan yoki yozib olinadigan ma’lumotlar buferini, shuningdek buferning baytlarda o‘lchanadigan uzunligini ko‘rsatish lozim. Bu quyida ko‘rsatilganidek amalga oshiriladi:

input_file. read(buffer, sizeof(buffer));

output_file write(buffer, sizeof(buffer));

Masalan, quyidagi dasturda tuzilma ichidagisini EMPLOYEE. DAT fayliga chiqarish uchun, write funksiyasidan foydalanadi:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
 struct employee
 {
 char name[64];
 int age;
 float salary;
 } worker = { "Djon Doy", 33, 25000. 0 };
 ofstream emp_file("EMPLOYEE. DAT");
 emp_file. write((char *) &worker, sizeof(employee));
return 0;
```

```
}
```

Odatda write funksiyasi belgilar satriga ko‘rsatkich oladi. (char*) belgilari turlarga keltirish operatori bo‘lib, bu operator siz ko‘rsatkichni boshqa turga uzatayotganingiz haqida kompilyatorga axborot beradi. Xuddi shunday tarzda quyidagi dasturda read usulidan xizmatchi haqidagi axborotni fayldan o‘qib olish uchun foydalanadi:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
 struct employee
 {
 char name[64] ;
 int age;
 float salary;
 } worker = { "Djon Doy", 33, 25000. 0 };
 ifstream emp_file("EMPLOYEE. DAT");
 emp_file. read((char *) &worker, sizeof(employee));
 cout << worker. name << endl;
 cout << worker. age << endl;
 cout << worker. salary << endl;
return 0;
}
```

Fayllar bilan ma’lumot almashish. Fayllar bilan ishlovchi oqimlar quyidagi sinflar obektlari sifatida yaratiladi:

ofstream –ma’lumotlarni faylga yozish uchun ;

ifstream –fayldan ma’lumotlarni o‘qish uchun;

fstream – ma’lumotlarni o‘qish va yozish uchun.

Bu sinflardan foydalanish uchun dastur matniga yordamchi sarlovhali fayl fstream. h qo‘shilishi lozim. Shundan so‘ng faylli oqimlarni quyidagicha aniqlash mumkin :

ofstream out_file;

ifstream in_file ;

fstream io_file;

Faylli oqimli yaratishdan so‘ng konkret faylga open komponenta funksiyasi yordamida ulanish mumkin. Bu funksiya quyidagi ko‘rinishga ega:

void open (const char *filename, int mode=ko‘zda tutilgan qiymat

int protection =qo‘zda tutilgan qiymat)

Birinchi parametr filename mavjud yoki yaratilayotgan fayl nomi, ikkinchi parametr mode –fayl bilan ishlash rejimlari ko‘rsatuvchi belgilar dizyunksiyasi, uchinchi parametr protection (ximoya) –kam ishlatiladi. To‘g‘rirog‘i dasturchi uchun ko‘zda tutilgan qiymati yetarlidir.

Fayl bilan ishlash rejimlari belgilari quyidagicha aniqlanadi:

enum ios:: open _mode {

in = 0x01 //faqat o‘qish uchun ochish:

out =0x02//faqat yozish uchun ochish:

ate=0x04 //ochilganda fayl oxirini izlash:

app=0x08 //ma’lumotlarni fayl oxiriga qo‘shish:

trunc=0x10 //mavjud fayl o‘rniga yangisini yaratish:

nocreate =0x20//yangi fayl ochilmasin (fayl mavjud bo‘lmasa open funksiyasi xato xaqida ma’lumot beradi)

noreplace=0x40//mavjud fayl ochilmasin

binary=0x80//ikkilik (matnli emas) almashinuv uchun ochilsin.

open funksiyasini chaqirish quyidagicha amalga oshiriladi

Oqim_ nomi open(fayl nomi,rejim, ximoya)

Misollar:

outFile. open(“C:\\user\\result. dat”);

inFile. open(“Data. txt”);

ioFile. Open(“Chance. dat”, ios::out);

Oqim ofstream sinfiga tegishli bo‘lsa, ikkinchi parametr ios:out qiymatga ega bo‘ladi.

Ochish open() funksiyasining mufaqqiyatli bajarilganligini tekshirish uchun ortiqcha yuklangan ! amalidan foydalaniladi. Agar xato mavjud bo‘lsa natija 0 dan farqli bo‘ladi. Misol uchun :

if(!int file)

```
{
cerr <<”faylni ochishda xato:\n”;
exit(1);
}
```
fstream sinfga tegishli oqimlar uchun ikkinchi parametr aniq ko‘rsatilishi shart.
Misol keltiramiz :
```
#include <iostream>
#include <fstream>
using namespace std;
const int lenname =13;
const int lenstring =60;
int main()
{
char source[lenname];
cout<<"\n fayl nomini kiriting:";
cin>>source;
ifstream infile;
infile. open(source);
if (!infile)
{
cerr<<"\n hato" <<source;
exit(1);
}
char string [lenstring];
char next;
cout<<"\n fayl matni :\n \n";
cin. get();
while (1)
{
infile >> string;
next= infile. peek();
```

```
if (next==EOF) break;
cout << string <<" ";
if (next=='\n')
{
cout<<'\n';
static int i=4;
if ( !(++i % 20))
{
cout<<"\n ENTER bosing \n " << endl;
cin. get();
} }
}
return 0;
}
```

Dastur ishlashi natijasi ekranga matnli faylni saxifalab chiqarishdan iborat. Saxifa 20 qatordan iborat.

TESTLAR BANKI

1. Dasturlarni yaratish va ularni taxlash usullari va yo'llari uchun kerakli metodlar sistemasiga. . .

A) Dasturlash texnologiyasi deyiladi;

B) Amaliy dasturlar paketi deyiladi;

C) Ma'lumot bazasini yaratish gexnologiyasi deyiladi; D) Kasbiy grafika paketi deyiladi.

2. Dasturlar yaratishda asosan qaysi texnologiyalar ishlatiladi?

A) Yuqoridan pastga qarab texnologiyasi va pastdan yuqoriga qarab texnologiyasi;

B) Informasion texnologiyalar; C) Pedagogik texnologiyalar; D) Innovasion texnologiyalar;

3. Algoritmlar qanday yoziladi?

A) So'zlar, sxemalar (blok-sxemalar) yordamida, komandalar yordamida.

B) Formulalar yordamida. Davlat tilida.

C) Ingliz tilida, davlat tilida.

D) 2-lik yoki 10-lik sanoq sistemasida.

4. EXM algoritmdagi mantiqiy xatoni topa oladimi?

A) Yo'q

B) Ha

C) To'g'ri javoblar keltirilmagan

D) Javob bera olmayman.

5. EXM bajaradigan har bir algoritm

A) Chekli kadamlarda tugashi kerak. B) Tugallanishi shart emas

C) Juft kadamlarda tugallanishi kerak D) Toq qadamlarda tugallanishi kerak

6. Butun tipli uzgaruvchilarga

A) Butun qiymatlar mos keladi. B) Haqiqiy kiymatlar mos keladi

C) Butun va haqiqiy kiymatlar mos keladi. D) Mantiqiy qiymatlar mos keladi

7. Dasturdagi xatoliklarni topish, aniklash va ularni tuzatishga. .

A) Taxlash B) Xujjatlashtirish C) Tanlash D) Modellashtirish deyiladi

8. Qattiq S: diskning TURBO katalogida joylashgan Turbo. exe faylni yuklash buyrug'ini ko'rsating

A) C:\>Turbo\ Turbo. exe kiritish (Enter) B) C:\> Turbo. exe kiritish (Enter)

C) C:/> Turbo. exe kiritish (Enter) D) C:/>Turbo/ Turbo. exe kiritish (Enter)

*9. Kuyidagi shablon *. exe nimani bildiradi*

A) Kengaytimasi exe bulgan barcha fayllarni

B) Nomi bitta xarfdan iborat, kengaytimasi exe bulgan barcha fayllarni

C) shablon noto'gri yozilgan D) Xamma javoblar tugri

10. Qaysi buyruq ekranni informasiyadan tozalaydi?

A) clear B) vol C) rem D) cls

11. Dasturda o'zgarmaslar bo'limi qaysi so'z bilan boshlanadi?

A) const B) var C) type D) label

12. Dasturda o'zgaruvchilar bo'limi qaysi so'z bilan boshlanadi?

A) var B) type C) const D) label

13. Dasturda turlar bo'limi qaysi so'z bilan boshlanadi?

A) type B) var C) const D) label

14. Belgilar (nishonalar) bo'limi qaysi so'z bilan boshlanadi?

A) label B) var C) type D) const

15. Dasturni bajarish buyrug'i qaysi?

A) run B) edit C) save D) comple

16. Parametrli sikl operatori xizmat so'zlari qaysi javobda to'gri keltirilgan?

A) for, to, do; B) if, then, else C) goto D) while, do

17. Object Pascal tilida operatorlar bir-biridan qaysi belgi bilan ajratiladi?

A) nuqta vergul B) vergul C) ikki nuqta D) probel

18. Dastur natijasi nimaga teng?

```
var a,b,x:real;
begin a:=8; b:=a/2; x:=A/B;
x:=a*sqr(sqrt(x+b/2)); writeln(x); end.
```

A) 32 B) 8 C) 256 D) 24

19. Agar i,y o'zgaruvchilar integer va x,z o'zgaruvchilar real bo'lsa, quyidagilarning qaysi birida xatolik ќosil bo'ladi?

A) x:=trunc(y) B) x:=sqrt(z) C) x:=sqrt(i) D) z:=sin(j)

20. Quyidagi operator necha marta takrorlanadi?

for x:=1 downto 10 do write (x);

A) 1 marta B) 5 marta C) 10 marta D) 0 marta

21. Quyida write operatori necha marta bajariladi?

for i:=1 to 3 do for j:=0 to 3 do write (i+j);

A) 12 marta B) 1 marta C) 3 marta D) 9 marta

22. type hafta=(dush, sesh, chor, paysch, schanba, yaksch);

var x,y: hafta; t:23. . 56;

tavsiflash berilgan bo'lsa,

quyidagi ta'minlashlardan qaysisi noto'g'ri?

A) y:=t B) x:=sesh C) y:=x D) t:=31

23. type meva=(gilos, olma, nok, usum);

var x,y: meva; t:1. . 6;

kabi tavsiflash berilgan bo'lsa, quyidagilardan qaysi biri to'g'ri?

A) for x:=olma to usum do B) read(x) C) write(y) D) for y:=1 to 6 do s:=s+1;

24. Quyida S ning qiymati nimaga teng?

```
begin
S:=0;
for I:=1 to 3 do S:=S+2*i;
writeln(S);
end.
```

A) 12 B) 24 C) 48 D) 96

25. Dastur qiymati nimaga teng?

```
begin X:=-2; y:=4; A:=-2;
if A*A-y>x*x then z:=y+A else z:=X+A;
writeln(z);
```

end.

A) –4 B) –5 C) 1 D) 7

26. Quyida S ning qiymati nimaga teng?

begin for j:=-2 to 3 do begin S:=0; S:=S+(j+1); end; writeln(S); end.

A) 4 B) 2 C) 10 D) 6

27. Dastur natijasini aniqlang. begin A:=12; B:=8; D:=20;

if A>18 then D:=A-B; writeln(D); end.

A) 20 B) 5 C) 8 D) 4

28. Dastur natijasini aniqlang.

begin X:=2; p:=1;

1:P:=P(2*x-2); X:=X+3; if X<=6 then goto 1; writeln(P); end.*

A) 16 B) 20 C) 2 D) 24

29. Sharti oldin qo'yilgan sikl operatori qaysi?

A) while, do B) for, to, do C) if, then, else D) goto

30. Hisob natijasini aniqlang?

begin A:=10; B:=8; C:= 6;

if (A>B) and (C>B) then y:=A+B-C else y:=A-B+C; writeln(y); end.

A) 8 B) 12 C) 10 D) 14

31. Ḱisob natijasini aniqlang?

begin a:=12; b:=14; c:=10;

if (a>b) or (b>c) then y:=a+b-c else y:=a-b+c; writeln(y);

end.

A) 16 B) 14 C) 8 D) 6

32. Satrlar ketma-ketligini qo'shish funksiyasi qaysi?

A) concat B)insert C) delete D) copy

33. Algorimlarning ifodalanish usullar qaysi javobda to'lik ko'rsatilgan

A) matnli, sxemali, jadval, grafik B) ogzaki, yozma

C) matnli, sxemali D) jadval kurinishda, sxemalar orkali

34. Object Pascal tilida chikarish operatorining qaysi birida xatolik mavjud

A) wrate(a,b); B) WRITE(a,b); C) WRITELN(a,' ',b); D) writeln(a,b);

35. Object Pascal tilida 1-chi pogonali ma'lumotlar strukturasiga nimalar kiradi?

A) Massiv, tuplam, yozuv B) Stek, dek, navbat C) Ruyxat

D) Daraxt, tur

36. Pascal tilida 2-chi pogonali ma'lumotlar strukturasiga nimalar kiradi

A) Stek, navbat, ruyxat B) Massiv, tuplam, yozuv C) Ruyxat, fayllar D) Daraxt, tur

37. Object Pascal tilida 3-chi pogonali ma'lumotlar strukturasiga nimalar kiradi

A) Bog'lamli ruyxat, daraxt, tur B) Massiv, tuplam, yozuv

C) Stek, dek, navbat D) Ro'yxat

38. Delphi muhitida tug'ri burchakli to'rtburchak chizuvchi prosedurani ko'rsating

A) Rectangle(X1,Y1,X2,Y2); B) Line(X1,Y1,X2,Y2);

C) Circle(X1,Y1,R); D) LineTO(X,Y);

39. Delphi muhiti qaysi algoritmik til asosida masalalarni yechadi?

A) Object Pascal B) Pascal C) Turbo Pascal D) Fortran

40. Obyektlarga yunaltirilgan dasturlashning asosini ... lar tashkil etadi

A) Vorislilik, polimorfizm, inkapsulyasiya B) Voqyealilik, borlik

C) Xaqqaniylik va soflik D) strukturaviylik

41. Delphi muxiti obyektlar inspektori (yo'rikchisi) qaysi qismlardan tashkil topgan

A) Xodisalar va xossalar B) Yuklanishlar va xodisalar

C) Yuklanishlar va xossalar D) Xossalar va natijalar

42. Delphi muxitida formada biror obyektni, masalan, buyruqli tugmani yaratish nima orqali sodir etiladi

A) Jixozlar paneli komponentalari orqali B) Menyular orkali

C) Obyektlar inspektori orkali D) Dastur yordamida

43. Object Pascalda to'plam bilan yozuv orasida qanday farq mavjud

A) To’plam elementlari bir turga mansub, yozuvda esa xar xil tipli ma’lumotlar bulishi mumkin

B) Ularning orasida farq yo’k

C) Yozuv elementlari bir turga mansub, to’plamda esa xar xil tipli ma’lumotlar bulishi mumkin

D) To’plam 1-chi pogonali ma’lumotlar strukturasiga kiradi, yozuv esa 2-chi pogonali ma’lumotlar strukturasiga kiradi

44. Turbo Pascal ning Graph muxitini yuklovchi prosedurani aniqlang

A) InitGraph(Drayver:integer; Rejim: Integer; Drayverga yul:String);

B) DetectGraph(Detect,mode:integer);

C) SetGraphMode(mode:integer);

D) GetGraphMode(mode:integer);

45. SetTextStyle(a,b,c:integer) proseduraga izox bering.

A) a-shrifrning stili b-tekstning orintasiyasi c-shrifrning o’lchami

B) a-tekstning orintasiyasi b-shrifrning stili c-shrifrning o’lchami

C) a-tekstni chapdan tugirlash b-shrifrning ulchami c-tekstni markaz buyicha to’girlash

D) a-shrifrning stili b-tekstning orintasiyasi c-tekstni chapdan to’girlash

46. Quyidagi muloxaza qaysi proseduraga tegishli:

"Berilgan satrdan ma’lum joydan boshlab kerakli ma’lumotni ko’chirish"

A) Copy(s:string):string; B) Delete(s:string,n,m:integer):string;

C) Lenght(s:string):integer; D) Insert(s,l:string,n:integer):string;

47. Object Pascal da fayllarning qaysi turlari ishlatiladi.

A) 3 turi – matnli, tiplashgan va tiplashmagan turlari

B) 2 turi tiplashgan va tiplashmagan turlari

C) 1 turi – fakat matnli fayllar

D) tugri javob yuk

48. Dastur tekstini bir tildan mashina tiliga o’tkazuvchi sistemali dastur:

A) Translyator; B) Dasturlash tili; C) Yukori poganadagi til; D) Past poganadagi til;

49. Grafik rejimni yopuvchi prsedura.

A) CloseGraph; B) Close; C) EndGraph; D) GraphClose;

50. Massivlarni tasvirlash va xotiraga joylashishning usullari.

A) statikli va dinamikli, kator va ustunlar buyicha; B) kator va ustunlar buyicha;

C) ustun buyicha va statikli; D) statikli va dinamikli;

51. Massivning statik tasvirlashi dinamik tasvirlashdan qanday fark qiladi.

A) Birinchisida xotira translyasiya paytida ajratiladi, ikkinchisida dasturni bajarish paytida ajratiladi;

B) Birinchisida dasturni bajarishda, ikkinchisida translyasiya jarayenida;

C) Bu usullarning farqi yuk;

D) Massivlar statik tasvirlanmaydi;

52. Ctekning ishlash prinsipi qanday?

A) “Birinchi kelgan-oxirda ketadi” prinsipi asosida;

B) Massiv elementlarini kayta ishlash prinsipiga mos;

C) To’plam elementlarini kayta ishlash prinsipiga mos;

D) “Birinchi kelgan- birinchi ketadi” prinsipi asosida.

FOYDALANILGAN ADABIYOTLAR RO‘YXATI

1. M. Ashurov, N. Mirzahmedova, N. Xaytullayeva. Algoritmlash va dasturlash asoslari. Uslubiy qo‘llanma. T. : “Bayoz”, 2016 y.
2. A. R. Azamatov, B. Boltayev. Algoritmlash va dasturlash asoslari. O‘quv qo‘llanma. T. :“Cho‘lpon”, 2010 y.
3. A. R. Azamatov, B. Boltayev. Algoritmlash va dasturlash asoslari. O‘quv qo‘llanma. T. :“Cho‘lpon”, 2013 y.
4. Sh. I. Razzoqov, M. J. Yunusova. Dasturlash: Kasb-hunar kollejlari uchun o‘quv qo‘llanma. T. :“Ilim Ziyo”, 2011y.
5. М. Ашуров, М. Мирмахмудов, Ш. Сапаев. Замонавий дастурлаш тиллари фанидан лаборатория ишлари. Т. :ТДПУ, 2008 й.
6. Меняев Михаил Федорвич. Информационные технология управления. Москва, «Издательский Омегал», 2003 г.
7. Peter Gottschling. Discovering Modern C++. An Intensive Course for Scientists, Engineers, and Programmers. “Addison-Wesley”, 2015 y.
8. Sh. A. Nazirov va boshqalar. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013. 488 b.
9. П. Дарахвелидзе, Э. Марков. Программирование в Delphi7. Учебник. Санкт-Петербург, “БХВ-Петербург” 2003 г.
10. В. В. Фаронов Программирование на языке высокого уровня Delphi. Учебник. М. : “Питер”, 2003 г.
11. В. Т. Безручко. Практикум по курсу информатики. М. : «Финансы и статистика», 2004 г.