004
K15

M.M.KADIROV

TEXNIK TIZIMLARDA AXBOROT TEXNOLOGIYALARI



O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA
MAXSUS TA’LIM VAZIRLIGI

M.M. KADIROV

TEXNIK TIZIMLARDA AXBOROT TEXNOLOGIYALARI

2-qism

O‘zbekiston Respublikasi Oliy va o‘rta maxsus ta’lim vazirligi tomonidan oliy o‘quv yurtlarining texnik tizimlarda axborot texnologiyalari fanini o‘rganayotgan barcha ta’lim yo‘nalishi talabalari uchun darslik sifatida tavsiya etilgan

«O‘zbekiston faylasuflari milliy jamiyati» nashriyoti
Toshkent — 2019

004
K 15

UO'K: 004.43(075)
KBK: 32.81ya73
K 15

Kadirov, M.M.

K 15 Texnik tizimlarda axborot texnologiyalari [Matn] 2-qism: darslik / M.M. Kadirov. – Toshkent: «O'zbekiston faylasuflari milliy jamiyati» nashriyoti, 2019. – 306 bet.

UO'K: 004.43(075)
KBK: 32.81

Mazkur darslikda "Texnik tizimlarda axborot texnologiyalari" fanining asosiy bo'limlaridan bo'lgan dasturlash texnologiyalari haqida bayon etilgan. Darslikda zamonaviy dasturlash tillari va texnologiyalari, Borland C++ Builder 6 integrallashgan sohasi, uning tashkil etuvchilari va asosiy konstruksiyalaridan foydalanish xususiyatlari haqida to'liq ma'lumotlar berilgan. Texnik tizimlardagi jarayonlarni dasturlash misollar yordamida ko'rsatilgan.

Darslik "Texnik tizimlarda axborot texnologiyalari" fanini o'rganayotgan barcha ta'lim yo'nalishidagi bakalavr talabalariga mo'ljallangan. Shuningdek, darslik magistrantlar, professor-o'qituvchilar hamda fanni mustaqil o'rganuvchilarga tavsiya etiladi.

Taqrizchilar:

Sh.M. Gulyamov – ToshDTU, "ICHJA" kafedrasi professori, t.f.d., prof.,

A.X. Nishanov – TATU, "ATDT" kafedrasi professori, t.f.d., prof.

UQUV ZALI

Namangan muhandislik-texnologiya instituti
AXBOROT-RESURS MARKAZI
Inv. № 44267

ISBN 978-9943-6169-7-4

© «O'zbekiston faylasuflari milliy jamiyati» nashriyoti, 2019

KIRISH

Texnik tizimlarda axborot texnologiyalari iqtisodda, boshqarishda, aloqada, ilmiy tadqiqotlarda, ta'limda, xizmat ko'rsatish sohasida, tijorat, moliya va inson faoliyatining boshqa sohalarida qo'llanilishining rivoji axborotlashtirish va jamiyat rivojini belgilovchi yo'nalish hisoblanadi. Kompyuter texnikasining qo'llanishi evaziga erishiluvchi samara axborot ishlanishi ko'lamining oshishi bilan ortib boradi.

Bugungi kunda mamlakatimizda olib borilayotgan keng ko'lamli islohotlar ko'p jihatdan uzluksiz ta'lim tizimini shakllantirishni taqozo etadi. Yangicha fikrlaydigan, bozor sharoitlarida muvaffaqiyatli xo'jalik yurita oladigan malakali, chuqur bilimli mutaxassislarni, ayniqsa, axborot texnologiyalaridan keng foydalana oladigan malakali kadrlarni tayyorlash davr talabi bo'lib qolmoqda.

Jamiyatning jadal sur'atlarda rivojlanishi axborot texnologiyalarini barcha sohalarga joriy etish bilan chambarchas bog'liq. Bu esa yosh mutaxassis kadrlardan zamonaviy dasturlash texnologiyalarini bilishni talab etadi.

Darslik kadrlar tayyorlash milliy dasturi "Ta'lim to'g'risida"gi O'zbekiston Respublikasi qonunining qoidalariga muvofiq holda amalga oshirilgan bo'lib, milliy tajribaning tahlili va zamonaviy dasturlash tizimidagi jahon miqyosidagi yutuqlar asosida tayyorlangan.

Darslikning birinchi bobida zamonaviy dasturlash tillari va texnologiyalari, ularning ishlatilishi va tasnifi haqida to'liq ma'lumotlar berilgan.

Darslikning ikkinchi bobida Borland C++ Builder 6 integrallashgan sohasi, uning tashkil etuvchilari, dasturni o'rnatish jarayoni va komponentalar palitrasi haqida bayon etilgan.

Darslikning uchinchi bobida C++ dasturlash tilining asosiy konstruksiyalari va ulardan foydalanish xususiyatlari, ma'lumotlarning toifalari, standart funksiyalar, shuningdek, konsol va vizual muhitlarda dasturlarni tuzishga alohida e'tibor berilgan.

Darslikning to'rtinchi bobida Borland C++ Builder 6 dasturlash tizimida chiziqli jarayonlarni dasturlash va operatorlar tasnifi batafsil bayon etilgan.

Darslikning beshinchi bobida C++ Builder 6 dasturlash tizimida tarmoqlanuvchi jarayonlarni dasturlash, shartsiz o'tish operatori, shartli o'tish operatori va tanlash operatorlari misollar yordamida to'liq bayon etilgan.

Darslikning oltinchi bobida C++ Builder 6 dasturlash tizimida takrorlanuvchi jarayonlarni dasturlash, avval sharti tekshiriladigan takrorlanish jarayoni, sharti keyin tekshiriladigan takrorlanish jarayoni va parametrli takrorlanish jarayonlariga alohida e'tibor berilgan.

Darslikning yettinchi bobi Borland C++ Builder 6 dasturlash tizimida massivlar bilan ishlashga bag'ishlangan. Massiv tushunchasi va uning turlari, C++ tilida massivlarni tavsiflash va massiv elementlarini xotiraga kiritish misollarda batafsil bayon etilgan.

Darslikning sakkizinchi bobida C++ tilida struktura toifasi va ularni tavsiflash, struktura elementi va ular ustida bajariladigan amallar, shuningdek, C++ dasturlash tilida aralash toifani qo'llash misollarda batafsil keltirilgan.

Darslikning to'qqizinchi bobida C++ da funksiya va protseduralar, strukturaviy dasturlashni amalga oshirish va muhandislik masalalarini obyektga mo'ljallangan dasturlarga tatbiq etish vazifalariga e'tibor qaratilgan.

Darslikning o'ninchi bobida C++ dasturlash tizimida ma'lumotlarning faylli va murojaat toifasi bilan ishlash, murojaat toifasi va uni dasturdagi o'rni batafsil yoritilgan.

Darslikning o'n birinchi bobida C++ algoritmik tilida sinflar va obyektlarning tavsiflari keltirilgan. Abstraksiya, vorislik va polimorfizmga ta'riflar berilgan.

Darslikning o'n ikkinchi bobida Borland C++ Builder 6 ning grafik imkoniyatlari, texnik masalalarni vizuallashtirish usullari, grafik modullarning imkoniyatlari va ulardan foydalanish haqida batafsil bayon qilingan.

Darslikning o'n uchinchi bobida Borland C++ Builder 6 integrallashgan sohasida ma'lumotlar bazasini yaratish, dastur yordamida ma'lumotlar bazasini import va eksport qilish, shuningdek komponentalari yordamida texnik tizimdagi ma'lumotlar bazasini qayta ishlash batafsil yoritilgan.

I bob. TEXNIK TIZIMLARDA ZAMONAVIY DASTURLASH TEXNOLOGIYALARI

Tayanch so'zlar: dasturlash, dasturlash tillari, obyektga yo'naltirilgan dasturlash texnologiyalari, dasturlash tillarining tasnifi, ochiq kodli dasturlash tillari.

1.1. Dasturlash tillari va ularning kelib chiqishi

Birinchi dasturlashtirilgan hisoblash mashinalari yaratilganidan beri, insoniyat sakkiz mingdan ziyod dasturlash tillarini ishlab chiqqan. Har yili ularning soni ortib bormoqda. Ba'zi tillar o'zlarining ishlab chiquvchilarining oz sonli foydalanuvchilar tomonidan ishlatilishi mumkin, boshqa dasturlash tillari esa millionlab foydalanuvchilar tomonidan dasturiy ilovalar yaratish uchun foydalaniladi.

Dasturlash tillari kompyuter dasturlarini yozish uchun mo'ljallangan bo'lib, ular o'z ichida ma'lum qoidalar to'plamini mujassamlashtirgan va shu asosida kompyuterning muayyan hisoblash jarayonini bajarishiga imkon beradi va turli obyektlarning boshqaruvini tashkil qiladi. Ko'pchilik dasturlash tillari ma'lumotlar strukturalarini aniqlash, boshqarish va hisoblash jarayonini nazorat qilish uchun maxsus konstruksiyalardan foydalanadi.

Dasturlash tillari rivojlanish darajasiga ko'ra bir nechta davrlarga bo'linadi. Birinchi davr tillari 50-yillargacha bo'lgan davr, birinchi kompyuterlar paydo bo'lgan vaqtda yaratilgan. Ularga misol qilib Assembler tilini olish mumkin.

Ikkinchi davr tillari 50–60-yillarga to'g'ri keladi. Bu vaqtda Assemblerning yangi ko'rinishi yaratildi va unga o'zgaruvchi tushunchasi kiritildi. Ushbu tildan foydalanish tufayli dasturlarning bajarilish tezligi va ishonchliligi oshdi.

Uchinchi davr – 60-yillarda yuqori darajali universal tillar paydo bo'la boshladi, ular yordamida turli sohadagi masalalarni yechish imkoniyati ko'paydi. Bu tillarning mukammal konstruksiyalari,

soddaligi, kompyuter turiga bo'g'liq bo'lmaganligi dasturchilarning ish unumdorligini oshirdi. Bu tillarning ko'pchiligi hozirgi kunda ham ishlatiladi.

70-yillar to'rtinchi davr tillarining yaratilish davri deb hisoblanadi. Ushbu tillar yirik loyihalarni amalga oshirish uchun mo'ljallangan bo'lib, ular asosan muayyan sohadagi masalalarni yechish uchun qulaydir. Dasturlarda muammoga yo'naltirilgan til vositalari yordamida vazifalar bajariadi, ya'ni bitta funksiya orqali amallar ifodalanadi.

1949-yilda Jon Mouchli yuqori darajali dasturlash tillarining dastlabkilariga asos bo'lgan Short Code sistemasini ishlab chiqdi. 1951-yilda Greys Xopper birinchi bo'lib A-O kompilyatorini yaratdi.

1954-yilda FORTRAN (FORmula TRANslation) tili yaratildi. Bu dastur IBM kompaniyasida Djon Bekusa boshchilida yaratilgan. Fortran asosan ilmiy va muhandislik hisob-kitoblari uchun keng qo'llaniladi.

1958-yilda LISP (LISt Processing language – "Ro'yxatni qayta ishlash tili") dasturlar chiziqli ro'yxatlar bilan ifodalangan. Djon Makkarti tomonidan yaratilgan yuqori darajali dasturlash tili hisoblanadi.

1959-yilda COBOL (Common Business Oriented Language – umumiy tijoratga yo'naltirilgan til) tili o'ylab chiqildi. Bu til yuqori darajali biznesga yo'naltirilgan dasturlash tili hisoblanadi. Bu dasturlash tili Greys Xopper boshchiligida yaratilgan.

1960-yili ALGOL tili yaratilgan bo'lib, ALGOritmic Langauge – algoritmik til degan ma'noni anglatadi va ilmiy-texnik masalalarni hisoblashlarda qo'llaniladi.

1964-yilda Jon Kemeni va Tomas Kurts boshchiligida Dartmut universitetida BASIC dasturlash tili ishlab chiqildi. Bu til sodda, o'rganishga oson va murakkab bo'lmagan hisoblashlarni bajarishga mo'ljallangan.

1971-yilda PASCAL tili e'lon qilingan bo'lib, fransuz olimi Blez Pascal nomiga qo'yilgan. Toifali xildagi masalalar yechimini olishda tartiblangan dasturlar tuzishda qo'llaniladi.

1960–1970-yillarda dasturlash tillarining asosiy g'oyalari va tushunchalari ishlab chiqildi. Shu asosida hozirgi zamonaviy dasturlash tillari yaratilmoqda va takomillashtirilmoqda.

1.2. Zamonaviy dasturlash tillari

Hozirgi kunda xalq xo'jaligining ko'plab masalalarini kompyuter yordamida yechish uchun mo'ljallangan zamonaviy dasturlash tillari mavjud. GitHub xalqaro saytining dunyoning axborot texnologiyalari sohasida yoqori o'rinlarda turadigan kompaniya dasturchilari o'rtasida o'tkazilgan so'rovnomada eng mashhur dasturlash tillari reytingida e'lon qilingan ro'yxatdan foydalanib, ba'zi bir zamonaviy dasturlash tillarini ko'rib chiqamiz:

1. JavaScript – multiparadigmali dasturlash tili hisoblanib, obyektga yo'naltirilgan, funksional uslublarni qo'llab-quvvatlaydi[1]. JavaScript – yuqori darajali dasturlash tili bo'lib, ishlash jarayonida mijoz brauzerida, ya'ni oxirgi foydalanuvchining kompyuterida buyruqlar qayta ishlanadi, bu esa serverdagi yukni kamaytiradi va dastur ishlash tezligini oshiradi. JavaScript Netscape kompaniyasi tomonidan ishlab chiqilgan. Bu dasturlash tili Internet texnologiyalarida keng qo'llaniladi.

2. Java – chuqur darajada klasslarga asoslangan (class-based), obyektga yo'naltirilgan (object-oriented) dasturlash tili bo'lib imkoni boricha oson ko'chirib yurishga va ko'plab platformalarda ishlashga moslashtirilgan. Java dasturlash tili Sun Microsystems kompaniyasi tomonidan ishlab chiqilgan. Keyinchalik bu kompaniyani Oracle kompaniyasi tomonidan sotib olgan. 1995-yili birinchi rasmiy ko'rinishi chiqqan. Java ilovalari odatda maxsus bayt kodlarda amalga oshirilganligi sababli, ular virtual Java mashinasi yordamida har qanday kompyuter arxitekturasida ishlashlari mumkin. Java dasturlash tili Jeyms Gosling tomonidan yaratilgan bo'lib boshlang'ich davrda elektron qurilmalarni

[1] Axel Rauschmayer. Speaking JavaScript. O-Reilly Media. USA 2014. p. 3–5.

dasturlash uchun ishlab chiqilgan. Keyinchalik mijoz va server dasturlarini yozish uchun foydalanilgan. Java dasturlash tili veb sohasida uzoq vaqtdan beri foydalanib kelinayotgan barqaror va kuchli tillardan biri hisoblanadi. Javani barcha platformalar, operatsion tizimlar va qurilmalarda keng foydalaniladi.

3. Python – yuqori darajadagi dasturlash tili bo'lib, soddaligi, o'qilishi va sintaksisi tufayli eng oson til hisoblanadi. 80-yillarda Gvido van Rossum tomonidan ishlab chiqilgan. Python odatda dasturchilarni qisqa vaqtda ko'p miqdorda oson o'qiladigan va funksional kodlarni yozilishini ta'minlaydigan skript til sifatida foydalaniladi, biroq u dinamik hamda obyektga yo'naltirilgan, protsedurali va funksional dasturlashni ham ta'minlaydi. Tez moslashuvchanligi hisobiga, Python bugungi kunda yuqori darajada keng qo'llaniladigan dasturlash tillaridan biri hisoblanadi. Veb ilovalarni yaratishda keng foydalaniladi.

4. Ruby – dinamik, ochiq kodli (open-sorce), obyektga yo'naltirilgan dasturlash tili bo'lib, 90-yillarda kompyuter olimi Yukihiro Matsumoto tomonidan ishlab chiqilgan eng yosh tillardan biri hisoblanadi. Bu tilda o'qish va yozish uchun sodda sintaksisdan foydalanilgan, juda ko'p buyruqlarni o'rganib chiqish shart emas. Shuning hisobiga, bu tilni o'rganish nisbatan oson. Tilning o'zi obyektga yo'naltirilgan bo'lsada, protsedurali, funksional va imperativ dasturlashni ta'minlashligi bu tilni juda ham tez moslashuvchi tillardan qiladi. Veb ilovalarni yaratishda keng foydalaniladi.

5. PHP (Hypertext Preprocessor – "PHP: gipermatnli prosessor") – dinamik veb-saytlarni yaratish va rivojlantirish uchun eng keng tarqalgan tillardan biridir. PHP 1995-yilda ishlab chiqilgan. PHP ma'lumotlarni serverda qayta ishlaydi va natijada foydalanuvchi kerakli ma'lumotlarni HTML shaklida oladi. PHP – ochiq kodli dasturlash tili, shuning uchun minglab modullarning tayyor andozalari yozib qo'yilgan, ularni kerakli funksiyalarga moslashtirishi mumkin[2]. U juda ko'p mavjud ma'lumotlar bazasini

[2] Matt Doyle. Beginning PHP 5.3. Wiley Publishing, Inc. USA 2010. p. 3–8.

boshqarish tizimlari bilan birgalikda o'zaro ishlash imkonini beradi (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape va Apache Derby, Informix, Ovrimos SQL, Lotus Notes, DB++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird/InterBase, Paradox File Access, MaxDB).

6. C++ – kompilyatsiyalanadigan, statik usulda kiritilgan umumiy maqsadli dasturlash tilidir. 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++ tili operatsiyon tizimlarga aloqador qisimlarni, kliyent-server dasturlarini, kompyuter o'yinlarini, kundalik ehtiyojda qo'llaniladigan dasturlarni va shu kabi turli maqsadlarda ishlatiladigan dasturlarni ishlab chiqishda qo'llaniladi. C ++ tilining erkin va tijorat maqsadidagi turli platformalar uchun juda ko'p qo'llanishi mumkin. Masalan, x86 platformasida bu GCC, Visual C ++, Intel C ++ kompilyatori, Embarcadero (Borland) C ++ Builder va boshqalar.

7. CSS (ingl. tilidan Cascading Style Sheets – kaskadli uslublar jadvallari) – belgilash tili yordamida yozilgan hujjat ko'rinishini tasvirlab berish uchun mo'ljallangan dasturlash tili. Veb-sahifalarning bo'laklarga bo'lish va veb-saytlarning dizaynini yaratish uchun ishlab chiqilgan. Bo'laklarga bo'lish veb-saytlardagi ma'lumotlardan foydalanish darajasini oshiradi va tashqi ko'rinishini zamonaviy texnologiyalardan foydalanish imkoniyatini oshiradi. Bu dasturlash tilida ishlab chiqilgan veb-sahifalarni qulaylik tomoni shundaki o'qish uchun katta ko'rinishda, bosmaga chiqarish uchun ma'lum format ko'rinishida va mobil telefonlarga mundarija asosida ketma-ketlikda ko'rsatishni har xil dizaynlarda amalga oshirish imkoniyatini yaratadi.

8. C# – (ci sharp) obyektga yo'naltirilgan dasturlash tilidir. 1998-2001-yillarda Anders Heilesberg boshchiligidagi muhandislar guruhi tomonidan Microsoft kompaniyasida ishlab chiqilgan. Microsoft .NET Framework platformasida dasturlarni ishlab chiqish tili sifatida ishlatiladi. C# dasturini amalga oshirish uchun csc.exe kompilyatori shaklida .NET Frameworkning tarkibiga kiritilgan. C# ochiq kodli dasturlash tili hisoblanadi.

9. C – statik usulda tuzilgan kompilyatsiyalanuvchi umumiy maqsadli dasturlash tili. C tili 1972-yilda Bell Labs laboratoriyasida Dennis Ritchi tomonidan ishlab chiqilgan. Boshida ishlab chiqilganda UNIX operatsion tizimi platformasiga asoslangan. Keyinchalik ko'p platformalarda ishlash uchun mo'ljallangan versiyalari ishlab chiqilgan. U C++, Java, C#, JavaScript va Perl kabi dasturlash tillarining yaratilishiga asos bo'lgan. Shuning uchun ushbu tilni o'rganish boshqa tillarni tushunishga olib keladi. C tilidan past darajadagi ilovalarni ishlab chiqishda foydalaniladi, chunki u apparat vositalar uchun dasturlar yaratishda qulay hisoblanadi.

10. Go – Google tomonidan ishlab chiqilgan keng qamrovli dasturlash tili. Go dasturlash tilining dastlabki rivojlanishi 2007-yil sentyabr oyida boshlangan va uning bevosita loyihasi ustida Robert Grismer, Rob Pike va Ken Thompson ishlagan. Rasmiy ravishda, Go dasturlash tili 2009-yilning noyabr oyida keng ommaga namoyish qilingan.

Go tili zamonaviy taqsimlangan tizimlar va ko'p yadroli protsessorlarda ishlaydigan yuqori sifatli dasturlarni yaratish uchun tizim dasturiy tili sifatida ishlab chiqilgan. Buni C tilini almashtirishga urinish sifatida ko'rilishi mumkin.

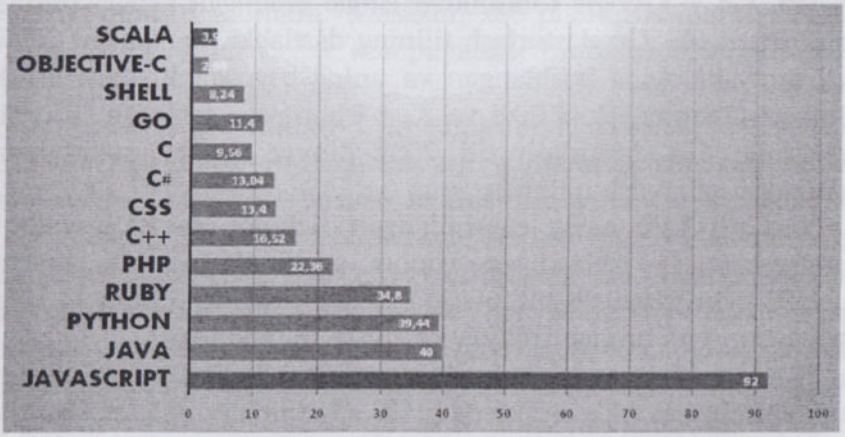
Rivojlanish jarayonida yuqori darajadagi kompilyatsiya qilishni ta'minlashga alohida e'tibor qaratildi. Go tilida yaratilgan dasturlar obyekt kodiga yoziladi va virtual mashinani ishga tushirishni talab qilmaydi[3].

[3] Mark Lewin. Go Web Development Succinctly. Syncfusion Inc.USA 2017. p. 12–16.

11. Objective-C – obyektga yo'naltirilgan kompilyatsiyalanuvchi dasturlash tili. Bu dasturlash tili C va Smalltalk dasturlash tili paradigmalariga asoslangan holda yaratilgan. Apple komponiyasi tomonidan foydalaniladi. Xususan, obyekt modeli Smalltalk dasturlash tili uslubida qurilgan, ya'ni obyektlarga xabarlar yuboriladi. Objective-C kompilyatori GCC tizimiga kiritilgan va ko'pchilik asosiy platformalarda mavjud.

Til birinchi navbatda Mac OS X (kakao) va GNUstep tizimlari uchun foydalaniladi. OpenStep interfeysi obyektga yo'naltirilgan ilovalarini yaratish uchun ishlatiladi.

12. Scala – keng qamrovli dasturlash tili bo'lib, funksional va obyektga yo'naltirilgan dasturlashni o'z ichiga qamrab olgan. Dasturiy ilovalarni yaratish uchun sodda va qisqa komponentalarga asoslangan dasturlash tili. 2003-yilda Martin Oderski tomonidan Java va .Net platformalari asosida ishlab chiqilgan. Scala dasturlash tilidan axborot texnologiyalariga tegishli juda ko'p kompaniyalar dasturiy ilovalarini yaratishda foydalanadi (Coursera, Linkedin, Twitter, UBS).



1.1-rasm. *Zamonaviy dastulash tillari.*

Axborot texnologiyalarining rivojlanishi zamonaviy dasturlash tillaridan keng foydalanishni talab etmoqda. Xorijiy davlatlarda

o'tkazilgan so'rovlar asosida zamonaviy dastulash tillarining dasturiy ilovalarni yaratishda foydalanishlik darajalari asosida quyidagi statistikani ko'rib chiqishimiz mumkin (1.1-rasm).

Hozirgi kunda dasturlash tillari ko'payib bormoqda va o'z navbatida dasturlash tillaridan ma'lum sohalardagi masalarni yechishda keng qo'llanilmoqda.

1.3. Dasturlash tillarining tasniflanishi

Yuqorida ko'rib chiqilgan dasturlash tillari asosida shuni aytish mumkinki, dasturlash tillarining hammasi bir xil xususiyatga ega emas. Ba'zi dasturlash tillari obyektga yo'naltirilgan bo'lsa boshqalar qurilmalarga asoslanadi. Shuni hisobga olgan holda dasturlash tillarini quyidagi ko'rinishda tasniflashimiz mumkin:

1. Past va yuqori darajali dasturlash tillari. Past darajali dasturlash tillari mashinaga bog'liq tillar hisoblanadi. Mashinaga bog'liq tillar, o'z navbatida, mashina tillari va mashinaga mo'ljallangan tillarga ajratiladi. Ular dasturlash tilining birinchi avlodiga kiradi. Past darajali dasturlash tillarining ikkinchi avlodi sifatida Assembler tillarini misol keltirish mumkin. Assembler dasturlash tili makrokomandalar yordamida avtokodlarni kiritish imkonini beradi. Misol uchun Assembler tili yordamida dastur tuzish quyidagi ko'rinishda amalga oshiriladi:

```
.386
.model flat, stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\kernel32.lib
.data
        msg db "Hello, world", 13, 10
        len equ $-msg
.data?
        written dd ?
```

```
.code
start:
        push    -11
        call    GetStdHandle
        push    0
        push    OFFSET written
        push    len
        push    OFFSET msg
        push    eax
        call    WriteFile
        push    0
        call    ExitProcess
end start
```

Yuqorida berilgan dastur kodi orqali ekranga Hello, world (Salom olam) so'zini chiqarish mumkin.

Birinchi va ikkinchi avlod dasturlash tillari qurilmaning xususiyatlarini hisobga olgan holda ushbu protsessorda kerakli funksiyalarning bajarilishini yaxshilash imkonini beradi.

1970-yillarga kelib axborot texnologiyalarining jadal sur'atlarda rivojlanishi dasturchilardan mashina kodlarida dasturlashdan yuqori darajali dasturlash tillariga o'tishni talab qildi. Yuqori dasturlash tillari tilning semantik modelini, ya'ni insoning fikrlash xususiyatlarini o'z ichiga olgan. Buning natijasida dasturlarni o'zgartirish va bir qurilmadan boshqa qurilmalarga ko'chirish osonlashadi. Bu dasturlash tillarining uchinchi avlodi bo'ldi.

To'rtinchi va beshinchi avlod dasturlash tillari dasturning nima qilishi kerakligini ifodalash bilan to'ldirildi. Bunday tillarda, aksariyat hollarda, ma'lum bir kompyuter arxitekturasi va operatsion tizim uchun dasturni qayta kompilyatsiya qilish orqali amalga oshirildi. Yuqori darajali dasturlash tillari samaradorlikning ustunligini ta'milaydi.

2. Xavfsiz va xavfsiz bo'lmagan dasturlash tillari. Dasturlash tili xavfsiz hisoblanadi qachonki, unda tuzilgan dastur kompilyator tomonidan qabul qilinib, mumkin bo'lgan xatti-harakatlar

chegarasidan tashqariga chiqmasa. Bunda kompilyator tomonidan qabul qilingan bo'lishiga qaramay dasturda mantiqiy xatoliklar mavjud bo'lishi mumkin. Mantiqiy xatoliklar mavjud bo'lishiga qaramay tuzilgan ma'lumotlar yaxlitligi buzilmasa xavfsiz dasturlash tili hisoblanadi. Ko'pincha toifalarni qabul qilishda va ulardan samarali foydalanishda xatoliklarni to'g'rilash imkonini beruvchi qo'shimcha kutubxonalardan foydalaniladi. Bu esa kompyuter xotirasiga to'g'ri murojaatni ta'minlaydi.

Xavfsiz bo'lmagan tillarda kiritilayotgan toifalar qaysi ko'rinishda kiritilsa shu ko'rinishda qabul qilinadi. Xavfsiz bo'lmagan tillarda dasturlarni zaiflashtiruvchi holatlarni va xotiraga murojaat qilishda xatoliklarga olib keladi.

3. Kompilyatsiya va interpretatsiya qilinadigan dasturlash tillari. Kompilyatsiya bu dasturning dastlabki kodini, kompilyator deb ataladigan maxsus dastur yordamida maqsadli mashina kodiga aylantirilishini anglatadi. Natijada alohida fayl sifatida saqlanadi va dasturni bajarish uchun qo'shimcha modul sifatida ishga tushurish imkonini beradi.

Interpretatsiya qilinadigan dasturlash tillari dastlabki kodni mashina kodiga aylantirmaydi. Dastlabki kodni markaziy protsessor tomonidan to'g'ridan to'g'ri bajarilishi uchun maxsus interpretator dasturi yordamida amalga oshiriadi.

Ba'zi bir dasturlash tillari dasturni amalga oshirish uchun ikki xil usuldan ham foydalanadi. Kompilyatsiya qilinadigan tillar tezkor va kichik hajmga ega dasturlarni ishlab chiqishda keng qo'llaniladi va samaradorlikni oshiradi. Interpretatsiya qilinadigan dasturlash tillari operatsion tizimning bir qancha apparat qurilmalari bilan ishlash imkonini beruvchi dasturlarni yaratishda keng qo'llaniladi. Uning kamchilik tomoni ko'p resurslarni talab qilishi va bajarilish tezligining pastligi hisoblanadi. Qo'llanish sohasiga qarab kompilyatsiya va interpretatsiya qilinadigan dasturlash tillarining o'ziga xos xususiyatlari mavjud.

4. Dasturlash paradigmasi. Dasturlash paradigmasi bu kompyuter dasturlarini yozish uslubini belgilovchi g'oyalar va tushunchalar to'plamidir (dasturlashning yondashuvi). Dasturlashning asosiy modellari sifatida quyidagilarni ko'rsatib o'tish mukin:

- imperativ dasturlash;
- deklarativ dasturlash;
- tizimli dasturlash;
- funksional dasturlash;
- mantiqiy dasturlash;
- obyektga yo'naltirilgan dasturlash.

Ba'zi bir dasturlash tillari ko'rib chiqilgan tasniflarning ko'pchilik xususiyatlarini o'zida jamlagan bo'lishi mumkin.

1.4. Zamonaviy dasturlash texnologiyalari

Dastlab dasturlash anchayin boshqotirma ixtiro bo'lib, u dasturchilarga dasturlarni kommutatsiya bloki orqali kompyuterning asosiy xotirasiga to'g'ridan to'g'ri kiritish imkonini berdi. Dasturlar mashina kodida ikkilik sanoq tizimida yozilar, dasturlarni mashina tilida yozishda tez-tez xatolarga yo'l qo'yilar edi. Bundan tashqari, mashina kodlaridagi dastur tushunish uchun g'oyat murakkab edi.

Vaqt o'tishi bilan kompyuterlar tobora kengroq qo'llanila boshlandi hamda yuqoriroq darajadagi protsedura tillari paydo bo'ldi. Bularning dastlabkisi FORTRAN tili edi. Biroq obyektga mo'ljallangan yondashuv rivojiga asosiy ta'sirni keyinroq paydo bo'lgan, masalan, ALGOL kabi protsedura tillari ko'rsatdi. Protsedura tillari dasturchiga axborotga ishlov berish dasturini pastroq darajadagi bir nechta protseduraga bo'lib tashlash imkonini beradi. Pastroq darajadagi bunday protseduralar dasturning umumiy tuzilmasini belgilab beradi. Ushbu protseduralarga izchil murojaatlar protseduralardan 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 protseduralar qo'shilgan edi. Har bir protsedura 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, hatto eng kichik to'g'rilash ham butun dasturda qator o'zgarishlar sodir bo'lishiga olib kelar edi.

Modulli dasturlashda, masalan, modullar kabi tilda protsedurali dasturlashda topilgan ayrim kamchiliklarni bartaraf etishga urinib ko'rildi. Modulli dasturlash dasturni bir necha tarkibiy bo'laklarga yoki boshqacha qilib aytganda, modullarga bo'lib tashlaydi. Agar protsedurali dasturlash ma'lumotlar va protseduralarni bo'lib tashlasa, modulli dasturlash, undan farqli o'laroq, ularni birlashtiradi. Modul ma'lumotlarning o'zidan hamda ma'lumotlarga ishlov beradigan protseduralardan iborat. Dasturning boshqa qismlariga moduldan foydalanish kerak bo'lib qolsa, ular modul interfeysiga murojaat etib qo'ya qoladi. Modullar barcha ichki axborotni dasturning boshqa qismlarida yashiradi.

Biroq modulli dasturlash ham kamchiliklardan xoli emas. Modullar kengaymas bo'ladi, bu degani kodga bevosita kirishsiz hamda uni to'g'ridan to'g'ri o'zgartirmay turib modulni qadamma-qadam o'zgartirish mumkin emas. Bundan tashqari, bitta modulni ishlab chiqishda, uning funksiyalarini boshqasiga o'tkazmay turib boshqasidan foydalanib bo'lmaydi, garchi modulda turni belgilab bo'lsada, bir modul boshqasida belgilangan turdan foydalana olmaydi.

Modulli dasturlash – bu yana protseduraga mo'ljallangan gibridli sxema bo'lib, unga amal qilishda dastur bir necha protseduralarga bo'linadi. Biroq endilikda protseduralar ishlov berilmagan ma'lumotlar ustida amallarni bajarmaydi, balki modullarni boshqaradi.

UQUV ZALI
Namangan muhandislik texnologiya instituti
AXBOROT-RESURS MARKAZI
Inv. № 44267

Obyektga yo'naltirilgan dasturlash texnologiyalari – dasturiy ta'minotning inqiroziga javob sifatida yuzaga kelgan dasturlash texnologiyalari hisoblanadi. Bu inqirozning sababi shunda ediki, strukturali dasturlash metodlari murakkablik darajasi borgan sari ortib borayotgan masalalar uchun dasturiy ta'minot yaratish imkonini bera olmay qoldi. Buning natijasida turli loyihalarni bajarish rejalari buzildi, qilinayotgan xarajatlar belgilangan budjetdan ortib ketdi, dasturiy ta'minotning funksionalligi buzildi, xatoliklari ortdi.

Dasturiy ta'minotning eng muhim tomonlaridan biri – uning murakkablik darajasidir. Biror dasturchi tizimning barcha xususiyatlarini to'liq hisobga ola olmaydi. Shuning uchun uni ishlab chiqishda dasturchi va boshqa mutaxassislarning yirik jamoasi qatnashadi. Demak, qo'yilgan masalaga to'g'ridan to'g'ri bog'liq bo'lgan murakkabliklarga ana shu jamoaning ishini bir maqsadga qaratilgan boshqarish ham qo'shiladi. An'anaviy dasturlash tillarida bunday murakkabliklarni hal qilishda "ajrat va boshqar" prinsipidan foydalanilgan. Ya'ni, masala kichik-kichik masalalarga ajratib, keyin har bir masala uchun alohida dastur ishlab chiqilgan va birlashtirilgan. Obyektga yo'naltirilgan dasturlash texnologiyalari esa masalaga boshqacha usulda yondashadi. Unda masalalarni yechish uchun kerak bo'ladigan elementlarni muammoli sohaning turli abstraksiyalariga taalluqli ekanligi asosiy o'rinda turadi. Bu abstraksiyalarni dasturchilar ishlab chiqishgan. Dasturchilar tomonidan ma'lum bir soha o'rganilib, uning alohida obyektlari ajratib olingan. Bu obyektlar uchun masalalarni yechishda qo'llash mumkin bo'lgan xususiyatlar aniqlangan. Ehtiyojga qarab har bir xususiyat ustida bajarish mumkin bo'lgan amallar aniqlangan. So'ngra o'rganilayotgan sohaning har bir real obyektiga mos dasturiy obyekt ishlab chiqilgan.

Ma'lumki, kompyuter yordamida hal qilinadigan har bir masala uchun maxsus dastur ishlab chiqish, yozish talab qilinadi. Bunday masalalar sinfining kengayib borishi albatta yangi-yangi dasturlar yaratishga olib keladi. Yangi dasturlarni yaratish uchun "eski"

dasturlash tillarining imkoniyati yetmay qolganda yoki dastur yaratish jarayonini mukammallashtirish uchun yangi dasturlash tiliga ehtiyoj paydo bo'ladi.

Obyektga yo'naltirilgan dasturlash – bu dasturlashning shunday yangi yo'nalishiki, dasturiy sistema o'zaro aloqada bo'lgan obyektlar majmuasi sifatida qaraladi va har bir obyektni ma'lum bir klassga mansub hamda har bir klass qandaydir shajarani hosil qiladi deb hisoblanadi. Alohida olingan klass ma'lumotlar to'plami va ular ustida bajariladigan amallarning to'plami sifatida qaraladi. Bu klassning elementlariga faqat shu klassda aniqlangan amallar orqali murojaat qilish mumkin. Dasturdagi ma'lumotlar va ular ustida bajariladigan amallar o'rtasidagi o'zaro bog'liqlik an'anaviy dasturlash tillariga nisbatan dasturiy sistemalarning ishonchliligini ta'minlaydi. Obyektga yo'naltirilgan dasturlashning eng asosiy tushunchasi obyekt va klass hisoblanadi.

Obyektga yo'naltirilgan dasturlash uzoq-yillar davomida an'anaviy, ya'ni standart hisoblangan dasturlashga nisbatan xos bo'lgan tasavvurlarni bir chetga qo'yishni talab qiladi. Natijada obyektga yo'naltirilgan dasturlash juda ham sodda, ko'rgazmaliligi yuqori bo'lib, dasturiy ta'minot yaratishdagi ko'plab muammolarni hal qilishning juda ajoyib vositasiga aylanadi.

Yuqorida aytilgan fikrlarni hisobga olsak, obyektga yo'naltirilgan dasturlash texnologiyalari quyidagicha masalalarni hal qilishi mumkin:

- an'anaviy dasturlash tillarida mavjud bo'lgan kamchiliklarni bartaraf qilish;
- an'anaviy dasturlash tillari yordamida yechib bo'lmaydigan yoki juda katta qiyinchiliklar bilan yechilishi mumkin bo'lgan masalalarni hal qilish;
- qayta ishlash mumkin bo'lgan ma'lumotlar va ularning tiplari doirasi an'anaviy dasturlash tillariga nisbatan ancha keng;
- foydalanuvchilar uchun qulay bo'lgan muloqot interfeysini yaratish;

■ kiritilayotgan va chiqarilayotgan turli tipdagi ma'lumotlarni nazorat qilish;

■ yangi tipdagi ma'lumotlar, klasslar va modullarni osongina tashkil etish va ma'lumotlarni nazorat qilish;

■ multimedia va animatsion vositalardan foydalanib, turli darajadagi tovushli va harakatli effektlarni hosil qilish va qayta ishlash;

■ ma'lumotlar bazasi va undagi ma'lumotlar ustida amallarni bajarish, SQL so'rovnomalari yordamida ma'lumotlarni qidirib topish kabi masalalarni juda osonlik bilan hal qilish;

■ OLE konteyneri yordamida Windows muhiti uchun mo'ljallangan ilovalardagi obyektlar bilan ishlash;

■ foydalanuvchilar uchun yaratilgan dasturiy ta'minotdan foydalanish uchun yordamchi ma'lumotnomalar tizimini yaratish;

■ dasturiy ta'minotni boshqa kompyuterlarga ko'chirish uchun o'rnatuvchi disklarni yaratish;

■ dastur matnini tashkil qilishda yuzaga kelishi mumkin bo'lgan xatoliklar bilan ishlash masalasini hal qilish va h.k..

Ko'rinib turibdiki, yechilayotgan masalalarni an'anaviy dasturlash tillari yordamida yechishda yuzaga kelishi mumkin bo'lgan kattakon bo'shliqni obyektga yo'naltirilgan dasturlash texnologiyalari to'ldiradi hamda zamonaviy dasturlashning ko'p-lab talablariga javob beradi.

Nazorat savollari

1. Dasturlash tillari va ularning kelib chiqishi.
2. Dasturlash tillari qanday tasniflanadi?
3. Obyektga yo'naltirilgan dasturlash texnologiyalari nimaga asoslanadi?
4. Zamonaviy dasturlash tillariga qaysi dasturlash tillari kiradi?
5. Ochiq kodli dasturlash tillari nima uchun ishlatiladi?

Test savollari

1. Obyektga yo‘naltirilgan dasturlash tillari to‘g‘ri ko‘rsatilgan qatorni belgilang.

a) C++, Java
b) Fortran, Java
c) Assembler, C#
d) Cobol, Html

2. Go dasturlash tili qaysi kompaniya tomonidan ishlab chiqilgan?

a) Microsoft
b) Google
c) C ++
d) Delphi

3. Mashina kodiga asoslangan dasturlash tili qaysi qatorda to‘g‘ri ko‘rsatilgan?

a) Pascal
b) Assembler
c) Java
d) C#

4. Ochiq kodli dasturlash tili to‘g‘ri ko‘rsatilgan qatorni belgilang.

a) Go
b) C++
c) Ruby
d) Java

5. Veb sahifalarni ishlab chiqishga mo‘ljallangan dasturlash tili?

a) C++
b) Lisp
c) FORTRAN
d) PHP

II bob. BORLAND C++ BUILDER 6 INTEGRALLASHGAN SOHASI, UNING TASHKIL ETUVCHILARI

Tayanch so'zlar: dastur, dasturlash tizimi, komponentalar kutubxonasi, ilova, komponentalar palitrasi, Borland C++ Builder 6 integrallashgan sohasi.

2.1. Borland C++ Builder 6 dasturlash tizimi va uning asosiy xususiyatlari

Kompyuterda dasturlash oxirgi-yillarda juda tez rivojlanib dastur tuzishga qiziquvchilar soni oshib bormoqda. Dasturlashtirish vositalarining zamonaviy texnologiyalari yuqori darajali ilovalarni yaratish imkoniyatlarini amalga oshirish uchun qaratilgan.

C++ dasturlash vositasining yaratilishi esa nafaqat professional dasturchilar, balki oddiy dastur tuzuvchilar uchun ham keng yo'l ochib berdi. Juda qisqa vaqt ichida Borland kompaniyasi C++ning bir qator lahjalarini ishlab chiqdi. C++ ning oxirgi lahjalarida ma'lumotlar bazasini yaratish va qayta ishlash, Internet tarmog'idan foydalangan holda ma'lumotlar alamashinuvini o'rnatish, dasturlashning obyektga yo'naltirilgan modelini keng qo'llash, vizual dasturlashda yangi komponentalar kutubxonasini (VCL) yaratish kabi asosiy farqli imkoniyatlarni o'z ichiga oladi.

Dasturlashda qulay bo'lgan Borland C++ Builder 6 platformasini to'liqroq ko'rib chiqamiz. Borland C++ Builder 6 – Windows operatsion tizimida dastur yaratish obyektga yo'naltirilgan dasturlash muhitidir. Borland C++ Builder 6 dasturlash muhitida dastur tuzish zamonaviy vizual loyihalash texnologiyalariga asoslangan bo'lib, unda dasturlashning obyektga yo'naltirilgan g'oyasi mujassamlashgan.

Borland C++ Builder 6 – bir necha muhim ahamiyatga ega bo'lgan texnologiyalar kombinatsiyasini o'zida mujassam etgan:

– yuqori darajadagi mashinali kodda tuzilgan kompilyator;
– obyektga yo'naltirilgan komponentalar modellari;
– dastur ilovalarini vizual tuzish;
– ma'lumotlar bazasini tuzish uchun yuqori masshtabli vosita.

Borland C++ Builder 6 – Windows muhitida ishlaydigan dastur tuzish uchun qulay bo'lgan vosita bo'lib, kompyuterda dastur yaratish ishlarini avtomatlashtiradi, xatoliklarni kamaytiradi va dastur tuzuvchi mehnatini yengillashtiradi. Borland C++ Builder 6 da dastur zamonaviy vizual loyihalash texnologiyasi asosida obyektga yo'naltirilgan dasturlash nazariyasini hisobga olgan holda tuziladi. Ma'lumki, dastur tuzish sermashaqqat jarayon, lekin Borland C++ Builder 6 tizimi bu ishni sezilarli darajada soddalashtiradi va masala turiga qarab dastur tuzuvchi ishining 50–80% ni tizimga yuklaydi.

Borland C++ Builder 6 tizimi dasturni loyihalash va yaratish vaqtini kamaytiradi hamda Windows muhitida ishlovchi dastur ilovalarini tuzish jarayonini osonlashtiradi.

Borland C++ Builder 6 o'zida bir qancha zamonaviy ma'lumotlar bazasini boshqarish tizimlari, dasturlash texnologiyalarini ham ma'lumotlar bazasini yaratishda ishlatadi.

Komponentalarning obyektga yo'naltirilgan modeli tayyor obyektlardan foydalanib yangi ilovalar yaratish, shu bilan birga foydalanuvchining shaxsiy obyektlarini yaratish imkonini beradi.

Borland C++ Builder 6 ning standart obyektlari 270 tadan ortiq asosiy sinflarni birlashtiradi. Borland C++ Builder 6 sinflari murakkab iyerarxik strukturaga ega bo'lgan vizual komponentalar kutubxonasini (Visual Component Library – VCL) tashkil qiladi. VCL tarkibiga kiruvchi yuzlab sinflar mavjud.

Vizual dasturlash texnologiyasida obyekt deganda muloqat oynasi va boshqarish elementlari (kiritish va chiqarish maydoni, buyruq tugmalari, pereklyuchatellar va boshqalar) tushuniladi.

Borland C++ Builder 6 da dasturlash ikkita o'zaro ta'sir etuvchi bir-biri bilan bog'liq jarayon asosida tashkil qilinadi:
– dasturni vizual loyihalash jarayoni;
– dastur kodlarini kiritish (yozish) jarayoni.

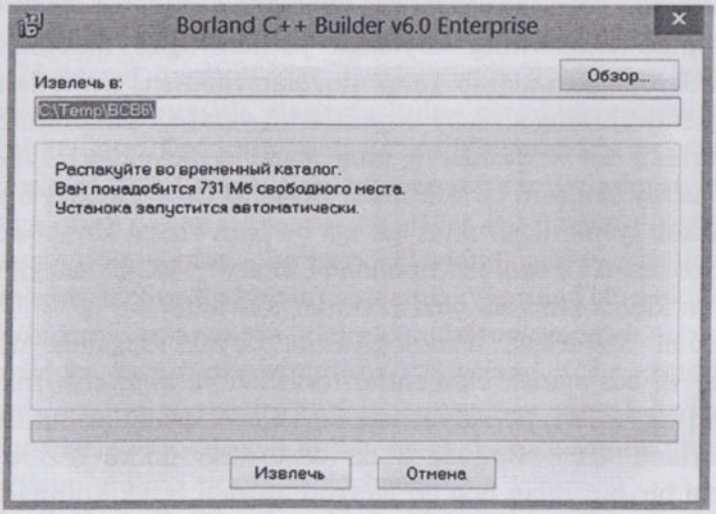
Vizual loyihalash jarayonida dasturda yaratilayotgan ilovaning dizayni shakllanadi. Dasturchi vizual loyihalash jarayonini bajarganda Borland C++ Builder 6 avtomatik ravishda dastur kodini yaratishni boshlaydi. Dasturchi loyihasini ishlashi mobaynida dastur kodini C++ tilining maxsus operatorlari bilan to'ldiradi.

2.2. Borland C++ Builder 6 dasturini o'rnatish jarayoni

Borland C++ Builder 6 dasturi Windows operatsion tizimida ishlashga mo'ljallangan. Dasturni kompyuterga yuklash uchun bir nechta qadamlarni amalga oshirish talab etiladi.

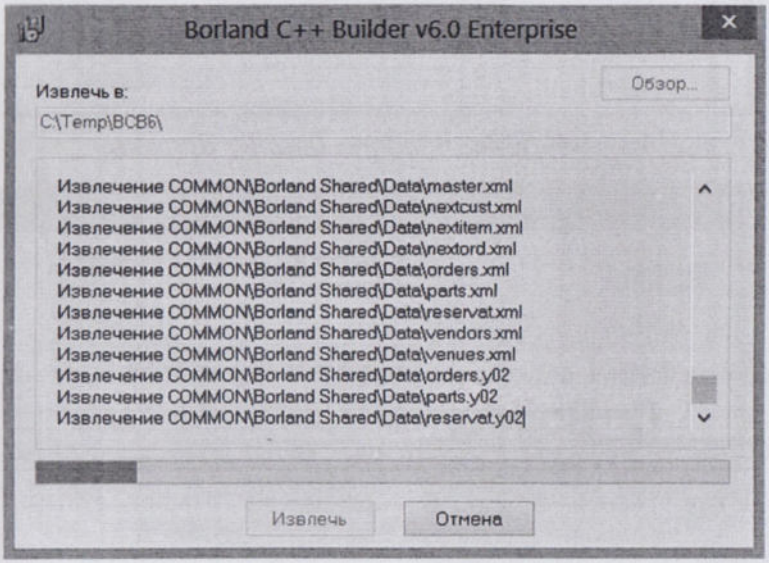
1-qadam. Borland C++ Builder 6 yozilgan diskni oching. Ochilgan katalogdan bcb6.exe faylni toping. bcb6.exe faylini ustida 2 marta sichqoncha bosing. Faylning hajmi 186 Mb tashkil etadi.

2-qadam. Kompyuterning xotirasiga bcb6.exe fayli yuklanishi uchun arxivdan chiqarish talab qilinadi (2.1-rasm). Fayllarni vaqtinchalik katologga (C:\Temp\BCB6) yuklanishi uchun kompyuter xotirasidan 731 Mb bo'sh joy talab etiladi.



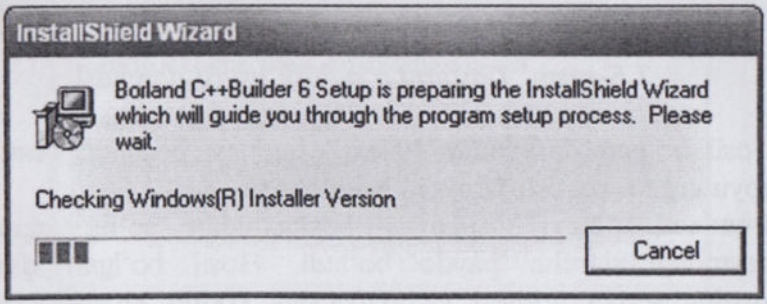
2.1-rasm. *Fayllarni vaqtinchalik katalogga yuklash.*

Kompyuter xotirasida bo‘sh joy mavjud bo‘lsa “Извлечь” tugmasi bosiladi va natijada jarayon ishga tushadi. Fayllarni arxivdan chiqarish jarayoni quyidagi ko‘rinishda amalga oshiriladi (2.2-rasm).

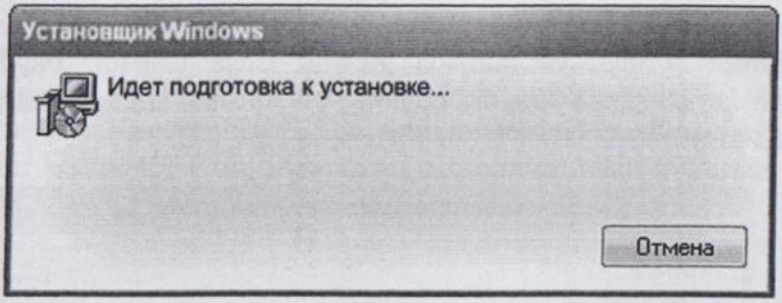


2.2-rasm. *Fayllarni arxivdan chiqarish jarayoni.*

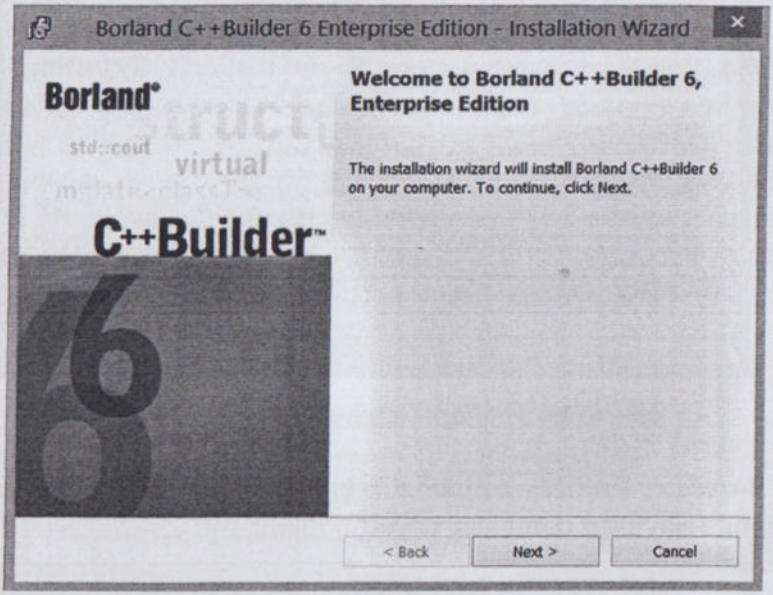
3-qadam. Fayllar vaqtinchalik katalogga yuklanganidan so‘ng ekranda quyidagi darchalar paydo bo‘ladi:



2.3-rasm. *Install Shield Wizard darchasi.*



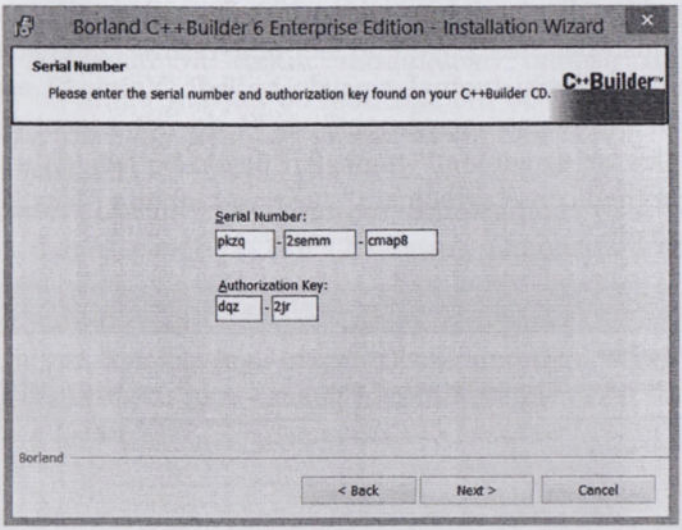
2.4-rasm. *Windows Installer darchasi.*



2.5-rasm. *Dasturni o'rnatish ustasi darchasi.*

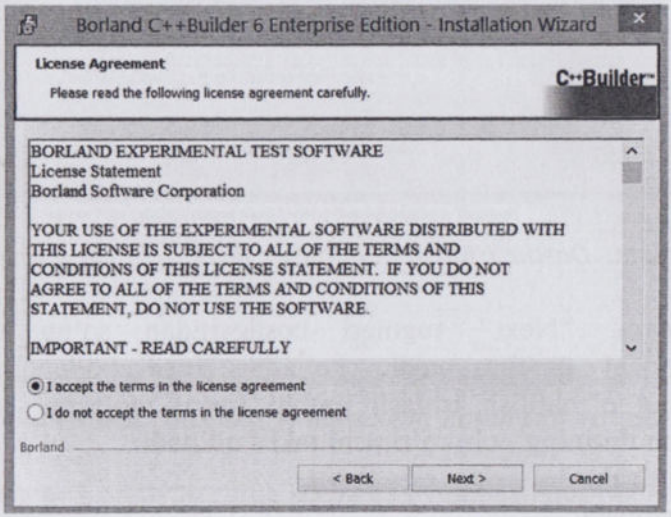
Hosil bo'gan darchadan "Next" tugmasi bosiladi. Natijada kompyuterga o'rnatish jarayoni boshlanadi.

4-qadam. "Next" tugmasi bosilganidan so'ng ekranda 2.6-rasmdagi darcha paydo bo'ladi. Hosil bo'lgan darcha dasturning seriya nomeri va aktivatsiya kodini kiritishni talab etadi.



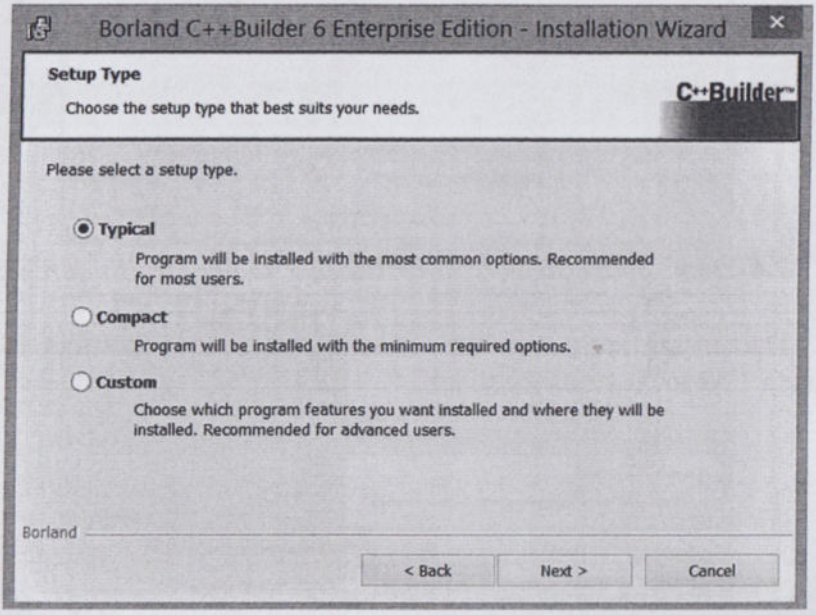
2.6-rasm. *Seriya nomeri va aktivatsiya kodini kiritish darchasi.*

Dasturning seriya nomeri va aktivatsiya kodi kiritilganidan keyin "Next" tugmasi bosiladi.



2.7-rasm. *Litsenziya shartlarini qabul qilish darchasi.*

5-qadam. Borland tomonidan qo'yilgan litsenziya shartlarini qabul qilish darchasi paydo bo'ladi (2.7-rasm). Darchada tanlash imkonini beruvchi tugmalar paydo bo'ladi. Ya'ni, "I accept the terms in the license agreement" va "I do not accept the terms in the license agreement" tugmalari paydo bo'ladi. "I accept the terms in the license agreement" tugmasi tanlansa "Next" tugmasi aktivlashadi.



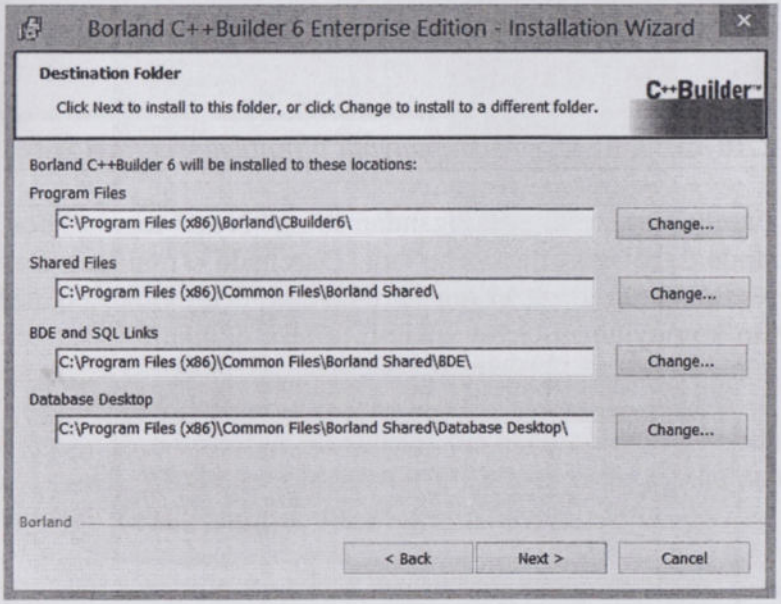
2.8-rasm. *Dastur ishlash muhitining turini tanlash darchasi.*

6-qadam. "Next" tugmasi bosilganidan so'ng ekranda 2.8-rasmdagi darcha paydo bo'ladi. Hosil bo'lgan darcha dasturning o'rnatilish ishlash muhiti turini tanlashni so'raydi. Ishlash muhitining uch ko'rinishi taklif qilinadi:

- Typical;
- Compact;
- Custom.

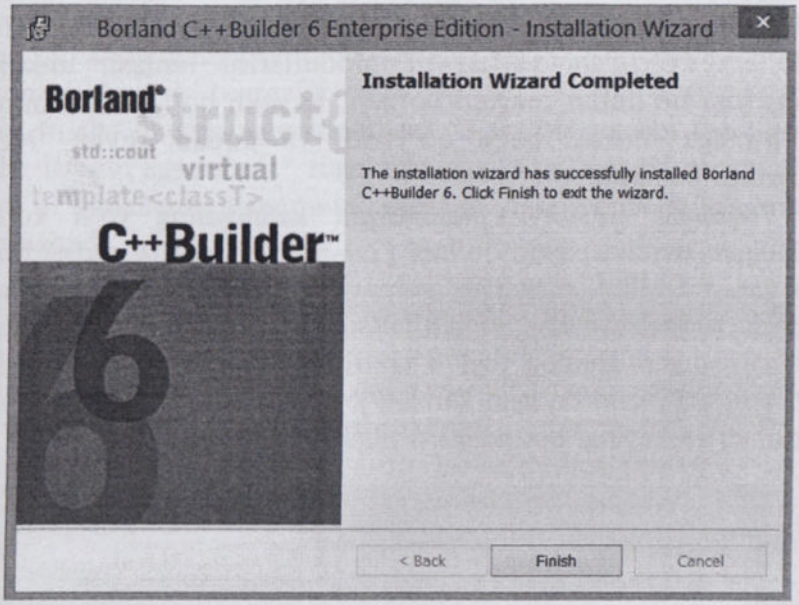
Bunda dasturning tipik ko‘rinishi (Typical), dasturning ixcham ko‘rinishi va dastur modullarini tanlash imkonini beruvchi bo‘limlar mavjud bo‘ladi. Kerakli bo‘lgan bo‘limning to‘g‘risiga maxsus belgi qo‘yiladi va undan keyin “Next” tugmasi bosiladi.

7-qadam. Dastur o‘rnatiladigan katalogning yo‘li ko‘rsatiladigan darcha paydo bo‘ladi (2.9-rasm). Darchada dasturning to‘rtta modulini o‘rnatish uchun har biriga alohida katalog yolini ko‘rsatish talab etiladi. “Change” tugmasi orqali dastur o‘rnatiladigan katalog joyi o‘zgartirilishi mumkin. Agar dastur o‘rnatilishi lozim bo‘lgan katalog joyi mos kelsa “Next” tugmasi bosiladi va keyingi bosqichga o‘tiladi.



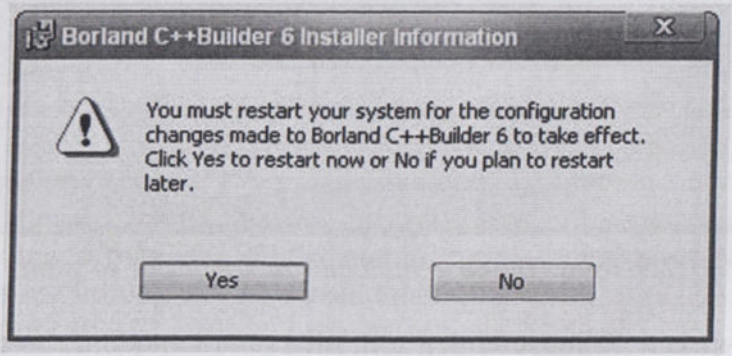
2.9-rasm. *Dastur o‘rnatiladigan kataloglar ro‘yxati.*

8-qadam. Kompyuterning xotirasida yetarli miqdorda bo‘sh joy bo‘lsa, dastur muvaffaqiyatli o‘rnatilganligi haqida darcha paydo bo‘ladi (2.10-rasm).



2.10-rasm. *Dastur muvaffaqiyatli o'rnatilganligi haqida darcha.*

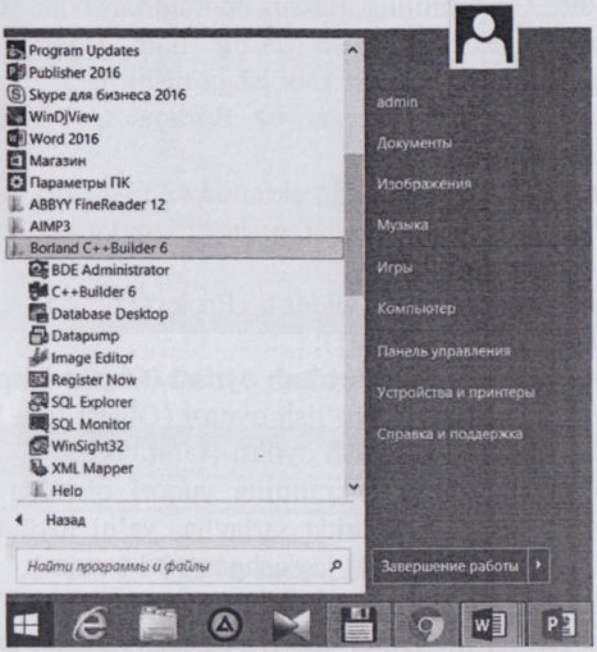
Barcha jarayonlar tugatilgandan so'ng "Next" tugmasi bosiladi. Ekranda 2.11-rasm paydo bo'ladi. Darchada o'rnatilgan Borland C++ Builder 6 dasturining barcha imkoniyatlari to'liq ishlashi uchun kompyuterni qayta yuklash tavsiya qilinadi.



2.11-rasm. *Borland C++ Builder 6 Installer Information darchasi.*

Paydo bo‘lgan darchada “Yes” tugmasini bosing. Shundan so‘ng dasturni yuklash jarayoni o‘z nihoyasiga yetadi.

Dastur yuklanib bo‘lingandan keyin pusk bo‘limida yangi darcha paydo bo‘ladi (2.12-rasm).



2.12-rasm. *Dasturni pusk menyusida ko‘rinishi.*

Hosil bo‘lgan dasturlar menyusida bir nechta dastur modullari paydo bo‘ladi. Modullar ichidan C++ Builder 6 dasturi tanlanadi va dasturda ishlash imkoniyati paydo bo‘ladi.

2.3. Borland C++ Builder 6 dasturining integrallashgan sohasi

Borland C++ Builder 6 dasturida dasturlash ikkita o‘zaro ta’sir etuvchi bir-biri bilan bog‘liq jarayon asosida tashkil qilinadi:

– dasturni vizual loyihalash jarayoni;
– dastur kodlarini konsol muhitida kiritish jarayoni.

Vizual loyihalash jarayonida dasturda yaratilayotgan ilovaning dizayni shakllanadi. Dasturchi vizual loyihalash jarayonini bajarganda Borland C++ Builder 6 avtomatik ravishda dastur kodini yaratishni boshlaydi. Dasturchi loyihasini ishlashi mobaynida dastur kodini C++ tilining maxsus operatorlari bilan to'ldiradi.

Borland C++ Builder 6 dastur ham Windows amaliyot tizimining boshqa dasturlari kabi ishga tushiriladi:

Пуск => Все программы => Borland C++ Builder 6 => C++ Builder 6.

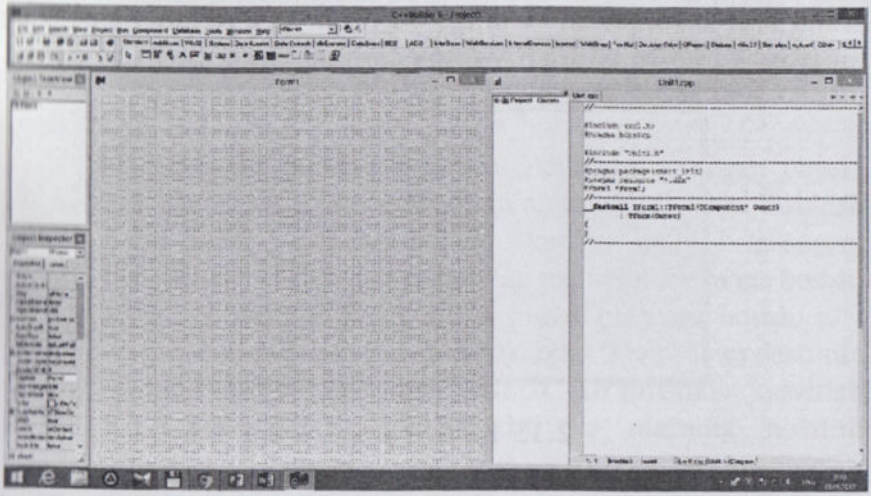
Dastur yuklangandan keyin ekranda (2.13-rasm) darcha paydo bo'ladi. Borland C++ Builder 6 dasturi quyidagi beshta asosiy darchani o'z ichiga oladi:

1. Bosh oyna – C++ Builder 6 (Project1);
2. Forma oynasi (Forma1);
3. Obyekt xossalarini tahrirlash oynasi (Object Inspector);
4. Obyektlar ro'yxatini ko'rish oynasi (Object tree View);
5. Dastur kodlarini kiritish oynasi (Unit.cpp).

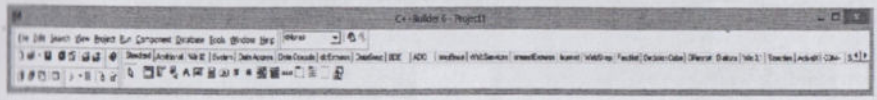
Bosh oyna (Project1) ekranning yuqori qismida joylashgan bo'lib, uning birinchi qatorida sarlavha, ya'ni loyihaning nomi (C++ Builder 6 – Project1) joylashgan (2.14-rasm).

Ikkinchi qatorda asosiy menyular qatori gorizontal ko'rinishda joylashgan. Asosiy menyular qatori dasturni yaratish uchun kerak bo'ladigan barcha buyruqlar va funksiyalarga murojaat qilish imkonini yaratadi.

Keyingi qatorning chap tarafida tezkor murojaat etish tugmalari mavjud. Bajaradigan vazifasiga qarab ketma-ketlikda birlashtirilgan. Ular tez-tez ishlatiladigan buyruqlarga tezkor murojaat etish imkonini beradi. O'ng tarafida vizual komponentalar palitrasi VCL (Visual Component Library, vizual komponentalar kutubxonasi) keltirilgan. Windows operatsion tizimi ilovalarni yaratish uchun vizual komponentalarni o'z ichiga oladi. Vizual komponentalar palitrasi bir nechta qismlardan iborat guruhlarga bo'lingan. Bu vizual komponentalar palitrasi yordamida tezkor va oson usulda dasturlarni yaratish mumkin.



2.13-rasm. *Borland C++ Builder 6 integrallashgan sohasi.*

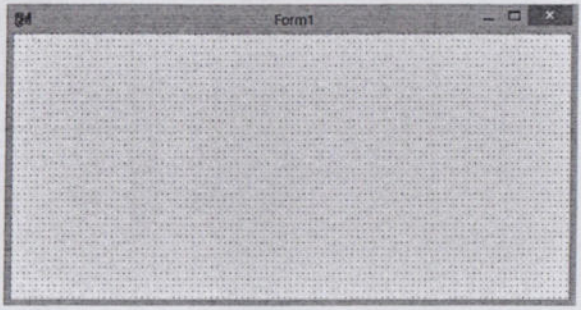


2.14-rasm. *Bosh oyna.*

Forma oynasi (Forma 1) yangi yaratilishi kerak bo‘ladigan dasturning ko‘rinishi hisoblanadi (2.15-rasm). Forma oynasi C++ Builder 6 ilovalari uchun asos bo‘lib, unda yaratilayotgan dasturga komponentalarni joylashtirish mumkin. U dasturning sarlavhasidan boshlanadi.

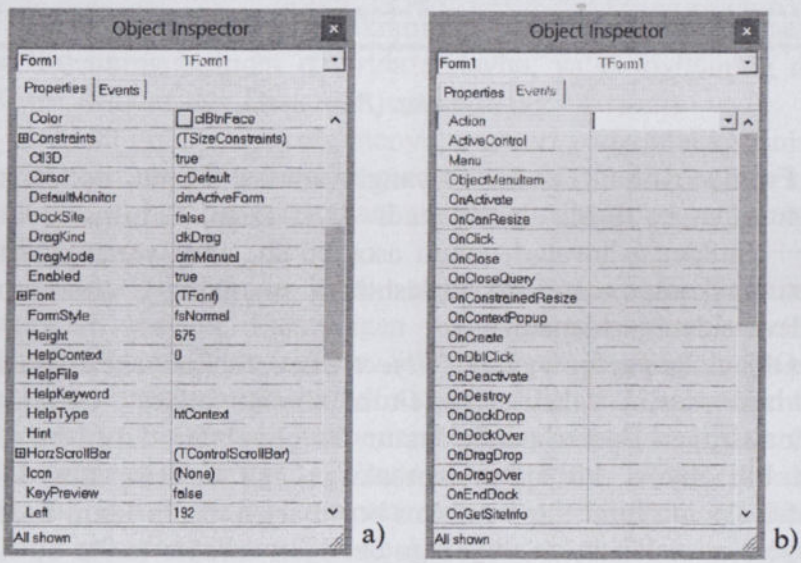
Object Inspector oynasi *(Object Inspector)* obyekt xossalari va hodisalarini tahrirlash uchun xizmat qiladi. Obyektga yo‘naltirilgan dasturlashda dastur bu obyektlar tizimi bo‘lib, har bir obyekt bir qator xossalarga ega bo‘lishi mumkin. Xossa esa ma’lumotlar va ularni boshqarish usullaridan iborat. Obyekt xossalari bu – obyektga berilgan xarakteristika bo‘lib, uning ko‘rinishi, joylashishi va holatidir. Bundan tashqari obyekt turli hodisalarni ham o‘rnatishi mumkin. Hodisa deb

bajarish, boshqarish usuliga aytiladi, masalan, sichqonni bosish, kursorni siljitish va hokazo amallarga aytiladi.



2.15-rasm. *Forma oynasi.*

Object Inspector oynasi xossa va hodisalar parametrlarini o'rnatish uchun mo'ljallangan bo'lib, u ikkita sahifadan iborat: Propierities (xossalar: 2.16-a rasm) va Events (hodisalar: 2.16-b rasm).

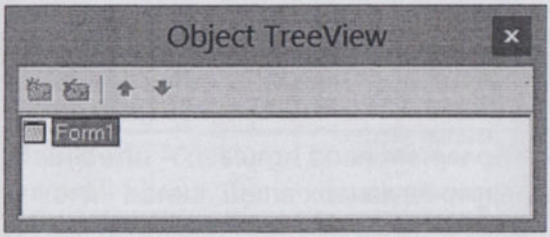


2.16-rasm. *Object Inspector oynasi.*
a) "Properties" darchasi. b) "Events" darchasi.

Propierities sahifasi ajratilgan obyekt yoki komponentaning xossalarini o'rnatadi. Masalan, Caption (yozuv) yordamida obyekt yozuvini o'rnatish mumkin, Color (rang) xossasi obyektning rangini o'rnatadi.

Events (hodisalar) sahifasida Forma ilovasida tanlangan obyekt uchun dastur bajarilishi jarayonida hodisa, ya'ni uni ishga tushirish holati belgilanadi. Har bir holatning standart nomi belgilangan. Masalan, OnClick – sichqonchaning chap tugmasini bir marta bosish, OnDblClick – sichqonchaning chap tugmasini bir marta bosish.

Obyektlar ro'yxatini ko'rish oynasi (Object Tree View) dasturda ishlatilayotgan komponentalarni daraxt ko'rinishida tasvirlab beradi. Komponentalarni joylashuvini va ularning holatini ko'rsatib turadi (2.17-rasm).



2.17-rasm. *Obyektlar ro'yxatini ko'rish oynasi.*

Dastur kodlarini kiritish oynasi (Unit.cpp) yaratilayotgan yangi dasturning kodini (matni) kiritish va uni tahrirlash uchun foydalaniladi (2.18-rasm). Dasturda ma'lum jarayonini amalga oshirishi uchun kerak bo'ladigan operatorlar ketma-ketligi kiritiladi.

Asosiy menyular qatori quyidagilarni o'z ichiga olgan:

– **File** bo'limi fayllar ustida ish bajarish uchun kerakli buyruqlarni o'z ichiga olgan;

– **Edit** bo'limi fayl ichidagi ma'lumotlarni tahrirlash uchun kerakli buyruqlarni o'z ichiga olgan;

– **Search** bo'limi fayllar, modullar tarkibidagi kerakli bo'laklarni izlab topish imkonini beruvchi buyruqlar to'plami;

– **View** bo‘limi dastur oynasiga kerakli instrumentlar palitrasini o‘rnatish, loyiha kodi, shuningdek, loyiha menedjerini ochish va ko‘rish uchun mo‘ljallangan buyruqlar to‘plami;

– **Compile** bo‘limi loyiha va dasturlarni ishga tushirish, kompilyatsiya buyruqlaridan tashkil topgan;

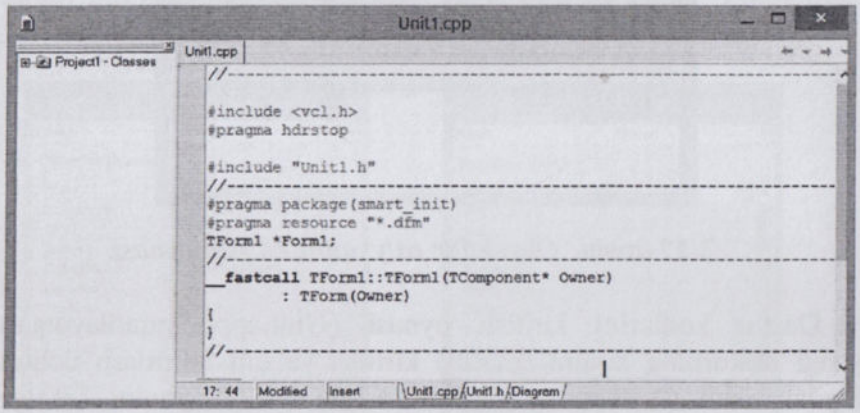
– **Run** dasturni ishga tushirish va to‘xtatish uchun kerak bo‘ladigan buyruqlar to‘plami;

– **Options** bo‘limi muhit oynasining konfiguratsiya parametrlarini o‘rnatish uchun xizmat qiladigan buyruqlar to‘plami;

– **Tools** servis xizmatidan va qo‘shimcha utilitalardan foydalanish imkonini beradi;

– **Help** yordam chaqirish uchun mo‘ljallangan buyruqlar to‘plami.

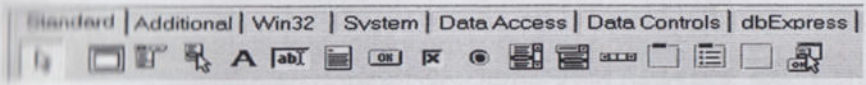
Asosiy menyular qatori ko‘p bo‘limlarni o‘z ichiga oladi.



2.18-rasm. *Dastur kodlarini kiritish oynasi.*

2.4. Borland C++ Builder 6 dasturining komponentalar palitrasi

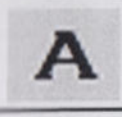
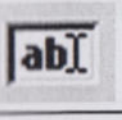
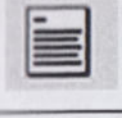
Standart komponentalar palitrasi. Komponentalar palitrasining birinchi sahifasida 16 ta obyektlar joylashgan bo‘lib, ko‘p dasturlarni yaratishda albatta ushbu obyektlardan foydalaniladi.

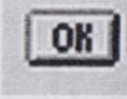
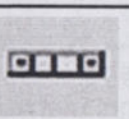


2.19-rasm. *Standart komponentalar palitrasi.*

Har bir sahifadagi komponentalar palitrasi va joylashishini o'zgartirish mumkin. Demak, siz mavjud komponentalar o'rnini o'zgartirish bilan birgalikda ularning tarkibiga yangilarini qo'shishingiz mumkin.

Borland C++ Builder 6 standart komponentalari va ularni ishlatish haqidagi qisqacha izoh quyidagi jadvalda keltirilgan:

	**Kursor** – komponent hisoblanmaydi, balki biror obyektni tanlashdan voz kechish vositasidir.
	**Frame** – boshqa komponentalarni joylashtirish uchun darcha sifatida xizmat qiladi. Bu komponenta forma dizayneriga juda o'xshab ketadi.
	**MainMenu** – dasturga bosh menyu qo'shish imkonini beradi. Items xususiyati orqali yangi menyu bo'limi qoshiladi.
	**PopupMenu** – yordamchi menyuni yaratish. Bu menyu sichqonchaning o'ng tugmasi bosilganda ko'rinadi.
	**Label** – matnlarni ekranda namoyish qilish uchun qo'llanadi. Caption xususiyati orqali matnlar kiritiladi.
	**Edit** – qisqa matnlarni namoyish qilishi va dastur bajarilish vaqtida foydalanuvchiga o'z matnini kiritish imkonini beradi.
	**Memo** – ko'p qatorli mantlar bilan ishlashni ko'zda tutadi. Lines xossasi orqali ma'lumotlar kiritiladi.

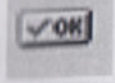
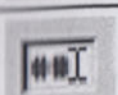
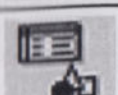
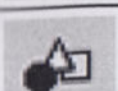
	**Button** – dastur bajarilish vaqtida tugma bosilishi bilan biror amal yoki jarayon bajarilishini ko'zda tutadi.
	**CheckBox** – chap tomonida bir nechta variantni tanlanishiga imkoniyat beradigan matn satrini akslantiradi.
	**RadioButton** – bir nechta holatlardan birini tanlash imkonini beradigan aylana ko'rinishidagi tugma.
	**ListBox** – ro'yxatli ma'lumotlarni namoyish qilishga mo'ljallangan komponenta.
	**Combob ox** – tahrir sohasi hamda matn variantlarining ro'yxati kombinatsiyasini tanlash uchun yaratadi.
	**Scrollbar** – o'tkazish yo'lakchasi. Ko'pincha tahrirlanadigan yoki ko'rinadigan ma'lumotlarni ekrandan chiqib ketganda foydalaniladi.
	**GroupBox** – Windows operatsion tizimiga formadagi obyektlarning qanday joylashganligini bildirish uchun qo'llaniladi.
	**RadioGroup** – ro'yxatlar hosil qilish uchun va ro'yxatlardan birini tanlash orqali dastur bajarilishini amalga oshirishda foydalaniladi.
	**Panel** – boshqa komponentalarni o'z ichiga olishi mumkin bo'lgan bo'sh panelni yaratadi.
	**Action List** – interfeys elementlari va dastur o'rtasidagi o'zaro aloqani boshqaradi.

Additional komponentalar palitrasi. Additional komponentalar palitrasi dasturning foydalanuvchi interfeysini yanada yaxshilash imkonini beradi.



2.20-rasm. *Additional komponentalar palitrasi.*

Additional komponentalar palitrasida joylashgan komponentalar quyidagi jadvalda keltirilgan:

	BitBtn – ma'lum bir jarayonini amalga oshirish uchun foydalaniladi. Buttonga o'xshash tugma, lekin unda tasvirlarni o'rnatish mumkin.
	SpeedButton – buyruqlarga tezkor murojaat panelini yaratish tugmasi. Odatda bu tugmaga faqat tasvir (glyph) joylashtiriladi.
	MaskEdit – Edit ga o'xshash, lekin kiritishni formatlash mumkin. Format EditMask xususiyatida aniqlanadi.
	StringGrid – matnli ma'lumotlarni jadval ko'rinishida tasvirlashda foydalaniladi.
	DrawGrid – ixtiyoriy tipdagi ma'lumotlarni jadval ko'rinishida chiqaradi.
	Image – formada grafik tasvirlarni namoyish qiladi. Picture xususiyati orqali kerakli tasvir fayli o'rnataladi.
	Shape – formada oddiy grafik obyektlarni (aylana, kvadrat) yaratishda foydalaniladi.

	ScrollBox – formada ekranga sig'maydigan obyektlarni ko'rsatish imkonini beruvchi yo'lakchalarni hosil qiladi.

System komponentalar palitrasi. Windows operatsion tizimida multimedia ilovalarini yaratishda foydalaniladi.



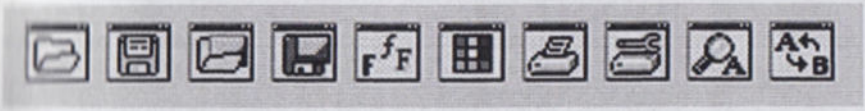
2.21-rasm. *Additional komponentalar palitrasi.*

System komponentalar palitrasida joylashgan komponentalar quyidagi jadvalda keltirilgan:

	Timer – dasturda vaqt oraliqlarini o'rnatish va ularni nazorat qilish uchun foydalaniladi.
	PaintBox – chizish uchun joy. Holatlar qayta ishlovchisiga Paintboxdagi sichqonchaning mos koordinatalari qaytariladi.
	MediaPlayer – turli shakllardagi fayllarni ishlatish va multimediya uskunalarini boshqarish uchun formada panel yaratadi.
OLE	OleContainer – dasturda Windows muhitida turli xil dasturlar o'rtasida ma'lumotlarni uzatish imkoniyatiga ega bo'lgan OLE obyektlarini kiritish va unga ulanish mexanizmini joriy etadi.

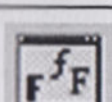
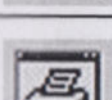
Dialogs komponentalar palitrasi. Dialogs komponentalar palitrasida Windows muhitining standart muloqot oynalarini chaqiruvchi komponentalar joylashgan. Muloqot oynalarining tashqi ko'rinishi Windows muhitining versiyasiga bog'liq bo'ladi.

Bu sahifadagi obyektlar dastur bajarilish vaqtida ko'rinmaydi va mos muloqot oynalarini dasturiy yo'l bilan chaqirish kerak bo'ladi.



2.22-rasm. Dialogs komponentalar palitrasi.

Dialogs komponentalar palitrasida joylashgan komponentalar quyidagi jadvalda keltirilgan:

	OpenDialog – fayllarni ochish uchun dasturda dialog darchasini yaratadi. Faylning nomi va kengaytmasiga filtr o'rnatish mumkin.
	SaveDialog – fayllarni saqlash uchun dasturda dialog darchasini yaratadi.
	OpenPictureDialog – grafik tasvirli fayllarni ochish uchun dasturda dialog darchasini yaratadi. Tasvirni dialog oynasida ko'rish imkoniyati paydo bo'ladi.
	SavePictureDialog – grafik tasvirli fayllarni saqlash uchun dasturda dialog darchasini yaratadi.
	FontDialog – dasturda shriftning o'lchami va ko'rinish turini tanlash imkonini beruvchi dialog darchasini yaratadi.
	ColorDialog – Windows ranglar palitrasidan rang tanlash uchun dialog darchasini yaratadi.
	PrintDialog – kompyuterda o'rnatilgan bosmaga chiqarish qurilmalarini tanlash va ularni boshqarish dialog darchasini yaratadi.

	PrinterSetupDialog – bosib chiqarish qurilmasining parametrlarini o'rnatish uchun dialog darchasini yaratadi.
	FindDialog – dasturda qidirish parametrlarini sozlash imkoniyatiga ega bo'lgan matn terish dialogini ochadi.
	ReplaceDialog – dasturda topilgan so'z fragmentini almashtirish imkoniyatiga ega bo'lgan matn qidirish dialogini ochadi.

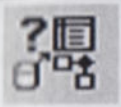
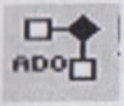
ADO komponentalar palitrasi. Active Data Objects (ADO) ma'lumotlar bazasi bilan aloqa o'rnatish imkonini beradi. Ma'lumotlar bazasi Microsoft OLE komponentalari bog'lanadi. Bu komponentalar palitrasi ma'lumotlar bazasi bilan ishlashga mo'ljallangan.



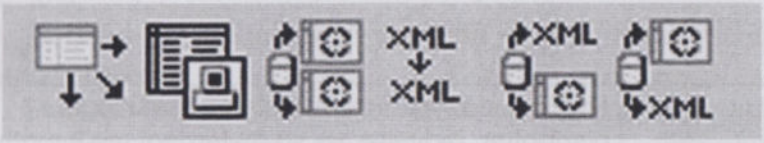
2.23-rasm. *ADO komponentalar palitrasi.*

ADO komponentalar palitrasida joylashgan komponentalar quyidagi jadvalda keltirilgan:

	ADOConnection – ma'lumotlar bazasiga ulanish imkonini beruvchi komponenta.
	ADOCommand – mavjud ma'lumotlar bazasi asosida SQL buyruqlarni bajarish imkonini beruvchi komponenta.

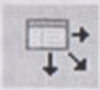
	ADODataSet – ADO ma'lumotlar bazasining bir yoki bir nechta jadvaliga kirishni ta'minlaydi.
	ADOTable – obyekti ma'lumotlar bazasidagi mavjud jadval bilan muloqot o'rnatish uchun xizmat qiladi.
	ADOQuery – ADO so'rovi bo'lib, ADO ma'lumot bazasidan ma'lumotlarni olish uchun SQL buyruqlarini bajarish imkonini beradi.
	ADOStoredProc – ADOga saqlangan protsedura bo'lib, ADO interfeysi yordamida ilovalarga saqlangan ma'lumotlarga murojaat etish imkonini beradi.
	RDSConnection –RDS aloqasi bo'lib, ushbu komponentadan server ilovalarini yaratish uchun ishlatiladi.

Data Access komponentalar palitrasi. Data Access komponentalar palitrasi dasturda ma'lumotlar bazasini boshqarish elementlarini yaratishga xizmat qiladi.



2.24-rasm. *Data Access komponentalar palitrasi.*

Data Access komponentalar palitrasida joylashgan komponentalar quyidagi jadvalda keltirilgan:

	DataSource – obyekti bevosita TTable yoki TAdoTable ga bog'lanib, ma'lumotlar bazasidagi yozuvlarni tahrirlash, ularga murojaat qilish imkonini beradi.

	ClientDataSet – mijoz texnologiyalarga asoslangan ma'lumotlar to'plami. Mijoz texnologiyalariga asoslangan ma'lumotlar bazasini yaratishda foydalaniladi.
	DataSetProvider – ma'lumotlar to'plami provayderi. Ma'lumotlarni keshlash kerak bo'lgan ma'lumotlar bazasida foydalaniladi.
	XML Transform – XML hujjatlar to'plamini ma'lumotlar paketiga aylantirish va orqaga qaytarish uchun foydalaniladi.

Data Controls komponentalar palitrasi. Data Controls komponentalar palitrasi dasturda foydalanuvchi ma'lumotlar bazalarini boshqarish uchun mavjud bo'lgan maxsus vositalarini yaratish imkonini yaratadi.



2.25-rasm. *Data Controls komponentalar palitrasi.*

Data Controls komponentalar palitrasida joylashgan komponentalar quyidagi jadvalda keltirilgan:

	DBGrid – obyekti ma'lumotlar bazasidagi hisobotlar, jadvallar va so'rovlardagi ma'lumotlarni jadval ko'rinishida namoyish etish uchun qo'llanadi.
	DBNavigator – tugmalar majmui bo'lib, ma'lumotlar bazasidagi yozuvlarni boshqarish imkonini beradi.
	DBText – ma'lumotlar bazasidagi jadvallar ustuniga matn qo'shish imkonini beradi.

	DBMemo – ma'lumotlar bazasidagi jadvallar ustuniga ko'p qatorli matn qo'shish imkonini beradi.
	DBImage – ma'lumotlar bazasidagi mavjud tasvirlarni ko'rsatish imkonini beradi.
	DBListBox – ma'lumotlar bazasida ro'yxatli ma'lumotlarni hosil qilish uchun foydalaniladi.
	TDBChart – har xil turdagi grafiklarni hosil qilish uchun foydalaniladi.

Nazorat savollari

1. Borland C++ Builder 6 tizimi qanday dasturlash texnologiyalarini o'zida birlashtirgan?
2. Borland C++ Builder 6 tizimi asosiy ishchi oynasi qanday tarkibiy qismlardan iborat?
3. Borland C++ Builder 6 tizimi oynasining asosiy instrument guruhlari?
4. Borland C++ Builder 6 tizimida qanday komponentalar palitralari mavjud?
5. Standart komponentalar palitrasi qaysi komponentalardan tashkil topgan?

Test savollari

1. Object Inspector darchasining vazifasi?

a) Formada o'rnatilgan komponentalarni keltiradi.

b) Tanlangan komponentaga turli xossa va hodisa parametrlarini o'rnatadi.

c) Komponenta uchun xususiyatlar ro'yxatini tasvirlab beradi.

d) Formada o'rnatilgan komponenta xususiyatlarini boshqaradi.

2. Edit komponentasi nima uchun ishlatiladi?

a) Formada dasturda ishlatiladigan ma'lumotlarni chiqarish uchun

b) Formada ishlatiladigan ma'lumotlarni kiritish va chiqarish uchun

c) Formada tavsiflovchi matnni hisoblash uchun

d) Formani ishga tushirish uchun

3. Qaysi komponenta formaga bir nechta qatorli matn kiritish va chiqarish uchun ishlatiladi?

a) Memo

b) Edit

c) Label

d) RadioGroup

4. Borland C++ Builder 6 dasturining asosiy oynalari nomi?

a) Project, Form, Object Inspector, Object tree View, Component

b) Project, Form, Object Inspector, Object tree View, Standart

c) Project, Form, Object Inspector, Object tree View, Unit

d) Project, Form, Object Inspector, Object tree View, Additional

5. Object Inspector oynasining qaysi sahifasida hodisalar ro'yxati mavjud?

a) Properties

b) Standard

c) Events

d) Additional

III bob. C++ DASTURLASH TILINING ASOSIY KONSTRUKSIYALARI VA ULARDAN FOYDALANISH XUSUSIYATLARI

Tayanch so'zlar: C++ tilining alifbosi, o'zgaruvchilar, o'zgarmaslar, identifikatorlar, operatorlar, derektivalar, obyektlar, ma'lumotlar toifasi, kompilyatorlar, translyatorlar, standart funksiyalar, fayllar, arifmetik va mantiqiy ifodalar, lokal va global o'zgaruvchilar.

3.1. C++ tilining asosiy tashkil etuvchilari

C++ tilining alifbosi. C++ tilining alifbosi o'z ichiga quyidagi simvollarni oladi:

- Lotin alifbosining katta va kichik harflari (A, B, C, ... a, b, c).
- 0 dan 9 gacha bo'lgan arab raqamlari (0, 1 , 2, 3, 4, 5, 6, 7 , 8, 9).
- Maxsus simvollar (', ; * / | \ : ; . % ? ! = < > {} [] () # & ^ ~).

Alifbo simvollaridan tilning leksemasi shakllanadi, jumladan: xizmatchi so'zlar, izohlar, identifikatorlar, amal belgilari, o'zgarmas va o'zgaruvchilar, ajratish belgilari.

Dasturda izohlar istalgan joyda berilishi mumkin. Ular odatda "//" belgisidan keyin yoziladi.

Xizmatchi so'zlar. Tilda ishlatiluvchi, ya'ni dasturchi tomonidan o'zgaruvchilar nomlari sifatida ishlatish mumkin bo'lmagan identifikatorlar xizmatchi so'zlar deyiladi.

C ++ tilida quyidagi xizmachi so'zlar mavjud:

int	extern	else	char	register	for
float	typedef	do	double	static	while
struct	goto	switch	union	return	case
long	sizeof	default	short	break	entry
unsigned	continue	auto	if		

Identifikator. Identifikatorlar lotin harflari, ostki chiziq belgisi va sonlar ketma-ketligidan iborat bo'ladi. Identifikator lotin harfidan yoki ostki chizish belgisidan boshlanishi lozim. Bunda shuni esda tutish lozimki, katta va kichik harflarning farqi bor, misol uchun: MAX, SUMMA, summa.

O'zgaruvchilar. Dastur bajarilishi jarayonida o'z qiymatini o'zgartira oladigan kattaliklar o'zgaruvchilar deyiladi. O'zgaruvchilarning nomlari harflardan 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 qiymatiga ega bo'lishi kerak. O'zgaruvchiga murojaat qilish uning nomi orqali amalga oshiriladi. O'zgaruvchi uchun xotiradan ajratilgan joyning tartib raqami uning adresi hisoblanadi. O'zgaruvchi ishlatilishidan oldin u tavsiflangan bo'lishi lozim. O'zgaruvchilar lokal va global ko'rinishlarda e'lon qilinishi mumkin. Lokal ko'rinishda e'lon qilingan o'zgaruvchilar faqat ma'lum bir jarayonga tegishli bo'ladi. Global o'zgaruvchilar dasturning ixtiyoriy joyida murojaat etish imkonini beradi. O'zgaruvchilarni lokal va global ko'rinishlarda e'lon qilish dastur xususiyatidan kelib chiqadi.

O'zgarmaslar. O'zgaruvchilar kabi o'zgarmaslar ham ma'lumotlarni saqlash uchun mo'ljallangan xotira yacheykalarini o'zida ifodalaydi. O'zgaruvchilardan farqli ravishda ular dastur 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 belgili o'zgarmaslar aniqlangan:

Literal o'zgarmaslar to'g'ridan to'g'ri dasturga kiritiladi. *Masalan:*

```
int myAge=39;
```

Belgili o'zgarmas – bu nomga ega bo'lgan o'zgarmasdir. *Masalan:*

```
const int St=16.
```

Belgili o'zgarmaslarni literal o'zgarmaslarga nisbatan ishlatish qulayroqdir. Chunki, agarda bir xil nomli literal o'zgarmasning 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.

Amallar. C++ da quyidagi amallardan foydalanish mumkin:

1. Arifmetik amallar: +, -, /, *, %. Barcha amallar odatdagidek bajariladi. Bo'lish amalining bajarilishida natijalar toifasi unda ishtirok etayotgan kattaliklar toifasiga ko'ra aniqlanadi. Agar bo'lish amali butun sonlar ustida bajarilayotgan bo'lsa, natija butun son bo'ladi, ya'ni kasr qism tashlab yuboriladi (10/4=2). Agar natija haqiqiy (kasr) son ko'rinishida olinishi lozim bo'lsa, suratdagi yoki maxrajdagi son ketidan nuqta (.) qo'yiladi, masalan, 9./5=1.8. % belgisi (modul operatori) esa butun sonni butun songa bo'lgandan hosil bo'ladigan qoldiqni bildiradi. Masalan: 7 % 3=1.
2. Taqqoslash amallari: = = (tengmi?); != (teng emas); < ; > ; >=; <=.
3. Mantiqiy amallar: && (and) mantiqiy ko'paytirish; || (or) mantiqiy qo'shish; ! (not) mantiqiy inkor.

Mantiqiy amallarni ixtiyoriy sonlar ustida bajarish mumkin. Agar natija rost bo'lsa, natija 1ga teng bo'ladi, agar natija yolg'on bo'lsa, natija 0 ga teng bo'ladi. Umuman olganda 0 (nol)dan farqli natija rost deb qabul qilinadi. Masalan: (i>50) && (j==24) yoki (s1 < s2) && (s3>50 || s4<=20);

yoki $6 \le x \le 10$ yozuvi (x>=6) && (x<=10) deb yoziladi.

4. Qiymat berish amallari:

a) qiymat berish amali belgisi – "=" bo'lib, uni yordamida odatda ma'lum o'zgaruvchiga qiymat o'zlashtiriladi, masalan, a=5; b=2*c; x=y=z=1;

b) inkrement amali (++) ikki ma'noda ishlatiladi: o'zgaruvchiga murojaat qilinganidan keyin uning qiymati 1 ga oshadi (a++) va o'zgaruvchining qiymati unga murojaat qilishdan oldin 1 ga oshadi (++a);

c) dekrement amali (--), xuddi inkrement amali kabi bajariladi, 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 1ga kamaytirib, keyin a ga qo'shadi).

d) C++ tilida qisqartirilib yoziladigan amallar ham ishlatiladi (3.1-jadval):

3.1-jadval

Qisqartirilgan yozuv	To'liq yozuv
x += a;	x = x + a;
x -= a;	x = x - a;
x *= a;	x = x * a;
x /= a;	x = x / a;
x %= a;	x = x % a;

3.2-jadvalda C++ tilida ishlatiladigan amallar keltirilgan:

3.2-jadval

Arifmetik amallar	Razryadli amallar	Nisbat amallari	Mantiqiy amallar
+ qo'shish	& va	= = teng	&& va
- bo'lish	| yoki	!= teng emas	|| yoki
* ko'paytirish	<<chapga surish	> katta	! inkor
/ bo'lish	>>o'ngga surish	>= katta yoki teng	
% modul olish	~ inkor	< kichik	
- unar minus		<=kichik yoki teng	
+ unar plus			
++ oshirish			
-- kamaytirish			

Direktivalar. Direktivalar kompilyatsiya oldidan dasturning boshlang'ich matnini qayta ishlash uchun mo'ljallangan. Har qanday direktiva "#" belgisidan boshlanadi. Bitta qatorda faqat bitta direktiva yozilishi mumkin. Masalan, #include <myfile> dastur matniga myfile nomli e'lon qilish faylining tarkibi qo'yiladi. E'lon qilish fayli dastur kompilyatsiyaning muvaffaqiyatli bajarilishi uchun zarur bo'lgan turli axborotlarni o'zida saqlaydi.

3.2. Ma'lumot toifalari

Dasturda ma'lumotlar toifasi berilganlarning qiymatlar to'plamini va shu bilan birga ular ustida bajariladigan amallarni belgilaydi. Kompilyator buyruqlarni shakllantirish uchun ma'lumotlar xotirada qancha joy egallashini va bajariladigan amallarni aniq bilishi lozim. Bularning hammasi ma'lumotlarning toifasini tavsiflash bilan belgilanadi. Dasturda ishlatiladigan o'zgaruvchilar, o'zgarmaslar va amallarning natijalari aniq toifaga mansub bo'lishi lozim.

C++ tili toifalari oddiy (asosiy-tayanch) va tarkiblashgan (strukturaviy) turlarga bo'linadi. Oddiy toifa butun, haqiqiy, mantiqiy va simvolli toifalarni o'z ichiga oladi:

- bool (mantiqiy);
- char (simvolli);
- int (butun);
- float (haqiqiy);
- double (aniqligi ikkilangan haqiqiy toifa).

Bular asosida tarkibiy toifalar shakllanadi. Tarkibiy toifalar massivlar, strukturali (tarkiblashgan), ko'rsatkichlar va sinflarni o'z ichiga oladi.

Butun toifalar. Butun toifalar butun sonlarni tavsiflash uchun ishlatiladi. C++ tilida butun toifadagi ma'lumotlar quyidagi turlarda beriladi:

- short (qisqa);
- long (uzun);

– signed (ishorali);
– unsigned (ishorasiz).

Ishorali butun sonlar: signed int, short int, int, long int va ishorasiz butun sonlar: unsigned int, unsigned short int, unsigned long int ishlatiladi.

Unsigned manfiy butun sonlarni ifodalaydi. Dasturda short int, long int, signed int va unsigned int nomlarni mos ravishda short, long, signed va unsigned nomlar bilan almashtirish mumkin. Bundan tashqari butun sonlarga avtomat ravishda signed int toifasi beriladi.

Haqiqiy toifa. C++ standartida float, double va long double turlari aniqlangan. Ularning hammasi ishoralidir. Haqiqiy son mantissa va tartibdan tashkil topib, mantissa sonning aniqligini, tartibi uning qiymatlar diapazonini belgilaydi. Qo'zg'aluvchan nuqtali haqiqiy sonlar avtomat ravishda double toifasi bilan beriladi. Sonlar toifasini aniq ko'rsatish mumkin, buning uchun F, f (float) va L, l (long) sufikslarini ko'rsatish mumkin. Masalan, 3.14F, 2E+6L (long double toifasiga tegishli).

Mantiqiy toifa. Mantiqiy toifadagi kattaliklar true va false qiymatlarini qabul qiladilar. Ular arifmetik amallarda ishtrok etishi mumkin. Bu kattaliklarni butun toifaga o'girganda mos ravishda true-1 ga false-0 ga tenglashtiriladi.

Belgili (simvolli) toifa. C++ tilida 3 ta simvolli toifa belgilangan: char, signed char va unsigned char. Har bir simvolga xotirada faqat 1 bayt joy ajratiladi:

sizeof (char) = sizeof (signed char) = sizeof (unsigned char) = 1

Quyidagi jadvalda C++ da ishlatiladigan toifalar ro'yxati va ularning qiymat diapazonlari keltirilgan (3.3-jadval):

3.3-jadval.

Toifa	Qiymatlar diapazoni	O'lchami (bayt)
bool	true va false	1
signed char	-128 ... 127	1

unsigned char	0 ... 255	1
signed short int	-32 768 ... 32 767	2
unsigned short int	0 ... 65 535	2
signed int	-2 147 483 648 ... 2 147 483 647	4
unsigned int	0 ... 4 294 967 295	4
signed long int	-2 147 483 648 ... 2 147 483 647	4
unsigned long int	0 ... 4 294 967 295	4
float	3.4e-38 ... 3.4e+38	4
double	1.7e-308 ... 1.7e+308	8
long double	3.4e-4932 ... 3.4e+4932	10

3.3. Standart funksiyalar

Dasturlarda buyruqlar tarkibida turli ko'rinishdagi standart funksiyalardan foydalanish mumkin. Matematik funksiyalar arifmetik amallarda ishlatiladi. Quyidagi jadvalda standart matematik va boshqa funksiyalar keltirilgan (3.4-jadval):

3.4-jadval

Funksiya	Ifodalanishi	Funksiya	Ifodalanishi
sin x	sin(x)	$\sqrt{x}$	sqrt(x); pow(x,1/2.)
cos x	cos(x)	$\|x\|$	abs(x) yoki fabs(x)
tg x	tan(x)	arctan x	atan(x)
ex	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.)
xa	pow(x,a)	log2x	log(x)/log(2)
ceil	Haqiqiy sonni kattasiga yaxlitlash		

floor	Haqiqiy son kasr qismini yaxlitlash (kasrni olib tashlash)
fmod (a/b)	a/b ning qoldiq qismini hisoblash

Masalan, $\frac{-b+\sqrt{b^2-4a}}{2a}$ → (-b + sqrt(b*b-4*a*c))/(2*a); yoki (-b + pow(b*b-4*a*c,1/2))/(2*a);

$\frac{\sin x^2+\cos y^2}{\sqrt{x+y}}$ → (sin(pow(x,2)))/(pow(x+y,1/2));

$e^{\sin x} + tg^2(x+3)$ → exp(sin(x)) + pow(tan(x+3),2);

Quyida keltirilgan funksiyalar ma'lumotlar toifasini o'zgartirish (o'girish) vazifasini bajaradi (3.5-jadval):

3.5-jadval.

Funksiyaning C++ dagi ifodasi	Funksiyaning vazifasi
IntToStr(k)	Butun k sonini satrga o'tkazish
StrToInt (s)	s satrni butun songa o'tkazish
FloatToStr (n)	Haqiqiy n sonini satrga o'tkazish
StrToFloat (s)	s satrni haqiqiy songa o'tkazish

3.4. Borland C++ Builder 6 da dastur tarkibi

C++ tilida dastur preprotsessor direktivalari, global obyektlar va funksiyalarning tavsiflari va ta'riflari ketma-ketligidan tashkil topadi. Direktivalar dasturni kompilyatsiya qilingungacha boshqaradi. Ta'riflar funksiyalarni va obyektlarni kiritadi. Obyektlar dasturdagi qayta ishlanadigan ma'lumotlarni tasvirlash uchun zarur. Funksiyalar dasturdagi amallarni aniqlaydi.

Tavsiflar kompilyatorga dasturning turli qismlari yoki boshqa fayllarda aniqlangan obyekt va funksiyalarning nomi va xossalari haqida ma'lumot beradi.

Dastur bitta yoki bir nechta matnli fayl ko'rinishida shakllanadi. Matnli fayl qatorlarga ajratilgan bo'ladi. Har bir qator qator oxiri belgisi bilan tugatiladi.

Dastur qayta ishlashning uchta bosqichidan o'tadi:

– matnni preprotsessor yordamida o'girish;

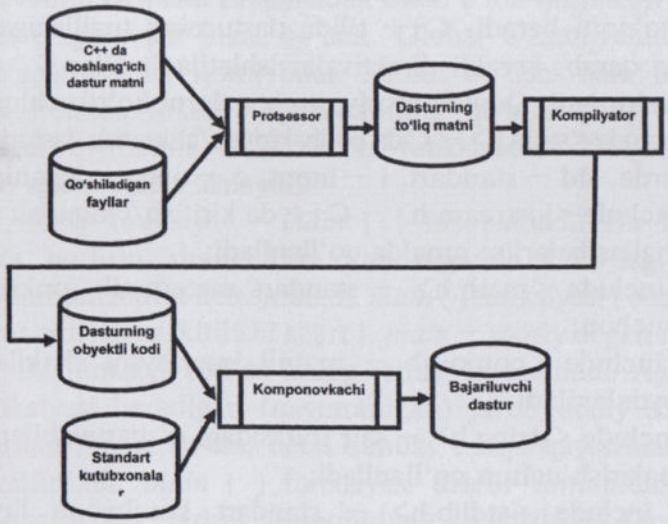
– kompilyatsiya qilish;

– kompanovkalash (barcha bog'lanishlarni tahrirlash yoki yig'ish).

Bunda shuni ta'kidlab o'tish zarurki, #include <e'lon fayli nomi> deriktivasining ishlatilishi dasturga kerakli standart kutubxonalarni qo'shmaydi, ularni qo'shish bog'lanishlarni tahrirlash – kompanovkalash bosqichida amalga oshiriladi.

Yuqorida keltirilgan bosqichlar muvaffaqiyatli tugatilgandan so'ng dasturning mashina kodi shakllanadi.

Preprotsesorning vazifasi dastur matnini uni kompilyatsiyagacha o'girishdir. Preprotsesorli qayta ishlashning tartibini dasturchi direktivalar yordamida belgilaydi. Har bir direktiva "#" belgisidan boshlanadi, masalan, #include yoki #define. #define matnda o'zgartirish qoidalarini, "#" include esa dasturda qanday matnli fayllarni qo'shish lozimligini ko'rsatadi. 3.1-rasmda dasturni qayta ishlash jarayonlari ko'rsatilgan.



3.1-rasm. *Dasturni qayta ishlash jarayonlari.*

#include direktivasi kompilyatorning standart kutubxonalari bilan birgalikda uzatiladigan e'lon fayllari katalogidan kerakli fayl matnini dastur matniga qo'shadi. Standart kutubxonalar uchun sarlavha fayllari ro'yxati C++ tili standartida belgilangan.

Global obyektlar va funksiyalarning tavsiflari va ta'riflari albatta main nomli funksiyasi orqali berilishi shart. Chunki bu funksiya dasturning asosiy funksiyasi bo'lib, usiz dastur bajarilmaydi. C++da dastur tuzish va bajarish natijasida turli ko'rinishagi ilovalar yaratiladi. Asosan 2 ta muhitda dastur tuzish va bajarish mumkin: konsol va vizual forma ilovasi muhitida.

Shunday qilib, dasturning umumiy ko'rinishi quyidagichadir:

```
Preprotsessor direktivalari
Void main( )
{
Obyektlar tavsiflari;
Bajariladigan operatorlar;
}
```

C++ tilidagi dasturning har bir bo'lagini ko'rib chiqamiz.

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. Quyida asosiy direktivalarni ko'rib chiqamiz:

– # include <stdio.h> – C 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 amalda qo'llaniladi;

– # include <math.h> – standart matematik funksiyalarni ishlatish uchun;

– # include <conio.h> – matnli interfeysni shakllantirish uchun foydalaniladi;

– # include <string.h> – satr toifasidagi o'zgaruvchilar ustida amallar bajarish uchun qo'llaniladi;

– # 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 hisoblanadi va ular alohida include deb nomlanadigan papkada saqlanadi. Bu fayllarda funksiya prototiplari, toifalari, o'zgaruvchilar, o'zgarmaslar ta'riflari yozilgan bo'ladi.

Direktivalar dasturni uni kompilyatsiya qilinishidan oldin tekshirib chiqadi.

2. Makroslar – # define makro qiymati. Masalan:

#define y sin(x+25) – y = sin(x+25) (qiymati berildi);

#define pi 3.1415 - pi = 3.1415

#define s(x) x*x - s(x) = x*x (; belgisi qo'yilmaydi)

3. 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 dasturdagi barcha obyektlar uchun ishlatiladi. O'zgaruvchini bevosita ishlatishdan oldin e'lon qilsa ham bo'ladi, u holda o'zgaruvchi 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 ichidagina qiymatini o'zgartiradi, funksiyadan chiqishi bilan global o'zgaruvchilar ishlaydi.

4. 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 iboralardan iborat bo'lsa, hech qanday natija qaytarmasa, void so'zi keltiriladi. *main* ( ) funksiyasi dastur tomonidan emas, balki operatsion tizimi tomonidan chaqiriladi. Operatsion tizimga 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 ( ) qavslari ichiga uning parametri keltiriladi. Ba'zan u bosh bo'lishi ham mumkin. Bu funksiyadan chiqish uchun odatda *return* operatori ishlatiladi. 0 (nol) qiymatining qaytarilishi amaliyot tizimiga ushbu dastur normal bajarilib tugaganini 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 parametrlari e'lon qilinadi. So'ngra dasturning asosiy operatorlari (kiritish/chiqarish, o'zlashtirish va h.k.) yoziladi. Agar bu operatorlar tarkibiy bo'lsalar, ularni alohida {} qavslarga olinadi. C++ tilida dastur kichik harflarda yoziladi. Ba'zi operatorlar katta harflar bilan kelishi mumkin, bunday hollarda ular alohida aytib o'tiladi. Operatorlar oxiriga ";" belgisi qo'yiladi. Operatorlar bir qatorga ketma-ket yozilishi mumkin. Dasturda izohlar ham kelishi mumkin, ular /* ....*/ belgisi orasiga olinadi. Agar izoh bir qatorda tugasa, uni // belgisidan keyin yoziladi. Masalan:

```
main ( ) // C++ tilining asosiy funksiyasi
cout << "1-funksiyaning qiymati=" <<f1; // f1 funksiyaning qiymati chiqadi.
```

3.5. Borland C++ Builder 6 da konsol ilovasini yaratish

Borland C++ Builder 6 da konsol ilovalarini har xil usullarda yaratish mumkin. Ulardan eng oson usuli quyidagicha:

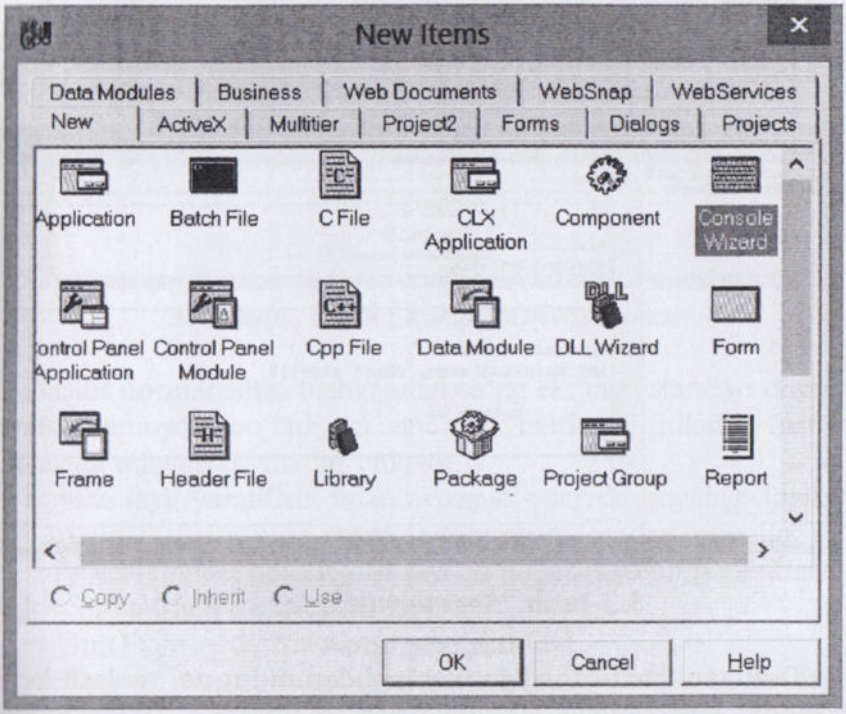
1. Borland C++ Builder 6 muhiti ishga tushiriladi:

Пуск=> Все программы=> Borland C++ Builder 6 => C++ Builder 6

2. Bosh menyuning "File" bo'limidan "New" va undan "Other" buyruqlari ketma-ketligi tanlanadi:

File=> New=> Other

3. Forma va loyihalarni saqlash uchun ochilgan maxsus oynadan "Console Wizard" piktogrammasi tanlanadi va "Ok" tugmasi bosiladi (3.2-rasm).



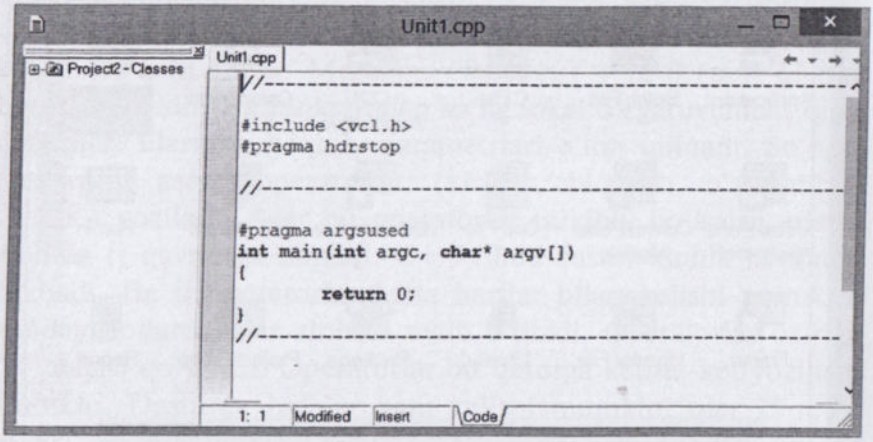
3.2-rasm. *Konsol muhitini tanlash.*

4. Natijada ekranda loyiha oynasi ochiladi (Unit.cpp kengaytmali nom bilan).

{ } qavslari ichiga loyiha faylining dastur matni kiritiladi (int main(int argc, char* argv[]) qatoridan so'ng).

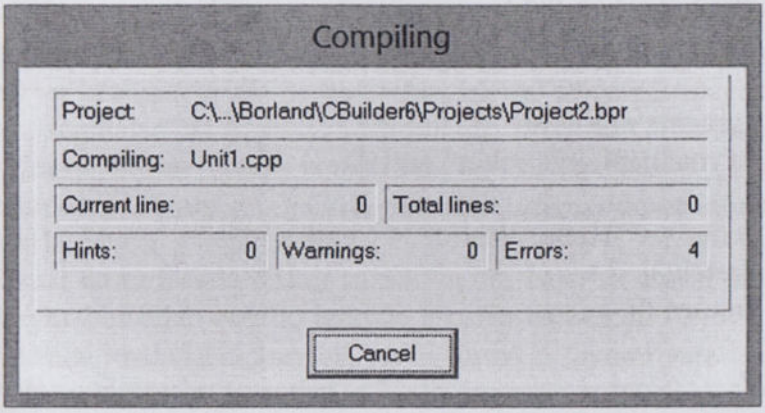
```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
```

```
int main(int argc, char* argv[])
{
//dastur matni kiritiladi
return 0;
}
//---------------------------------------------------------------
```



3.3-rasm. *Konsol muhitining ko'rinishi.*

Tuzilgan dasturni ishga tushirishdan oldin uni saqlash kerak bo'ladi. Uni saqlash uchun asosiy menyudan "File=> Save All" buyruqlar ketma-ketligini tanlash lozim. Har bir loyiha alohida yangi papkaga saqlanishi tavsiya etiladi. Loyihaga avtomat ravishda "Project1.bpr" nomi berilgan bo'lib, bu nomni dasturchi o'zgartirishi mumkin. Bu yerda "1" har bir ketma-ket nomlanadigan loyiha nomeri (son, masalan 1, 2, 3, ...). Loyihani saqlab bo'lgandan so'ng, uni bajarishga beramiz. Buning uchun asosiy menyudan quyidagi buyruqlar ketma-ketligini berish lozim: "Run=>Run" yoki klaviatura orqali [F9] tugmachasini bosish kerak bo'ladi. Shundan so'ng loyiha kompilyatsiya qilinadi, ekranda quyidagi oyna paydo bo'ladi (3.4-rasm).



3.4-rasm. *Loyiha kompilyatsiyasi oynasi.*

Dastur normal ishga tushgandan so'ng ekranda standart dastur oynasi namoyon bo'ladi va unda qiymatlar kiritiladi, dastur natijasida olingan qiymatlar chiqadi.

Loyiha fayli yaratilishi bilan avtomatik tarzda quyidagi fayllar ham tuziladi, ularni bitta papkada saqlash maqsadga muvofiq:

– Project1.bpr – bosh loyiha fayl bo'lib, asosiy componentlarni saqlovchi va ishga tushiruvchi fayl;

– Unit1.cpp – dastur matni yozilgan fayl;

– Project1.exe – ilova fayli yoki bajariluvchi fayl. Bu fayl kompilyator yordamida, ya'ni kompilyatsiya jarayonida tuziladi. [F9] tugmasini bosish bilan, bajariluvchi fayl avtomatik ravishda tuziladi. Bajariluvchi fayl avtonom fayl bo'lib uning uchun boshqa fayl yoki biror dasturiy tizim mavjud bo'lishi shart emas. Uni siz boshqa dasturlar kabi ishga tushirishingiz mumkin;

– Project1.res – loyiha resurs fayli.

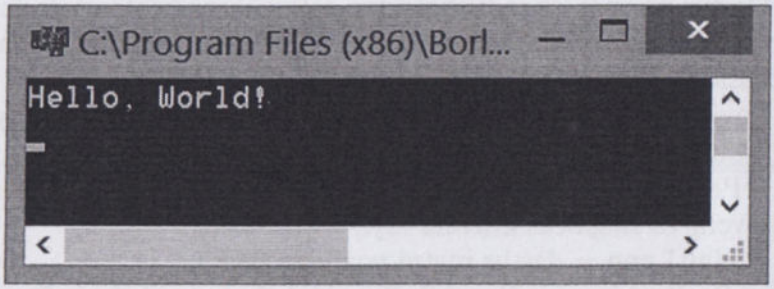
Konsol muhitida "Hello world" so'zini chiqaruvchi dasturni ko'rib chiqamiz[4]. Buning uchun yangi konsol muhitini ochamiz. Ochilgan muhitda quyidagi dastur matni kiritiladi:

```
#include <vcl.h>
#include <iostream.h>
```

[4] Alex Allain. Jumping into C++. e.book.-USA, 2014. p. 46–48.

```
#include <conio.h>
#pragma hdrstop
//---------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
  cout << "Hello, World!" << endl;
  getch();
  return 0;
}
//---------------------------------------------------------------
```

Natijada quyidagi dastur ko‘rinishi paydo bo‘ladi:



3.5-rasm. *Natija oynasi.*

3.6. Forma oynasida ilovalar yaratish

Borland C++ Builder 6da RAD (Rapid Application Development) muhiti interfeysidan foydalaniladi. Bunday interfeys dasturchiga qulaylik yaratib beradi, ya’ni ko‘pgina murakkab amallarni kompyuterning o‘zi bajaradi, dasturning asosiy loyihasini tayyorlab beradi, dasturchi bu loyihani C++ tili operatorlari bilan to‘ldiradi, tayyor vositalarni ishlatadi. Forma oynasida ilovalar yaratish usuli vizual dasturlash texnologiyasiga asoslangan bo‘lib, ilova (Forma) murakkab strukturali obyekt

ko'rinishida qabul qilingan. Forma o'zida bir qancha boshqa obyektlarni joylashtirishi mumkin. Obyekt deganda muloqat oynasi va boshqarish elementlari (kiritish va chiqarish maydoni, buyruq tugmalari va boshqa) tushuniladi. Borland C++ Builder 6 da forma ilovasi vizual komponentalar kutubxonasiga (Visual Component Library – VCL) mansub turli komponentalarni (obyektlarni) o'z ichiga oladi.

Vizual dasturlashda dastur tuzish va uni bajarish asosan ikkita o'zaro bir-biri bilan bog'liq jarayon asosida bajariladi:

– vizual loyihalash jarayoni, ya'ni forma dizaynini yaratish;

– ilovaga mos bo'lgan dastur kodlarini kiritish (yozish).

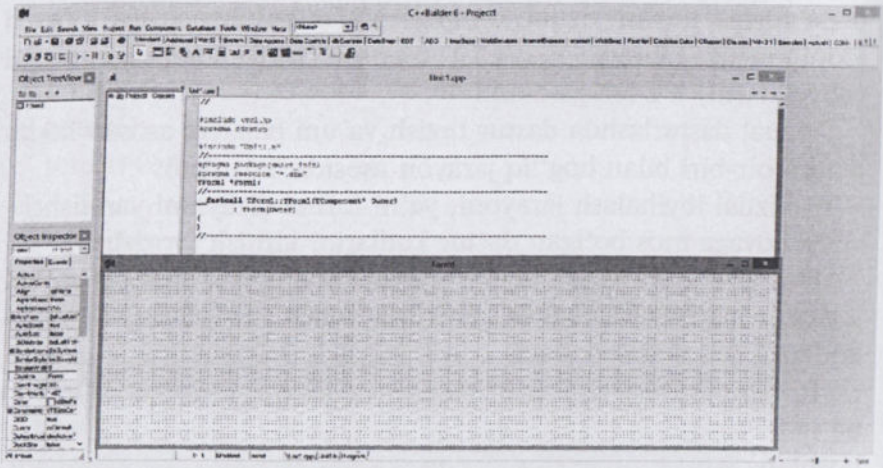
Dastur kodini yozish uchun maxsus kod oynasi mavjud bo'lib, u dastur matnini kiritish va tahrirlash uchun mo'ljallangandir. Bu kodlar C++ tilida yoziladi.

Borland C++ Builder 6 yuklanishi bilan forma oynasi paydo bo'ladi. Kodlar oynasida ushbu formani ifodalovchi dasturning boshlang'ich kodi matni tizim tomonidan yozib chiqariladi, kodlar oynasiga o'tish uchun Unit1.cpp – darchasini bosish kerak (3.6-rasm).

Dastur kodi modul ko'rinishida tuziladi. Yuqorida ta'kidlaganimizdek, modul bu dastur bo'lib, u forma ilovasidagi komponentalar toifasini, dasturda ishlatiladigan ma'lumotlarning toifasini va ularni boshqarish usullarini tavsiflaydi. Bo'sh forma modulining "Unit1.cpp" darchasining ko'rinishi quyidagicha bo'ladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
```

```
    : TForm(Owner)
{
}
//---------------------------------------------------------------------
```



3.6-rasm. *Dastur kodiga o'tish.*

Borland C++ Builder 6 da dasturlash forma oynasini shakllantirishdan boshlanadi, ya'ni formaga komponentalar o'rnatilib, uning dizayni shakllanadi. So'ngra bu formadagi komponentalarni boshqarish uchun modul kodi yoziladi. Modul kodi yozilishi bilan Borland C++ Builder 6 tizimi bu modulni boshqaruvchi asosiy loyiha dasturini "Project1.cpp" faylida yaratadi. Loyiha dasturi konsol ilovasining asosiy dasturidan birmuncha farq qiladi. Quyida "Hello world" so'zini chiqaruvchi dasturni ilova (Form1) uchun ko'rib chiqamiz. Buning uchun quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
```

```
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
fastcall Tform1::Tform1(Tcomponent* Owner)
  : Tform(Owner)
{
}
//---------------------------------------------------------
void __fastcall Tform1::Button1Click(Tobject *Sender)
{
Label1->Caption="Hello World";
}
//---------------------------------------------------------
```

Natijada quyidagi dastur ko'rinishi paydo bo'ladi:



3.7-rasm. *Natija oynasi.*

Asosiy loyiha dasturini o'zgartirish tavsiya etilmaydi, ilova obyektlari ustida bajariladigan barcha amallar asosan modulda beriladi.

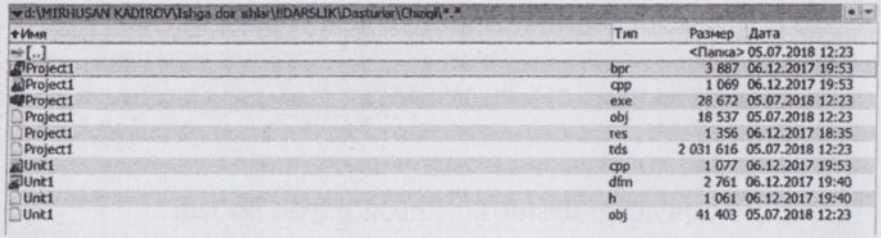
Modul kodi saqlangandan so'ng u ishga tushiriladi, ya'ni tayyorlanadigan C++ dasturi uchta asosiy bosqichdan o'tadi:

– kompilyatsiya;

– komponovka;
– bajarish.

Kompilyatsiya bosqichida tayyorlangan dastur matni mashina kodiga o'tkaziladi. Kompanovka bosqichida esa kerakli qo'shimcha-yordamchi dasturlar va qism-dasturlar unga birlashtiriladi.

Dastur kompilyatsiya qilinishi paytida *.cpp, *.dfm, *.res, *.obj, *.h, *.bpr, *.exe kengaytmali fayllar tuziladi (3.8-rasm). *.cpp kengaytmali fayl kodlarni yozish oynasida kiritilgan dastur matnini, *.dfm forma oynasi tashkil etuvchilarini saqlaydi.



3.8-rasm. *C++ Builder 6 dasturida yaratilgan fayl kengaytmalari.*

Dastur bajarilganda Forma ilovasi oynasi chiqadi va unda o'rnatilgan komponentalarni ishga tushirib kutilgan natijaga erishish mumkin. Windows operatsion tizimi boshqa ilovalar oynasiga o'xshaydi. Uni siljitish, o'lchamini o'zgartiritsh, yopish kabi amallarini bajarish mumkin. Bu esa dastur yaratishda qulayliklar yaratib beradi.

Nazorat savollari

1. C++ qanday dasturlash texnologiyalarini o'zida birlashtirgan?
2. C++ tilining asosiy tashkil etuvchilari nimalardan iborat?
3. C++ tilida o'zgaruvchilar nima?
4. C++ tilida ma'lumotlarning qanday toifalari mavjud?
5. Dastur loyihasi qanday fayllarni o'ziga birlashtiradi?

Test savollari

1. Project.exe faylining asosiy vazifasi?

a) dasturni ishga tushuruvchi fayl.

b) loyiha komponentalarini va piktogrammalarini saqlovchi fayl.

c) dastur matnini saqlovchi fayl

d) dastur matnini va komponentalarini saqlovchi fayl.

2. O'zgaruvchilar deb nimaga aytiladi?

a) a, b, x, y o'zgaruvchilar hisoblanadi.

b) dastur bajarilishi jarayonida o'z qiymatini o'zgartira olmaydigan kattaliklar.

c) dastur bajarilishi jarayonida o'z qiymatini o'zgartira oladigan kattaliklar.

d) dastur bajarilishi jarayonida o'z qiymatini o'zgartira oladigan va olmaydigan kattaliklar.

3. O'zgarmaslar qaysi operator yordamida e'lon qilinadi?

a) int

b) float

c) char

d) const

4. Toifalar to'g'ri ko'rsatilgan qatorni tanlang?

a) Bool, char, int, float

b) Long, Bool, int, float

c) Long, char, int, float

d) **Bool**, int, float, signed

5. Bool qanday toifa?

a) butun

b) mantiqiy

c) simvol

d) haqiqiy

IV bob. BORLAND C++ BUILDER 6 DASTURLASH TIZIMIDA CHIZIQLI JARAYONLARNI DASTURLASH

Tayanch so'zlar: operatorlar, o'zlashtirish operatori, qiymatlarni o'zlashtirish, ma'lumotlarni kiritish va chiqarish, konsol muhit, vizual muhit, dasturni kompilyatsiya qilish, tarkibiy operator, modul, standart oqimlar.

4.1. Operator tushunchasi. C++ tilidagi operatorlar tasnifi

Operatorlar masalani yechish algoritmida keltirilgan buyruqlarning muayyan dasturlash tili sintaksisiga mos bo'lgan ifodasidir.

C++ tilida operatorlar bajariladigan va bajarilmaydigan, sodda va tarkibiy operatorlarga bo'linadi. Bajariladigan operatorlar ma'lumotlar ustida biron-bir amallar bajaradi, bajarilmaydigan operatorlar ma'lumotlarni tavsiflash uchun ishlatiladi, masalan, int a; – butun toifali a – o'zgaruvchini tavsiflash operatori bo'lib xizmat qiladi.

Sodda operatorlar qatoriga kiritish va chiqarish, o'zlashtirish operatorlari kiradi.

Tarkibiy operator yoki blok – "{ }" figurali qavslar ichiga olingan operatorlar guruhidir.

Bundan tashqari boshqaruvchi operatorlar toifasi ham ajratilgan, ularga quyidagilar kiradi:

– tanlash operatorlari – if va switch;
– takrolanish (sikl) operatorlari – while, do while, for;
– o'tish operatorlari – break, goto, return.

4.2. O'zlashtirish operatori

Masalani yechish jarayoni qator bajariluvchi bosqichlarga bo'linib ketadi. Bu bosqichlarning har birida ma'lum qiymatlar bo'yicha yangi qiymatlar aniqlanadi (hisoblanadi). Bu aniqlangan

qiymatlarning ba'zilari natijaviy qiymatlar bo'lsa, ba'zilari esa oraliq qiymatlar bo'lib, keyingi bosqichlar uchun boshlang'ich qiymat hisoblanadi.

Yangi qiymatlarni aniqlash uchun ifoda tushunchasi xizmat qiladi, har bir ifoda bitta qiymatni aniqlash qoidasini belgilaydi.

Hisoblangan qiymatni hisoblash jarayonining keyingi bosqichida foydalanish uchun eslab qolish zarur, bunday eslab qolish hisoblangan qiymatni ma'lum o'zgaruvchiga o'zlashtirish yo'li bilan amalga oshiriladi. Bunday amalni bajarish asosiy operatorlardan biri hisoblanuvchi qiymat berish, ya'ni o'zlashtirish operatori bilan bajariladi.

O'zlashtirish operatori sintaksis jihatdan quyidagicha aniqlanadi:

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

Bu yerda 1 ta belgidan iborat bo'lgan asosiy belgi "=" "o'zlashtirish" deb o'qiladi.

O'zlashtirish operatorining bajarilishida "=" belgisining o'ng tomonidagi ifodaning qiymati hisoblanib, belgining chap tomonidagi o'zgaruvchi tomonidan o'zlashtiriladi. Operatorning o'ng tomonidagi ifoda o'rnida konstanta, qiymati aniqlangan o'zgaruvchi ham ishlatilishi mumkin.

Masalan,

y=sin(x+6.78) + cos(x+5.889); x=6; s= 'Informatika';

Operatorning chap va o'ng tomonida yozilgan kattaliklar bir toifada e'lon qilinishi zarur. O'zlashtirish operatori bajarilishi natijasida ma'lum o'zgaruvchilar keyingi bosqichlarda ishlatilishi mumkin bo'lgan yangi joriy qiymatni qabul qiladi.

Operatorning o'ng tomonidagi ifoda o'rnida yana o'zlashtirish operatori qo'llanishi mumkin, masalan, a = b = c;

O'zlashtirish operatorining o'ng va chap tomonida +=, *=, /= va h.k. amallarini ham ishlatish mumkin, masalan, a += b yoki b=a++. Bu amallarni keyingi bo'limlarda to'liqroq ko'rib chiqamiz.

4.3. Ma'lumotlarni kiritish va chiqarish

Ma'lumotlarni kiritish jarayoni bu dasturda ishlatilgan o'zgaruvchilarga boshlang'ich qiymatlar berish jarayonidir.

C++ tilida ma'lumotlarni kiritish va chiqarishning maxsus vositalari yo'q, bu jarayon standart kutubxonaga mansub funksiyalar, toifalar va obyektlar yordamida amalga oshiriladi:

iostream.h – kutubxona faylida ma'lumotlarni klaviaturadan kiritish "cin" va ekranga chiqarish "cout" oqimlari, ularga mos amallar aniqlangan:

1) << – ma'lumotlarni oqimga yozish;
2) >> – ma'lumotlarni oqimdan o'qish.

Masalan:

```
#include <iostream.h>;
. . . . . .
cout << "n o'zgaruvchining qiymati= ";
cin >> n;
```

Ma'lumotlarni chiqarish oqimi "cout<<" ko'rinishida va ma'lumotlarni kiritish "cin>>" ko'rinishida yoziladi. cout – console output va cin – console input ma'nolarini bildiradi.

Ma'lumotlarni kiritishda "cin" so'zidan keyin o'zgaruvchilarning ismlari keltiriladi. Agar ular bir nechta bo'lsa, har bir o'zgaruvchi orasiga >> belgisi qo'yiladi.

Masalan:

```
cin >> a >> b;
cin >>max >>min >> a2;
```

bu o'zgaruvchilarning son qiymatlari dastur kompilyatsiyadan o'tganidan keyin klaviatura orqali beriladi. Sonlarni bo'sh joy orqali yoki "Enter" klavishasi orqali kiritish mumkin. "cin" operatori orqali faqat son qiymatlar kiritiladi, ifodalar yozilishi mumkin emas.

Ma'lumotlarni chiqarish uchun "cout <<" operatori ishlatiladi. Bu yerda o'zgaruvchi nomlari, qo'shtirnoq ichida istalgan so'z yoki iboralar yozilishi mumkin. Agar qiymati chiqarilayotgan

o'zgaruvchilar bir nechta bo'lsa ularni alohida qatorlarda yoki ketma-ket chiqarilishi mumkin. Masalan:

```
cout << a; cout << x << y;
cout << "y=" << y;
cout << "ifodaning qiymati teng =" << fax;
```

Bir nechta natijalar chiqarilayotgan bo'lsa, ular ekranda ketma-ket ko'rinishda namoyon bo'ladilar. Bu holat esa natijalarni o'qishda noqulaylik tug'dirishi mumkin. Shuning uchun cout << oqimi oxirida endl (end line-satr oxiri) so'zi qo'yilsa, kursor keyingi qatorga o'tadi va keyingi natija ko'rinadi. Masalan:

```
cout << "max="<<max<<endl;
cout << "min="<<min<<endl;
```

Ekranda quyidagi javoblar ko'rinadi:

```
max=9
min=2
```

cout << operatori tarkibida arifmetik amallarni ham ishlatsa bo'ladi:

```
cout << "ifodaning qiymati=" << t*2+sin(t))<<endl;
```

Natijalarni sakkizlik va o'n oltilik sanoq sistemalarida ham chiqarsa bo'ladi. Buning uchun "oct" (sakkizlik), "hex" (o'n oltilik) xizmatchi so'zlari ishlatiladi. Misol:

```
cin >> a;
cout << a << endl;
cout << oct << a << endl;
cout << hex << a << endl;
```

"cout" operatori bilan yana quyidagi belgilar ham ishlatilishi mumkin (ularni ESC simvollari ham deyiladi):

\ n – yangi satr. Kursor yangi qator boshiga o'tadi;

\ t – gorizontal tabulyatsiya, kursor bir nechta harf o'ngga siljiydi;

\ v – vertikal tabulyatsiya, kursor bir nechta satr tashlab o'tadi;

\ r – qaytish, ya'ni kursor ayni satr boshiga qaytadi, yangi satrga o'tmaydi;

\ a – kompyuter dinamik ovoz chiqaradi;

\ n belgisi bilan endl so'zi orasida farq bor.

Haqiqiy sonlarni ekranga chiqarishda ularni formatlash mumkin, ya'ni nuqtadan keyin necha xonagacha aniqlikda olishni boshqarish mumkin. Buning uchun

cout. precision(n); – funksiyasi ishlatiladi, bu yerda n – aniqlik ko'rsatkichi.

Masalan: cout.precision(4);

cout << y;

Bu funksiya dasturda bir marta yozilsa kifoya qiladi.

C++ tilida simvollarni kiritish va chiqarish uchun bir qator standart funksiyalar ishlatiladi:

– getch(arg) – bitta simvol kiritilishini kutish. Kiritilayotgan simvol monitorda aks etmaydi. Bu funksiyani dastur oxirida argumentsiz ishlatilsa, monitorda ma'lumotlarni toki klavisha bosilguncha o'qish mumkin bo'ladi;

– putch(arg) – bitta simvolni standart oqimga 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 klavisha bosilguncha o'qish mumkin bo'ladi.

– putchar(arg) – bitta simvolni standart oqimga chiqarish uchun ishlatiladi. Simvol monitorda aks etadi. Bu funksiyalar iostream.h modulida joylashgandir.

Dasturda simvollarni kiritish va chiqarish quyidagicha amalga oshiriladi:

```
c=getchar();
putchar(c);
c=getch();
putchar();
getch();
```

Dasturdagi ma'lumotlarni kiritish yoki ekranga chiqarishda ularni formatlash mumkin. Buning uchun quyidagi funksiyalar ishlatiladi:

1. Scanf funksiyasi iostream.h modulida joylashgan bo'lib, umumiy ko'rinishi quyidagicha yoziladi:

scanf (control, arg1, arg2, ...)

Funksiya standart oqimdan simvollarni o'qib boshqaruvchi qator asosida formatlab mos parametrlarga yozib qo'yadi. Parametr ko'rsatkich bo'lishi lozim. Boshqaruvchi qator quyidagi o'zgartirish spetsifikatsiyalaridan iborat: bo'shlik, tabulyatsiya, keyingi qatorga o'tish simvollari.

1. Oddiy simvollar (% dan tashqari) kiritish oqimidagi navbatdagi simvollar bilan mos kelishi lozim;

2. % simvolidan boshlanuvchi spetsifikatsiya simvollari;

3. % simvolidan boshlanuvchi qiymat berishni taqiqlovchi * simvoli;

4. % simvolidan boshlanuvchi maydon maksimal uzunligini ko'rsatuvchi son.

Quyidagi spetsifikatsiya simvollarini ishlatish mumkin:

1. d – unli butun son kutilmoqda.

2. o – 0 bilan boshlangan yoki boshlanmagan sakkizlik son kutilmoqda.

3. x – 0 x belgili yoki belgisiz o'n oltilik son kutilmoqda.

4. h – o'nlik son kutilmoqda.

5. c – bitta simvol kutilmoqda.

6. s – satr kutilmoqda.

7. f – float toifasidagi son kutilmoqda. Kiritilayotgan sonning butun raqamlari va nuqtadan so'ng kasr raqamlari soni va "E" yoki "e" belgisidan so'ng mantissa raqamlari soni ko'rsatilishi mumkin.

2. Printf funksiyasi ko'rsatilgan parametrlarni standart oqimga chiqarish uchun ishlatiladi. Funksiya iostream.h modulida joylashgan bo'lib, umumiy ko'rinishi quyidagichadir:

printf (control, arg1, arg2, ...);

– control boshqaruvchi qator deb atalib, ikki turdagi simvollardan iborat bo'ladi: oddiy chiqariluvchi simvollar va navbatdagi parametrni o'zgartirib chiqaruvchi spetsifikatsiya-

lar. Har bir spetsifikatsiya % simvolidan boshlanib o'zgartirish turini ko'rsatuvchi simvol bilan tugaydi. % belgisi va o'zgartirish simvoli orasiga quyidagi simvollarni qo'yish mumkin:

– chiqarilayotgan argument chapga tekislash lozimligini ko'rsatuvchi minus belgisi – "-" ;

– maydon minimal uzunligini ko'rsatuvchi raqamlar qatori – "n";

– maydon uzunligini keyingi raqamlar qatoridan ajratuvchi nuqta – "." .

Biror qatordan qancha simvol ajratib olish lozimligini hamda float yoki double toifasidagi sonlarda nuqtadan keyin qancha kasr raqamlari bosib chiqarilishini ko'rsatuvchi raqamlar ketma-ketligi.

Chiqarilayotgan son long toifasiga tegishli ekanligini ko'rsatuvchi uzunlik markeri l.

O'zgartirish simvollari quyidagilardan iborat:

d – parametr unli butun songa aylantiriladi;

o – parametr ishorasiz va birinchi raqami 0 bo'lmagan sakkizlik songa aylantiriladi;

x – parametr ishorasiz va 0x belgisiz o'n oltilik songa aylantiriladi;

h – parametr ishorasiz o'nlik songa aylantiriladi;

c – parametr bitta simvol deb qaraladi;

s – parametr satr simvollar nolinchi simvol uchramaguncha yoki ko'rsatilgan sondagi simvollar bosiladi;

e – parametr float eki double toifasidagi son deb qaraladi va ishorali m,nnnnnnE+-xx (2.33333E+21) ko'rinishidagi o'nlik songa keltiriladi;

f – parametr float yoki double toifasidagi son deb qaraladi va ishorali m.nnnnnn ko'rinishidagi o'nlik songa keltiriladi;

g – %e yoki %f sifatida ishlatiladi.

% dan keyingi simvol o'zgartirish simvoli bo'lmasa u bosmaga chiqariladi. % simvolini o'zini bosmaga chiqarish uchun %% belgisini berish lozim.

4.4. Chiziqli jarayonlarni konsol muhitida dasturlash

Konsol muhitida ba'zi bir oddiy chiziqli jarayonlarning texnik masalalarini yechish dasturlarini ko'rib chiqamiz.

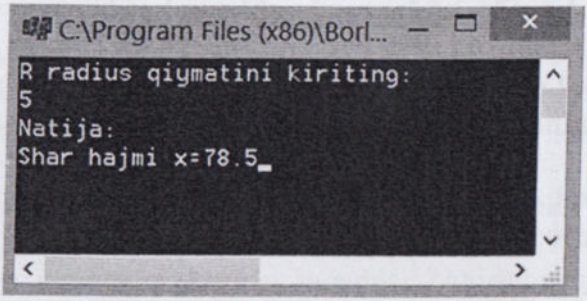
1-misol. R radiusli sharning hajmi quyidagi formula bo'yicha hisoblansin:

$$V = \frac{4}{3}\pi R^2$$

Dastur quyidagi ko'rinishda bo'ladi.

```
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
#include <math.h>
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{ float pi=3.14;
 float r,v;
cout << "R radius qiymatini kiriting:"<<"\n";
cin >>r;
v=4/3*(pi*pow(r,2));
cout << "Natija: "<<"\n";
cout <<"Shar hajmi x="<<v;
  getch();
  return 0;
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



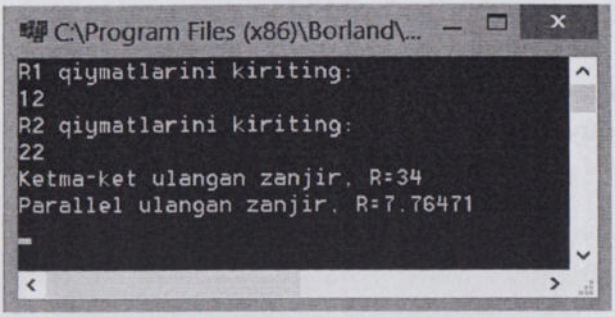
4.1-rasm. *Natija oynasi.*

2-misol. Quyidagi qarshiliklardan tashkil topgan zanjirning umumiy qarshiligini hisoblang: Ketma-ket qarshilik $R_{ket}=R_1+R_2$, parallel qarshilik $R_{par}=R_1*R_2/(R_1+R_2)$ bo'lsin. R_1 ni R_1, R_2 ni R_2, Rket ni R_{KET}, Rparni R_{PAR} deb belgilaylik. Zanjir qarshiligini hisoblash dasturini tuzamiz:

```
//---------------------------------------------------------
 #include <stdio.h>
#include<iostream.h>
#include<conio.h>
#include <math.h>
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{ float R1, R2,RKET,RPAR;
cout<<"R1 qiymatlarini kiriting: "<<"\n";
cin>>R1;
cout<<"R2 qiymatlarini kiriting: "<<"\n";
cin>>R2;
RKET=R1+R2;
RPAR=R1*R2/(R1+R2);
cout<<"Ketma-ket ulangan zanjir, R="<<RKET<<"\n";
cout<<"Parallel ulangan zanjir, R="<<RPAR<<"\n";
```

```
getch();   return 0; }
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



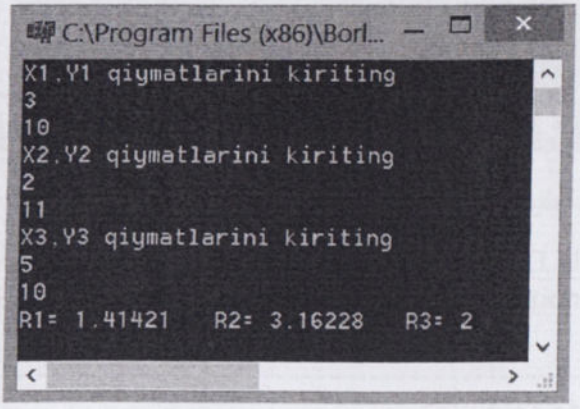
4.2-rasm. *Natija oynasi.*

3-misol. Dekart sohasida koordinatalari (x1,y1, x2,y2,x3,y3) bo'lgan 3 nuqta berilgan. Nuqtalar orasidagi R1, R2, R3 masofani toping.

```
//---------------------------------------------------------
#include <stdio.h>
#include<iostream.h>
#include<conio.h>
#include <math.h>
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
float  x1, y1, x2, y2, x3, y3, r1, r2, r3;
 cout<<"X1,Y1 qiymatlarini kiriting"<<"\n"; cin>>x1>>y1;
 cout<<"X2,Y2 qiymatlarini kiriting"<<"\n"; cin>>x2>>y2;
 cout<<"X3,Y3 qiymatlarini kiriting"<<"\n";  cin>>x3>>y3;
   r1= sqrt(pow(x2-x1,2)+pow(y2-y1,2));
```

```
    r2= sqrt(pow(x3-x2,2)+pow(y3-y2,2));
    r3= sqrt(pow(x3-x1,2)+pow(y3-y1,2));
  cout<<"R1= "<<r1<<"   R2= "<<r2<<"   R3= "<<r3;
   getch();   return 0; }
 //-------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



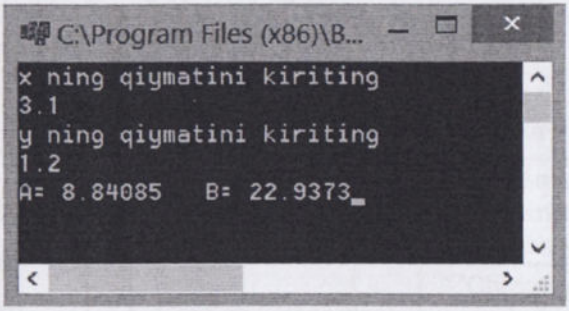
4.3-rasm. *Natija oynasi.*

4-misol. $A=\frac{\sin x^2+e^{y+1}}{\sqrt{x+y}}$ va $B=tgx^3+\ln y+|x+y|$ ifodalar berilgan. Bu yerda x=3.1, y=1.2 ga teng. Ushbu ifodalari hisoblovchi dastur tuzilsin.

```
//-------------------------------------------------------------
 #include <stdio.h>
#include<iostream.h>
#include<conio.h>
#include <math.h>
#include <vcl.h>
#pragma hdrstop
//-------------------------------------------------------------
```

```
#pragma argsused
int main(int argc, char* argv[])
{  float a,b,x,y;
   cout<<"x ning qiymatini kiriting"<<"\n"; cin>>x;
   cout<<"y ning qiymatini kiriting"<<"\n"; cin>>y;
   a=sin(pow(x,2))+exp(y+1);
   b=tan(pow(x,3))+log(y)+fabs(x+y);
   cout<<"A= "<<a<<"   B= "<<b;
  getch();
 return 0; }
//---------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



4.4-rasm. *Natija oynasi.*

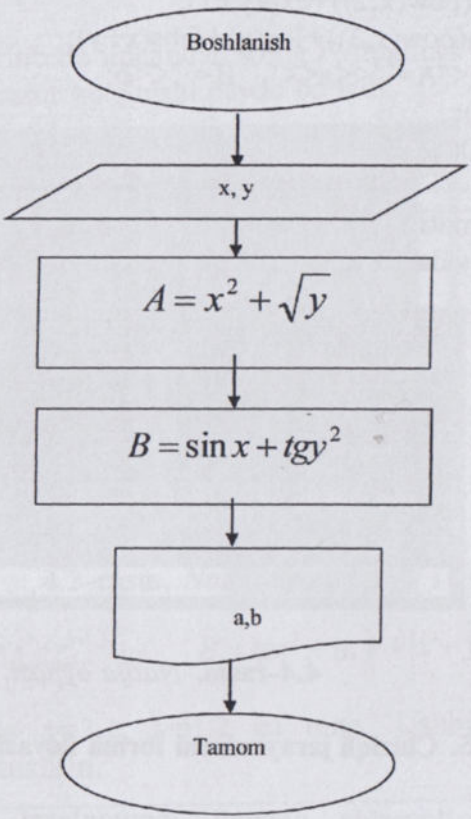
4.5. Chiziqli jarayonlarni forma ilovasida dasturlash

Forma ilovasida chiziqli jarayonlarni dasturlash vizual komponentalar yordamida amalga oshiriladi. Formaga qiymatlarni kiritish, natijani ilovaga chiqarish va qiymatlarni o'zlashtirgan o'zgaruvchilar ustida amallar bajarish maxsus komponentalar yordamida bajariladi. Vizual dasturlash muhitida bu turdagi masalalarni yechish uchun Formaga o'zgaruvchilarga qiymat kiritish – Edit komponenetasi, matn yozish uchun – Label, dasturni

ishga tushirish uchun Button yoki BitBit komponentalaridan foydalaniladi.

1-misol. $A = x^2 + \sqrt{y}$ va $B = \sin x + tgy^2$ ifodalar berilgan. Bu yerda x=2.1, y=4.2 ga teng. Ushbu ifodalari hisoblovchi dastur tuzilsin.

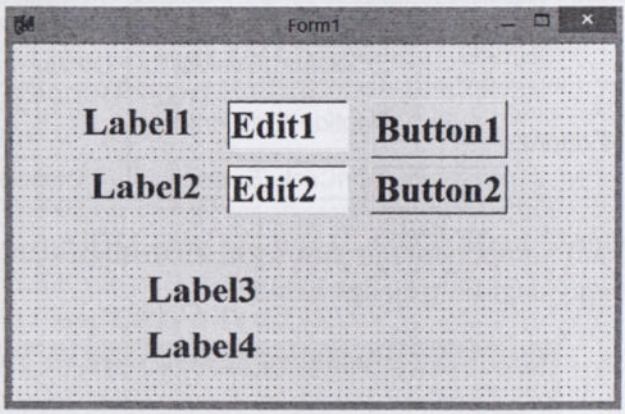
Bu misolni yechish algoritmi quyidagi ko'rinishda bo'ladi:



4.5-rasm. *Dastur blok sxemasi.*

Misolni vizual muhitda dasturlash uchun 4 ta Label, 2 ta Edit va 2 Button komponentalari kerak bo'ladi.

Forma darchasiga kerakli komponentalar o'rnatilgandan so'ng dasur ko'rinishi quyidagi holatga keladi:



4.6-rasm. *Dastur ko'rinishi.*

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

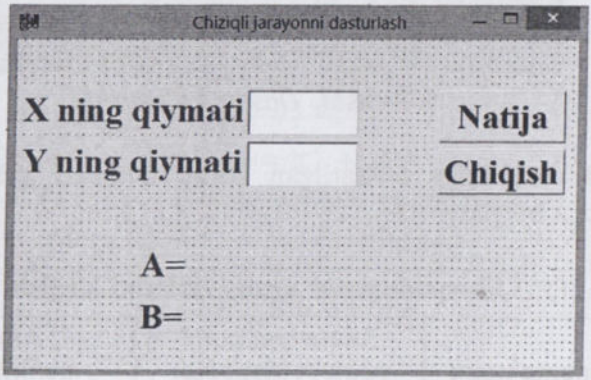
4.1-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Chiziqli jarayonni dasturlash" so'zi kiritiladi.
Label1	Caption (Properties)	"X ning qiymati" so'zi kiritiladi.
Label2	Caption (Properties)	"Y ning qiymati" so'zi kiritiladi.
Label3	Caption (Properties)	"A=" so'zi kiritiladi.
Label4	Caption (Properties)	"B=" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.
Edit2	Text (Properties)	"Edit2" so'zini o'chirib tashlang.

Button1	Caption (Properties)	"Natija" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
Button2	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



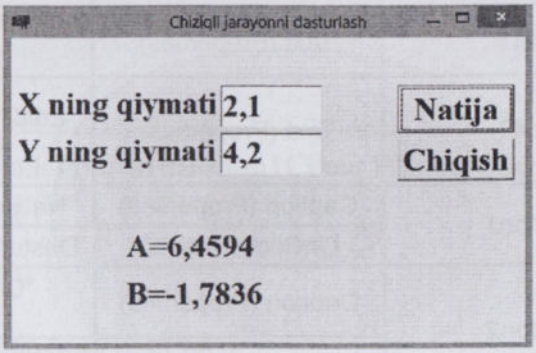
4.7-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
 #include <math.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
//---------------------------------------------------------
fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{ float  a,b,x,y;
 x=StrToFloat(Edit1->Text);
 y=StrToFloat(Edit2->Text);
 a=pow(x,2)+sqrt(y);
  b=sin(x)+tan(pow(y,2));
  Label3->Caption= "A="+FloatToStrF(a,ffFixed,6,4);
   Label4->Caption= "B="+FloatToStrF(b,ffFixed,6,4);
}
//---------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



4.8-rasm. *Natija oynasi.*

2-misol. Yuqorida keltirilgan qarshiliklardan tashkil topgan zanjirning umumiy qarshiligini hisoblash dasturini vizual muhitda bajarishni ko'rib chiqamiz. Zanjir qarshiligini hisoblash dasturini tuzamiz.

Misolni vizual muhitda dasturlash uchun 4 ta Label, 2 ta Edit va 2 Button komponentalari kerak bo'ladi.

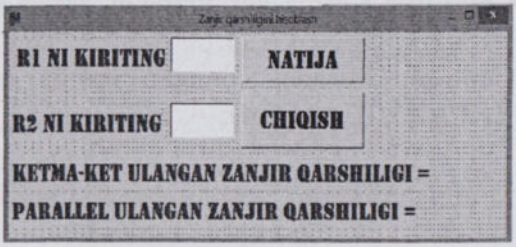
Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

4.2-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Zanjir qarshiligini hisoblash" so'zi kiritiladi.
Label1	Caption (Properties)	"R1 ni kiriting" so'zi kiritiladi.
Label2	Caption (Properties)	"R1 ni kiriting" so'zi kiritiladi.
Label3	Caption (Properties)	"Ketma-ket ulangan zanjir qarshiligi =" so'zi kiritiladi.
Label4	Caption (Properties)	"Parallel ulangan zanjir qarshiligi =" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.
Edit2	Text (Properties)	"Edit2" so'zini o'chirib tashlang.
Button1	Caption (Properties)	"Natija" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
Button2	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so‘ng dastur dizayni quyidagi ko‘rinishga keladi:



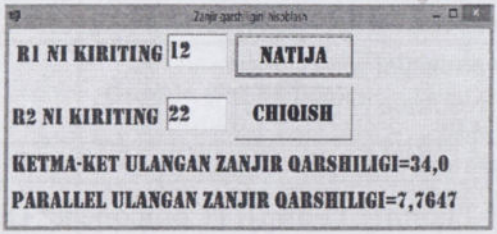
4.9-rasm. *Dastur ko‘rinishi.*

Dastur dizayni tayyor bo’lganidanso‘ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
float r1, r2;
float RKET;
float RPAR;
r1=StrToFloat(Edit1->Text);
r2= StrToFloat(Edit2->Text);
```

```
    RKET=r1+r2;
    RPAR=r1*r2/(r1+r2);
    Label3->Caption="Ketma-ket ulangan zanjir qarshiligi=" +
FloatToStrF (RKET, ffFixed,6,1);
    Label4->Caption="Parallel ulangan zanjir qarshiligi=" +
FloatToStrF (RPAR, ffFixed,6,4);
    }
    //---------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
    {
    Close();
    }
    //---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



4.10-rasm. *Natija oynasi.*

Nazorat savollari

1. "Chiziqli jarayon" deb nimaga aytiladi?
2. Kattaliklarni kiritishni tashkil qilish.
3. Kattaliklarni bosmaga chiqarishni tashkil qilish.
4. Borland C++ Builder 6 ning konsol ilovasida dastur tuzish jarayonini tavsiflab bering.
5. Borland C++ Builder 6 ning Forma ilovasida qaysi komponentalardan foydalaniladi?

Test savollari

1. O‘zgaruvchilar to‘g‘ri e’lon qilingan qatorni ko‘rsating.

a) Xx,y,a,b float;
b) x,y,a,b int;
c) integer x,y,a,b;
d) int x,y,a,b;

2. To‘g‘ri e’lon qilingan qatorni ko‘rsating.

a) a= StrToFloat(Edit1->Caption);
b) a:= StrToFloat(Edit1.Text);
c) a= StrToFloat(Edit1->Text);
d) b= strtofloat(Edit1->Text);

3. To‘g‘ri e’lon qilingan qatorni ko‘rsating.

a) Label5.Caption: = "Y="+FloatToStr(y);
b) Label1->Caption = "Y="+FloatToStr(y);
c) Label5->Text = "Y="+FloatToStr(y);
d) Label5->Add = "Y="+FloatToStr(y);

4. Vizual muhitda natijani ekranga chiqarish uchun qaysi komponentadan foydalaniladi?

a) Label
b) MainMenu
c) Button
d) BitBtn

5. Standart matematik funksiyalardan foydalanish uchun qaysi kutubxonadan foydalaniladi?

a) # include <math.h>
b) # include <vcl.h>
c) # include <fstream.h>
d) # include < conio.h>

V bob. C++ BUILDER 6 DASTURLASH TIZIMIDA TARMOQLANUVCHI JARAYONLARNI DASTURLASH

Tayanch so'zlar: tarmoqlanuvchi jarayon, shartli o'tish operatori, tanlash operatori, shartsiz o'tish operatori, mantiqiy ifodalar, operator belgilari, oddiy va murakkab tarmoqlanish jarayonlari.

Ko'pgina masalalarni yechishda ba'zi bir jarayonlar ma'lum shart yoki shartlarning qo'yilishiga nisbatan bajariladi, ya'ni shartning bajarilishi yoki bajarilmasligiga ko'ra boshqa jarayonlar, amallar tanlanadi. Bunday jarayonlar "Tarmoqlanuvchi jarayonlar" deb yuritiladi.

Tarmoqlanuvchi hisoblash jarayonlari oddiy va murakkab bo'lishi mumkin. Bu esa jarayondagi tarmoqlar soniga bog'liq. Ma'lum bir tarmoqlanuvchi jarayon tarkibida yana tarmoqlanishlar bo'lishi mumkin. Bunday tarmoqlanishlari bor bo'lgan hisoblash jarayonlari "Murakkab tarmoqlanuvchi hisoblash jarayonlari" deb ataladi.

C++ tilida tarmoqlanuvchi jarayonlarni dasturlash uchun shartsiz, shartli o'tish va tanlash operatorlaridan foydalaniladi.

5.1. Shartsiz o'tish operatori

Dasturda ba'zi bir hollarda boshqaruvni to'g'ridan to'g'ri biron-bir operatorga uzatishga, ya'ni dasturning bajarilish ketma-ketligini buzishga to'g'ri keladi. Bu jarayon shartsiz o'tish operatori yordamida bajariladi.

Shartsiz o'tish operatorining umumiy ko'rinishi quyidagicha:

goto <operator belgisi>;

Bu yerda operator belgisi boshqaruv uzatiladigan operator belgisidir. Belgi sifatida 0-9999 oraliqdagi natural sonlar, CHAR toifasidagi belgilar (simvollar) va ular aralashmasidan foydalanish mumkin.

Masalan:

```
int matrix [n][m];
int value;
...
for (int i=0; i<n; i++)
for (int j=0; j<m; j++)
if (matrix [i][j]==value)
{pintf ("value %d found in cell (%d, %d)\n", value,i,j);
goto end_loop;
}
printf("value %d not found \n", value);
end_loop: ;
```

Belgili operatorda belgi bilan operator o'rtasida « : » belgisi qo'yiladi. Bu operatorning noto'g'ri qo'llanilishi dasturning bajarilishiga xalaqit beradi. Shuning uchun, dasturda bu operatorning kamroq uchrashi maqsadga muvofiqdir.

5.2. Shartli o'tish operatori

Dasturda boshqaruvni ma'lum shart asosida u yoki bu tarmoqqa (ma'lum jarayonlar ketma-ketligi) uzatish shartli o'tish operatori yordamida amalga oshiriladi. Shartli o'tish operatori ikki xil: to'liq va qisqa ko'rinishda ishlatilishi mumkin.

Shartli o'tish operatorining to'liq ko'rinishini ko'rib chiqamiz. Uning dasturda ko'rinishi quyidagicha yoziladi:

if (<mantiqiy ifoda>) <operator −1>; else <operator−2>;

Bu yerda: if − agar, else− aks holda ma'nosini anglatuvchi xizmatchi so'zlar, operator −1 va operator −2 ixtiyoriy operatorlar.

Operatordagi mantiqiy ifoda boshqaruvni uzatish shartini belgilaydi.

Operatorning ishlash tartibi quyidagicha: agar keltirilgan mantiqiy ifoda true (rost) qiymatni qabul qilsa, ya'ni qo'yilgan shart bajarilsa; operator − 1 bajariladi, aks holda else xizmatchi so'zdan keyingi operator − 2 bajariladi.

Mantiqiy ifoda o'rnida munosabat amallari, mantiqiy amallar ishlatilishi mumkin. Masalan,

a>b, a=b, x<4. 55, 2+z>0, x+y<=1 va h. k.

Shartlar oddiy va murakkab bo'lishi mumkin.

Agar mantiqiy ifodada bitta munosabat amali berilgan bo'lsa, "oddiy shart"ni ifodalaydi.

Kattaliklar orasidagi shartlar "hamda", "yoki", "emas" (C++ tilida &&, ||, !) mantiqiy amallari belgilari orqali bog'lanuvchi bir necha munosabatlardan iborat bo'lsa "Murakkab shartlar" deb ataladi.

Munosabat amal belglari 5.1-jadvalda ko'rsatilgan:

5.1-jadval.

Munosabat amal belglari

Munosabat amal belgisi	Nomlanishi	Misolda ko'rinishi
==	Teng	2=2;
!=	Teng emas	5!=4;
<	Kichik	2<3;
>	Katta	5>4;
>=	Katta yoki teng	1<=Z;
<=	Kichik yoki teng	1>=Z;

Masalan, dasturda murakkab mantiqiy ifodalar (shartlar) quyidagicha yoziladi:

– 1<x<=4 mantiqiy ifoda (x>1 && x<=4) ko'rinishida yoziladi.

– a=b=0 mantiqiy ifoda (a=0 && b=0) ko'rinishida yoziladi;

– 6<=x<10 mantiqiy ifoda (x>=6 && x<10) ko'rinishida yoziladi.

Shartli o'tish operatorining ishlatilishini misollarda ko'rib chiqamiz.

1) if (u>0) d=sqrt(y); else d=u;

Shartli operatorda u>0 bo'lsa d=sqrt(y) operatori, aks holda d=u operatori bajariladi.

2) if (x=0 && x>0) x=sqrt (x); else x=pow(x,2);

Ushbu operatorning bajarilishi natijasida x ning qiymati 0 ga teng va musbat bo'lsa, uning qiymati ildiz ostidan chiqariladi, aks holda kvadratga oshiriladi.

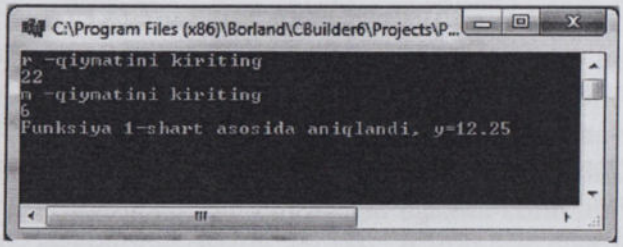
1-misol. y funksiyaning qiymatini aniqlash dasturi tuzilsin. Konsol muhitida amalga oshirilsin.

$$y=\begin{cases}\dfrac{4r+3m^2}{r-m}, & agar \quad r \geq m+1\\ |r-m|, & agar \quad r < m+1\end{cases}$$

Dastur matni quyidagi ko'rinishda bo'ladi:

```
//---------------------------------------------------------
 #include <stdio.h>
#include<iostream.h>
#include<conio.h>
#include <math.h>
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{  float  y, r, m;
cout<<"r -qiymatini kiriting"<<"\n"; cin>>r;
cout<<"m -qiymatini kiriting"<<"\n"; cin>>m;
    if (r>=m+1)
        {y=(4*r+3*pow(m,2))/(r-m);
        cout<<"Funksiya 1-shart asosida aniqlandi, y="<<y;}
else
{y=fabs(r-m);
     cout<<"Funksiya 2-shart asosida aniqlandi, y="<<y;
   }    getch();
 return 0;
 }
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:

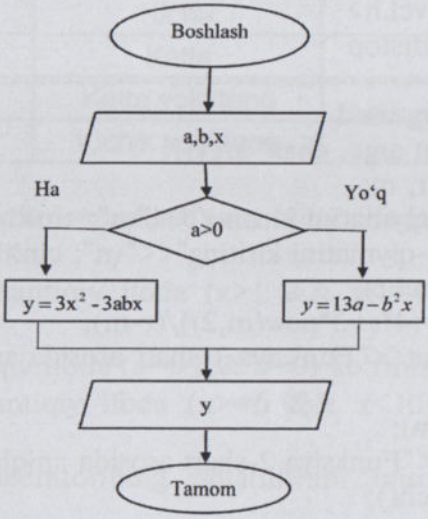


5.1-rasm. *Natija oynasi.*

2-misol. y funksiyaning qiymatini aniqlash dasturi tuzilsin. Vizual muhitida amalga oshirilsin:

$$y=\begin{cases} 3x^2-3abx & agar \quad a>0 \\ 13a-b^2x & agar \quad a\le 0 \end{cases}$$

Bu misolni yechish algoritmi quyidagi ko'rinishda bo'ladi:



5.2-rasm. *Dastur blok shemasi.*

Misolni vizual muhitda dasturlash uchun 4 ta Label, 2 ta Edit, 2 BitBtn va RadioGroup komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

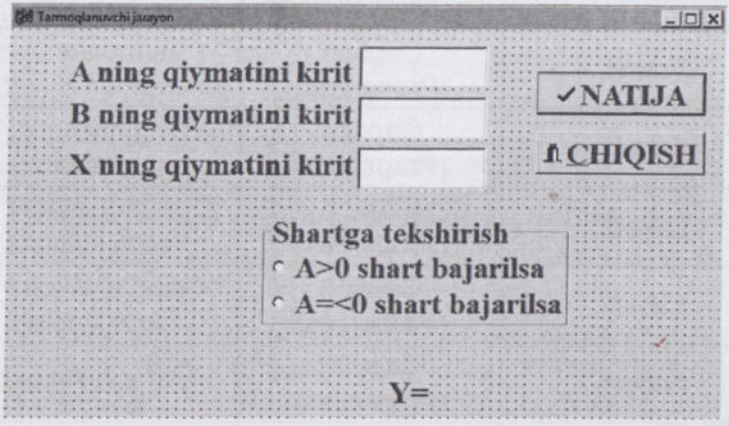
5.2-jadval.

Komponentalar xususiyatlarini kiritish

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Tarmoqlanuvchi jarayon" so'zi kiritiladi.
Label1	Caption (Properties)	"A ning qiymatini kirit" so'zi kiritiladi.
Label2	Caption (Properties)	"B ning qiymatini kirit" so'zi kiritiladi.
Label3	Caption (Properties)	"X ning qiymatini kirit" so'zi kiritiladi.
Label4	Caption (Properties)	"Y=" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.
Edit2	Text (Properties)	"Edit2" so'zini o'chirib tashlang.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Natija" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

BitBtn2	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.
RadioGroup1	Caption (Properties)	"Shartga tekshirish" so'zi kiritiladi.
	Items (Properties)	"A>0 shart bajarilsa va A=<0 shart bajarilsa" so'zi kiritiladi



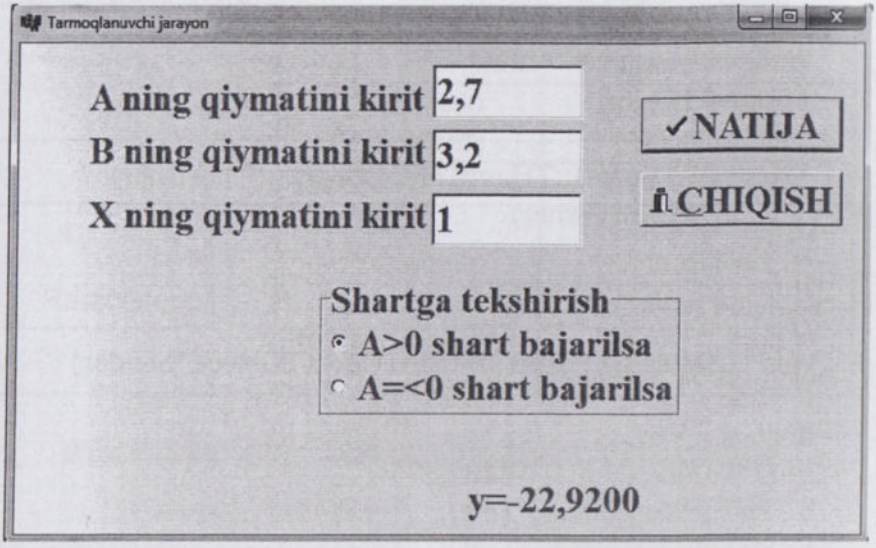
5.3-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
 #include <math.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
```

```
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
float  y,x,a,b;
a=StrToFloat(Edit1->Text);
b=StrToFloat(Edit2->Text);
x=StrToFloat(Edit3->Text);
if (a>0) {
    y= 3*pow(x,2)-3*a*b*x;
    RadioGroup1->ItemIndex=0;
 Label4->Caption ="y="+FloatToStrF(y,ffFixed,6,4);
        }
    else  {
    y=13*a-pow(b,2)*x;
    RadioGroup1->ItemIndex=1;
 Label4->Caption ="y="+FloatToStrF(y,ffFixed,6,4);
}
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------
```

Dastur matni kiritib bo‘lingandan so‘ng F9 tugmasi bosiladi va natijada quyidagi dastur ko‘rinishi paydo bo‘ladi:



5.4-rasm. *Natija oynasi.*

Ayrim algoritmlarda ba'zan shunday hol uchrashi mumkinki, bunda hisoblash jarayonida ayrim amallar ba'zi bir shartlar bajarilgandagina hisoblanadi. Aks holda, hech qanday amal bajarilmaydi. Bu holda shartli o'tish operatorini qisqa ko'rinishda ifodalash mumkin. Uning yozilishi quyidagicha:

if (<mantiqiy ifoda>) <operator>;

Operatorning bajarilish tartibi quyidagicha: agar mantiqiy ifoda TRUE (rost) qiymat qabul qilsa, operator bajariladi, aks holda IF dan keyingi turgan operator bajariladi.

Misol:

if (x<0) t=x*x;

Shartli o'tish operatorining metaformulasidagi operator o'rnida o'z navbatida yana shartli o'tish operatorining to'la va qisqa ko'rinishlari ishlatilishi mumkin. Masalan,

1) if (b1) {if (b2) a};

bu yerda: b1, b2 – mantiqiy ifoda, a – operator.

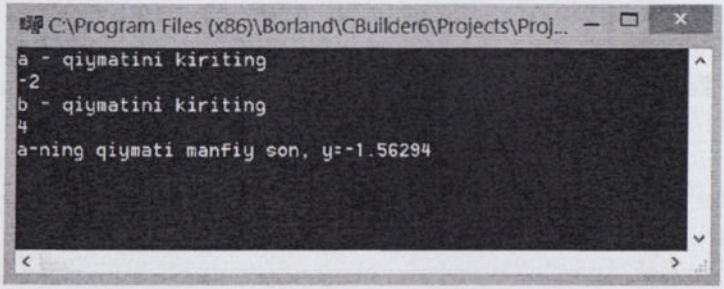
Bu operatorning bajarilishi natijasida b1 mantiqiy ifodaning qiymati hisoblanadi, agar TRUE qiymat qabul qilsa b2 mantiqiy ifodaning qiymati hisoblanadi, u ham rost (TRUE) bo'lsa a – operator bajariladi. Agar mantiqiy ifodalar b1 yoki b2 yolg'on bo'lsa (FALSE) shartli o'tish operatoridan keyingi operator bajariladi.

3-misol. Agar "a" o'zgaruvchi manfiy son bo'lsa $y = \sin(a) + \cos(b)$ funksiyaning qiymatini aniqlash dasturi tuzilsin. Konsol muhitida amalga oshirilsin.

Dastur matni quyidagi ko'rinishda bo'ladi:

```
//---------------------------------------------------------
 #include <stdio.h>
#include<iostream.h>
#include<conio.h>
#include <math.h>
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{  float  y, a, b;
cout<<"a – qiymatini kiriting"<<"\n"; cin>>a;
cout<<"b – qiymatini kiriting"<<"\n"; cin>>b;
   if (a<0)
        {y=sin(a)+cos(b);
        cout<<" a-ning qiymati manfiy son, y="<<y;}
    getch();
         return 0; }
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



5.5-rasm. *Natija oynasi.*

4-misol. Agar "a" o'zgaruvchi musbat son bo'lsa $y = a^2 + b^2$ funksiyaning qiymatini aniqlash dasturi tuzilsin. Dastur vizual muhitda amalga oshirilsin.

Misolni vizual muhitda dasturlash uchun 3 ta Label, 2 ta Edit, 2 BitBtn komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

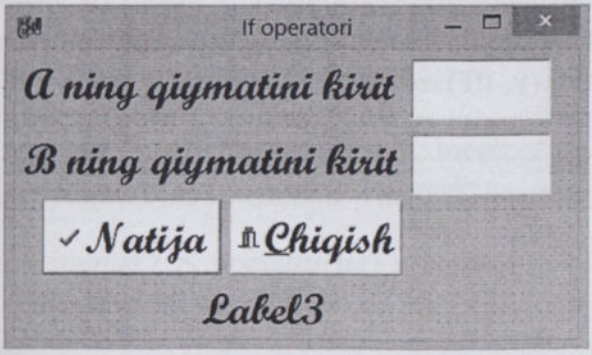
5.3-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"if operatori" so'zi kiritiladi.
	Color (Properties)	"clAppWorkSpace" tanlanadi.
Label1	Caption (Properties)	"A ning qiymatini kirit" so'zi kiritiladi.
Label2	Caption (Properties)	"B ning qiymatini kirit" so'zi kiritiladi.
Label3	Caption (Properties)	"Label3" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.
Edit2	Text (Properties)	"Edit2" so'zini o'chirib tashlang.

BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Natija" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



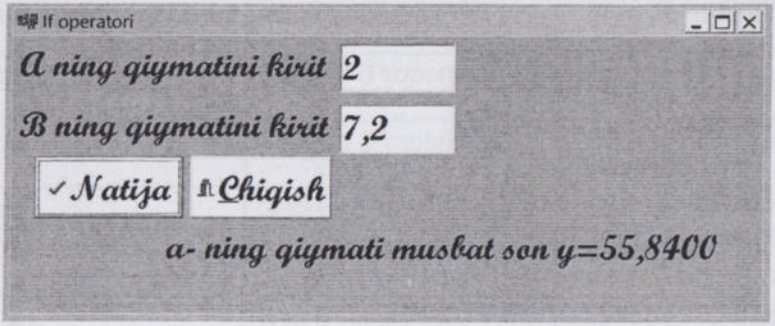
5.6-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
 #include <math.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
```

```
TForm1 *Form1;
//---------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{ float  y,x,a,b;
a=StrToFloat(Edit1->Text);
b=StrToFloat(Edit2->Text);
if (a<0)
{ y= y= pow(a,2)+pow(b,2);
Label3->Caption ="a – ning qiymati musbat son y="+
FloatToStrF (y, ffFixed, 6,4);
}}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



5.7-rasm. *Natija oynasi.*

5.3. Tanlash operatori

Juda ko'p tarmoqlanish jarayonlarida tarmoqlanish 2 ta yoki undan ortiq tarmoqqa ajraladi. Umuman olganda buni bizga tanish shartli o'tish operatori yordamida amalga oshirish mumkin:

```
if (b1) a1; else
                if (b2) a2; else
                                .. .. .. .. .. .. .. .. .. .. ..
                                        if (bk) ak;
```

Lekin bu hollarda shartli o'tish operatorlarining yozilishi noqulay.

Ko'p hollarda dasturchi uchun shartli operatorning umumiylashgan ko'rinishi – tanlash operatorini ishlatish qulay. Tanlash operatorining umumiy ko'rinishi quyidagicha yoziladi:

```
switch (ifoda yoki o'zgaruvchi – selektor)
{
case <1–qiymat >: <1– operator(lar)>; break;
case < 2–qiymat>: < operator(lar)>; break;
...
case < n – qiymat> :< operator(lar)>; break;
default:  <aks holdagi operator(lar)>;
}
```

Bu yerda, switch (tanlash yoki boshqa holatga o'tkazuvchi) – xizmatchi so'z.

Switch – operatori tarmoqlanish jarayonini berilgan bir nechta operatordan birini tanlash yo'li bilan amalga oshiradi. Tanlash operatorida barcha operatorlar, shu jumladan bajarilishi uchun tanlangan operator ham aniq ravishda keltiriladi (berilgan operatorlar ketma-ketligi chegaralangan). Bajarilishi kerak bo'lgan operator yoki operatorlar ketma-ketligi operator selektorining qiymatiga ko'ra aniqlanadi. Agar biror variant mos kelmasa default orqali ko'rsatilgan operator bajariladi. Break operatori har bir holatdan chiqish, orqaga qaytish uchun ishlatiladi, uning o'rnida return operatori ham qo'llanishi mumkin.

Operator selektori sifatida haqiqiy bo'lmagan, skalyar ko'rinishdagi har qanday ifoda yoki o'zgaruvchi ishlatilishi mumkin. Operatorning ishlashida uning tarkibidagi har bir operator "tanlash belgisi" deb ataluvchi belgi bilan ta'minlanadi. Bu belgi operatorning bajarilishi uchun zarur bo'lgan selektorning maxsus qiymatini qabul qiladigan selektorning tavsifiga mos konstantadir. Operator bir nechta mavjud qiymatlar bilan ishlashi uchun unda tanlash belgilari ro'yxati keltirilishi kerak.

Operator bajarilishida dastlab selektorning qiymati hisoblanadi. So'ngra selektorning qiymatiga mos belgili operator – "case" so'zidan keyin turgan qiymatga mos kelgan operator bajariladi. Agar operatorlar ketma-ketligida bunday belgili operator topilmasa, dasturda xato qayd etiladi. Shuning uchun dastur bajarilishi jarayonida selektorning qiymatiga mos keladigan maxsus belgili operator yoki operatorlar ketma-ketligida bo'lishi shart. Bunda tanlash operatorida beriladigan belgilar e'lon qilinmaydi.

Tanlash operatorining bajarilishi uning tarkibidagi operatorlar ketma-ketligidagi bitta operatorning bajarilishiga olib keladi.

Masalan:

```
#include <iostream.h>
#include<conio.h>
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
int baho;
cin>> baho;
switch(baho)
{case 2: cout <<"\n qoniqarsiz"; break;
 case 3: cout <<"\n  qoniqarli"; break;
 case 4: cout <<"\n  yahshi" ;break;
 case 5: cout <<"\n  a'lo"; break;
```

```
default: cout <<"\n baho noto'gri kiritilgan";
}       getch();
        return 0;
}
//---------------------------------------------------------
```

Bu operatorning bajarilishi natijasida, agar "baho" – o'zgaruvchining qiymati kiritilgandan so'ng, uning qiymati case variantida keltirilgan qiymatlar bilan solishtiriladi. Agar qiymat 2 ga teng bo'lsa "Qoniqarsiz", 3 ga teng bo'lsa "Qoniqarli" va h.k. so'zlari ekranga chiqariladi, aks holda "baho noto'g'ri kiritilgan" qator chiqadi.

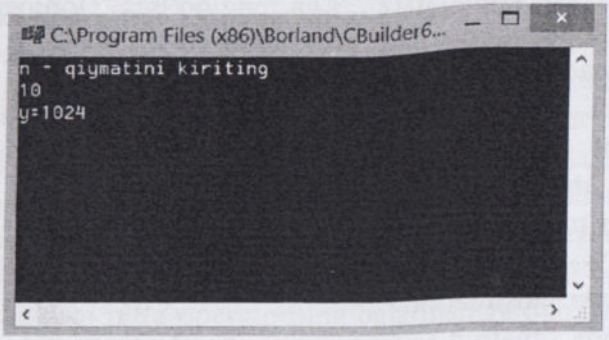
5-misol. Klaviaturadan kiritilgan songa mos ravishda $y = 2^n$ funksiyani darajaga oshirish dasturi ishlab chiqilsin. Konsol muhitida amalga oshirilsin.

Dastur matni quyidagi ko'rinishda bo'ladi:

```
//---------------------------------------------------------
#include <iostream.h>
#include <math.h>
#include<conio.h>
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{int n,y;
cout<<"n – qiymatini kiriting"<<"\n"; cin>> n;
switch(n)
{case 2: y=pow (2,2); cout <<"y="<<y; break;
 case 3: y=pow (2,3); cout <<"y="<<y; break;
 case 4: y=pow (2,4); cout <<"y="<<y; break;
 case 5: y=pow (2,5); cout <<"y="<<y; break;
 case 6: y=pow (2,6); cout <<"y="<<y; break;
 case 7: y=pow (2,7); cout <<"y="<<y; break;
```

```
case 8: y=pow (2,8); cout <<"y="<<y; break;
case 9: y=pow (2,9); cout <<"y="<<y; break;
case 10: y=pow (2,10); cout <<"y="<<y; break;
default: cout <<"n noto'g'ri kiritilgan";
} getch();
    return 0;}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



5.8-rasm. *Natija oynasi.*

6-misol. Kiritilgan sonning juft yoki toqligini ko'rsatuvchi dastur tuzilsin. Dastur vizual muhitda amalga oshirilsin.

Misolni vizual muhitda dasturlash uchun 3 ta Label, 1 ta Edit, 2 BitBtn komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

5.4-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Tanlash operatori" so'zi kiritiladi.

Label1	Caption (Properties)	"Son kiriting" so'zi kiritiladi.
Label2	Caption (Properties)	"Kiritilgan son:" so'zi kiritiladi.
Label3	Caption (Properties)	"Label3" so'zini o'chirib tashlang.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Natija" so'zi kiritiladi.
	BitBtn2	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



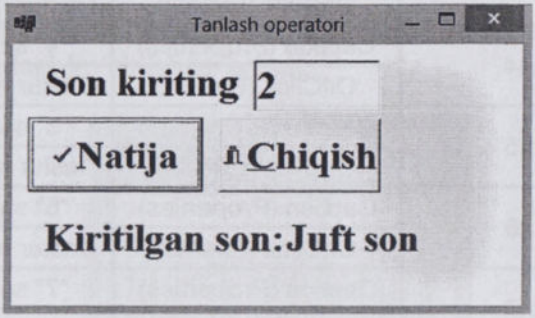
5.9-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{int n;
n=StrToFloat(Edit1->Text);
switch (n)
{
case 1:
case 3:
case 5:
case 7:
case 9: Label3->Caption=("Toq son"); break;
case 2:
case 4:
case 6:
case 8:
case 10: Label3->Caption=("Juft son"); break;
default: Label3->Caption=("1-10 oraliqdagi sonlarni kiriting!");
}
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
```

```
{
Close();
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



5.10-rasm. *Natija oynasi.*

7-misol. Kalkulyator dasturi tuzilsin. Dastur vizual muhitda amalga oshirilsin.

Misolni vizual muhitda dasturlash uchun 1 ta Edit, 16 ta BitBtn komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

5.5-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Calc" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.

BitBtn1	Caption (Properties)	"1" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Caption (Properties)	"2" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Caption (Properties)	"3" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn4	Caption (Properties)	"4" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn5	Caption (Properties)	"5" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn6	Caption (Properties)	"6" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn7	Caption (Properties)	"7" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn8	Caption (Properties)	"8" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn9	Caption (Properties)	"9" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn10	Caption (Properties)	"0" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn11	Caption (Properties)	"+" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn12	Caption (Properties)	"-" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn13	Caption (Properties)	"*" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn14	Caption (Properties)	"/" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

BitBtn15	Caption (Properties)	"C" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn16	Caption (Properties)	"=" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



5.11-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#include <math.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
```

```
#pragma resource "*.dfm"
TForm1 *Form1;
float a,b,c;
char k;
//---------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{ Edit1->Text=Edit1->Text+"1";
b=StrToFloat(Edit1->Text);
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{Edit1->Text=Edit1->Text+"2";
b=StrToFloat(Edit1->Text);
 }
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn3Click(TObject *Sender)
{Edit1->Text=Edit1->Text+"3";
b=StrToFloat(Edit1->Text);
 }
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn4Click(TObject *Sender)
{Edit1->Text=Edit1->Text+"4";
b=StrToFloat(Edit1->Text);
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn5Click(TObject *Sender)
{Edit1->Text=Edit1->Text+"5";
b=StrToFloat(Edit1->Text);
}
```

```
//---------------------------------------------------------
void __fastcall TForm1::BitBtn6Click(TObject *Sender)
{Edit1->Text=Edit1->Text+"6";
b=StrToFloat(Edit1->Text);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn7Click(TObject *Sender)
{Edit1->Text=Edit1->Text+"7";
b=StrToFloat(Edit1->Text);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn8Click(TObject *Sender)
{Edit1->Text=Edit1->Text+"8";
b=StrToFloat(Edit1->Text);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn9Click(TObject *Sender)
{Edit1->Text=Edit1->Text+"9";
b=StrToFloat(Edit1->Text);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn10Click(TObject *Sender)
{Edit1->Text=Edit1->Text+"0";
b=StrToFloat(Edit1->Text);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn11Click(TObject *Sender)
{a=StrToFloat(Edit1->Text);
k='+';
Edit1->Text="";
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn12Click(TObject *Sender)
{a=StrToFloat(Edit1->Text);
```

```
k='-';
Edit1->Text="";
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn13Click(TObject *Sender)
{a=StrToFloat(Edit1->Text);
k='*';
Edit1->Text="";
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn14Click(TObject *Sender)
{a=StrToFloat(Edit1->Text);
k='/';
Edit1->Text="";
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn16Click(TObject *Sender)
{switch(k){
   case '+':a=a+b;
            Edit1->Text=FloatToStr(a);break;
   case '-':a=a-b;
            Edit1->Text=FloatToStr(a);break;
   case '*':a=a*b;
            Edit1->Text=FloatToStr(a);break;
   case '/':a=a/b;
}
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn15Click(TObject *Sender)
{Edit1->Text="";
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



5.12-rasm. *Natija oynasi.*

Nazorat savollari

1. Tarmoqlanuvchi hisoblash jarayoniga ta'rif bering.
2. Shartsiz o'tish operatori va uning qo'llanishi.
3. Shartli o'tish operatorining to'liq ko'rinishi.
4. Shartli o'tish operatorining qisqa ko'rinishi.
5. Tanlash operatori va uning dasturda qo'llanishi.

Test savollari

1. C++ da tarmoqlanuvchi jarayonlar turlari to'g'ri ko'rsatilgan qatorni belgilang.

a) Shartli o'tish, shartsiz, parametrli
b) Shartli o'tish, shartsiz, old shartli
c) Shartli o'tish, shartsiz, tanlash
d) Shartli o'tish, shartsiz, ket shartli

2. Shartsiz o'tish operatorining umumiy ko'rinishi to'g'ri ko'rsatilgan qatorni belgilang.

a) Case < operator belgisi>;
b) Goto < operator belgisi>;
c) If < operator belgisi>;
d) While < operator belgisi>;

3. To'g'ri e'lon qilingan qatorni ko'rsating?

a) if(x>0){y=x*x*x; Label3->Caption=("y="+FloatToStr(y));
}
b) x>0 if { y =x*x*x;label3->Caption->("y="+floattostr(y));
}
c) if x>0{y=x*x*x; Label3->Caption->("y="+FloatToStr(y));
}
d) if(x>0){y=x*x*x; label3->Caption->("y="+FloattoStr(y));
}

4. 6<=X<10 C++ da qanday ifodalanadi?

a) (X<=6) and (X>10)
b) (X<=6) and (X<10)
c) (X>=6) and (X>10)
d) (X>=6) and (X<10) +

5. Shartsiz o'tish operatori to'g'ri ko'rsatilgan qatorni belgilang.

a) case <1 – qiymat >: <1 – operator(lar)>; break;
b) if (x=0) {y=x*x;}
c) goto 25; .. 25: y=x*x;
d) if U>0 D=sqrt (U); else D=U;

VI bob. C++ BUILDER 6 DASTURLASH TIZIMIDA TAKRORLANUVCHI JARAYONLARNI DASTURLASH

Tayanch so'zlar: takrorlanuvchi jarayon, sikl operatorlari, sikl tanasi, sikl parametri, old shartli takrorlanish, sharti keyin qo'yilgan takrorlanish, karrali sikllar, parametrli takrorlanish jarayoni.

6.1. Takrorlanish jarayonlarini tashkil qilish

Amaliyotda murakkab jarayonlarni dasturlashda ma'lum buyruqlar ketma-ketligini ma'lum shartlar asosida qayta-qayta bajarish zaruriyati tug'iladi. Ma'lum bir o'zgaruvchining turli qiymatlarida ma'lum buyruqlar tizimining biron-bir qonuniyatga asosan qayta-qayta bajarilishi "Takrorlanuvchi hisoblash jarayoni (sikl)" deb ataladi.

Takrorlanuvchi hisoblash jarayonining takror-takror hisoblanadigan qismini "Takrorlanishning tanasi" deb ataladi.

Takrorlanish ichida qiymatlari o'zgarib boradigan o'zgaruvchi "Takrorlanish o'zgaruvchisi" yoki "Takrorlanishning boshqaruvchi o'zgaruvchisi (sikl parametri)" deb yuritiladi.

Takrorlanuvchi jarayonning algoritmi umumiy holda quyidagilarni o'z ichiga olishi kerak:

1. Takrorlanishni tayyorlash – takrorlanishni boshlashdan oldin takrorlanishda qatnashadigan o'zgaruvchilarning boshlang'ich qiymatlari yoki takrorlanish o'zgaruvchisining boshlang'ich qiymati o'rnatiladi, takrorlanish o'zgaruvchisining o'zgarish qadami belgilanadi.

2. Takrorlanish tanasi – takrorlanish o'zgaruvchilarining turli qiymatlari uchun takror bajariladigan amallar ketma-ketligi ko'rsatiladi.

3. Takrorlanish o'zgaruvchisiga yangi qiymat berish – har bir takrorlanishdan avval o'zgaruvchiga o'zgarish qadamiga mos ravishda yangi qiymat beriladi.

4. Takrorlanishni boshqarish – takrorlanishni davom ettirish sharti tekshiriladi, takrorlanishning boshiga o'tish ko'rsatiladi.

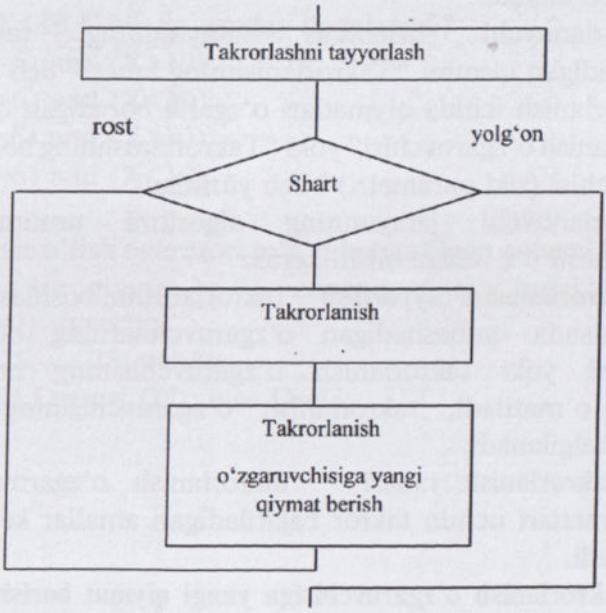
C++ algoritmik tilida uch xil ko'rinishda takrorlanuvchi hisoblash jarayonini tashkil qilish mumkin va bu jarayonlarni dasturlash uchun maxsus operatorlar belgilangan:

– avval sharti tekshiriladigan (sharti oldin kelgan "Toki") takrorlanish jarayoni, bu maxsus "While" operatori orqali amalga oshiriladi;

– sharti keyin tekshiriladigan ("... gacha") takrorlanish jarayoni: bu jarayon maxsus "Do ... while" operatori orqali amalga oshiriladi;

– parametrli takrorlanish jarayoni, bu jarayon maxsus "For" operatori yordamida amalga oshiriladi.

6.2. Avval sharti tekshiriladigan takrorlanish jarayoni



6.1-rasm. *Avval sharti tekshiriladigan takrorlanish algoritmi.*

Takrorlanuvchi jarayonning bu ko'rinishi takrorlanish soni oldindan noma'lum bo'lgan hollarda, ya'ni takrorlanishdan chiqish ma'lum shartga bog'liq bo'lgan hollarda ishlatiladi. Takrorlanishning bu jarayonida takrorlanishdan chiqish sharti takrorlanish tanasini bajarishdan oldin tekshiriladi (6.1-rasm).

Ushbu operatorning umumiy ko'rinishi quyidagichadir:

<Sharti avval tekshiriladigan takrorlanish operatori> (mantiqiy ifoda) <operatorlar>;

yoki while (m) o;

bu yerda, while – toki, "m" – mantiqiy ifoda, "o" – operatorlar yoki operatorlar guruhi bo'lib, u takrorlanish tanasini belgilaydi. Takrorlanish tanasida bitta yoki bir nechta operatorlar guruhi bo'lishi mumkin. Bunda operatorlar guruhi albatta {va} orasida yozilishi kerak.

Operatorning bajarilishi quyidagicha:

"m" mantiqiy ifodaning qiymati hisoblanadi. Agar "m" mantiqiy ifoda rost qiymatga ega bo'lsa, "o" operatori bajariladi va bu operator "m" mantiqiy ifodaning qiymati yolg'on bo'lgungacha qayta-qayta bajariladi.

Agar "m"– mantiqiy ifodaning qiymati birinchi tekshirishdayoq yolg'on bo'lsa, "o" – operatori biron marta ham bajarilmaydi va boshqaruv while operatoridan keyingi operatorga uzatiladi.

Agar "m" – mantiqiy ifoda rost bo'lib, kompyuter "o" operatorini bajarish davomida, biror sababga ko'ra takrorlanishdan chiqish talab etilsa, unda break operatori orqali amalga oshiriladi.

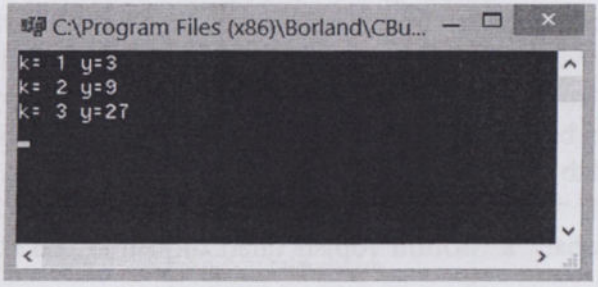
1-misol. "m" haqiqiy son berilgan bo'lsin. Shunday eng kichik butun musbat "k" sonini topish talab qilinsinki, bu son $3^k < m$ bo'lsin.

Masalaning dasturini tuzish uchun 3^k ifodaning qiymatini saqlaydigan qo'shimcha o'zgaruvchi kattalik kiritishimiz lozim. Agar biz bu kattalikni "y" identifikatori bilan belgilasak, u holda k=0 da u=1 dan boshlab, bitta qadam bilan o'zgarishida 3^k formulani (darajaga ko'tarishni) y=y*3 rekkurent formula bilan almashtiramiz. U holda takrorlanishdagi hisobdan chiqish sharti y>m bo'ladi.

Yuqorida ko‘rib chiqilgan operatordan foydalanib, ushbu misolning konsol muhitda dasturini tuzamiz.

```
//---------------------------------------------------------------
#include <stdio.h>
#include<iostream.h>
#include<conio.h>
#include <math.h>
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{float y, m; int k;
     y=1; k=0; m=27;
while (y<m)
   {y=y*3; k=k+1;
    cout<< "k= " << k<< " y="<< y <<endl;
   }getch();       return 0;  }
```

Dastur matni kiritib bo‘lingandan so‘ng F9 tugmasi bosiladi va natijada quyidagi dastur ko‘rinishi paydo bo‘ladi:



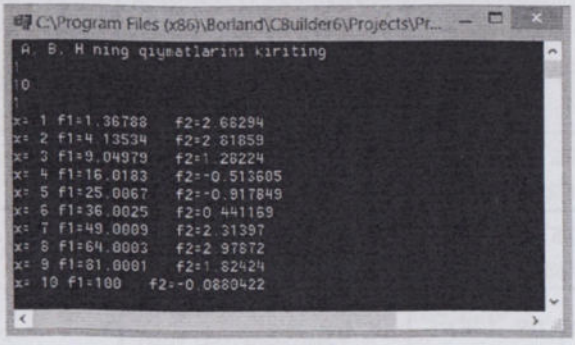
6.2-rasm. *Natija oynasi.*

2-misol. f1 va f2 funksiyalarning qiymatini hisoblash dasturi tuzilsin. X o‘zgaruvchisining qiymatlari a dan b gacha h qadam bilan o‘zgarsin.

f1= $\sqrt{x^3+e^{-x}}$ va f2= $1+2\sin x$.

Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
//---------------------------------------------------------
 #include <math.h>
#include<iostream.h>
#include<conio.h>
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------
int main(int argc, char* argv[])
{float a, b, h, f1, f2, x;
   cout<<" A, B, H ning qiymatlarini kiriting" <<endl;
   cin>>a>>b>>h;
   x=a;
   while (x<=b)
   {
f1=sqrt(pow(x,2))*x+1*exp(-x);
    f2=1+2*sin(x);
    cout<<"x= "<<x<<" f1="<<f1<<"   f2="<<f2<< endl;
    x=x+h;
   } getch();
return 0;
}
//---------------------------------------------------------
```

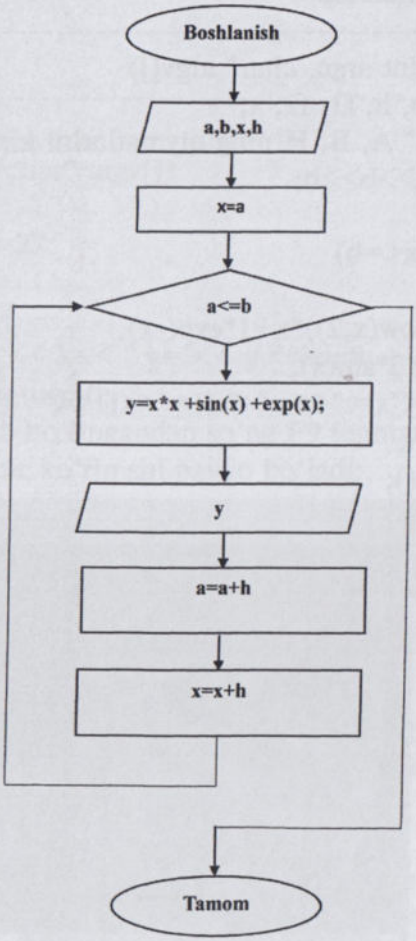


6.3-rasm. *Natija oynasi.*

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:

3-misol. $y = x^2 + \sin x + e^x$ funksiyaning qiymatini hisoblash dasturi tuzilsin. X o'zgaruvchisining qiymatlari a dan b gacha h qadam bilan o'zgarsin. Dastur vizual muhitda amalga oshirilsin.

Bu misolni yechish algoritmi quyidagi ko'rinishda bo'ladi:



6.4-rasm. *Dastur blok sxemasi.*

Misolni vizual muhitda dasturlash uchun 4 ta Label, 3 ta Edit, 3 BitBtn va 1 Memo komponentalari kerak bo'ladi.

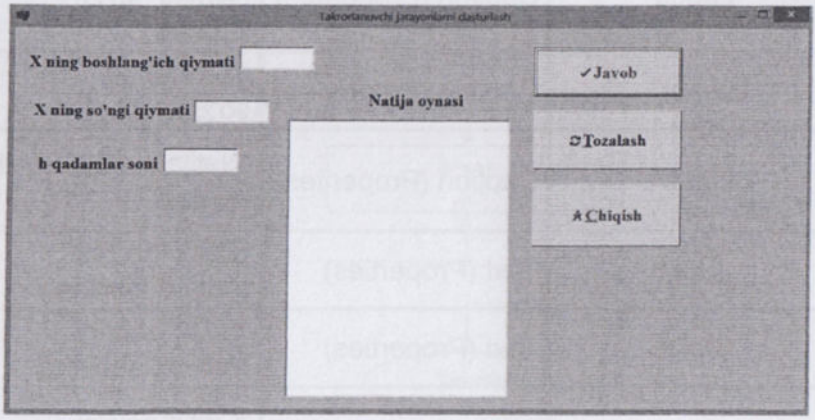
Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

6.1-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Takrorlanuvchi jarayonlarni dasturlash" so'zi kiritiladi.
Label1	Caption (Properties)	"X ning boshlang'ich qiymati" so'zi kiritiladi.
Label2	Caption (Properties)	"X ning so'nggi qiymati" so'zi kiritiladi.
Label3	Caption (Properties)	"h qadamlar soni" so'zi kiritiladi.
Label4	Caption (Properties)	"Natija oynasi" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.
Edit2	Text (Properties)	"Edit2" so'zini o'chirib tashlang.
Edit3	Text (Properties)	"Edit3" so'zini o'chirib tashlang.

BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Javob" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkRetry" xususiyati tanlanadi.
	Caption (Properties)	"Tozalash" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.
Memo1	Lines (Properties)	"Memo1" so'zini o'chirib tashlang.



6.5-rasm. *Dastur ko'rinishi.*

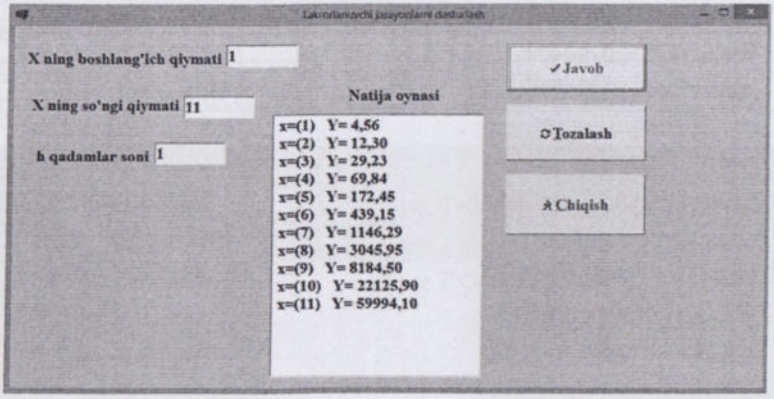
Mavjud komponentalar xususiyatlari kiritilgandan so‘ng dastur dizayni 6.5-rasmdagi ko‘rinishga keladi:

Dastur dizayni tayyor bo‘lganidan so‘ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#include <math.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
float x,y,a,b,h;
  a=StrToFloat(Edit1->Text);
  b=StrToFloat(Edit2->Text);
  h=StrToFloat(Edit3->Text);
  Memo1->Clear();
    x=a;
  while (a<=b)
{
    y=x*x +sin(x) +exp(x);
Memo1->Lines->Add("x=("+FloatToStr(a)+") Y= " + Float-
ToStrF (y, ffFixed,6,2));
    x=x+h;
    a=a+h; }
```

```
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn2Click (TObject *Sender)
{
Edit1->Clear();
Edit2->Clear();
Edit3->Clear();
Memo1->Clear();
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn2Click (TObject *Sender)
{
Close();
}
//---------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



6.6-rasm. *Natija oynasi.*

6.3. Sharti keyin tekshiriladigan takrorlanish jarayoni

Takrorlanish jarayonining bu ko'rinishi ham takrorlanish soni oldindan ma'lum bo'lmagan hollarda ishlatiladi. Bunday jarayonda

biror buyruq yoki buyruqlar tizimi berilgan shart bajarilgunga qadar takror-takror bajariladi. Bu jarayonni oldin sharti tekshiriladigan takrorlanish jarayonidan farqi shundaki, bunda takrorlanishning tanasi hech bo'lmaganda bir marta bo'lsa ham bajariladi, chunki takrorlanishdan chiqish sharti takrorlanishning tanasi bajarilgandan keyin tekshiriladi.

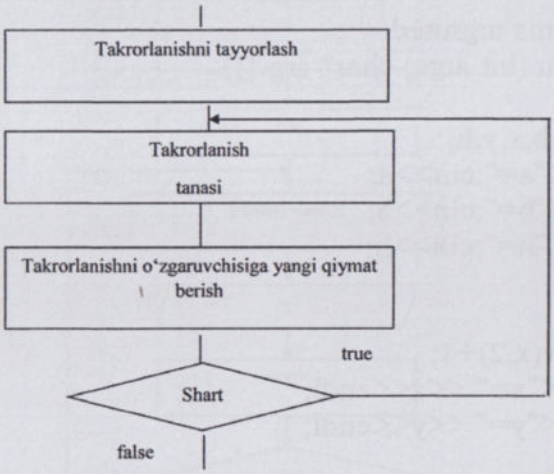
Bunday takrorlanish jarayonini dasturlashda maxsus "do... while" operatoridan foydalaniladi.

Ushbu operatorning umumiy ko'rinishi quyidagicha:

<Sharti keyin tekshiriladigan takrorlanish operatori> = do <operatorlar guruhi> while <mantiqiy ifoda>;

yoki do L while (M);

Bu yerda: do – bajarmoq, while – "toki" ma'nosini anglatuvchi xizmatchi so'zlar; L – takrorlanish tanasi, uning tarkibida bitta operator yoki operatorlar guruhi bo'lishi mumkin; M – mantiqiy ifoda.



6.7-rasm. *Sharti keyin tekshiriladigan takrorlanish algoritmi.*

Operator quyidagi tartibda bajariladi:

– takrorlanish tanasiga kirgan operatorlar birin-ketin bajariladi. So'ng M mantiqiy ifodaning qiymati topiladi, ya'ni

shart tekshiriladi. Agar bu shart bajarilmasa (M ning qiymati False (yolg'on) bo'lsa) boshqaruv takrorlashdan tashqariga, while so'zidan keyingi operatorga uzatiladi. Aks holda takrorlanish davom etadi.

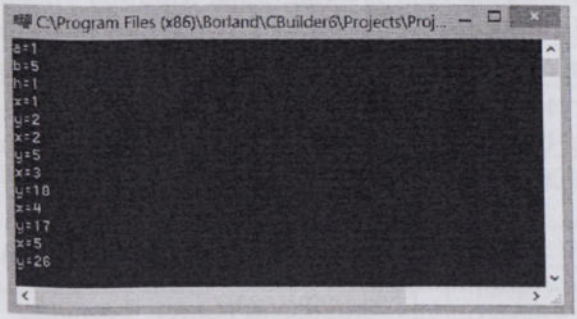
Takrorlanish jarayonining bunday ko'rinishi yuqorida keltirilgan takrorlanishdan shu bilan farq qiladiki, bunda takrorlanish tanasi hech bo'lmaganda bir martda bajariladi.

1-misol. $y = x^2 + 1$ funksiyaning qiymatini hisoblash dasturi tuzilsin. X o'zgaruvchisining qiymatlari a dan b gacha h qadam bilan o'zgarsin.

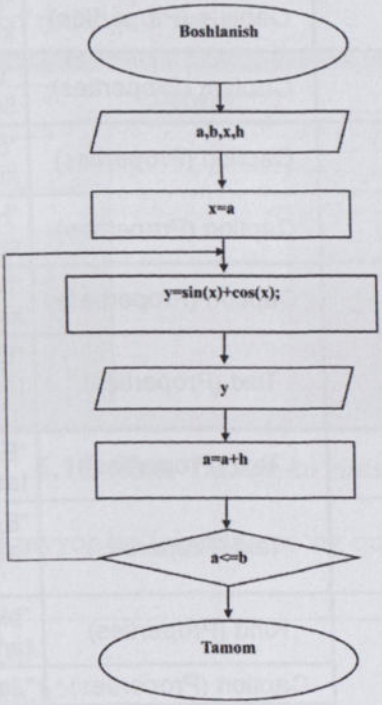
Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
#include <vcl.h>
#include <iostream.h>
#include <math.h>
#include <conio.h>
#pragma hdrstop
//---------------------------------------------------------
 #pragma argsused
int main(int argc, char* argv[])
{
float a,b,x,y,h;
cout<<"a=";cin>>a;
cout<<"b=";cin>>b;
cout<<"h=";cin>>h;
x=a;
do
{y=pow(x,2)+1;
cout <<"x=" <<x<<endl;
cout <<"y=" <<y<<endl;
x=x+h;
}
while(x<=b);
getch();
return 0;
}
```

Dastur matni kiritib bo‘lingandan so‘ng F9 tugmasi bosiladi va natijada quyidagi dastur ko‘rinishi paydo bo‘ladi:



6.8-rasm. *Natija oynasi.*



6.9-rasm. *Dastur blok sxemasi.*

2-misol. $y = \sin x + \cos x$ funksiyaning qiymatini hisoblash dasturi tuzilsin. X o'zgaruvchisining qiymatlari a dan b gacha h qadam bilan o'zgarsin. Dastur vizual muhitda amalga oshirilsin.

Bu misolni yechish algoritmi quyidagi ko'rinishda bo'ladi:

Misolni vizual muhitda dasturlash uchun 4 ta Label, 3 ta Edit, 3 BitBtn va 1 Memo komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

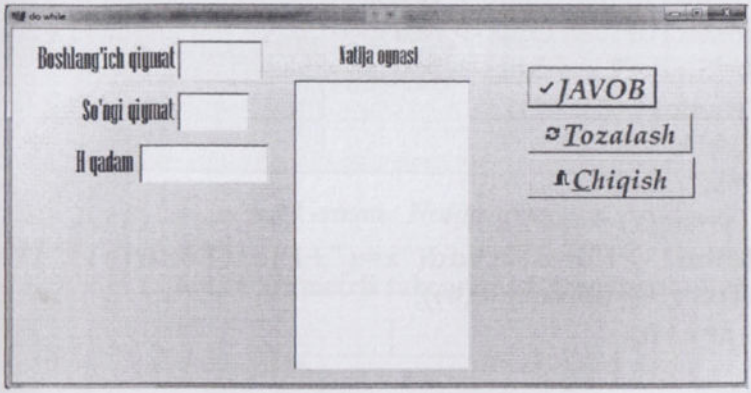
6.2-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"do while" so'zi kiritiladi.
Label1	Caption (Properties)	"Boshlang'ich qiymat" so'zi kiritiladi.
Label2	Caption (Properties)	"So'nggi qiymat" so'zi kiritiladi.
Label3	Caption (Properties)	"H qadamlar soni" so'zi kiritiladi.
Label4	Caption (Properties)	"Natija oynasi" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.
Edit2	Text (Properties)	"Edit2" so'zini o'chirib tashlang.
Edit3	Text (Properties)	"Edit3" so'zini o'chirib tashlang.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Javob" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

BitBtn2	Kind (Properties)	"bkRetry" xususiyati tanlanadi.
	Caption (Properties)	"Tozalash" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
		Close(); kiritiladi.
Memo1	Lines (Properties)	"Memo1" so'zi o'chiriladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



6.10-rasm. *Dastur ko'rinishi.*

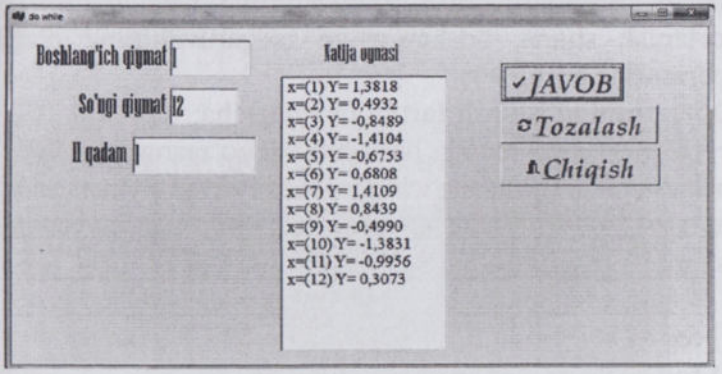
Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------------
#include <vcl.h>
#include <math.h>
#pragma hdrstop
```

```
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
float a,b,h,y,x;
  h=StrToFloat(Edit3->Text);
  a=StrToFloat(Edit1->Text);
  b=StrToFloat(Edit2->Text);
  Memo1->Clear();
 do {
 x=a;
    y=sin(x) +cos(x);
  Memo1->Lines->Add("x=("+FloatToStr(a)+")Y="+
FloatToStrF (y,ffFixed, 6,4));
    a=a+h;
 }
  while (a<=b);
 }
 //---------------------------------------------------------
 void __fastcall TForm1::BitBtn2Click (TObject *Sender)
 {
 Edit1->Clear();
 Edit2->Clear();
 Edit3->Clear();
 Memo1->Clear();
```

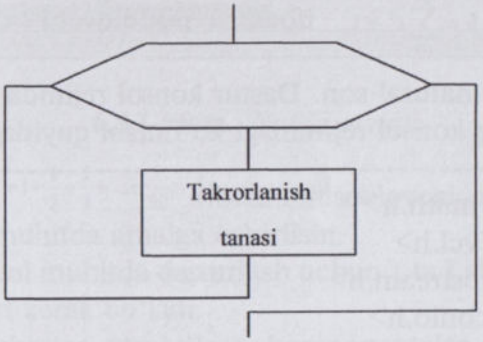
```
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click (TObject *Sender)
{
Close();
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



6.11-rasm. *Natija oynasi.*

6.4. Parametrli takrorlanish jarayoni



6.12-rasm. *Parametrli takrorlanish algoritmi.*

Yuqorida keltirilgan takrorlanish operatorlarini odatda takrorlanish soni noma'lum bo'lgan hollarda ishlatish maqsadga muvofiqdir. Takrorlanish jarayonining takrorlanish soni uning bajarilishidan oldin ma'lum bo'lsa parametrli takrorlanish operatoridan foydalanish qulay hisoblanadi. Bunday takrorlanish jarayonining algoritmi quyidagi tarkibga ega:

Bu operatorning dasturdagi umumiy yozilish konstruksiyasi quyidagi ko'rinishga ega: <parametrli takrorlanish operatori> = for (< boshqaruvchi o'zgaruvchining boshlang'ich qiymati, takrorlanish sharti, boshqaruvchi o'zgaruvchining o'zgarishi>) {<operatorlar guruhi>};

Operatorning ishlash tartibi quyidagicha:

– takrorlanish tanasi boshqaruvchi o'zgaruvchisining hamma qiymatlari uchun (boshlang'ich qiymatidan oxirgi qiymatigacha) qayta-qayta takrorlanadi. Masalan, bu operatorlarni quyidagi ko'rinishlarda yozish mumkin:

1) for (x=a; x<=b; x++) y=m;
2) for (x=b; x<=a; x--) y=m;
3) for (x=a + b; x<=c*k; x++) y=k;
4) for (int i =1, s = 0; i<=100; i++) s += i;

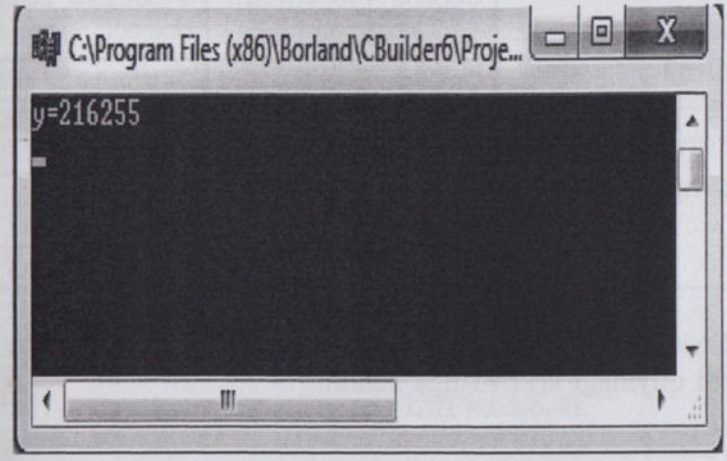
1-misol. $y=\sum_{i=1}^{30} i^3+1$ ifodani hisoblovchi dastur tuzilsin. Bu yerda, i – natural son. Dastur konsol rejimda bajarilsin.

Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
//---------------------------------------------------------
#include <math.h>
#include <vcl.h>
#include<iostream.h>
#include<conio.h>
#pragma hdrstop
//---------------------------------------------------------
```

```
#pragma argsused
int main(int argc, char* argv[])
{ float y;
y=0;
for (int i=1;i<=30;i++)
y=y+pow(i,3)+1;
cout<<"y="<<y<<endl;
getch(); return 0; }
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



6.13-rasm. *Natija oynasi.*

2-misol. $S=1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{50}$ ifodani hisoblovchi dastur tuzilsin. Dastur vizual muhitda amalga oshirilsin.

Misolni vizual muhitda dasturlash uchun 1 ta Label va 2 BitBtn komponentalari kerak bo'ladi.

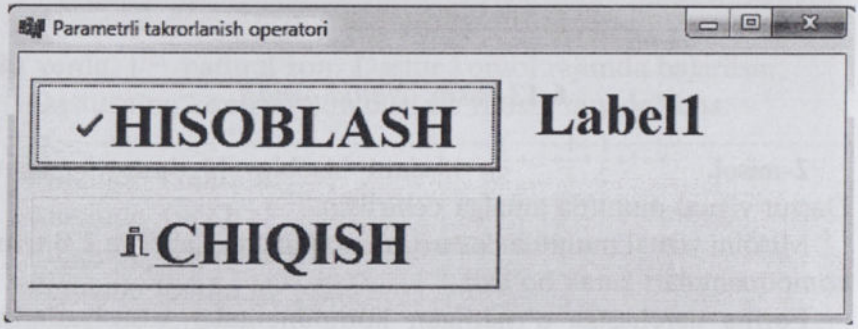
Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

6.3-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Parametrli takrorlanish operatori" so'zi kiritiladi.
Label1	Caption (Properties)	"Natija" so'zi kiritiladi.
	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Hisoblash" so'zi kiritiladi.
BitBtn2	OnClick (Events)	Dastur matni kiritiladi.
BitBtn1	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:

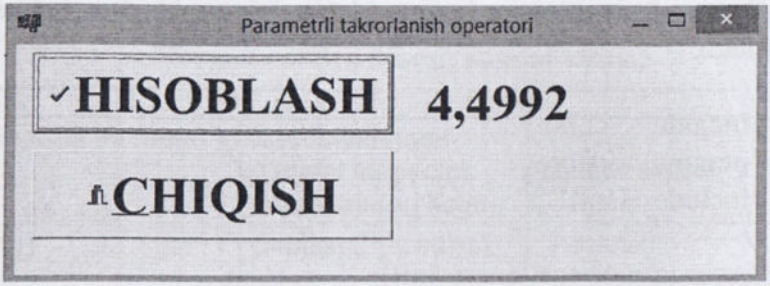


6.14-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
int i;
float s;
s=0;
for (i=1;i<=50; i++)
{s=s+(float)1/i;}
Label1->Caption= FloatToStrF(s,ffFixed,6,4);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------
```

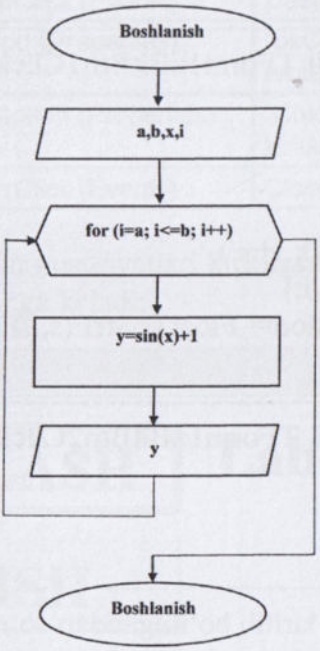
Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



6.15-rasm. *Natija oynasi.*

3-misol. $y = \sin(x) + 1$ funksiyaning qiymatini hisoblash dasturi tuzilsin. X o'zgaruvchisining qiymatlari a dan b gacha 1 qadam bilan o'zgarsin. Dastur vizual muhitda amalga oshirilsin.

Bu misolni yechish algoritmi quyidagi ko'rinishda bo'ladi:



6.16-rasm. *Dastur blok sxemasi.*

Misolni vizual muhitda dasturlash uchun 2 ta Label, 2 ta Edit, 2 BitBtn va 1 Memo komponentalari kerak bo'ladi.

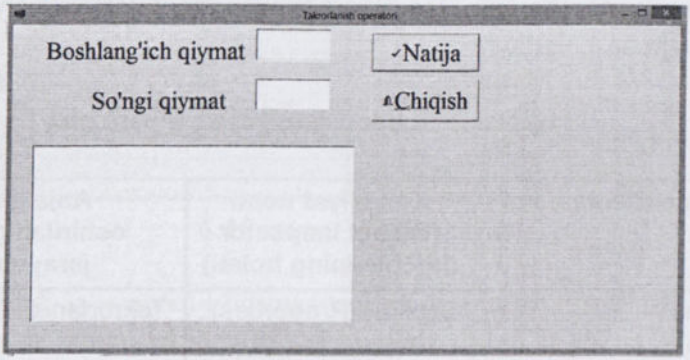
Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

6.3-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Takrorlanish operatori" so'zi kiritiladi.
Label1	Caption (Properties)	"Boshlang'ich qiymat" so'zi kiritiladi.
Label2	Caption (Properties)	"So'nggi qiymat" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.
Edit2	Text (Properties)	"Edit2" so'zini o'chirib tashlang.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Natija" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.
Memo1	Lines (Properties)	"Memo1" so'zini o'chirib tashlang.

Mavjud komponentalar xususiyatlari kiritilgandan so‘ng dastur dizayni quyidagi ko‘rinishga keladi:

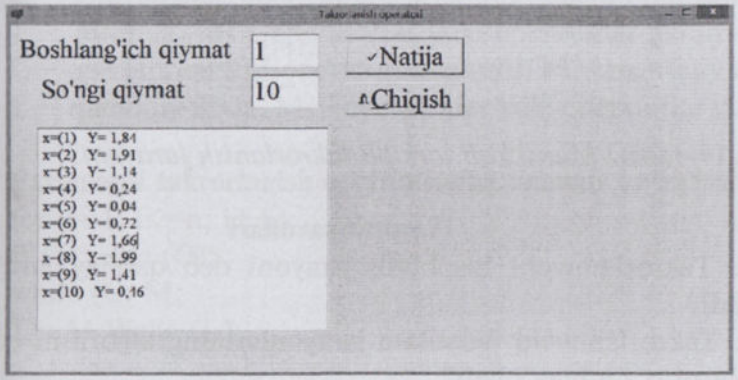


6.17-rasm. *Dastur ko‘rinishi.*

```
//---------------------------------------------------------
#include <vcl.h>
#include <math.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
float x,y;
int a,b;
a=StrToFloat(Edit1->Text);
```

```
    b=StrToFloat(Edit2->Text);
        Memo1->Clear();
    for (x=a; x<=b; x++)
   {
   y=sin(x)+1;
   Memo1->Lines->Add("x=("+FloatToStr(x)+")
Y="+FloatToStrF(y,ffFixed, 6,2));
   }
   }
   //---------------------------------------------------------
   void __fastcall TForm1::BitBtn2Click(TObject *Sender)
   {
   Close();
   }
   //---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



6.18-rasm. *Natija oynasi.*

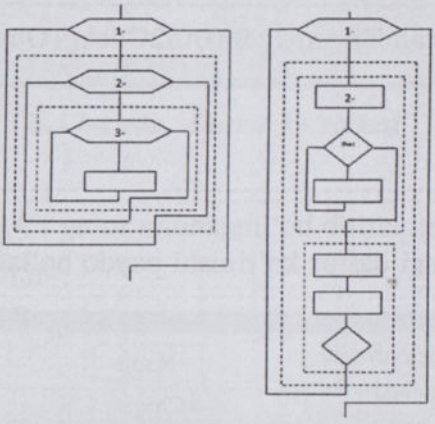
6.5. Murakkab takrorlanish jarayonlari

Yuqorida keltirilgan operatorlar asosida murakkab takrorlanishlar tashkil qilish mumkin. Agar takrorlanish jarayonlarining tanasi takrorlanish strukturasidan tashkil topgan

bo'lsa u holda bunday takrorlanish "Ichma-ich joylashgan yoki murakkab" deb ataladi, ya'ni boshqacha qilib aytganda, bitta takrorlanish ichiga bir yoki bir necha boshqa takrorlanishlar kirsa murakkab tarkibli takrorlanishli dasturlar hosil bo'ladi. Bunday murakkab tarkibli jarayon 6.19-rasmda keltirilgan.

Boshqa takrorlanish jarayonlarini o'z ichiga olgan takrorlanish "Tashqi takrorlanish" deb ataladi.

Takrorlanish jarayonlarining ichida joylashgan takrorlanish "Ichki takrorlanish" deb ataladi.



6.19-rasm. *Murakkab tarkibli takrorlanish jarayoni.*

Nazorat savollari

1. Takrorlanuvchi hisoblash jarayoni deb qanday jarayonga aytiladi?
2. Takrorlanuvchi hisoblash jarayonlarining algoritmi qanday qismlarni o'z ichiga oladi?
3. Takrorlanuvchi jarayon tanasi va takrorlanish o'zgaruvchisi haqida tushuncha bering.
4. Avval sharti tekshiriladigan takrorlanish jarayonini tashkil qilish (while, takrorlanish, takrorlanuvchi hisoblash jarayoni).
5. Sharti keyin tekshiriladigan takrorlanish jarayonini tashkil qilish (do while, takrorlanish, takrorlanuvchi hisoblash jarayoni).

Test savollari

1. C++ da takrorlanishni hosil qilish uchun qaysi opreatorlardan foydalaniladi?

a) for, while, do while, repeat
b) for, while, do while
c) for, while, repeat
d) if else, goto, case

2. C++ da avval sharti tekshiriladigan takrorlanish operatori qaysi qatorda keltirilgan.

a) for
b) do while
c) while
d) goto

3. while (L) M; takrorlanish operatorida (L) va M nima?

a) L – tenglamalar, M – operatorlar yoki operatorlar guruhi
b) L – formula, M – operatorlar yoki operatorlar guruhi
c) L – operatorlar yoki operatorlar guruhi, M – mantiqiy ifoda
d) L – mantiqiy ifoda, M – operatorlar yoki operatorlar guruhi

4. Parametrli takrorlanish operatorining umumiy ko'rinishi?

a) for (i=1; i<=n; i++)
b) if (M) else (O);
c) while (L) M;
d) Do M Wxile (L);

5. C++ da keyin sharti tekshiriladigan takrorlanish operatori

a) goto
b) do while
c) for
d) while

VII bob. BORLAND C++ BUILDER 6 DASTURLASH TIZIMIDA MASSIVLAR BILAN ISHLASH

Tayanch so'zlar: massiv, massiv elementi, bir o'lchovli massiv, ko'p o'lchovli massiv, massivni xotiraga kiritish, matritsa, StringGrid komponentasi.

7.1. Massiv tushunchasi va uning turlari

C++ tilida ishlatiladigan ma'lumotlar oddiy va murakkab toifadagi ma'lumotlarga bo'linadi. Oddiy toifadagi ma'lumotlarning boshqa toifadagi ma'lumotlardan asosiy farqlanuvchi belgisi – ularning tartiblanganligi va yaxlitligidadir, ya'ni masalan int toifasiga mansub ixtiyoriy kattalik alohida raqamlarga bo'linmaydigan yaxlit kattalikdir (int toifasidagi kattalik sifatida bitta butun son tushuniladi). Murakkab toifadagi ma'lumotlar bir nechta kattaliklar ketma-ketligidan, to'plamidan iborat. Shuning uchun bu toifalarni tarkiblashgan toifalar deb yuritiladi.

C++ tilida massiv toifasi deyarli har bir dasturda ishlatiladi. Massiv – bu umumiy nomga ega bo'lgan bir xil toifadagi ma'lumotlarning tartiblangan ketma-ketligidir. Odatda ma'lumotlar ro'yxatlarini, turli ko'rinishdagi jadval elementlarini massivlar ko'rinishida ifodalash qulaydir. Massivlar xotiradan joy ajratish uchun xizmat qiladi. Masalan int toifasidagi 5 sonni saqlash kerak. Buning uchun quyidagi qatorni kiritish kerak bo'ladi:

```
int myMassiv[5];
```

Kompilyator xotiradan massiv uchun joy ajratadi. Int toifasidagi ma'lumot uchun 4 mb joy talab etilganligi sababli, massiv uchun umumiy 20 mb joy ajratiladi. Buni quyidagi rasmda ko'rish mumkin[5].

myMassiv[0]	myMassiv[1]	myMassiv[2]	myMassiv[3]	myMassiv[4]
baseAddr	baseAddr+4	baseAddr+8	baseAddr+12	baseAddr+16

7.1-rasm. *Massiv uchun xotiraning taqsimlanishi.*

[5] Alex Allain. Jumping into C++. USA, 2014. – p.111–114.

Massiv tushunchasini A – umumiy nomga ega va bir toifadagi qo'zg'almas kattaliklar ketma-ketligidan iborat bo'lgan sonli vektor misolida tushuntirish mumkin:

$$A(5) = (a1, a2, a3, a4, a5),$$

Bu yerda a1, a2...– massiv elementlaridir. Ularni ifodalashda ko'rsatkichli (indeksli) o'zgaruvchilardan foydalaniladi.

Matematika kursidan ma'lumki, ko'rsatkich (indeks) o'zgaruvchilarning tartiblangan ketma-ketlikdagi o'rnini bildiradi.

7.2. C++ tilida massivlarni tavsiflash

Agar dasturda massiv ishlatilayotgan bo'lsa, undan foydalanishdan avval e'lon qilinishi lozim. Massiv quyidagi ko'rinishda e'lon qilinadi:

<toifa> <massiv nomi> [<elemtlar soni>] =

= {boshlang'ich qiymatlar};

bu yerda, <massiv nomi> – ixtiyoriy identifikator, <elementlar soni> – ko'rsatkichli ifoda, u massiv elementlarining sonini belgilaydi va ko'rsatkichlarni yozish uchun ishlatiladigan belgilarni ko'rsatadi, shuning uchun bu toifa sifatida haqiqiy va cheklanmagan butun toifadan tashqari barcha oddiy toifalarni ishlatish mumkin, <toifa> – massiv elementlarining toifasi bo'lib, bu toifa sifatida fayl va to'plam toifasidan boshqa barcha toifalarni ishlatish mumkin.

Quyida to'g'ri tavsiflangan massivlarga misollar keltirib o'tamiz:

– float a_massiv [4];
– char year [10];
– int butun [22];
– bool mantiq [44].

Keltirilgan misoldagi a_massiv elementlari haqiqiy sonlardan iborat bo'lgan, 4 ta elementdan tashkil topgan massiv. Indekslari 0 dan 3 gacha bo'lgan sonlardan iborat va uni quyidagi ko'rinishda tasvirlash mumkin bo'ladi:

float a_massiv [4];				
Massiv elementlari	a_massiv [0]	a_massiv [1]	a_massiv [2]	a_massiv [3]
Massiv qiymatlari	2.2	6	1.3	6.8

7.2-rasm. *Massivning elementi va qiymatlari.*

Ko‘rsatkich va ko‘rsatkich toifasi tushunchalari o‘rtasida o‘zaro farq mavjud bo‘lib, ko‘rsatkich toifasi massiv elementlari soni va ularning tartiblanganligini bildiradi va u massivni tavsiflash bo‘limida ishlatiladi. Ko‘rsatkich esa massiv elementining tartib raqamini belgilaydi va operatorlar bo‘limidagina ishlatiladi. Agar biror-bir massivga murojaat qilish uchun uning to‘liq nomi, ya’ni o‘zgaruvchining nomi ishlatilsa, massivning alohida elementiga murojaat qilish uchun ko‘rsatkichli o‘zgaruvchi ishlatiladi.

Ifodalarda ko‘rsatkichli o‘zgaruvchilar qiymat berish operatorining chap tarafida ham, o‘ng tarafida ham ishtirok etishi mumkin va ular ustida solishtirish amallarini, tartiblash, arifmetik amallar, eng kichik va eng katta qiymatni topish amallarini, ya’ni uning tayanch toifasi ustida bajarilishi mumkin bo‘lgan barcha amallarni bajarish mumkin. Masalan, agar bazaviy toifa int bo‘lsa, u holda butun toifa ustida bajarilishi mumkin bo‘lgan barcha amallar, hatto standart funksiyalarni ham qo‘llash mumkin.

Shu vaqtgacha biz elementlari faqat bitta ko‘rsatkichli massivlarni, ya’ni bir o‘lchamli massivlarni ko‘rib chiqdik. C++ dasturlash tilining massiv elementlari toifasiga ularning hammasi bir toifaga mansub bo‘lishi kerakligidan boshqa hech qanday cheklanishlar qo‘yilmasligi massiv elementlari sifatida satr va ustunlar orqali qiymatlarni ikki o‘lchamli o‘zgaruvchilar orqali saqlash imkoniyatini beradi. Bunday massivlar ko‘p o‘lchamli massivlarni tashkil qiladi.

Ko'p o'lchamli massivlar dasturda quyidagicha tavsiflanadi:

<toifa> <massiv nomi> [<massiv satrlar soni>] [< massiv ustunlar soni >];

Bir o'lchamli massivlardan farqi shundaki, ikki o'lchamli massivda nomidan keyin [] qavs ichida ikkita qiymat yoziladi. Birinchisi satrlar sonini, ikkinchisi esa ustunlar sonini bildiradi. Ya'ni ikki o'lchamli massiv elementiga ikkita indeks orqali murojaat qilinadi. Masalan, ikki o'lchovli massivlarga misol qilib matritsalarni keltirish mumkin. Ikki o'lchovli massivni e'lon qilish quyidagicha amalga oshiriladi:

- int a [3][3];
- float b [2][4].

A matritsaning elementlari quyidagicha ko'rinishga ega bo'ladi:

$$a = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$

A matritsa 3 ta satr va 3 ta ustunga ega. M ta satr va n ta ustunga ega bo'lgan massivga m*n o'lchamli massiv deb ataladi. Agar satrlar va ustunlar soni teng bo'lsa kvadrat massiv deyiladi.

7.3. Massiv elementlarini xotiraga kiritish

Dasturda massivlardan foydalanish uchun massiv elementlari qiymatlari xotiraga kiritilgan bo'lishi zarur. Massiv elementlariga qiymat berishda ma'lumotlarni kiritish yoki o'zlashtirish operatorlaridan foydalanish mumkin.

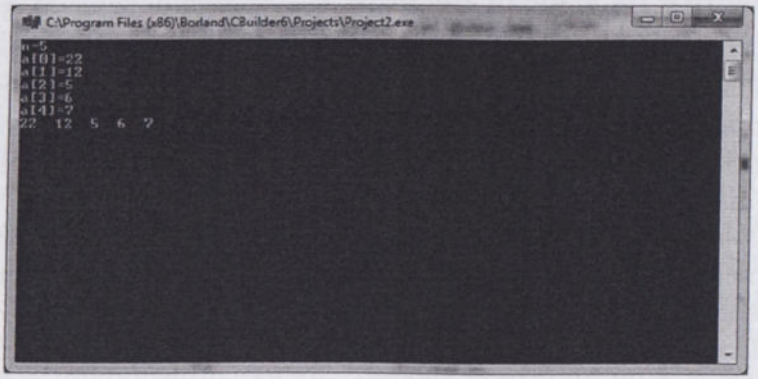
1-misol. Elementlari butun sonlardan iborat bo'lgan, n elementdan tashkil topgan massiv elementlarini kirituvchi va ekranga chiqaruvchi dastur tuzilsin.

Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
//---------------------------------------------------------
#include <vcl.h>
```

```
#include <iostream.h>
#include <conio.h>
#pragma hdrstop
//---------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{int a[10]={0};
int n;
cout << "n="; cin>>n;
for (int i=0; i<n;i++)
{
cout <<"a["<<i<<"]=";
cin>> a[i];
}
for (int i=0; i<n; i++)
cout<<a[i]<<"";
  getch ();
  return 0;
}
//---------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



7.3-rasm. *Natija oynasi.*

2-misol. N ta elementdan tashkil topgan massiv berilgan. Shu massiv elementlari yig'indisini hisoblovchi dastur tuzilsin.

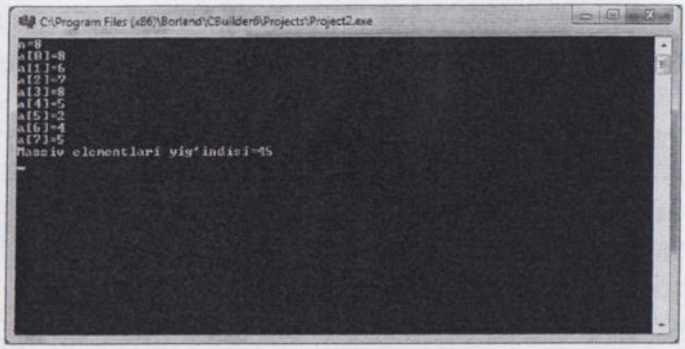
Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
//---------------------------------------------------------
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
int a[10]={0};
int n;
int s=0;
cout << "n="; cin>>n;
for (int i=0; i<n;i++)
{
cout <<"a["<<i<<"]=";
cin>> a[i];
s+=a[i];
}
cout<<"Massiv elementlari yig'indisi="<<s<<endl;
      getch ();
      return 0;
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:

Massivlar tavsiflanganda initsializatsiya qilinishi, ya'ni boshlang'ich qiymatlari ko'rsatilishi mumkin. Misol uchun:

```
float C[]={1,-1,2,10,-12.5};
```



7.4-rasm. *Natija oynasi.*

Bu misolda massiv chegarasi avtomatik aniqlanadi. 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, qolgan elementlar qiymati 0 ga teng deb olinadi.

Quyidagi misolda ikki o'lchamli m va d massivi tavsiflangan va qiymatlar kiritilgan:

```
const int qator = 3, ustun= 4;
int m[qator][ustun] = {{1, 3, 3, 6}, {1, 1, 2, 2}, {3, 3, 2, 0}};
float d[2][3]={(1,-2.5,10),(-5.3,2,14)};
```

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)} ;
double x[][2]={(1.1,1.5),(-1.6,2.5),(3,-4)} ;
```

7.4. Forma ilovasida massivlar bilan ishlash usullari

Forma ilovasida massivlar bilan ishlash uchun StringGrid jadval komponentasi juda qulay hisoblanadi. StringGrid

komponentasi additional komponentlar palitrasida joylashgan bo'lib, ma'lumotlarni, masalan matritsa elementlari qiymatini ekranda jadval ko'rinishida tasvirlash, ular qiymatini kiritish va tahrirlash uchun ishlatiladi. Jadval qator va ustun nomerlari noldan boshlanadi. Jadval ustun va qatorlar sonini keraklicha o'zgartirish mumkin. Bu uning xossasi yordamida aniqlanadi. Jadvalning har bir kesishgan ustun va satri yacheyka deyilib, unga kiritilgan ma'lumot simvol qatori bo'lib aniqlanadi.

StringGrid jadval komponentasining asosiy xossalarini 7.1-jadvalda ko'rib chiqamiz:

7.1-jadval.

StringGrid komponentasining asosiy xossalari

№	Xossalari	Vazifasi
1.	Name	Komponentaning nomi. Komponentaning xossaliriga kirish uchun ishlatiladi
2.	ColCount	Jadvaldagi ustunlar sonini aniqlaydi
3.	RowCount	Jadvaldagi satrlar sonini aniqlaydi
4.	Cells	Jadval yacheykasi, col ustun nomeri va row qator nomerini belgilaydi
5.	FixedCols	Fiksirlangan ustunlar sonini aniqlaydi
6.	FixedRows	Fiksirlangan satrlar sonini aniqlaydi
7.	Options -> goEd-iting	Jadval holatini aniqlaydi (aniqlash uning parametrlariga asosan bajariladi, masalan GoEditing parametri true qiymatga ega bo'lsa yacheykani tahrirlash mumkin, aks holda mumkin emas)
8.	Options -> goTab	<Tab> klavishasini jadvalning keyingi yacheykasiga o'tish uchun ishlatish holatini belgilaydi, true – o'tish mumkin, false – o'tish taqiqlanadi
9.	DefaultColWidth	Jadvalning boshlang'ich ustunlar kengligini aniqlaydi

10.	DefaultRowHeight	Jadvalning boshlang'ich ustunlar balandligini aniqlaydi
11.	GridLineWidth	Jadval yacheykalari chegara chizig'ining kengligini belgilaydi
12.	Left	Formaning chap chegarasidan jadvalning chap chegarasigacha masofa o'rnatadi
13.	Top	Jadvalning yuqori chegarasidan formaning yuqori chegarasigacha masofani o'rnatish
14.	Height	Jadval maydonining balandligini o'rnatadi
15.	Width	Jadval maydonining kengligini o'rnatadi
16.	Font	Jadval yacheykalaridagi yozuvlarning shriftini belgilash

3-misol. 10 ta elementdan iborat massivni hosil qiling. Massivning 10 ta elementini har xil sonlar bilan to'ldirish va ularni oshib borishi bo'yicha tartiblash dasturi tuzilsin. Dastur vizual muhitda amalga oshirilsin.

Misolni vizual muhitda dasturlash uchun 2 ta Label, 2 ta StringGrid, 3 ta BitBtn komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

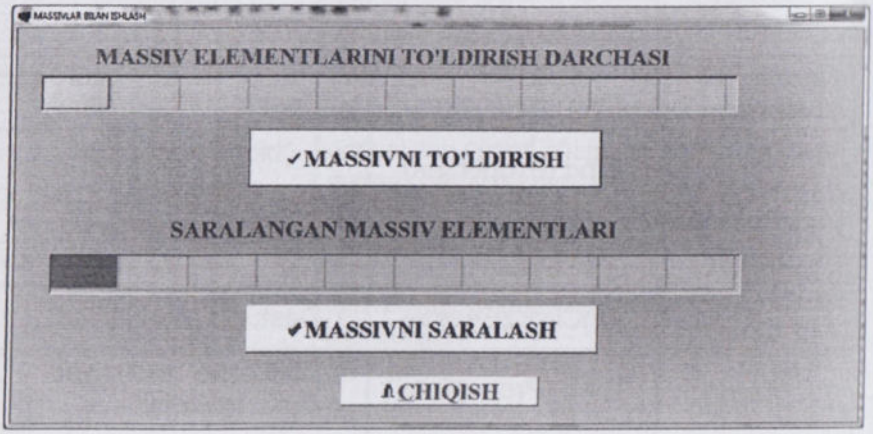
7.2-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Massivlar bilan ishlash" so'zi kiritiladi.
Label1	Caption (Properties)	"Massiv elementlarini to'ldirish darchasi" so'zi kiritiladi.
Label2	Caption (Properties)	"Saralangan massiv elementlari" so'zi kiritiladi.

BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Massivni to'ldirish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkAll" xususiyati tanlanadi.
	Caption (Properties)	"Massivni saralash" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.
StringGrid1	FixedCols (Properties)	0 sonini kiritamiz .
	FixedRows (Properties)	0 sonini kiritamiz.
	ColCount (Properties)	10 sonini kiritamiz.
	RowCount (Properties)	1 sonini kiritamiz.
StringGrid2	FixedCols (Properties)	0 sonini kiritamiz.
	FixedRows (Properties)	0 sonini kiritamiz.
	ColCount (Properties)	10 sonini kiritamiz.
	RowCount (Properties)	1 sonini kiritamiz.

Mavjud komponentalar xususiyatlari kiritilgandan so‘ng dastur dizayni quyidagi ko‘rinishga keladi:



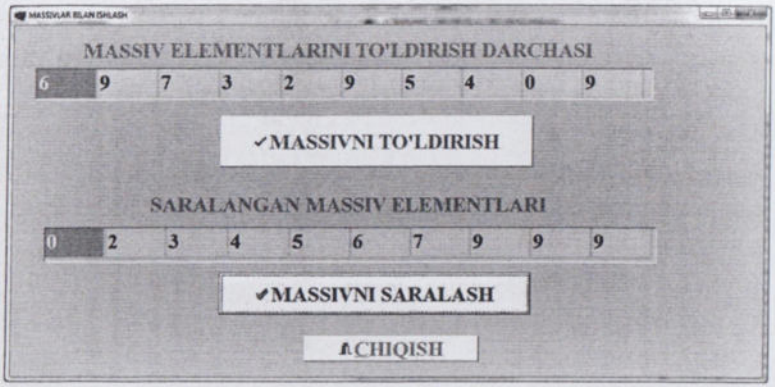
7.5-rasm. *Dastur ko‘rinishi.*

Dastur dizayni tayyor bo‘lganidan so‘ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int  a[10][1];
int i,j;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
```

```
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
randomize();
for(i=0;i<=9;i++)
{
a[i][0]=random(10);
StringGrid1->Cells[i][0]=IntToStr(a[i][0]);
}
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{ int k;
      for(j=0;j<=9;j++)
       {for(i=0;i<=8;i++)
            if (a[i][0]>a[i+1][0])
             {  k=a[i][0];
                a[i][0]=a[i+1][0];
                a[i+1][0]=k ;
}
            StringGrid2->Cells[j][0]=IntToStr(a[j][0]);
   }
}
//---------------------------------------------------------
---------------
void __fastcall TForm1::BitBtn3Click(TObject *Sender)
{
Close ();
}
//---------------------------------------------------------
---------------
```

Dastur matni kiritib bo‘lingandan so‘ng F9 tugmasi bosiladi va natijada quyidagi dastur ko‘rinishi paydo bo‘ladi:



7.6-rasm. *Natija oynasi.*

4-misol. 4 ta qator va 4 ta ustundan iborat massivni hosil qiling. Massivning 4*4 elemenlarini har xil sonlar bilan to'ldirish va elementlari yig'indisini hisoblash dasturi tuzilsin. Dastur vizual muhitda amalga oshirilsin.

Misolni vizual muhitda dasturlash uchun 1 ta Label, 1 ta StringGrid, 3 ta BitBtn komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

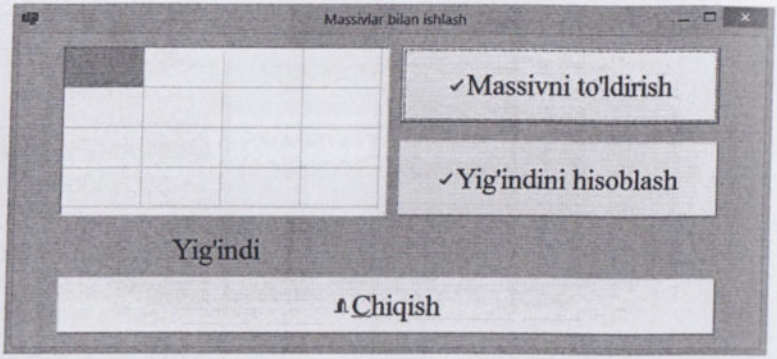
7.3-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Massivlar bilan ishlash" so'zi kiritiladi.
Label1	Caption (Properties)	"Yig'indi" so'zi kiritiladi.

BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Massivni to'ldirish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Yig'indini hisoblash" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.
StringGrid1	FixedCols (Properties)	0 sonini kiritamiz.
	FixedRows (Properties)	0 sonini kiritamiz.
	ColCount (Properties)	4 sonini kiritamiz.
	RowCount (Properties)	4 sonini kiritamiz.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



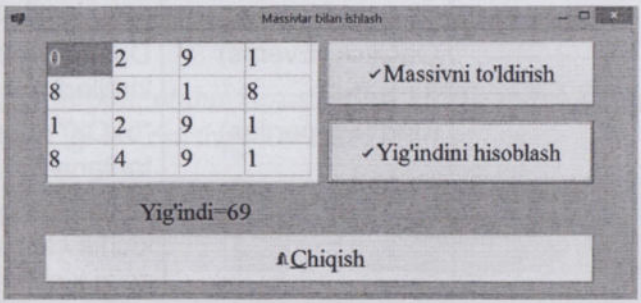
7.7-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int  sum=0;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
int i,j, a[4][4];
randomize();
for(i=0;i<4;i++)
```

```
for(j=0;j<4;j++)
{ a[i][j]=random(10);
StringGrid1->Cells[i][j]=IntToStr(a[i][j]);
sum+=a[i][j];
}
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
Label1->Caption="Yig'indi=" +IntToStr(sum);
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



7.8-rasm. *Natija oynasi.*

5-misol. Massiv elementlarini har xil sonlar bilan to'ldirish va massivning maksimal, minimal elementlari, shuningdek elementlar yig'indisini hisoblash dasturi tuzilsin. Dastur vizual muhitda amalga oshirilsin.

Misolni vizual muhitda dasturlash uchun 1 ta Label, 1 ta StringGrid, 4 ta BitBtn va 1 ta Edit komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

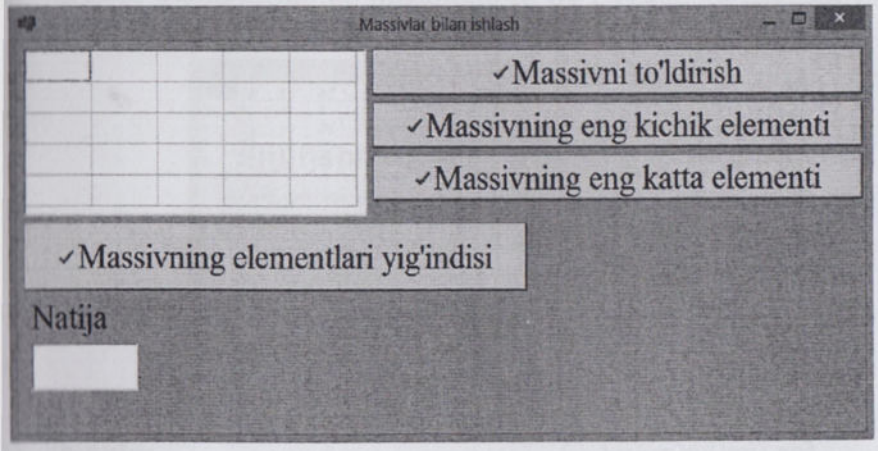
7.4-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Massivlar bilan ishlash" so'zi kiritiladi.
Label1	Caption (Properties)	"Natija" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Massivni to'ldirish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Massivning eng kichik elementi" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Massivning eng katta elementi" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

BitBtn4	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Massivning elementlari yig'indisi" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
StringGrid1	FixedCols (Properties)	0 sonini kiritamiz.
	FixedRows (Properties)	0 sonini kiritamiz.
	ColCount (Properties)	5 sonini kiritamiz.
	RowCount (Properties)	5 sonini kiritamiz.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



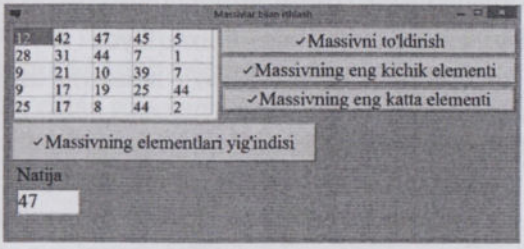
7.9-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
 int  a[5][5];
 int i,j,s,max,min;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
for (i=0;i<=4;i++)
  for (j=0;j<=4;j++)
  {
   a[i][j]=random(50);
   StringGrid1->Cells[i][j]=IntToStr(a[i][j]);
   }
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
min=a[0][0];
 for (i=0;i<=4;i++)
  for (j=0;j<=4;j++)
   if (a[i][j]<min) min=a[i][j];
```

```
 Edit1->Text=IntToStr(min);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn3Click(TObject *Sender)
{
max=a[0][0];
 for (i=0;i<=4;i++)
  for (j=0;j<=4;j++)
   if (a[i][j]>max) max=a[i][j];
 Edit1->Text=IntToStr(max);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn4Click(TObject *Sender)
{
s=0;
for (i=0;i<=4;i++)
  for (j=0;j<=4;j++)
   s=s+a[i][j];
 Edit1->Text=IntToStr(s);
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



7.10-rasm. *Natija oynasi.*

Nazorat savollari

1. Massivni ta'riflang va ulardan foydalanish zaruriyati nimadan kelib chiqadi?
2. Massiv elementlari va ular ustida bajariladigan amallar.

3. Bir va ko'p o'lchamli massivlar nima?

4. Massivlar xotiraga qaysi yo'llar bilan kiritiladi?

5. Vizual muhitda massivlar bilan ishlash qanday amalga oshiriladi?

Test savollari

1. Quyidagi massiv nechta elementdan iborat? Int a [10][15];

a) 45

b) 25

c) 35

d) 150

2. Massiv elementlarini xotiraga kiritishda qaysi operatordan foydalaniladi?

a) takrorlanish

b) tarmoqlanish

c) aralash toifa

d) shartli o'tish operatori

3. StringGrid komponentasining ColCount xossasi nimani bildiradi?

a) jadvaldagi ustunlar va qatorlar sonini aniqlaydi

b) jadvaldagi qatorlar sonini aniqlaydi

c) jadvaldagi ustunlar sonini aniqlaydi

d) jadvaldagi yacheykalar sonini aniqlaydi

4. To'g'ri ko'rsatilgan ko'p o'lchovli massivni tanglang.

a) int matrix[3] [5];

b) float a[1..4];

c) massiv Int [4 5];

d) massiv of [4] [5];

5. StringGrid komponentasining FixedRows xossasi nimani bildiradi?

a) jadvaldagi qatorlar sonini aniqlaydi

b) fiksirlangan ustunlar sonini aniqlaydi

c) fiksirlangan qatorlar sonini aniqlaydi

d) fiksirlashni amalga oshiradi

VIII bob. C++ DASTURLASH TILINING STRUKTURA TOIFASI

Tayanch so'zlar: strukturalar, struktura elementlari, strukturaga murojaat, aralash toifa, aralash toifa ko'rsatkichlari.

8.1. C++ tilida struktura toifasi va uni tavsiflash

Iqtisod va axborotni qayta ishlash masalalarini yechishda ma'lum turdagi hujjatlar, kataloglar, ro'yxatlar, ro'yxatlar ishlatiladi. Masalan, talabalarning anketa ma'lumotlari: familiyasi, ismi, otasining ismi, turar joyi, tug'ilgan yili, mutaxassisligi, guruh raqami va hokazo. Bu hollarda turli toifadagi ma'lumotlarni bir guruhga birlashtirish zaruriyati tug'iladi. Bizning misolimizda bu ma'lumotlarni talaba guruhiga birlashtirish mumkin. Ko'rinib turibdiki, bu guruhdagi ma'lumotlarning toifalari turlicha: familiya, ism – qator (char toifasi), tug'ilgan yili, guruh raqami – butun toifaga tegishli.

C++ tilida bunday ma'lumotlarni struktura (aralash) toifasi yordamida ifodalash imkoniyati berilgan. Struktura – bu ma'lumotlarni bir butun nomlangan elementlar to'plamiga birlashtirish. Struktura elementlari (maydonlar) har xil turda bo'lishi mumkin va ular har xil nomlarga ega bo'lishi kerak.

Struktura quyidagicha aniqlanadi:

struct {<ta'riflar ro'yxati >}

Strukturada bitta komponenta bo'lishi kerak. Struktura turidagi o'zgaruvchi quyidagicha ta'riflanadi:

<struktura_nomi><o'zgaruvchi>;

Dasturda struktura toifasi bir necha usullarda tavsiflanishi mumkin:

```
1 struct <toifa nomi>
    {
      <toifa>   <1 element>;
     <toifa>   <2 element>;
```

```
    . . .
};
```

Masalan:

```
struct Sana
{
int day;
int month;
int year;
        }; Sanatav;   // Sana toifasidagi o'zgaruvchi
```

2. struct

```
{
<toifa>   <1 element>;
<toifa>   <2 element>;
. . .
};<o'zgaruvchilar ro'yxati>;
```

Masalan: struct

```
{
  int min;
   int sec;
   int msec;
} vaqt;
```

3. Struktura toifasini typedef xizmatchi so'zi orqali ham tavsiflash mumkin.

Masalan: typedef struct

```
{
 float re;
 float im;
} Complex;
Complex a[100];
```

Elementlar ramziy nom bilan nomlanadi. Elementlar turli toifaga tegishli bo'lganligi uchun ularning toifasi tavsifi alohida-alohida beriladi. Element nomi va uning tavsifi keltirilgan qator struktura maydonini tashkil etadi. Demak, strukturalar bir nechta maydondan tashkil topgan bo'lishi mumkin.

Har bir obyektga nom berilgani kabi strukturaga va uning elementlariga ham ramziy nom beriladi.

8.2. Struktura elementi va ular ustida bajariladigan amallar

Bir strukturaga tegishli bo'lgan elementlar turlicha nomlanishi shart. Lekin turli strukturada bir xil nomli elementlar uchrashi mumkin. Chunki har bir elementga u tegishli bo'lgan yozuv nomi orqali murojaat qilinadi.

Dasturda struktura elementlari quyidagicha ifodalanadi:

<struktura nomi> < element nomi>

Masalan: stud.nomgr, stud.baho[2], stud.fam

Bu yerda, stud – struktura toifasidagi o'zgaruvchi nomi (aralash toifa nomi), nomgr, baho[2], fam – element nomlari.

Yuqorida ta'kidlab o'tilganidek, element toifasi turlicha bo'lishi mumkin. Element toifasi to'g'ridan to'g'ri yozuv ichida yoki toifalarni tafsivlash bo'limida aniqlangan bo'lishi mumkin. O'z navbatida element toifasi yozuvdan iborat bo'lishi mumkin. Bu holatda yozuvlar murakkab tuzilishni tashkil qiladi.

Struktura elementlari ustida aniqlangan toifadagi ma'lumotlar ustida bajarilishi mumkin bo'lgan amallarni bajarish mumkin.

Dasturda struktura toifasi elementlariga qiymatlarni kiritish oqimi cin yoki initializatsiyalash yordamida kiritish mumkin. Masalan:

```
struct talaba
{
char familiya [15];
int kurs;
float baho;
};
talaba s={"Abdullayev", 2, 5.0};
```

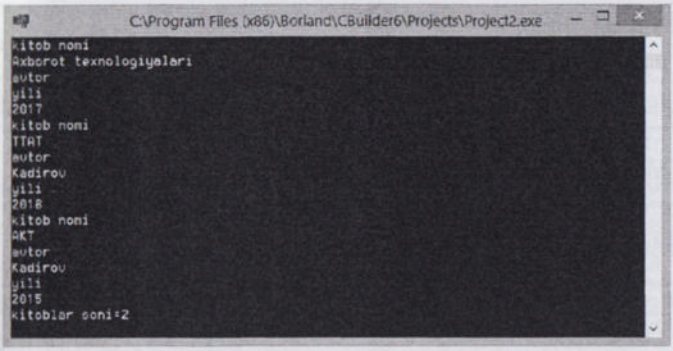
1-misol. Kitoblar haqida ma'lumotlar berilgan. Bu ma'lumotlar asosida 2017-yil yoki undan keyin nashr etilgan kitoblarni aniqlang.

Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
//-----------------------------------------------------------
```

```
#include <iostream.h>
#include <conio.h>
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
typedef struct
{char   title [40];
char   author [20];
int    entry;} book;
int sum=0;
book  k;
book b[10];
int  i;
for (i=1; i<=5; i++)
{
cout<<"kitob nomi"<<endl;
     cin>>b[i].title;
     cout<<"avtor"<<endl;
     cin>>b[i].author;
     cout<<"yili"<<endl;
     cin>>b[i].entry;
}
for (i=1; i<=3; i++)
    if (b[i].entry<=2017)  sum=sum+1;
    cout<<"kitoblar soni="<<sum;
     getch();
        return 0;
}
//---------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



8.1-rasm. *Natija oynasi.*

8.3. C++ dasturlash tilida aralash toifani qo'llash

Aralash toifa C++ dasturlash tilida keng qo'llaniladi. Aralash toifa elementlari har xil toifalardan tashkil topganligi sababli ro'yxatli ma'lumotlarni qayta ishlashda qulay hisoblanadi. Bunga misol qilib C++ Builderning vizual muhitda bir nechta dasturlarni ko'rib chiqamiz.

2-misol. Talabalarning familiyasi va nechinchi kursligidan iborat bo'lgan ro'yxat yaratilsin. Talabalar kursiga qarab saralash dasturi tuzilsin.

Misolni vizual muhitda dasturlash uchun 2 ta Label, 2 ta Edit, 7 ta BitBtn va 2 ta Memo komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

8.1-jadval.

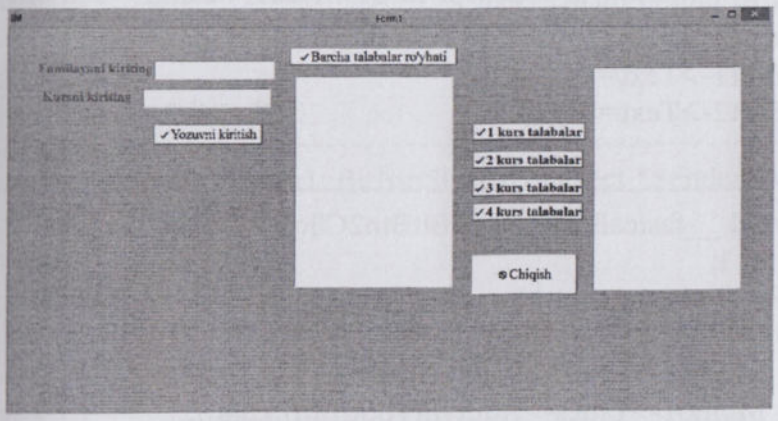
Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Label1	Caption (Properties)	"Familiyani kiriting" so'zi kiritiladi.

Label2	Caption (Properties)	"Kursni kiriting" so'zi kiritiladi.
Memo1	Lines (Properties)	"Memo1" so'zi tozalab qo'yiladi.
Memo2	Lines (Properties)	"Memo2" so'zi tozalab qo'yiladi.
Edit1	Text (Properties)	"Edit1" so'zi o'chirib tashlanadi.
Edit2	Text (Properties)	"Edit2" so'zi o'chirib tashlanadi.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Yozuvni kiritish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Barcha talabalar ro'yxati" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"1 kurs talabalari" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn4	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"2 kurs talabalari" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

BitBtn5	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"3 kurs talabalari" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn6	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"4 kurs talabalari" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn7	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



8.2-rasm. *Dastur ko'rinishi.*

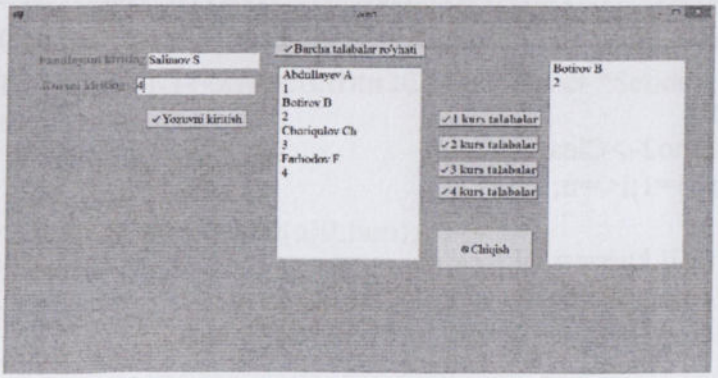
Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
TForm1 *Form1;
typedef struct
{String fam;
int    kurs;} talaba;
talaba a[10];
int n;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
n=n+1;
a[n].fam=Edit1->Text;
a[n].kurs=StrToInt(Edit2->Text);
Edit1->Text="";
Edit2->Text="" ;
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{int i;
for (i=1;i<=n; i++)
{
  Memo1->Lines->Add(a[i].fam);
  Memo1->Lines->Add(IntToStr(a[i].kurs));
}
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn3Click(TObject *Sender)
```

```
{
int i;
Memo2->Clear();
for (i=1;i<=n; i++)
{if (a[i].kurs==1){
Memo2->Lines->Add(a[i].fam);
Memo2->Lines->Add(IntToStr(a[i].kurs));}
}
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn4Click(TObject *Sender)
{
int i;
Memo2->Clear();
for (i=1;i<=n; i++)
{
if (a[i].kurs==2){
Memo2->Lines->Add(a[i].fam);
Memo2->Lines->Add(IntToStr(a[i].kurs));
}
}
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn5Click(TObject *Sender)
{
int i;
Memo2->Clear();
for (i=1;i<=n; i++)
{
if (a[i].kurs==3){
Memo2->Lines->Add(a[i].fam);
Memo2->Lines->Add(IntToStr(a[i].kurs));
}
}
```

```
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn6Click(TObject *Sender)
{
int i;
Memo2->Clear();
for (i=1;i<=n; i++)
{
if (a[i].kurs==4){
  Memo2->Lines->Add(a[i].fam);
  Memo2->Lines->Add(IntToStr(a[i].kurs));
}
}
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn7Click(TObject *Sender)
{
Close ();
}
//---------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



8.3-rasm. *Natija oynasi.*

3-misol. Abituriyentning familiyasi, ismi, javob varaqasi va bali berilgan. Olgan bali bo'yicha saralash dasturi tuzilsin.

Misolni vizual muhitda dasturlash uchun 4 ta Label, 4 ta Edit, 3 ta BitBtn va 2 ta Memo komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

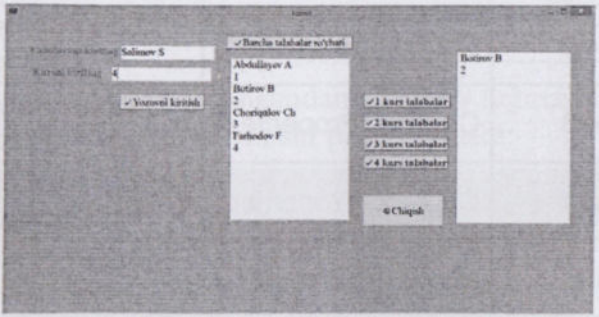
8.2-jadval.

Komponentalar xususiyatlarini kiritish

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Abituriyent" so'zi kiritiladi.
Label1	Caption (Properties)	"Abituriyent familiyasi" so'zi kiritiladi.
Label2	Caption (Properties)	"Abituriyent ismi" so'zi kiritiladi.
Label3	Caption (Properties)	"Javob varaqasi" so'zi kiritiladi.
Label4	Caption (Properties)	"To'plagan bali" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zi o'chirib tashlanadi.
Edit2	Text (Properties)	"Edit2" so'zi o'chirib tashlanadi.
Edit3	Text (Properties)	"Edit3" so'zi o'chirib tashlanadi.
Edit4	Text (Properties)	"Edit4" so'zi o'chirib tashlanadi.

Memo1	Lines (Properties)	"Memo1" so'zini o'chirib tashlaymiz.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Yozuvni kiritish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Saralash" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Kind (Properties)	"bkIgnore" xususiyati tanlanadi.
	Caption (Properties)	"Dastur xayr" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.



8.4-rasm. *Dastur ko'rinishi.*

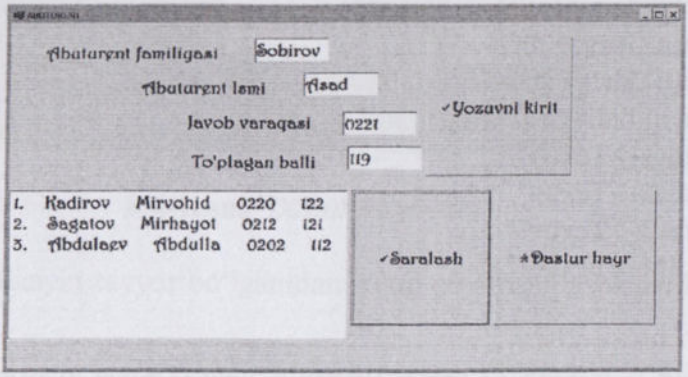
Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------------
#include <vcl.h>
```

```
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
typedef struct
{String fam;
String ism;
String jav;
int ball;} abituriyent;
abituriyent ab [10];
int i,j,m; abituriyent k; int n;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
n=n+1;
ab[n].fam=Edit1->Text;
ab[n].ism=Edit2->Text;
ab[n].jav=Edit3->Text;
ab[n].ball=StrToInt(Edit4->Text);
Edit1->Text="";
Edit2->Text="" ;
Edit3->Text="" ;
Edit4->Text="" ;
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
```

```
 Memo1->Clear();
 for (i=1;i<=n-1; i++)
 for (j=i+1;j<=n;j++)
  {
 if  (ab[i].ball<ab[j].ball)
   {
k=ab[i];
 ab[i]=ab[j];
 ab[j]=k;
}
}
    for (i=1;i<=n; i++)
    {
 Memo1->Lines->Add(IntToStr(i) + ". " + ab[i].fam + " " +
ab[i].ism + "  " + ab[i].jav + "  " + IntToStr(ab[i].ball));
}
}
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn3Click(TObject *Sender)
{
Close ();
}
//---------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



8.5-rasm. *Natija oynasi.*

4-misol. Talabalar haqida quyidagi ma'lumotlar berilgan: Familiyasi, ismi, axborot texnoligiyalari, fizika, matematika fanidan to'plagan ballari.

Talabalarning to'plagan ballari asosida a'lochi talabalarni saralash dasturi tuzilsin.

Misolni vizual muhitda dasturlash uchun 5 ta Label, 2 ta StringGrid, 4 ta BitBtn va 5 ta Edit komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

7.3-jadval.

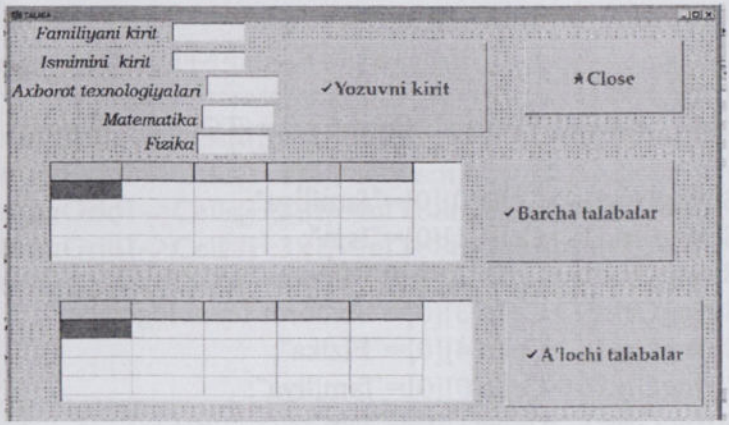
Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Talaba" so'zi kiritiladi.
	OnCreate (Events)	Dastur matni kiritiladi.
	OnCloseQuery (Events)	Dastur matni kiritiladi.
Label1	Caption (Properties)	"Familiyani kirit" so'zi kiritiladi.
Label2	Caption (Properties)	"Ismini kirit" so'zi kiritiladi.
Label3	Caption (Properties)	"Axborot texnologiyalari" so'zi kiritiladi.
Label4	Caption (Properties)	"Matematika" so'zi kiritiladi.
Label5	Caption (Properties)	"Fizika" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zi o'chirib tashlanadi.

Edit2	Text (Properties)	“Edit2” so‘zi o‘chirib tashlanadi.
Edit3	Text (Properties)	“Edit3” so‘zi o‘chirib tashlanadi.
Edit4	Text (Properties)	“Edit4” so‘zi o‘chirib tashlanadi.
Edit5	Text (Properties)	“Edit5” so‘zi o‘chirib tashlanadi.
BitBtn1	Kind (Properties)	“bkOK” xususiyati tanlanadi.
	Caption (Properties)	“Yozuvni kirit” so‘zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	“bkOK” xususiyati tanlanadi.
	Caption (Properties)	“Barcha talabalar” so‘zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Kind (Properties)	“bkOK” xususiyati tanlanadi.
	Caption (Properties)	“A’lochi talabalar” so‘zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn4	Kind (Properties)	“bkClose” xususiyati tanlanadi.
	Caption (Properties)	“Chiqish” so‘zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

StringGrid1	FixedCols (Properties)	0 sonini kiritamiz.
	FixedRows (Properties)	1 sonini kiritamiz.
	ColCount (Properties)	5 sonini kiritamiz.
	RowCount (Properties)	5 sonini kiritamiz.
StringGrid2	FixedCols (Properties)	0 sonini kiritamiz.
	FixedRows (Properties)	1 sonini kiritamiz.
	ColCount (Properties)	5 sonini kiritamiz.
	RowCount (Properties)	5 sonini kiritamiz.

Mavjud komponentalar xususiyatlari kiritilgandan so‘ng dastur dizayni quyidagi ko‘rinishga keladi:



8.6-rasm. *Dastur ko‘rinishi.*

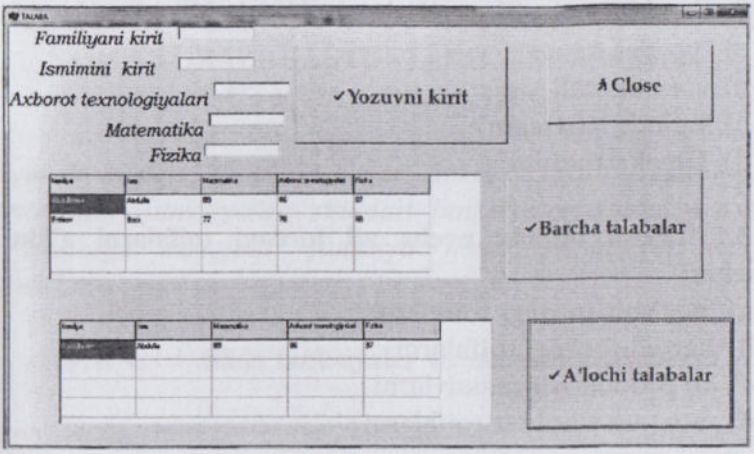
Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
typedef struct {
        String fam;
        String ism;
        int axborot;
        int mat;
        int fiz; }talaba ;
talaba a[30];
 int i,j,n;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 StringGrid1->Cells[0][0]="familiya";
 StringGrid1->Cells[1][0]="Ism";
 StringGrid1->Cells[2][0]="Matematika";
 StringGrid1->Cells[3][0]="Axborot texnologiyalari";
 StringGrid1->Cells[4][0]="Fizika";
 StringGrid2->Cells[0][0]="familiya";
 StringGrid2->Cells[1][0]="Ism";
 StringGrid2->Cells[2][0]="Matematika";
```

```
 StringGrid2->Cells[3][0]=" Axborot texnologiyalari";
 StringGrid2->Cells[4][0]="Fizika";
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click (TObject *Sender)
{
n=n+1;
a[n].fam=Edit1->Text;
a[n].ism=Edit2->Text;
a[n].axborot=StrToInt(Edit3->Text);
a[n].mat=StrToInt(Edit4->Text);
a[n].fiz=StrToInt(Edit5->Text);
Edit1->Text="";
Edit2->Text="" ;
Edit3->Text="" ;
Edit4->Text="" ;
Edit5->Text="" ;
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
j=0;
for (i=1;i<=n; i++)
 {
StringGrid1->Cells[j][i]=a[i].fam;
 StringGrid1->Cells[j+1][i]=a[i].ism;
 StringGrid1->Cells[j+2][i]=IntToStr(a[i].axborot);
 StringGrid1->Cells[j+3][i]=IntToStr(a[i].mat);
 StringGrid1->Cells[j+4][i]=IntToStr(a[i].fiz);
}
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn3Click(TObject *Sender)
{
```

```
j=0;
for (i=1;i<=n; i++)
  if (a[i].axborot>85 && a[i].mat>85 && a[i].fiz>85)
 {
StringGrid2->Cells[j][i]=a[i].fam;
 StringGrid2->Cells[j+1][i]=a[i].ism;
 StringGrid2->Cells[j+2][i]=IntToStr(a[i].axborot);
 StringGrid2->Cells[j+3][i]=IntToStr(a[i].mat);
 StringGrid2->Cells[j+4][i]=IntToStr(a[i].fiz);
}
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn4Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------
void__fastcall TForm1::FormCloseQuery (TObject *Sender,
bool &CanClose)
{
if(MessageDlg("Dasturdan chiqasizmi", mtInformation,
TMsgDlgButtons() <<mbYes <<mbNo,0)==mrYes)
{
CanClose=true;
}
else
{
CanClose=false;
}
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



8.7-rasm. *Natija oynasi.*

Nazorat savollari

1. Ma'lumotlarning struktura toifasidan dasturda foydalanish zaruriyati nimadan kelib chiqadi?
2. Aralash toifani dasturda tavsiflash usullari qanday?
3. Struktura elementi, uning toifasi qanday bo'lishi mumkin?
4. Struktura elementi ustida bajariladigan amallar?
5. Aralash toifadagi ma'lumotlarni kiritish qanday amalga oshiriladi?

Test savollari

1. C++ da struktura deb –

a) turli toifadagi ma'lumotlarning cheklangan to'plamiga aytiladi.

b) funksiyalarni o'z ichiga oluvchi bir xil nomlangan ma'lumotlarga aytiladi.

c) nomlangan, bir turdagi ma'lumot elementlari ketma-ketligiga aytiladi.

d) formada o'rnatilgan komponenta xususiyatlarini boshqaradi.

2. Struktura tavsifi qaysi xizmatchi so‘z bilan boshlanadi?

a) typedef struct
b) void fastcall
c) include <fstream>
d) Typef structura

3. Aralash toifada necha xil turdagi toifalarni e’lon qilish mumkin?

a) faqat bir turdagi toifalarni
b) har xil turdagi toifalarni
c) ko‘p o‘lchovli massivlarni
d) faqat int va float toifalarni

4. Struktura toifasi (aralash toifa) qaysi qatorda to‘g‘ri ko‘rsatilgan?

a) {float re;float im;}Complex;Complex a[100];
b) typedef struct {float re;float im;} Complex;
c) structure {float re; float im;} talaba;
d) talaba {float re;float im;} typedef struct;

5. Struct {String fam;int fiz; }talaba; nimani bildiradi?

a) funksiyani (ko‘p o‘zgaruvchili)
b) massivni (ko‘p o‘chamli)
c) struktura (aralash toifani)
d) protsedurani (talaba nomli)

IX bob. C++ DA FUNKSIYA VA PROTSEDURALAR. STRUKTURALI DASTURLASH USULLARINI C++ DA QO'LLASH

Tayanch so'zlar: qism dasturlar, lokal o'zgaruvchilar, global o'zgaruvchilar, funksiyalar, standart funksiyalar, funksiya toifasi, qism dasturga murojaat, funksiya nomi, rasmiy va haqiqiy parametrlar.

9.1. C++ tilida funksiyalar va ulardan foydalanish

Amaliyotda shunday masalalar ham uchraydiki, ularni yechish jarayonida birgina funksiyani hisoblash emas, balki bir nechta amallar ketma-ketligini o'zgaruvchi kattalikning turli qiymatlarida bir necha marotaba takrorlashga to'g'ri keladi.

Bunday masalalarni kompyuterda yechish dasturini ixchamlashtirish maqsadida takrorlanayotgan buyruqlarni dasturda bir marotaba yozib, kerak bo'lganda unga istalgancha murojaat qilish mumkin.

Dasturning turli joyidan murojaat qilinishi natijasida ko'p marotaba bajarilishi mumkin bo'lgan operatorlarning to'plami "protsedura" (qism dastur), protsedura joylashgan dastur "asosiy dastur" deb yuritiladi.

C++ tilida qism dasturlarni ikki xil ko'rinishda tashkil qilish mumkin: protsedura va funksiya ko'rinishida. Bu ko'rinishdagi qism dasturlar asosiy dastur tarkibida joylashgan bo'lib, ular ham o'z sarlavhasiga va tarkibiga (operatorlar to'plamiga) ega bo'ladi.

Funksiya – bu mantiqan to'g'ri tugatilgan dasturiy qismdir. Ular yordamida katta va murakkab hisoblashlarni qayta-qayta yozish mashaqqatidan xalos bo'linadi va dastur bajarilishi yengillashadi. Uni bir marta tashkil etib yozib qo'yiladi va unga dasturning istalgan yeridan murojaat qilish mumkin bo'ladi. Funksiyani tashkil qilishda funksiyaning toifasi, uning nomi va tashkil etuvchi parametrlari haqida axborot keltiriladi. Bu parametrlar rasmiy

parametrlar deb yuritiladi, ularning qiymati funksiyaga murojaat qilish vaqtida aniqlanadi.

Funksiya umumiy ko'rinishda quyidagicha yoziladi:

funksiya toifasi funksiya nomi (rasmiy parametrlar)

```
{
    funksiya tanasi;
}
```

Funksiya nomi ixtiyoriy lotincha so'z bo'lishi mumkin, rasmiy parametrlar – ixtiyoriy o'zgaruvchilardir.

Funksiyaga murojaat qilishdan oldin rasmiy parametrlar o'rniga keladigan haqiqiy parametrlar aniqlanishi lozim. Unga murojaat qilish quyidagicha bo'ladi:

o'zgaruvchi = funksiya nomi (haqiqiy parametrlar);

Rasmiy va haqiqiy parametrlar soni, ularning toifasi va kelish tartibi albatta bir-biriga mos bo'lishi lozim. Rasmiy va haqiqiy parametrlar nomlari bir xil bo'lishi mumkin. Funksiyani bosh funksiya ichida e'lon qilinganida haqiqiy parametrlar nomlarini ko'rsatmasdan, faqat ularning toifalarini keltirish ham mumkin, masalan: float max (float, float);

Asosiy dasturda ham funksiyada ishlatish mumkin bo'lgan o'zgaruvchilar global o'zgaruvchilar deyiladi. Global o'zgaruvchilar asosiy dasturda e'lon qilinishi shart. Faqat funksiyaning ichida ishlatish mumkin bo'lgan o'zgaruvchilarga lokal o'zgaruvchilar deyiladi. Ular funksiyada e'lon qilinadi.

Funksiyalar main () funksiyasidan avval ham, keyin ham aniqlanishi mumkin. Agar bosh funksiyadan avval aniqlangan bo'lsa, uni main () funksiyasi ichida alohida e'lon qilish shart emas, agar bosh funksiyadan keyin keladigan bo'lsa, uni main () funksiyasi ichida albatta e'lon qilish kerak. Masalan, quyidagi dasturda "funk" nomli funksiya main bosh funksiyasidan keyin e'lon qilingan va aniqlangan:

```
#include <vcl.h>
#pragma hdrstop
#include <math.h>
```

```
# include <iostream.h>
# include <conio.h>
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{ int k, n, funk (int n);
  cin >>n;
  k=funk(n);
  cout << "k="<<k<<endl;
  getch( );
}
 int funk(int a)   // funksiyani aniqlash
 { int c;
    c=1+sin(a)+cos(a);
    return c; // funksiyaga natijani qaytarish
}
```

Yuqorida keltirilgan funksiya tanasini quyidagicha ham yozish mumkin:

```
int funk(int a)
{
return 1+sin(a)+cos(a);
 }
```

Funksiya bosh funksiyadan oldin e'lon qilinsa dastur quyidagi ko'rinishda yozilishi mumkin:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <math.h>
# include <iostream.h>
# include <conio.h>
//---------------------------------------------------------
#pragma argsused
 int funk (int a)
  {
```

```
 return 1+sin(a)+cos(a);
}
int main(int argc, char* argv[])
{ int k, n ;
 cin>>n;
 k=funk(n);
 cout << "k="<<k<<endl;
getch( );     return 0;
}
//---------------------------------------------------------
```

Demak, funksiyalarga murojaat qilish jarayonida amallar quyidagi tartibda bajariladi:

1. Funksiya bajarilayotganda rasmiy parametrlar uchun xotiradan joy ajratiladi, ya'ni ular funksiyaning ichki parametrlariga aylantiriladi. Bunda parametr toifasi o'zgartiriladi: float toifasi double toifasiga, char va shortint toifalari int toifasiga aylantiriladi.
2. Haqiqiy parametrlar qiymatlari beriladi yoki hisoblanadi.
3. Haqiqiy parametrlar rasmiy parametrlar uchun ajratilgan xotira qismiga yoziladi.
4. Funksiya tanasi ichki parametrlar yordamida bajariladi va qiymat qaytarish joyiga yuboriladi.
5. Funksiyadan chiqishda rasmiy parametrlar uchun ajratilgan xotira qismi bo'shatiladi.

1-misol. Ikkita ixtiyoriy sonlar ichidan kattasini topish uchun funksiya tashkil qilinsin.

Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
//-------------------------------------------------------------
--------------
#include <vcl.h>
#pragma hdrstop
#include <math.h>
# include <iostream.h>
# include <conio.h>
//-------------------------------------------------------------
```

```
#pragma argsused
int main(int argc, char* argv[])
{
float x=20, b=40, c, max(float x, float y);
    c = max(x, b);
    cout << "c="<<c<<endl;
     getch( );
}
  float max(float x, float y)
    {
if  (x > y)   return x;
              else   return y;
        return 0;
    }
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



9.1-rasm. *Natija oynasi.*

Ushbu misolda max funksiyasini e'lon qilishda rasmiy parametrlar ko'rsatilmagan, funksiya tanasida 2 ta return operatori ishlatilgan.

9.2. Protseduralar va ularni tashkil etish

Ba'zi masalalarda funksiya bilan ishlaganda funksiya tanasi ichida haqiqiy parametrlar qiymatlarini o'zgartirish zaruriyati

tug'iladi, ya'ni natija bir emas, balki bir nechta bo'lishi kerak bo'ladi. Bunday jarayonni protsedura ko'rinishidagi funksiyalar yordamida amalga oshirish maqsadga muvofiq. Funksiyani aniqlashtirishda rasmiy parametrlar bilan bir satrda natijalar nomlari ham ko'rsatiladi. Shuning uchun protseduralar bilan ishlaganda funksiya toifasini bo'sh (void) deb olish maqsadga muvofiqdir, return operatorini ishlatmasa ham bo'ladi.

Misol sifatida quyidagi dastur kodini keltirish mumkin:

```
double FSum(double X1,double X2, int A);
void SPrint(AnsiString S);
void F1(void);
void SPrint(AnsiString S)
{
if (S != "")
ShowMessage(S);
}
```

Dasturlashda qo'llashda qulay bo'lib, uning nomiga murojaat etishning o'zi yetarli. Masalan:

```
void showmes() {
ShowMessage("Funksiya va proseduralarni qo'llash");
}
void __fastcall TForm1::Button1Click(TObject *Sender)
{showmes;
}
//---------------------------------------------------------
```

2-misol. "Hisobla" nomli qism dastur yaratilsin. Qism dastur yordamida berilgan sonlarni aniqlash va o'rta arifmetigini hisoblash dasturi tuzilsin. Dastur vizual muhitda amalga oshirilsin.

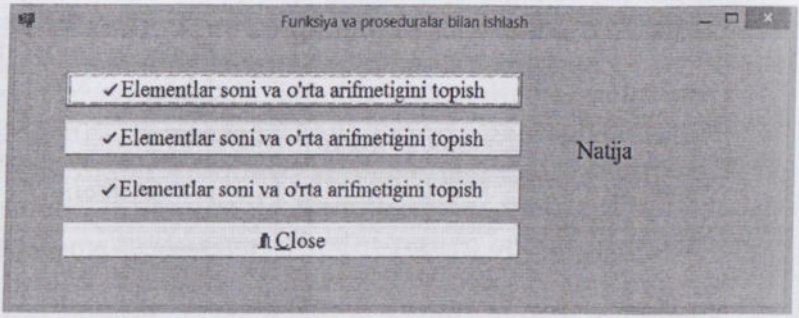
Misolni vizual muhitda dasturlash uchun 1 ta Label, 4 ta BitBtn komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

9.1-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Funksiya va proseduralar bilan ishlash" so'zi kiritiladi.
	FormCloseQuery (Events)	Dastur matni kiritiladi.
Label1	Caption (Properties)	"Natija" so'zi kiritiladi.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Elementlar soni va o'rta arifmetigini topish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Elementlar soni va o'rta arifmetigini topish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Elementlar soni va o'rta arifmetigini topish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn4	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.



9.2-rasm. *Dastur ko'rinishi.*

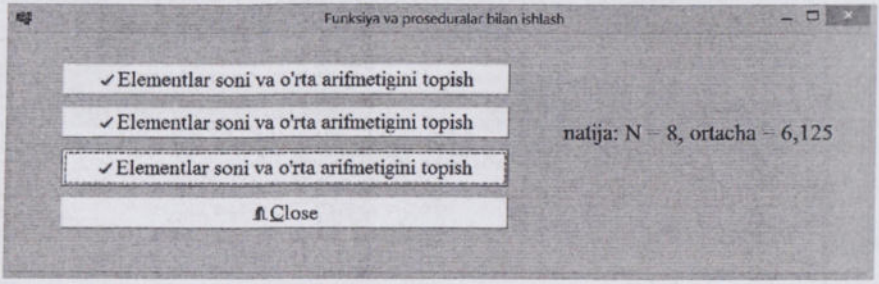
Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
void Hisobla(AnsiString mess, int N,...)
{
double A = 0;
va_list ap;
va_start(ap, N);
for(int i = 0; i < N; i++)
A += va_arg(ap,int);
Form1->Label1->Caption = mess + "N = " +IntToStr(N) +
", ortacha = " + FloatTo Str(A/N);
va_end(ap);
}
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
```

```
        : TForm(Owner)
    {
    }
    //---------------------------------------------------------
    void __fastcall TForm1::FormCloseQuery(TObject *Sender,
bool & CanClose)
    {
    if(MessageDlg("Dasturdan chiqasmi", mtInformation,
TMsgDlgButtons() <<mbYes <<mbNo,0)==mrYes)
    {CanClose=true;
    }
    else{
    CanClose=false;
    }
    }
    //---------------------------------------------------------
    void __fastcall TForm1::BitBtn1Click(TObject *Sender)
    {
    Hisobla ("natija: ",5,4,2,3,5,4);
    }
    //---------------------------------------------------------
    void __fastcall TForm1::BitBtn2Click(TObject *Sender)
    {
    Hisobla ("natija: ",6,4,2,3,5,4,10);
    }
    //---------------------------------------------------------
    void __fastcall TForm1::BitBtn3Click(TObject *Sender)
    {
    Hisobla ("natija: ",8,4,2,3,5,4,8,11,12);
    }
    //---------------------------------------------------------
    void __fastcall TForm1::BitBtn4Click(TObject *Sender)
    {
    Close();
```

```
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



9.3-rasm. *Natija oynasi.*

9.3. Funksiyalarda massivlarni ishlatish

Funksiyalarning rasmiy parametrlari sifatida massivlarni ham ishlatish mumkin. Bunda massiv toifasi, nomi va [ ] qavslar beriladi, bundan tashqari massiv o'lchami ham ko'rsatilishi mumkin. Masalan:

```
void mas(int a[], int n);
```

Dasturda esa, funksiya chaqirilganda, massivning faqat ismi beriladi xolos, [] qavslarning keragi yo'q:

```
int num=0;
int first[6][9], second[6][9], third[6][9];
void Mass(int zero[6][9])
{
   for(int i=0;i<6;i++)
     for(int j=0;j<9;j++)
       zero[i][j]=num;
   num++;
}
```

Funksiyaga murojaat quyidagicha amalga oshiriladi:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Mass(first);
Mass(second);
Mass(third);
}
```

Funksiyaga massivlarni uzatganimizda, massivdagi elementlar sonini berish muammolardan biridir. Bu holda optimal variantlardan biri bu massiv kattaligini qo'shimcha rasmiy parametr orqali funksiyada e'lon qilishdir.

3-misol. Quyidagi funksiyani hisoblang:

$$y = \frac{\sqrt{AXT^2 + 1.3}}{b * e^{-x}} * \sin t$$

Bu yerda, $T = 2X^2$; ($t = 2X^2$)

X: $X^2 + 2X + 4 = 0$ tenglamani [1; 2] oraliqda xatolik bilan taqribiy ildizi iteratsiya usuli bilan topilsin:

$$c = \int_{0.6}^{1.5} \frac{dx}{\sqrt{x^2 + 1}}, n = 16$$

Integralni Simpson usuli bilan hisoblang.

A: Max $\{a_i\}$, $i \in 1 \div 20$; $a_i \in [1 \div 100]$, A(20)

B: Max $\{b_i\}$ $i \in 1 \div 20$; $b_i \in [1 \div 50]$, B(20)

A va B massiv a'zolarining eng katta qiymati qism dastur yordamida hisoblansin.

Misolni vizual muhitda dasturlash uchun 14 ta Label, 2 ta Button, 2 ta StringGrid va 1 ta image komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

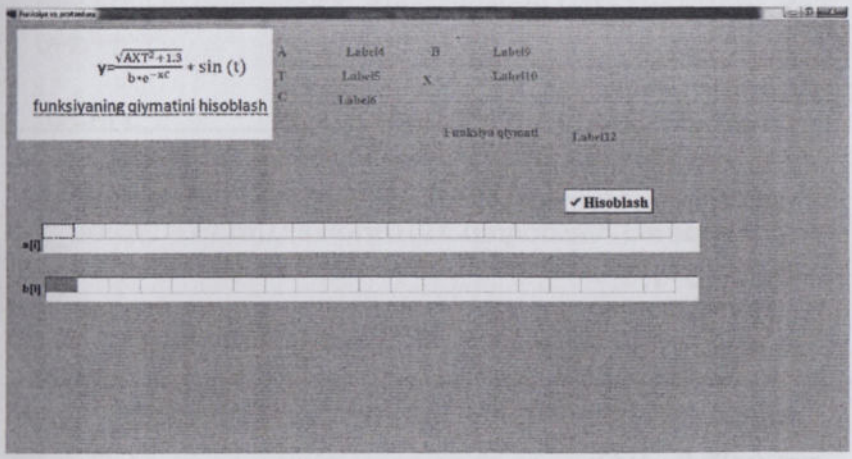
9.2-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Funksiya va protsedura" so'zi kiritiladi.
Label1	Caption (Properties)	"A ning qiymati" so'zi kiritiladi.
Label2	Caption (Properties)	"T ning qiymati" so'zi kiritiladi.
Label3	Caption (Properties)	"C ning qiymati" so'zi kiritiladi.
Label4	Caption (Properties)	"Label 4" so'zi qoladi.
Label5	Caption (Properties)	"Label 5" so'zi qoladi.
Label6	Caption (Properties)	"Label 6" so'zi qoladi.
Label7	Caption (Properties)	"B ning qiymati" so'zi kiritiladi.
Label8	Caption (Properties)	"X ning qiymati" so'zi kiritiladi.
Label9	Caption (Properties)	"Label 9" so'zi qoladi.
Label10	Caption (Properties)	"Label 10" so'zi qoladi.
Label11	Caption (Properties)	"Funksiyaning qiymati" so'zi kiritiladi.
Label12	Caption (Properties)	"Label 12" so'zi qoladi.
Label13	Caption (Properties)	"a[i]=" so'zi kiritiladi.
Label14	Caption (Properties)	"b[i]=" so'zi kiritiladi.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Hisoblash" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

StringGrid1	FixedCols (Properties)	0 sonini kiritamiz.
	FixedRows (Properties)	0 sonini kiritamiz.
	ColCount (Properties)	20 sonini kiritamiz.
	RowCount (Properties)	1 sonini kiritamiz.
StringGrid2	FixedCols (Properties)	0 sonini kiritamiz.
	FixedRows (Properties)	0 sonini kiritamiz.
	ColCount (Properties)	20 sonini kiritamiz.
	RowCount (Properties)	1 sonini kiritamiz.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



9.4-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

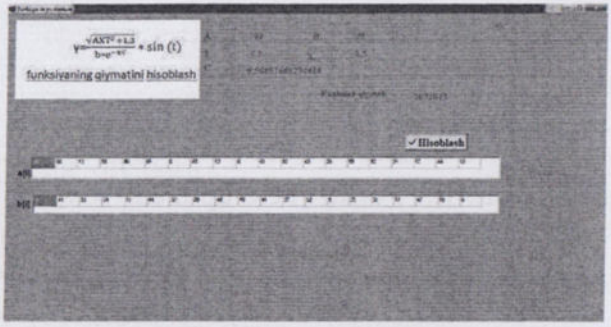
```
//---------------------------------------------------------
  #include <iostream.h>
```

```
#include <math.h>
#include <stdio.h>
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int  A_ning_qiymati(int );
int A_ning_qiymati(int a[20])
{
int max;
max=a[0];
for(int i=1; i<20; i++)
if(a[i]>max) max=a[i];
return max;
}
//---------------------------------------------------------
float X_ning_qiymati();
float X_ning_qiymati()
{
float y,y0=1;
y=-(y0*y0+4)/2;
if(fabs(y-y0)>0.0004)
{
y0=y;
y=-(y0*y0+4)/2;
}
return 1.5;
}
//---------------------------------------------------------
float T_ning_qiymati();
float T_ning_qiymati()
```

```
{
float t;
t=2*pow(X_ning_qiymati(),2);
return t;
}
//---------------------------------------------------------
float C_ning_qiymati();
float C_ning_qiymati()
{
float c=0,a,b,h;
a=1.5;
b=0.6;
h=(a-b)/16;
for(int i=0;i<16; i++)
{
c=c+(b+h)/sqrt(pow(X_ning_qiymati(),2)+1);
b=b+h;
}
return c;
}
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{ float y;
int a[20], b[20];
for(int i=0; i<20; i++)
{
a[i]=random(100);
StringGrid1->Cells[i][0]=a[i];
b[i]=random(50);
```

```
    StringGrid2->Cells[i][0]=b[i];
    }
    Label4->Caption=FloatToStr(A_ning_qiymati(a));
    Label9->Caption=FloatToStr(A_ning_qiymati(b));
    Label5->Caption=FloatToStr(T_ning_qiymati());
    Label10->Caption=FloatToStr(X_ning_qiymati());
    Label6->Caption=FloatToStr(C_ning_qiymati());
    y=sqrt(A_ning_qiymati(a)*X_ning_qiymati()*T_ning_
qiymati()*T_ning_qiymati()+1.3)*sin(T_ning_qiymati())/
(A_ning_qiymati(b)*pow(2.7,X_ning_qiymati ()* C_ning_
qiymati()));
    Label12->Caption=FloatToStr(y);
    }
    //---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



9.5-rasm. *Natija oynasi.*

Nazorat savollari

1. C++ tilida qism dastur tushunchasi va ulardan foydalanish.
2. Funksiyaning dasturdagi o'rni.
3. Funksiyalarga qanday murojaat qilinadi?
4. Haqiqiy parametrlar va ulardan foydalanish.
5. Funksiya va uning xususiyatlari.

Test savollari

1. C++ tilida funksiya bu –

a) komponentalarni o'z ichida saqlovchi qism
b) dasturning bosh qismi hisoblanadi
c) matematik funksiyalarni o'z ichiga olgan qism
d) mantiqan to'g'ri tugatilgan dasturiy qismdir

2. Funksiya to'g'ri ko'rsatilgan qatorni tanglang.

a) function (float, int, char, string);
b) float max(float , float);
c) Max int (matn);
d) Funktion max(int);

3. Funksiya bir marta tashkil etib yozib qo'yiladi va unga dasturning istalgan yeridan murojaat qilish mumkin bo'ladi. Murojaat qilish qanday amalga oshiriladi?

a) funksiyaning toifasi orqali
b) soxta parametrlari orqali
c) funksiyaning nomi orqali
d) haqiqiy parametrlari orqali

4. Funksiyaning qanday parametrlari mavjud bo'ladi?

a) Software va hardware
b) Rasmiy va haqiqiy
c) Max int (matn);
d) Nomi va toifasi

5. Funksiyada massivlar qaysi ko'rinishda e'lon qilinadi?

a) abc int [2];
b) typedef struct matrix(int a[11], int n);
c) a int [2][3];
d) void mas(int a[4], int n);

X bob. C++ DASTURLASH TIZIMIDA MA'LUMOTLARNING FAYLLI VA MUROJAAT TOIFASI

Tayanch so'zlar: fayl, fayl toifasi, fayl ko'rsatkichi, fayl elementi, dinamik xotira, matnli fayllar, binar fayllar, standart oqimlar, kiritish-chiqarish oqimlari, murojaat toifa, ko'rsatkich funksiyalari.

10.1. Ma'lumotlarning faylli toifasi

Shu vaqtgacha biz C++ dasturiy tilida kattaliklarning bir nechta toifalari bilan tanishdik. Bular skalyar, oddiy va murakkab – tarkiblashgan toifalardir. Bu toifadagi ma'lumotlar yordamida masalalarni yechishda boshlang'ich ma'lumotlar klaviaturadan operativ xotiraga kiritiladi va natija ekranga chiqariladi. Ulardan boshqa dasturlarda foydalanib bo'lmaydi, chunki ular tizimidan chiqilgandan so'ng hech qayerda saqlanmaydi, bundan tashqari ular oldindan aniqlangan miqdordagi chekli elementlardan iborat bo'ladi. Hatto dinamik aniqlanganda ham, operativ xotiraning amalda cheklanganligi sababli, bu berilganlar miqdori yuqoridan chegaralangan elementlardan iborat bo'ladi. Juda ko'p tadbiqiy masalalar esa berilganlarni doimiy ravishda saqlab turishni talab qiladi. Masalan, muassasa xodimlari haqidagi ma'lumotlarni doimiy ravishda saqlab turish zarur. Bu ma'lumotlarni xotirada saqlash uchun C++ tilida ma'lumotlarning faylli toifasi belgilangan. Fayl toifasi alohida o'rin egallaydi. Fayl toifasi bilan ishlashda ma'lum tushunchalarni o'zlashtirish talab qilinadi.

Fayl – ma'lumotlarni saqlab qo'yish uchun, tashqi xotiraning nomlangan qismi. Bunday fayllar fizik fayllar deyiladi.

Yana bir muhim tushunchalardan biri fayl ko'rsatkichi tushunchasidir. Fayl ko'rsatkichi – ayni paytda fayldan o'qilayotgan yoki unga yozilayotgan joyni (yozuv o'rnini) ko'rsatib turadi, ya'ni fayl ko'rsatkichi ko'rsatib turgan joydan bitta yozuvni o'qish yoki shu joyga yangi yozuvni joylashtirish mumkin.

Fayl yozuvlariga murojaat ketma-ket ravishda amalga oshiriladi: n-yozuvga murojaat qilish uchun n-1 yozuvni o'qish zarur bo'ladi. Shuni ta'kidlab o'tish zarurki, fayldan yozuvlarni o'qish jarayoni qisman "avtomatlashgan", unda i-yozuvni o'qilgandan keyin, ko'rsatkich navbatdagi i+1 yozuv boshiga ko'rsatib turadi va shu tarzda o'qishni davom ettirish mumkin (massivlardagidek indeksni oshirish shart emas). Fayl bu berilganlarni saqlash joyidir va shu sababli uning yozuvlari ustida to'g'ridan to'g'ri amal bajarib bo'lmaydi. Fayl yozuvi ustida amal bajarish uchun yozuv qiymati operativ xotiraga mos turdagi o'zgaruvchiga o'qilishi kerak. Keyinchalik, zarur amallar shu o'zgaruvchi ustida bajariladi va kerak bo'lsa natijalar yana faylga yozilishi mumkin.

C++ tilida mantiqiy fayl tushunchasi bo'lib, u fayl turidagi o'zgaruvchini anglatadi. Fayl turidagi o'zgaruvchlarga boshqa turdagi o'zgaruvchilar kabi qiymat berish operatori orqali qiymat berib bo'lmaydi. Boshqacha aytganda fayl turidagi o'zgaruvchilar ustida hech qanday amal aniqlanmagan. Ular ustida bajariladigan barcha amallar funksiyalar vositasida bajariladi.

Fayllar bilan ishlash quyidagi bosqichlarni o'z ichiga oladi:

— fayl o'zgaruvchisi albatta diskdagi fayl bilan bog'lanadi;

— fayl ochiladi;

— fayl ustida yozish yoki o'qish amallari bajariladi;

— fayl yopiladi;

— fayl nomini o'zgartirish yoki faylni diskdan o'chirish amallarini bajarilishi mumkin.

Matn va binar fayllar. C++ tili C tilidan o'qish-yozish amalini bajaruvchi standart funksiyalar kutubxonasini vorislik bo'yicha olgan. Bu funksiyalar <stdio.h> sarlavha faylida e'lon qilingan. O'qish-yozish amallari fayllar bilan bajariladi. Fayl matn yoki binar (ikkilik) bo'lishi mumkin.

Matnli fayl – ASCII kodidagi belgilar majmuasi. Belgilar ketma-ketligi satrlarga bo'lingan bo'ladi va satrning tugash alomati sifatida CR ('\r') va LF ('\n') belgilar juftligi hisoblanadi.

Matnli fayldan berilganlarni o'qishda bu belgilar juftligi bitta CR belgisi bilan almashtiriladi va aksincha, yozishda CR belgisi ikkita CR va LF belgilariga almashtiriladi. Fayl oxiri #26 (^Z) belgisi bilan belgilanadi.

Matnli faylga boshqacha ta'rif berish ham mumkin. Agar faylni matn tahririda ekranga chiqarish va o'qish mumkin bo'lsa, bu matnli fayldir. Boshqacha aytganda dastur tomonidan ekranga chiqariladigan barcha ma'lumotlarni "stdout" nomidagi matn fayliga chiqarilmoqda deb qarash mumkin. Xuddi shunday klaviaturadan o'qilayotgan har qanday berilganlarni matn faylidan o'qilmoqda deb hisoblanadi.

Matnli fayllarning komponentalari satrlar deb nomlanadi. Satrlar uzluksiz joylashib, turli uzunlikda va bo'sh bo'lishi mumkin.

Binar fayllar – bu oddiygina baytlar ketma-ketligi. Odatda binar fayllardan berilganlarni foydalanuvchi tomonidan bevosita "ko'rish" zarur bo'lmagan hollarda ishlatiladi. Binar fayllardan o'qish-yozishda baytlar ustida hech qanday konvertatsiya amallari bajarilmaydi.

10.2. Fayldan o'qish-yozish funksiyalari

Fayl oqimi bilan o'qish-yozish amalini bajarish uchun fayl oqimini ochish zarur. Bu amal prototipi quyidagicha:

FILE * fopen(const char* filename, const char *mode);

Bu jarayon *fopen()* funksiyasi orqali amalga oshiriladi. Funksiya filename nomi bilan faylni ochadi, u bilan oqimni bog'laydi va oqimni identifikatsiya qiluvchi ko'rsatkichni javob tariqasida qaytaradi. Faylni ochish muvaffaqiyatsiz bo'lganligini *fopen()* funksiyasining nol qiymatli javobi bildiradi.

Parametrlar ro'yxatidagi ikkinchi – mode parametri faylni ochish rejimini aniqlaydi. U qabul qilishi mumkin bo'lgan qiymatlar 10.1-jadvalda keltirilgan.

10.1-jadval.

Fayl ochish rejimlari

Mode qiymati	Fayl ochilish holati tavsifi
R	Fayl faqat o'qish uchun ochiladi
W	Fayl yozish uchun ochiladi. Agar bunday fayl mavjud bo'lsa, u qaytadan yoziladi (yangilanadi).
A	Faylga yozuvni qo'shish rejimi. Agar fayl mavjud bo'lsa, fayl uning oxiriga yozuvni yozish uchun ochiladi, aks holda yangi fayl yaratiladi va yozish rejimida ochiladi.
r+	Mavjud fayl o'zgartirish (o'qish va yozish) uchun ochiladi.
w+	Yangi fayl yaratilib, o'zgartirish (o'qish va yozish) uchun ochiladi. Agar fayl mavjud bo'lsa, undagi oldingi yozuvlar o'chiriladi va u qayta yozishga tayyorlanadi.
a+	Faylga yozuvni qo'shish rejimi. Agar fayl mavjud bo'lsa, uning oxiriga (EOF alomatidan keyin) yozuvni yozish (o'qish) uchun ochiladi, aks holda yangi fayl yaratiladi va yozish rejimida ochiladi.

Matn fayli ochilayotganligini bildirish uchun fayl ochilish rejimi satriga 't' belgisini qo'shib yozish zarur bo'ladi. Masalan, matn fayl o'zgartirish (o'qish va yozish) uchun ochilayotganligini bildirish uchun "rt+" satri yozish kerak bo'ladi. Xuddi shunday binar fayllar ustida ishlash uchun 'b belgisini ishlatish kerak. Misol uchun fayl ochilishining "wb+" rejimi binar fayl yangilanishini bildiradi.

Fayl o'zgartirish (o'qish-yozish) uchun ochilganda, berilganlarni oqimdan o'qish hamda oqimga yozish mumkin. Biroq yozish amalidan keyin darhol o'qib bo'lmaydi, buning uchun o'qish amalidan oldin *fseek()* yoki *rewind()* funksiyalari chaqirilishi shart.

Faraz qilaylik "C:\Users\admin\Desktop\Talaba\guruh.txt" nomli matn faylni o'qish uchun ochish zarur bo'lsin. Bu talab quyidagi ifodani yozish orqali amalga oshiraladi.

FILE*f=fopen("C:\Users\admin\Desktop\Talaba\guruh.txt", "r+");

Natijada diskda mavjud bo'lgan fayl dasturda "f" o'zgaruvchisi nomi bilan aynan bir narsa deb tushuniladi. Boshqacha aytganda, dasturda keyinchalik "f" ustida bajarilgan barcha amallar, diskdagi "guruh.txt" fayli ustida ro'y beradi.

Fayl oqimi bilan ishlash tugagandan keyin u albatta yopilishi kerak. Buning uchun *fclose()* funksiyasidan foydalaniladi. Funksiya prototipi quyidagi ko'rinishga ega:

int fclose(FILE * stream);

fclose() – funksiyasi oqim bilan bog'liq buferlarni tozalaydi (masalan, faylga yozish ko'rsatmalari berilishi natijasida buferda yig'ilgan berilganlarni diskdagi faylga ko'chiradi) va faylni yopadi. Agar faylni yopish xatolikka olib kelsa, funksiya EOF qiymatini, normal holatda 0 qiymatini qaytaradi.

Quyida fayldan ma'lumotlarni o'qish va faylga yozish amallari uchun funksiyalarni qarab chiqamiz:

1. *fgetc()* – funksiyasi quyidagi ko'rinishga ega:

int fgetc(FILE *stream);

Funksiyasi prototipi yuqoridagi ko'rinishida aniqlangan bo'lib, fayl oqimidan belgini o'qishni amalga oshiradi. Agar o'qish muvafffaqiyatli bo'lsa, funksiya o'qilgan belgini int turidagi ishorasiz butun songa aylantiradi. Agar fayl oxirini o'qishga harakat qilinsa yoki xatolik ro'y bersa, funksiya EOF qiymatini qaytaradi.

Ko'rinib turibdiki, *getc()* va *fgetc()* funksiyalari deyarli bir xil ishni bajaradi, farqi shundaki, *getc()* funksiyasi belgini standart oqimdan o'qiydi. Boshqacha aytganda, *getc()* funksiyasi, fayl oqimi standart qurilma bo'lgan *fgetc()* funksiyasi bilan aniqlangan makrosdir.

2. *fputc()* – funksiyasi quyidagi ko'rinishga ega:

int fputc(int c, FILE *stream);

fputc() funksiyasi fayl oqimiga argumentda ko'rsatilgan belgini yozadi (chiqaradi) va u amal qilishida putc() funksiyasi bilan bir xil.

3. Fayl oqimidan satrni o'qish uchun quyidagi funksiyadan foydalaniladi:

char * fgets(char * s, int n, FILE *stream)

fgets() funksiyasi fayl oqimidan belgilar ketma-ketligini "s" satriga o'qiydi. Funksiya o'qish jarayonini oqimdan n-1 belgi o'qilgandan keyin yoki keyingi satrga o'tish belgisi ('\n') uchraganda to'xtatadi. Oxirgi holda '\n' belgisi ham "s" satrga qo'shiladi. Belgilarni o'qish tugagandan keyin "s" satr oxiriga, satr tugash alomati '\0' belgisi qo'shiladi. Agar satrni o'qish muvaffaqiyatli bo'lsa, funksiya s argument ko'rsatadigan satrni qaytaradi, aks holda nol qiymat qabul qiladi.

4. Fayl oqimiga satrni *fputs()* funksiyasi yordamida chiqarish mumkin. Bu funksiya quyidagi ko'rinishda aniqlangan.

int fputs (const char *s, FILE *stream);

Satr oxiridagi yangi satrga o'tish belgisi oqimga chiqarilmaydi. Oqimga chiqarish muvaffaqiyatli bo'lsa, funksiya nomanfiy son qaytaradi, aks holda EOF amalga oshiriladi.

5. *foef()* – funksiyasi aslida makros bo'lib, fayl ustida o'qish-yozish amallari bajarilayotganda fayl oxiri belgisi uchragan yoki yo'qligini bildiradi. Funksiya quyidagi ko'rinishda amalga oshiriladi:

int feof(FILE *stream);

U fayl oxiri belgisi uchrasa, noldan farqli sonni qaytaradi, boshqa holatlarda nol qiymatini qaytaradi.

10.3. Fayl obyektlarini kiritish-chiqarish

C++ tilida kiritish-chiqarish oqimlarining sinflari mavjud bo'lib, ular kiritish-chiqarish standart kutubxonasining obyektga

mo'ljallangan ekvivalentidir. Ular quyidagi direktivalardan iborat:

- istream – kiritish oqimi;
- ostream – chiqarish oqimi
- iostream – kiritish/chiqarish oqimi

Satrli oqimlar xotirada joylashtirilgan satrli buferlardan ma'lumotlarni kiritish-chiqarish uchun quyidagi xizmatchi so'zlar xizmat qiladi:

- ifstream – faylli kiritish
- ofstream – faylli chiqarish
- fstream – faylli kiritish/chiqarish

Odatda bu oqimlar include <.....> deriktivalarida yoziladi.

Ifstream, ofstream va fstream oqimlari dasturda fayllar hosil qilish, ulardagi ma'lumotlardan foydalanish uchun ishlatiladi. Ularning qo'llanilishi quyidagicha:

ofstream name ("path\file_name");

Bu funksiya yordamida ma'lumotli fayl hosil qilish, ya'ni ma'lumotlar bazasi uchun ochish mumkin. Bu yerda name – ixtiyoriy nom (lotincha) – ya'ni oqim nomi. Keyinchalik fayldagi ma'lumotlarni yozish yoki o'qish uchun shu nomdan foydalaniladi.

Masalan, ofstream mk ("C:\Users\admin\Desktop\fayl.txt");

"fayl.txt" biz hosil qilgan fayl nomi bo'lib, ular oqim nomlari bilan bog'langandir.

Endi hosil bo'lgan ma'lumotlardan foydalanish uchun uni ochishni ko'ramiz. Buning uchun quyidagi operatorni kiritish kerak:

ifstream mk ("C:\Users\admin\Desktop\fayl.txt");

Ochilgan fayllarni albatta yopish kerak. Bu jarayonni quyidagicha amalga oshiriladi:

name.close();

Masalan, mk.close( );

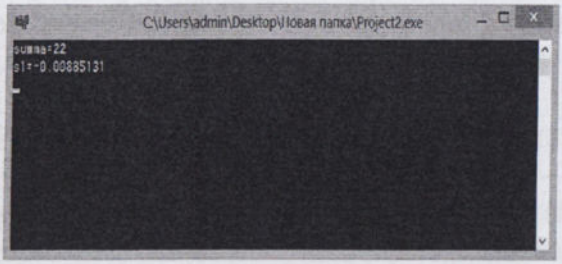
Demak, mk bilan fayl.txt nomlari o'zaro ma'lumot almashinuvini ta'minlaydi.

1-misol. 2 ta butun sonlarning yig'indisini o'zida saqlovchi fayl hosil qilinsin va olingan natijadan foydalanuvchi dastur tuzilsin.

Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
//---------------------------------------------------------
#include <vcl.h>
# include <iostream.h>
# include <fstream.h>
# include <conio.h>
# include <math.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{int a=12, b=10;
int summa;
float s1;
ofstream  mk ("hujjat.txt");
summa = a + b;
cout <<"summa=" << summa << endl;
mk << summa <<endl;
mk.close ( );
ifstream  mk1 ("hujjat.txt");
mk1 >>summa;
s1 = sin (summa);
cout <<"s1="<<s1<<endl;
mk1.close ( );
getch ( );
     return 0;
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



10.1-rasm. *Natija oynasi.*

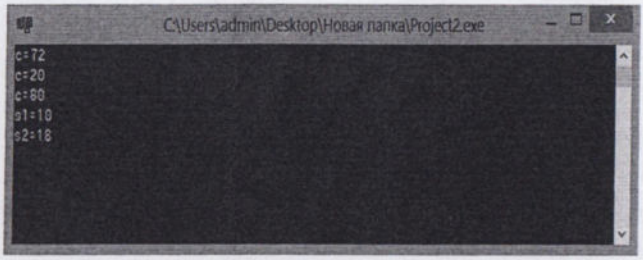
2-misol. Matrisa va vektorlar berilgan. Son qiymatlari ixtiyoriy. Ushbu qiymatlardan foydalanib, matrisani vektorga ko'paytirish, matrisaning izini hisoblash va vektorning yig'indisini hisoblash dasturini tuzing.

Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
//---------------------------------------------------------
#include <vcl.h>
# include <iostream.h>
# include <fstream.h>
# include <conio.h>
# include <math.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
int  a [3][3], b[3], c[3];
int i, j;
int s1=0, s2=0;
ofstream said ("mk.txt");
for ( i=0; i<3; i++)
{
for (j=0; j<3; j++)
{
a[i][j] = random (9);
```

```
said <<a[i][j]<<endl;
}
}
for (i=0; i<3; i++)
{
b[i] = random (11);
said << b[i]<<endl;
}
said.close ( );
ifstream said1 ("mk.txt");
for ( i=0; i<3; i++)
for (j=0; j<3; j++)
said1 >>a[i][j];
for ( i=0; i<3; i++)
said1 >> b[i];
for ( i=0; i<3; i++)
{
c[i] = 0;
for (j=0; j<3; j++)
c[i] = c[i] + a[i][j] * b[i];
cout << "c="<<c[i]<<endl;
}
for ( i=0; i<3; i++)
s1 = s1 + a[i][i];
for ( i=0; i<3; i++)
s2=s2 + b[i];
cout << "s1="<<s1<<endl;
cout <<"s2="<<s2<<endl;
getch ( );
return 0;
}
//-------------------------------------------------------------
```

Dastur matni kiritib bo‘lingandan so‘ng F9 tugmasi bosiladi va natijada quyidagi dastur ko‘rinishi paydo bo‘ladi:



10.2-rasm. *Natija oynasi.*

3-misol. Matnli ma'lumotlarni tashqi xotiraga yozuvchi dastur tuzilsin. Misolni vizual muhitda dasturlash uchun 1 ta Memo, 2 ta BitBtn komponentalari kerak bo'ladi.

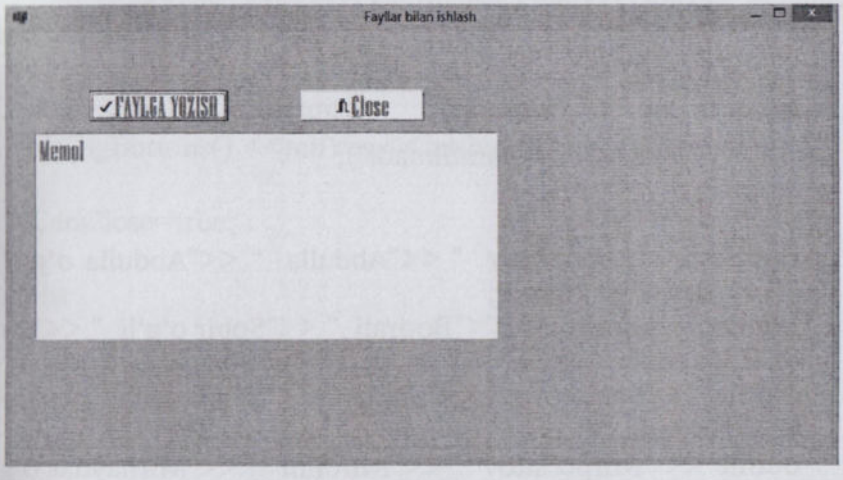
Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

10.2-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Fayllar bilan ishlash" so'zi kiritiladi.
	FormCloseQuery (Events)	Dastur matni kiritiladi.
Memo1	Lines (Properties)	"Memo1" so'zi kiritiladi.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Faylga yozish" so'zi kiritiladi.
	On click (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so‘ng dastur dizayni quyidagi ko‘rinishga keladi:



10.3-rasm. Dastur ko‘rinishi.

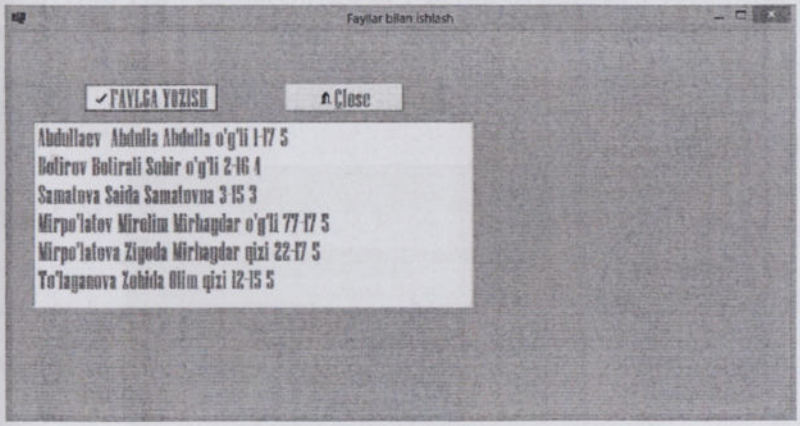
Dastur dizayni tayyor bo‘lganidan so‘ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#include <fstream.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
```

```
//---------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
char sin[100];
ofstream outfile("Test.txt");
if(!outfile)
{
ShowMessage("Fayl yaratilmadi");
return;
}
outfile << "Abdullayev " <<"Abdulla " <<"Abdulla o'g'li "
<<"1-17 " <<"5" <<endl;
outfile << "Botirov " <<"Botirali " <<"Sobir o'g'li " <<"2-16
" <<"4" <<endl;
outfile << "Samatova " <<"Saida " <<"Samatovna " <<"3-15
" <<"3" <<endl;
outfile << "Mirpo'latov " <<"Mirolim " <<"Mirhaydar o'g'li
" <<"77-17" <<"5 " <<endl;
outfile << "Mirpo'latova " <<"Ziyoda " <<"Mirhaydar qizi "
<<"22-17" <<"5 " <<endl;
outfile << "To'laganova " <<"Zohida " <<"Olim qizi "
<<"12-15 " <<"5" <<endl;
outfile.close();
ifstream infile("Test.txt");
if(!infile)
{
ShowMessage("Fayl ochilmadi");
return;
}
  Memo1->Clear();
while (!infile.eof())
{
infile.getline(sin,100);
Memo1->Lines->Add(AnsiString(sin));
}
 infile.close();
```

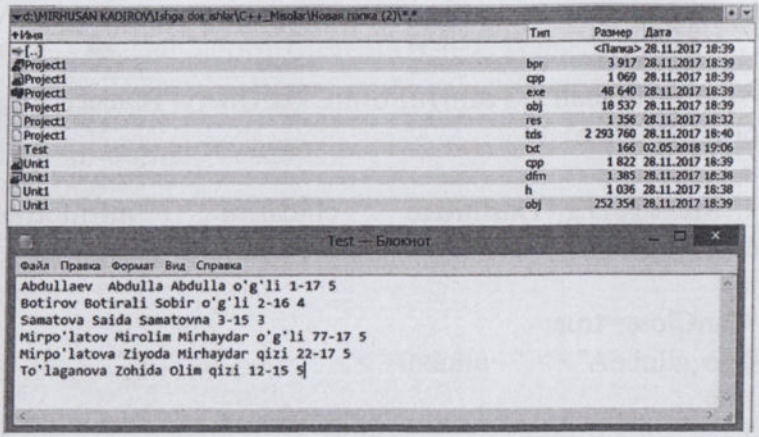
```
}
//---------------------------------------------------------
void __fastcall TForm1::FormCloseQuery(TObject *Sender,
bool &CanClose)
{
if(MessageDlg("Dasturdan chiqasmi", mtInformation,
TMsgDlgButtons() <<mbYes <<mbNo,0)==mrYes)
{
CanClose=true;
}
else
{
CanClose=false;
}
}
//---------------------------------------------------------
```

Dastur matni kiritib bo‘lingandan so‘ng F9 tugmasi bosiladi va natijada quyidagi dastur ko‘rinishi paydo bo‘ladi:



10.4-rasm. *Natija oynasi.*

Dastur bajarilishi natijasida tashqi xotiraga “test.txt” nomli fayl yaratiladi va quyidagi ko‘rinishda bo‘ladi:



10.5-rasm. *Natija oynasi.*

4-misol. Matnli fayllarni kiritish va ularni qayta ishlovchi matn muharriri tuzilsin. Dastur fayllarni saqlash, saqlangan fayllarni ochish, bosmaga chiqarish kabi amallarni bajarsin.

Misolni vizual muhitda dasturlash uchun 1 ta Memo, 1 ta MainMenu, 1 ta OpenDialog, 1 ta SaveDialog, 1 ta FontDialog, 1 ta PrintDialog, 1 ta FindDialog va 1 ta StatusBar komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

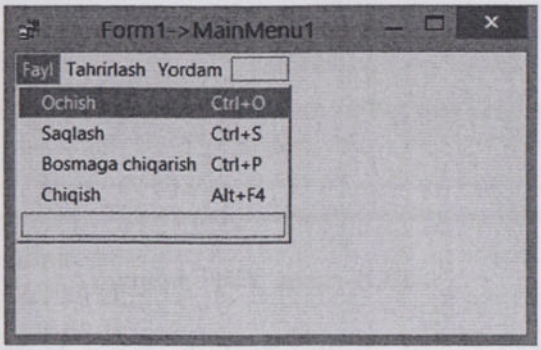
10.3-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Bloknot Uz" so'zi kiritiladi.
	FormCloseQuery (Events)	Dastur matni kiritiladi.

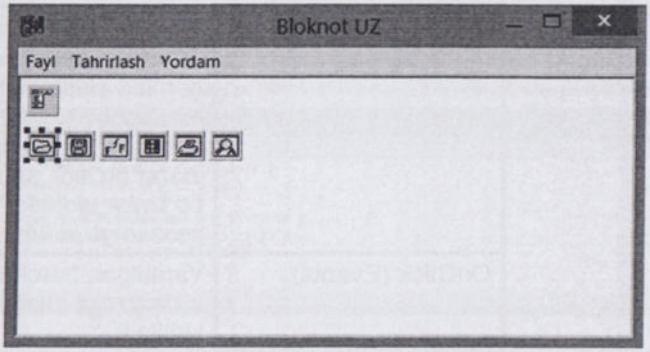
Memo1	Lines (Properties)	"Memo1" so'zi o'chirib tashlanadi.
	Align (Properties)	"alClient" tanlanadi.
MainMenu	Items (Properties)	"Fayl", "Tahrirlash", "Yordam" asosiy menyu nomlari kiritiladi. Har bir asosiy menyu bir nechta bo'limlardan iborat bo'ladi, shu sababli bo'limlar alohida "Caption" xossasiga kiritib chiqiladi.
	OnClick (Events)	Yaratilgan barcha bo'limlarga dastur kodi kiritiladi.
OpenDialog	Filter (Properties)	"Filter Name" xossasiga "Matnli fayllar (*.txt) so'zi va "Filter" xossasiga "*.txt" so'zi kiritiladi.
SaveDialog	Filter (Properties)	"Filter Name" xossasiga "Matnli fayllar (*.txt) so'zi va "Filter" xossasiga "*.txt" so'zi kiritiladi.

Dasturning asosiy menyusini yaratish darchasi quyidagi ko'rinishda bo'ladi:



10.6-rasm. *Dasturning asosiy menyusini yaratish darchasi.*

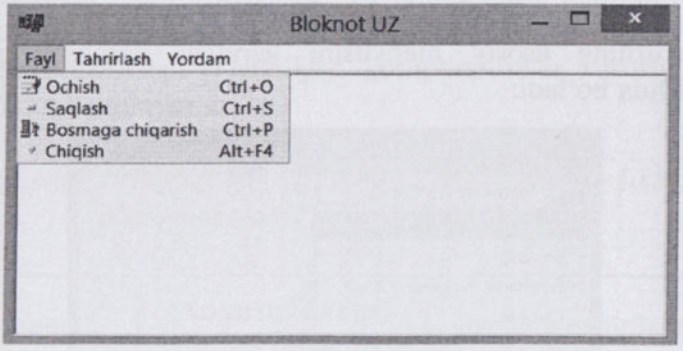
Yuqoridagi darcha yordamida barcha asosiy menyu va ularning bo'limlari yozib chiqiladi. Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



10.7-rasm. *Dastur ko'rinishi.*

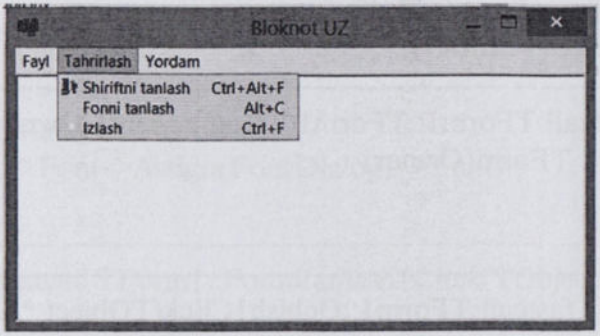
Asosiy menyular qatori uch qismdan iborat bo'lib, ular fayl, tahrirlash va yordamdan iborat.

Fayl menyusi ochish, saqlash, bosmaga chiqarish va chiqish bo'limlarini o'z ichiga olgan.



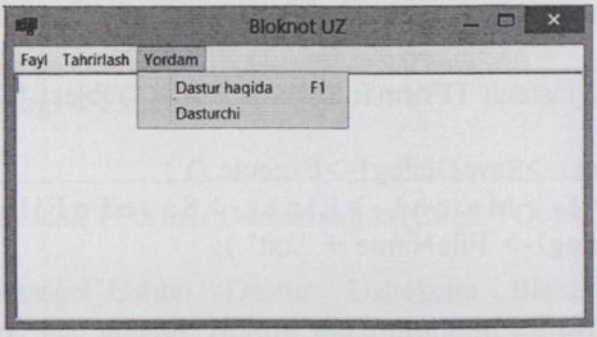
10.8-rasm. *Fayl menyusi.*

Tahrirlash menyusi shriftni tanlash, fonni tanlash va izlash bo'limlarini o'z ichiga olgan.



10.9-rasm. *Tahrirlash menyusi.*

Yordam menyusi dastur haqida va dasturchi bo'limlarini o'z ichiga olgan.



10.10-rasm. *Yordam menyusi.*

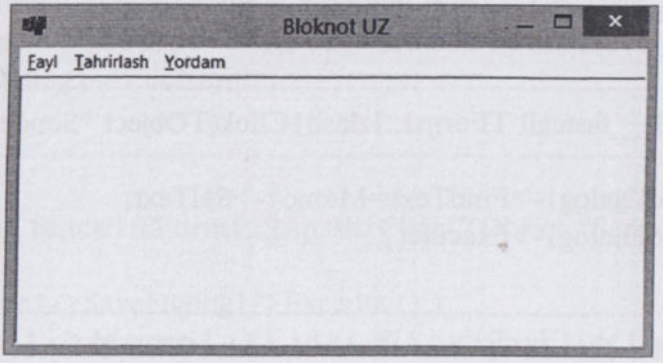
Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
//---------------------------------------------------------
#pragma package(smart_init)
```

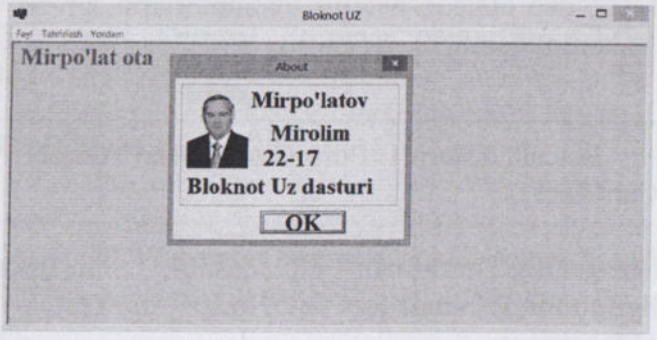
```
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::Ochish1Click(TObject *Sender)
{
if (Form1->OpenDialog1->Execute ())
Form1->Memo1->Lines->LoadFromFile(Form1->OpenDialog1->FileName);
}
//---------------------------------------------------------

void __fastcall TForm1::Saqlash1Click(TObject *Sender)
{
if(Form1->SaveDialog1->Execute () )
Form1->Memo1->Lines->SaveToFile(Form1->SaveDialog1-> FileName + ".txt" );
}
//---------------------------------------------------------
void __fastcall TForm1::Bosmaga chiqarish1 Click(TObject *Sender)
{
PrintDialog1->Execute() ;
}
//---------------------------------------------------------
void __fastcall TForm1::Chiqish1Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------
```

```
void __fastcall TForm1::Shriftni tanlash1Click(TObject
*Sender)
{
if (FontDialog1->Execute())
Memo1->Font->Assign(FontDialog1->Font) ;
}
//---------------------------------------------------------
void __fastcall TForm1::Fonnitanlash1Click(TObject *Sender)
{
if (ColorDialog1->Execute ())
Memo1->Color=ColorDialog1->Color;
}
//---------------------------------------------------------
void __fastcall TForm1::Izlash1Click(TObject *Sender)
{
FindDialog1->FindText=Memo1->SelText;
FindDialog1->Execute() ;
}
//---------------------------------------------------------
void __fastcall TForm1::Dasturhaqida1Click(TObject *Sender)
{
ShowMessage("Ushbu Dastur Uzbekcha Bloknot bo'lib
Bloknot UZ deb ataladi. Matnli ma'lumotlarni kiritsh va qayta
ishlash imkonini beradi");
}
//---------------------------------------------------------
void __fastcall TForm1::FormCloseQuery(TObject *Sender,
bool &CanClose)
{
if(MessageDlg("Dasturdan chiqasmi", mtInformation,
TMsgDlgButtons() <<mbYes <<mbNo,0)==mrYes)
{CanClose=true;
}
else
```

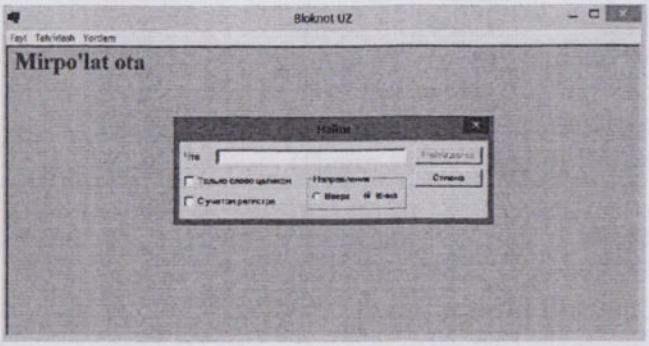
```
{
CanClose=false;
}
}
//---------------------------------------------------------------
void __fastcall TForm1::Dasturchi1Click(TObject *Sender)
{
AboutBox->ShowModal();
}
//---------------------------------------------------------------
```



10.11-rasm. *Natija oynasi.*



10.12-rasm. *Yordam menyusining dasturchi bo'limi.*



10.13-rasm. *Tahrirlash menyusining izlash bo'limi.*

10.4. C++ da ko'rsatkichlar bilan ishlash

O'zining qiymati sifatida xotira manzilini ko'rsatuvchi (saqlovchi) o'zgaruvchilarga – ko'rsatkich o'zgaruvchilar deyiladi.

Masalan, ko'rsatkichning qiymati:

1) 0x22ff40

2) 0x22ff33

3) yuqorida ko'rsatilgan kabi xotiraning aniq qismi bo'lishi mumkin.

Boshqa o'zgaruvchilar kabi, ko'rsatkichlardan foydalanish uchun ularni e'lon qilish, toifasini aniqlash shart. Ko'rsatkich quyidagicha beriladi:

int *countPtr, count;

Bu yerda, countPtr – int toifasidagi obyektga ko'rsatkich, count esa oddiy butun (int) toifasidagi o'zgaruvchi. Ko'rsatkichlarni e'lon qilishda har bir o'zgaruvchi oldiga * belgisi qo'yilishi shart.

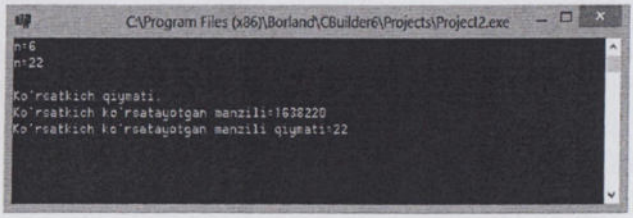
5-misol. Ko'rsatkich ko'rsatayotgan manzil qiymatini topish dasturi tuzilsin.

Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
//---------------------------------------------------------
#include <vcl.h>
#include <iostream.h>
```

```
#include <conio.h>
#pragma hdrstop
//---------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
int n = 6;
int * nPtr;
// & manzilini olish amali
nPtr = &n;
cout << "n=" << n << endl;
*nPtr = 22;
cout << "n=" << n << endl;
cout << "\n Ko'rsatkich qiymati,\n";
cout << "Ko'rsatkich ko'rsatayotgan manzili=" << nPtr<<endl;
cout << "Ko'rsatkich ko'rsatayotgan manzili qiymati="
<<*nPtr<<endl;
getch();
return 0;
}
//---------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



10.14-rasm. *Natija oynasi.*

Ushbu dasturni vizual muhitda amalga oshiramiz. Buning uchun 3 ta Label, 3 ta Edit va 2 ta BitBtn komponentalari kerak bo'ladi.

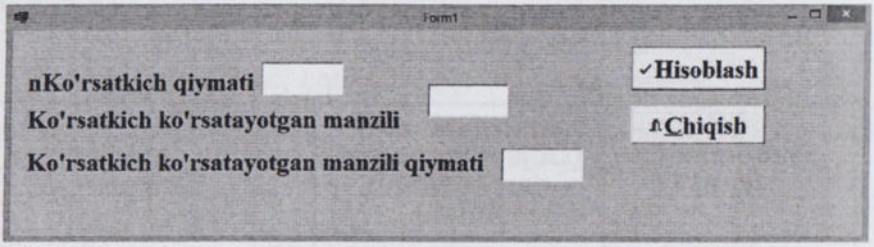
Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

10.4-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Ko'rsatkichlar bilan ishlash" so'zi kiritiladi.
Label1	Caption (Properties)	"nKo'rsatkich qiymati" so'zi kiritiladi.
Label2	Caption (Properties)	"Ko'rsatkich ko'rsatayotgan manzili=" so'zi kiritiladi.
Label3	Caption (Properties)	"Ko'rsatkich ko'rsatayotgan manzili qiymati=" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zi o'chirib tashlanadi.
Edit2	Text (Properties)	"Edit2" so'zi o'chirib tashlanadi.
Edit3	Text (Properties)	"Edit3" so'zi o'chirib tashlanadi.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Hisoblash" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so‘ng dastur dizayni quyidagi ko‘rinishga keladi:

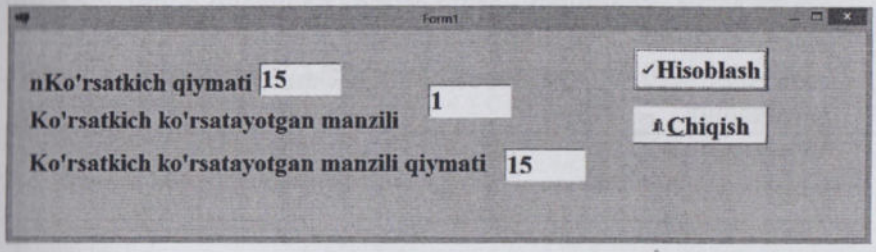


10.15-rasm. *Dastur ko‘rinishi.*

Dastur dizayni tayyor bo‘lganidan so‘ng quyidagi dastur matni kiritiladi:

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
int n = 5;
int * nPtr;
nPtr = &n;
*nPtr = 15;
Edit1->Text=IntToStr(n);
```

```
Edit2->Text=IntToStr(nPtr);
Edit3->Text=IntToStr(*nPtr);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------
```



10.16-rasm. *Natija oynasi.*

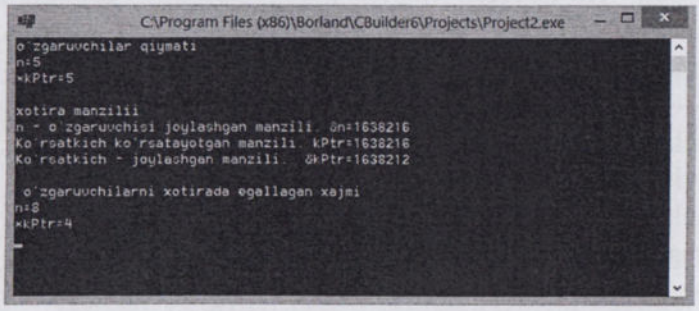
6-misol. Ko‘rsatkich joylashgan manzilini topish dasturi tuzilsin.

Dasturning konsol rejimidagi ko‘rinishi quyidagicha:

```
//---------------------------------------------------------
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{double n = 5;
double *kPtr;
kPtr = &n;
cout << "o‘zgaruvchilar qiymati" << endl;
```

```
    cout << "n=" << n << endl;
    cout << "*kPtr=" << *kPtr << endl;
    cout << "\n xotira manzili" << endl;
    cout << "n - o'zgaruvchisi joylashgan manzili. &n=" << &n
<< endl;
    cout << "Ko'rsatkich ko'rsatayotgan manzili. kPtr=" << kPtr
<<endl;
    cout << "Ko'rsatkich joylashgan manzili.  &kPtr=" << &kPtr
<<endl;
    cout << "\n o'zgaruvchilarni xotirada egallagan hajmi" <<endl;
    cout << "n=" << sizeof(n) << endl;
    cout << "*kPtr=" << sizeof(kPtr) << endl;
    getch();
    return 0;
    }
    //---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



10.17-rasm. *Natija oynasi.*

Ushbu dasturni vizual muhitda amalga oshiramiz. Buning uchun 4 ta Label, 4 ta Edit va 2 ta BitBtn komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

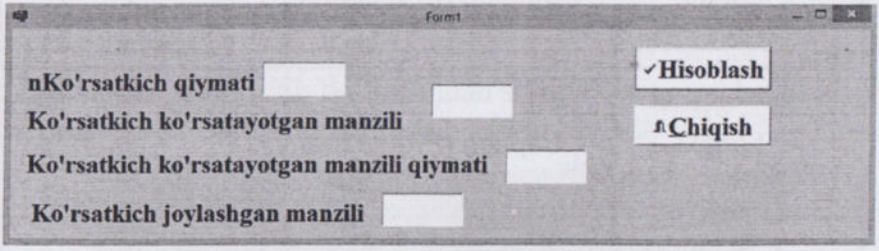
10.4-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Ko'rsatkichlar bilan ishlash" so'zi kiritiladi.
Label1	Caption (Properties)	"nKo'rsatkich qiymati" so'zi kiritiladi.
Label2	Caption (Properties)	"Ko'rsatkich ko'rsatayotgan manzili" so'zi kiritiladi.
Label3	Caption (Properties)	"Ko'rsatkich ko'rsatayotgan manzili qiymati" so'zi kiritiladi.
Label4	Caption (Properties)	"Ko'rsatkich joylashgan manzil" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zi o'chirib tashlanadi.
Edit2	Text (Properties)	"Edit2" so'zi o'chirib tashlanadi.
Edit3	Text (Properties)	"Edit3" so'zi o'chirib tashlanadi.
Edit4	Text (Properties)	"Edit4" so'zi o'chirib tashlanadi.

BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Hisoblash" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:

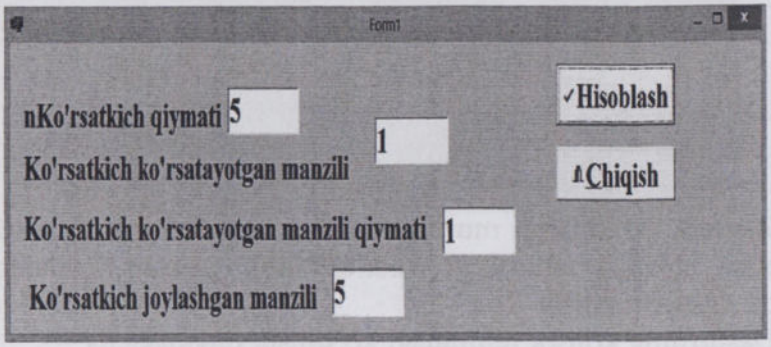


10.18-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
int n = 5;
int * nPtr;
nPtr = &n;
Edit1->Text=IntToStr(n);
Edit2->Text=IntToStr(&n);
Edit3->Text=IntToStr(nPtr);
Edit4->Text=IntToStr(*nPtr);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------
```



10.19-rasm. *Natija oynasi.*

Murojaatlar e'londa ko'rsatilgan nomning sinonimi sifatida ishlatiladi, ya'ni bitta o'zgaruvchiga har xil nom bilan murojaat qilish mumkin. Murojaatni doimiy qiymatga ega bo'lgan ko'rsatkich deb qarash mumkin bo'ladi. Murojaat quyidagicha e'lon qilinadi:

<toifa> & <nomi>;

Bu yerda, <toifa> – murojaat ko'rsatuvchi qiymatning toifasi, '&' belgisi, undan keyin yozilgan <nomi> – murojaat toifasidagi nomi ekanligini bildiruvchi operator.

Boshqacha aytganda '&' belgisiga adresni olish amali deyiladi.

Namuna:

```
int k;
int & p = k;
```

Murojaat asosan funksiyalarda adres orqali uzatiluvchi parametrlar sifatida ishlatiladi.

10.5. Matnli ma'lumotlarga murojaat etish funksiyalari

C++ dasturlash tizimida matnli ma'lumotlarga murojaat etish va ular ustida amallar bajarish uchun quyidagi ko'rsatkich funksiyalari mavjud:

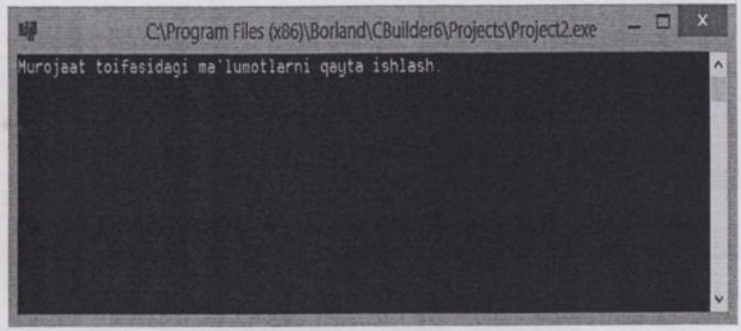
— strcat – qatorlarni bir-biri bilan birlashtirish;
— strcmp – qatorlarni bir-biri bilan solishtirish;
— strcpy – bir qatordan boshqasiga nusha olish;
— strstr – qatordan qidirish;
— strlen – qator uzunligini aniqlash;
— strend – ko'rsatkichi oxiriga olib borish;
— strpos – ko'rsatkich holatini aniqlash;

7-misol. Matnlarga murojaat etish va qatorlarni birlashtirish amalini bajarish uchun ko'rsatkich funksiyasidan foydalangan holda dastur tuzilsin.

Dasturning konsol rejimidagi ko'rinishi quyidagicha:

```
//---------------------------------------------------------
#include <vcl.h>
```

```
#include <string.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop
//---------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
char str[100];
  strcpy( str, "Murojaat ");
  strcat( str, "toifasidagi ");
  strcat( str, "ma'lumotlarni ");
  strcat( str, "qayta ");
  strcat( str, "ishlash." );
  std::cout << str << std::endl;
  getch ();
return 0;
}
//---------------------------------------------------------------
```



10.20-rasm. *Natija oynasi.*

8-misol. Matnlarga murojaat etish va ular ustida bir nechta amallarni bajarish uchun ko'rsatkich funksiyalaridan foydalangan holda dastur tuzilsin.

Misolni vizual muhitda dasturlash uchun 5 ta Label, 4 ta BitBtn komponentalari kerak bo'ladi.

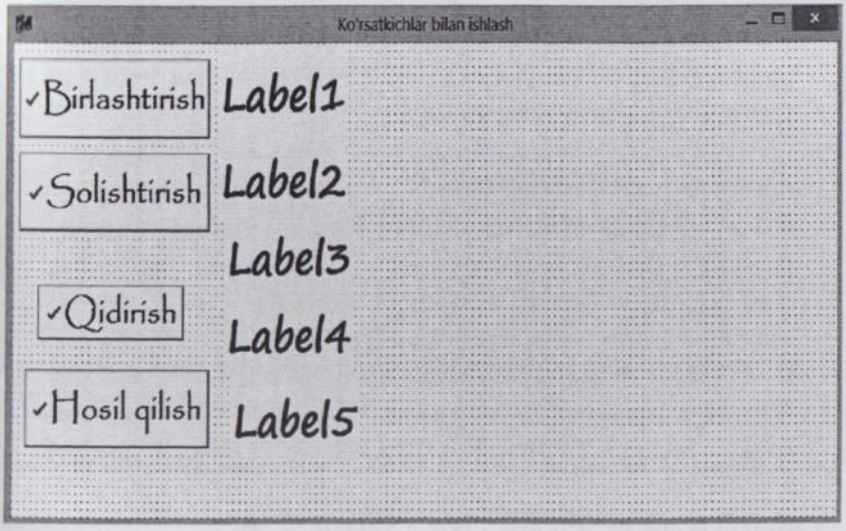
Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

10.5-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Ko'rsatkichlar bilan ishlash" so'zi kiritiladi.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Birlashtirish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Solishtirish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn3	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Qidirish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn4	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Hosil qilish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:

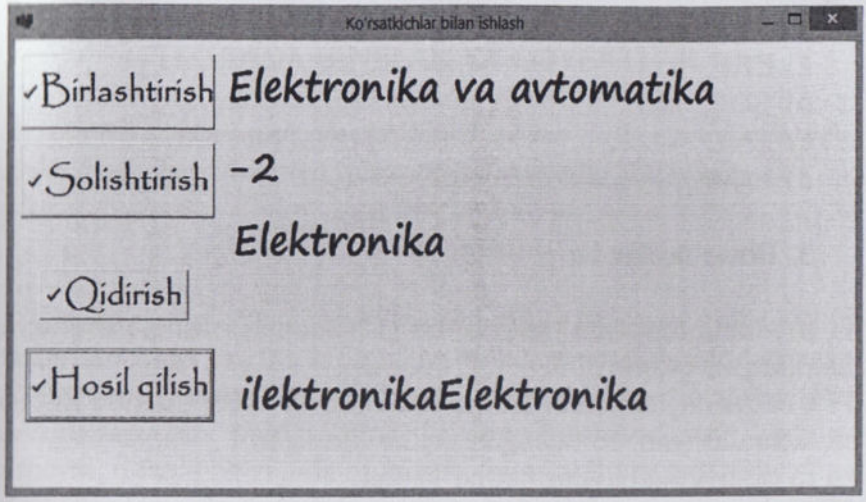


10.21-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#include <string.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
```

```
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
char S1[20] = "Elektronika va ", S2[10] = "avtomatika";
Label1->Caption=strcat(S1,S2);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
char S1[20] = "ener", S2[10] = "getika";
Label2->Caption=strcmp(S1,S2);
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn3Click(TObject *Sender)
{
char S1[20] = "Elektronika", S2[10] = "E";
Label3->Caption=strstr(S1,S2);
char S3[20] = "Elektronika", S4[10] = "y";
Label4->Caption=strstr(S3, S4);}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn4Click(TObject *Sender)
{ char S1[20]="ilektronika", S2[20]="i", S3[20]="E", S[60],
*St;
St = strstr(S1,S2);
if(St)
{ *St = 0;
St += strlen(S2);
Label5->Caption = strcat(strcat(strcpy(S,S1),S3),St);
} else Label5->Caption = "matn topilmadi"; }
//---------------------------------------------------------
```



10.22-rasm. *Natija oynasi.*

Nazorat savollari

1. Fayllar bilan ishlash asoslari.
2. Fayldan o'qish-yozish funksiyalari.
3. Fayl ko'rsatkichini boshqarish funksiyalari.
4. Fayl bilan ishlash uchun qaysi funksiyalardan foydalaniladi?
5. Matnli ma'lumotlarga murojaat etish funksiyalarini sanab o'ting.

Test savollari

1. Ifstream nima uchun ishlatiladi?

a) faylni kiritish
b) faylni tahrirlash
c) faylni chiqarish
d) faylli yozish

2. Faylning oxirini aniqlovchi funksiya bu –

a) EEF
b) EOF
c) END
d) EOA

3. Binar fayllar bu –

a) ma'lumotlarni saqlab qo'yish uchun tashqi xotiraning nomlangan qismi.

b) fizik fayllar bilan ishlash uchun dasturlash tillarida maxsus strukturalashgan, toifalangan fayllar kiritilgan.

c) oddiygina baytlar ketma-ketligi.

d) fizik xotirani band qilmasdan ma'lumotlarning mantiqiy modelini o'zida saqlaydi.

4. Ofstream nima uchun ishlatiladi?

a) faylli kiritish
b) faylni tahrirlash
c) faylli yozish
d) faylli chiqarish

5. Strcat(); ko'rsatkich funksiyasi nima vazifani bajaradi?

a) qatorlarni bir-biri bilan birlashtirish
b) qatorlarni bir-biri bilan solishtirish
c) bir qatordan boshqasiga nusha olish
d) qatordan qidirish

XI bob. BORLAND C++ BUILDER DASTURLASH TIZIMIDA SINFLAR VA OBYEKTLAR

Tayanch so'zlar: sinf, obyekt, obyektga mo'ljallangan yondashuv, atributlar, inkapsulyatsiya, abstraksiya, vorislik, polimorfizm.

11.1. Sinflar

Obyektga mo'ljallangan yondashuv dasturiy tizimlarni dasturlash tiliga bog'liq bo'lmagan holda yaratishda modellardan sistematik foydalanishga asoslangan. Har bir model uning o'zi aks ettirayotgan predmetning hamma xususiyatlarini ifodalay olmaydi, u faqat ba'zi juda muhim belgilarini ifodalaydi. Obyektga mo'ljallangan dasturlashda dastur obyektlarini va ularning xususiyatlarini (atributlarini) va ularni birlashtiruvchi sinflarni tavsiflash imkonini beradi.

Atributlar va usullarni tadqiq qilish asosida bazaviy sinflar va ularning hosilalarini yaratish imkoniyati paydo bo'ladi.

Obyektga mo'ljallangan dasturlashning yana bir nazariy jihatdan juda muhim va zarur xususiyatlaridan biri hodisalarni ishlash mexanizmi hisoblanadi, ular yordamida obyektlar atributlari qiymatlari o'zgartiriladi. Obyektga mo'ljallangan dasturlashda avval yaratilgan obyektlar kutubxonasi va usullaridan foydalanish hisobiga obyektga yo'naltirilgan dasturlashda ancha mehnat tejaladi.

Obyektlar, sinflar va usullar polimorfizm bo'lishlari mumkin, bu esa dasturiy vositaning qulay foydalanishligi va universalligini ta'minlaydi.

Sinf termini bilan obyektlar turi aniqlanadi. Sinfning har bir vakili (nushasi) obyekt deb nomlanadi. Har bir obyekt o'zining alohida holatiga ega bo'ladi. Obyekt holati uning berilgan a'zolarining ayni paytdagi qiymati bilan aniqlanadi. Sinf vazifasi uning funksiya a'zolarining sinf obyektlari ustida bajaradigan amallar imkoniyati bilan aniqlanadi. Sinf jismoniy mohiyatga ega

emas, tuzilmaning e'lon qilinishi uning eng yaqin analogiyasidir. Sinf obyektni yaratish uchun qo'llangandagina, xotira ajralib chiqadi. Bu jarayon ham sinf nushasini yaratish deb ataladi.

C++ tilining har qanday obyekti bir xil atributlarga, shuningdek ushbu sinfning boshqa obyektlari bilan funksionallikka ega. O'z sinflarini yaratish hamda ushbu sinflar obyektlarining xulq-atvori uchun to'liq mas'uliyat dasturchi zimmasiga yuklanadi. Biron-bir muhitda ishlar ekan, dasturchi standart sinflarning kattagina kutubxonasiga (masalan, C++ Builder Visual Komponentlar kutubxonasi) kirish huquqiga ega bo'ladi.

Berilganlarni abstraksiyalash – berilganlarning yangi turini yaratish imkoniyati bo'lib, bu turlar bilan xuddi berilganlarning tayanch turlari bilan ishlagandek ishlash mumkin. Odatda yangi turlarni berilganlarning abstrakt turi deyiladi, garchi ularni soddaroq qilib "foydalanuvchi tomonidan aniqlangan tur" deb atash mumkin.

Inkapsulyasiya – bu berilganlar va ularni qayta ishlovchi kodni birlashtirish mexanizmidir. Inkapsulyasiya berilganlar va kodni tashqi ta'sirdan saqlash imkonini beradi.

Abstraksiya murakkab masalani soddalashtirish jarayonidir. Muayyan masalani yechishga kirishar ekansiz, siz barcha detallarni hisobga olishga urinmaysiz, balki yechimni osonlashtiradiganlarini tanlab olasiz.

Aytaylik, siz yo'l harakati modelini tuzishingiz kerak. Shunisi ayonki, bu o'rinda siz svetoforlar, mashinalar, shosselar, bir tomonlama va ikki tomonlama ko'chalar, ob-havo sharoitlari va h.k. sinflarini yaratasiz. Ushbu elementlarning har biri transport harakatiga ta'sir ko'rsatadi. Biroq bu o'rinda hasharotlar va qushlar ham yo'lda paydo bo'lishi mumkin bo'lsada, siz ularning modelini yaratmaysiz. Siz haqiqiy olamni soddalashtirasiz hamda uning faqat asosiy elementlaridan foydalanasiz. Mashina – modelning muhim detali, biroq bu BMW yoki boshqa biron markadagi mashinami, yo'l harakati modeli uchun bu detallar ortiqcha.

Abstraksiyaning ikkita afzal jihati bor. Birinchidan, u masala yechimini soddalashtiradi. Muhimi yana shundaki, abstraksiya tufayli dasturiy ta'minot komponentalaridan takroran foydalanish mumkin. Takroran qo'llanadigan komponentalarni yaratishda ular odatda g'oyat ixtisoslashadi. Ya'ni, komponentalar biron-bir ma'lum masala yechimiga mo'ljallangani, yana ular keraksiz o'zaro bog'liqlikda bo'lgani sababli dastur fragmentining boshqa biron o'rinda takroran qo'llanishi qiyinlashadi. Imkoni boricha bir qator masalalarni yechishga qaratilgan obyektlarni yaratishga harakat qiling. Abstraksiya bitta masala yechimidan ushbu sohadagi boshqa masalalarni ham yechishda foydalanish imkonini beradi.

Sinflarni yozishda biz funksiyalarni yozishdagi tartib-qoidalarga rioya qilamiz. Sinfning birinchi qatoriga kalit so'z class va sinf nomi, so'ngra yangi qatordan figurali qavslar ochiladi va uning ichiga sinf usullari va atributlari yoziladi.

Sinf quyidagi bo'limlarga ega bo'lishi mumkin:

1. Private (private, ichki).
2. Protected (protected, himoyalangan qism).
3. Public (public, umumiy).

Endi bazaviy sinfning umumiy yozilish sintaksisini quyidagicha yozish mumkin:

class className {private:

<private berilmalar a'zolari> <private konstruktorlar> <private usullar>

protected:

<Himoyalangan a'zo berilmalar> <Himoyalangan konstruktorlar>

<Himoyalangan usullar>

public:

<Umumiy murojaatli xususiyatlar> <Umumiy murojaatli a'zo berilmalar>

<Umumiy huquqli konstruktorlar va destruktorlar> <Umumiy huquqli usullar>

}

C++ ning bazaviy sinflarining bo'limlariga quyidagicha huquqlar aniqlangan:

1. Private qismi – shu sinfning faqat usullariga murojaatni aniqlaydi. Hosilaviy sinflar uchun private usullarga foydalanish huquqi berilmaydi.

2. Himoyalangan protected nomlari faqat shu sinf usullariga va shu sinf hosila sinfi usullariga murojaat qilishni beradi.

3. Umumiy huquqli public nomlari hamma turdagi sinflarning usullariga murojaat qilishni beradi.

Sinflarni aniqlashda bo'limlardan foydalanishning asosiy qoidalari:

1. Bo'limlar istalgan tartibda e'lon qilinishlari mumkin, hatto qayta tavsiflashlar ham uchrashi mumkin.

2. Agar bo'lim nomlangan bo'lmasa, u holda kompilyator sinfda oxirgi aniqlangan nomlarni private berilma deb qabul qiladi.

Agar biz a'zo berilmalarga murojaat qilishni cheklamoqchi bo'lsak ularni umumiy foydalanish bo'limiga joylashtirmasligimiz lozim.

11.2. Abstraksiya

Abstraksiya – bu identifikatorlardan farqli bo'lgan istalgan dasturlash tili ifodasi hisoblanadi.

Obyektga mo'ljallangan dasturlashda har bir obyekt prinsipial dinamik mohiyatga ega, ya'ni u vaqtga bog'lik holda va unga nisbatan tashqi faktorlar ta'sirida o'zgaradi. Boshqacha aytganda obyekt ma'lum bir darajada o'zini tutishiga ega. Obyektga mo'ljallangan dasturlashda abstraksiya obyektga mo'ljallangan dasturlashning modeli hisoblanadi. Sinf umumiy xususiyatlar va xulq-atvorga ega bo'lgan obyektlarni birlashtiradi. Bitta sinfga mansub obyektlar bir xil xususiyatlarga ega bo'lib, bir xil xatti-harakatni namoyon etadi.

Sinflar shablon (qolip)ga o'xshaydi: ular obyektlarning nushalarini 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 tomonidan 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 chaqirish", "funksiyasini chaqirish" yoki "xabarni uzatish" atamalari qo'llaniladi.

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 obyektga mo'ljallangan dasturlashda bo'lganidek, dastur obyektlardan tashkil topganidek, obyektlar ham, o'z navbatida, jamlovchi yordamida boshqa obyektlardan jamlanishi mumkin. Ushbu obyektlarning har bittasida uslub va belgilarga ega bo'lgan interfeys mavjud bo'ladi.

Xabar – obyektga mo'ljallangan yondashuvning 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.

11.3. Vorislik

Vorislik mavjud bo‘lgan sinfning ta’rifi asosida yangi sinfni yaratish imkonini beradi. Yangi sinf boshqasi asosida yaratilgach, uning ta’rifi avtomatik tarzda mavjud sinfning barcha xususiyatlari, xulq-atvori va joriy qilinishiga vorislik qiladi. Avval mavjud bo‘lgan sinf interfeysining barcha metodlari va xususiyatlari avtomatik tarzda voris interfeysida paydo bo‘ladi. Vorislik voris sinfida biron-bir jihatdan to‘g‘ri kelmagan xulq-atvorni avvaldan ko‘ra bilish imkonini beradi. Bunday foydali xususiyat dasturiy ta’minotni talablarning o‘zgarishiga moslashtirish imkonini beradi. Agar o‘zgartirishlar kiritishga ehtiyoj tug‘ilsa, bu holda eski sinf funksiyalariga vorislik qiluvchi yangi sinf yozib qo‘ya qolinadi. Keyin o‘zgartirilishi lozim bo‘lgan funksiyalarga qaytadan ta’rif beriladi hamda yangi funksiyalar qo‘shiladi. Bunday o‘rniga o‘rin qo‘yishning mazmuni shundan iboratki, u dastlabki sinf ta’rifini o‘zgartirmay turib, obyekt ishini o‘zgartirish imkonini beradi. Axir bu holda qayta test sinovlaridan puxta o‘tkazilgan asosiy sinflarga tegmasa ham bo‘ladi. Agar siz ko‘p martalab qo‘llash yoki boshqa biron maqsadlarga ko‘ra vorislikni qo‘llashga ahd qilsangiz, avval har gal qarang vorislik sinf bilan vorislikni berayotgan sinfning turlari o‘zaro mos keladimi? Vorislikda turlarining mos kelishi ko‘pincha boshqa sinfga vorislik bo‘layotgan sinf voris berayotgan sinf bilan shunday munosabatda bo‘lishi lozimki, bunda natijaviy munosabatlar o‘z ma’nosiga ega bo‘lishi, ya’ni vorislik tabaqalanishiga amal qilinishi kerak.

Vorislik yordamida qurilgan sinf metodlar va xususiyatlari uchta ko‘rinishga ega bo‘lishi mumkin:

– o‘rniga o‘rin qo‘yish (almashtirish): yangi sinf ajdodlarining metodi yoki xususiyatini shunchaki o‘zlashtirib olmaydi, balki unga yangi ta’rif ham beradi;

– yangi: yangi sinf butunlay yangi metodlar yoki xususiyatlarni qo‘shadi;

— rekursiv: yangi sinf o‘z ajdodlari metodlari yoki xususiyatlarini to‘g‘ridan to‘g‘ri olib qo‘ya qoladi.

Obyektga mo‘ljallangan tillarning ko‘pchiligi ta’rifni ma’lumot uzatilgan obyektdan qidiradi. Agar u yerdan ta’rif topishning iloji bo‘lmasa, biron ta’rif topilmaguncha, qidiruv tabaqalar bo‘yicha yuqoriga ko‘tarilib boradi. Ma’lumotni boshqarish aynan shunday amalga oshiriladi hamda aynan shu tufayli o‘ringa o‘rin qo‘yish jarayoni amalga oshadi.

Voris sinflar himoyalangan kirish darajasiga ega bo‘lgan metodlar va xususiyatlarga kirish huquqini olishlari mumkin. Bazaviy sinfda faqat avlodlar foydalanishi mumkinligi aniq bo‘lgan metodlargagina himoyalangan kirish darajasini berishi mumkin. Boshqa hollarda xususiy yoki ommaviy kirish darajasidan foydalanish lozim. Bunday yondashuv barcha sinflarga, shu jumladan, tarmoq sinflarga ham kirish huquqi berilganidan ko‘ra, mustahkamroq konstruksiyani yaratish imkonini beradi.

Vorislik uch asosiy hollarda qo‘llanadi:

— ko‘p martalab foydalanishda;

— ajralib turish uchun;

— turlarni almashtirish uchun.

Vorislikning ayrim turlaridan foydalanish boshqalaridan ko‘ra afzalroq hisoblanadi. Vorislik yangi sinfga eski sinfning amalda qo‘llanishidan ko‘p martalab foydalanish imkonini beradi. Kodni qirqib tashlash yoki kiritish o‘rniga, vorislik kodga avtomatik tarzda kirishni ta’minlaydi, ya’ni kodga kirishda, u yangi sinfning bir qismidek olib qaraladi. Ko‘p martalab qo‘llash uchun vorislikdan foydalanar ekansiz, siz voris qilib olingan joriy qilinish bilan bog‘liq bo‘ladi. Vorislikning bu turini ehtiyotkorlik bilan qo‘llash lozim.

Farqlash uchun vorislik faqat avlod-sinf va ajdod-sinf o‘rtasidagi farqlarni dasturlash imkonini beradi. Farqlarni dasturlash yaxshi vosita hisoblanadi. Kodlash hajmining kichikligi va kodning oson boshqarilishi loyiha ishlanmasini osonlashtiradi. Bu holda kod

satrlarini kamroq yozishga to‘g‘ri keladi, bu qo‘shiladigan xatolar miqdorini ham kamaytiradi.

Vorislik obyektga mo‘ljallangan dasturlashning muhim xususiyatlariga kiradi. Misol uchun, V sinfi A sinfini meroslashini ko‘rsatish uchun (V sinfi A sinfidan tashkil etilgan) V sinfini aniqlashda sinf nomidan keyin ikki nuqta qo‘yiladi va so‘ngra V vorislanayotgan sinflar keltiriladi:

```
class A {public: A (); A ();
MethodA ();};
Class V: public A {public: V();
...};
```

11.4. Polimorfizm

Agar inkapsulyatsiyalash va vorislikni obyektga mo‘ljallangan yondashuvning foydali vositalari sifatida olib qarash mumkin bo‘lsa, polimorfizm – eng universal va radikal vositadir. Polimorfizm inkapsulyatsiyalash va vorislik bilan chambarchas bog‘liq, polimorfizmsiz obyektga mo‘ljallangan yondashuv samarali bo‘lolmaydi. Polimorfizm – obyektga mo‘ljallangan yondashuv paradigmasida markaziy tushunchadir. Polimorfizmni egallamay turib, obyektga mo‘ljallangan yondashuvdan samarali foydalanish mumkin emas.

Polimorfizm shunday holatki, bunda qandaydir bitta narsa ko‘p shakllarga ega bo‘ladi. Dasturlash tilida "ko‘p shakllar" deyilganda, bitta nom avtomatik mexanizm tomonidan tanlab olingan turli kodlarning nomidan ish ko‘rishi tushuniladi. Shunday qilib, polimorfizm yordamida bitta nom turli xulq-atvorni bildirishi mumkin.

Vorislik polimorfizmning ayrim turlaridan foydalanish uchun zarurdir. Aynan o‘rindoshlik imkoniyati mavjud bo‘lgani uchun, polimorfizmdan foydalanish mumkin bo‘ladi. Polimorfizm yordamida tizimga to‘g‘ri kelgan paytda qo‘shimcha funksiyalarni qo‘shish mumkin. Dasturni yozish paytida hatto

taxmin qilinmagan funksionallik bilan yangi sinflarni qo'shish mumkin, buning ustiga bularning hammasini dastlabki dasturni o'zgartirmay turib ham amalga oshirish mumkin. Yangi talablarga osongina moslasha oladigan dasturiy vosita deganda, mana shular tushuniladi.

Polimorfizmning uchta asosiy turi mavjud:

— qo'shilish polimorfizmi;

— parametrik polimorfizm;

— ortiqcha yuklanish.

Qo'shilish polimorfizmini ba'zida sof polimorfizm deb ham ataydilar. Qo'shilish polimorfizmi shuning bilan qiziqarliki, u tufayli tarmoq sinf nusxalari o'zini turlicha tutishi mumkin. Qo'shilish polimorfizmidan foydalanib, yangi tarmoq sinflarni kiritgan holda tizimning xulq-atvorini o'zgartirish mumkin. Uning bosh afzalligi shundaki, dastlabki dasturni o'zgartirmay turib, yangi xulq-atvorni yaratish mumkin.

Aynan polimorfizm tufayli joriy qilishdan takroran foydalanishni vorislik bilan aynanlashtirish kerak emas. Buning o'rniga vorislikdan avvalambor o'zaro almashinish munosabatlari yordamida polimorf xulq-atvorga erishish uchun foydalanish lozim. Agar o'zaro almashinish munosabatlari to'g'ri belgilansa, buning ortidan albatta takroran qo'llash chiqib keladi. Qo'shilish polimorfizmidan foydalanib, bazaviy sinfdan, har qanday avloddan, shuningdek bazaviy sinf qo'llaydigan metodlardan takroran foydalanish mumkin.

Parametrik polimorfizmdan foydalanib, turdosh metodlar va turdosh (universal) turlar yaratish mumkin. Turdosh metodlar va turlar dalillarning ko'plab turlari bilan ishlay oladigan dasturni yozish imkonini beradi. Agar qo'shilish polimorfizmidan foydalanish obyektni idrok etishga ta'sir ko'rsatsa, parametrik polimorfizmdan foydalanish qo'llanayotgan metodlarga ta'sir ko'rsatadi. Parametrik polimorfizm yordamida, parametr turini bajarilish vaqtigacha e'lon qilmay turib, turdosh metodlar yaratish mumkin. Metodlarning parametrik parametrlari

bo‘lganidek, turlarning o‘zi ham parametrik bo‘lishi mumkin. Biroq polimorfizmning bunday turi barcha tillarda ham uchrayvermaydi.

Ortiqcha yuklanish yordamida bitta nom turlicha metodlarni bildirishi mumkin. Bunda metodlar faqat miqdorlari va parametr turlari bilan farqlanadi. Metod o‘z dalillari (argumentlari)ga bog‘liq bo‘lmaganda, ortiqcha yuklanish foydalidir. Metod o‘ziga xos parametrlar turlari bilan cheklanmaydi, balki har xil turdagi parametrlarga nisbatan ham qo‘llanadi. Masalan max metodini ko‘rib chiqaylik. Maksimal – turdosh tushuncha bo‘lib, u ikkita muayyan parametrlarni qabul qilib, ularning qaysi biri kattaroq ekanini ma’lum qiladi. Ta’rif butun sonlar yoki suzuvchi nuqtali sonlar qiyoslanishiga qarab o‘zgarmaydi.

Polimorfizmdan samarali foydalanish sari qo‘yilgan birinchi qadam bu inkapsulyatsiyalash va vorislikdan samarali foydalanishdir. Inkapsulyasiyalashsiz dastur osongina sinflarning joriy qilinishiga bog‘liq bo‘lib qolishi mumkin. Agar dastur sinflarning joriy qilinish aspektrlaridan biriga bog‘liq bo‘lib qolsa, tarmoq sinfda bu joriyni to‘g‘rilash mumkin bo‘lmaydi.

Vorislik – qo‘shilish polimorfizmining muhim tarkibiy qismi. Hamma vaqt bazaviy sinfga imkon darajada yaqinlashtirilgan darajada dasturlashga uringan holda o‘rinbosarlik munosabatlarini o‘rnatishga harakat qilish kerak. Bunday usul dasturda ishlov berilayotgan obyektlar turlari miqdorini oshiradi.

Puxta o‘ylab ishlab chiqilgan tabaqalanish o‘rinbosarlik munosabatlarini o‘rnatishga yordam beradi. Umumiy qismlarni abstrakt sinflarga olib chiqish kerak hamda obyektlarni shunday dasturlash kerakki, bunda obyektlarning ixtisoslashtirilgan nusxalari emas, balki ularning o‘zlari dasturlashtirilsin. Bu keyinchalik har qanday voris sinfni dasturda qo‘llash imkonini beradi.

Biroq ko‘p o‘rinlarda tajribasiz loyihachilar polimorfizmni kuchaytirish maqsadida xulq-atvorni juda baland tabaqaviy

darajaga olib chiqishga urinadilar. Bu holda har qanday avlod ham bu xulq-atvorni ushlab tura oladi. Shuni esdan chiqarmaslik kerakki, avlodlar o'z ajdodlarining funksiyalarini chiqarib tashlay olmaydilar. Dasturni yanada polimorfizm qilish maqsadida puxta rejalashtirilgan vorislik tabaqalarini buzish to'g'ri kelmaydi.

Nazorat savollari

1. Inkapsulyatsiya nima?
2. Polimorfizm haqida tushuncha.
3. Vorislikning qo'llanishi.
4. Sinf xossalari va usullari?
5. Abstraksiya nima?

Test savollari

1. Inkapsulyasiya bu –

a) berilgan obyektlarni nushasini oluvchi jarayondir.
b) berilganlar va ularni qayta ishlovchi kodni birlashtirish mexanizmidir.
c) obyektga yo'naltirilgan dasturlash tili.
d) obyekt holati uning berilganlar – a'zolarning ayni paytdagi qiymati.

2. Abstraksiya bu –

a) berilganlar va ularni qayta ishlovchi kodni birlashtirish mexanizmidir.
b) berilgan obyektlarni nushasini oluvchi jarayondir.
c) identifikatorlardan farqli bo'lgan istalgan dasturlash tili ifodasi hisoblanadi.
d) obyekt holati uning berilganlar – a'zolarning ayni paytdagi qiymati.

3. Vorislik dasturda qanday imkoniyat yaratadi.

a) identifikatorlardan farqli bo'lgan istalgan dasturlash tili ifodasi hisoblanadi.

b) mavjud bo'lgan sinfning ta'rifi asosida yangi sinfni yaratish imkonini beradi.

c) berilgan obyektlarni nushasini olish.

d) funksiyalarni qayta ishlash.

4. Polimorfizmning asosiy nechta turi mavjud?

a) 5

b) 2

c) 3

d) 4

5. Sinfning har bir vakili (nushasi) nima deb nomlanadi?

a) C++ Builder

b) vorislik

c) abstraksiya

d) obyekt

XII bob. BORLAND C++ BUILDER 6 NING GRAFIK IMKONIYATLARI

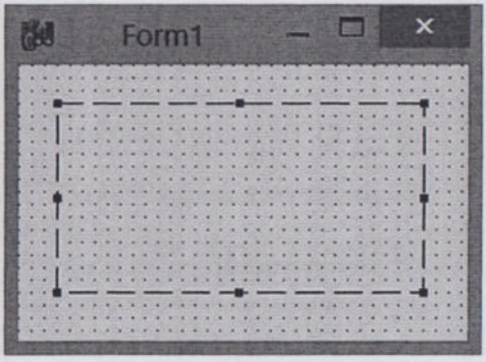
Tayanch so'zlar: interfeys, grafika, canvas, image obyekti, koordinata, piksel, qalam, mo'yqalam, tasvir.

12.1. Tayyor rasmlar bilan ishlash

Borland C++ Builder dasturchiga grafik dasturlar yordamida shemalar, chizmalar va funksiyalarning grafiklarini yaratish imkonini beradi. Borland C++ Builder dasturlash muhiti yordamida 2 xil ko'rinishdagi grafika bilan ishlash imkoniyati mavjud.

1. Tayyor rasmlar bilan.
2. Foydlanuvchi tomonidan yaratilgan tasvirlar.

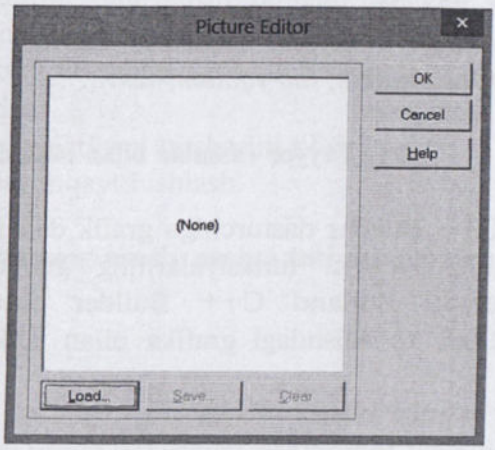
Tayyor rasmlar bilan ishlash uchun additional komponentalar palitrasida joylashgan image komponentasidan foydalaniladi. Komponenta forma darchasiga o'rnatilgandan so'ng quyidagi ko'rinishga ega bo'ladi:



12.1-rasm. *Image komponentasining ko'rinishi.*

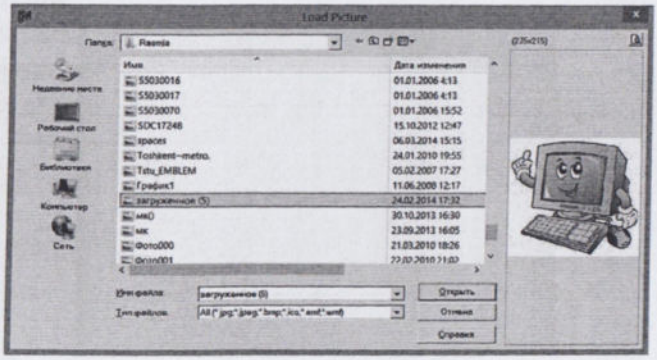
Image komponentasining o'lchamlari height va width xususiyati orqali o'zgartiriladi. Image komponentasining picture

xossasi orqali kerakli rasm tanlanadi. Picture xossasi bosilganda ekranda quyidagi oyna hosil bo'ladi:



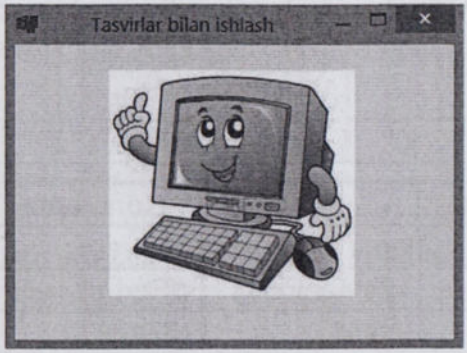
12.2-rasm. *Image komponentasi picture xususiyati.*

Hosil bo'lgan darchada load tugmasi bosiladi va rasm joylashgan yo'l tanlanadi.



12.3-rasm. *Rasmni tanlash darchasi.*

Kerakli fayl tanlangandan so'ng dastur ko'rinishi quyidagicha bo'ladi:



12.4-rasm. *Natija oynasi.*

12.2. Qalam va mo'yqalam

Borland C++ Builderning grafik imkoniyatlari ham qalam va mo'yqalamdan foydalanish imkoniyatlarini yaratadi. Qalamdan chiziq va kontur chizishda, mo'yqalamdan esa kontur bilan chegaralangan yuzani bo'yash uchun foydalaniladi.

Qalam va mo'yqalam grafikani chizish yuzasida hosil qilishda mos ravishda pen (qalam) va brush (mo'yqalam) xususiyatlariga xosdir.

Qalam. Qalamdan nuqta, chiziq, geometrik shakllar: to'g'ri to'rtburchak, aylana, ellips va h.k.larni chizishda qurol sifatida foydalaniladi. TPen obyekt xususiyati 12.1-jadvalda keltirilgan.

12.1-jadval.

TPen obyekt xususiyati

Xususiyati	Vazifasi
Color	Chiziq (kontur) rangi
Width	Chiziq qalinligi
Style	Chiziq ko'rinishi
Mode	Tasvirlash rejimi

TPen obyektining Color xususiyati chizuvchi qalam rangini belgilaydi. Quyidagi 12.2-jadvalda PenColor xususiyatlari keltirilgan:

12.2-jadval.

PenColor xususiyati

Konstanta	Rang	Konstanta	Rang
clBlack	qora	clSilver	kumush rang
clMaroon	kashtan rang	clRed	qizil
clGreen	yashil	clLime	och yashil
clFuchsia	pushti	clBlue	ko'k
clNavy	to'q ko'k	clYellow	sariq
clPurple	atirgulrang	clMedGray	o'rta kulrang
clGray	kulrang	clWhite	oq

TPen obyektining width xususiyati chizuvchi qalam qalinligini (pikselda) belgilaydi.

Masalan, Canvas->Pen->Width->2 chiziq qalinligi 2 pikselga teng bo'ladi.

TPen obyektining style xususiyati chiziluvchi chiziqning turini belgilaydi. TPen obyektining style xususiyatlari 12.3-jadvalda keltirilgan.

12.3-jadval.

Style xususiyatlari

Konstanta		Ko'rinishi
psSolid	To'g'ri chiziq	————
psDash	Uzun shtrixli punktir chiziq	- - - - - - - - -
psDot	Qisqa shtrixli punktir chiziq	.....
psDashDot	Uzun-qisqa shtrixli punktir chiziq	-·-·-·-·-·-·-·-

PsDashDotDot	Bir uzun va ikki qisqa shtrixli punktir chiziq	
PsClear	Ko'rinmas chiziq	

2. Mo'yqalam. Mo'yqalamdan (Brush) yopiq sohalarni to'ldirish uchun foydalaniladi, masalan, geometrik shakllarni bo'yash va h.k. Mo'yqalam obyekt sifatida quyidagi ikki xususiyatni o'z ichiga oladi:

- Color – bo'yaluvchi soha rangi;
- Style – to'ldiruvchi soha ko'rinishi.

Masalan, konturning ichki sohasi bo'yalishi yoki shtrixlanishi mumkin. Color xususiyati sifatida barcha o'zgarmaslaridan foydalanish mumkin.

Style xususiyatlari 12.4-jadvalda keltirilgan.

12.4-jadval.

Style xususiyati

Konstanta	Bo'yash turi
bsSolid	yaxlit bo'yash
bsClear	soha bo'yalmaydi
bsHorizontal	gorizantal shtixlash
bsVertical	vertical shtixlash
bsFDiagonal	oldinga egilish bilan diagonal shtrixlash
bsBDiagonal	orqaga egilish bilan diagonal shtrixlash
bsCross	gorizontal-vertikal shtrixlash, to'r ko'rinishida
bsDiagCross	diagonal shtrixlash, to'r ko'rinishida

1-misol. Mo'yqalamning barcha xususiyatlarini ko'rsatuvchi dastur tuzilsin.

Misolni vizual muhitda dasturlash uchun 1 ta BitBtn va 1 Label komponentalari kerak bo'ladi.

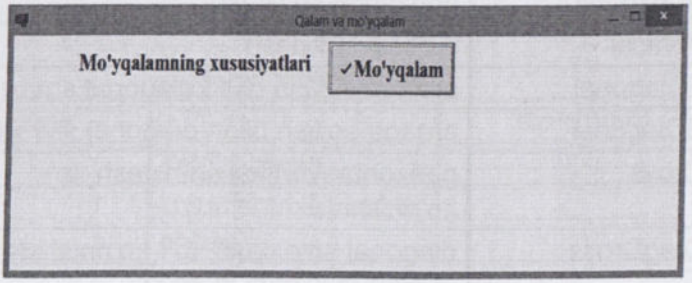
Forma darchasiga o‘rnatilgan komponentalarning xususiyatlarini quyidagicha belgilaymiz:

12.5-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	“Qalam va mo‘yqalam” so‘zi kiritiladi.
Label1	Caption (Properties)	“Mo‘yqalamning xususiyatlari” so‘zi kiritiladi.
BitBtn1	Kind (Properties)	“bkOK” xususiyati tanlanadi.
	Caption (Properties)	“Mo‘yqalam” so‘zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so‘ng dastur dizayni quyidagi ko‘rinishga keladi:



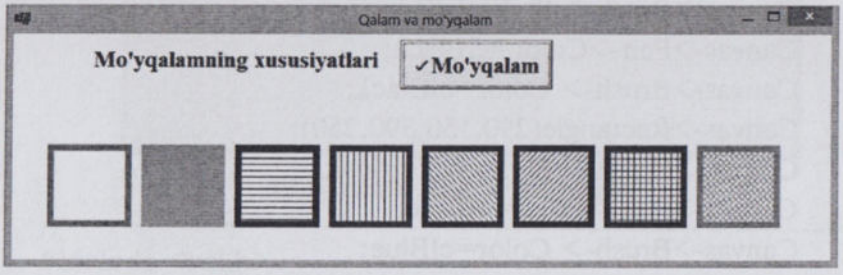
12.5-rasm. *Dastur ko‘rinishi.*

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
```

```
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
Canvas->Pen->Width=8;
Canvas->Brush-> Style=bsClear;
Canvas->Pen->Color=clRed;
Canvas->Rectangle(50,150,150,250);
//---------------------------------------------------------
Canvas->Brush-> Style=bsSolid;
Canvas->Pen->Color=clMedGray;
Canvas->Brush-> Color=clMedGray;
Canvas->Rectangle(170,150,270,250);
//---------------------------------------------------------
Canvas->Brush-> Style=bsHorizontal;
Canvas->Pen->Color=clBlack;
Canvas->Brush-> Color=clBlack;
Canvas->Rectangle(290,150,390,250);
Canvas->Brush-> Style=bsVertical;
Canvas->Pen->Color=clBlue;
Canvas->Brush-> Color=clBlue;
Canvas->Rectangle(410,150,510,250);
//---------------------------------------------------------
```

```
Canvas->Brush-> Style=bsFiagonal;
Canvas->Pen->Color=clMaroon;
Canvas->Brush-> Color=clMaroon;
Canvas->Rectangle(530,150,630,250);
//-------------------------------------------------------------
Canvas->Brush-> Style=bsBDiagonal;
Canvas->Pen->Color=clNavy;
Canvas->Brush-> Color=clNavy;
Canvas->Rectangle(650,150,750,250);
//-------------------------------------------------------------
Canvas->Brush-> Style=bsCross;
Canvas->Pen->Color=clPurple;
Canvas->Brush-> Color=clPurple;
Canvas->Rectangle(770,150,870,250);
//-------------------------------------------------------------
Canvas->Brush-> Style=bsDiagCross;
Canvas->Pen->Color=clFuchsia;
Canvas->Brush-> Color=clFuchsia;
Canvas->Rectangle(890,150,990,250);
}
//-------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



12.6-rasm. *Natija oynasi.*

12.3. Oddiy figuralar chizish

Grafik elementlarni (to‘g‘ri chiziq, aylana, to‘g‘ri to‘rtburchak va h.k.) obyekt yuzasida hosil qilish uchun Canvas xususiyatidan foydalaniladi.

Borland C++ Builder dasturida quyidagi grafik elementlarni hosil qilsih mumkin:

1. To‘g‘ri chiziq. Borland C++ da to‘g‘ri chiziq hosil qilish uchun LineTo dan foydalaniladi. Uning yozilishi quyidagicha:

Canvas->LineTo (x, y);

LineTo to‘g‘ri chiziqni qalam (ko‘rsatkich) turgan koordinatadan boshlab x, y – nuqtagacha chizadi. Shuning uchun chiziqning boshlang‘ich nuqtasini kerakli joyga o‘rnatib olish lozim bo‘ladi. Bunda biz MoveTo ga murojaat qilamiz:

Canvas->MoveTo (x0, y0);

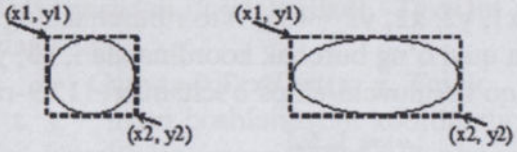
Chiziqning ko‘rinishi (rangi, qalinligi va turi) Pen obyekti bilan ifodalanadi.

2. Aylana va ellips. Ellipse uslubi ellips va aylana chizish uchun qo‘laniladi. Ellipsening yozilish formati quyidagicha:

Canvas->Ellipse (x1, y1, x2, y2);

Bu yerda, x1, y1, x2, y2 – hosil bo‘luvchi aylana yoki ellipsga tashqi chizilgan to‘g‘ri to‘rtburchakning mos ravishda yuqori chap va quyi o‘ng nuqtalari koordinatalari (12.7-rasm).

Chiziqning ko‘rinishi (rangi, qalinligi va turi) Pen obyekti bilan ifodalanadi.



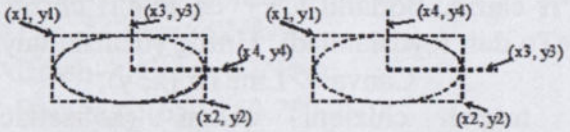
12.7-rasm. *Aylana va ellipsga qiymat berish.*

3. Yoy. Yoy hosil qilish uchun Arc uslubidan foydalaniladi. Uning yozilish formati quyidagicha:

Canvas->Arc (x1, y1, x2, y2, x3, y3, x4, y4);

Bu yerda, x1, y1, x2, y2 – hosil bo'luvchi yoyni davom ettirib hosil qilinuvchi ellips (aylana)ga tashqi chizilgan to'g'ri to'rtburchakning mos koordinatalari; x3, y3 – yoyning boshlang'ich nuqtasi; x4, y4 – yoyning tugash nuqtasi.

Shuni aytib o'tish lozimki, yoy soat strelkasi yo'nalishiga qarama-qarshi yo'nalishda chiziladi (12.8-rasm).



12.8-rasm. *Yoyga qiymat berish.*

Chiziqning ko'rinishi (rangi, qalinligi va turi) Pen obyekti bilan ifodalanadi.

4. To'g'ri to'rtburchak. To'g'ri to'rtburchak hosil qilishda Rectangle uslubidan foydalaniladi. Uning yozilish formati quyidagicha:

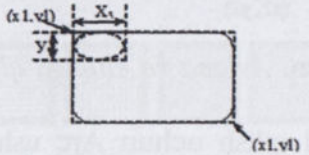
Canvas-> Rectangle (x1, y1, x2, y2)

Bu yerda, x1, y2, x2, y2 – to'g'ri to'rtburchakning mos ravishda yuqori chap va quyi o'ng burchak koordinatalari.

RoundRect uslubi ham to'g'ri to'rtburchak chizadi, faqat Rectangle dan farqi shundaki, uning burchaklari yumaloq (silliq) shaklda bo'ladi. Yozilish formati:

Canvas->RoundRect (x1, y1, x2, y2, x3, y3)

Bu yerda, x1, y2, x2, y2 – to'g'ri to'rtburchakning mos ravishda yuqori chap va quyi o'ng burchak koordinatalari; x3, y3 – yumaloq hosil qilishda qo'llaniluvchi ellips o'lchamlari (12.9-rasm).



12.9-rasm. *Chetlari yumaloq ko'rinishidagi to'g'ri to'rtburchakga qiymat berish*

5. Ko‘pburchak. Polygon xussusiyatidan foydalanib ko‘pburchak chizish mumkin. Polygon TPoint tipli massivni parametr sifatida qabul qiladi. Har bir massiv elementi o‘zida ko‘pburchakning bitta burchagi koordinatasi (x, y) ni saqlaydi. Polygon xussusiyati esa shu nuqtalarni ketma-ket to‘g‘ri chiziqlar bilan tutashtirib chiqadi. Yozilish formati quyidagicha:

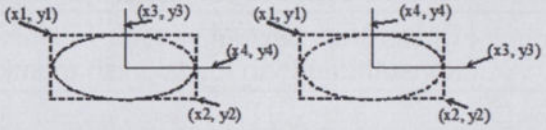
Canvas->Polygon (p, n)

Bu yerda, p – TPoint tipli massivlar majmuasi bo‘lib o‘zida ko‘pburchakning koordinatalarini o‘z ichiga oladi. n – ko‘pburchakning burchaklar soni.

6. Sektor. Ellips yoki aylana sektorini hosil qilishda Pie uslubidan foydalaniladi. Pie uslubining umumiy yozilish formati:

Canvas->Pie (x1, y1, x2, y2, x3, y3, x4, y4)

Bu yerda, x1, y1, x2, y2 – hosil bo‘luvchi sektorni davom ettirib hosil qilinuvchi ellips (aylana)ga tashqi chizilgan to‘g‘ri to‘rtburchakning mos koordinatalari; x3,y3 – sektorning boshlang‘ich nuqtasi; x4,y4 – sektorning tugash nuqtasi (12.10-rasm).

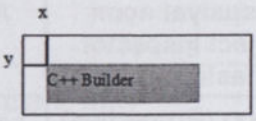


12.10-rasm. *Sektorga qiymat berish.*

7. Matn hosil qilish. Grafik obyekt sirtida matnni hosil qilish uchun TextOut usulidan foydalaniladi. TextOut ning yozilish formati quyidagicha:

Canvas->TextOut(x, y, Text);

bu yerda, x, y – matn boshlanuvchi koordinata; Text – hosil bo‘luvchi belgi kattalikdagi matn yoki satrli o‘zgaruvchi (12.11-rasm).



12.11-rasm. *Matn hosil bo‘luvchi soha koordinatasi.*

Hosil bo'luvchi matn belgilari Canvas obyektiga muvofiq keluvchi Font xususiyati orqali ifodalanadi. Font xususiyati TFont obyektiga tegishli bo'lib, 12.1-jadvalda belgi xarakteristikalari va qo'llaniluvchi uslublari keltirilgan.

12.6-jadval

Xususiyat	Aniqlanishi
Name	Foydalaniluvchi shrift. Qiymat sifatida shrift nomi yoziladi. (Masalan, Times New Roman)
Size	Punktlarda ifodalaniluvchi shrift o'lchami. Punkt-poligrafiyada qo'llaniluvchi o'lchov birligi bo'lib, u taxminan 1/72 duymga teng.
Style	Belgini yozish usuli, quyidagicha bo'lishi mumkin: oddiy, qalin, kursiv, ostiga chizilgan, ustiga chizilgan. Bular quyidagi konstantalar yordamida amalga oshiriladi: fsBold (qalin), fsItalic (kursiv), fsUnderline (ostiga chizilgan), fsStrikeOut (ustiga chizilgan).
Color	Belgi rangi. Qiymat sifatida TColor konstantalaridan foydalanish mumkin.

2-misol. Yuqorida keltirilgan usullardan foydalanib grafik obyektlar hosil qilinsin.

Misolni vizual muhitda dasturlash uchun 1 ta Label, 1 Image va 2 BitBtn komponentalari kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

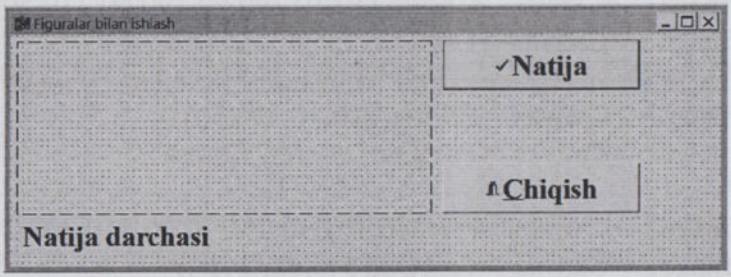
12.7-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Figuralar bilan ishlash" so'zi kiritiladi.

Label1	Caption (Properties)	"Natija darchasi" so'zi kiritiladi.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Natija" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



12.12-rasm. *Dastur ko'rinishi.*

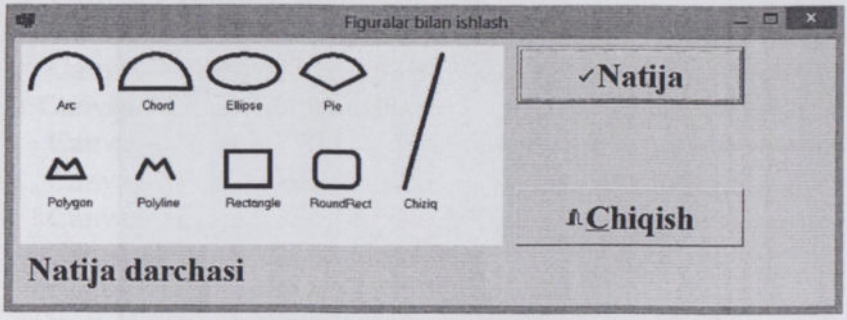
Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
```

```
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
Image1->Canvas->Font->Style << fsBold;
Image1->Canvas->Pen->Width=5;
 Image1->Canvas->Arc(10,10,90,90,90,50,10,50);
 Image1->Canvas->TextOut(40,60,"Arc");
//---------------------------------------------------------
 Image1->Canvas->Chord(110,10,190,90,190,50,110,50);
 Image1->Canvas->TextOut(135,60,"Chord");
//---------------------------------------------------------
 Image1->Canvas->Ellipse(210,10,290,50);
 Image1->Canvas->TextOut(230,60,"Ellipse");
//---------------------------------------------------------
 Image1->Canvas->Pie(310,10,390,90,390,30,310,30);
 Image1->Canvas->TextOut(340,60,"Pie");
 TPoint points[5];
 points[0] = Point(30,150);
 points[1] = Point(40,130);
 points[2] = Point(50,140);
 points[3] = Point(60,130);
 points[4] = Point(70,150);
 Image1->Canvas->Polygon(points,4);
 Image1->Canvas->TextOut(30,170,"Polygon");
//---------------------------------------------------------
 points[0].x += 100;
 points[1].x += 100;
 points[2].x += 100;
```

```
 points[3].x += 100;
 points[4].x += 100;
 Image1->Canvas->Polyline(points,4);
 Image1->Canvas->TextOut(130,170,"Polyline");
//---------------------------------------------------------
 Image1->Canvas->Rectangle(230,120,280,160);
 Image1->Canvas->TextOut(230,170,"Rectangle");
//---------------------------------------------------------
 Image1->Canvas->RoundRect(330,120,380,160,20,20);
 Image1->Canvas->TextOut(325,170,"RoundRect");
//---------------------------------------------------------
 Image1->Canvas->MoveTo(430,160);
 Image1->Canvas->LineTo(470,10);
 Image1->Canvas->TextOut(430,170,"Chiziq");
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



12.13-rasm. *Natija oynasi.*

3-misol. Olimpiada bayrog'ini chizuvchi dastur tuzilsin.

Misolni vizual muhitda dasturlash uchun 1 ta BitBtn komponentasi kerak bo'ladi.

Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

12.8-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Grafika" so'zi kiritiladi.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Natija" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



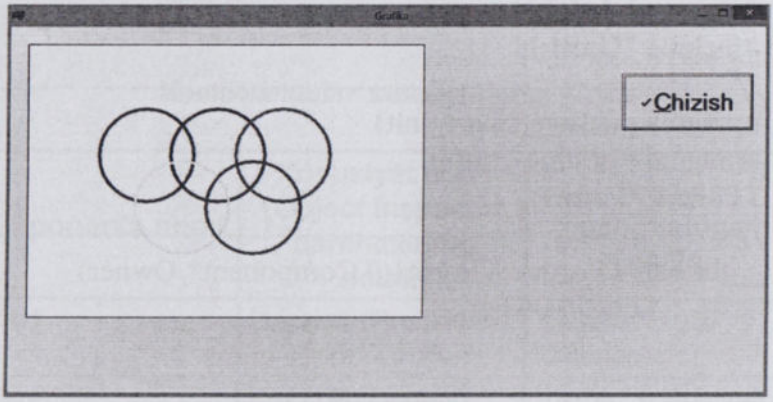
12.14-rasm. *Dastur ko'rinishi.*

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
Canvas->Pen->Width=1;
 Canvas->Pen->Color=clBlack;
 Canvas->Brush->Color =clCream;
 Canvas->Rectangle(30,30,700,500);
 Canvas->Pen->Width=5;
 Canvas->Brush-> Style=bsClear;
 Canvas->Pen->Color=clBlue;
 Canvas->Ellipse(150,140,310,300);
Canvas->Pen->Color=clBlack;
 Canvas->Ellipse(270,140,430,300);
 Canvas->Pen->Color=clRed;
 Canvas->Ellipse(380,140,540,300);
 Canvas->Pen->Color=clYellow;
 Canvas->Ellipse(210,230,370,390);
 Canvas->Pen->Color=clGreen;
 Canvas->Ellipse(330,230,490,390);
```

```
}
//---------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



12.15-rasm. *Natija oynasi.*

4-misol. $U = \cos(x)$ funksiya grafigini chizuvchi dastur tuzilsin.

Misolni vizual muhitda dasturlash uchun 1 ta BitBtn komponentasi kerak bo'ladi.

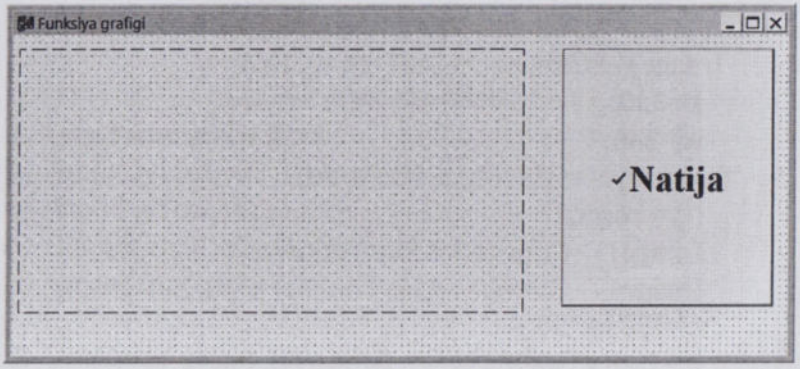
Forma darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

12.9-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Fu nksiya grafigi" so'zi kiritiladi.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Natija" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so‘ng dastur dizayni quyidagi ko‘rinishga keladi:



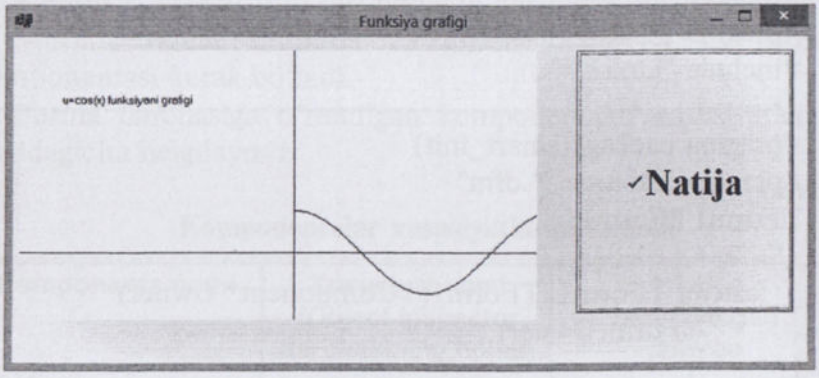
12.16-rasm. *Dastur ko‘rinishi.*

Dastur dizayni tayyor bo‘lganidan so‘ng quyidagi dastur matni kiritiladi:

```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
float y,x,e;
        int z,w;
```

```
Image1->Canvas->MoveTo(320,0);
Image1->Canvas->LineTo(320,480);
Image1->Canvas->MoveTo(0,240);
Image1->Canvas->LineTo(640,240);
  z=320;
  w=240;
for(x=0;x<=3*3.14;x+=3.14/60)
  { y=cos(x);
  Image1->Canvas->MoveTo(z,w);
  Image1->Canvas->LineTo(320+x*50,240-y*50);
  z=320+x*50;
  w=240-y*50;
  Image1->Canvas->TextOut(50,50,"u=cos(x) funksiyani
grafigi"); } }
//---------------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



12.17-rasm. *Natija oynasi.*

Nazorat savollari

1. C++ tilining grafik imkoniyatlari.
2. Tasvirlarni hosil qiluvchi funksiyalar.
3. Image komponentasining imkoniyatlari.

4. Grafik tasvirlarni hosil qilishda qalam va mo'yqalamning imkoniyatlari.

5. Grafik tasvirlarga matnlar kiritish funksiyalari.

Test savollari

1. Qaysi usul yoy chizadi?

a) Arc(x1,y1,x2,y2,x3,y3,x4,y4)
b) Arc(x1,y1,x2,y2,x3,y3)
c) Ellipse(x1,y1,x2,u2,x3,u3,x4,u4)
d) Circle(x1,y1,x2,u2,x3,u3,x4,u4)

2. Grafikada qalam (pen)ning xususiyatlari

a) bsBDiagonal, color, style
b) Canvas, image, pen
c) Color, Width, Style
d) Brush, color, style

3. C++ da chiziq chizish uchun qaysi usullardan foydalaniladi?

a) Canvas->Ellipse(x1,y1,x2,y2)
b) MoveTo(x0,y0), LineTo(x,y)
c) Canvas->Arc(x1,y1,x2,y2,x3,y3,x4,y4)
d) Canvas->Rectangle(x1,y1,x2,y2)

4. Obyektning rangi qaysi xususiyati orqali beriladi?

a) Pen
b) Style
c) Color
d) Width

5. C++ da aylana chizish uchun qaysi usuldan foydalaniladi?

a) Ellipse
b) Circle
c) Rectangle
d) Arc

XIII bob. BORLAND C++ BUILDER 6 INTEGRALLASHGAN SOHASIDA MA'LUMOTLAR BAZASINI QAYTA ISHLASH

Tayanch so'zlar: ma'lumotlar bazasi, ma'lumotlar bazasini boshqarish tizimlari, maydon, qator, ustun, relyatsion model, tarmoqli model, so'rov, SQL, Microsft Access, jadval

13.1. Ma'lumotlar bazasini tashkil qilish va ularni boshqarish

Ma'lumotlar bazasi texnologiyasi an'anaviy fayllarni tashkil etishning ko'pgina muammolarini kamaytiradi. Ma'lumotlar bazasini ta'riflaydigan bo'lsak – bu ma'lumotlar majmui bo'lib, u ma'lumotlarni samarali nazorat qilish va ma'lumotlarni markazlashtirish orqali bir nechta ilovalarni boshqarish uchun tashkil etiladi. Bunda har bir ilova uchun ma'lumotlarni alohida fayllarda saqlash o'rniga foydalanuvchi barcha ma'lumotlarni markazlashtirilgan joyda saqlaydi [6].

Ma'lumotlar bazasi (MB) – bu o'zaro bog'langan va tartiblangan ma'lumotlar majmuasi bo'lib, u ko'rilayotgan obyektlarning xususiyatini, holatini va obyektlar o'rtasidagi munosabatni ma'lum sohada tavsiflaydi.

Ma'lumotlarni saqlash, uzatish va qayta ishlash uchun ma'lumotlar bazasini yaratish, so'ngra undan keng foydalanish bugungi kunda dolzarb bo'lib qolmoqda. Katta hajmdagi ma'lumotlarni boshqarish, ulardan kerakli ma'lumotlarni so'rov orqali istalgan ko'rinishda chiqarib olish, ma'lumotlarning zaxira nusxalarini olish, katta hajmdagi ma'lumotlarni siqish, qulay interfeysda ma'lumotlar bazasi ustidan nazorat o'rnatish, ma'lumotlar asosida hisobotlar hosil qilish va bulardan boshqa ma'lumotlar ustida juda katta ko'lamdagi

[6] Kenneth C. Laudon, Jane. P. Laudon. Management Information Systems: Managing the Digital Firm, 13th Edition, Pearson Education, USA 2014. – p. 244–245.

ishlarni amalga oshiradigan dasturiy komplekslar mavjud. Bunday dasturlar majmui ma'lumotlar bazasini boshqarish tizimlari deb yuritiladi.

Ma'lumotlar bazasini boshqarish tizimlari (MBBT) – bu dasturiy ta'minot bo'lib, tashkilot ma'lumotlarini markazlashtirish imkonini beradi hamda bu orqali ma'lumotlarni samarali boshqarish va amaliy dasturiy vositalar yordamida saqlanayotgan ma'lumotlardan foydalanish imkonini beradi. MBBT amaliy dasturiy vositalar va fayllarning fizik ma'lumotlari o'rtasida interfeys vazifasini bajaradi. An'anaviy ma'lumotlar fayllari tizimidan foydalanib, dasturchi dasturda ishlatiladigan har bir ma'lumotlarning elementi hajmini va formatini belgilashi, so'ngra ular joylashgan maydonni kompyuterda belgilashi kerak bo'ladi.

MBBT – bu ko'plab foydalanuvchilar tomonidan MBni yaratish, unga qo'shimcha ma'lumotlarni kiritish va MBni birgalikda ishlatish uchun zarur bo'lgan dasturlar majmuidir. MBBTning tarkibidagi asosiy komponenti – bu ma'lumotlardir.

Ma'lumotlar bazasini boshqarish tizimi bu umumiy tushuncha bo'lib, uning tarkibiga ma'lumotlar bazasi ham kiradi. Misol uchun, mashina chiqaradigan zavod bu MBBT hisoblanadi, mashinalar esa ma'lumotlar bazasidir. MBBT umumiy bir dastur bo'lib, ma'lumotlar bazasini boshqaradi va uni bir tekis ishlashini ta'minlaydi.

MBBT ortiqcha ma'lumotlarni kamaytirib, qayta takrorlanuvchi ma'lumotlarni minimum darajagacha kamaytirish imkonini beradi. MBBT dasturlar va ma'lumotlarni ajratadi, buning natijasida ma'lumotlardan mustaqil foydalanish imkonini beradi. Ma'lumotlardan foydalanish imkoniyati oshiriladi, dasturni ishlab chiqish va qo'llab-quvvatlash uchun sarf-harajatlar kamayadi, shuningdek foydalanuvchilar va dasturchilar ma'lumotlar bazasida ma'lumotlar uchun maxsus so'rovlarini bajarishlari mumkin bo'ladi. MBBT tashkilotga ma'lumotlarni markazlashtirilgan holda boshqarish va axborot xavfsizligini oshirish imkonini beradi.

MBBTga misol qilib: Oracle MBBT, MySQL MBBT, MS SQL Server MBBT, MS Access MBBT kabilarni olish mumkin, bular ichida ma'lumotlar bazasini yaratish mumkin.

Texnik qismi tashqi qo'shimcha xotiradan iborat bo'lsa, dastur qismi esa MB bilan foydalanuvchi o'rtasidagi muloqotni tashkil qilishni amalga oshiradi. MBning tuzilishi o'rganilayotgan obyektning ma'lumotlari ko'rinishi, ma'nosi, tuzilishi va hajmiga bog'liq bo'ladi.

Hozirgi kunda, kompyuter uchun, shuningdek, server kompyuterlar uchun MBBTning eng mashhur turidan biri relyatsion model hisoblanadi. Relyatsion modellarda obyektlar va ularning o'zaro aloqalari ikki o'lchovli jadval ko'rinishida tasvirlanadi. Ma'lumotlarning bunday ko'rinishda tasvirlanishi obyektlarning o'zaro aloqalarini yaqqol tasvirlanishiga asos bo'ladi.

Misol sifatida 13.1-jadvalni keltirish mumkin.

13.1-jadval.

Relyatsion model

№	Familiyasi	Ismi	Tug'ilgan sanasi	Guruhi	Turar joyi
1.	Kadirov	Mirvohid	22.05.1992	117-16	Mirzo Ulug'bek 20 uy
2.	Kobulov	Farxod	12.02.1993	118-16	Ibn Sino 14 uy
3.	Aminov	San'at	14.05.1994	11-16	Olmazor tumani 15 uy
4.	Tolipov	Jasur	15.03.1992	22-16	Beruniy ko'chasi 22 uy

13.2. Borland C++ Builder 6 da ma'lumotlar bazasi va uni qayta ishlash

Ushbu bo'limda Borland C++ Builder 6 muhitida ma'lumotlar bazasi bilan ishlash asoslarini ko'rib o'tamiz. Borland C ++ Builder 6 dasturidan foydalanib, bitta foydalanuvchiga asoslangan

ma'lumotlar bazasini, shuningdek serverlar bilan ishlaydigan ilovalarni yaratish mumkin, bunga misol qilib, Oracle, Sybase, Informix, Interbase, MS SQL Server, DB2 va ODBC MBBTlarni keltirish mumkin. Borland C++ Builder 6 dasturining ma'lumotlar bazalarini ishlatadigan ilovalar yaratish bilan bog'liq bo'lgan imkoniyatlari juda keng.

Borland C ++ Builder 6 dasturida ma'lumotlar bazasi bilan ishlovchi juda ko'p komponentalar mavjud. Ba'zi bir komponentalar bilan tanishib chiqamiz:

-  – TTable obyekti ma'lumotlar omboridagi mavjud jadval bilan muloqot o'rnatish uchun xizmat qiladi. TTable ixtiyoriy tipdagi (FoxPro, ODBC, SQL va hakazo) ma'lumotlar bazasining har bir yozuviga va maydoniga to'g'ridan to'g'ri murojaat qila oladi. Bu komponent shuningdek, alohida hisobotlar bilan ham muloqot o'rnata oladi.

TTable obyektidan foydalanishdan oldin unga ma'lumotlar bazasini ulash kerak, ya'ni shu komponentning DatabaseName xususiyatida chiqadigan ro'yxatdan kerakli jadvalni tanlash va TableName xususiyatidagi ro'yxatdan kerakli jadval nomini tanlash kerak. TTable obyektining bir necha xususiyatlarini ko'rib chiqamiz:

- **Active** – xususiyati ikkita qiymat qabul qiladi, agar "true" qiymat bo'lsa jadval faol hisoblanadi, "false" qiymat bo'lsa jadval yopiqligini bildiradi.
- **DatabaseName** – kerak bo'lgan jadval joylashgan katalog nomi yoki masofada joylashgan ma'lumotlar bazasining tashqi xotiraga yozilgan qo'shimcha nomi (alias). "Alias" qo'shimcha dastur BDE konfiguratsiyasi yordamida yoki Database/Explore bosh menyusida joylashgan SQL Explorer yordamida o'rnatiladi. Bu xususiyat faqat jadval yopilgan (uning "active" xususiyati "false") bo'lsa, o'zgarishi mumkin, masalan:

```
Table1->Active = false;
Table1->DatabaseName = "BCDEMOS"
Table1->Active = true;
```

• **TableName** – jadvalning nomi.

• **Exclusive** – agar bu xususiyat "true" bo'lsa, ushbu joriy dastur tomonidan ochilgan bo'lsa, boshqa hech bir foydalanuvchi jadvalni ocholmaydi. Agar bu xususiyat "false" bo'lsa (asl qiymati), boshqa foydalanuvchilar ushbu jadvalni ochishlari mumkin bo'ladi.

• **IndexName** – jadval uchun ikkilamchi indeksni aniqlaydi. Jadval ochiq holatda bo'lganda bu xususiyatni o'zgartirib bo'lmaydi.

• **MasterFields** – boshqa jadvalga havola yaratish uchun maydon nomini belgilaydi.

• **MasterSource** – TDataSource komponentasining nomi bo'lib, TTable komponentasi bog'langan jadvaldan ma'lumotlarni olish imkonini beradi.

• **Fields** – TField obyektlar massivi bo'lib, ushbu xususiyatdan maydonning tartib raqamiga murojaat etish mumkin. Misol uchun:

Edit1->Text=Table1->Fields[2]->AsString;

· TADOTable obyekti ham xuddi TTable obyekti kabi ma'lumotlar bazasidagi biror jadvalga bog'lanish va unga murojaat qilish uchun xizmat qiladi. Bu obyektdan asosan MSAccess ma'lumotlar bazasini boshqarish tizimida yaratilgan ma'lumotlar bazasi bilan ishlashga mo'ljallangan. Bu obyekt asosan TADOconnection obyekti bilan birga qo'llanilib, TADOConnection ma'lumotlar bazasiga ulanadi. Shundan so'ng bir yoki bir nechta TADOTable obyektlari Connection xususiyati yordamida TADOConnectionga ulanadi va TableName xususiyati yordamida kerakli jadval bilan bog'lanadi. Obyektni faollashtirish uchun "Active" xususiyati qiymatini "true"ga o'tkazish kerak. Bu obyekt yordamida ma'lumotlar bazasidan ma'lumotlarni biror filtr yordamida ajratib olish mumkin.

TADOTable obyektining bir necha xususiyatlarini ko'rib chiqamiz:

• **Active** – xususiyati ikkita qiymat qabul qiladi, agar "true" qiymat bo'lsa jadval faol hisoblanadi, "false" qiymat bo'lsa jadval yopiqligini bildiradi.

• **ReadOnly** – xususiyati agar "true" qiymat qabul qilsa jadval "Faqat o'qish uchun" rejimi faol bo'ladi. Jadval ochiq bo'lganda ReadOnly xususiyatini o'zgartirib bo'lmaydi.

• **TableName** – jadvalning nomi.

• **Connection** – jadvaldagi ma'lumotlar bazasi bilan bog'lanish uchun xizmat qiladi.

• **Filter** – jadvaldagi ma'lumotlar bazasini saralash imkonini beradi. Misol uchun, saralashni amalga oshirish uchun quyidagi dastur kodini kiritish mumkin:

```
ADOTable1->Filtered = false;
ADOTable1->Filter = "Name like 'C*'";
ADOTable1->Filtered = true;
```

• **Name** – komponentaning nomi.

 TDataSource obyekti bevosita TTable yoki TAdoTablega bog'lanib, ma'lumotlar bazasidagi yozuvlarni tahrirlash, ularga murojaat qilish imkonini beradi. Buning uchun komponentaning DataSet xususiyatidagi ro'yxatdan kerakli Table elementi tanlanadi va shu orqali ikki obyekt bir-biriga bog'lanadi. TDataSource obyekti bitta ma'lumotlar bazasidagi bitta jadvalga ulana oladi.

Yuqoridagi uchala obyekt ham dastur bajarilish vaqtida ko'rinmaydigan obyekt bo'lib, Formalar Dizayneri ko'rinishida ularni formaga tashlaganda o'lchamlarni o'zgartirib bo'lmaydi. Ularni ma'lumotlar omboriga Formalar Dizayneri rejimida ham, dasturiy yo'l bilan dastur bajarilish vaqtida ham bog'lash mumkin.

Buning uchun quyidagicha kodlar yoziladi:

```
{
Table1->DatabaseName = "DBDEMOS";
 Table1->TableName = "animals.dbf";
 Table1->Active = True;
 DataSource1->DataSet = Table1;
 DBGrid1->DataSource = DataSource1;
}
```

 TDBgrid obyekti ma'lumotlar bazasidagi hisobotlar, jadvallar va so'rovlardagi ma'lumotlarni jadval ko'rinishida

namoyish etish uchun qo'llanadi. Bu obyekt yordamida ma'lumotlar bazasidagi yozuvlarni namoyish qilish, tahrirlash va o'zgartirish mumkin. Kiritilgan o'zgartirishlar joriy yozuv ustida boradi va bu o'zgarishlar faqat boshqa yozuvga o'tganda yoki dastur yopilganida saqlab qolinadi. TDBgrid obyekti bevosita DataSource xususiyati yordamida TDataSource obyektga bog'lanadi va shu orqali ma'lumotlarni namoyish etadi.

 TDBNavigator (QDBCtrls) obyekti dasturda TDBgrid yoki TDBedit komponenta orqali ma'lumotlar bazasi yozuvlariga murojaat qilinayotgan vaqtda qo'llaniladi. TDBNavigator foydalanuvchiga ma'lumotlar bazasidagi yozuvlarni tahrirlash yoki ko'rib chiqishda foydalaniladi. Foydalanuvchi TDBNavigator tugmalardan birini bosganda shu tugma bilan bog'langan amal dasturda bajariladi.

Quyidagi 13.2-jadvalda TDBNavigator obyektining tugmalari va ular bajaradigan amallar ko'rsatib o'tilgan.

13.2- jadval.

TDBNavigator obyektining tugmalari

Tugma	Bajaradigan amali
 First	Ma'lumotlar bazasidagi dastlabki yozuvni faollashtirish. U faqat joriy yozuv dastlabki yozuv bo'lmagandagina faol bo'ladi.
 Prior	Ma'lumotlar bazasida joriy yozuvdan oldingi yozuvni faollashtirish. U faqat joriy yozuv dastlabki yozuv bo'lmagandagina faol bo'ladi.
 Next	Ma'lumotlar bazasida joriy yozuvdan keyingi yozuvni faollashtirish. U faqat joriy yozuv oxirgi yozuv bo'lmagandagina faol bo'ladi.
 Last	Ma'lumotlar omboridagi oxirgi yozuvni faollashtirish. U faqat joriy yozuv oxirgi yozuv bo'lmagandagina faol bo'ladi.
 Insert	Jadvalga ma'lumotlarni kiritish uchun yangi satr qo'shish. Bunda satrning ixtiyoriy maydoniga ma'lumot kiritilganda o'zgarishlar saqlanadi.

 Delete	Joriy yozuvni o'chirish. Bunda yozuvni o'chirish haqida so'rov chiqariladi va o'chirilgan yozuv qayta tiklanmaydi.
 Edit	Joriy yozuvni o'zgartirish, tahrirlash mumkin bo'lgan holatga o'tkazish.
 Post	Kiritilgan o'zgaishlarni xotirada saqlash. Bunda joriy maydonning oldingi ma'lumotlari o'rniga kiritilgan o'zgarishlar saqlanadi.
 Cancel	Joriy yozuvga kiritilgan o'zgarishlarni bekor qilish. Bu amaldan joriy yozuvni almashtirguncha foydalanish mumkin.
 Refresh	Kiritilgan ma'lumotlarni yangilash.

13.3. Texnik tizimlarda ma'lumotlar bazasini qayta ishlash

Yuqorida ko'rib chiqilgan komponentalar asosida texnik tizimlarda ma'lumotlar bazasini qayta ishlash jarayonini ko'rib chiqamiz.

1-misol. MS Access dasturida yaratilgan ma'lumotlar bazasidan foydalanib, forma darchasida guruhining ma'lumotlar bazasini qayta ishlash dasturi ishlab chiqilsin.

Misolni vizual muhitda dasturlash uchun 1 ta ADOConnection, 1 ta ADOTable, 1 ta DataSourse, 1 ta DBNavigator, 1 DbGrid, 3 ta Label, 2 ta Edit va 2 BitBtn komponentalari kerak bo'ladi. Dastur 2 ta Form darchasida amalga oshiriladi.

Forma1 darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

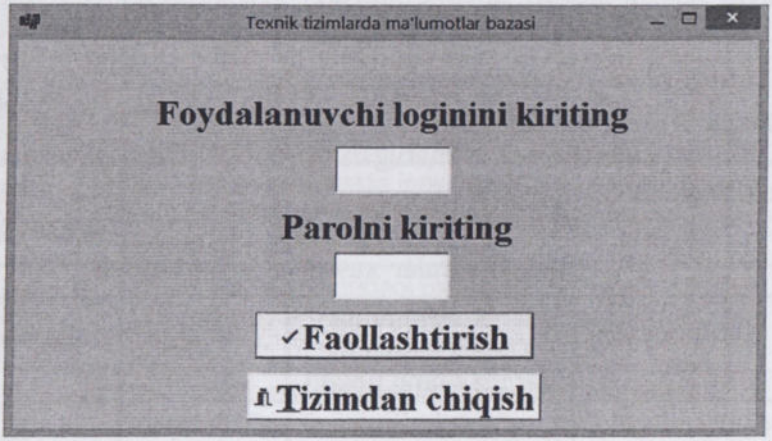
13.3-jadval.

Komponentalar xususiyatlarini kiritish

Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Texnik tizimlarda ma'lumotlar bazasi" so'zi kiritiladi.

Label1	Caption (Properties)	"Foydalanuvchi loginini kiriting" so'zi kiritiladi.
Label2	Caption (Properties)	"Parolni kiriting" so'zi kiritiladi.
Edit1	Text (Properties)	"Edit1" so'zini o'chirib tashlang.
Edit2	Text (Properties)	"Edit2" so'zini o'chirib tashlang.
BitBtn1	Kind (Properties)	"bkOK" xususiyati tanlanadi.
	Caption (Properties)	"Faollashtirish" so'zi kiritiladi.
	OnClick (Events)	Dastur matni kiritiladi.
BitBtn2	Kind (Properties)	"bkClose" xususiyati tanlanadi.
	Caption (Properties)	"Tizimdan chiqish" so'zi kiritiladi.
	OnClick (Events)	Close(); kiritiladi.

Mavjud komponentalar xususiyatlari kiritilgandan so'ng dastur dizayni quyidagi ko'rinishga keladi:



13.1-rasm. *Dastur ko'rinishi.*

Form 2 darchasiga o'rnatilgan komponentalar xususiyatlarini quyidagicha belgilaymiz:

13.4-jadval.

Komponentalar xususiyatlarini kiritish

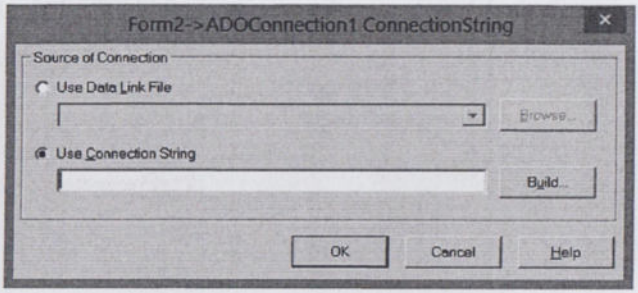
Komponenta nomi	Xususiyat nomi (Object Inspector darchasining holati)	Amalga oshiriladigan jarayon
Form1	Caption (Properties)	"Ma'lumotlar bazasi" so'zi kiritiladi.
Label1	Caption (Properties)	"Ma'lumotlar bazasini qayta ishlash" so'zi kiritiladi.
AdoConnection	ConnectionString (Properties)	Use Connection String→Build→ MicrosoftJet 4.0 OLE DB Provider→далее→ Baza.mdb→Ok
	LoginPrompt (Properties)	false
ADOTable	Connection (Properties)	Connection1
	TableName (Properties)	Jadval
	Active (Properties)	True
DataSourse	Dataset (Properties)	ADOTablel1
DBGrid	DataSourse (Properties)	DataSourse1
DBNavigator	DataSourse (Properties)	DataSourse1



13.2-rasm. *Dastur ko'rinishi.*

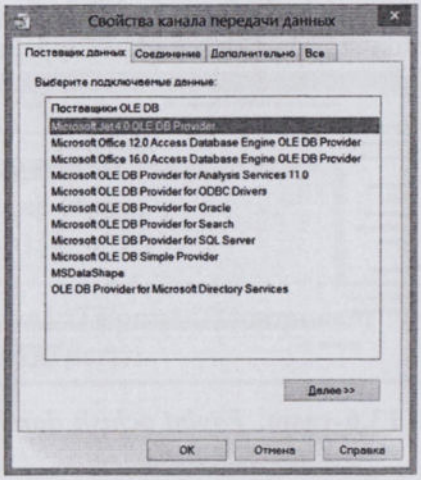
Kerakli komponentalar joylashtirilgandan so'ng komponentalar ustida quyidagi amallarni bajaramiz:

AdoConnection obyektining ConnectionString xususiyatiga kiriladi va mavjud ma'lumotlar bazasi bilan bog'lanadi. Buning uchun Microsoft Access dasturida "Baza.mdb" nomi bilan jadval yaratamiz. Faylni loyiha yaratilgan joriy katalogda saqlaymiz. Faylga bog'lash quyidagicha amalga oshiriladi (13.3-rasm):



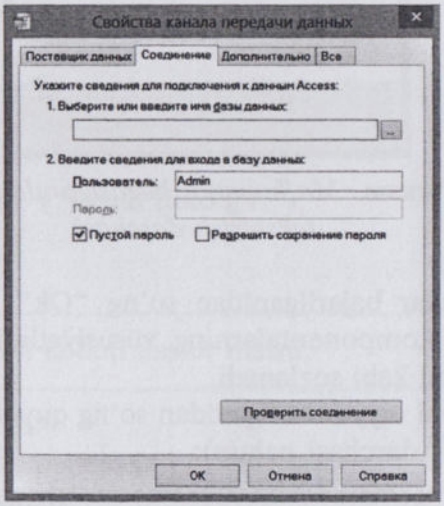
13.3-rasm. *ConnectionString xususiyati.*

Hosil bo'lgan darchadan "Build" tugmasini bosamiz. Natijada bog'lanish imkonini beruvchi ro'yxat paydo bo'ladi. Ro'yxatdan MS Aceess dasturi bilan ishlovchi Microsoft Jet OLEDB 4.0 Provider tanlaymiz (13.4-rasm):



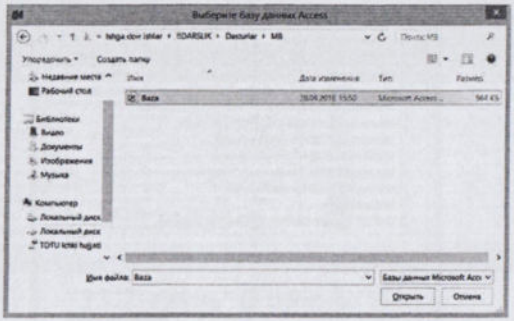
13.4-rasm. *Microsoft Jet OLE DB 4.0 Provider tanlash darchasi.*

Microsoft Jet OLE DB 4.0 Provider tanlangandan so‘ng “Далее” tugmasi bosiladi. Hosil bo‘lgan darchada ma’lumotlar bazasi bilan bog‘lanishni amalga oshirish mumkin bo‘ladi. Buning uchun ... tugmasi bosiladi va MS Accesda yaratilgan fayl ko‘rsatiladi.



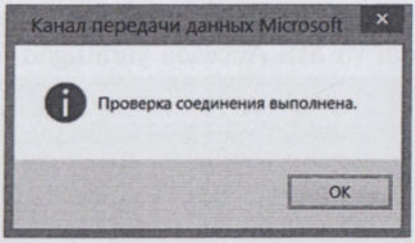
13.5-rasm. *Ma’lumotlarni jo‘natish xususiyatlari darchasi.*

Ma'lumotlar bazasi joylashgan faylni tanlash uchun Windows operatsion tizimida standart darchadan foydalaniladi.



13.6-rasm. *Faylni ochish darchasi.*

Fayl tanlangandan so'ng "Проверить подключение" tugmasi bilan tekshiriladi. Agarda "Проверка подключение выполнена" natijasi chiqsa unda dastur ma'lumotlar bazasiga ulangan sanaladi.



13.7-rasm. *Ma'lumotlar bazasiga ulanganligini tekshirish darchasi.*

Barcha amallar bajarilganidan so'ng "Ok" tugmasi bosladi. Qolgan barcha komponentalarning xususiyatlari 13.3- va 13.4-jadvalda berilgani kabi sozlanadi.

Dastur dizayni tayyor bo'lganidan so'ng quyidagi dastur matni kiritiladi (Form1 darchasi uchun):

```
//---------------------------------------------------------
#include <vcl.h>
```

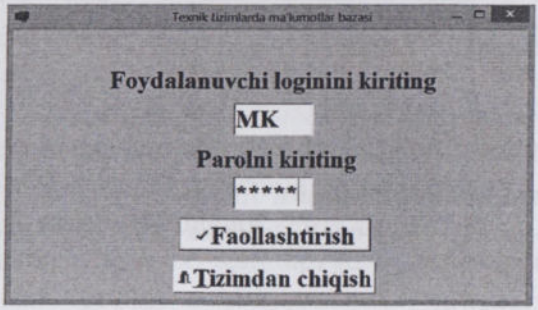
```
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
if (Edit1->Text == "MK")
{
      if (Edit2->Text == "12345")
{
            Form2->ShowModal();
    } else
{ ShowMessage ("parol noto'g'ri");
}
} else
{ ShowMessage ("Login noto'g'ri");
}
}
//---------------------------------------------------------
```

Form2 darchasi uchun dastur matni:

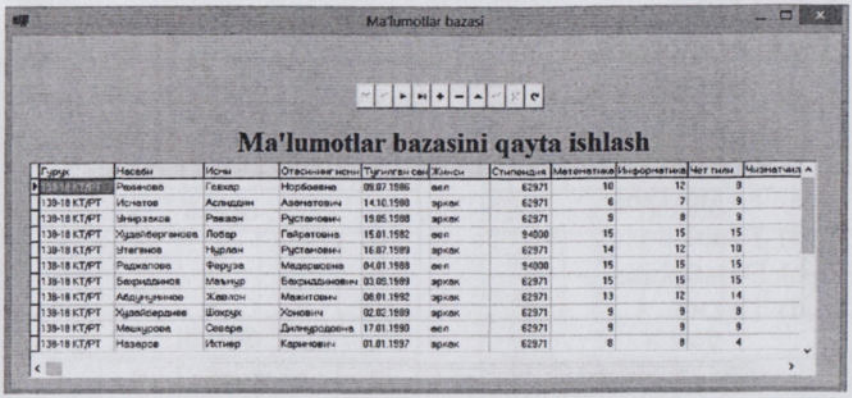
```
//---------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
//---------------------------------------------------------
```

```
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
void __fastcall TForm2::DBGrid1TitleClick(TColumn
*Column)
{
if (ADOTable1->Active)
 if ((ADOTable1->Sort.Pos(Column->FieldName) > 0) &&
(ADOTable1->Sort.Pos("ASC") > 0))
{
ADOTable1->Sort = Column->FieldName + " DESC";
}else
{
ADOTable1->Sort = Column->FieldName + " ASC";
}
}
//---------------------------------------------------------
```

Dastur matni kiritib bo'lingandan so'ng F9 tugmasi bosiladi va natijada quyidagi dastur ko'rinishi paydo bo'ladi:



13.8-rasm. *Natija oynasi.*



13.9-rasm. *Natija oynasi.*

Nazorat savollari

1. Ma'lumotlar bazasi nima?
2. C++ Builder 6 muhitida ma'lumotlar bazasi bilan ishlovchi qanday komponentalar mavjud?
3. AdoConnection komponentasining vazifasi?
4. DBNavigator komponentasining vazifasi?
5. DBgrid komponentasining vazifasi?

Test savollari

1. AdoConnection obyektiga MS Acess dasturini bog'lash uchun qaysi xossasidan foydalaniladi?

a) ConnectionString
b) LoginPrompt
c) DataSourse
d) Connections

2. TDataSource obyekti qaysi komponentalar menyusida joylashgan?

a) DataControls

b) ADO
c) DataAccess
d) DataSourse

3.  qaysi komponenta?
a) TADOTable
b) TTable
c) DataSourse
d) TADO

4.  qaysi komponenta?
a) TADOTable
b) DataSourse
c) TTable
d) DBNavigator

5.  qaysi komponenta?
a) TDBgrid
b) DataSourse
c) TTable
d) TADOTable

Boblar bo‘yicha test savollarining javoblari

I bob				
1	2	3	4	5
a	b	b	c	d
II bob				
1	2	3	4	5
b	b	a	c	c
III bob				
1	2	3	4	5
a	c	d	a	b
IV bob				
1	2	3	4	5
d	c	b	a	a
V bob				
1	2	3	4	5
c	b	a	d	c
VI bob				
1	2	3	4	5
b	c	d	a	b
VII bob				
1	2	3	4	5
d	a	c	a	c

VIII bob				
1	2	3	4	5
a	a	b	b	c
IX bob				
1	2	3	4	5
d	b	c	b	d
X bob				
1	2	3	4	5
a	b	c	d	a
XI bob				
1	2	3	4	5
b	c	b	c	d
XII bob				
1	2	3	4	5
a	c	b	c	a
XIII bob				
1	2	3	4	5
a	c	b	a	a

GLOSSARIY

A

Algoritm – amallarning cheklangan soni yordamida masala yechimini belgilovchi buyruqlarning to'plami. Dasturlashda algoritmlar psevdokod, blok-shema va UML diagramma vositalari ko'rinishida tasvirlanadi.

Algorithm *– the order of actions that must be performed to solve a particular task. In programming, algorithms are described by means of pseudocode, flowcharts and UML diagrams.*

Amaliy dasturlash – muayyan faoliyat sohasidagi amaliy muammolarni hal qilish uchun mo'ljallangan dasturiy ta'minotni ishlab chiqish jarayoni. Ushbu dasturlashni amaliy deyilishining sababi, u apparat resurslaridan bevosita foydalanmaydi va uni operatsion tizim orqali amalga oshiradi.

Applied programming *– the process of developing software designed to solve applied problems in a certain field of activity. Such software is called application software, and it is characterized by the fact that it does not use the hardware resources of the hardware directly, but does it through the operating system.*

Amaliy dasturlash interfeysi – dasturlash ilovalari interfeysi bo'lib, dasturlash darajasida dasturning tashqi ko'rinishi xususiyatlarini o'z ichiga oladi.

Application programming interface (API) *– application programming interface, available at the software level to external software components.*

Assembler – ko'rsatmalari mashina kodi ko'rsatmalariga mos keladigan past darajadagi dasturlash tili. Bundan tashqari, assembler dasturi deb, past darajadagi dasturlash tilidan kompyuter kodiga aylantiruvchi dasturga aytiladi.

Assembler *– a low-level programming language, the instructions of which correspond to the instructions of the machine code. In addition, an assembler is called a program – a translator from a low-level programming language into machine code.*

Axborot tizimi – foydalanuvchilarni axborot bilan ta'minlash maqsadida yaratilgan ma'lumotlar to'plami va ushbu ma'lumotlarga xizmat ko'rsatuvchi texnik, dasturiy va tashkiliy resurslar.

An information system *– is a collection of data and technical, program and organizational resources serving this data, created for the purpose of informational support of users.*

B

Blok-shema – algoritmlarni tavsiflash uchun grafik yozuv. Dasturiy komponentalarni ishlab chiqishda va ishlashini mantiqiy tahlil qilish jarayonida dasturchilar tomonidan qo'llaniladi.

Flowchart *– graphical notation for the description of algorithms. Used by programmers in the process of developing and analyzing the logic of the operation of software components.*

Boshqariladigan kod – virtual mashina tomonidan bajariladigan dastur kodi.

Managed code *– the program code that is executed by the virtual machine.*

Bulutli hisoblash – hisoblashni tashkil etish modeli bo'lib, bunda mijoz tomonidan amalga oshiriladigan hisoblash xizmatlari mijoz resurslaridan ko'ra yuqori imkoniyatga ega tarmoq texnologiyalarida bajariladi. Server – mijoz o'rtasidagi aloqa tarmoqqa ulanish yo'li orqali amalga oshiriladi. Bu texnologiya mijozlarga barcha hisoblash xizmatlari uning tomonida amalga oshayotgani kabi holatda bo'ladi.

Cloud computing *– a model of computing, in which the computing processes requested by the client, occur on remote, much more powerful compared to client computing resources. The interaction of the server with the client is carried out through network access, but the process of computing for the user is inseparable – as if everything is happening on the client side.*

D

Dastur – muallif tomonidan ishlab chiqilgan va yaratilgan tizimda ishga tushurish mumkin bo'lgan tugallangan mahsulot.

The program – *a complete product, suitable for running by its author on the system on which it was developed.*

Dasturiy mahsulot – ixtiyoriy foydalanuvchi tomonidan ishga tushuruluvchi, to'g'rilanuvchi, rivojlantiriluvchi va testlash imkoniyati mavjud dastur.

Software product – *a program that any person can launch, test, correct and develop.*

Dasturiy majmua – funksiyalar va formatlar bo'yicha muvofiqlashtirilgan interaktiv dasturlarning to'plami, muayyan interfeysga ega va birgalikda katta vazifalarni hal etish uchun to'liq vosita bo'lib xizmat qiladi.

Software package – *a set of interacting programs that are coordinated by functions and formats, precisely defined interfaces, and together constitute a complete tool for solving large tasks.*

Delphi – obyektga yo'naltirilgan dasturlash tili, Pascal dasturlash tili asosida yaratilgan. Dasturiy ta'minotni ishlab chiqish muhiti Borland kompaniyasiga tegishli.

Delphi – *an object-oriented programming language based on the Pascal programming language and the Borland software development environment.*

E

ER diagrammasi – ER modelining ma'lumotlarini vizuallashtirishning grafik belgilari.

ER diagrams – *graphical notation of data visualization ER model.*

ER model (Mohiyat-Munosabat) – ma'lum sohada relyatsion ma'lumotlar bazasini loyihalash uchun foydalaniladigan model. U o'z ichiga ma'lumotlarning mohiyati va ularning o'zaro munosabatlarini qamrab oladi.

ER model (Entity Relationship) *– a domain data model used to design relational databases in terms of entities and the relationships between them.*

F

Foydalanuvchining grafik interfeysi (FGI) – Windows (Microsoft), Mac OS (Apple) kabi zamonaviy operatsion tizimlar tomonidan taqdim etilgan grafik foydalanuvchi interfeysi. FGI sichqoncha va klaviatura yordamida manipulyatsiya qilinadigan grafik oynalar, tugmalar, ro'yxatlar va boshqa boshqaruv elementlari bilan ifodalanadi.

Graphical User Interface (GUI) *– is a graphical user interface provided by modern operating systems, such as Windows (Microsoft), Mac OS (Apple), etc. GUI is represented by graphic windows, buttons, lists and other controls, which are manipulated by mouse and keyboard.*

Freymvork – turli dasturiy ta'minot mahsulotlarining asosi bo'lgan dasturiy ta'minot turi hisoblanadi.

Framework *– a kind of software, which is the basis (framework) of various application software products.*

G

Geoaxborot tizimi – ma'lumotlar tizimi, uning vazifalari orasiga joylarning geografik ma'lumotlarni saqlash, grafik ko'rinishini ta'minlash va ma'lumotlardan foydalanishni nazorat qilishni o'z ichiga oladi.

Geoinformation system (GIS) *– is an information system, whose tasks also include storage, graphical display and control of access to spatial (geographic) data.*

X

Xotirani dinamik taqsimlash – dasturni bajarish vaqtida xotirani ajratishni va bo'shatishni nazarda tutadigan xotira resurslarini boshqarish.

Dynamic memory allocation *– memory resource management, which involves allocation, freeing up memory during program execution.*

HTML (HyperText Markup Language) – Veb-sahifalar uchun belgilash tili bo'lib, ularni tashkil etuvchilarini formatlash uchun mo'ljallangan. Internet brauzerlar orqali namoyish etiladi.

HTML (HyperText Markup Language) *– a Web page markup language designed to format their content (content) displayed by Internet browsers.*

I

Inkapsulyatsiya – axborotni yashirish, shuningdek obyekt ichidagi ma'lumotlar va funksiyalarni (usullarni) birlashtirish.

Encapsulation *– is the hiding of information and the combination of data and functions (methods) within an object.*

J

Java – obyektga yo'naltirilgan dasturlash tili. Sun Microsystems kompaniyasi tomonidan ishlab chiqilgan.

Java *– an object–oriented programming language developed by Sun Microsystems.*

JavaScript – Internet-brauzerga o'rnatilgan protsessual dasturlash tili. JavaScript tilining maqsadi HTML formatlash elementlari yuklangan Veb-sahifa obyekt modelining dastur interfeyslari orqali ishlov berish.

JavaScript *– is a procedural programming language built into the Internet browser. The purpose of JavaScript is to manipulate HTML markup elements through the program interfaces of the object model of the loaded web page.*

jQuery – biblioteka hisoblanib, JavaScript dasturlash tilida yaratilgan. Maqsadi dinamik Internet sahifalarni yaratishda HTML belgilash tili bilan ishlash va uni boshqarishni soddalashtirish uchun qo'llaniladi.

jQuery *– is a library written in JavaScript and created to simplify*

interaction with HTML markup elements when creating dynamic web pages. jQuery provides a program interface for querying the data of the object model of the loaded page with the ability to perform a series of operations on the results of these queries.

K

Kodni tekshirish – dasturiy ta'minot kodini muntazam va tizimli tahlil qilish, dasturiy ta'minotni ishlab chiqishning dastlabki bosqichlaridan xatoliklarni aniqlash, shuningdek, sifatsiz yechimlarni va dasturdagi tanqidiy joylarni aniqlash.

Code review *– is a systematic and periodic analysis of program code aimed at searching for undetected errors in the early stages of software product development, and also for identifying poor-quality architectural solutions and critical locations in the program.*

Kutubxona – dasturlarda ishlatiladigan proseduralarni, ichki dasturlarni, funksiyalarni, makroslarni va boshqa ma'lumotlarni saqlaydigan fayl yoki fayllar to'plami.

Library *– in programming a file or a collection of files in which procedures, subroutines, functions, macros and other data used by programs are stored.*

Kodni qayta ishlash – dastur kodiga ma'lum qoidalar asosida o'zgartirishlar kiritish jarayoni. Qoidalarga muvofiq dasturning ma'nosini o'zgartirmasdan dastur kodini takomillashtirish imkonini beradi bu esa inson uchun dastur kodini oson va qulay qiladi.

Refactoring *– the process of making changes to the code in accordance with some set of rules – refactoring techniques that do not change the meaning of the program, but make its code more slim and easy to interpret by the person.*

M

Manba kodi – algoritmik tilda dasturning matni. Kompyuterda dastlabki matn kodi to'g'ridan to'g'ri interpretatorda bajariladi yoki oldin kompilyator tomonidan muayyan hisob-kitob muhitida

takroriy bajarilishi mumkin bo'lgan standart yuklovchi kodga kompilyator yordamida tarjima qilinadi.

Source code *– the text of the program in an algorithmic language. In the computer, the source text is either directly executed by the interpreter, or it is previously translated by the compiler into a standard boot code that can be repeatedly executed in a certain computing environment.*

Mashina tili – kompyuter buyruqlari asosiy elementlarini tashkil qiluvchi dasturlash tili.

Machine language *– is a programming language, the elements of which are computer commands.*

Moslashtirish – dasturiy mahsulotlarni ishlab chiqish nuqtayi nazaridan dasturiy mahsulotning funksional imkoniyatlarini foydalanuvchining talablariga moslashtirish jarayoni sifatida nazarda tutiladi.

Customization *– in the context of software development can mean the process of setting the functionality of the software product to the requirements of the end user.*

Modulli dasturlash – bu dasturlashning shunday usuli bo'lib, bunda dastur modul deb ataladigan tarkibiy qismlarga bo'linadi va ularning har biri nazorat qilinadigan o'lchamlari, aniq maqsadi va tashqi muhit bilan ishlovchi batafsil interfeysga ega bo'ladi.

Modular programming *– is a method of programming in which the whole program is divided into a group of components called modules, each of which has its own controlled size, a clear purpose and a detailed interface with the external environment.*

Modul – 1) buyruqlar to'plami bo'lib, nomi bo'yicha murojaat etish imkonini yaratadi;

2) dasturning operatorlari to'plami bo'lib, chegaralangan elementlar va indentifikatorga ega.

Module *– 1) is a set of commands, which can be accessed by name;*

2) it is a collection of program statements that has boundary elements and an identifier.

Microsoft Visual Studio – dasturiy mahsulotlarni ishlab chiqishga mo'ljallangan integrallashgan muhit bo'lib, Microsoft kompaniyasining mahsuloti hisoblanadi. Microsoft.NET Framework platformasiga mo'ljallangan dasturlash tillarini qo'llab-quvvatlaydi.

Microsoft Visual Studio *– is an integrated development environment for Microsoft software products, which, among other things, supports programming languages for the Microsoft .NET Framework.*

Microsoft .NET Framework – CLR virtual mashinasi tomonidan foydalaniladigan platformalarga bog'liq bo'lmagan mustaqil dasturlarni ishlab chiqishga mo'ljallangan Microsoft kompaniyasining eng so'nggi dasturiy ta'minot texnologiyalaridan biri.

Microsoft .NET Framework *– is one of the latest Microsoft software technologies designed to develop platform-independent applications run by the CLR virtual machine.*

Ma'lumotlar bazasini boshqarish tizimi (MBBT) – ma'lumotlar bazasini boshqarish bo'yicha barcha jarayonlarni o'z ichiga olgan dasturiy ta'minot (axborot tizimi). Bunday jarayonlarga ma'lumotlar bazasini saqlash, SQL yordamida qayta ishlash, zaxira nushalarini ko'paytirish, zaxira nushalarini qayta tiklash va boshqalarni o'z ichiga olgan axborot tizimi.

Database Management System (DBMS) *– is a software (information system) that performs the whole spectrum of database management operations, which include the data storage organization itself, processing of SQL instructions, organizing backups, restoring backup copies, and so on.*

O

Obeyktga yo'nalatirilgan ma'lumotlar bazasi – obyektga yo'naltirilgan ma'lumotlar modeliga asoslangan ma'lumotlar bazasi. Obyektga yo'naltirilgan modelning asosiy elementlari: sinflar, obyektlar, interfeyslar, atributlar (xususiyatlar), usullar va boshqalar.

Object–oriented database is a database based on an object–oriented data model. The main elements of the object-oriented model are classes, objects, interfaces, attributes (properties), methods, and so on.

Obeyktga yo'nalatirilgan dasturlash – dasturni obyektlar to'plami sifatida ko'rsatishga mo'ljallangan dasturlash metodologiyasi bo'lib, ularning har biri ma'lum bir sinfning namunasi sifatida ko'riladi. Sinflar merosning iyerarxiyasini tashkil qiladi.

Object oriented programming (OOP) – *programming methodolgy based on the representation of the program in the form of a collection of objects, each of which is an instance of a particular class, and the classes form a hierarchy of inheritance.*

P

Pascal – dasturlashni o'rgatishda keng qo'llaniladigan yuqori darajali dasturlash tili.

Pascal – *one of the most famous high-level programming languages that is widely used for teaching programming.*

R

Relyatsion ma'lumotlar bazasi – ma'lumotlarning relyatsion modeliga asoslangan ma'lumotlar bazasi.

Relational database – *a database based on a relational data model.*

S

SQL – ma'lumotlar bazasiga tuzilgan tizimli so'rovlarni tavsiflash tili bo'lib, natija sifatida tuzilgan ma'lumotlar to'plami yoki ma'lumotlar tarkibidagi o'zgarishlarni amalga oshirish uchun mo'ljallangan. Undan tashqari SQL so'rovlari yordamida ma'lumotlar bazasi tuzilishini o'zgartirish, kirish parametrlarini turli bo'limlarga sozlash va tranzaksiyalarni boshqarish imkonini beradi.

SQL – *a language for describing structured queries to databases, the result of which can be either a structured set of information or changes in the composition of the data. Also, SQL statements*

allow you to change the structure of a database, configure access parameters to its various sections and manage transactions.

C++ – Bjorn Stroustrup tomonidan ishlab chiqilgan obyektga yo'naltirilgan dasturlash tili.

C ++ *– is an object-oriented programming language developed by Bjorn Stroustrup.*

T

Tizim dasturchisi – tizimli dasturlarni ishlab chiqish, ishlatish va ulardan foydalanishga xizmat ko'rsatish bilan shug'ullanadigan mutaxassis.

The system programmer *– is a specialist, engaged in the development, operation and maintenance of system software.*

V

Vizual dasturlash – ko'rgazmali vositalar yordamida dasturiy ta'minotlarni ishlab chiqishga mo'ljallangan dasturlash.

Visual programming *– programming, providing for the creation of applications using visual aids.*

Veb dasturlash – Internet tarmog'i uchun veb ilovalarni yaratishga mo'ljallangan dasturlash yo'nalishi. Foydalanuvchi veb ilovalarga Internet brauzer orqali o'zaro boglanadi.

Web programming *– a direction in programming, focused on the development of applications for the Internet (web applications). The user interacts with the web application via an Internet browser.*

Y

Yuqori darajali dasturlash tili – inson qabul qilishi uchun qulay tushuncha va tuzulishga ega bo'lgan dasturlash tili.

High level language *– is a programming language, the concepts and structure of which are convenient for human perception.*

FOYDALANILGAN ADABIYOTLAR RO'YXATI

1. Mirziyoyev Sh.M. "Tanqidiy tahlil, qat'iy tartib-intizom va shaxsiy javobgarlik – har bir rahbar faoliyatining kundalik qoidasi bo'lishi kerak". – Toshkent: "O'zbekiston", 2017-yil.

2. Mark Lewin. Go Web Development Succinctly. Syncfusion Inc. – USA, 2017. – P.240.

3. Kenneth C. Laudon, Jane. P. Laudon. Management Information Systems: Managing the Digital Firm, 13th Edition, Pearson Education. – USA 2014. – P.621.

4. Axel Rauschmayer. Speaking JavaScript. O'Reilly Media. USA, 2014. — P.419.

5. Alex Allain. Jumping into C++. – USA, 2014. – P.340.

6. Стенли Липпман. Язык программирование C++. Базовой курс. – Москва: Вильямс, 2014 год.

7. Сидхарма Рао. Освой самостоятельно C++ за 21 день. – Москва: Вильямс, 2013 год.

8. Nazirov Sh.A., Qobulov R.V., Bobojonov M.R., Raxmanov Q.S. C va C++ tili. – Toshkent: "Voris", 2013-yil. – 488-b.

9. Kjell Backman. Structured Programming with C++. BookBoon. – USA, 2012. – P.246.

10. Matt Doyle. Beginning PHP 5.3. Wiley Publishing, Inc. – USA, 2010. – P.775.

11. G'ulomov S.S., Begalov B.A. Informatika va axborot texnologiyalari. Darslik. – Toshkent: "Fan", 2010-yil. – 686-b.

12. Nazirov Sh.A., Qobulov R.V. Obyektga mo'ljallangan dasturlash. – Toshkent: "Aloqachi", 2007. – 337-b.

13. Aripov M.M., Yakubov A.X., Sagatov M.V., Irmuhamedova R.M. va boshq. Informatika. Axborot texnologiyalari. O'quv qo'llanma. 1,2-qism. – Toshkent: TDTU, 2005-yil.

14. Xaldjigitov A.A., Madraximov Sh.F., Adamboyev U.E. Informatika va programmalashtirish. O'quv qo'llanma. – O'zMU, 2005-yil. – 145-bet.

MUNDARIJA

KIRISH 3

I bob. TEXNIK TIZIMLARDA ZAMONAVIY DASTURLASH TEXNOLOGIYALARI

1.1. Dasturlash tillari va ularning kelib chiqishi 6
1.2. Zamonaviy dasturlash tillari 8
1.3. Dasturlash tillarining tasniflanishi 13
1.4. Zamonaviy dasturlash texnologiyalari 16

II bob. BORLAND C++ BUILDER 6 INTEGRALLASHGAN SOHASI, UNING TASHKIL ETUVCHILARI

2.1. Borland C++ Builder 6 dasturlash tizimi va uning asosiy xususiyatlari 22
2.2. Borland C++ Builder 6 dasturini o'rnatish jarayoni 24
2.3. Borland C++ Builder 6 dasturining integrallashgan sohasi .31
2.4. Borland C++ Builder 6 dasturining komponentalar palitrasi .. 36

III bob. C++ DASTURLASH TILINING ASOSIY KONSTRUKSIYALARI VA ULARDAN FOYDALANISH XUSUSIYATLARI

3.1. C++ tilining asosiy tashkil etuvchilari 47
3.2. Ma'lumot toifalari 51
3.3. Standart funksiyalar 53
3.4. Borland C++ Builder 6 da dastur tarkibi 54
3.5. Borland C++ Builder 6 da konsol ilovasini yaratish 58
3.6. Forma oynasida ilovalar yaratish 62

IV bob. BORLAND C++ BUILDER 6 DASTURLASH TIZIMIDA CHIZIQLI JARAYONLARNI DASTURLASH

4.1. Operator tushunchasi. C++ tilidagi operatorlar tasnifi 68
4.2. O'zlashtirish operatori 68
4.3. Ma'lumotlarni kiritish va chiqarish 70

4.4. Chiziqli jarayonlarni konsol muhitida dasturlash 75
4.5. Chiziqli jarayonlarni forma ilovasida dasturlash..... 79

V bob. C++ BUILDER 6 DASTURLASH TIZIMIDA TARMOQLANUVCHI JARAYONLARNI DASTURLASH

5.1. Shartsiz o'tish operatori 88
5.2. Shartli o'tish operatori 89
5.3. Tanlash operatori 101

VI bob. C++ BUILDER 6 DASTURLASH TIZIMIDA TAKRORLANUVCHI JARAYONLARNI DASTURLASH

6.1. Takrorlanish jarayonlarini tashkil qilish 115
6.2. Avval sharti tekshiriladigan takrorlanish jarayoni 116
6.3. Sharti keyin tekshiriladigan takrorlanish jarayoni..... 124
6.4. Parametrli takrorlanish jarayoni 131
6.5. Murakkab takrorlanish jarayonlari 139

VII bob. BORLAND C++ BUILDER 6 DASTURLASH TIZIMIDA MASSIVLAR BILAN ISHLASH

7.1. Massiv tushunchasi va uning turlari 142
7.2. C++ tilida massivlarni tavsiflash..... 143
7.3. Massiv elementlarini xotiraga kiritish..... 145
7.4. Forma ilovasida massivlar bilan ishlash usullari 148

VIII bob. C++ DASTURLASH TILINING STRUKTURA TOIFASI

8.1. C++ tilida struktura toifasi va uni tavsiflash..... 163
8.2. Struktura elementi va ular ustida bajariladigan amallar... 165
8.3. C++ dasturlash tilida aralash toifani qo'llash 167

IX bob. C++ DA FUNKSIYA VA PROTSEDURALAR. STRUKTURALI DASTURLASH USULLARINI C++ DA QO'LLASH

9.1. C++ tilida funksiyalar va ulardan foydalanish 185

9.2. Protseduralar va ularni tashkil etish 189
9.3. Funksiyalarda massivlarni ishlatish..... 194

X bob. C++ DASTURLASH TIZIMIDA MA'LUMOTLARNING FAYLLI VA MUROJAAT TOIFASI

10.1. Ma'lumotlarning faylli toifasi 202
10.2. Fayldan o'qish-yozish funksiyalari 204
10.3. Fayl obyektlarini kiritish-chiqarish..... 207
10.4. C++ da ko'rsatkichlar bilan ishlash 223
10.5. Matnli ma'lumotlarga murojaat etish funksiyalari..... 232

XI bob. BORLAND C++ BUILDER DASTURLASH TIZIMIDA SINFLAR VA OBYEKTLAR

11.1. Sinflar 239
11.2. Abstraksiya 242
11.3. Vorislik..... 244
11.4. Polimorfizm 246

XII bob. BORLAND C++ BUILDER 6 NING GRAFIK IMKONIYATLARI

12.1. Tayyor rasmlar bilan ishlash..... 251
12.2. Qalam va mo'yqalam 253
12.3. Oddiy figuralar chizish 259

XIII bob. BORLAND C++ BUILDER 6 INTEGRALLASHGAN SOHASIDA MA'LUMOTLAR BAZASINI QAYTA ISHLASH

13.1. Ma'lumotlar bazasini tashkil qilish va ularni boshqarish. 272
13.2. Borland C++ Builder 6 da ma'lumotlar bazasi va uni qayta ishlash..... 274
13.3. Texnik tizimlarda ma'lumotlar bazasini qayta ishlash ... 279
GLOSSARIY 290
FOYDALANILGAN ADABIYOTLAR RO'YXATI 296

MIR-XUSAN MIRPULATOVICH KADIROV

TEXNIK TIZIMLARDA AXBOROT TEXNOLOGIYALARI

Darslik

2-qism

Muharrir *M. Tursunova*
Musahhih *M. Turdiyeva*
Dizayner sahifalovchi *X. Abdullayev*

"O'zbekiston faylasuflari milliy jamiyati" nashriyoti,
100029, Toshkent shahri, Matbuotchilar ko'chasi, 32-uy.
Tel./faks: 239-88-61.

Nashriyot litsenziyasi: AI №216, 03.08.2012.
Bosishga ruxsat etildi 15.11.2019. "Uz-Times" garniturasi. Ofset usulida chop etildi. Qog'oz bichimi 60x84 $^{1}/_{16}$. Bosma tabog'i 19,5. Nashr hisob tabog'i 19,0. Adadi 200 nusxa. Buyurtma №.25

"FAYLASUFLAR" MChJ bosmaxonasida chop etildi.
Manzil: Toshkent shahri, Matbuotchilar ko'chasi, 32-uy.

ISBN 978-9943-6169-7-4

9 789943 616974

“O'zbekiston faylasuflari milliy jamiyati” nashriyoti