C++

ZAMONAVIY DASTURLASH TILLARI

KASB-HUNAR KOLLEJLARI UCHUN O‘QUV QO‘LLANMA

Mazkur o‘quv qo‘llanma O‘zbekiston-Shveysariya «Kasbiy ko‘nikmalarni rivojlantirish» loyihasi doirasida ishlab chiqilgan

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

O‘RTA MAXSUS, KASB-HUNAR TA’LIMI MARKAZI

H. RAHIMOV, T. DEHQONOV

ZAMONAVIY DASTURLASH TILLARI

Kasb-hunar kollejlari uchun o‘quv qo‘llanma

„O‘QITUVCHI“ NASHRIYOT-MATBAA IJODIY UYI
TOSHKENT — 2007

ББК 32.973.26-018.1я722

Mazkur o'quv qo'llanma O'zbekiston-Shveysariya "Kasbiy ko'nikmalarni rivojlantirish" loyihasi doirasida yaratilgan

Ilmiy-uslubiy maslahatchi: Bahodir Haydarov — O'zbekiston-Shveysariya „Kasbiy ko'nikmalarni rivojlantirish" loyihasining milliy maslahatchisi.

Taqrizchilar: Akram To'xtaboyev — „UzDEUavto" korxonasi texnik xizmat ko'rsatish bo'limi menejeri.

Hakimjon Zayniddinov — Toshkent axborot texnologiyalari universiteti informatsion texnologiyalar kafedrasi mudiri.

Maxsus muharrir: Odiljon Parpiyev — Andijon muhandislik-iqtisodiyot instituti umumtexnika fanlari kafedrasining mudiri.

O'quv qo'llanma kasb-hunar kollejlarining „Mashinasozlik texnologiyasi, mashinasozlik ishlab chiqarish jihozlari va ularni avtomatlashtirish" ta'lim yo'nalishining „Avtomatlashtirilgan tizimlarni ta'mirlash va sozlash" mutaxassisligi o'quv rejasidagi „Zamonaviy dasturlash tillari" fani bo'yicha ishlab chiqilgan namunaviy o'quv dasturi asosida yozilgan. Unda zamonaviy dasturlash tillaridan C++ tilida dasturlashning asoslari, umumiy tamoyillari va amaliy yo'l-yo'riqlari yoritilgan.

Mazkur o'quv qo'llanma kasb-hunar kollejlarining „Avtomatlashtirilgan tizimlarni ta'mirlash va sozlash" mutaxassisligi bo'yicha tahsil olayotgan talabalar uchun mo'ljallangan. Unda, shuningdek, boshqa kasb-hunar kollejlarining talabalari va zamonaviy dasturlash tililarini o'rganmoqchi bo'lgan barcha kitobxonlar ham foydalanishlari mumkin.

Shartli belgilar tavsifi

Yodda saqlash lozim bo'lgan ma'lumotlar

O'zlashtirilgan bilimlarni mustahkamlash uchun mashq

Mavzu bo'yicha qo'shimcha ma'lumotlar

O'zlashtirilgan bilimlarni nazorat qilish uchun savollar

R $\frac{4306020500-161}{353(04)-2007}$ Qat'iy buyurtma – 2007

ISBN 978-9943-02-130-3

SO'ZBOSHI

Zamonaviy kompyuterlarda turli dasturlash tillari keng qo'llanilmoqda. Bu dasturlar iqtisodiyot, boshqarish, xizmat ko'rsatish va ishlab chiqarishning turli sohalarida muhim ahamiyatga molik masalalarni hal qilishda ba'zan yagona omil bo'lib qolmoqda. Zamonaviy ishlab chiqarish korxonalarida ishlab chiqarish jarayonini takomillashtirish va avtomatlashtirishda dasturlashning ahamiyati katta. Sanoatning turli sohalarida ishlatilayotgan avtomatlashtirilgan jihozlar, robot texnikalari ham turli algoritmik tillarda yozilgan ma'lum dasturlar asosida kompyuterlar yordamida boshqariladi. Bu esa, o'z navbatida, ishlab chiqarish korxonalarida xizmat qilayotgan faqat boshqaruv xodimlari va muhandislari emas, balki ishchilar ham dasturlash tillarini puxta egallashlarini taqozo etadi.

Maskur o'quv qo'llanmani yaratishdan maqsad, „Avtomatlashtirilgan tizimlarni ta'mirlash va sozlash" mutaxassisligini egallayotgan kasb-hunar kollejlari talabalariga zamonaviy dasturlash tillari haqida umumiy tushuncha berish va C++ dasturlash tilidan foydalanish bo'yicha dastlabki ko'nikmalarni shakllantirishdan iborat.

O'quv qo'llanma 15 mavzudan iborat bo'lib, u dasturlash tillarining umumiy tavsifidan boshlanadi. Shuningdek, o'quv qo'llanmaning boshlanishida algoritmlar nazariyasi va uning tavsifi ham keltirilgan. Keyingi mavzular C++ dasturlash tilining tavsifiga bag'ishlangan. Har bir mavzu bo'yicha ko'plab mashq va nazorat savollari berilgan. O'quv qo'llanmaga o'zgaruvchi va o'zgarmas tipli kattaliklar, standart matematik funksiyalar, C++ dasturlash tilida qo'llaniladigan asosiy dasturlash operatorlari, massivlar, funksiyalar kabi tushunchalar haqidagi mavzular kiritilgan bo'lib, ular yordamida C++ dasturlash tili imkoniyatlari ochib berilgan.

Shuningdek, undan strukturalar, vizual dasturlar tuzish va C++ Builder kompilatori komponentlari haqidagi mavzular ham o'rin olgan.

O'quv rejasi bo'yicha, „Zamonaviy dasturlash tillari" fanidan 24 soat nazariy va 36 soat amaliy mashg'ulotlar o'tkazish nazarda tutilgan. Ushbu o'quv qo'llanma mazkur fan doirasida nazariy va amaliy mashg'ulotlar o'tishga mo'ljallangan o'quv materiallarini qamrab oladi.

Mualliflar

1-mavzu DASTURLASHNING AHAMIYATI

Dasturlash tillari qanday kelib chiqqan? Ularning vazifasi va mohiyati nimadan iborat? Qanday dasturlash tillari bor? Bu mavzuda shu va shunga o'xshash savollarga javob olasiz. Shuningdek, dasturlash tillarini o'zaro taqqoslash natijasida ularning o'ziga xos xususiyatlari, afzalliklari va kamchiliklari bilan tanishasiz.

1.1. Dasturlash tillari haqida tushuncha

EHM berilgan algoritmlarni formal bajaruvchi avtomat hisoblanadi. Shuning uchun biror masalani EHMda yechish uchun unga mos algoritmni berish zarur.

Algoritmni EHMga uzatishda esa uni maxsus „mashina tili"ga o'girib, mashina kodida yozilgan dasturga aylantiriladi. Shu bilan bir qatorda, turli xil EHMlar uchun turli tillar yaratilgan bo'lib, biror EHM uchun yozilgan dastur boshqa EHM uchun tushunarsiz bo'lishi mumkin. Shunday qilib, har bir EHM faqat o'zining „mashina tili"da yozilgan dasturlarnigina tushunishi va bajarishi mumkin.

Mashina kodida yozilgan dasturlarning ko'rinishi juda sodda, chunki bu dasturlar faqat 0 va 1 raqamlarining maxsus ketma-ketligidan tashkil topadi. Bu yozuv mutaxassis bo'lmagan odamga tushunarsiz bo'lib, dastur tuzishda noqulayliklar keltirib chiqaradi. Demak, mashina tilidan foydalanish odam uchun uni qiziqtirgan, ya'ni yechishi kerak bo'lgan masalaning algoritmini ishlab chiqishda va yozishda juda katta qiyinchiliklar va muammolar tug'diradi. Yuqorida aytib o'tilgan qiyinchiliklarni bartaraf qilish, dasturchining ishini osonlashtirish va yaratilgan dasturlarning ishonchlilik darajasini oshirish maqsadida, yuqori darajadagi dasturlash tillari yoki algoritmik tillar yaratilgan.

Algoritmik tillarning mashina tillaridan asosiy farqi quyidagilardan iborat:

- algoritmik til alifbosi mashina tili alifbosidan ancha kengligi;
- tuzilgan dastur matni juda sodda va qulay ko'rinishda ekanligi;
- ishlatilishi mumkin bo'lgan amallar majmuyi mashina amallari majmuyiga bog'liq emasligi;
- bajariladigan amallar qulay ko'rinishda, ya'ni amalda qabul qilingan matematik belgilashlar yordamida berilishi;

• amallar operandlariga dasturchi tomonidan shaxsiy ismlar qo'yish mumkinligi;

• mashina uchun ko'zda tutilgan ma'lumot tiplaridan tashqari yana yangi tiplarni kiritish imkoniyatining yaratilganligi.

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

Algoritmik tilda yozilgan masala yechimining algoritmi to'g'ridan to'g'ri EHMda bajarilishi mumkin emas ekan. Buning uchun, algoritm oldindan ishlatilayotgan EHMning mashina tiliga translator (kompilator yoki interpretator) yordamida o'girilishi kerak.

Translator — mashina tilida yozilgan maxsus dastur bo'lib, uning asosiy vazifasi algoritmik tilda yozilgan dastur matnini EHM tiliga tarjima qilishdan iboratdir.

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

Misol

1. $A = n$, $B - m$ deylik.
2. Agar $A = B$ bo'lsa, 5-bandga, aks holda 3-bandga o't.
3. Agar $A > B$ bo'lsa, A ning yangi qiymati deb $A—B$ ni qabul qil, B ning qiymatini o'zgartirma.

Aks holda B ning yangi qiymati deb $B—A$ ni qabul qil, A ning qiymatini o'zgartirma.

4. 2-bandga o't.
5. EKUB = A va hisobni to'xtat.

Bu algoritmni qisqaroq ko'rinishda quyidagicha ifodalashimiz ham mumkin:

Misol

1. $A = n$, $B - m$ deylik.
2. Agar $A > B$ bo'lsa, $A = A—B$, aks holda $B = B—A$, $A = B$ bo'lguncha 2-bandni takrorla.
3. EKUB = A va hisobni to'xtat.

Bu misoldan ko'rinib turibdiki, algoritmlarni bunday yozish uslubi ham qulay, ham tushunarlidir. Lekin bu uslubda ham quyidagi kamchiliklar ko'zga tashlanadi:

- algoritm uzun va unda ortiqcha so'zlar ko'p ishlatilgan;
- bir xil ma'nodagi ko'rsatmani turli xil uslublarda berish mumkin;
- erkin ko'rinishda ifodalangan algoritmni EHM tiliga o'tkazish imkoniyati kam.

Yuqoridagi kabi kamchiliklarni bartaraf qilish uchun formallashgan, qat'iy aniqlangan algoritmik tillar ishlab chiqilgan.

Endi algoritmik tiliarning qaysilari amalda ko'proq ishlatilishi haqida fikr yuritsak. Ma'lumki, XX asrning 70-yillarida bir guruh muammoli-yo'naltirilgan algoritmik tillar yaratilgan bo'lib, bu dasturlash tillaridan foydalanib juda ko'p sohalardagi muammoli vazifalar hal qilingan.

Hisoblash jarayonlarining algoritmlarini ifodalash uchun „Algol-60" va „Fortran" tillari, iqtisodiy masalalar algoritmlari uchun „Kobol" va „Algek" tillari, matnli axborotlarni tahrir qilish uchun esa „Snobol" tillari ishlatilgan.

Sanab o'tilgan bu algoritmik tillar „asosan" katta hajmni egallaydigan, ko'pchilikning foydalanishiga mo'ljallangan EHM lar uchun yaratilgan edi.

Hozirda insoniyat faoliyatining barcha jabhalariga shaxsiy elektron hisoblash mashinalari (SHEHM) shaxdam qadamlar bilan kirib bormoqda.

SHEHMlarga mo'ljallangan hamda murakkab jarayonlarning hisoblash ishlarini bajarish va juda katta ma'lumotlar tizimi bilan ishlashni tashkil etuvchi yangi algoritmik tillar sinfi borgan sari kengayib bormoqda. Bular jumlasiga quyidagi tillarni kiritish mumkin:

„Beysik" tili; „Paskal" tili; „Si" tili va hokazo.

1.2. „Beysik"(„Basic") dasturlash tili

Dastur tuzishni o'rganishni boshlovchilarga mo'ljallangan, savol-javob tizimida ishlaydigan, turli-tuman jarayonlar algoritmini yozishga qulay bo'lgan tillardan biri „Beysik"(„Basic") tilidir.

„Basic“ (o‘qilishi: „Beysik“) so‘zi ingliz tilida „Beginner’s All-purpose Symbolic Instruction Code“, ya’nl „Boshlovchilar uchun belgili ko‘rsatmalar kodi (tili)“ degan ma’noni anglatadi.

Beysik tilini yaratish ustidagi ishlar 1963-yili boshlangan. Tilning ijodkorlari taniqli olimlar T.Kurs va J.Kemeni hisoblanadi. Hozirga kelib, „Beysik“ tilining turli xil yangi ko‘rinishlari ishlab chiqilgan va ulardan samarali foydalanilmoqda.

1.3. „Si“ va C++ dasturlash tillari

Hozirda amalda foydalanilayotgan ko‘plab operatsion tizimlar „Si“ tilida yaratilgan. Shunday bo‘lsa-da, eng universal til sifatida tan olingan til bu, C++ (o‘qilishi, „Si plus-plus“) dasturlash tilidir. Mazkur kitob shu til haqida bo‘lib, bu tilning eng asosiy boshlang‘ich tushunchalari va eng oddiy imkoniyatlari bilan kelgusi mavzularda tanishishni boshlaymiz.

2-mavzu	**MASALALARNI EHMDA YECHISH BOSQICHLARI**

2.1. Masalalarni EHMda yechish bosqichlari

Kompyuterda biror muammoni hal qilish bir necha bosqichlarga bo‘linadi. Eng avvalo, tahlil qilinayotgan jarayon yoki inshootning zarur jihatlarini o‘zida mumkin qadar to‘la akslantirgan matematik in’ikosi (modeli) tuzib olinadi. Matematik model formula va tenglamalar tizimi ko‘rinishida ifodalanadi. Hosil bo‘lgan matematik masalani yechish uchun eng maqbul hisoblash algoritmi tuziladi. Aniqlangan algoritmga xos hisoblash usuli tanlab olinadi va bu usulni kompyuter va foydalanuvchi tushunadigan tilda xotiraga joylashtiriladi. Biror algoritmik tilda tuzilgan dastur bo‘yicha olingan natijalar tahlil qilinadi va ular asosida o‘rganilayotgan jarayon uchun xos bo‘lgan umumiy qonuniyatlar aniqlanadi. Algoritmlash jarayonida qo‘yilgan masalalarni EHMda qayta ishlashga tayyorlash — asosiy vazifalardan biri hisoblanadi.

EHM bilan bevosita ishlashdan oldin qanday bosqichlarni bajarish lozimligini ko‘rib chiqamiz.

1-bosqich. Masalani qo‘yish. Istalgan masalani yechish uning qo‘yilishidan boshlanadi. Masala shartining aniq ifodasi masalaning matematik qo‘yilishi deb ham ataladi. Masalaning qo‘yilishida

boshlang'ich ma'lumotlar yoki argumentlar, qiymatlari aniqlanishi kerak bo'lgan kattaliklar, ya'ni natijalar ajratiladi. Masalani qo'yish — uni yechishning birinchi bosqichi bo'ladi.

2-bosqich. Masalaning matematik modelini qurish. Amaliy masalalarni hal etishda obyektlar — tabiat hodisalari (fizik yoki kimyoviy jarayonlar), mahsulot ishlab chiqarish jarayonlari, rejalari va shu kabilar bilan ish ko'rishga to'g'ri keladi. Ana shunday masalalarni qo'yish uchun avval tekshirilayotgan obyektni matematik atamalarda tavsiflash, ya'ni iloji bo'lsa, uning matematik modelini (in'ikosi) qurish kerak bo'ladi.

3-bosqich. Masalani yechish usulini aniqlash. Masalaning matematik modeli yaratilgandan so'ng, uni yechish usuli izlana boshlanadi. Ayrim hollarda masalani qo'yilishidan keyin to'g'ridan to'g'ri masalani yechish usuliga ham o'tish mumkin. Chunki, bunday masalalar oshkor ko'rinishdagi matematik model bilan ifodalanmasligi mumkin.

4-bosqich. Masalani yechish algoritmini tuzish. Navbatdagi bosqichda masalani yechish algoritmi tuziladi. Algoritmni turlituman ko'rinishda yozish mumkin. Dasturlash fanining asosiy vazifalaridan biri algoritm tuzish usullarini o'rganishdan iborat.

5-hosqich. Masalani yechish algoritmini dasturlash tiliga ko'chirish. Algoritmning EHMda bajarilishi uchun bu algoritm dasturlash tilida yozilgan bo'lishi kerak. Masalani yechishning bu bosqichida biror bir usulda tuzilgan algoritm dasturlash tiliga ko'chiriladi. Masalan, agar algoritm blok-sxema ko'rinishida tasvirlangan bo'lsa, uni dasturlash tiliga ko'chirish uchun har bir blokni tilning mos buyruqlari bilan almashtirish yetarli.

6-bosqich. EHMda tuzilgan dasturni amalga oshirish. Bu bosqichda dastur ko'rinishida yozilgan algoritm EHMda bajariladi. Bu bosqich dastur tuzuvchilar uchun eng qiyin hisoblanadi. Chunki dasturni mashina xotirasiga kiritishda ayrim xatoliklarga yo'l qo'yish mumkin. Shuning uchun dasturni EHM xotirasiga kiritishda juda ehtiyot bo'lish kerak. Bu bosqich natija olish bilan tugallanadi.

7-bosqich. EHMda olingan natijalarni tahlil qilish. Masalani yechishning yakunlovchi bosqichi olingan natijalarni tahlil qilishdir. Bu bosqich olingan natijalar qanchalik haqiqatga yaqinligini aniqlash maqsadida bajariladi. Natijalarni tahlil qilish zarur bo'lgan hollarda algoritmni, yechish usuli va modelini tanlashga yordam beradi.

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

3-mavzu	**ALGORITMLAR VA ULARNI TAVSIFLASH**

„Algoritm“ (**algorifm**) so‘zi o‘rta asrlarda paydo bo‘lgan bo‘lib, u vatandoshimiz buyuk mutafakkir Al-Xorazmiyning ishlari bilan yevropaliklarning birinchi bor tanishishi natijasida yuzaga kelgan. Bu ishlar ularda juda chuqur taassurot qoldirib, algoritm **(algorithmi)** so‘zining kelib chiqishiga sabab bo‘ldiki, u Al-Xorazmiy ismining lotincha aytilishidir. U paytlarda bu so‘z arablarda qo‘llaniladigan o‘nlik sanoq tizimi (sistemasi) va bu sanoq tizimida hisoblash usulini bildirar edi. Shuni ta’kidlash kerakki, yevropaliklar tomonidan arab sanoq tizimining Al-Xorazmiy ishlari orqali o‘zlashtirilishi, keyinchalik, hisoblash usullarining rivojlanishiga katta turtki bo‘lgan.

Al-Xorazmiy (783—855)

3.1. Algoritmlar nazariyasiga kirish

Algoritm deb, berilgan masaiani yechish uchun ma’lum tartib bilan bajarilishi kerak bo‘lgan chekli sondagi buyruqlar ketma-ketligiga aytiladi.

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

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

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

Yuqorida keltirilgan xossalarni har bir o‘quvchi o‘zi tuzgan biror masalaning algoritmidan foydalanib tekshirib ko‘rishi mumkin.

Misol

***ax*²+ *bx* + *c* = 0 kvadrat tenglamani yechish algoritmi uchun yuqorida sanab o‘tilgan algoritmning xossalarini quyidagicha tekshirib ko‘rish mumkin.**

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

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

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

Dastur tuzuvchi uchun EHMning ikkita asosiy parametri o‘ta muhimdir: hisoblash mashinasi xotirasining hajmi va mashinaning tezkorligi.

Shuningdek, algoritm tuzuvchidan ikki narsa talab qilinadi. Birinchisi, u tuzgan dastur mashina xotirasidan eng kam joy talab etsin, ikkinchisi esa, eng kam amallar bajarib masalaning natijasiga erishilsin. Umuman olganda, bu ikki talab bir-biriga qarama-qarshidir, ya’ni algoritmning ishlash tezligini oshirish algoritm uchun kerakli xotirani oshirishga olib kelishi mumkin. Bu hoi, ayniqsa, murakkab masalalarni yechish algoritmini tuzishda yaqqol seziladi. Shuning uchun ham bu ikki parametrning eng maqbul holatini topishga harakat qilish kerak.

3.2. Algoritmni tavsiflash usullari

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

Misol

Algoritmning so‘zlar yordamida berilishiga, tavsiflanishiga misol tariqasida liftda kerakli qavatga ko‘tarilish algoritmini keltirish mumkin.

Bu quyidagi ketma-ketlikda bajariladi:

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

Algoritm matematik formulalar yordamida tavsiflanganda har bir qadam aniq formulalar bilan ifodalanadi.

Misol

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

kvadrat tenglama ildizlari bo‘lgan x_1 va x_2 ni aniqlash algoritmini ko‘rib chiqaylik.

1. a, b, c koeffitsiyentlar qiymatlari berilsin.
2. $D = b^2 - 4ac$ diskriminant hisoblansin.
3. $D < 0$ bo‘lsa, tenglamaning haqiqiy yechimlari yo‘q.
Faqat haqiqiy ildizlar izlanayotgan bo‘lsa, masala hal bo‘ldi.
4. $D = 0$ bo‘lsa, tenglama ikkita bir-biriga teng, ya’ni karrali ildizlarga ega bo‘ladi va ular kvadrat tenglama ildizlarini topish formulalari bilan hisoblanadi. Masala hal bo‘ldi.
5. $D > 0$ bo‘lsa, tenglama ikkita haqiqiy yechimga ega, ular formulalar bilan hisoblanadi. Masala hal bo‘ldi.

Shunday qilib, kvadrat tenglamaning haqiqiy yechimlarini aniqlashda:

- „Tenglamaning haqiqiy yechimlari yo‘q“;
- „Tenglama karrali yechimga ega“;

• „Tenglama ikkita haqiqiy ildizga ega“

degan matnlarni ham javob bilan birga yozish mumkin.

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

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

Algoritmlarni geometrik tarhlar yordamida tavsiflash ko'rgazmali va shu sababli tushunarliroq bo'lgani uchun ko'p qo'llaniladi. Bunda har bir o'ziga xos operatsiya alohida geometrik shakl (blok) bilan tavsiflanadi va ularning bajarilish tartibi, ular orasidagi ma'lumotlar uzatilishi va yo'nalishi bloklarni bir-biri bilan ko'rsatkichli to'g'ri chiziqlar (strelkalar) yordamida tutashtirib ko'rsatiladi.

Algoritmning geometrik tarhi uning blok-sxemasi deyiladi.

Bloklarga mos geometrik shakllar, ularning o'lchamlari va ular yordamida blok-sxemalarni chizish qoidalari davlat standartlarida berilgan. Bloklarni bajarish tabiiy yozish tartibida bo'lsa, ya'ni yuqoridan pastga yoki chapdan o'ngga bo'lsa, tutashtiruvchi chiziq ko'rsatkichsiz (strelkasiz) bo'lishi mumkin. Boshqa barcha hollarda ma'lumot oqimi yo'nalishini ko'rsatuvchi ko'rsatkich qo'yilishi shart.

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

MASHQLAR

1. Quyidagi holatning algoritmini tuzing: „Agar Botir uyda bo'lsa, matematika fanidan vazifani bajaramiz. Aks holda, Sayyoraga qo'ng'iroq qilib devoriy gazeta chiqarishga

kirishamiz. Agar Sayyora uvda yo‘q bo‘lsa, insho yozish kerak bo‘ladi“.

2. Quyidagilarning algoritmini tuzing:

a) $ax = 7$ tenglamani yechish;

b) $ax > 5$ tengsizlikni yechish.

3. Quyidagi kvadrat tenglamaning haqiqiy ildizlarini topish algoritmini tuzing:

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

Nazorat savollari

1. Algoritmga ta’rif bering.
2. Algoritmlarning asosiy xususiyatlarini sanang.
3. Algoritmlarni qanday usullar bilan tavsiflash mumkin?
4. Algoritmlarning nechta asosiy shakllari bor?

4-mavzu	**C++ DASTURLASH TILI VA UNING TUZILMASI**

4.1. C++ dasturlash tiliga kirish

Dasturlash tilini o‘rganishni boshlashning eng yaxshi usuli bu biror bir soddaroq dasturni olib, uni tahlil qilishdir. Quyida ana shunday dasturlardan biri berilgan bo‘lib, C++ dasturlash tilini o‘rganishni shu misolni tahlil qilishdan boshlaymiz.

```
// C++ da ilk dasturim
#include <iostream.h>
int main ()
{
  cout<<"Assalomu-alaykum!";
  return 0;
}
```

```
Assalomu alaykum!!!
```

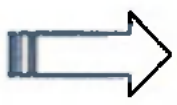
Yuqorida o‘ng tomonda birinchi dasturimizning kodi ko‘rsatilgan va uni biz ixtiyoriy ravishda nomlashimiz mumkin. Masalan,

salom.cpp. Chap tomonda esa kompilatsiya qilinganidan keyingi holat, ya'ni dasturning natijasi ko'rsatilgan.

Dasturni tahrir va kompilatsiya qilish yo'li siz foydalanayotgan kompilatorga, uning rivojlantirilgan interfeysi yoki versiyasiga bog'liqdir.

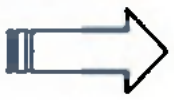
Yuqoridagi dastur C++ da yozilgan eng sodda dasturlardan biridir. Ammo u C++ dagi boshlang'ich komponentlarni o'z ichiga olgan. Biz dasturning har bir komponentlarini birma-bir ko'rib chiqamiz va izohlaymiz:

// C++ da ilk dasturim



Bu izoh satridir. // belgisi bilan boshlanadigan barcha satrlar **izoh qatorlari** deb tushuniladi. Bu satrlar dasturning ishlashiga hech qanday ta'sir qilmaydi. U faqat dasturchi uchun dastur kodlariga qisqa izoh berish uchun xizmat qiladi.

 # include <iostream.h>



belgisi bilan boshlanadigan satr preprotsessor uchun buyruq (direktiva) hisoblanadi. Ular dasturda bajaradigan kod satrlari emas, balki kompilyator uchun ko'rsatmadir. Demak, **# include <iostream.h>** satri kompilator preprotsessoriga **"iostream standart header"** faylini kiritib qo'yishga buyruq beryapti. Bu maxsus fayl C++ dagi standart kiritish-chiqarish kutubxonasini e'lon qiladi. Bu fayl dasturga undan keyinchalik foydalanilsh mumkin bo'lganligi uchun kiritiladi.

int main()

Bu qator **main** funksiyasi e'loni boshlanayotganini bildiradi. C++ da dastur ishlashni boshlaydigan joy bu **main** funksiyasidir. Dasturning asosiy qismi **main** funksiyasi ichiga yoziladi. Bu funksiya mustaqil bo'lib, u dasturning boshida, o'rtasida yoki oxirida bo'lishidan qat'i nazar, uning tarkibi dastur

ishlaganda birinchi bo'lib ishga tushadi. Shuning uchun ham C++ dagi dasturlarning barchasida **main** qismi mavjud bo'ladi.

main funksiya bo'lgani uchun undan keyin ( ) qavslari bor. C++ da barcha funksiyalardan keyin kichik qavs ( ) lar qo'yiladi, chunki ularning ichiga funksiya argumentlari yoziladi. **main** funksiyasi tarkibida har doim uning e'loni yozilgan va u { } qavslari bilan berkitilgan bo'ladi.

Bu satr dasturda juda muhim ahamiyatga ega. **cout** C++ da standart chiqarish oqimi hisoblanib, bu oqim ichiga qanday yozuv yozilsa, shu narsa ekranga chigadi. **cout "iostream.h header"** fayli orqali e'lon qilinadi. Demak, **cout** dan foydalanish uchun ushbu faylni kiritib qo'yish zarur. Satr nuqta-vergul (;) bilan tugayotganiga e'tibor bering. Bu belgi satr yakunini bildiradi va C++ da har bir qator oxiriga qo'yilishi shart.

cout <<"Assalomu alaykum!"; ⇨

return 0; ⇨

return main() funksiyasi ishini yakunlaydi va **0** holatga dasturni qaytaradi. Bu ishlash davomida xato topilmagan dasturni yakunlashning eng odatiy holatidir.

E'tibor bergan bo'lsangiz, dasturdagi hamma satrlar ham ishlamadi. Dastur tuzish davomida faqat izoh beradigan satrlar, kompilator preprotsessoriga ko'rsatma beradigan satrlar keldi. Keyin esa funksiya e'lonini bildiruvchi satr keldi. So'ngra **main** funksiyasining {} qavslari ichiga yoziladigan so'ngi satrlar yozildi.

Dasturni o'qishga oson bo'lishi uchun uni turli satrlarga ajratilib yozildi, ammo bunday qilib yozish shart emas.

Masalan,

```
int main ()
        {
    cout << "Salom,dunyo!";
        return0;
}
```

o'rniga quyidagicha ham yozish mumkin:

```
int main(){cout<<"Salom, dunyo!";return0;}
```

C++ da barcha satrlar bir-biridan yakunlovchi nuqta-vergul (;) bilan ajratilib yoziladi.

Dastur satrlarini ajratib yozilishi uni o'qishni osonlashtiradi va dasturni qulay va sxematik tasvirlashga yordam beradi.

Misol

```
// C++ da ikkinchi dasturim
#include <iostream.h>
int main ()
{
 cout<<"Salom!";
 cout<<"Men C++dasturiman";
 return 0;
}
```

⇒ **Men C++ dasturiman**

Bu holatda **cout <<** ni biz ikkita satr uchun alohida-alohida ishlatdik. Bu dasturni o'qishda qulaylik yaratadi.

Preprotsessor direktivalari bu qoidadan mustasnodir. Ular preprotsessor tomonidan o'qib bo'lingan va tark etilgan hamda ular hech qanday kod ishlab chiqmaydilar. Ular o'zlarining qatorida aniqlandi va bu holatda satrga „:“ belgisi qo'yilmaydi.

4.2. Izohlar satrini tavsiflash

Izohlar kod manbasining bir qismi bo'lib, ularning vazifasi dasturchiga dastur kodini tushuntirish uchun qisqa yozuv va izohlar berishga imkon beradi.

C++ da izoh qo'yishning ikki xil yo'li mavjud:

Birinchi usulda satr boshiga // belgisi qo'yiladi va satrning boshidan oxirigacha bo'lgan barcha yozuvlarni bekor qiladi. Ikkinchisi blok izohi bo'lib, /* */ lar ichida bo'lgan ma'lumotni bekor qiladi. Bu ma'lumotlar bir necha satrlardan iborat bo'lishi ham mumkin.

```
// satr izohi
/* blok izohi */
```

Agar siz dasturingizga izohni //, /* yoki */ kombinatsiyalarisiz qo'ysangiz, kompilator bu izohni dasturning bir qismi deb oladi va bu xatolik kelib chiqishiga olib keladi.

Biz ikkinchi dasturimizga izoh qo'yib chiqamiz.

Misol

```
Men C++ dasturiman
```

```
/* C++ dagi bir necha izohli
ikkinchi dasturim */
#include <iostream.h>
int main ()
{
  cout<<"Salom!";
  cout<<"Men C++dasturiman";
  return0;
}
```

MASHQLAR

1. Kompilatsiya qilingandan keyin dasturning natijasi quyidagi gaplar bo'ladigan dasturlar tuzing:
   a) Salom, O'zbekiston!
   b) Men C++ tilida dastur tuzayapman!
   d) Olg'a, „Paxtakor"!
2. // C++tilida yozilgan dastur yozuvi numani anglatadi?
3. #include <iostream.h> yozuvini qanday tushunasiz?

Nazorat savollari

1. „//" belgisi bilan boshlanadigan satr qanday vazifani bajaradi?
2. „#" belgisi bilan boshlanadigan satr qanday vazifani bajaradi?
3. **main** funksiyasi haqida nima bilasiz?
4. **cout** ning vazifasi nimadan iborat?
5. Izohlar satri qanday usullar bilan tavsiflanadi?

5-mavzu	**KONSOL ORQALI MULOQOT QILISH**

Konsol bu kompyuterning asosiy interfeysidir, ya'ni u kompyuterning klaviaturasi va ekrani orasidagi aloqa natijasidir. Klaviatura kompyuterning kiritish qurilmasi, ekran esa uning chiqarish qurilmasidir.

iostream C++ standart kutubxonasida dastur uchun kiritish va chiqarish operatsiyalarini ikki operator: kiritish uchun ***cin*** va chiqarish uchun ***cout*** amalga oshiradi. Shu bilan bir qatorda xatoni ko'rsatib beruvchi xabarni foydalanuvchiga ko'rsatadigan ***cerr*** va ***clog*** operatorlaridan ham foydalaniladi. Ular faylni chiqarish yoki ro'yxatga olish uchun mo'ljallangan.

Shu sababli, ***cout*** (standart chiqarish tizimi) to'gridan to'g'ri ekranga chiqarish uchun xizmat qilsa, ***cin*** (standart kiritish tizimi) esa klaviatura bilan ma'lumot kiritish imkonini beradi. Bu ikki operatordan foydalanib, siz o'z dasturingiz orqali dasturingiz foydalanuvchisi bilan muloqotda bo'lishingiz mumkin. Siz ekranga muloqot xabarini chiqarib, foydalanuvchiga klaviaturadan ma'lumot kiritishi uchun ko'rsatma berish imkoniga ega bo'lasiz.

5.1. Chiqarish operatori

cout operatori qo'shimcha yuklanuvchi << operatori bilan birgalikda foydalaniladi.

Misollar va izohlar


```
cout << x;      // ekranga x o'zgaruvchi tarkibidagi ma'lumot
                chiqadi
```

<< operatori kirish operatori sifatida qabul qilingan bo'lib, shunga ko'ra ***cout*** operatori ekranga chiqaradigan ma'lumotdan avval yoziladi. Yuqoridagi misolda qo'shtirnoq (" ") ichidagi matn qanday

ko‘rinishda bo‘lsa, konsolga shunday holatda chiqadi. Ikkinchi qatorda raqamli konstanta — 120 qo‘shtirnoqsiz ham ekranga chiqaveradi. Uchinchi qatordagi x o‘zgaruvchining qiymati konsolga chiqadi. Shunga e’tibor berish kerakki, birinchi qatorda harf(belgi)lar satrini to‘liq konsolga chiqarish uchun („ “)dan foydalanilyapti. Shunday ekan, har doim belgilar satrini ekranga chiqarmoqchi bo‘lsak, qo‘shtirnoqdan foydalanishimiz lozim.

Misollar va izohlar

```
cout <<"Salom";
```

Ekranda

Salom

```
cout << Salom;      // ekranga Salom o‘zgaruvchining qiymati chiqadi
```

<< operatori dastur tuzilayotganda, bir qatorda bir necha marta ishlatilishi mumkin. << operatorini bir necha marta ishlatishning ijobiy tomoni — o‘zgaruvchilar kombinatsiyasini yoki bir nechta o‘zgaruvchilar qiymatlarini birato‘la chiqarilayotganda ko‘rinadi.

```
cout << "Salom," << "men"<<
"C++ dasturiman";
```

Ekranda

Salom, men C++ dasturiman

Aytaylik, yosh o‘zgaruvchisining qiymati 24 va indeks o‘zgaruvchisining qiymati 710020 bo‘lsin.

```
cout << "Salom, mening yoshim " << yosh << " da va mening
indeks raqamim " << indeks;
```

Dastur natijasida ekranda quyidagi so‘zlar chiqadi:

Salom, mening yoshim 24 da va mening indeks raqamim 710020

Shuni ta’kidlash kerakki, dasturda satrning tugashi ko‘rsatilmasa, ***cout*** operatorining o‘zi satrni to‘xtatmaydi. Quyidagi misolda buni ko‘rish mumkin. Ikki ibora alohida ikki satrga yozilgan bo‘lsa ham natija ekranda bitta satrda chiqadi:

```
cout << "Bu dastur satri.";
cout << "Bu dasturning boshqa satri.";
```

Ekranda

```
Bu dastur satri. Bu dasturning boshqa satri.
```

Yangi satrga otkazish operatori. C++ tilida dastur satrida uzilish hosil qilib, yangi qatorga o'tish uchun ***\n*** ko'rinishda yoziladigan yangi satrga otkazish operatoridan foydalaniladi.

```
cout << "Birinchi ibora.\n ";
cout << "Ikkinchi ibora.\n
Uchinchi  ibora.";
```

Ekranda

```
Birinchi ibora.
Ikkinchi ibora.
Uchinchi ibora.
```

Bundan tashqari, yangi satrga o'tish uchun yana **endl** manipulyatoridan foydalanish mumkin.

```
cout << "Birinchi ibora." << endl;
cout << "Ikkinchi ibora." << endl;
```

Ekranda

```
Birinchi ibora.
Ikkinchi ibora.
```

5.2. Kiritish operatori

C++ da standart kiritish operatsiyasi ***cin*** operatori va qo'shimcha yuklanuvchi >> operatori bilan birgalikda amalga oshiriladi. U o'qilayotgan qiymat joylashgan o'zgaruvchisidan avval keladi.

```
int yosh;
cin >> yosh;
```

Bu misolda ***yosh*** o'zgaruvchi ***int*** tipi bilan e'lon qilinyapti va dastur qiymatni o'zgaruvchiga joylashtirish uchun uni klaviaturadan kiritilishini kutyapti.

Qaytish ***(Return)*** tugmasi bosilgan bo'lsa, ***cin*** ma'lumotlarni faqat klaviaturadan kiritish imkonini beradi. Shuning uchun, agar sizga yagona belgi - harfni ***cin*** orqali kiritmoqchi bo'lsangiz, unda qaytish tugmasi bosilmasa, bu jarayon amalga oshmaydi.

Siz har doim ***cin*** orqali kiritayotgan qiymat joylashadigan o'zgaruvchi tipini ko'rib chiqishingiz kerak bo'ladi. Agar sizga butun son kerak bo'lsa, siz ***integer*** tipli qiymatini olasiz, agar biron-bir belgi yoki harf kerak bo'lsa, ***char*** tipli qiymatini olasiz.

```
// i/o example
#include <iostream.h>
int main ()
{
  int i;
  cout << "Integer qiymat kiriting:
";
  cin >> i;
cout << "Kiritilgan qiymat " << i;
  cout << " va uning kvadrati " <<
i*2 << ".\n";
  return 0;
}
```

Ekranda

```
Integer qiymat kiriting:
702  Kiritilgan qiymat 702 va
uning kvadrati  51802
```

cin bilan ham bir satrning o‘zida bir necha marta qiymat kiritish mumkin:

cin >> ***a*** >> ***b*** ;

Bu yozuv quyidagiga teng kuchli bo‘ladi:

cin >> ***a***;

cin >> ***b***;

Ikkala holatda ham foydalanuvchi ikkita qiymat: ***a*** o‘zgaruvchi uchun alohida, ***b*** o‘zgaruvchi uchun esa alohida qiymat beradi. Albatta, bu ikki qiymatni bo‘sh joy bilan, bo‘sh qator bilan yoki **tab** bilan ajratib qo‘yish mumkin.

MASHQLAR

1. Konsolda „Salom, men talabaman“ yozuvini chiqaring.
2. Ekranga „Salom, men talabaman“ yozuvining har bir so‘zini alohida qatorga chiqaring.
3. Konsolda 10 ta ixtiyoriy sonlarni kiriting va ularning yig‘indisini ekranga chiqaring.

Nazorat savollari

1. Konsol orqali muloqot qilish deganda nimani tushunasiz?
2. Chiqarish operatorining vazifasi nima?
3. Chiqarish operatoridan qanday foydalaniladi?
4. Kiritish operatorining vazifasi nima?

6-mavzu	**O‘ZGARUVCHI VA O‘ZGARMAS TIPLI KATTALIKLAR**

Oddiy bir matnni ekranga chiqaruvchi dasturni tuzish sodda va tez amalga oshiriladi. Biz bir necha satr dastur yozamiz. Uni kompilatsiya qilamiz va bir necha satrdan iborat bo‘lgan iborani natija sifatida qabul qilamiz. Lekin dastur tuzish oddiy bir matnni ekranga chiqarishdangina iborat emas. Dastur tuzishda keyingi qadamlarni bosamiz. Sodda dasturlar o‘rniga foydali mashqlarni bajaradigan dasturlar tuzish uchun o‘zgaruvchilar haqida tushunchaga ega bo‘lish kerak.

6.1. O‘zgaruvchilar

Og‘zaki yechiladigan arifmetik misol bilan o‘zgaruvchilar mavzusini o‘rganishni boshlaymiz. Tasavvur qiling, 5 va 2 sonlari berilgan bo‘lsin. Sizga bu sonlarni eslab qoling, deyilganda, xotirangizda, ya’ni miyangizning qaysidir joyida 5 va 2 sonlarini joylashtirasiz. Birinchi songa 1 ni qo‘shing deyilganda, xotirangizda 6 va 2 sonlari saqlanadi. Bu sonlarning birinchisidan ikkinchisini ayiring deyilganda esa 6 dan 2 ni ayirib, 4 sonini xotirangizda hosil qilasiz. Sizning miyangizda sodir bo‘lgan bu jarayon kompyuterning xotirasida ham ikkita o‘zgaruvchi bilan amalga oshiriladi. Bu jarayon C++ tilida quyidagicha yoziladi:

```
a = 5; b = 2;  a = a + 1;  natija = a - b;
```

Bu oddiy, ikkita kichik butun son haqidagi misol edi. Ammo tasavvur qiling, kompyuter xotirasiga bunga o‘xshash millionlab sonlar joylangan bo‘lib, bir vaqtning o‘zida o‘sha sonlar bilan matematik amallar bajariladi. Shuning uchun o‘zgaruvchilarni — aniq qiymatlarni joylaydigan xotiraning bir qismi deb olamiz. Har bir o‘zgaruvchida bir-biridan farq qiluvchi tomoni bo‘lishi mumkin. Shuning uchun ularga nom — **identifikatorlar** qo‘yiladi. Misol uchun, yuqoridagi misolda o‘zgaruvchi identifikatorlari **a**, **b** va **natija** edi. O‘zgaruvchi identifikatorlariga ixtiyoriy nom berish mumkin.

6.2. Identifikatorlar

Identifikatorlar bir yoki bir necha harflar ketma-ketligini va pastki chiziqchadan „ _“ tashkil topadi. Ba‘zi kompilatorlar uchun identifikatorlar 32 ta belgi bilan belgilangan bo‘lsa ham, ularning

uzunligi chegaralanmagan. Bo'sh joy (probel) yoki belgilangan harf identifikatorning biron-bir qismi bo'la olmaydi. Faqat harflar, raqamlar va pastki chiziqcha „_" lardan tashkil topishi mumkin. Bundan tashqari, o'zgaruvchi identifikatori faqat harf bilan boshlanishi kerak. Identifikatorlar raqamlar bilan ham, pastki chiziqcha „_" bilan ham boshlanmaydi. „_" belgisi boshqa maqsadlar uchun ham ishlatilishi mumkin.

Identifikator tuzayotganda, yana shunga e'tibor berish kerakki, identifikatorlar C++ dagi kalit so'zlar bilan bir xil bo'lib qolmasligi kerak. Aks holda dastur tuzilayotganda qiyinchiliklarga duch kelish mumkin.

Quyidagilar standart C++ning kalit so'zlaridir. Ular identifikator sifatida ishlatilishi mumkin emas:
asm, auto, bool, break, case, catch, char, class, const, const_cast, continue, default, delete, do, double, dynamic_cast, else, enum, explicit, extern, false, float, for, friend, goto, if, inline, int, long, mutable, namespace, new, operator, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static, static_cast, struct, switch, template, this, throw, true, try, typedef, typeid, typename, union, unsigned, using, virtual, void, volatile, wchar_t.

6.3. Ma'lumotlar tipi

Dastur tuzilayotganda, kompyuter xotirasiga o'zgaruvchilar joylanadi. Ammo, o'zgaruvchilarga qanday qiymat berilishi kompyuterga ma'lum bo'lishi kerak. Chunki har bir raqam, belgi yoki harf xotirada bir xil joyni egallamaydi.

Kompyuter xotirasi baytlardan tashkil topgan. Bir bayt esa biz boshqarishimiz mumkin bo'lgan eng kichik xotira kattaligidir. Albatta, bir baytga kichik o'lchovdagi ma'lumot, odatda, 0 dan 255 gacha bo'lgan butun son **(integer)** yoki yagona bir belgi **(character)** joylashadi. Lekin, bundan tashqari, kompyuter uzun sonlar yoki kasr sonlar kabi baytlar guruhidan tashkil topgan murakkab ma'lumotlar tiplarini ham boshqarishi mumkin.

Yon tomondagi jadvalda siz C++ ga tegishli bo'lgan mavjud asosiy ma'lumotlar tiplari ro'yxati va ularning o'lchovlari bilan tanishishingiz mumkin.

Tipi	Baytlar	O'lchov diapazoni
char	1	isborali: - 128 dan 127 gacha ishorasiz: 0 dan 255 gacha
short	2	ishorali: - 32768 dan 32767 gacha ishorasiz: 0 dan 65535 gacha
long	4	ishorali: - 2147483648 dan 2147483647 gacha ishorasiz: 0 dan 4294967295 gacha
int	4	ishorali: - 2147483648 dan 2147483647 gacha ishorasiz: 0 dan 4294967295 gacha
float	4	3.4e + / - 38 (7 xonali raqam)
douhle	8	1.7e + / - 308 (15 xonali raqam)
long double	10	1.2e + / - 4932 (19 xonali raqam)
bool	1	true yoki false

6.4. O'zgaruvchilarni e'lon qilish

C++ da o'zgaruvchilarni ishlatish uchun qaysi tipga tegishli ekanini aniqlab olish kerak. Buning uchun birinchi bo'lib, o'zgaruvchilarni e'lon qilib olish zarur. E'lon qilish sintaksisi, avvalo, bizga qaysi tipdagi o'zgaruvchi kerak bo'lsa, o'sha tipni yozib, so'ng uning identifikatorini yozishdan iborat bo'ladi.

Misollar va izohlar

Birinchi satrda **int** tipidagi a identifikatori e'lon qilinyapti. Ikkinchi satrda esa o'zgaruvchi **float** tipi bilan **mynumber** identifikatori e'lon qilinyapti. E'lon qilingan a va **mynumber** o'zgaruvchilari dasturning boshqa qismlarida ham ishlatilishi mumkin.

```
int a;
float mynumber;
```

Agar siz bir xil tipdagi bir necha o'zgaruvchini e'lon qilmoqchi bo'lsangiz, ish joyini tejash uchun, barcha o'zgaruvchini bitta satrga, bir-biridan vergul bilan ajratgan holda yozishingiz mumkin.

Bu bo‘limdagi o‘rganilgan o‘zgaruvchilar haqidagi bilimlarni mustahkamlash maqsadida, C++ da o‘zgaruvchilarni e’lon qilish va ular bilan ishlash haqidagi namunaviy dasturni quyida ko‘rib chiqamiz.

6.5. O‘zgaruvchilarni initsializatsiya qilish

O‘zgaruvchi e’lon qilinganida kompyuter xotirasida o‘zgaruvchi haqidagi eski keraksiz ma’lumot bo‘ladi. Lekin bu ma’lumot o‘rniga yangi aniq bir son yoki belgi qo‘yish mumkin. Buning uchun e’lon qilingan o‘zgaruvchidan keyin tenglik belgisi, keyin esa biron-bir qiymat yoziladi.

```
tur identifikator = ilk_qiymat;
```

Misol uchun, **int** tipida **a** o‘zgaruvchini e’lon qilib, unga 0 qiymat berish uchun quyidagicha yozamiz:

```
int a = 0;
```

Bundan tashqari, C++ da o‘zgaruvchilarni initsializatsiya qilishning boshqa bir usuli bo‘yicha qiymatni qavs () ichida berish mumkin:

```
tur identifikator = ilk_qiymat;
```

Masalan: **int a(0);**

Misollar va izohlar

```
// ozgaruvchilar bilan dasturlash
#include <iostream.h>
int main ()
{
  // o‘zgaruvchilarni e’lon qilish:
  int a, b;
  int result;
// jarayon:
a = 5;
b = 2;
a = a + 1;
result = a - b;
// natijani ekranga chiqarish:
cout << result;
// dasturni o‘chirish:
return 0;
```

MASHQLAR

1. Quyidagi raqamlar qaysi ma'lumot tiplariga tegishli ekanini aniqlang:
   1) 500; 2) 0.0094; 3) -280; 4) 69.745;
   5) -0.03; 6) -150.0004; 7) 10^{-4}; 8) $-5.85 \cdot 10^{3}$.
2. Quyidagilardan qaysilari identifikator bo'la olmaydi? Nega?
   1) A15a; 2) 1B17C; 3) nnNN;
   4) x+3; 5) BU15; 6) B^{7}.
3. a harfi va 5 sonini o'z ichiga olgan 10 ta turli xil identifikator tuzing.
4. 10 ta turli xil o'zgaruvchilarni e'lon qiling va ularni ekranga chiqaring.
5. Konsolda 5 ta sonni shunday kiritingki, ular ekranda ikki marotaba yozilib chiqsin.

Nazorat savollari

1. O'zgaruvchi nima?
2. Identifikator nima?
3. Identifikator tuzayotganda nimalarga e'tibor berish kerak?
4. Ma'lumotlar tipi nima?
5. O'zgaruvchilar qanday e'lon qilinadi?
6. O'zgaruvchilar qanday initsializatsiya qilinadi?
7. O'zgaruvchilarni initsializatsiya qilishning qanday **usullari** bor?

7-mavzu DASTURLASH OPERATORLARI

O'zgaruvchilar haqida tushunchaga ega bo'ldik, endi esa ular bilan ishlashni o'rganishni boshlaymiz. Buning uchun C++ dasturlash tili kalit so'zlardan iborat bo'lgan operatorlar va ma'lum bir vazifalarni bajaradigan belgilarni taqdim qiladi. Bular C++ tilining asosiy qismi bo'lganligi uchun ularni puxta o'rganish juda muhim.

7.1. C++ dasturlash tilidagi operatsiyalar

Dasturning asosiy vazifasi boshlang'ich ma'lumotlarni qayta ishlab, qo'yilgan masalaning natijasini beruvchi amallarni, ya'ni operatsiyalarni bajarishdan iborat.

Qo‘yilgan masalani yechish maqsadida boshlang‘ich ma’lumotlarni qayta ishlash amallari operatsiyalar deb ataladi.

C++ tilida ma’lumotlar ustida turli xil operatsiyalar amalga oshiriladi. Bu operatsiyalarni operatorlar amalga oshiradi. C++ tilida ishlatiladigan operatsiyalar quyidagi sinflarga bo‘linadi:

1. Arifmetik operatsiyalar:

+	qo‘shish;	/	bo`lish;
-	ayirish;	%	modul olish.
*	ko‘paytirish ;		

2. Taqqoslash operatsiyalari

= =	tengmi? (tenglikka tekshirish);	<=	kichik yoki teng;
!=	teng emasmi?	>	katta;
<	kichik;	>=	katta yoki teng.

3. Mantiqiy operatsiyalar

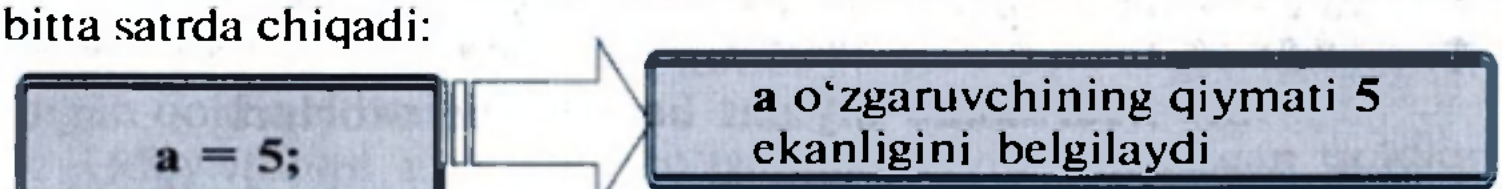
&& „va“ (mantiqiy ko‘paytirish) operatsiyasi;
|| „yoki“ (mantiqiy qo‘shish) operatsiyasi;
! „emas“ operatsiyasi;

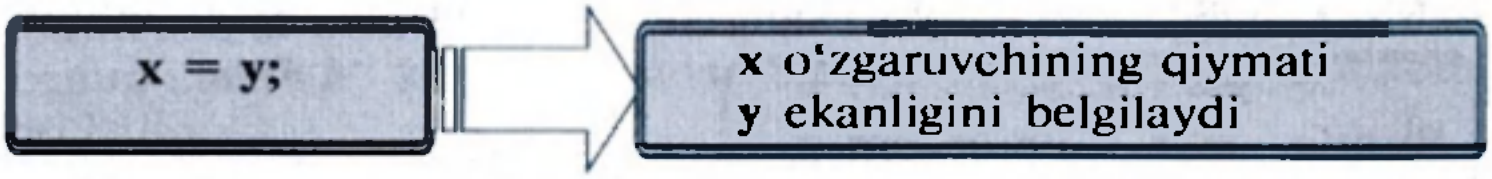
4. Qiymat berish yoki tenglik (=) operatori.

Tenglik operatori o‘zgaruvchining kattaligini belgilash uchun xizmat qiladi.

Misollar va izohlar

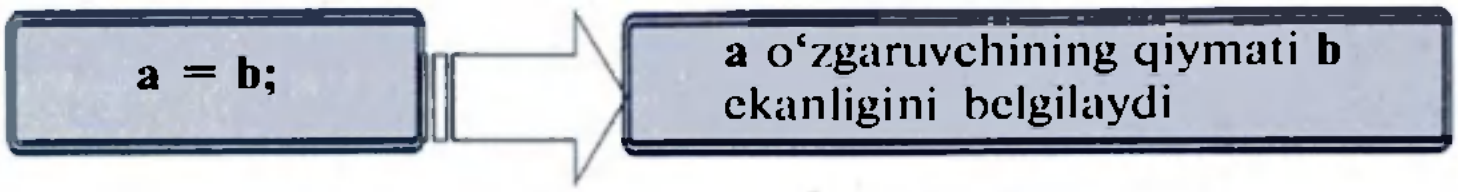
Ikki ibora alohida ikki satrga yozilgan bo‘lsa ham ekranda natija bitta satrda chiqadi:





Tenglik operatorining chap va o‘ng tomonida ikki xil belgi bor. Chap tomondagi belgi har doim e’lon qilingan o‘zgaruvchidan iborat, o‘ng tomonda esa yangi o‘zgaruvchi yoki ozgaruvchining qiymati bo‘lishi mumkin.

Shuni e'tiborga olish kerakki, qiymat berish operatsiyasi chap tomondan o'ng tomonga emas, balki o'ng tomondan chap tomonga qarab amalga oshiriladi.



Bu misolda **b** o'zgaruvchining keyingi qiymatlari **a** ning qiymatiga ta'sirini o'tkazmaydi. Masalan, quyidagi misolni olaylik:

Ko'rinib turibdiki, natija **a** ning qiymati 4 ga teng, **b** ning qiymati esa 7 ga tengligini ko'rsatyapti. Avval biz **a = b** deb e'lon qilgan bo'lsak ham, **b** qiymatining oxirgi marta o'zgarishi **a** ga ta'sir qilgani yo'q.

```
int a, b;    // a:?  b:?
a = 10;      // a:10 b:?
b = 4;       // a:10 b:4
a = b;       // a:4  b:4
b = 7;       // a:4  b:7
```

C++ da butun bir tenglikni o'zgaruvchining qiymati sifatida yozish mumkin.

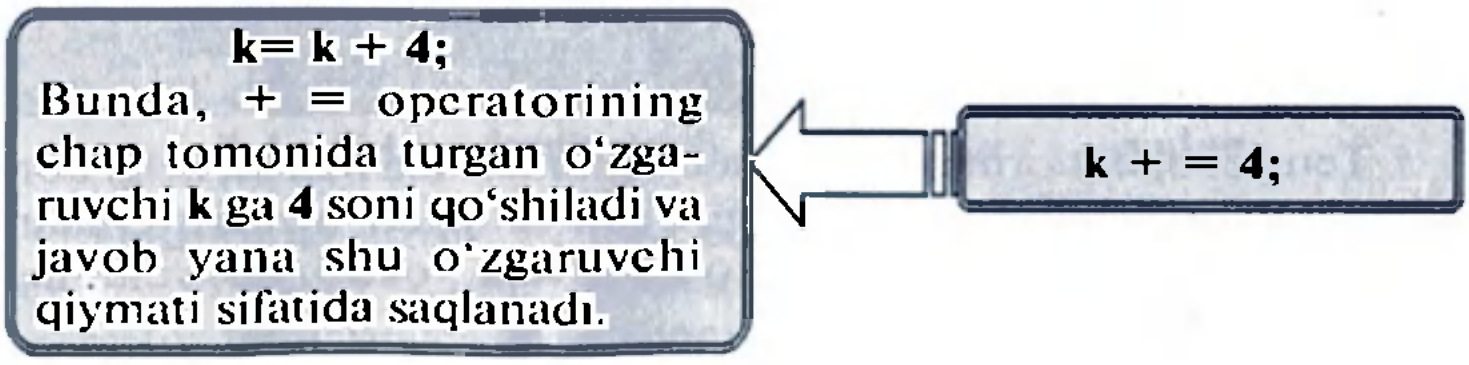
Misollar va izohlar

Bunda, **b** 5 ga teng, **a** esa 2 bilan avvalgi tenglik natijasining yig'indisiga teng. Oxirgi natija — **a** ning qiymati 7 ga teng bo'ladi.

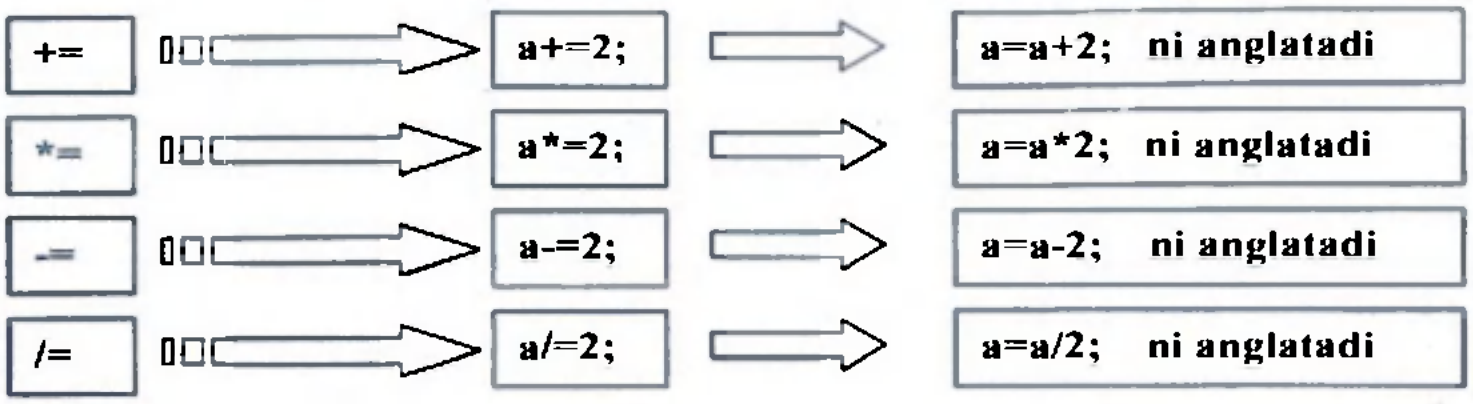
C++ da quyidagicha ifoda ham mavjuddir:

5. Murakkab qiymat berish operatorlari

C++ da hisoblash bajaruvchi va undan keyin javobni o'zgaruvchiga beruvchi bir necha operator mavjud. Misol uchun,



+= operatoriga o‘xshash -=, /=, *= va %= operatorlarini ham tavsiflash mumkin:



7.2. Arifmetik operatorlar

Arifmetik operatorlar bilan yuqorida tanishgan edik.

Arifmetik operatsiyalar:

+	qo‘shish;	/	bo‘lish;
—	ayirish;	%	modul olish.
*	ko‘paytirish ;		

Ulardan ba‘zi birlarining xususiyatlarini ko‘rib chiqaylik.

1. Butun sonli bo‘lishda, ya’ni bo‘luvchi ham, bo‘linuvchi ham butun son bo‘lganda, javob butun son bo‘ladi. Natijaning kasr qismi tashlab yuborilib, butun qismining o‘zi qoladi.

2. Modul operatori (%) butun songa bo‘lishdan kelib chiqadigan qoldiqni beradi.

x%y ifodasi x ni y ga bo‘lgandan keyin chiqadigan qoldiqni beradi. Demak, 7%4 bizga 3 javobini beradi. % operatori faqat butun sonlar bilan ishlaydi. Vergulli (haqiqiy) sonlar bilan ishlash uchun „**math.h**“ kutubxonasidagi **fmod** funksiyasini qo‘llash kerak bo‘ladi.

3. C++ tilida qavslarning ma’nosi va bajarilish ketma-ketligi algebraik xususiyatga ega.

4. Algebraik amallarni bajarish ketma-ketligi ham odatdagidek. Oldin ko‘paytirish, bo‘lish va modul olish operatorlari bajariladi.

Agar bu operatorlarning bir nechtasi ketma-ket kelsa, ular chapdan o'ngga qarab bajariladi. Bu operatorlardan keyin esa qo'shish va ayirish bajariladi.

Misollar va izohlar

```
k = m * 5 + 7 %
n / (9 + x);
```

Birinchi bo'lib **m * 5** hisoblanadi. Keyin **7 % n** topiladi va qoldiq **(9 + x)** ga bo'linadi. Chiqqan javob esa **m * 5** ning javobiga qo'shiladi.

Qisqasini aytsak, amallar matematikadagi kabi tartibda bajariladi. Lekin, o'qishni osonlashtirish va xato qilish ehtimolini kamaytirish maqsadida, qavslarni kengroq ishlatish mumkin. Yuqoridagi misol quyidagi ko'rinishda berilishi ham mumkin edi:

```
k = ( m * 5 ) + ( ( 7 % n ) / ( 9 + x ) );
```

7.3. Qiymatni bir birlikka o'zgartiruvchi operatorlar

C++ da bir argument oluvchi **inkrenet** (++) va **dekrement** (--) operatorlari mavjud.

++ operatori yordamida o'zgaruvchining qiymati bir birlikka oshiriladi.

-- operatori yordamida o'zgaruvchining qiymati bir birlikka kamaytiriladi. Yangi bir birlikka oshirilgan (kamaytirilgan) qiymatlar keyingi ifodalarda qo'llaniladi.

inkrenet operatsiyasi	**+ +**	***a* + +;**	**$a = a + 1$; ni anglatadi**
dekrement operatsiyasi	**- -**	***b* - -;**	**$b = b - 1$; ni anglatadi**

Bir birlikka oshirish va kamaytirish operatorlari yozilganda, ularning argumentlari orasida bo'sh joy qoldirilmasligi kerak. Bu operatorlar sodda ko'rinishdagi o'zgaruvchilarga nisbatan qo'llanilishi mumkin, xolos.

Masalan: **++(f * 5);** yozuv noto'g'ri.

Yuqorida tavsiflangan operatorlar qo'llanilgan quyidagi dasturni ko'rib chiqaylik.

Dasturdagi o'zgaruvchilar e'lon qilindi. Boshlang'ich qiymatlar olindi. **cout << k++ << endl**; ifodasida ekranga oldin **k** ning boshlang'ich qiymati chiqarildi, keyin esa uning qiymati 1 ga oshirildi. **l += 4**; da l ning qiymatiga 4 soni qo'shildi va yangi qiymat l da saqlandi. **cout << —m << endl**; ifodasida **m** ning qiymati oldin prindekrement qilindi, undan so'ng ekranga chiqarildi. **m = k + (++l)**; da oldin l ning qiymati birga oshirildi va l ning yangi qiymati **k** ga qo'shildi. **m** esa bu yangi qiymatni oldi.

```
//Postinkremet, preinkrement va qisqartirilgan teglashtirish operatorlari
# include <iostream.h>
 int main()
{
int k = 5, l = 3, m = 8;
cout << k++ << endl;        // ekranga 5 yozildi, k = 6 bo'ldi.
l += 4;                     // l = 7 bo'ldi.
cout << —m << endl;   // m = 7 bo'ldi va ekranga 7 chiqdi.
m = k + (++l);              // m = 6 + 8 = 14;
return (0);
}
```

7.4. Taqqoslash operatorlari

Ikki ifodani bir-biri bilan taqqoslash uchun taqqoslash operatorlaridan foydalaniladi. Biz taqqoslash operatorlari bilan yuqorida tanishgan edik.

==	**tengmi?**
!=	**teng emasmi?**
>	**katta**
<	**kichik**
>=	**katta yoki teng**
<=	**kichik yoki teng**

C++ standartida solishtirtish operatsiyasi natijasi **true** (rost) yoki **false** (yolg'on) qiymatini beruvchi **bool** mantiqiy operatsiyasi hisoblanadi.

Misollar va izohlar

Bularga quyidagilar misol bo'la oladi:

(7 == 5) **false** qiymat qabul qiladi.
(5 > 4) **true** qiymat qabul qiladi.
(3 != 2) **true** qiymat qabul qiladi.
(6 >= 6) **true** qiymat qabul qiladi.
(5 < 5) **false** qiymat qabul qiladi.

MASHQLAR

1. int i=5, x=3 bo'lsa, quyidagi amal bajarilgandan keyin x qiymatini aniqlang: x=i=i+2;
2. a va b o'zgaruvchilarning tiplari int va z o'zgaruvchining tipi **float** bo'lsin. Quyidagi amal bajarilgandan keyin z ning qiymatini toping: a=2; b=1.4; z=(a+b·2·a)·3.
3. q o'zgaruvchining tipi **int** va w o'zgaruvchining tipi **float** bo'lsin. Quyidagi amal bajarilgandan keyin w ning qiymatini toping: q=3; $w=(15*q)^2$.
4. Quyidagi oddiy mantiqiy ifodalar qiymatini aniqlang:
   1) agar a=2 va b=3.2 bo'lsa, a>b;
   2) agar a=3, b=4 va c=5 bo'lsa, a<b<5;
   3) agar a=2, b=2, c=3 bo'lsa, a≤b>c;
   4) agar a=2, b=1, c=1 bo'lsa, a+b<2 c;
5. Quyidagilarni qabul qiladigan mantiqiy ifoda tuzing: agar a≤x≤b bo'lsa – **true** (rost), aks holda **false** (yolg'on) bo'lsin.

Nazorat savollari

1. Operatsiya va operator tushunchalarini tushuntirib bering.
2. C++ tilida operatsiyalar nechta sinfga bo'linadi?
3. Har bir sinfga kiruvchi operatorlarni sanang va ularning belgilanishini yozing.
4. Murakkab qiymat berish operatorlariga misollar keltiring.
5. Qiymatni bir birlikka oshiruvchi va kamaytiruvchi operatorlar haqida nimalar bilasiz? Misollar keltiring.
6. Modul olish operatorining ta'sirini tushuntirib bering.
7. Taqqoslash operatorlari ta'sirini tushuntirib bering.

8-mavzu	**SHARTLI OPERATORLAR**

Ba'zida dasturning bajarilish jarayonida amallarni bajarish ketma-ketligi biror shartga asosan o'zgarib, tarmoqlanib ketadi. Bunday dasturlar tarmoqlangan dasturlar deb ataladi.

Agar dasturning bajarilish jarayonida amallarni bajarish ketma-ketligi biror shartga asosan o'zgarib, ya'ni tarmoqlanib ketsa, bunday dastur tarmoqlangan dastur deb ataladi.

Quyida shart operatori ta'sirida tarmoqlanadigan dasturlar ustida to'xtalamiz.

8.1. Shartli operatorlar tavsifi

Odatda, biron-bir mezonga, shartga ko'ra bir necha imkoniyatlar ichidan bittasini tanlashga to'g'ri kcladi. Buni C++ tilida **if** operatori yordamida yoziladi:

```
if (shart ifodasi)
{operator};
```

Misollar va izohlar

Agar bolaning yoshi 7 ga teng yoki katta bo'lsa, u maktabga borishi mumkin. Bu yerda shart bajarilishi yoki bajarilmasligi mumkin. Agar yosh o'zgaruvchisi 7 ga teng yoki undan katta bo'lsa, shart bajariladi va maktab () funksiyasi chaqiriladi. Bu holatda ifoda true (rost) deyiladi. Agar yosh 7 dan kichik bo'lsa, maktab () tashlab o'tiladi. Ya'ni false (yo'lg'on) holat vuzaga keladi.

```
if (yosh >= 7)
maktab();
```

Ko'rib turganingizdek, operatorning shart ifodasi mantiqiy operatorlarga asoslangan. Aslida esa shartdagi ifodaning ko'rinishi

muhim emas. Agar ifodani nolga keltirish mumkin bo'lsa, **false** bo'ladi, noldan farqli javob bo'lsa, musbatmi, manfiymi, **true** holat paydo bo'ladi va shart bajariladi. Bunga qo'shimcha qilib o'tish kerakki, C++ da maxsus o'zgaruvchilarning **bool** (mantiqiy) tipi mavjud. Bu tipdagi o'zgaruvchilar yordamida **bool** (mantiqiy) algebrasini amalga oshirish mumkin. **Bool** o'zgaruvchilari faqat **true** (rost) yoki **false** (yolg'on) qiymatlarni qabul qilishi mumkin.

8.2. Shartli operatorning if/else strukturasi

If operatorini qo'llaganimizda, shart ifodasi faqat haqiqat bo'lgandagina bajariladi. Aks holda tashlab o'tiladi. **if/else** yordamida esa shart bajarilmaganda, ya'ni **false** (yolg'on) natija chiqqanda, **else** orqali boshqa bir yo'ldan boriladi.

Misollar va izohlar

```
if (yosh >= 7)
   maktab();
else
   bog'cha();
```

Bu misolda bola 7 yoshli yoki undan katta bo'lsa, maktabga, 7 yoshdan kichik bo'lsa, bog'chaga boradi.

C++ da bitta ifoda turgan joyga ifodalar guruhini {} qavslarda olingan holda qo'ysa bo'ladi. Masalan:

```
if (yosh >= 7)
{
   cout << "Maktabga!\n";  maktab();
 }
else
 {
   cout << "Bog'chaga!\n";  bog'cha();
 }
```

If/else strukturalarini bir-birining ichida yozishimiz mumkin. Bunda ular bir-biriga ulanib ketadi. Misol uchun tezlikning kattaligiga qarab jarimani belgilab beruvchi blokni yozaylik.

```
if (tezlik > 120)
  cout << "Jarima
100 so‘m";
else if (tezlik > 100)
   cout << "Jarima
70 so‘m";
else if (tezlik > 85)
  cout << "Jarima
30 so‘m";
else
  cout << "Tezlik
me’yorda";
```

Agar tezlik 120 dan katta bo‘lsa, birinchi **if/else** strukturasining sharti bajariladi va bu holda albatta tezlik o‘zgaruvchimizning qiymati ikkinchi va uchinchi **if/else** strukturani ham qanoatlantiradi. Lekin solishtirish ulargacha bormaydi, chunki ular birinchi **if/else** ning **else** qismida, ya‘ni noto‘g‘ri javob qismida joylashgandir. Solishtirish birinchi **if/else** da tugashi (aynan shu misolda) tanlash amalini tezlashtiradi. Ya‘ni bir-biriga bog‘liq **if/else** lar alohida **if** strukturalari blokidan tezroq bajarilishi mumkin, chunki birinchi holda **if/else** blokidan vaqtliroq chiqish imkoni bor. Shu sababli ichma-ich joylashgan **if/else** operatorlar guruhida **true** (rost) bo‘lish imkoni ko‘proq bo‘lgan shartlarni oldinroq tekshirish kerak.

MASHQLAR

1. Shartli operatorning ishlashini tushuntiring;

```
if (i < j)
{
    i++;
}
else
{
    j = i-3;
    i++;
}
```

2. Shartli operatorning ishlashini tushuntiring:

```
int t = 2;
int b = 7;
```

```
int r = 3;
if (t>b)
{
        if (b < r)
        {
                r = b;
        }
}
else
{
        r = t;
        return (0);
}
```

3. If/else shartli operatorlaridan foydalangan holda, quyidagi ifodalardan dastur tuzing. Agar ifoda to‘g‘ri bo‘lsa, „Bu ifoda to‘g‘ri“, aks holda, „Bu ifoda noto‘g‘ri“ so‘zlari ekranga chiqsin.

4. Quyidagi oddiy mantiqiy ifodalar qiymatini aniqlang:

1) agar a=2 va b=3.2 bo‘lsa, a>b;
2) agar a=3, b=4 va c=5 bo‘lsa, a<b<5;
3) agar a=2, b=2, c=3 bo‘lsa, a≤b>c;
4) agar a=2, b=1, c=1 bo‘lsa, a+b<2·c.

Nazorat savollari

1. Shartli operatorlarni tavsiflab bering.
2. Shartli operatorlarga misollar keltiring.
3. Shartli operatorning **if\else** strukturasini tushuntirib bering va misol keltiring.
4. Ichma-ich joylashgan shartli operatorlarning ishlashini tushuntirib bering.

9-mavzu	**TAKRORLASH(SIKL) OPERATORLARI**

Ba’zi amaliy masalalarni yechish algoritmlarini tuzish jarayonida, ko‘p hollarda bir xil amalni qayta-qayta bajarishga to‘g‘ri keladi. Boshqa dasturlash tillari singari C++ tilida ham bunday qayta-qayta takrorlanadigan amallar „takrorlash operatori“ deb

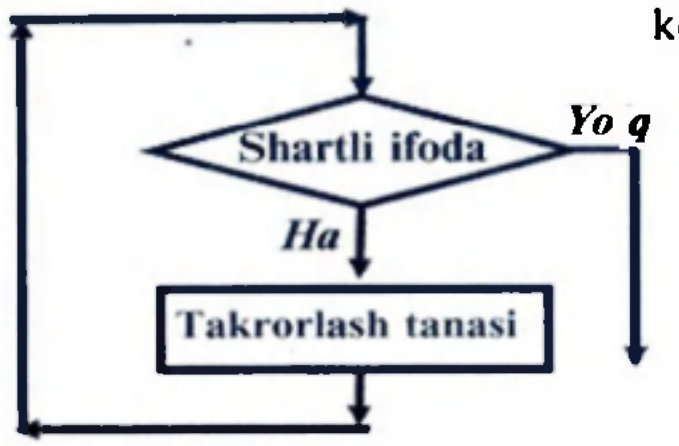
nomlanadigan operator yordamida amalga oshiriladi. C++ tilida takrorlash operatorining uch xil ko'rinishi mavjud va ular quyidagi nomlar bilan yuritiladi: **while, do, for.**

9.1. While takrorlash operatori

Bir yoki bir nechta ifodalar blokida ko'rsatilagan amallarni ma'lum hir shart to'g'ri (rost) bo'lib turgunga qadar qayta-qayta takrorlash while takrorlash operatori yordamida amalga oshiriladi.

Ravshanki, qaytarilayotgan amallar shartga ham ta'sir qiladi va qaysidir qadamdan so'ng berilgan shart bajarilmay qoladi, ya'ni yo'lg'on bo'lib qoladi. Bu takrorlashning nihoyasiga yetganini bildiradi. **While** operatori faqat o'zidan keyin kelgan ifodaga ta'sir qiladi. Shuning uchun agar biz bir guruh amallar blokini qaytarmoqchi bo'lsak, ushbu blokni { } qavslar ichiga olishimiz kerak.

While takrorlash operatori quyidagi ko'rinishga ega:




```
while(shartli ifoda)
takrorlash tanasi
```

While takrorlash operatorining blok-sxemasi

MISOL. 1 dan 10 gacha bo'lgan sonlar ko'paytmasini hisoblang.

```
int factorial = 1;
int son = 1;

while (son < 11)
{
  factorial = factorial * son;
  son = son + 1;
}
```

E S L A T M A !

„While“ so‘zi o‘zbek tilida „mobaynida“, „bo‘lgunga qadar“ degan ma’noni anglatadi.

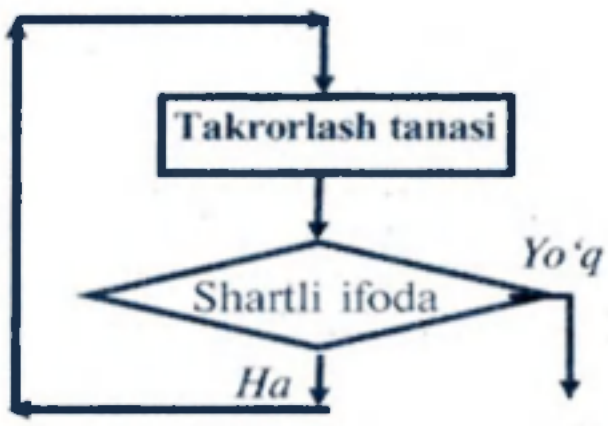
Shartli ifoda sifatida ko‘proq o‘zgaruvchili munosabat yoki mantiqiy ifoda ishlatiladi. Takrorlash tanasi qaytarilishi kerak bo‘lgan amallar ro‘yxatidan iborat bo‘lib, shartli ifoda rost bo‘lib turgunga qadar qayta-qayta takrorlanadi.

Takrorlash tanasi bu takrorlash operatsiyasi ichida amalga oshiriladigan amallar jamlanmasi hisoblanadi. Takrorlash tanasi takrorlash operatsiyasi{} qavslari ichiga yoziladi.

9.2. Do/while takrorlash operatori

Do takrorlash operatori — while operatoriga o‘xshash. Do takrorlash operatorining While operatoridan farqi: undagi shartli ifoda takrorlash tanasidan keyin keladi.

Shu bois, takrorlanuvchi jarayon shart tekshirilgunga qadar bir marta amalga oshib bo‘ladi. Shart rost bo‘lsa, takrorlash jarayoni qaytariladi va bu jarayon toki shart bajarilmay qolgunga qadar davom etadi. Agar **do** operatori tanasidagi qaytarilishi kerak bo‘lgan amallar ko‘p bo‘lsa, ular {} qavslar ichiga olib qo‘yiladi.



Do takrorlash operatorining blok-sxemasi.

Do takrorlash operatori quyidagi ko‘rinishga ega:

```
do
takrorlash tanasi;
while (shartli ifoda);
```

MISOL. 5 ning darajasi bo‘lgan, 1000 dan kichik eng katta sonni toping.

```
int k=1
do {
        k=k*5
} while (k<=1000);
```

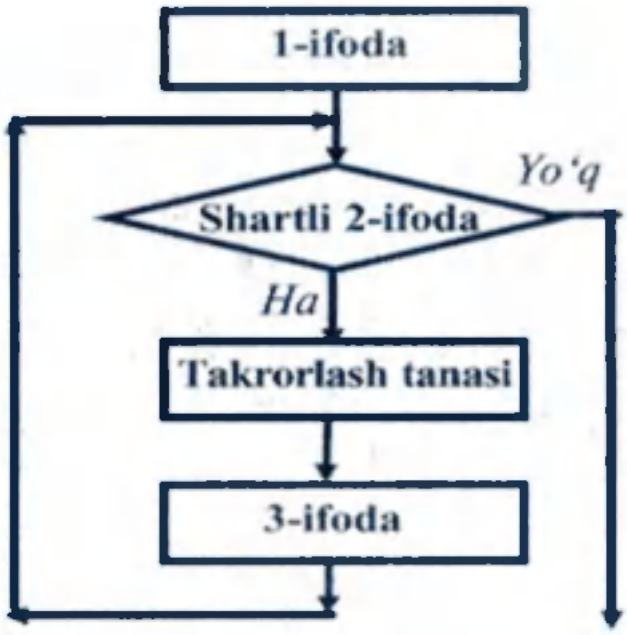
E S L A T M A !

Do operatorida {} qavslarning yo'qligi dasturchini adashtirishi mumkin. Chunki qavssiz **do** operatori oddiy **while** ning boshlanishiga o'xshaydi. Buning oldini olish uchun {} qavslarni har doim qo'yishni tavsiya etamiz.

9.3. For takrorlash operatori

For operatori sanoq bilan bajariladigan takrorlashni bajaradi. Oldingi takrorlash operatorlarida ham takrorlashlar sonini nazorat qilish uchun sanovchini qo'llasa bo'lar edi. Buning uchun takrorlanishlar sonini o'ldindan bilish zarur edi. Agar **for** operatorida bir necha ifoda takrorlanishi kerak bo'lsa, ifodalar bloki {} qavs ichiga olinadi.

For takrorlash operatori quyidagi ko'rinishga ega:



For takrorlash operatorining blok-sxemasi.

```
for ( 1-ifoda; 2-ifoda; 3-ifoda)
{
   takrorlash tanasi
}
```

Bu yozuvdagi:

1-ifoda boshlang'ich shart bo'lib, u takrorlashda qatnashayotgan o'zgaruvchilarni e'lon qilish va ularga boshlang'ich qiymat berishni ifodalaydi;

2-ifoda odatdagi shartli ifoda bo'lib, takrorlash jarayonining davom etishi yoki tugashini anglatadi;

3-ifoda takrorlash jarayonida ishtirok etayotgan o'zgaruvchilarning qiymati qanday o'zgarishini ifodalaydi.

For operatori ta'sirida takrorlash jarayoni quyidagi tartibda kechadi:

• 1-ifodaga asosan ozgaruvchilar e'lon qilinadi, ularga boshlang'ich qiymatlar beriladi. So'ng 2-ifodaga o'tiladi va kelgusida bu ifodaga qaytilmaydi;

• 2-ifodadagi shartli ifoda tekshiriladi. Agar u rost bo'lsa, takrorlash tanasiga o'tiladi;

- takrorlash tanasida ko'rsatilgan amallar bajariladi;
- 3-ifodaga asosan o'zgaruvchilarga yangi qiymatlar beriladi.
- song, yana 2-ifodaga, ya'ni shartli ifodaga o'tiladi va hokazo.

Takrorlash jarayoni shu tartibda davom ettiriladi.

MISOL. $x_i=2i+5$ ketma-ketlikning 5 ta hadini hisoblab, monitorda yozib ko'rsatadigan dastur tuzing.

```
# include <iostream.h>
int main()
{
 for (int i = 0; i < 5; i++)
{
    cout << 2i+5 << endl;
 }
 return (0);
}
```

```
5
7
9
11
13
```

MISOL. „Assalomu alaykum!" so'zini monitorda 10 marta yozib ko'rsatadigan dastur tuzing.

```
for (int i = 0; i < 10 ; i++)
cout << "Assalomu alaykum!"<< endl;
```

E S L A T M A !

1. Endi **for** operatorini tashkil etuvchi uchta qismninig har birini alohida ko'rib chiqsak. Birinchi qismda, asosan, takrorlashni boshqaradigan sanovchi o'zgaruvchilar e'lon qilinadi va ularga boshlang'ich qiymatlar beriladi. Yuqoridagi dasturda buni **int i = 0;** deb yozganmiz. Bu qismda bir necha o'zgaruvchilarni e'lon qilmoqchi bo'lsak, ularni vergul bilan ajratib berishimiz mumkin. Ayni shu kabi uchinchi qismda ham bir nechta o'zgaruvchilarning qiymatini o'zgartirishimiz mumkin.

2. Ba'zida **for** operatorining qismlari tushirib qoldirilishi mumkin. Masalan:

```
for(;;)
{ }
```

ifodasi cheksiz marta qaytariladi. Bunday **for** operatoridan chiqish uchun **break** operatori qo'llaniladi.

3. Ba'zida **for** operatorining 3-qismi tushirib qoldirilib, o'zgaruvchi qiymatini o'zgartirish takrorlash tanasi ichida amalga oshriladi. Masalan:

```
for(int g = 0; g < 10; )
    {
  cout << g;   g++;
    }
```

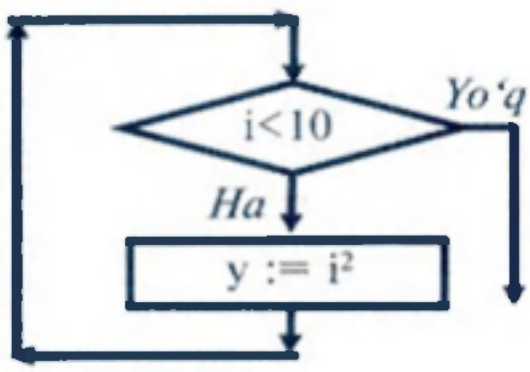
4. Standartga ko'ra 1-qismda e'lon qilingan o'zgaruvchilarning qo'llanish sohasi faqat o'sha **for** operatori bilan chegaralanadi. Yani bitta blokda joylashgan ikkita **for** operatori mavjud bo'lsa, ular uchun ayni bir xil ismli o'zgaruvchilarni qo'llab bo'lmaydi.

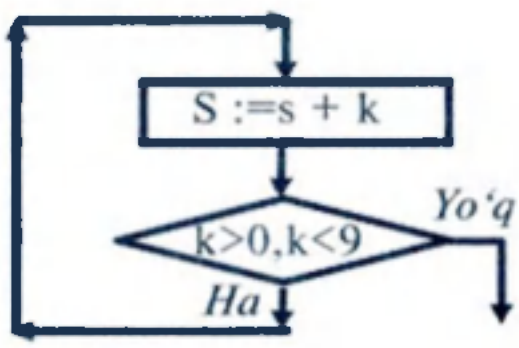
MASHQLAR

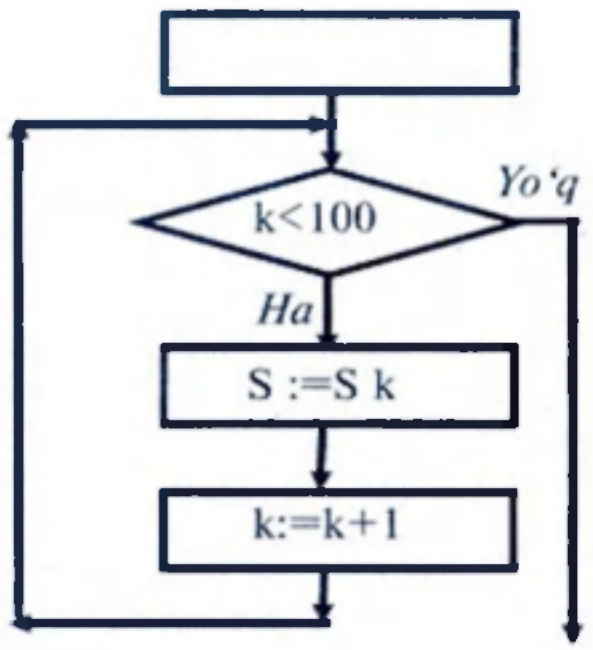
1. 1 dan 100 gacha bo'lgan butun sonlar yig'indisini hisoblang.
2. 1 dan 100 gacha bo'lgan butun sonlar ko'paytmasini hisoblang.
3. 100 dan 300 gacha bo'lgan juft sonlar yig'indisini hisoblang.
4. Takrorlash operatorlari yordamida „Biz kollejda o'qiymiz!" degan gapni 10 marta monitorda yozadigan dastur tuzing.
5. Takrorlash operatorlari yordamida „A'lo o'qish — burchimiz!" degan gapni 8 marta monitorda yozadigan dastur tuzing.
6. $y = 2x^2 - 5$ funksiyaning x ning 1 dan boshlab, 0,5 qadam bilan hosil qilinadigan 10 ta nuqtadagi qiymatlarini toping.
7. $a_n = (n+2)/(n^2+4)$ ketma-ketlikning:
   a) birinchi 15 ta hadini;
   b) 10-dan 20-hadigacha bo'lgan hadlarini hisoblaydigan dastur tuzing.
8. 7 ning darajasi bo'lgan, 10000 dan kichik eng katta sonni toping.
9. 6 ning darajasi bo'lgan, 10000 dan katta eng kichik sonni toping.
10. for (int y = 100; y >= 0; y-=5) { ..., ifoda(lar); ...} ko'rinishdagi dasturda y o'zgaruvchi qaysi chegarada va qanday o'zgaradi? Bu dasturda takrorlash tanasi necha marta qaytarilyapti?

11. for(int d = -30; d<=30; d++) {... , ifoda(lar); ... } ko'rinishdagi dasturda d o'zgaruvchi qaysi chegarada va qanday o'zgaradi? Bu dasturda takrorlash tanasi necha marta qaytariladi?

12. Ushbu blok-sxemalar asosida dasturlar tuzing va ularni sharhlang:







Nazorat savollari

1. Takrorlash operatorini tavsiflab bering.
2. **While** operatoriga ta'rif bering.
3. **While** operatorining blok-sxemasini chizib ko'rsating.
4. **Do** takrorlash operatorini ta'riflang va unga doir misollar keltiring.
5. **For** operatoriga ta'rif bering va misol keltiring.
6. **For** takrorlash operatori bilan **while** operatorining farqini aytib bering ?
7. **For** operatorining tuzilishini va ishlash tamoyilini tushuntirib bering.

10-mavzu MASSIVLAR

Bu mavzuda dasturlash tilidagi ma'lumotlarning tarkibi bilan tanishishni boshlaymiz. Dasturda statik va dinamik turdagi ikki asosiy ma'lumot tuzilmalari mavjud. Statik turdagi ma'lumot deganda xotirada egallagan joyi o'zgarmas, dastur boshida beriladigan

tuzilmalarni nazarda tutamiz. Dinamik turdagi ma'lumot deganda esa dastur davomida o'z hajmini, xotirada egallagan joyini o'zgartirishi mumkin bo'lgan ma'lumot tuzilmalarini tushunamiz.

10.1. Massivlar haqida tushuncha

Ko'p hollarda, bir necha sonlarni yoki o'zgaruvchilarni bitta nom ostida tizib chiqishga to'g'ri keladi. Buning uchun quyidagicha tartibda ish bajariladi. Bu sonlar bitta to'plamga (massivga) tizib chiqiladi va unga biror nom beriladi.

Bitta nom bilan ataladigan va ma'lum tartibda joylashtirilgan sonlar (o'zgaruvchilar) to'plami massiv deb ataladi. Massivga kirgan har bir son (o'zgaruvchi) massivning elementi deb yuritiladi.

Massivning elementlari massivda tizilgan o'rniga qarab nomerlab chiqiladi va bu nomerlar massiv elementlarining indeksi deb yuritiladi. Massivga murojaat qilinayotganda, massiv nomi ko'rsatiladi va uning yoniga [] qavslar ichiga bu massivning jami elementlar soni ham yozib qo'yiladi. Massivga kirgan har bir o'zgaruvchi ham massiv nomi bilan yuritiladi va nom yoniga uning bu massivdagi tartib raqami ham yozib qo'yiladi.

Misol

Masalan, biror massivga B nomi berilgan bo'lib, u 1,6; 1,1; 2,4; 5,9 va 10,6 sonlarini birlashtirgan, ya'ni uning 5 ta elementi bor bo'lsin. U holda bu massiv B[5] tarzida belgilanadi. Uning birinchi elementi B[0] := 1,6; ikkinchi elementi B[1] := 1,1; va hokazo, beshinchi elementi B[4] := 10,6 bo'ladi.

Massivlar dasturlashda eng ko'p qo'laniladigan ma'lumot turlari hisoblanadi. Bundan tashqari, ma'lumot tuzilmalari turlicha o'zgaruvchilardan tashkil topgan bo'lishi mumkin. Bunday ma'lumot tuzilmasi *klass* deb nomlanadi. Masalan, bunday tuzilmada odam ismi va yoshi bo'lishi mumkin.

Massivlar xotirada ketma-ket joylashgan, bir tipdagi o'zgaruvchilar guruhidir. Alohida bir o'zgaruvchini ko'rsatish uchun massiv nomi va kerakli o'zgaruvchi indeksi yoziladi. C++ dasturlash tilida massivlardagi elementlar indeksi har doim noldan boshlanadi. [] qavslar ichidagi indeks butun son yoki butun songa olib keluvchi ifoda bo'lmog'i kerak. Oxirgi element indeksi n—1 bo'ladi (n — massiv elementlari soni).

Massivlar bilan ishlaganda eng ko‘p yo‘l qoyiladigan xatolik n ta elementi bo‘lgan massivga 0 dan kichik va n—1 dan katta indeks bilan murojaat qilishdir!

Misol

Uch dona elementi bor bo‘lgan **char** tipidagi **m** nomli massiv quyidagicha tavsiflanadi:

m[0] -> -44 , m[1] -> 109, m[2] -> 23

Ko‘rib turganimizdek, massivning elementiga murojaat qilish uchun massiv nomi va [] qavslar ichiga element indeksi, ya’ni tartib raqami yoziladi. Bu yerda 0 indeksli birinchi elementning qiymati —44 ga, 1 indeksli ikkinchi elementning qiymati 109 ga, 2 indeksli uchinchi elementning qiymati 23 ga teng bo‘ladi.

10.2. Massivlarni tavsiflash va ulardan foydalanish

Massivlardan dastur tuzishda foydalanish uchun ularni dastlab e’lon qilish va kerak bo‘lsa, massiv elementlarini initsializatsiya qilish, ya’ni tavsiflash kerak bo‘ladi. Massiv e’lon qilinganda, kompilator elementlar soniga teng miqdorda xotiradan joy ajratadi. Masalan, **char** tipidagi o‘zgaruvchili, **m** nomli massiv quyidagicha e’lon qilinadi:

```
char m[4];
```

Bir vaqtning o‘zida bir necha massivni e’lon qilish ham mumkin:

```
int m1[4], m2[99], k, l = 0;
```

Massiv elementlarini dastur davomida initsaliziatsiya qilish, ya’ni tavsiflash mumkin. Massivning boshlang‘ich qiymatlarini e’lon qilish vaqtida, {} qavslar ichida ham bersa bo‘ladi. {} qavslardagi qiymatlar massivning initsializatsiya ro‘yxati deyiladi.

Misollar va izohlar

```
int n[5] = {3, 5, -33, 5, 90};
```

Yuqoridagi misolda massiv birinchi elementining qiymati 3, ikkinchiniki 5, uchinchisiniki —33, to'rtinchisiniki 5, beshinchi elementining qiymati esa 90 ga teng.

```
double array[10] = {0.0, 0.4, 3.55};
```

Bu massiv **double** tipli bo'lib, u 10 ta elementdan iborat deb e'lon qilingan. Lekin {} qavslar ichida faqat boshlang'ich uchta elementining qiymatlari berildi. Bunday holda, qolgan elementlarning qiymati nolga teng deb tushuniladi.

{ } qavslar ichida berilgan qiymatlar soni massiv elementlar sonidan ko'p bo'lib qolsa, sintaksis xatolik vujudga keladi.

Misollar va izohlar

```
char k[3] = {3, 4, 6, -66, 34, 90};
```

Massivni bunday tavsiflash xatolik hisoblanadi, chunki massivni uch elementdan iborat deb e'lon qilib, unga 6 ta boshlang'ich qiymat berilgan.

```
int w[] = {3, 7, 90, 78};
```

w nomli massiv e'lon qilinib, [] qavslar ichida massivning elementlari soni berilmagan. Bu holda necha elementga joy ajratishni kompilator {} qavslar ichidagi boshlang'ich qiymatlar soniga qarab bilib oladi. Demak, vuqoridagi misolda **w** massiv 4 ta elementdan iborat ekan.

```
char string[] = "Malibu";
char *p = "Ikkinchi!?!";
```

Harfli ifodalar C++ da **char** tipi orqali beriladi. Harfli ifodalar qo'shtirnoq ostida beriladi. Masalan, „S“, „*“ kabi. Satrlar esa qo'shtirnoqlarga olinadi.

10.3. Ko'p o'lchovli (indeksli) massivlar

Massivlar bir necha indeksga ega bo'lishi mumkin. C++ kompilatorlari eng kamida 12 ta indeks bilan ishlashi mumkin.

Masalan, matematikadagi *m* x *n* o'lchamli matritsani ikkita indeksli massiv yordamida bersa bo'ladi.

```
int matritsa [4][10];
```

Yuqorida to'rt satrli va, 10 ustunli matritsani e'lon qildik. Bir o'lchovli massivlar kabi ko'p o'lchovli massivlarni initsializatsiya ro'yxati bilan birga e'lon qilish mumkin.

Misol

```
char c[3][4] = {
{   2, 3, 9, 5},      // 1-satr qiymatlari
{-10, 7, 5, 1},       //2-satr qiymatlari
 {  9, 3, 3, -3}      // 3-satr qiymatlari
                      };
```

2	3	9	5
-10	7	5	1
9	3	3	-3

Misol

Ikki o'lchovli **int arr[4][3]** massivini e'lon qilganda, xotirada **arr** nomli massivining elementlari bo'lmish 4 ta elemcntli massiv uchun joy ajratiladi. Har bir massiv esa, o'z navbatida, uchta **int** tipidagi elementdan iborat bo'ladi. Shunday qilib, kompyuter xotirasida har biri uchtadan **int** tipidagi elementdan iborat 4 ta massiv uchun joy ajratiladi.

Arr				
?				
arr[0]	→	arr[0][0]	arr[0][1]	arr[0][2]
arr[1]	→	arr[1][0]	arr[1][1]	arr[1][2]
arr[2]	→	arr[2][0]	arr[2][1]	arr[2][2]
arr[3]	→	arr[3][0]	arr[3][1]	arr[3][2]

Massiv tipidagi elementlarga murojaat qilish **arr[2]** yoki ***(arr+2)** yozuvi orqali amalga oshiriladi. Ikki o'lchovli massiv elementlariga murojaat qilish uchun esa **arr[1][2]** yoki ***(*(arr+1)+2)** va **(*(arr+1))[2]** yozuvlardan foydalaniladi. Shuni ta'kidlash kerakki, C++ tilida **arr[0], arr[1], arr[2], arr[3]** lar o'zgarmas miqdorlar bo'lib, ularning qiymatini dastur davomida o'zgartirish mumkin emas.

MASHQLAR

1. a) Alfa nomli 10 elementdan iborat; b) Betta nomli 5 ta elementdan iborat massiv qanday belgilanadi?
2. A[15], B[23] massivlarining nechtadan elementi bor? Ularning indeksi 7 ga teng elementiga qanday murojaat qilinadi?
3. char k[5] = {3, 4, 6, -66, 34, 90};
   int w[2] = {3, 7, 9, 0, 7, 8};
   massivlarni tavsiflashda qanday xatolikka yo'l qo'yilgan?
4. int w[] = {3, 7, 9, 0, 7, 8};
   char k[] = {3, 4, 6, -66, 34, 90};
   massivlarning nechta elementi bor?
5. double array[15] = {0.0, 0.4, 3.55};
   char k[12] = {3, 4, 6, -66, 34, 90};
   massivlarning oxirgi elementi nimaga teng?
6. a) ATS nomli 10×12 elementdan iborat; b) BTS nomli 5×29 ta elementdan iborat ikki o'lchovli massiv qanday belgilanadi?
7. C[5][4], D[2][3] massivlarning nechtadan elementi bor? Ularning indeksi 2×2 ga teng elementiga qanday murojaat qilinadi?
8. Quyida tasvirlangan elementlarga ega bo'lgan biror nom ostida berilgan massivlarni tavsiflang.

2	3	9	5
-10	7	5	1
9	3	3	-3

2	3	9
-10	7	5
9	3	3

Nazorat savollari

1. Massivga ta'rif bering va misollar keltiring.
2. Massivlarning qanday turlari bor? Ularni tavsiflang.
3. Massivning elementi nima va u qanday tavsiflanadi?
4. Massiv indeksi deb nimaga aytiladi?

11-mavzu	**FUNKSIYALAR**

C++ tilida dasturlashning asosiy bloklaridan biri funksiyalar hisoblanadi. Funksiyalarning ahamiyatli jihati shundaki, katta masala bir necha kichik bo'laklarga bo'linib, har biriga alohida funksiya yozilsa, masala yechish algoritmi biroz soddalashadi. Bunda dasturchi yozgan funksiyalar C++ ning standart kutubxonasi va boshqa kutubxonalar ichidagi funksiyalar bilan birlashtiriladi. Bu esa ishni osonlashtiradi. Ko'p hollarda, dasturda takroran bajariladigan amalni funksiya sifatida yozish va kerakli joyda ushbu funksiyani chaqirish mumkin.

11.1. Funksiyalar haqida tushuncha va ularni yaratish

Funksiya bu — amallar bloki bo'lib, lozim bo'lganda dasturning boshqa bir qismida uni chaqirib foydalanish mumkin.

Funksiyaning formati quyidagicha bo'ladi:

Tip funksiya_nomi (argument1, argument 2, ...) funksiya_tanasi

- ***tip*** — funksiya tomonidan qaytarilayotgan ma'lumot tipi;
- ***funksiya_nomi*** — funksiyaga berilgan nom;
- ***argument*** — ma'lumot tipi va uning identifikatori;
- ***funksiya_tanasi*** — oddiy ifoda yoki ifodalar bloki.

Har bir argument o'zgaruvchi e'loni kabi ma'lumot tipi va uning identifikatoridan tashkil topadi (masalan, **int a**) va funksiya ichida boshqa ozgaruvchilar bilan bir xil rol o'ynaydi. Funksiya chaqirilganda argumentlar parametrlarni o'tkazishga yordam beradi. Funksiya argumentlari bir-biridan vergul bilan ajratib yoziladi.

Funksiya tanasi bu funksiya bajaradigan vazifa, ya'ni amallar joylashgan qismidir. Funksiya tanasi funksiya qavslari {} ichiga yoziladi.

Quyida siz ilk funksiya misoli bilan tanishasiz.

Misollar va izohlar

```
#include <iostream.h>

int yig'indi (int a, int b)
{
  int r;
  r=a+b;
  return (r);
}

int main ()
{
  int z;
  z = yig'indi (5,3);
  cout << "Natija  "
<< z;
  return 0;
}
```

Bu dasturni ishga tushirish uchun avvalo C++ da dastur **main** funksiyasi bilan boshlanishini esda tutish kerak. Bunda main funksiyasi **int** turi bilan **z** o'zgaruvchi e'loni bilan boshlanadi. Kevin esa **yig'indi** funksiyasi chaqiriladi. Agar quyidagiga e'tibor bersak. funksiyani chaqirish strukturasi bilan funksiya e'lonining oxshash tomoni ko'rinadi.

int yig'indi (int a, int b)

z = yig'indi (5 , 3)

Bu yerda parametrlarga nisbatan aniq moslik mavjud. **main** funksiyasida chaqirilgan ikki **5** va **3** qiymatli yig'indi funksiyani avval e'lon qilingan **int a** va **int b** parametrli yig'indi funksiyasiga mos kcladi.

main da funksiya chaqirilgan paytda **main** funksiyasi boshqaruvni yo'qotadi va boshqaruv **yig'indi** ga o'tadi. Ikkala parametrlardagi qiymat (**5** va **3**) funksiya ichidagi lokal o'zgaruvchilarga o'tadi.

Funksiya esa yangi o'zgaruvchini e'lon qiladi **(int r)** va **r = a+b** ifodasi yoziladi. **r** bunda **a** bilan **b** yig'indisiga teng. Chunki **a** va **b** uchun o'tayotgan parametrlar mos ravishda **5** va **3** dir. Natija esa **8** ga teng.

Ushbu satr:

return (r);

yig'indi funksiyasini tugatadi va boshqaruv **main** ga o'tadi. Dastur esa qayerda to'xtagan bo'lsa, o'sha yerdan ishga tushadi. Ammo, bundan tashqari, **return r** o'zgaruvchisi bilan chaqiriladi va bu qiymat hozir **8** dir. Demak, **8** ni biz funksiya qaytargan qiymat deb tushunsak ham bo'laveradi.

int yig'indi (int a, int b)

8

z = yig'indi (5 , 3);

Funksiya tomonidan qaytariladigan qiymat funksiya baholanayotganda berilgan qiymatdir. Shuning uchun yig'indi (5, 3) qaytarayotgan qiymat, ya'ni **8** z ga joylashadi.

Quyidagi **main** satri esa dastur natijasini ekranga chiqaradi.:

```
cout <<  "Natija"<< z;
```

Natija 8

Misollar va izohlar

```
// function example
#include <iostream.h>

int ayirma (int a, int b)
{
  int r;
  r=a-b;
  return (r);
}
int main ()
{
  int x=5, y=3, z;
  z = ayirma (7, 2);
  cout << "Birinchi natija "
<< z << "\n";
  cout << "Ikkinchi natija "
<< ayirma (7,2) << "\n";
 cout << "Uchinchi natija"
<< ayirma (x,y) << "\n";
  z= 4 + ayirma (x,y);
cout <<"To'rtinchi natija "
<< z << "\n";
  return 0;
}
```

Bu misolda biz **ayirma** degan funksiya yaratdik. Bu funksiyaning yagona vazifasi ikki parametrining ayirmasini hisoblab, natijani qaytarish. Agar **main** funksiyasini ko'rib chiqadigan bo'lsak, **ayirma** funksiyasini bir necha marta chaqirilayotganiga guvoh bo'lamiz. Bunda funksiya turli holatlarda bir necha xil yo'llar bilan chaqirilgan.

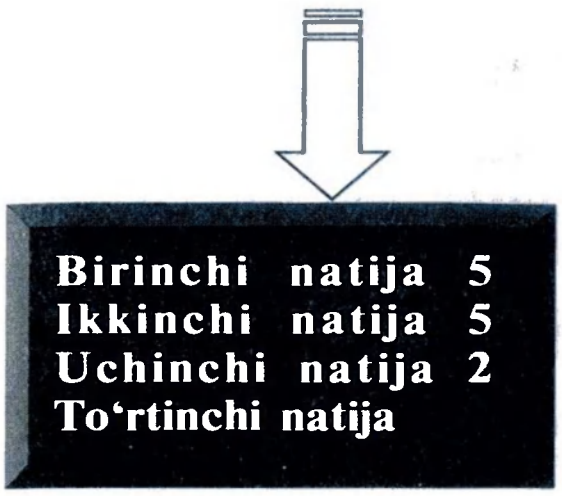
Misol uchun, birinchi holatni ko'rib chiqamiz:

```
z = ayirma ( 7 ,  2)
cout << "Birinchi natija  " << z;
```

Agar funksiya chaqirilishini uning natijasi bilan almashitrsak, u quyidagicha bo'ladi:

```
z = 5;
cout << "Birinchi natija" << z;
cout << "Ikkinci natija" <<
ayirma ( 7,  2 );
```

Bu holatda ham xuddi avvalgi chaqiruv kabi bir xil natijaga ega bo'lamiz. Ammo bu holatda **ayirma** chaqiruvi to'gridan to'gri **cout** parametri sifatida yoziladi. **5 ayirma (7, 2)** ning natijasi bo'lgani uchun quyidagicha yoziladi:



Monitorga yuqoridagi natija chiqadi.

cout << “Ikkinchi natija ” << 5;

Quyidagi holatda esa

cout << “Uchinchi natija ” << ayirma (x, y);

Biz o‘zgartirgan yagona narsa bu ayirma parametrlaridagi konstantalar o‘rniga o‘zgaruvchilarning yozilishidir. Bunday holatda **ayirma** funksiyasining qiymatlari **x** va **y** dir. Ular, mos ravishda, **5** va **3** bo‘lib, natija esa **2** bo‘ladi.

To‘rtinchi holat ham oldingi holat bilan deyarli bir xil bo‘ladi.

E’tibor bering

z = 4 + ayirma (x, y);

o‘rniga

z = ayirma (x, y) + 4;

yozilyapti, natija esa bir xil bo‘lmoqda.

Har bir ifoda oxiriga (;) belgisi yoziladi. Bu belgi albatta funksiya chaqiruvidan keyin qo‘yilishi shart emas.

11.2. Tipsiz funksiyalar. Void funksiyasidan foydalanish

Agar yodingizda bo‘lsa, funksiyani e’lon qilish sintaksisi quyidagicha edi:

```
Tip funksiya_nomi
(argument1, argument2, ...)
funksiya _tanasi
```

Bu yerda funksiya e’loni majburiy ravishda **tip** bilan boshlanyapti, ya`ni **return** bilan qaytariladigan ma’lumot tipi funksiya tomonidan qaytarilmoqda. Lekin qaytariladigan qiymatning o‘zi bo‘lmasa-chi?

Tasavvur qiling, biz faqat biron-bir yozuvnigina ekranga chiqaradigan funksiya tuzmoqchimiz. Biz hech qanday qiymatni qaytarmaymiz. Bundan tashqari, hech qanday parametrni qabul qilmoqchi emasmiz. Bunday holat uchun C++ dasturlash tilida **void** tipi ajratilgan.

Misollar va izohlar

C++ tilida **void** hech qanday qiymat qaytarmasa va hech qanday parametr qabul qulmasa ham, bu tipdagi funksiyaning o‘ziga xos ijobiy tomonlari mavjud.

Biz funksiya chaqiruvi formatida hushyor bolishimiz kerak bo‘lgan joy — bu uning nomi va argumentlarni qavs ichiga yozishdir. Ammo argument bo‘lmagan holatida ham qavslardan foydalanish majburiyati o‘z kuchida qoladi. Masalan, **void funksiya** chaqiruvi:

```
voidfunksiya ();
```

Bu hech qanday o‘zgaruvchisiz funksiya qaytarilishini bildiradi.

```
// void funksiyasiga misol
#include <iostream.h>

void voidfunksiya (void)
{
cout<<"Men funk-
siyaman!";
}

int main ()
{
  voidfunksiya ();
  return 0;
}
```

Men funksiyaman!

MASHQLAR

1. Ikkita son bo‘linmasini topish dasturining funksiyasini yarating.
2. Ikkita son ko‘paytmasini topish dasturining funksiyasini yarating.
3. $ax^2+bx+c=0$, $x\neq0$ kvadrat tenglama ildizlarini topish dasturining funsiyasini yarating.
4. **void** funksiyasidan foydalangan holda e’lon qilingan o‘zgaruvchi qiymati 25 ga teng bo‘lmagunga qadar, ekranga o‘zgaruvchining har bir qiymati bilan birga „Talaba“ yozuvi ham ekranga chiqsin.
5. (a-b)/(c+a/(c=b)) arifmetik ifodani hisoblash dasturining funksiyasini tuzing. Ifodadagi har bir o‘zgaruvchining oxirgi qiymati bilan birga, ekranga qaysi qiymatga tegishli ekanligi haqidagi tushuntirish matni chiqsin. Misolni yechishda **void** finksiyasidan foydalaning.

Nazorat savollari

1. Funksiya nima?
2. Funksiyalar nima uchun tuziladi?
3. Funksiyani tuzish formati qanday?
4. Tipsiz funksiyalar qanday funksiyalar hisoblanadi?
5. **Void** funksiyasiga ta'rif bering.

12-mavzu KO'RSATKICHLAR VA SATRLAR

Biz identifikator orqali aloqa qiladigan xotira yacheykalari bo'lgan o'zgaruvchilarni ko'rib chiqdik. Ammo bu o'zgaruvchilar kompyuter xotirasining aniq bir joyida joylashadi. Bizning dasturlarimiz uchun kompyuter xotirasi esa 1 bayt yacheyka hisobida qabul qilinadi va har bir bayt o'zining adresiga ega bo'ladi.

Kompyuter xotirasiga misol tariqasida shahardagi ko'chalarni keltirish mumkin. Ko'chada uylar aniq bir sonlar bilan raqamlangan bo'ladi. Demak, agar biz Navoiy ko'chasidagi 35-uy haqida gapiradigan bo'lsak, biz bu uyni muammosiz topa olamiz. Sababi bunday raqamdagi uy faqat bitta bo'ladi. Bundan tashqari, bu uy 34- va 36- uylar orasida joylashadi. Ko'chadagi raqamlangan uylar singari, operatsion tizim xotirani aniq raqamlar bilan tashkillashtiradi. Demak, agar biz xotiraning 1776 joyi haqida gapirsak, bilamizki, bunday adresli (manzilli) joy faqat bitta bo'ladi va u 1775 va 1777 adreslar orasida joylashadi.

12.1. Adres operatori (&)

Har bir e'lon qilinadigan o'zgaruvchi xotiradagi aniq bir yacheykaga joylashadi. Biz, odatda, o'zgaruvchining qayerga joylashishini hal qila olmaymiz. Bu ish avtomatik ravishda bir paytning o'zida kompilator va operatsion tizim tomonidan amalga oshiriladi. Operatsion tizim o'zgaruvchiga adres berganidan keyin u joylashgan adresni bilib olishimiz mumkin.

Adres operatori o'zgaruvchi joylashgan joyni aniqlash uchun ishlatiladi va bu & belgisini o'zgaruvchi oldiga qo'yish orqali amalga oshiriladi.

"&ABC" yozuv - *„ABC ning adresi"* deb o'qiladi.
(& belgisi —... ning adresi degan ma'noni anglatadi) Misol uchun:

```
far = &nod
```

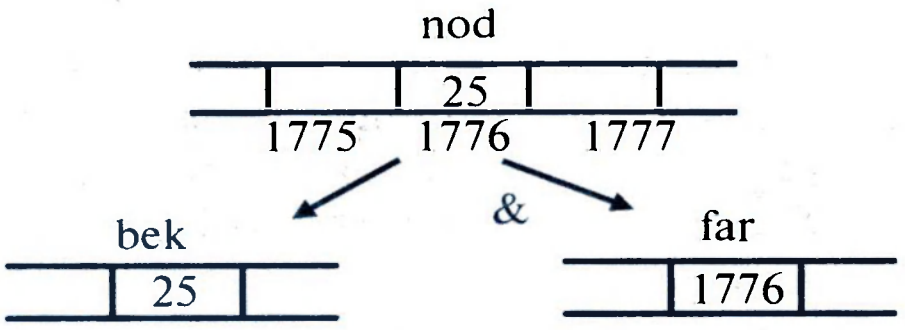
Bunda **nod** ning adresi **far** o'zgaruvchisining qiymati ekanligi bildirilyapti.

Misol

nod o'zgaruvchini **1776** xotira adresida joylashgan deb olib,

```
nod = 25;
bek = nod;
far = &nod;
```

larni yozsak, natijada ushbu diagrammadagi holat yuzaga keladi:



Bunda **bek** ning qiymatini **nod** da bor bo'lgan qiymat deb oldik. Ammo **far** ning qiymati esa operatsion tizim belgilab bergan (biz tasavvur qilgan **1776) nod** ning adresidir. Chunki biz **nod** dan avval **&** belgisini qo'ygan edik.

O'zgaruvchining boshqa o'zgaruvchi adresini qabul qilishi ko'rsatkichlar deyiladi. C++ da ko'rsatkichlarning ko'p afzallik tomonlari mavjud va shu sabab ular tez-tez ishlatiladi. Keyinchalik, o'zgaruvchining bunday tipini qanday e'lon qilishni ko'rib chiqamiz.

12.2. Jo'natish operatori (*)

Ko'rsatkichlardan foydalanib, biz ko'rsatkich identifikator oldiga jo'natish operatori * belgisini qo'ygan holda ko'rsatilgan o'zgaruvchi qiymatini olishimiz mumkin.

Ko'rsatkich sifatida e'lon qilinayotgan har bir o'zgaruvchi ismi oldida „*" belgisi qo'yiladi.
Bu belgi *„ ...ko'rsatgan qiymat"* degan ma'noni anglatadi.
***A yozuv *„ A ko'rsatgan qiymat"* deb o'qiladi.**

Misol

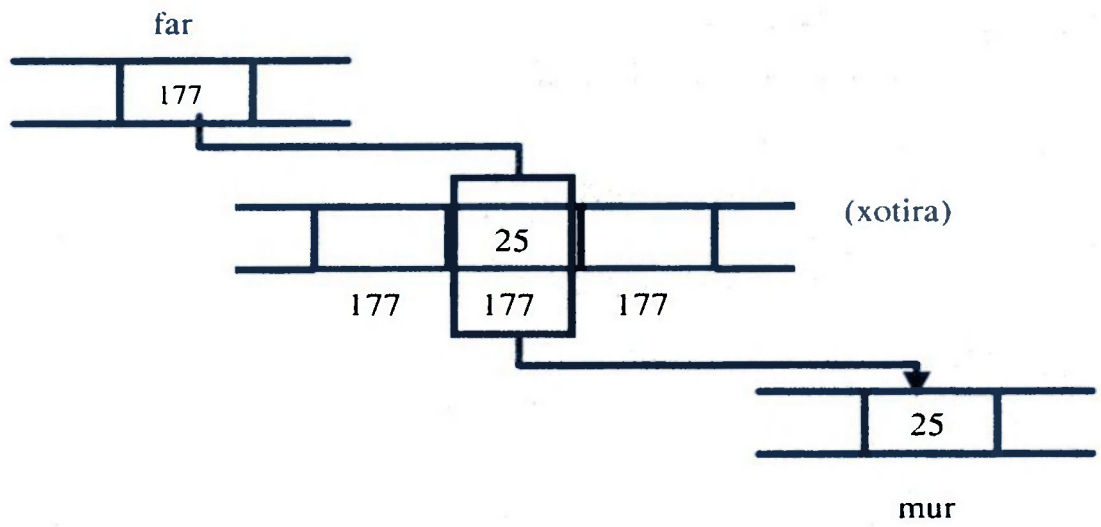
Avvalgi misoldagi qiymatdan kelib chiqib,

mur = *far

ni yozsak, bu yozuv „**mur far** ko‘rsatgan qiymatga teng“ deb o‘qiladi. **far** ning qiymati **25** va **1776** ko‘rsatgan qiymat **25** bo‘lganligi uchun **mur** ning qiymati **25** ga teng.

Shuni farqlab olish kerakki, **far 1776** ga teng, ammo ***far** esa **1776** adresda joylashgan **25** qiymatni ko‘rsatadi. (*) belgisi qo‘yilgan va qo‘yilmagan holatlar farqiga e’tibor bering.

mur = far; // **mur** teng ***far (1776)ga***
mur = *far; // **mur** *teng* ***far*** ***ko‘rsatgan qiymat (25) ga***



„*“ va „&“ operatorlari qoidalariga asosan berilgan

nod = 25;
far = &nod;

misolda quyidagi ifodalarning to‘g‘ri ekanligini ko‘rsatish mumkin:

nod == 25
&nod == 1776
far == 1776
***far == 25**

Birinchi tenglikdan **nod = 25** ekanligi aniq ko‘rinib turibdi. Ikkinchi satrda biz **1776** deb tasavvur qilgan **nod** adresini qaytaruvchi & operatori ishlatiladi. Ikkinchi qator to‘g‘ri bo‘lganligi va **far** ifodasi **far = &nod** bo‘lganligi uchun uchinchi satr ham ravshan. To‘rtinchi ifoda (*) operatoridan foydalanyapti va **far** ko‘rsatgan adres qiymati **25** ekani ko‘rsatilmoqda.

12.3. Ko‘rsatkich turidagi o‘zgaruvchilarni e’lon qilish

Shuni e’tiborga olish kerakki, biz ko‘rsatkichni e’lon qilayotganda ishlatayotgan (*) belgisi „bu ko‘rsatkich“ ma’nosini anglatadi va uni avvalroq ishlatgan jo‘natish operatori (*) bilan almashtirib yubormaslik kerak. Bu ikkita bir xil belgi masalada turli vazifalarni bajaradi.

```
// mening birinchi ko‘rsatkichim
#include <iostream.h>

int main ()
{
int qiymat1 = 5, qiymat 2 = 15;
int * mening ko‘rsatkichim;
mening ko‘rsatkichim =& qiymat 1;
  *mening ko‘rsatkichim = 10;
  mening ko‘rsatkichim =& qiymat 2;
  *mening ko‘rsatkichim = 20;
  cout << "qiymat 1==" << qiymat 1
<< "/ qiymat 2==" << qiymat 2;
  return 0;
}
```

```
qiymat 1==10 /
qiymat 2==20 /
```

qiymat1 va **qiymat2** qiymatlari o‘zgarib ketganiga e’tibor bering. Avval bitta **&** belgisidan foydalangan holda **qiymat1** ning adresini **mening ko‘rsatkichim** ga belgilab berdik. So‘ng 10 ni **mening ko‘rsatkichim** ko‘rsatgan qiymatga belgiladik.

Ko‘rsatkichning bir dasturning o‘zida bir necha qiymatni qabul qila olishini bilib olish uchun xuddi shu jarayonni o‘sha ko‘rsatkich va **qiymat2** bilan qaytarib qo‘ya qoldik.

Misol

Bitta murakkabroq misolni qaraymiz:

```
qiymat1==10 / qiymat2==20
```

```
// Ko‘rsatkichlar
#include <iostream.h>
int main ()
{
  int qiymat1 = 5, qiymat2 = 15;
  int *p1, *p2;
  p1 = &qiymat1;   // p1 = qiymat1 ning adresi
  p2 = &qiymat2;   // p2 = qiymat2 ning adresi
  *p1 = 10;        // p1 = 10 ko‘rsatgan qiymat
  *p2 = *p1;       // p2 ko‘rsatgan qiymat = p1 ko‘rsatgan qiymat
  p1 = p2;         // p1 = p2 (ko‘rsatkich qiymati ko‘chrildi )
  *p1 = 20;        // p1 = 20 ko‘rsatgan qiymat
  cout << "qiymat1==" << qiymat1 << "/ qiymat2==" << qiymat2;
  return 0;
}
```

12.4. Ko‘rsatkichlar va massivlar

Massiv tushunchasi ko‘rsatkich tushunchasiga juda ham bog‘liqdir. Massiv identifikatori uning birinchi elementi adresiga ekvivalentdir. Ko‘rsatkich ham birinchi element adresiga ekvivalent hisoblanadi. Demak, ular deyarli bir xildir. Masalan, mana bu ikki e’lonni olsak, ular bir xil natijani beradi.

```
int  raqamlar [20];
int  *  p;
```

Quyidagucha ham bo‘lishi mumkin:

```
p = raqamlar;
```

Bunda **p** bilan **raqamlar** ekvivalent va ular bir xil tarkibga ega. Ular orasidagi yagona farq, biz **p** ko‘rsatkichiga boshqa qiymatni belgilashimiz mumkin, raqamlar esa **int** turidagi 20 ta butun **(integer)** sonlarning birinchisiga ko‘rsatadi. Demak, oddiy qilib aytganda, **p** bu o‘zgaruvchi ko‘rsatkich, **raqamlar** esa o‘zgarmas **(konstanta)** ko‘rsatkich (haqiqatan ham, massiv nomi konstanta ko‘rsatkich ma’nosini beradi). Shuning uchun avvalgi holat vujudga kelishi mumkin.

Quyidagi yozuv esa xatolik hisoblanadi:

raqamlar = p;

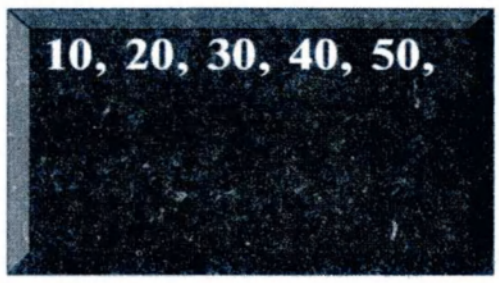
chunki raqamlar bu massiv (konstanta ko‘rsatkich) va konstanta identifikatorlarga hech qanday qiymat belgilamaydi.

Misol

O‘zgaruvchilar xususiyatlaridan kelib chiqib, quyidagi misoldagi ko‘rsatkichlarning ahamiyatini ko‘rish mumkin:

```
// ko‘rsatkichlar
#include <iostream.h>

int main ()
{
  int raqamlar[5];
  int * p;
  p = raqamlar;  *p = 10;
  p++;  *p = 20;
  p = & raqamlar[2];  *p = 30;
  p = raqamlar + 3;  *p = 40;
  p = raqamlar;  *(p+4) = 50;
  for (int n=0; n<5; n++)
  cout << raqamlar[n] << ", ";
  return 0;
}
```



MASHQLAR

1. 5 ta ixtiyoriy o‘zgaruvchi e’lon qiling va ularning adreslarini konsolga chiqaring.
2. 2 ta int tipidagi o‘zgaruvchi e’lon qiling. 1-o‘zgaruvchi qiymati ko‘rsatgan adres qiymati 2-o‘zgaruvchiga teng bo‘lsin va ekranga 2 qo‘shilgan holda, 10 marta ekranga chiqsin.
3. 1 ta **int** tipidagi o‘zgaruvchi va 1 ta **int** tipidagi ko‘rsatkich o‘zgaruvchini e’lon qiling. Ko‘rsatkich o‘zgaruvchi yordamida birinchi o‘zgaruvchining qiymatini ozgartiring va uni ekranga chiqaring.
4. 1 ta **int** tipidagi o‘zgaruvchi va 1 ta **int** tipidagi ko‘rsatkich o‘zgaruvchini e’lon qiling. Ko‘rsatkich o‘zgaruvchi yordamida birinchi o‘zgaruvchining qiymatini tasodifiy songa o‘zgartiring. Ushbu tasodifiy son 1000 dan ortmagunga qadar, o‘sha sonlarning har biri ekranga chiqsin.

Nazorat savollari

1. Ko‘rsatkich nima?
2. Adres operatoriga ta’rif bering va unga doir misol keltiring.
3. Korsatkichning xotirada ishlash diagrammasini tushuntirib bering.
4. Jo‘natish operatoriga ta’rif bering va unga doir misol keltiring.
5. Jo‘natish operatoridan foydalanganda, xotiradagi jarayon diagrammasini tushuntirib bering.
6. Ko‘rsatkich turidagi o‘zgaruvchilarni e’lon qilish qanday amalga oshiriladi?

13-mavzu STRUKTURALAR (TUZILMALAR)

Strukturalar murakkab obyekt bo‘lib, ularga turli xil tipli elementlar kirishi mumkin. Ma’lumki, massivlar bir xil turli elementlardan iborat edi. Strukturalar esa massivlardan farqli o‘laroq, turli elementlardan tashkil topadi.

13.1. Ma’lumot strukturalari

Ma’lumot strukturasi bu yagona e’lon qilingan turli uzunlikdagi ma’lumot tiplari guruhidir. Struktura

struct { ta’riflar ro‘yxati}

yozuvi bilan aniqlanadi.

Strukturada hech bo‘lmaganda bitta element ko‘rsatilgan bo‘lishi kerak. Strukturaning umumiy tuzilishi quyidagicha.

```
Struct model_nomi
    {
      tur1 element1;
      tur2 element2;
      tur3 element3;
        .
        .
        .
    } obyekt_nomi
```

Misollar

model_nomi o‘rniga struktura tipi modeli nomi yoziladi;

obyekt_nomi srtuktura uchun mos identifikator yoziladi;

{} qavslari ichida strukturaga tegishli bo‘lgan tiplar hamda ichki identifikatorlar yoziladi.

Agar struktura ***model_nomi*** parametrini o‘z ichiga olsa, bu parametr strukturaga ekvivalent mos tip nomi hisoblanadi.

```
struct
{
  double x,y;
} s1, s2, sm[9];
```

Bu misolda **s1, s2** o‘zgaruvchilar struktura sifatida tavsiflangan. Ularning har biri ikkita **x** va **y** komponentlardan iborat. **sm** o‘zgaruvchi esa 9 ta elementdan iborat massiv ko‘rinishda aniqlanmoqda.

```
struct
{
   int  yil;
   char  oy, kun;
 } 1sana, 2sana;
```

Bu misolda **1sana** va **2sana** strukturalari uchta — **yil, oy** va **kun** komponentlardan iborat.

```
struct mahsulotlar
       {
         char nom [30];
         float narx;
};
mahsulotlar  olma;
mahsulotlar tarvuz, apelsin;
```

Bu misolda **mahsulotlar** struktura modeli berilgan. Bu struktura ikki **nom** va **narx** komponentlarni o‘z ichiga oladi. Bu o‘zgaruvchilar turlicha. Dastur so‘ngida, **mahsulotlar** struktura turi bilan uch obyektni — **olma, tarvuz** va **apelsin** larni e’lon qildik.

Endi mahsulotlar **int**, **char** yoki **short** ga oxshash mos yangi tipga aylandi va biz bu tip bilan o‘zgaruvchilarni e’lon qila olamiz.

```
struct mahsulotlar
       {
         char nom [30];
         float narx;
};
olma tarvuz, apelsin;
```

Struktura e’loni oxirida kelayotgan obyekt_nomi maydoni struktura turidagi obyektni e’lon qilish uchun xizmat qiladi. Masalan, bu yo‘l bilan biz **olma, tarvuz** va **apelsin** obyektlarini e’lon qilishimiz mumkin.

```
olma.nom
olma.narx
apelsin.nom
apelsin.narx
tarvuz.nom
tarvuz.narx
```

Struktura modeli bilan struktura obyekti orasidagi farqni bilish juda muhimdir. Model bu tip, obyekt esa e’lon qilinayotgan o‘zgaruvchidir. Biz bir qancha obyektlarni (o‘zgaruvchilarni) yagona model orqali e’lon qilishimiz mumkin.

Struktura modelida biz uch obyektni **(olma, tarvuz, apelsin)** e’lon qilganimizdan keyin, endi ularni maydonlar bilan shakllantirishimiz mumkin. Buni amalga oshirish uchun obyekt nomi bilan

maydon nomi o‘rtasiga nuqta (.) belgisini qo‘yish kerak bo‘ladi. Masalan, biz bu elementlardan ularga mos tipdagi standart o‘zgaruvchilardek foydalanishimiz mumkin.

Yuqoridagilarning har biri o‘zining mos ma’lumot turi: **olma.nom, apelsin.nom** va **tarvuz.nom** lar **char** tipiga, olma. narx, **apelsin.narx** va **tarvuz.narx** lar esa **float** tipiga tegishlidir.

Endi kino haqidagi misolni ko‘rib chiqamiz.

Misol

```
// strukturaga doir misollar
#include <iostream.h>
#include <string.h>
#include <stdlib.h>

struct film_t {
  char nom [50];
  int yil;
} mening, sening;

void printmovie (film_t film);

 int main ()
 {
  char buffer [50];

  strcpy (mening.nom, "O‘rgimchak
odam");
  mening.yil = 2004;

  cout << "Film nomini kiriting: ";
  cin.getline (sening.nom,50);
  cout << "Film yilini kiriting: ";
  cin.getline (buffer,50);
  sening.yil = atoi (buffer);

  cout << "Mening sevimli filmim:\n
";
  printmovie (mening);
  cout << "Sening filming esa:\n ";
  printmovie (sening);
  return 0;
}
void printmovie (film_t film)
{
  cout << film.nom;
  cout << " (" << film.yil << ")\n";}
```

Ekranda

```
Film nomini kiriting:
O‘rgimchak odam
Film yilini kiriting:
2004
Mening sevimli
filmim:
O‘rgimchak odam
(2004)
Sening filming esa:
Alien (1979)
```

Misoldan ko‘rinib turibdiki, biz struktura elementlarini hamda strukturaning o‘zini o‘zgaruvchilar sifatida ishlatishimiz mumkin. Masalan, **sening. yil** bu **int** turiga mos o‘zgaruvchi, **mening.nom** esa 50 belgili mos massivdir.

```
// strukturalar masssivi
#include <iostream.h>
#include <stdlib.h>

#define N_KINOLAR 5

struct film_t {
  char nom [50];
  int yil;
} filmlar [N_KINOLAR];

void printmovie (film_t film);

int main ()
{
  char buffer [50];
  int n;
  for (n=0; n<N_KINOLAR; n++)
  {
    cout << "Film nomini kriting: ";
    cin.getline (filmlar[n].nom,50);
    cout << "Film yilini kiriting: ";
    cin.getline (buffer,50);
    filmlar[n].yil = atoi (buffer);
  }
  cout << "\n Siz quyidagi filmlarni
kiritdingiz:\n";
  for (n=0; n<N_KINOLAR; n++)
    printmovie (filmlar[n]);
  return 0;
}

void printmovie (film_t film)
{
  cout << film.nom;
  cout << " (" << film.yil << ")\n";
}
```

Ekranda

```
Film nomini kiriting:
Alien
Film yilini kiriting: 1979
Film nomini kiriting:
Blade Runner
Film yilini kiriting: 1982
Film nomini kiriting:
Matrix
Film yilini kiriting: 1999
Film nomini kiriting:
Rear Window
Film yilini kiriting: 1954
Film nomini kiriting:
Taxi Driver
Film yilini kiriting: 1975
Siz quyidagi filmlarni
kiritdingiz:
Alien (1979)
Blade Runner (1982)
Matrix (1999)
Rear Window (1954)
Taxi Driver (1975)
```

Mening va **sening** lar ham **printmovie()** funksiyasiga o‘tayotganida, **film_t** turiga mos o‘zgaruvchilar bo‘lib ko‘rilmoqda. Shuning uchun, strukturaning muhim ijobiy tomonlaridan biri bu, individual ravishda uning elementlaridan yoki to‘liq strukturadan blok sifatida foydalanish mumkinligidir. Strukturalar ma’lumotlar

bazasini tuzishda juda qulay bo‘lib, ayniqsa, uni massivlar bilan shakllantirilsa, qulay dastur yaraladi.

13.2 Strukturalarga ko‘rsatkichlar

Boshqa turlar kabi strukturalar ham ko‘rsatkichlar bilan tavsiflanadi. Qoida ham boshqa fundamental ma’lumot turlardagiga o‘xshash, ko‘rsatkich strukturaga ko‘rsatkich bo‘lib e’lon qilinishi kerak.

Bunda **afilm filmlar_t struct** tipining obyekti **pfilm** esa **filmlar_t struct** tipi ob’yektiga ko‘rsatkichdir. Demak, quyidagi fundamental tipli satr ham to‘g‘ridir:

pfilm = & afilm;

```
//Strukturalarga ko‘rsatkichlar
#include <iostream.h>
#include <stdlib.h>

struct filmlar_t{
  char nom [50];
  int yil;
};

int main ()
{
  char buffer[50];

  filmlar_t afilm;
  filmlar_t * pfilm;
  pfilm = & afilm;
  cout << "Film nomini kiriting: ";
  cin.getline (pfilm->nom,50);
  cout << "Film yilini kiriting: ";
  cin.getline (buffer,50);
  pfilm->yil = atoi (buffer);
  cout << "\nSiz quyidagini kiritdingiz:\n";
  cout << pfilm->nom;
  cout << " (" << pfilm->yil << ")\n";

  return 0;
}
```

```
struct filmlar_t {
  char nom [50];
  int yil;
};

filmlar_t afilm;
filmlar_t * pfilm;
```

Bu yerda **afilm filmlar_t** struct tipining obyekti, pfilm esa **filmlar_t struct** tipi obyektiga ko‘rsatkichdir.

```
Film nomini kiriting: Matrix
Film yilini kiriting: 1999

Siz quyidagini kiritdingiz:
Matrix (1999)
```

Endi esa yangi operatorni tanishtiruvchi boshqa bir misolni ko‘rib chiqamiz. Bu misol juda muhim operator -> ni o‘z ichiga oladi. Bu jo‘natish operatori faqat strukturaga ko‘rsatkichlar va klasslarga ko‘rsatkichlar holatida ishlatiladi. Operator bizni struktura a’zosini qavs ichiga yozish majburiyatidan xalos qiladi. Masalan,

pfilm->nom

quyidagich yoziishi mumkin edi:

(*pfilm).nom

Bu ikki ifoda **pfilm->nom** va **(*pmovie).nom** imlo jihatdan to‘g‘ri hamda **pfilm** ko‘rsatayotgan strukturaning nom elementini baholayotganimizni bildiradi. Siz uni quyidagidan aniq farqlab olishingiz mumkin:

(*pfilm).nom

Bu, o‘z navbatida, quyidagiga ekvivalent:

***pfilm.nom**

bo‘lgan bu ifoda filmlar strukturasining nom elementi ko‘rsatgan qiymatni baholaydi.

MASHQLAR

1. 3 ta o‘zgaruvchi e’lon qilingan struktura e’lon qiling. Ushbu struktura o‘zgaruvchilariga tasodifiy qiymatlar bering. Qiymatlarni void funksiyasidan foydalangan holda ekranga chiqaring.
2. Struktura e’lon qiling va unda **int** va **float** tipli o‘zgaruvchilar e’lon qiling. Endi struktura tipidagi o‘zgaruvchilarga qiymatlar bering. **int** tipidagi ozgaruvchi qiymati 10 marta takrorlanib ekranga chiqsin va har bir chiqishida 1 ga kamayib borsin. Ushbu kamaygan qiymat **float** tipidagi o‘zgaruvchiga teng bo‘lib, u ham ekranga 10 marotaba chiqsin.

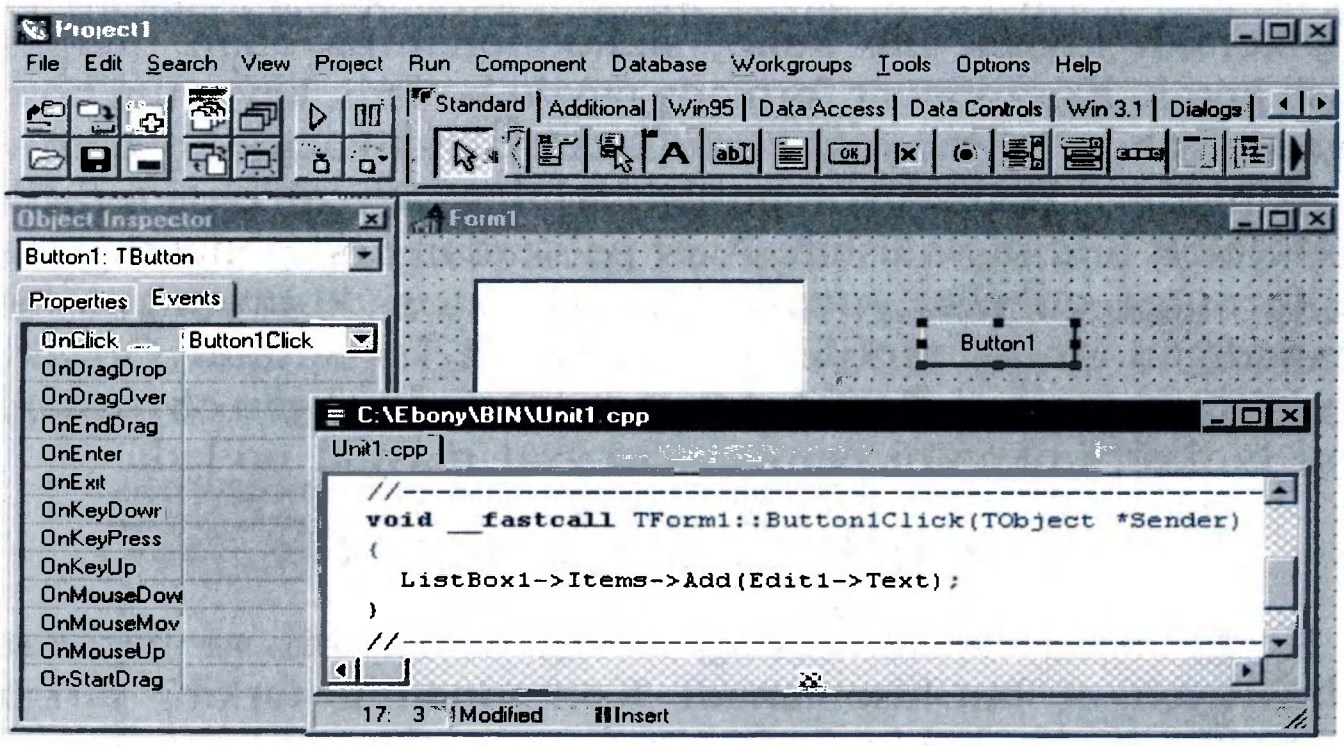
Nazorat savollari

1. Struktura nima?
2. Strukturalar nima uchun kerak?
3. Strukturalar qanday tuziladi?
4. Struktura modeli nima?
5. Struktura obyekti nima?
6. Strukturalarga ko‘rsatkichlar qanday ko‘rsatiladi?

14-mavzu	**VIZUAL DASTURLAR TUZISH**

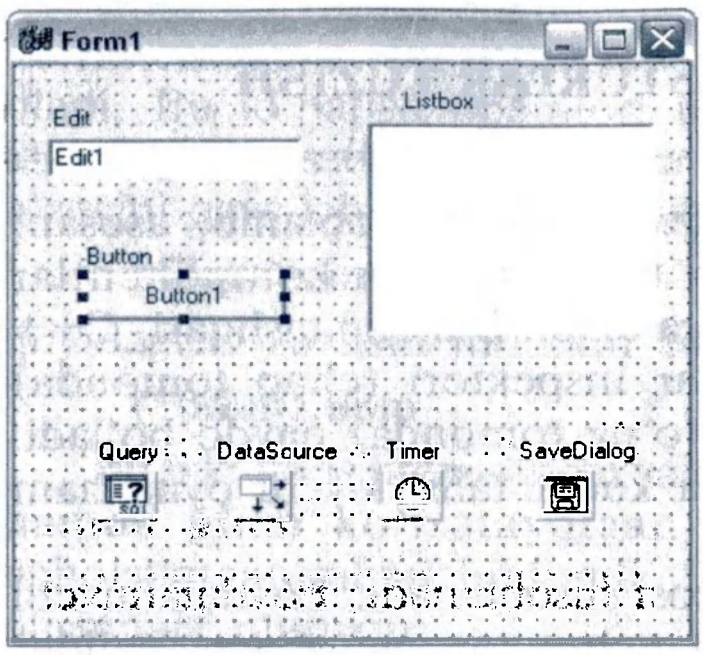
C++ Builder dasturi **SDI** ilova dasturi hisoblanib, asosan, sozlovchi asboblar paneli (chap tomonda) va komponentlar palitrasini (o‘ng tomonda) o‘z ichiga oladi. Bundan tashqari, C++ Builder yuklanayotganda obyektlar inspektori (chap tomonda) hamda yangi ilova dasturi shakli (o‘ng tomonda) paydo bo‘ladi. Yangi ilova dasturi shakli orqasida kodni tahrirlovchi oyna ham mavjud bo‘ladi.

Shakllar C++ Builder ning asosi hisoblanadi. Foydalanuvchi ilova dasturi interfeysini yaratish uchun oynaga C++ Builder obyekti formasi elementlarini, ya’ni komponentlarni qo‘yish bilan amalga oshiriladi. C++ Builder komponentlari o‘ng tomondagi komponentlar palitrasida joylashgan bo‘ladi. C++ Builderning eng yaxshi tomonlaridan biri shuki, unda dasturchi o‘zi xohlagan yangi komponent yaratishi va uni komponentlar palitrasiga sozlab olishi mumkin. Bundan tashqari, turli loyihalar uchun yangi komponentlar yaratish mumkin.



14.1. C++ Bider komponentlari

Komponentlar ko‘rinadigan (vizual) va ko‘rinmaydigan komponentlarga bo‘linadi. Vizual komponentlar loyihalashtirayotgan paytda qanday holatda ko‘rinsa, dastur ishga tushganda ham



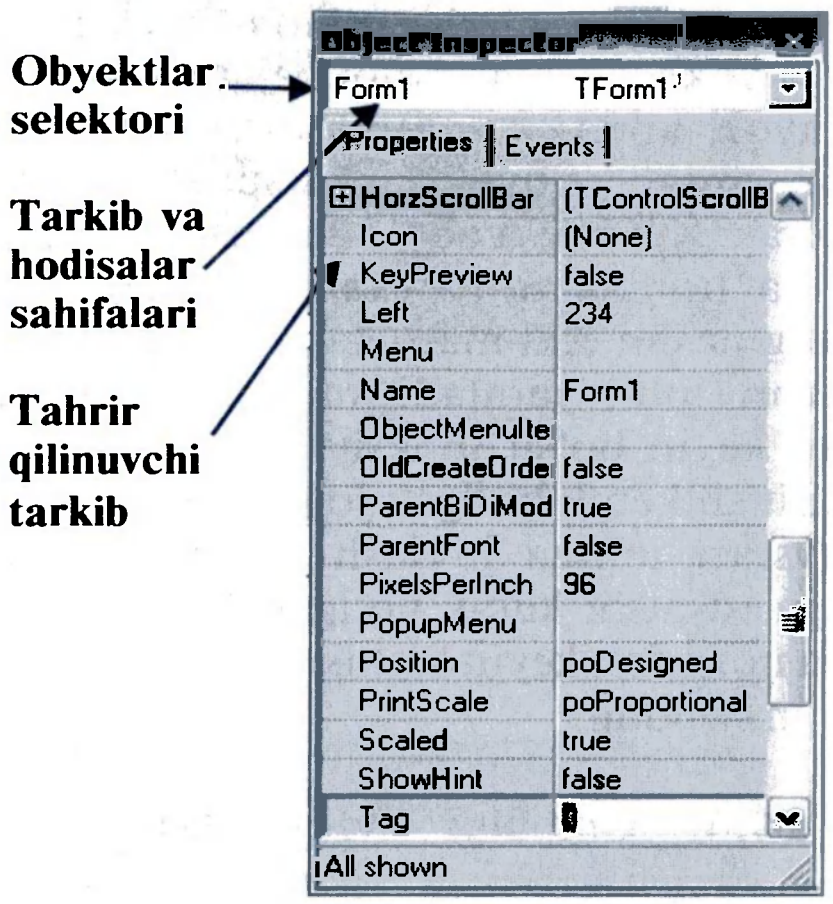
Ko‘rinadigan va ko‘rinmaydigan komponentlardan foydalanish bo‘yicha misol.

shunday ko‘rinadi. Ko‘rinmaydigan komponentlar loyihalashtirish paytida formadagi piktogramma ko‘rinishida bo‘ladi. Ular dastur ishga tushgan paytda umuman ko‘rinmaydi, ammo aniq bir vazifaga egadir (masalan, ma’lumotlar bilan aloqa qilish imkonini beradi yoki Windows standart dialoglarini chaqiradi).

Formaga component qo‘shish uchun sichqoncha yordamida palitradagi kerakli komponentni tanlab va loyihalanayotgan formaning kerakli joyiga olib kelib, sichqonchaning chap tugmasini bosish kerak. Komponent formada paydo bo‘ladi, keyin uni boshqa joyga siljitish, o‘lchovi yoki boshqa xarakteristikalarini o‘zgartirish mumkin. C++ Builder komponentlarining uch xil xarakteristikasi bor: tarkibi, hodisa va uslub. Agar palitradan komponentni tanlab uni formaga qo‘yilsa, obyektlar inspektori komponent bilan ishlanadigan uning tarkibi va hodisasini ko‘rsatadi. Obyektlar inspektorining yuqori qismida formada mavjud bo‘lgan obyektlardan birini tanlash imkonini beruvchi ro‘yxat mavjud bo‘ladi.

14.2. Komponentlar tarkibi

Komponent tarkibi uning tashqi ko‘rinishi va harakatini boshqaruvchi atribut hisoblanadi. Komponentlar tarkibi tarkiblar **(Properties)** sahifasida ko‘rinib turadi. Obyektlar inspektori nashrlangan **(published)** komponent tarkibini ko‘rsatib turadi. Shunday ekan, komponentlar faqat ilova dastur ishga tushganda kirish mumkin bo‘ladigan umumiy **(public)** tarkibga ega bo‘ladi. Obyektlar inspektori dastur loyihanayotgan paytda komponent tarkibini sozlash uchun foydalaniladi. Komponent tarkibini

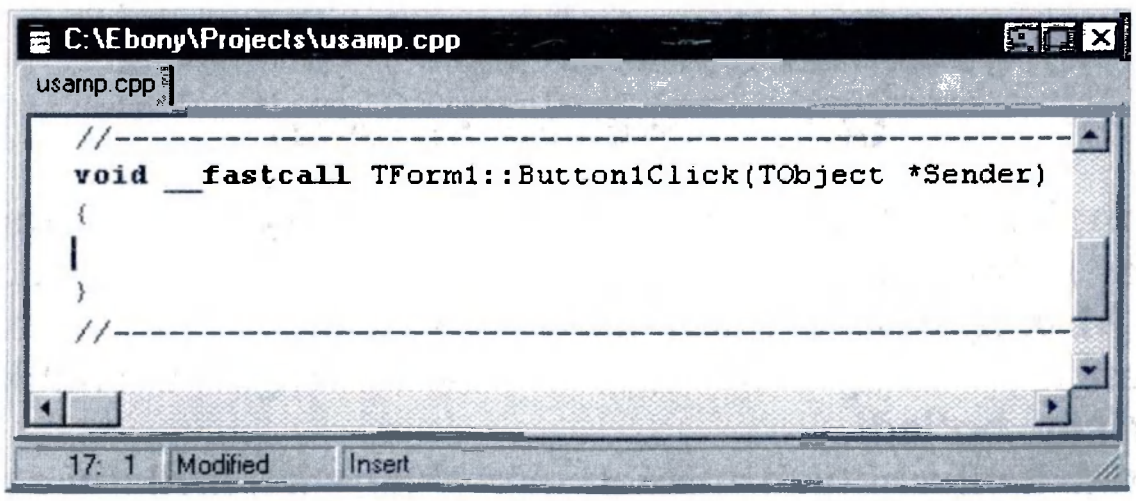


Obyektlar inspektori

loyialash davomida aniqlash, o‘zgartirish mumkin yoki dastur ishga tushayotgan paytda tarkibni visual o‘zgartirish uchun kod yozish mumkin.

14.3. Hodisalar

Obyektlar inspektoridagi hodisalar **(Events)** sahifasi komponentga tegishli bo‘lgan barcha hodisalar, ya’ni ushbu komponent bilan sodir bo‘ladigan barcha jarayonlar ro‘yxatini taqdim qiladi. Har bir komponent o‘zining komponent ustida ishlaydigan hodisalar ro‘yxatiga ega. C++ Builder da funksiyalar yoziladi va



Hodisalar jarayoning prototipi.

shu funksiyalar hodisalar bilan bo‘g‘lanadi. Hodisaga jarayon yaratib, siz dasturga yozilgan funksiyani bajarishga buyruq berasiz.

Hodisaga jarayon kiritish uchun formadagi komponentni sichqoncha bilan tanlab, obyektlar inspektoridagi hodisalar sahifasini ochish kerak. Keyin esa obyektlar inspektorining hodisalar sahifasini ochib va hodisalar ro‘yxatidan kerakligini tanlab, sichqoncha chap tugmasini ikki marta bosish kerak. Shunda, C++ Builder hodisa prototipini yaratadi va kod tahrirlagichida uni ko‘rsatadi. Bundan keyin bo‘sh funksiya matni paydo bo‘ladi va tahrirlagich kod kiritish kerak bo‘lgan joyda paydo bo‘ladi. Kursor {...} operator qavslari ichiga joylashadi. Keyin hodisa sodir bo‘layotganda ishga tushishi kerak bo‘lgan kod yoziladi. Hodisa jarayonida funksiya nomidan keyin kichik qavs ichiga ( ) yoziladigan parametrlar ham mavjuddir.

14.4. Uslublar (Methods)

Uslub komponentga bog‘liq bo‘lgan va obyektning bir qismi bo‘lib e’lon qilinadigan funksiya hisoblanadi. Hodisa jarayonini yaratish paytida -> belgidan foydalangan holda uslubni chaqirish mumkin.

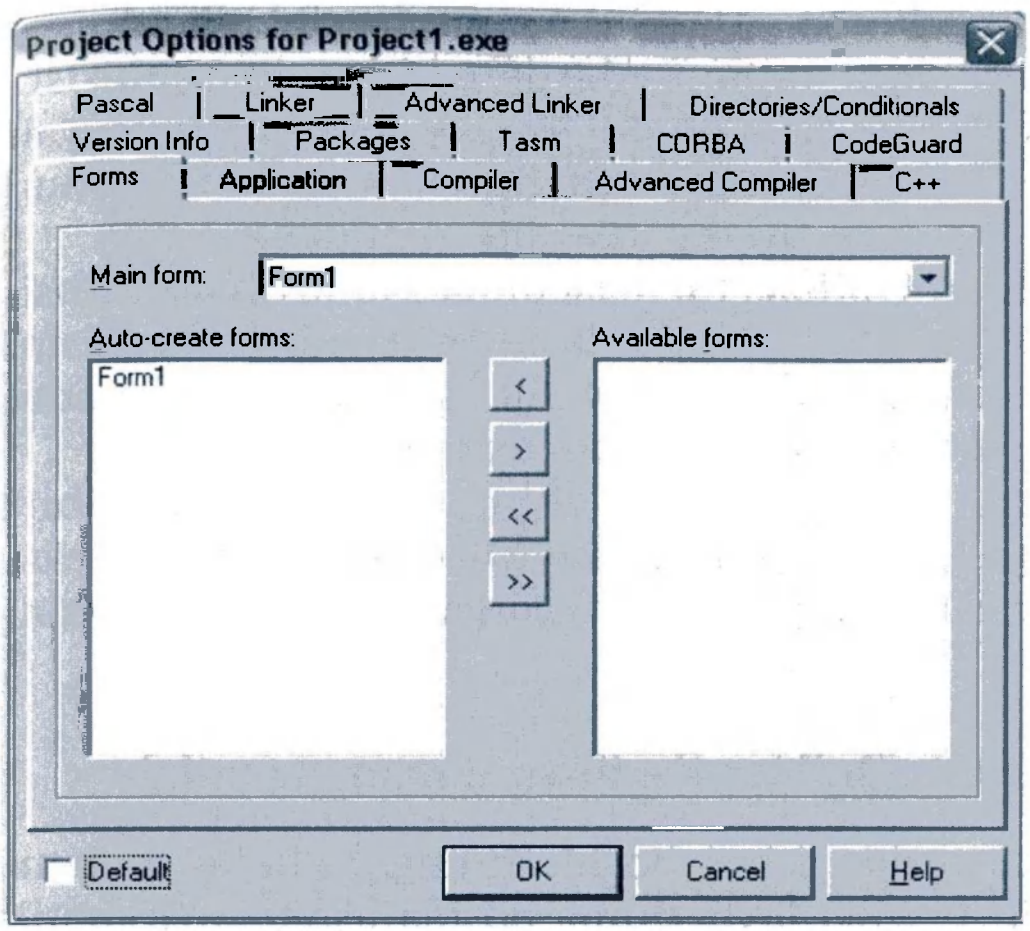
```
Edit1->Show();
```

Shuni e’tiborga olish kerakki, yangi forma yaratilayotganda unga bog‘liq bo‘lgan modul va ***.h** kengaytmali sarlavha fayli albatta yaraladi. Forma va modul nomlarini o‘zgartirish mumkin. Lekin bu ishni forma yangi yaratilishi bilanoq amalga oshirish maqsadga muvofiq. Chunki boshqa formalar va modullar bilan adashtirib yuborishingiz mumkin.

14.5. Loyihalar menejeri

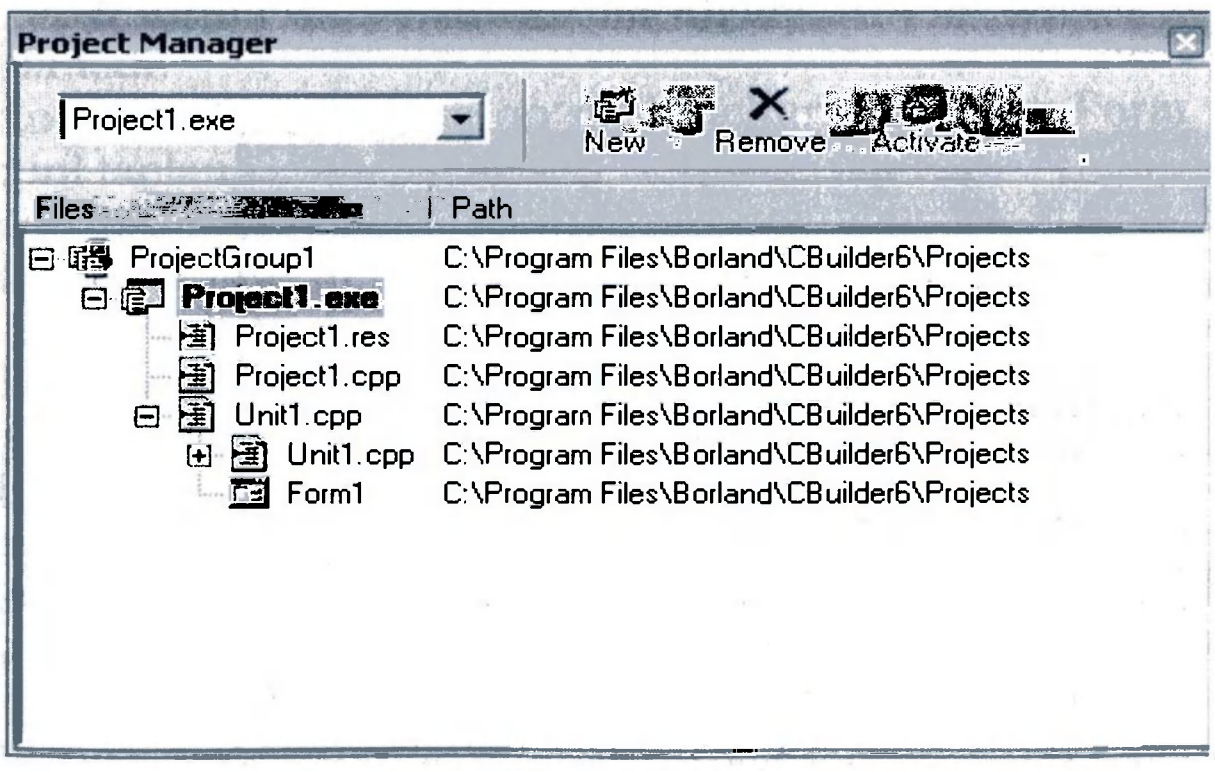
Loyihalar menejeri ilova dasturining fayllari va modullar ro‘yxatini ko‘rsatadi va ular orasida navigatsiya o‘rnatadi. **View/ Project Manager** yo‘li bilan loyiha menejerini ochish mumkin. Yaratilgan loyihaning avtomatik ravishda berilgan dastlabki nomi **Project1.cpp** bo‘ladi.

Dastlab loyiha bitta forma uchun boshlang‘ich fayllarni o‘z ichiga oladi. Ammo ko‘p loyihalarda bir necha forma va modullar mavjud



Loyiha opsiyasini sozlash..

bo‘ladi. Loyihaga modul yoki forma qo‘shish uchun sichqoncha o‘ng tugmasini bosib, kontekst menyudan **New Form** punktini tanlash kerak. Loyihaga kompyuterda mavjud forma yoki modulni qo‘shish ham mumkin. Buning uchun loyiha menejerining kontekst menyusidan **Add** ni bosib, qo‘shish kerak bo‘lgan modul yoki formani tanlash kerak. Loyiha ishlayotganda ixtiyoriy paytda forma



Loyihalar menejeri

yoki modulni o‘chirib yuborish mimkin. Ammo modul bilan forma bir-biriga bog‘liq bo‘lgani uchun ulardan birini ikkinchisisiz o‘chirmaslik kerak. Lekin bu ishni modul formaga bog‘liq bo‘lmagan paytda amalga oshirish mumkin. Loyihadan modulni o‘chirish uchun loyiha menejeridagi **Remove** tugmasidan foydalaniladi.

Loyiha menejeridan **Options** tugmasi tanlansa, loyiha opsiyasi, ya’ni muloqot paneli ochiladi. Unda dasturning asosiy formasini tanlash, dinamik ravishda qanday formalar yaratilishi va modul kompilatsiyasi parametrlari qanaqaligini aniqlash mumkin.

C++ Builder ish maydonida muhim elementlardan biri sichqonchaning o‘ng tugmasi bosilganda paydo bo‘ladigan kontekst menyudir. U tez-tez ishlanadigan buyruqlarga tez kirish imkonini beradi.

14.6. C++ Builder da ilova dastur yaratish

C++ Builder dasturida ishni boshlashning birinchi qadami bu loyiha yaratishdir. Loyiha fayllarining boshlanishida avtomatik ravishda dastlabki matnlari bo‘ladi. Kompilatsiya qilingan va ishlashga tayyorlanganida u ilova dasturning bir qismi bo‘lib qoladi. Yangi loyiha yaratish uchun **File/New Application** menyu punktini tanlash kerak.

C++ Builder dastlabki **Project1.cpp** nomli loyiha fayli va **Project1.mak** nomli **make** fayl yaratadi. Loyihaga yangi forma qo‘shish kabi o‘zgartirish kiritilganda C++ Builder loyiha faylini yangilaydi.

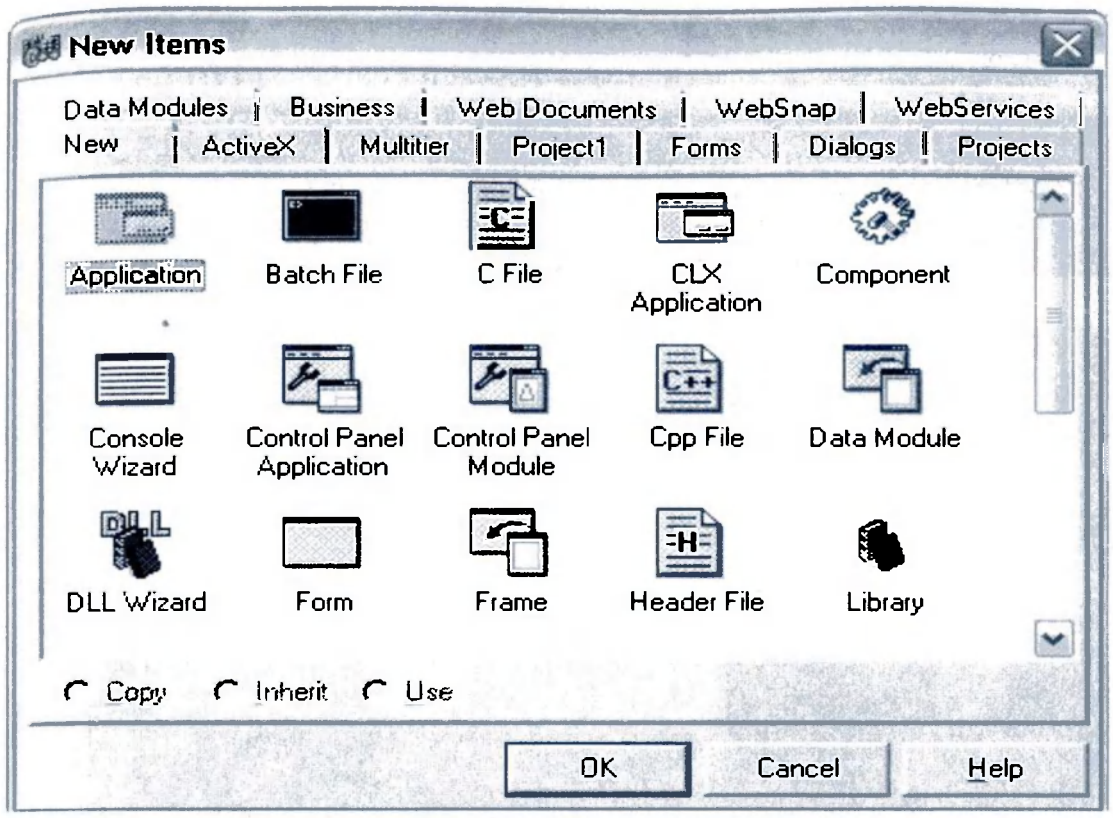
C:\Ebony\Projects\Project1.cpp

Unit1.cpp | Project1.cpp

```
//---------------------------------------------
#include <vcl\vcl.h>
#pragma hdrstop
//---------------------------------------------
USEFORM("Unit1.cpp", Form1);
USERES("Project1.res");
//---------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int
{
    Application->Initialize();
    Application->CreateForm(__classid(TForm1),
    Application->Run();
```

1: 1 | Modified | Insert

Loyiha fayli.



Formalar shabloni.

Odatda, loyiha yoki dastur bir necha formaga ega bo‘ladi. Loyihaga forma qo‘shish quyidagi qo‘shimcha fayllarni yaratadi:

• ***.DFM** kengaytmali forma fayli formani konstruksiyalash uchun oyna resurslari to‘g‘risidagi axborotni o‘z ichiga oladi.

• ***.CPP** kengaytmali modul fayli C++ dagi kodni o‘z ichiga oladi.

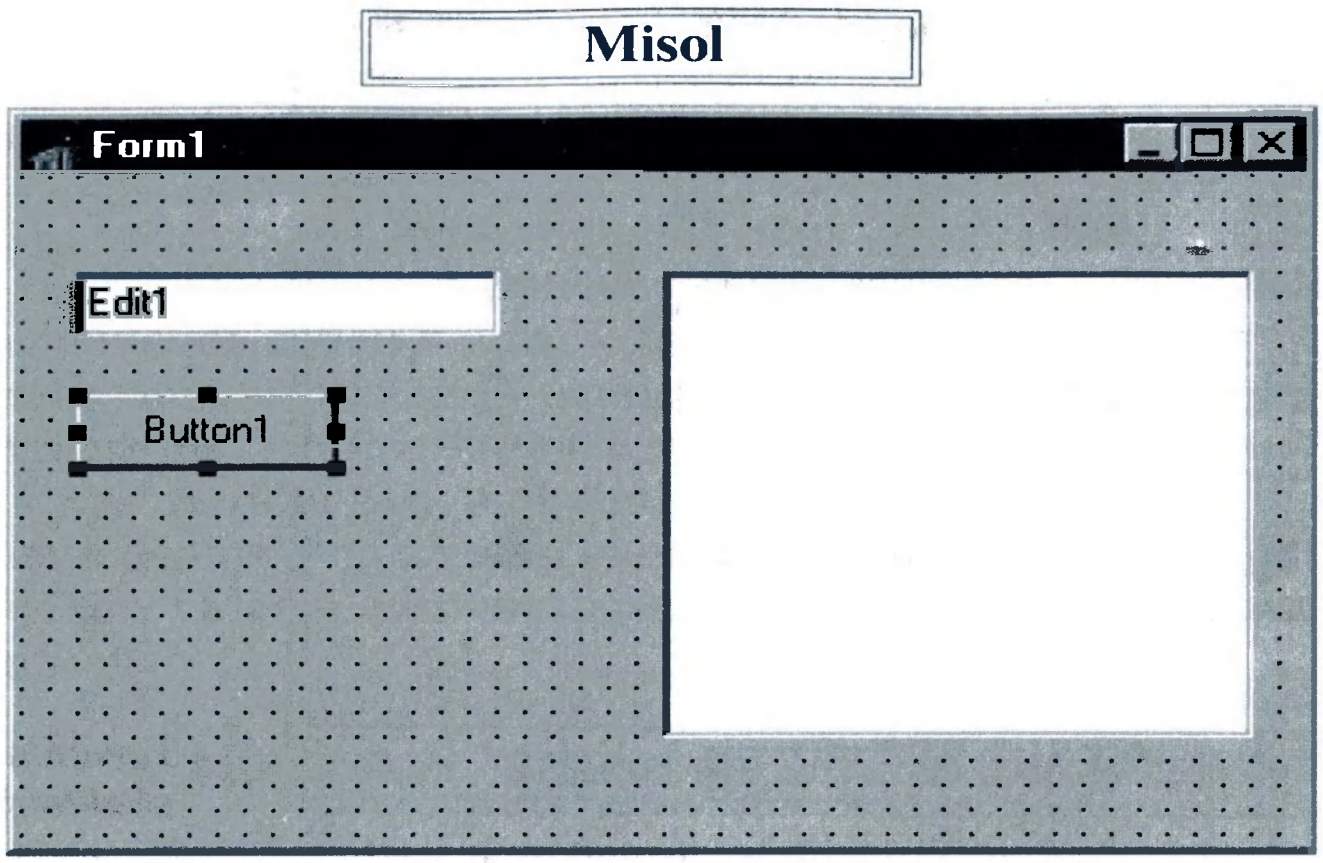
• ***.H** kengaytmali sarlavha fayli forma klassi tavsifini o‘z ichiga oladi.

Yangi forma qo‘shganingizda, loyiha fayli avtomatik ranishda yangilanadi.

Loyihaga yangi forma qo‘shish uchun **File/New Form** menyu punktini tanlang. Loyihaga qo‘shiladigan bo‘sh forma paydo bo‘ladi. Bundan tashqari, **File/New** menyu punktidan foydalanib, **Forms** sahifasini belgilash kerak hamda obyektlar ro‘yxatidan mos shablonni tanlash kerak.

Joriy loyihani faqat kompilatsiya qilishgina kerak bo‘lsa, **Compile** menyusidan **Compile** punktini tanlash kerak. Loyihani kompilatsiya qilish va joriy loyiha uchun bajariladigan fayl yaratish uchun, **Run** menyusidan **Run** punktini tanlash kerak.

Agar dastur ishga tushganida xatolik paydo bo‘lsa, C++ Builder dastur ishlashni to‘xtatadi va xatolik manbayi bo‘lgan qatorni kursor bilan belgilab qo‘yadi. Kerakli to‘g‘rilash ishlarini olib borishdan avval dasturni qayta ishga tushirib yuborish kerak. Buning uchun **Run** menyusidan **Run** punktini tanlash va dasturni yopish



Formaga komponentlarni joylashtirish.

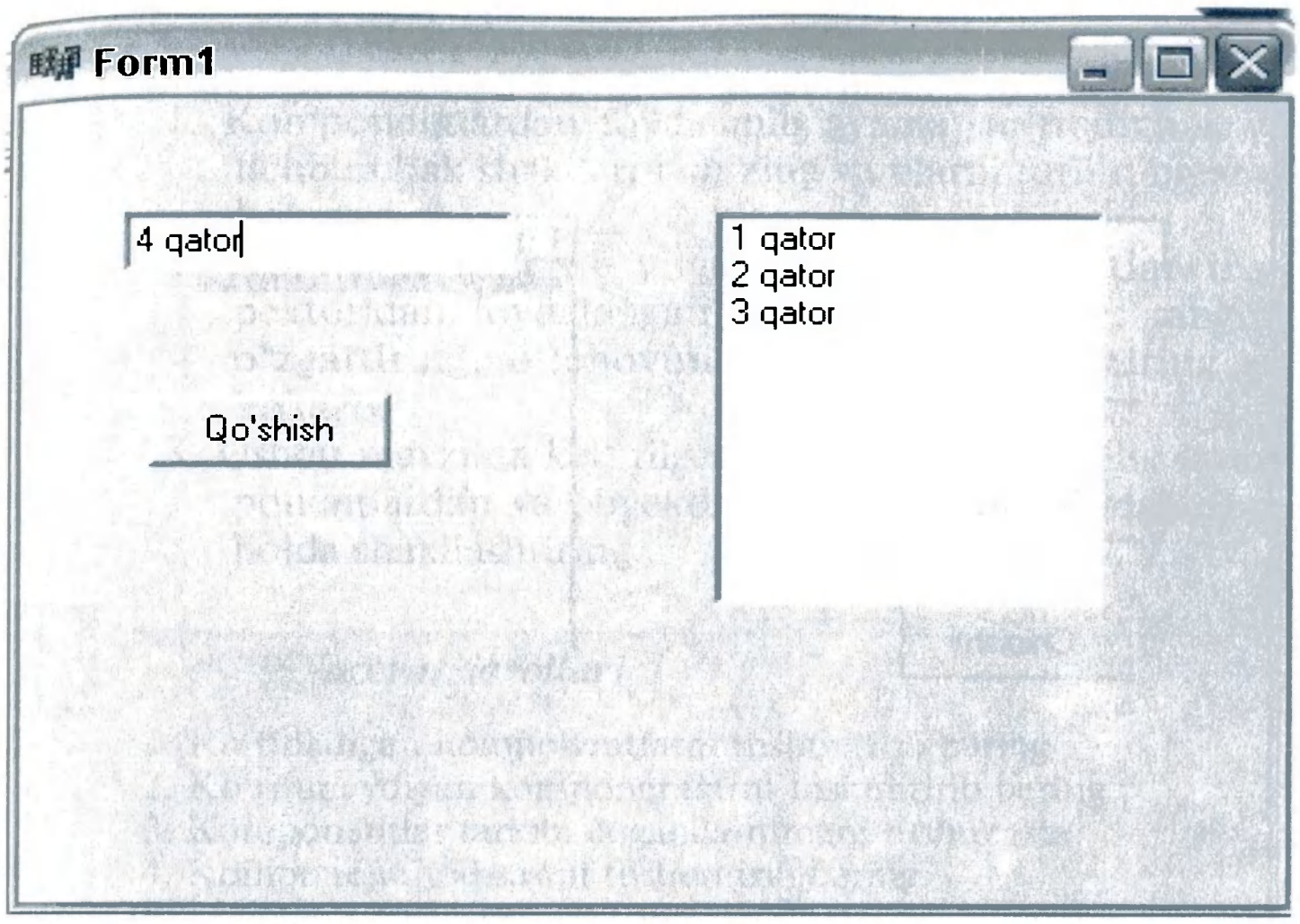
kerak. Keyin esa loyihaga o‘zgartirish kiritish kerak bo‘ladi. Shunda Windows resurslarini yo‘qotish ehtimoli kamayadi.

14.7. Oddiy ilova dasturini yararish

Endi tahrirlovchi maydonga matnni kiritib, uni tugma bosilishi bilan ro‘yxatga qo‘shadigan oddiy bir ilova dasturini yaratishga harakat qilamiz. Demak, yangi loyiha yaratish uchun **File/new Application** menyu punktini tanlaymiz, uning asosiy formasini **misol1.cpp** nom bilan, loyihani esa **misol1.mak** nom bilan saqlab qo‘yamiz. Formaga esa **Button, Edit** va **ListBox** komponentlarini palitradan olib qo‘yamiz:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
if (!(Edit1->Text == ""))
                {
                ListBox1->Items->Add(Edit1->Text);
                Edit1->Text = "" ;
                }

}
```

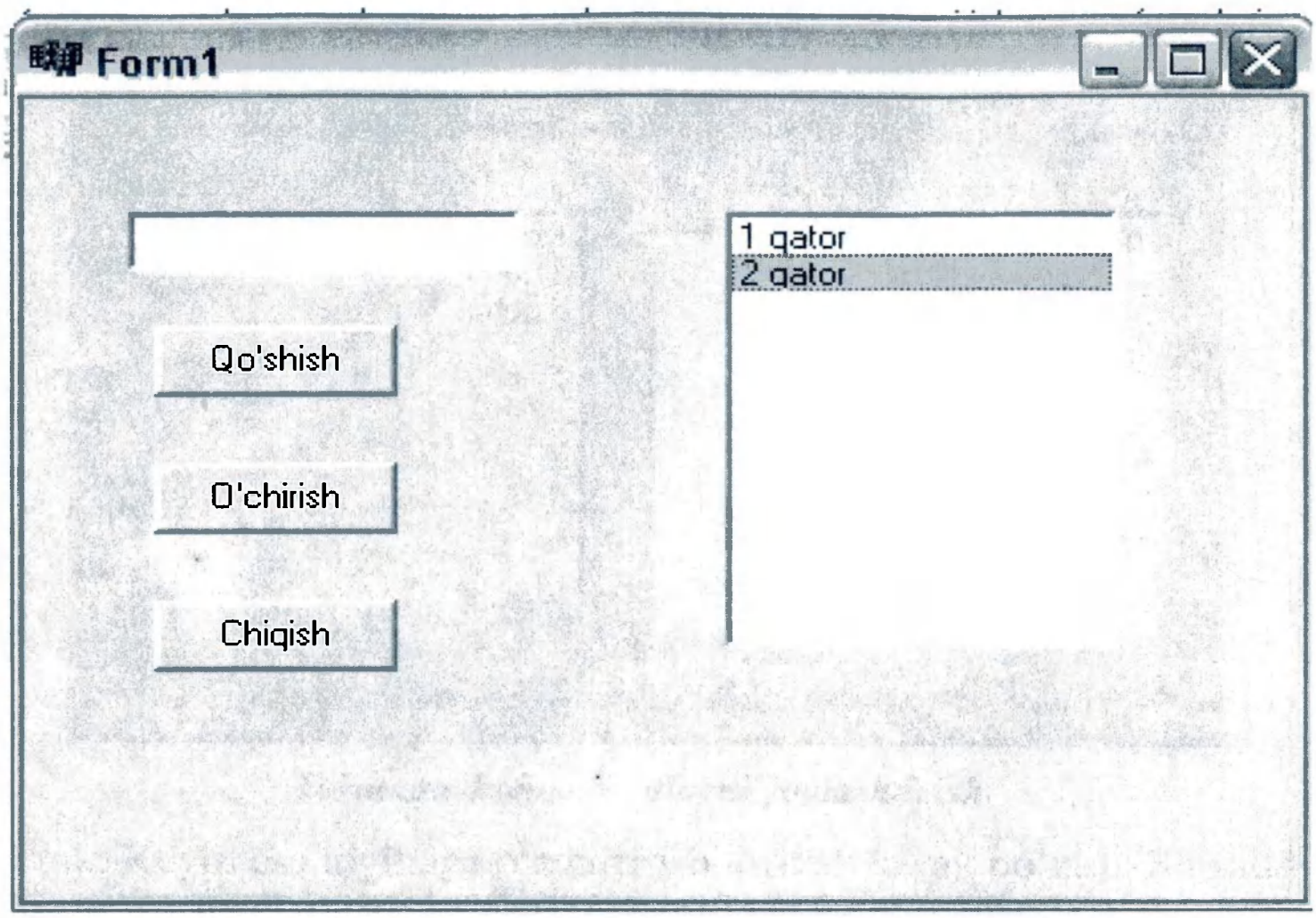


Tayyor ilova dastur ko‘rinishi.

Shundan keyin formadagi **Edit** komponentini tanlab, **Text** tarkibidagi joriy matnni o‘chirib yuboramiz. Keyin **Button1** uchun **Caption** tarkibini sozlaymiz. „Qo‘shish“ tugmasiga **OnClick** hodisasini qo‘shish uchun formadagi o‘sha tugmani belgilab, obyektlar inspektoridagi hodisalar sahifasini ochish kerak va ro‘yxat o‘ng tomonidagi **OnClick** hodisasini ikki marta bosish kerak. So‘ng **OnClick** ga mos kiritish satrida funksiya nomi paydo bo‘ladi. C++ Builder hodisa prototipini paydo qiladi va uni kod tahrirlagichda ko‘rsatadi. Shundan keyin, funksiya {} tanasiga kodni kiritish kerak bo‘ladi.

Dasturni kompilatsiya qilish uchun **Run** menyusidan **Run** punktini tanlaymiz. Endi tahrir maydoniga biror matnni kiritib, qo‘shish tugmasini bosish kerak va kiritilgan matn ro‘yxatga qo‘shilganligiga guvoh bo‘lamiz.

Endi esa, dasturimizga „Chiqish“ va „O‘chirish“ tugmalarini qo‘shib uni shakllantiramiz. Buning uchun ikkita tugma qo‘shib, ularning **Caption** tarkibini o‘zgartiramiz va o‘sha tugmalarga bog‘liq bo‘lgan hodisalarga jarayon yaratamiz.



Shakllantirilgan ilova dastur.

O‘chirish tugmasi uchun:

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
        if (!(ListBox1->ItemIndex == -1))
        ListBox1->Items->Delete(ListBox1-
>ItemIndex);
}
```

Chiqish tugmasi uchun:

```
void __fastcall TForm1::Button3Click(TObject *Sender)
{
        Close();
}
```

Dasturni saqlab qo‘yamiz va kompilatsiya qilamiz, keyin esa uni tekshirib ko‘ramiz.

MASHQLAR

1. Komponentlardan foydalanib aylana, to‘rtburchak va uchburchak shakllarni chizing va ularni turli ranglarga bo‘yang.
2. Komponentlar qo‘yilgan formani obyektlar inspektoridan foydalangan holda tahrir qiling: rangini o‘zgartiring, o‘lchovini piksellarda o‘zgartiring va hokazo.
3. Ushbu mavzuda keltirilgan ilova dasturi misolini komponentlardan va obyektlar inspektoridan foydalangan holda shakllashtiring.

Nazorat savollari

1. Ko‘rinadigan komponentlarni tushuntirib bering
2. Ko‘rinmaydigan komponentlarni tushuntirib bering.
3. Komponentlar tarkibi deganda nimani tushunasiz?
4. Komponent hodisasini tushuntirib bering.
5. Loyihalar menejerining vazifasi nima?
6. Yangi loyiha qanday yaratiladi?

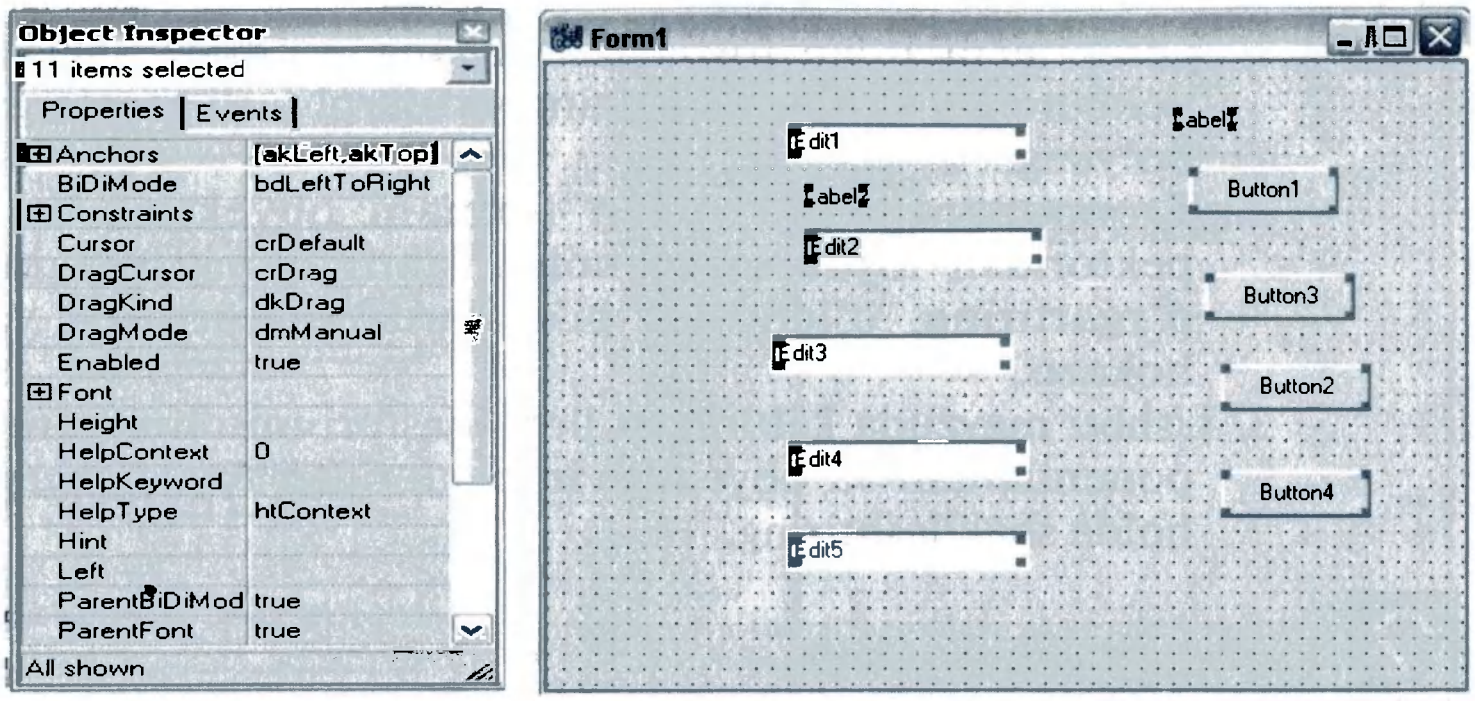
15-mavzu *C++ BUILDER KOMPONENTLARINI O‘RGANISH*

C++ Builder dasturida samarali foydalanuvchi interfeysini ishlab chiqish katta ahamiyat kasb etadi. Komponentlarni manipulatsiya qilishning ko‘p operatsiyalari **Edit** menyusida joylashgan. Bu menyuning ko‘plab opsiyalariga formadagi tarkibi o‘zgartirilishi kerak bo‘lgan komponentlar tanlangandagina murojaat qilish mumkin.

Bitta komponentni tanlash quyidagi usullar bilan amalga oshiriladi:

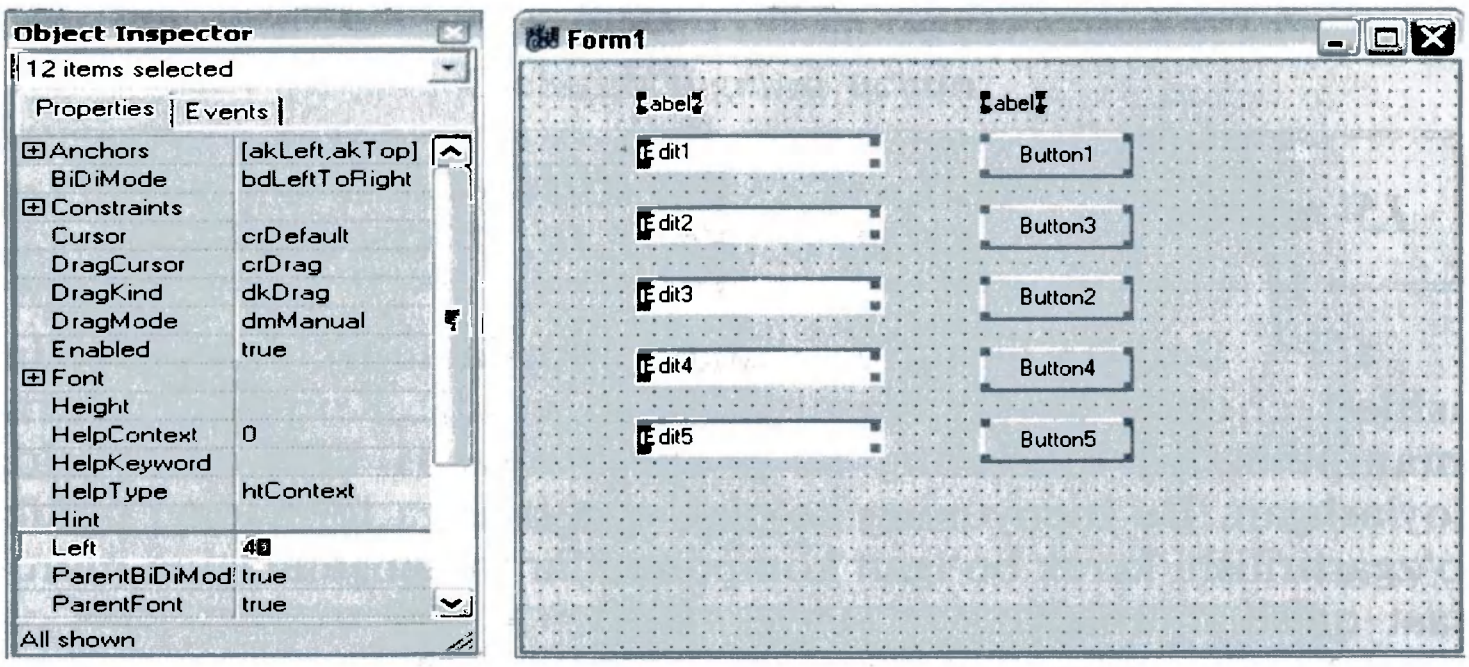
- **Shift** tugmasini bosib turgan holda, har bir komponentni sichqoncha tugmasi bilan bosish;
- sichqoncha chap tugmasini bosib, kerakli komponentlarni aylantirib belgilash.

15.1. Guruhli operatsiyalar uchun komponentlarni tanlash



Guruhli operatsiyalar uchun bir necha konponentlarni belgilash.

Umumiy belgilangan komponentlar tarkiblarini sozlash. Yuqoridagi rasm **Font** va **Left** tarkiblari o‘zgarish natijasini ko‘rsatib turmoqda. Barcha belgilangan komponentlar bir xil o‘zgartirilgan tarkiblarga ega bo‘ldi.



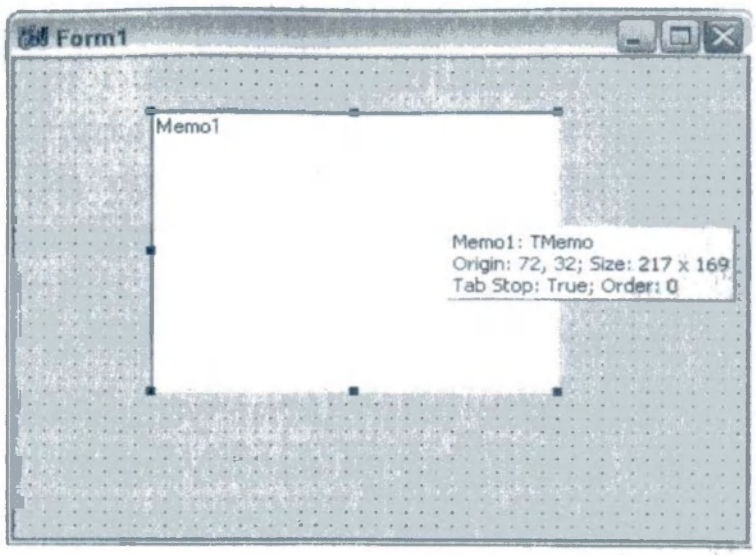
Komponentlarning bo‘linuvchi tarkiblarini o‘rnatish.

15.2. Komponentlar o‘lchovlarini o‘zgartirish

Komponentni qo‘shish uchun komponentlar palitrasini tanlab olish kerak. Keyin esa sichqoncha kursorini forma ichiga olib kelib, sichqoncha chap tugmasini bosish kerak. Natijada komponent

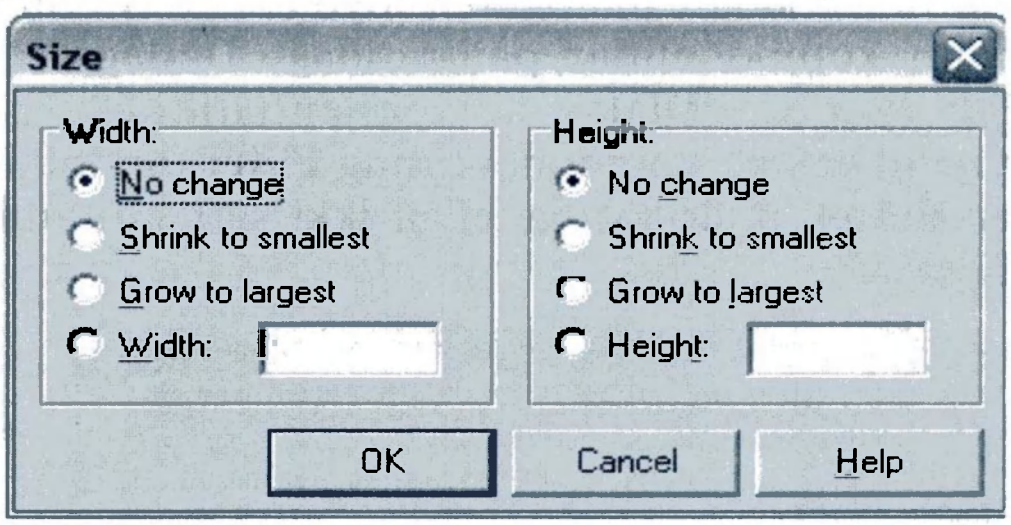
chegaralarini aks ettiruvchi to‘g‘ri to‘rtburchak forma ichida paydo bo‘ladi. To‘g‘ri to‘rtburchak kerakli o‘lchovga ega bo‘lganidan keyin sichqoncha tugmasini qo‘yib yuborish kerak.

Sichqoncha ko‘rsatkichini komponent atrofidagi kichkina qora to‘rtburchak nuqtalarga olib kelinsa, sichqoncha kursori ko‘rinishi o‘zgaradi. Sichqonchani bosgan holda ushbu kursorni komponent o‘zgarishi bilan birga jildirib, uning o‘lchovini o‘zgartirish mumkin.



Komponentni formaga qo‘shishda uning o‘lchovini o‘zgartirish.

Bir necha komponentlarning o‘lchovini o‘zgartirish uchun ularning hammasi belgilanadi. Keyin esa **Edit/Size** menyu punktini tanlash kerak. **Size** oynasi paydo bo‘ladi hamda o‘lchov opsiyasi tanlanadi. Komponent o‘lchovini piksellarda aniq o‘rnatish uchun **Width** va **Height** maydonlariga kerakli raqamlarni kiritish kerak. Keyin esa **OK** tugmasini bosish kerak.



Edit/Size dan foydalangan holda komponent tarkibini sozlash.

Bir xil turdagi komponentlarni formaga qo‘shish uchun **Shift** tugmasini bosgan holda komponentni palitradan tanlash kerak. Bu holda komponent atrofida to‘rtburchak paydo bo‘ladi. Shundan keyin sichqoncha tugmasining formaga har bir bosilishi unda komponentlar paydo bo‘lishiga olib keladi. Komponentni ko‘paytirish jarayonini yakunlagach, sichqoncha bilan komponentlar

palitrasidagi birinchi (ko‘rsatkich ko‘rinishidagi komponent) komponentni bosish kerak.

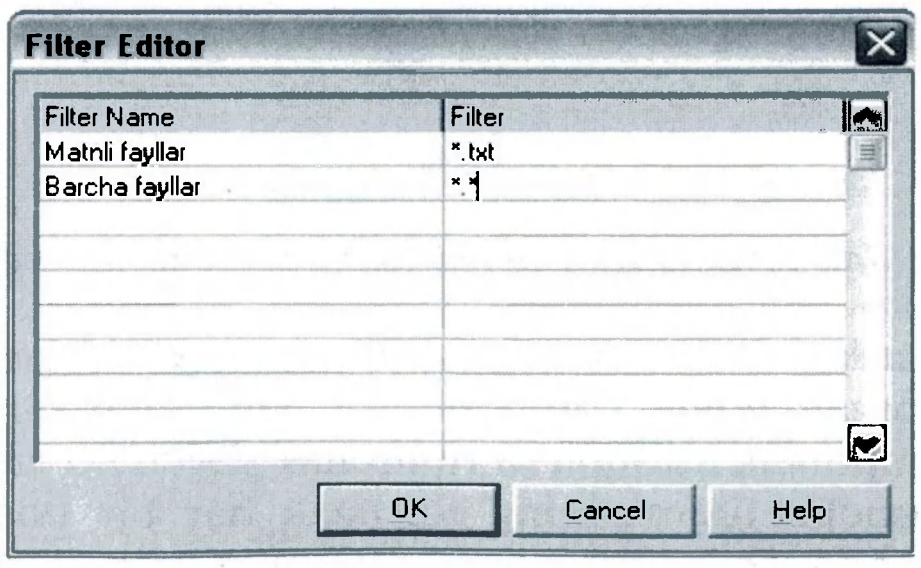
Misol: matn muharririni yaratish

15.3. Dastur formalarini loyihalash

Olingan bilimlarni yangi fayl yaratadigan, uni ochadigan, tahrir qiladigan va saqlaydigan hamda matnlar bilan ishlaydigan matn muharrini yaratishga harakat qilamiz. Buning uchun yangi formaga asoslangan yangi loyiha yaratamiz va uni **Tahrir.mak** deb nomlaymiz. Loyihaga esa **Tahrir.cpp** deb nom beramiz.

Bo‘sh formaga muharrirning kelgusida asboblar paneli bo‘ladigan **TPanel** komponentini qo‘yamiz. Olingan **Panel1** komponentinig **Align** tarkibi qiymatini **alTop** qilib belgilaymiz va **Caption** tarkibi qiymatini o‘chirib yuboramiz.

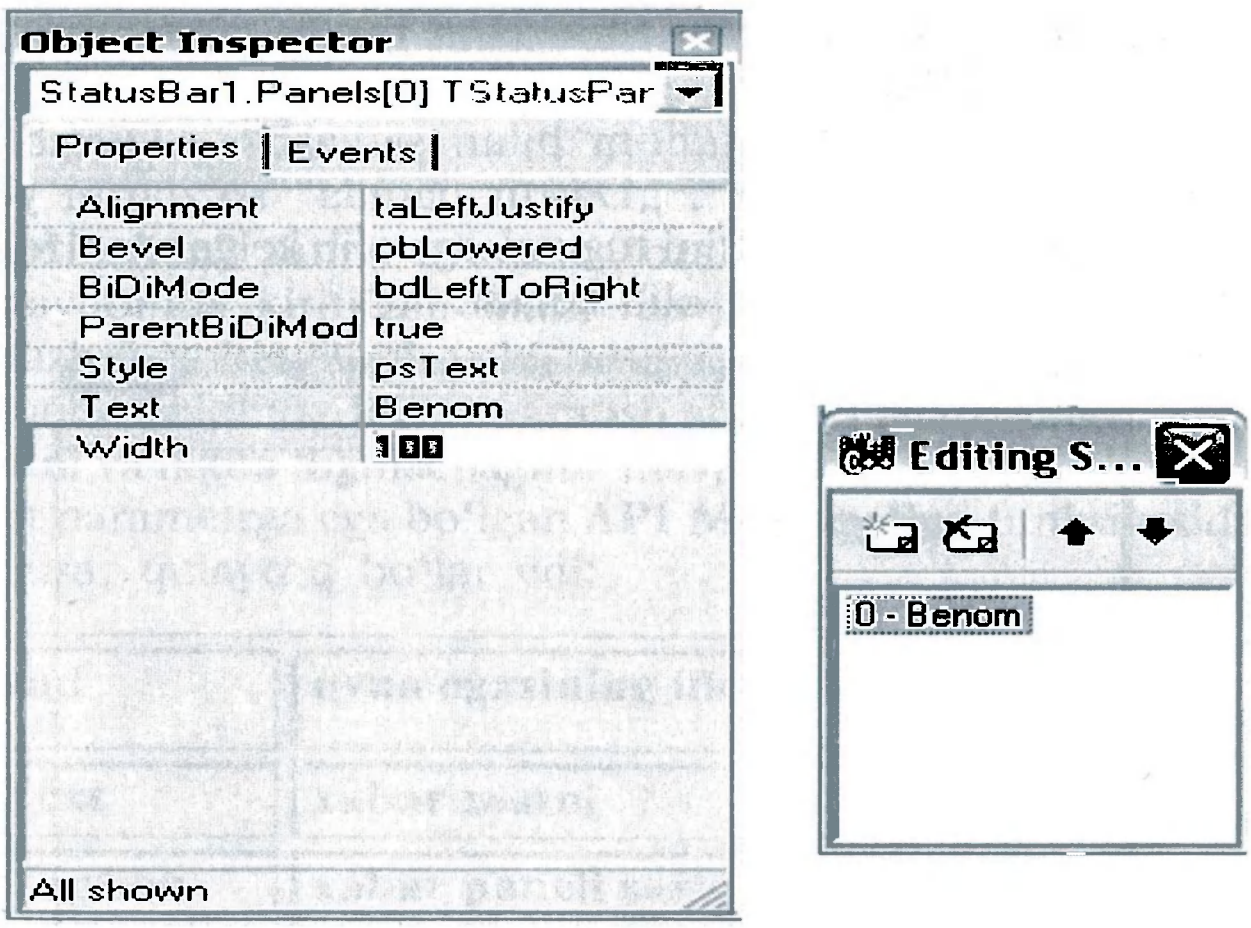
Keyin esa formaga **TMemo** komponentini joylashtiramiz. Uning **Align** tarkibi qiymatini **alClient, ScrollBars** tarkibi qiymatini **ssVertical** qilib belgilab, **Lines** tarkibi qiymatini bo‘sh qoldiramiz. Kelgusi matn muharririmiz fayllarni ochish va saqlash kerakligini unutmasligimiz kerak. Shuning uchun **comdlg32.dll** kutubhonasida mavjud Windows standart dialogidan foydalanamiz. Buni amalga oshirish uchun komponentlar palitrasidagi **Dialogs** sahifasidan **TOpenDialog** va **TSaveDialog** komponentlarini joylashtiramiz. Yaratilgan **OpenDialog1** komponentining **Filter** tarkibini o‘zgartiramiz: **Filter Editor** muloqot paneliga ikki satr kiritamiz va OK tugmasini bosamiz.



OpenDialog1 komponenti Filter tarkibini sozlash.

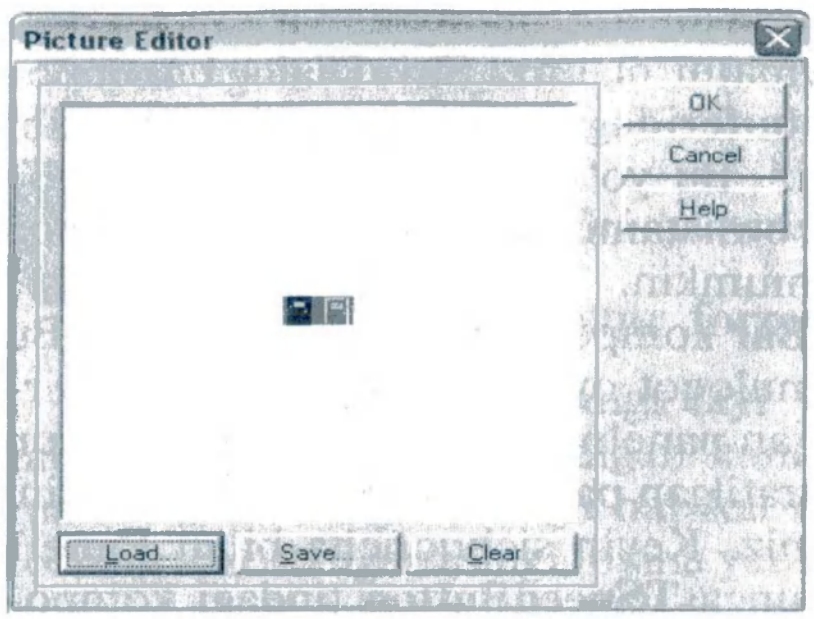
Endi **Filter** tarkibi to‘g‘risida joylashgan qiymatlar ro‘yxatidagi satr almashish buferidagi qiymatni olamiz. **SaveDialog1** komponentini tanlab, almashish buferidagi qiymatni **Filter** tarkibi to‘g‘risidagi qatorga qo‘yamiz. Bu yo‘l bilan biz ikkinchi dialog uchun **Filter** tarkibi qiymatini o‘rnatamiz. Ixtiyoriy ravishda boshqa parametrlarni ham sozlash mumkin.

Endi esa formaga **StatusBar** komponentini joylashtiramiz. Bu tarkib tahrirlagichi ekranga muloqot oyna bo‘lib chiqadi. Tahrirlanadigan fayl nomi chiqadigan panelni sozlaymiz. Buning uchun **New** tugmasini bosamiz va yaratilgan panelning **Width** parametrini 100 ga teng qilib o‘zgartiramiz. Keyin sichqoncha bilan **Panel1** komponentini tanlaymiz va unga **TSpeedButton** tipdagi komponentdan 9 ta joylaymiz. Buni osonroq amalga oshirish uchun esa **Shift** tugmasini bosib, **Additional** sahifasidan **Speed Button** komponentini tanlaymiz.



StatusBar1 komponenti Panels tarkibini sozlash.

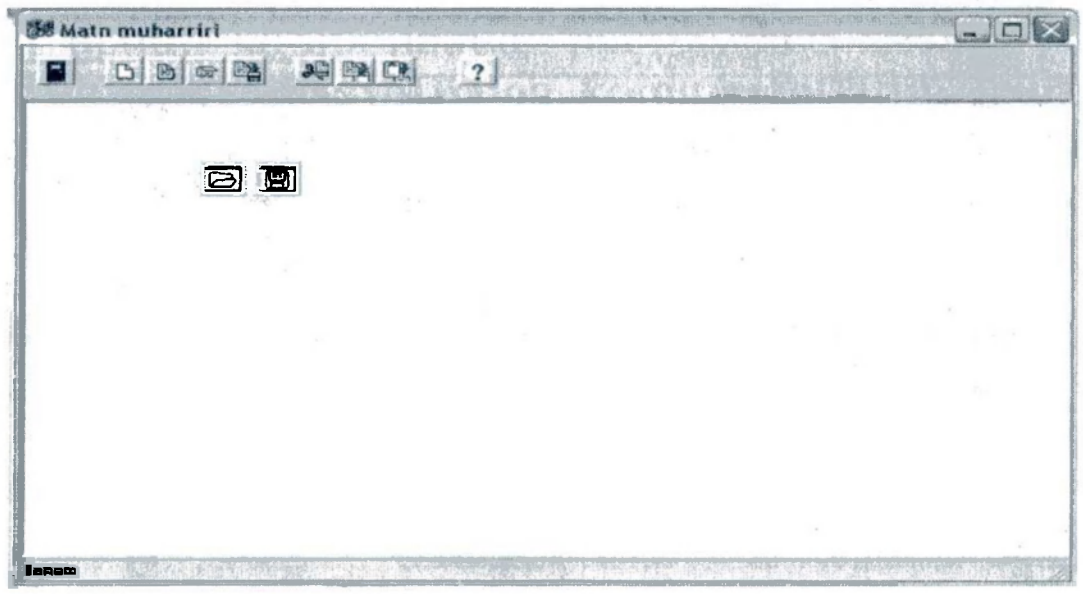
Tugmalarimizni rasmlar bilan jihozlaymiz. Buning uchun bu tugmalarning **Glyph** tarkibini o‘zgartiramiz. Bu bilan biz C++ Builder tarkibiga kiruvchi ko‘plab rasmlar to‘plamidan foydalanishimiz mumkin. Bizning misolimiz uchun ushbu katalogdan



SpeedButton1 ... SpeedButton9 komponentlari Glyph tarkibini sozlash

Doorshut.bmp, Filenew.bmp, Fileopen.bmp, Filesave.bmp, Cut.bmp, Copy.bmp, Paste.bmp, Help.bmp fayllari tanlangan.

Endi yuqoridagi rasmga qarab, formadagi tugmalarni tartibli joylashtiramiz. Bu tugmalar **ShowHint** tarkibi qiymatini **True** ga aylantiramiz, **Hint** tarkibi qiymatiga esa „Chiqish“, „Yaratish“, „Ochish“, „Saqlash“, „...nom bilan saqlash“, „Qirqib olish“, „Nusxa olish“, „Qo‘yish“, „Dastur haqida“ so‘zlarini yozamiz. Bu sichqoncha ko‘rsatkichini tugmalarga olib kelganda, izohli sariq yorliqlar paydo bo‘lishiga olib keladi.



Ilova dastur formasining umumiy ko‘rinishi.

15.4. Hodisa jarayonlarini yaratish

Endi tugmachalarimiz uchun **OnClick** hodisalari jarayonlarini yozamiz. **SpeedButton3** muloqot panelida fayl nomining paydo bo‘lishi va uni tahrir qilish uchun faylning ochilishiga javob beradi:

```
void __fastcall TForm1::SpeedButton3Click(TObject *Sender)
{
        if (OpenDialog1->Execute()) Memo1->Lines-
>LoadFromFile(OpenDialog1->FileName);
        StatusBar1->Panels->Items|0|->Text=OpenDialog1-
>FileName;
}
```

SpeedButton5 muloqot panelida tahrir qilinadigan faylni berilgan nom bilan saqlash va uning nomini panelda chiqishiga javobgardir:

```
void __fastcall TForm1::SpeedButton5Click(TObject *Sender)
{
        if (SaveDialog1->Execute()) Memo1->Lines-
>SaveToFile(SaveDialog1->FileName);
        StatusBar1->Panels->Items[0]->Text=SaveDialog1-
>FileName;
}
```

SpeedButton2 tahrir oynasini tozalashga javobgardir. **B**iroq tahrir maydonida kiritilgan matn mavjud bo‘lsa, foydalanuvchi uni saqlashni xohlaydimi yoki yo‘qmi, so‘rashi kerak bo‘ladi. Buning uchun alohida forma yaratish shart emas. Chunki, bunda savol matni va ikkita tugmachagina mavjud bo‘ladi. Agar Windows ning to‘rt parametrga ega bo‘lgan **API MessageBox** funksiyasidan foydalanilsa, qulayroq bo‘lar edi:

hwnd	**oyna egasining identifikatori**
lpText	**xabar matni**
lpCaption	**xabar paneli sarlavhasi**
uType	**xabar paneli stili**

Kiritilgan matnni saqalsh uchun **SpeedButton5Click** tayyor funsiyasidan foydalanilsa ham bo‘laveradi. Bunga mos ravishda **SpeedButton2** tugmasini bosganda, hodisa jarayoni quyidagicha ko‘rinishda bo‘ladi:

```
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
        if (Memo1->Lines->Count>0)
        {
        if (MessageBox(0,"Tahrir oynasidagi matnni
saqlaysizmi?",
        "Saqlashni tasdiqlang",MB_YESNO)==IDYES)
        {
        SpeedButton5Click(Sender)
        }
        };
        Memo1->Clear();
        StatusBar1->Panels->Items[0]->Text="Benom";
}
```

SpeedButton1 tugmasi dasturni yopadi. Bu holatda ham foydalanuvchiga kiritilgan matnni saqlash taklif etilishi kerak. Buning uchun yaratilgan **SpeedButton2Click** funksiyasidan foydalaniladi.

```
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
        SpeedButton2Click(Sender);
        Close();
}
```

SpeedButton4Click tugmasi tahrir qilinadigan faylni saqlashga javob beradi:

```
void __fastcall TForm1::SpeedButton4Click(TObject *Sender)
{
        if (StatusBar1->Panels->Items[0]->Text=="Без имени")
        SpeedButton5Click(Sender);
        else Memo1->Lines->SaveToFile(StatusBar1->Panels-
>Items[0]->Text)
}
```

Bu yerda ozgina tushuntirish ishi talab qilinadi. Agar foydalanuvchi mavjud faylni ochgan bo‘lsa va tahrirlangan faylni biron-bir nom bilan saqlagan bo‘lsa, u holat paneli **(StatusBar1)** da ko‘rsatiladi hamda faylga nom berish oynasi ochilishi talab qilinmaydi. Agar faylga nom berilmagan bo‘lsa, faylni saqlash oynasini **SpeedButton5Click** funksiyasidan foydalangan holda chaqirish kerak bo‘ladi.

SpeedButton6 va **SpeedButton7** tugmalari tahrir oynasidagi belgilangan matnni almashish buferiga o‘tkazish va nusxa olish uchun javobgardirlar:

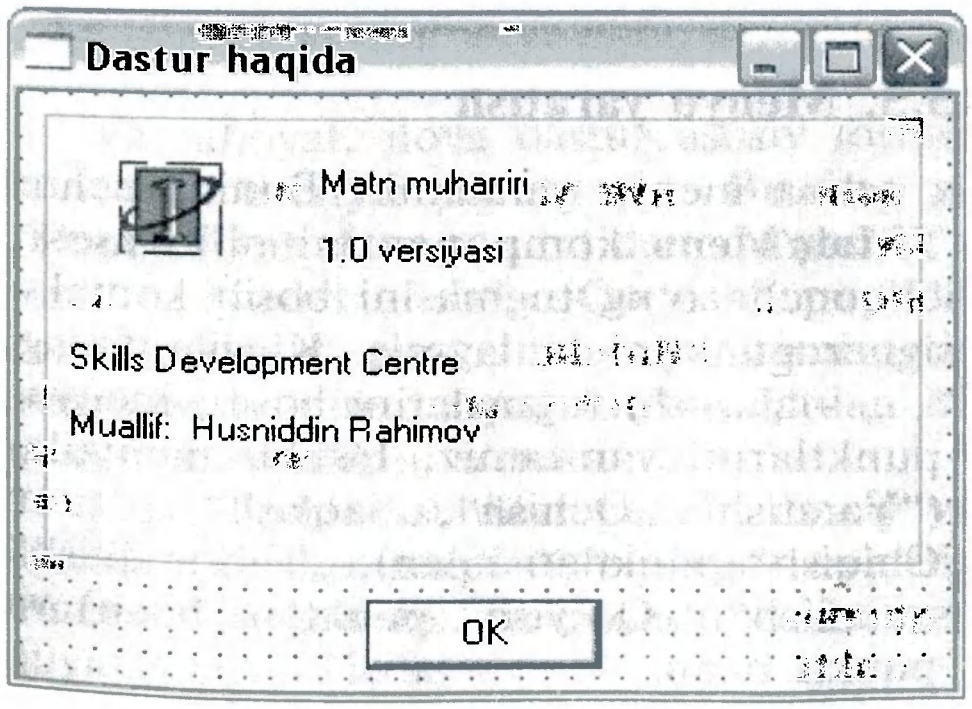
```
void __fastcall TForm1::SpeedButton6Click(TObject *Sender)
{
        Memo1->CutToClipboard();
}
```

```
void __fastcall TForm1::SpeedButton7Click(TObject *Sender)
{
        Memo1->CopyToClipboard();
}
```

SpeedButton8 tugmasi qirqib olingan yoki nusxasi olingan matnni kursor turgan joyga qo‘yish vazifasini bajaradi:

```
void __fastcall TForm1::SpeedButton8Click(TObject *Sender)
{
        Memo1->PasteFromClipboard();
}
```

SpeedButton9 tugmasi ekranga „Dastur haqida“ muloqot oynasini chiqaradi. Bunday muloqot oynalar zamonaviy ilova dasturlar uchun standart hisoblanadi. Keling misol uchun, C++ Builder obyektlari ro‘yxatidagi tayyor **About** paneli shablonidan



„Dastur haqida“ muloqot oynasi ko‘rinishi.

foydalanamiz. **File/New** menyu punktini tanlab, **New Items** muloqot oynasini ochamiz. Bu oynadan **Forms** opsiyasidan **AboutBox** shablonini tanlaymiz. Endi olingan formani tahrirlaymiz.

Hozir bizning ilova dasturimiz ikki formadan tashkil topdi. Dasturning asosiy formasi birinchi yaratilgan **Form1** hisoblanadi. Dastur ishga tushganda, ikkala forma avtomatik ravishda ishga tushadi va ekranda asosiy forma ko'rinib turadi. Lekin shuni e'tiborga olish lozimki, formaning yaratilishi va uning ekranda paydo bo'lishi operatsion tizimning ba'zi bir resurslarini oladi.

SpeedButton9 tugmasi bosilganda, hodisa jarayoni quyidagicha ko'rinishda bo'ladi:

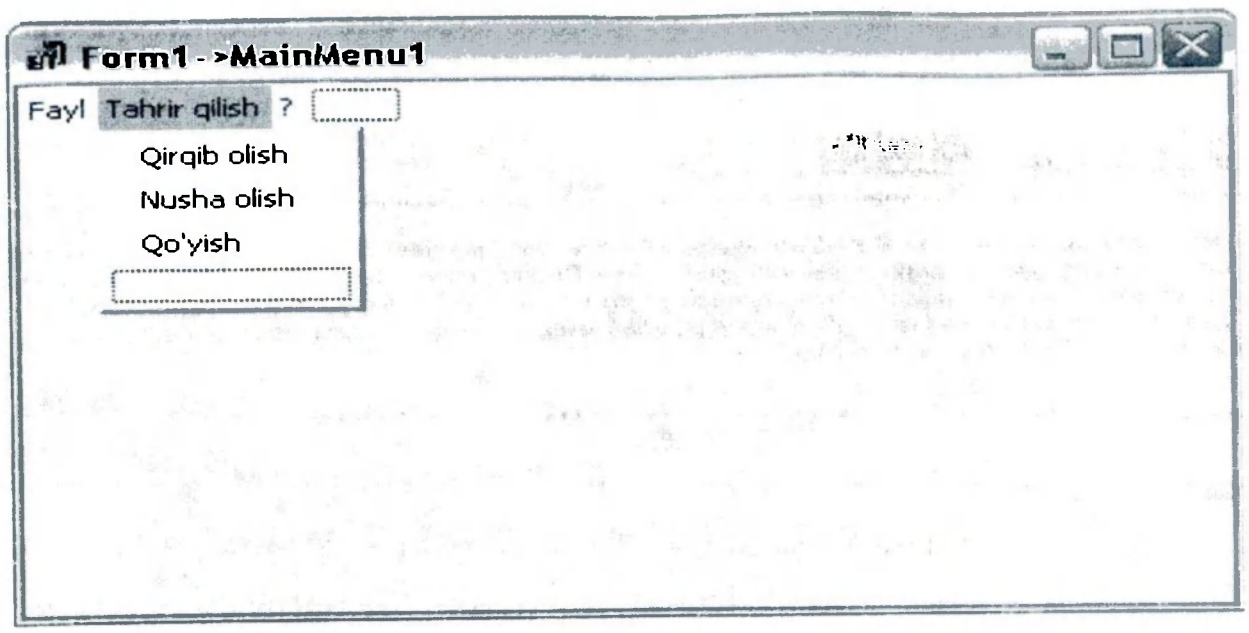
```
void __fastcall TForm1::SpeedButton9Click(TObject *Sender)
{
        Application->CreateForm(__classid(TAboutBox),
&AboutBox);
        AboutBox->ShowModal();
        AboutBox->Free();
}
```

Ushbu hodisa jarayonining birinchi operatori **AboutBox** formasi nusxasini yaratadi. Ikkinchi operator uni modal muloqot oyna (oynani yopmagunga qadar boshqa oynalarga murojaat qilishga ruxsat bermaydigan oyna) qilib ko'rsatadi.

Agar qolib ketgan keraksiz forma o'chirib yuborilmasa, (buning uchun **SpeedButton9Click** funksiyasidagi oxirgi operator kerak bo'ladi), ushbu funksiyaning har bir chaqirilishi resurslar batamom tugatilmagunga qadar **AboutBox** nusxalari operativ xotirada to'lib ketadi.

15.5. Menyu yaratish

Endi muharririmiz uchun menyu yaratamiz. Buning uchun standart sahifasidan **TMainMenu** komponentini olib, asosiy formaga qo'yamiz. Sichqoncha .o'ng tugmasini bosib kontekst menyudan **Menu Designer** punktini tanlaymiz. Klaviaturadagi ko'rsatkich („←“, „↑“, „→“, „→“) tugmalarini bosib, menyuning boshqa-boshqa punktlarini yaratamiz. Ushbu menyular quyidagilardir: „Fayl“ (“Yaratish“, „Ochish“, „Saqlash“, „...nom bilan saqlash“, „-“, „Chiqish“ punktlari bilan), „Tahrir qilish“ (“Qirqib olish“, „Nusxa olish“, „Qo'yish“ punktlari bilan) va „?“-“Dastur haqida“ punkti bilan.



Menu Designer yordamida menyu yaratish.

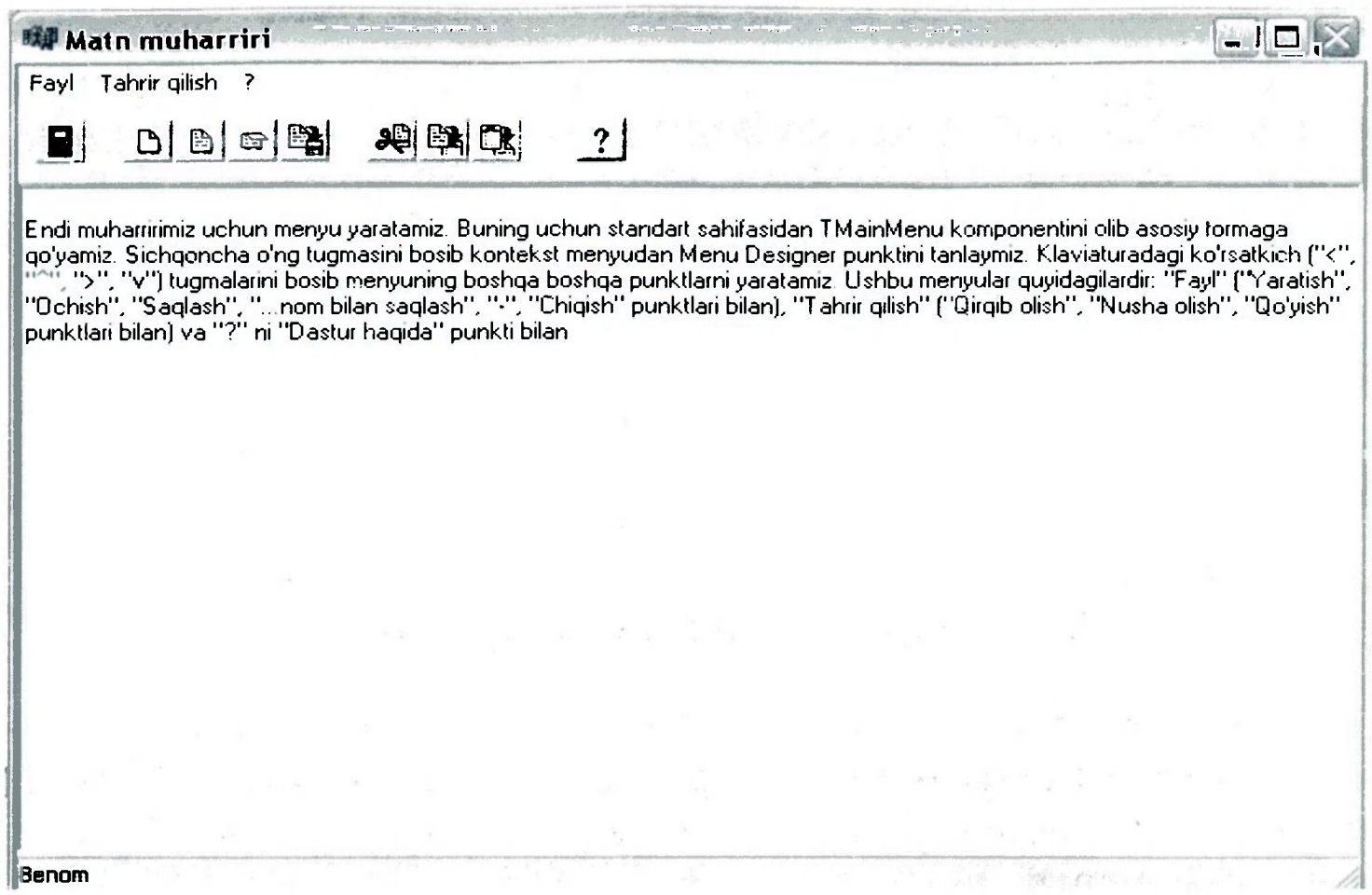
Endi obyektlar inspektoridan hodisalar sahifasini tanlab avval yaratilgan **SpeedButton1Click ... .. SpeedButton9Click** funksiyalarini menyu punktlaridan nomlari mos kelganlarini bir-biriga bog‘laymiz.

Bizda foydalanilmagan „Asboblar paneli“ menyu punkti qoldi. Ushbu menyu punktining **Checked** tarkib qiymatini **True** qilib qo‘yamiz. „Asboblar paneli“ menyu punkti uchun quyidagi **OnClick** hodisa jarayonini yaratamiz:

```
void __fastcall TForm1::N9Click(TObject *Sender)
{
        N9->Checked=!N9->Checked;
        Panel1->Visible=N9->Checked;
}
```

Va nihoyat, ilova dastur asosiy formasi elementlari uchun kontekst menyu yaratamiz. Buning uchun formaga ikki **TPopupMenu** komponentidan qo‘yamiz. Birinchisini „Qirqib olish“, „Nusha olish“, „Qo‘yish“ punktlari bilan, ikkinchisini esa „Berkitish“ punkti bilan yaratamiz. Ushbu menyu punktlar menyular uchun mavjud funksiyalardan mosini tanlab **OnClick** hodisa jarayonlarini o‘rnatamiz.

Demak, biz asboblar paneli, asosiy va kontekst menyulari hamda „Dastur haqida“ muloqot oynasiga ega bo‘lgan matn muharririni yaratdik. Ilova dasturimizning oxirgi to‘liq ko‘rinishi quyidagicha bo‘ladi:



Tayyor ilova dastur ko'rinishi.

MASHQLAR

1. Yuqoridagi „Matn muharriri“ ilova dasturiga qidiruv funsiyasi dasturini qo'shib uni mukammallashtiring.
2. Yuqoridagi „Matn muharriri“ ilova dasturiga chop etish funksiyasi dasturini qo'shib uni mukammallashtiring.
3. Komponentlardan foydalanib, sodda kalkulator dasturini tuzing.

Nazorat savollari

1. Komponentni belgilash usullarini aytib bering.
2. Bir necha komponentlarning bir xil tarkiblarini birdaniga o'zgartirish qanday amalga oshiriladi?
3. Komponent o'lchovi qanday o'zgartiriladi? Komponent o'lchovini aniq piksellarda qanday o'zgartiriladi?
4. **File Editor** tarkibining vazifasi nima?
5. Obyektlar inspektori nima?
6. Tugmalar rasmlar bilan qanday jihozlanadi?
7. Menyu yaratish qanday amalga oshiriladi?

MUNDARIJA

32.973.26-018.1
R33

Rahimov H.

Zamonaviy dasturlash tillari: Kasb-hunar kollejlari uchun o‘quv qo‘ll. / H. Rahimov, T.Dehqonov; **O‘zbekiston Respublikasi Oliy va o‘rta maxsus ta’lim vazirligi, O‘rta maxsus, kasb-hunar ta’limi markazi** – T.: «O‘qituvchi» NMIU, 2007. – 88b.

I. Yunusov A.S. va boshq.

ББК 32.97326-018.1я 722

HUSNIDDIN RAHIMOV,
TAVAKKAL DEHQONOV

ZAMONAVIY DASTURLASH TILLARI

Kasb-hunar kollejlari uchun o‘quv qo‘llanma

„O‘qituvchi“ nashriyot-matbaa ijodiy uyi
Toshkent – 2007

Muharrir *O‘. Husanov*
Badiiy muharrir *Sh. Xo‘jayev*
Tex. muharrir *S. Tursunoya*
Musahhih *Z. Sodiqova*
Kompyuterda sahifalovchi *M. Sagdullayeva*

Original-maketdan bosishga ruxsat etildi 05.11.2007. Bichimi $60\times90^{1}/_{16}$.
Kegli 11 shponli. Tayms garn. Ofset bosma usulida bosildi. Shartli b. t. 5,5.
Nashr. t. 5,5. 3780 nusxada bosildi. Buyurtma № 176.

O‘zbekiston Matbuot va axborot agentligining „O‘qituvchi“ nashriyot-matbaa ijodiy uyi. Toshkent —129, Navoiy ko‘chasi, 30-uy. // Toshkent, Yunusobod dahasi, Murodov ko‘chasi, 1- uy. Shartnoma 12-149-07.


«O‘QITUVCHI»
WWW.OQITUVCHI.UZ
124-04-12
ISBN 978-9943-02-130-3
9 789943 021303