O’ZBEKISTON RESPUBLIKASI

OLIY VA O’RTA-MAXSUS TA’LIM VAZIRLIGI

SAMARQAND DAVLAT UNIVERSITETI

MEXANIKA-MATEMATIKA FAKULTETI

AMALIY MATEMATIKA VA INFORMATIKA BO’LIMI

5110700-INFORMATIKA O’QITISH METODIKASI YO’NALISHI

„C++ DASTURLASH TILIDA HISOBLASH KALKULYATORI YARATISH“ MAVZUSIDAN

KURS ISHI

206 -guruh talabasi: Qudratova Z

Kurs ishi rahbari: Nazarov F

SAMARQAND-2016

MUNDARIJA

KIRISH

I. C++ dasturlash tilida hisoblash kalkulyatori yaratishda nazariy tushunchalar

1.1. C++ o’zgaruvchilarni tasvirlash

1.2. Ma’lumotlar tipi va ma’lumotlar tipiga keltirish

II. C++dasturlash tilida hisoblash kalkulatori yaratishning dasturiy ta’minoti

2.1. C++ dasturlash tilida hisoblash kalkulatori yaratishda foydalanilgan komponentalar

2.2. C++ dasturlash tilida hisoblash kalkulatori yaratish dasturi

XULOSA

FOYDALANILGAN ADABIYOTLAR

KIRISH

Hozirgi kunda respublikamizdagi oliy o‘quv yurtlarida, akademik litseylar, kasb-hunar kollejlarida va maktablarda ham “Informatika va axborot texnologiyalari” yo‘nalishi va mutaxassisliklariga turli xil dasturlash tillarini o‘rgatish mo‘ljallangan. “Informatika va axborot texnologiyalari” yo‘nalishiga talab kuchayib ketmoqda. Texnika rivojlangan zamonda malakali, bilimli, informatik va dasturchilarga talab ortib bormoqda. Shu sababli ham men ushbu kurs ishimda hisoblash tizimi bo’lgan kalkulyator dasturini tuzib chiqib o’rganishni niyat qildim. Kalkulyator tuzish uchun men C++ dasturlash tilinining Builder 6 dasturini tanladim. Bu dasturlash tilining tanlashimdan maqsad dasturlash tillarining yuzdan ortiq ko‘rinishlari mavjud, lekin qo‘llanilishi ko‘lamiga qarab C/C++ va C# dasturlash tillari yuqori dasturlash sinfiga mansubdir. Mutaxassislarning fikriga ko‘ra C++ dasturlash tili Assembler dasturlash tiliga eng yaqin bo‘lib, tezlik jihatidan 10 % ortda qolar ekan. Keyingi yillarda amaliy dasturchilarga juda ko‘p integratsion dastur tuzish muhitlari taklif etilmoqda. Bu muhitlar u yoki bu imkoniyatlari bilan bir–biridan farq qiladi. Aksariyat dasturlashtirish muhitlarining fundamental asosi C/C++ tiliga borib tarqaladi. Bu dasturda ishlashimga yana bir sabab boshqa obektlarga mo’ljallanmagan dasturlash tillarida faqat masalani matematik dasturi tuzib natija olinadigan bunda yani C++ Builderda esa boshqa imkoniyatlar va komponentalar bilan ham ishlashni o’rganmoqchiman. Builder 6 boshqa dasturlarga qaraganda qulay interfeysga egaligi bilan ajralib turadi.

Ushbu kurs ishida hozirgi kunda barcha sohalarda qo’llaniladigan hisoblash tizimi bo’lgan kalkulyator dasturini tuzib chiqmoqchiman. Ushbu dastur kichik dastur bo’lsada kelajakda boshqa dasturlar tuzishim uchun fundament vazifasini bajarishiga ishonaman.. Chunki hozirgi kunda hisoblash bilan katta dasturlar paketini yaratish imkoniyatiga egamiz. Shuning uchun kichik kalkulyator bilan ishlovchi dastur ishlab chiqaraman.

Bu dasturni tuzib nafaqat hisoblashni balki dasturni ishlashni bilmaydigan foydalanuvchi ham bir ko’rinishda tushinadigan qilib dastur tuzmoqchiman.

Men bu dasturni tuzib kalkulyator yaratishni va ular orqali shunga o’xshash yana boshqa obektlar yaratishni oldimga maqsad qilib qo’ydim. Bu dasturda ishlashimga yana bir sabab boshqa obektlarga mo’ljallanmagan dasturlash tillarida faqat masalani matematik dasturi tuzib natija olinadigan bunda yani C++ Builderda esa boshqa imkoniyatlar va komponentalar bilan ham ishlashni o’rganmoqchiman.

I. C++ dasturlash tilida hisoblash kalkulyatori yaratishda nazariy tushunchalar

§1.1. C++ o’zgaruvchilarni tasvirlash

Ixtiyoriy dasturda xisoblashlarni bajarish talab etiladi. Qiymatlarni hisoblash uchun operandlar, amallar ishoralari va qavslardan iborat ifadalardan foydalaniladi. Operandlar hisoblashlar uchun ma’lumotlarni beradi. Operandlar bajarish uchun zarur amallarni bajaradi. Har bir operand o’z navbatida ifoda yoki uning xususiyat holatidan biri hisoblanadi, masalan, konstanta yoki o’zgaruvchi bilan. Amallar bajarilish tartibini o’zgartirish uchun qavslar ishlatiladi. Ifodalar tarkibining qismlari va ularni hisoblash oidalarini qaraymiz.

O’zgaruvchilar

O’zgaruvchi - bu ma’lum tipdagi ma’lumotlar saqlanadigan xotiraning nomlangan sohasi. O’zgaruvchining nomi va qiymati bor. Nom qiymat saqlanadigan xotira sohasiga ko’rsatish uchun xizmat qiladi. Dasturni bajarish paytida o’zgaruvchi qiymatini o’zgartirish mumkin.

Foydalanishdan oldin ixtiyoriy o’zgaruvchi tavsiflanishi lozim. a ismli va x xaqiqiy o’zgaruvchili butun o’zgaruvchi tavsifiga misol.

```
int a: float x;
```

O’zgaruvchilarni tasvirlash operatorining umumiy ko’rinishi:

```
[xotira sinfi] [const] tip nom [initsializator]
```

Bu operatorning tarkibiy qismlarini berishning qoidalarini qaraymiz:

□ Xotiraning majburiy bo’lmagan sinfi auto, extern, satatic va register qiymatlardan birini qabul qilishi mumkin;

□ const modifikator o’zgaruvchi qiymatini o’zgartirish mumkin emasligini ko’rsatadi. Bunday o’zgaruvchi nomlangan konstanta yoki kanstantalar deb ataladi.

□ Tavsiflashda o’zgaruvchiga boshlang’ich qiymat berish mumkin, bu initsializatsiya deb ataladi. Initsializatorni ikki shaklda: tenglik belgsi bilan = qiymat

yoki oddiy qavslar ( ) bilan yozish mumkin.

Konstanta e’lon qilishda initsializatsiyalanishi mumkin bitta operatorda bir tipdagi bir nechta o’zgaruvchini vergullar bilan ajratib tasvirlash mumkin.

Misollar:

```
short int a = 1;            // a butun o’zgaruvchi
const char C = ' C ';       // c simvol konstanta
char s, sf = ' f ';         // initsializatsiya faqat sf ga taaluqli
char t (54);
float c = 0.22, x(3), sum;
```

Agar initsializatsiyalanadigan qiymat tipi o’zgarishi tipi bilan ustma-ust tushsa ma’lum qoidalar bo’yicha tip almashtiriladi va bajariladi. Xotira tipi va sinfidan tashqari tavsifi oshkora yoki jimlik bo’yicha uning amal qilish sohasini beradi. Xotira sinfi va amal qilish sohasi nafaqat xos tavsifga balki uning dasturi matni joyiga bog’liq.

§1.2. Ma’lumotlar tipi va ma’lumotlar tipiga keltirish

Ma’lumotlar tiplari konsepsiyalari

Ixtiyoriy dasturning asosiy maqsadi ma’lumotlarni qayta ishlashdan iborat. Turli tipdagi ma’lumotlar turlicha saqlanadi va qayta ishlanadi. Ixtiyoriy algoritmik tilda har bir konstanta, o’zgaruvchi, ifoda yoki funksiyani hisoblash natijasi ma’lum tipga ega bo'lishi lozim.

Ma’lumotlar tipi

- kompyuter xotirasidagi ma’lumotlarni ichki tasvirlash
- bu tipdagi miqdorlar qabul qilishi mumkin bo’lgan qiymatlari to’plami
- bu tipdagi miqdorlarga qo’llash mumkin bo’lgan amallar va funksiyalar

Bu xarakteristikalarga asoslanib dasturchi real obyektlarni tasvirlash uchun dasturda foydalaniladigan har bir miqdor tipi tanlanadi. Tipning majburiy tavsifi kompilyatorga dasturning turli konstruksiyalarining yo’l qo’yilishiga tekshirishni amalga oshirishga imkon beradi. Miqdorning tipi ma’lumotlarni qayta ishlash uchun foydalaniladigan mashina buyruqlariga bog’liq.

C++ tilning barcha tiplari asosiy va murakkab turlarga bo'linadi. C++ tilda butun, haqiqiy, simvol va mantiqiy miqdorlarni tasvirlash uchun ma'lumotlarning asosiy yettita tili aniqlangan. Bu tiplar asosida dasturchi murakkab tiplarning tavsifini kiritish mumkin. Ularga massivlar, sanashlar, funksiyalar, strukturalar, murojaatlar, ko'rsatkichlar, birlashmalar va sinflar kiradi.

Ma'lumotlarning asosiy tiplari

Ma'lumotlarning asosiy (standart) tiplari arifmetik deb ataladi, ularning arifmetik amallarda foydalanish mumkin. Asosiy tiplarni tavsiflash uchun quyidagi tayanch so'zlar aniqlangam.

- int (butun)
- char (simvol)
- wchar_t (kengaytirilgan simvol)
- bool (mantiqiy)
- float (haqiqiy)
- double (ikkilanma aniqlikdagi haqiqiy)

Birinchi to'rtta tip butun sonli (butun) deb oxirgi ikkitasi - suzuvchi nuqtali tiplar deb ataladi. Butun miqdorlarni qayta ishlash uchun kompilyator shakllantiradigan kod suzuvchi nuqtali miqdorlar uchun kodlardan farq qiladi.

Standart tiplar ichki tasviri va qiymatlar oralig'ini aniqlashtiruvchi to'rtta tip maxsuslashtiruvchilari mavjud.

- short (qisqa)
- long (uzun)
- signed (ishorali)
- unsigned (ishorasiz)

Butun tip (int)

int tipning o'lchovi standart bilan aniqlanmaydi, kompyuter va kompilyatorga bog'liq 16-razryadli prosessor uchun bu tipdagi miqdorlarga 2 bayt 32-razryadli uchun esa – 4 bayt ajratiladi.

Tip nomi oldidagi **short** maxsuslashtirgich kompilyatorga songa prosessor razryadligiga bog'liq bo'lmagan holda 2 bayt ajratiladi. **long** maxsuslashtirgich

butun miqdor 4 baytni egallashini anglatadi. Shunday qilib, 16-razryadli kompiyuterda **int** va **short int**, 32-razryadlarda esa **int** va **long int** ekvivalent.

Butun tipli miqdorlarni ichki tasvirlash – ikkilik kodda butun son. **signed** maxsuslashtiruvchidan foydalanganda sonning katta biti ishorali sifatida talqin etiladi. (0 – musbat son, 1 - manfiy). **unsigned** mazsuslashtirgich faqat musbat musbat sonlarni tasvirlaydi, chunki yo'qori razryad son kodining qismi sifatida qaraladi. Shunday qilib, **int** tipidagi qiymatlar oralig'i maxsuslashtirgichlarga bog'liq. IBM PC – muvofiqlashgan kompyuterlari uchun turli mahsuslashtirgichli butun tipdagi miqdorlar qiymatlar oraliqlari 1.4- jadvalda keltirilgan.

Jimlik bo'yicha barcha butun sonli tiplar ishorali deb hisoblanadi, ya'ni **signed** maxsuslashtirgichini tushurib qoldirish mumkin.

Dasturda uchraydigan konstantalarga ularning ko'rinishiga mos u yoki bu tip beriladi. Agar bu tip qandaydir sabablarga ko'ra dasturchini qanoatlatirmasa, u talab qilingan tipni **L, l (long)** va **U, u (unsigned)** yordamida oshkora ko'rsatishi mum0kin. Masalan 32L konstanta **long** tipiga ega bo'ladi va 4 baytni egallaydi. **L** va **U** suffikslardan bir vaqtda foydalanish mumkin, 0x22UL yoki 05Lu.

Simvol tip (char)

Simvol tipidagi miqdorga berilgan kompyuter uchun simvollar jamlanmasidan ixtiyoriy simvolni joylashtirish uchun yetarlicha sondagi baytlar ajratiladi va bu tipning nomiga asos bo'lgan. Odatda bu 1 bayt. **char** tip boshqa butun tiplar kabi ishorali yoki ishorasiz bo'lishi mumkin. Ishorali miqdorlarda -128 dan 127 gacha oraliqdagi qiymatlarni saqlash mumkin. **unsigned** maxsuslashtirgichdan foydalanganda qiymatlar 0 dan 255 gacha oraliqda joylashishi mumkin. Bu ASCII 256-simvollar jamlanmasidan ixtiyoriy simvolni saqlash uchun yetarli. **char** tipidagi miqdorlar ko'tsatilgan oraliqlar chegaralaridan oshmaydigan butun sonlarni saqlash uchun ham qo'llaniladi.

Kengaytirilgan simvol tipi (wchar_t)

wchar_t tipi kodlash uchun 1 bayt yetarli bo'lmagan, masalan, Unicode simvollar jamlanmasi bilan ishlash uchun mo'ljallangan. Bu tip o'lchami amalga

oshirilishiga bog'liq, odatda **shord** tipiga mos keladi. **wchar_t** tipidagi satr konstantalar L prefiksi bilan yoziladi. Masalan, L "Gates".

Mantiqiy tip (bool)

Mantiqiy tipdagi miqdorlar faqat **true** va **false** qiymatlari qabul qilishi mumkin, ular zahiralangan so'zlar hisoblanadi. **false** qiymatining tasvirining ichki shakli – 0 (nol). Ixtiyoriy boshqa qiymati **true** kabi tashkil etiladi. Butun tipga almashishda **true** 1 qiymatga ega.

Suzuvchi nuqtali tiplar (float, double va long double)

C++ standart haqiqiy qiymatlarni: **float, double** va **long double** saqlash uchun ma'lumotlar uchta tipini aniqlaydi.

Suzuvchi nuqtali ma'lumotlar tiplari butun sonli qiymatlarga nisbatan aksincha kompyuter xotirasida saqlanadi. IBM PC muvofiqlashgan kompyuterlarda **float** tipidagi miqdorlar 4 baytni egallaydi, ulardan bitta ikkilik razryad mantiqiy ishorasiga ajratiladi, 8 razryatli tartibga va 23 tasi mantissaga ajratiladi. Mantissaning katta raqami 1 ga teng bo'lgani uchun u saqlanmaydi.

8 bayt joy olgan **double** tipidagi miqdorlar uchun tartib va mantissalar mos ravishda, 11 va 52 razryad ajratiladi. Manrissa uzunligi sonning aniqligi, tartib uzunligi – uning oralig'ini aniqlaydi. 1.4 – jadvaldan ko'rinadiki **float** va **long int** tipidagi miqdorlarga ajratilgan bir xil sondagi baytlarda ularning mumkin bo'lgan qiymatlar oraliqlari tasvirlashning ichki shakli tufayli juda katta farq qiladi.

long maxsuslashtirgich double tipi nomi oldiga qo'yilsa, miqdorga 10 bayt ajratilganini ko'rsatadi.

Suzuvchi nuqtali konstantalar double tipiga ega. Konstanta tipini F, f (float) va L, l (long) suffikslar yordamida oshkora ko'rsatish mumkin. Masalan, 2E+6L konstanta long double tipiga ega, 1.82f konstanta – float tipiga ega.

Ma'lumotlar tipini keltirish (Data casting)

Gohida bir turdagi o'zgaruvchining qiymatini boshqa tipdagi o'zgaruvchiga berish kerak bo'ladi. Bu amal ma'lumot tipini keltirish (data type casting) deyiladi.

Ko'p hollarda bu amal avtomatik ravishda, kompilyator tarafidan bajariladi. Masalan ushbu parchani ko'raylik:

```
 char c = 33;

int k;

 k = c;
```

Bu yerda k ning sig'imi c nikidan kattaroqdir. Shuning uchun c ning qiymatini k ga berishda hech qanday muammo paydo bo'lmaydi. Quyidagi misolni ko'raylik:

```
 int i = 5;

float f = 9.77;

float result;

 result = f + i;
```

C++ ning kompilyatori ikki turdagi o'zgaruvchilar bilan ishlay olmaydi. Shu sababli ifodadagi sig'imi kichik bo'lgan o'zgaruvchilar ifodadagi qatnashgan eng katta sig'imga o'tqaziladi. Bu ham avtomatik tarzda bajariladi. i o'zgaruvchimiz qiymati vaqtinchalik **float** tipidagi o'zgaruvchiga beriladi. Bu vaqtinchalik o'zgaruvchi esa f ga qo'shiladi. Chiqqan javob result ga beriladi. Yuqorida ko'rib chiqqanlarimiz kompilyator tarafidan bajariladi. Bu kabi tip o'zgarishlarini avtomatik konversiya(implicit conversion) deymiz. Lekin gohida to'g'ri kelmaydigan tiplarni birga qo'llashga to'g'ri keladi. Masalan **float** tipiga double tipni o'tqazish, char ga int va hokazo. Bu hollarda ochiq konversiya (explicit conversion) amalini bajarishimiz kerak. Buni bajarishning ikki uslubi bor. Birinchisi C da qo'llaniladigan yo'l, ikkinchisi C++ uslubi. C da tipni keltirish uchun o'zgaruvchi oldiga kerakli tipni () qavslar ichida yozamiz.

```
 int k = 100;

char s;
```

s = (char)k;

Yuqorida k ning qiymatini char tipidagi vaqtinchalik o'zgaruvchiga berildi, keyin s ga ushbu o'zgaruvchi qiymatini berildi.Bu yerda etibor berilishi kerak bo'lgan narsa shuki, 100 char ga ham to'g'ri keladi. Agar k ning qiymati char oladigan qiymattan kattaroq/kichikroq bo'ladigan bo'lsa, bu hato olib keladi. Shu sababli C dagi tip keltirish nisbatan havfli hisoblanadi. Lekin albatta bilib qo'llanilsa katta yordam beradi. C++ da ma'lumotlar tipini keltirish uchun mahsus operatorlar kiritildi. C uslubidagi keltirish hamma sharoitda qo'llanilar edi. C++ ning keltirish operatorlari esa faqat o'ziga ajratilgan funksiyalarni bajaradi. Undan tashqari ular C dagi keltirishlardan ko'ra kuchsizroqdir. Shu sababli hato ehtimoli kamaytirildi. Yana bir afzallik tarafi shundaki, yangi stildagi keltirish operatorlari tip tekshirishlarini bajarishadi, agar noto'g'ri keltirish bajarilsa, bu sintaktik hatoga olib keladi.

Ular quyida berilgan:

static_cast

dynamic_cast

const_cast

reinterpret_cast

static_cast ni ko'rib chiqaylik.

```
int k = 200;
char h;
h = static_cast<char>(k);
```

static_cast dan keyin kerakli tip nomi <> qavslar ichida beriladi, va tipi o'zgarishi kerak bo'lgan o'zgaruvchi () qavslar ichida parametr sifatida beriladi.

Static_cast kompilyatsiya davrida tip keltirishlarida qo'llaniladi.**dynamic_cast** esa dastur ishlash davrida tip keltirishlari uchun qo'llaniladi.

const_cast esa o'zgaruvchilardan const (o'zgarmas) va volatile (o'zgaruvchan, uchuvchan) sifatlarini olib tashlashda qo'llaniladi. Odatda const o'zgaruvchining qiymatini o'zgartirib bo'lmaydi. Ushbu holda const_cast qo'llaniladi. **reinterpret_cast** odatiy bo'lmagan keltirishlarni bajarishda qo'llaniladi (masalan void* ni int ga). reinterpret_cast o'zgaruvchining bitlarini boshqa ma'noda qo'llashga imkon beredi. Bu operator bilib ishlatilinishi kerak. Ushbu operatorlar bilan keyinroq yanada yaqin tanishamiz.

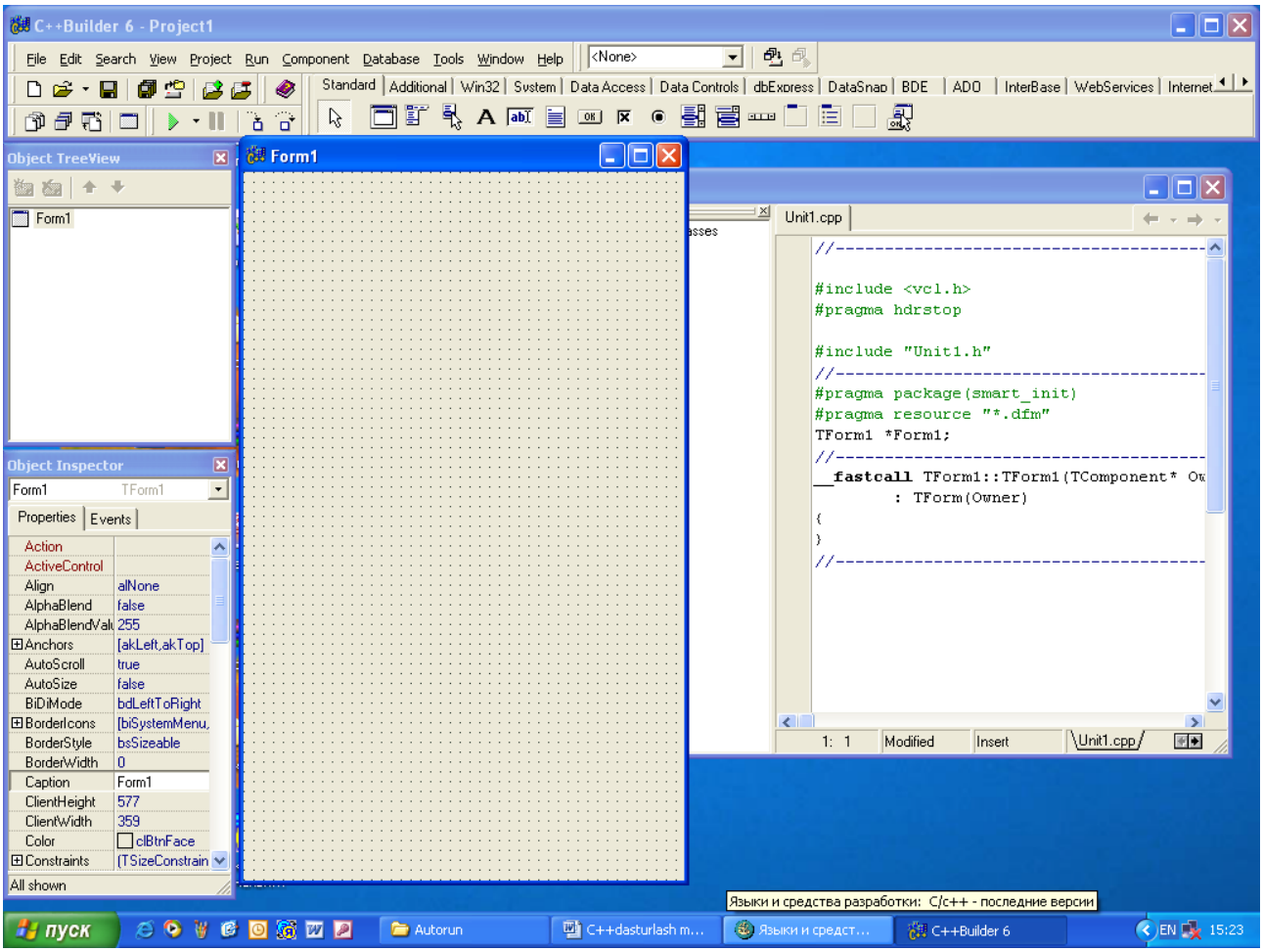
II. C++dasturlash tilida hisoblash kalkulator yaratishning dasturiy ta’minoti

§2.1. C++ dasturlash tilida hisoblash kalkulatori yaratishda foydalanilgan komponentalari

Borland C++Builder 6 dasturlash muhitini ishlatish uchun avval **Windows** OT da quyidagi amalni bajaramiz:

Пуск ⇨ Все программы ⇨ Borland C++Builder 6 ⇨ C++Builder 6.

Bu amalni bajargandan so’ng ekranda muloqot oynasi paydo bo‘ladi. Bu 1 – rasmda **Borland C++Builder 6** muhitining asosiy menyulari, shakllari va loyihasining ko‘rinishi keltirilgan.



Rasmda Borland C++Builder 6 muhitining umumiy ko‘rinishi.

Builder 6 oynasi koʻrinishi odatdagidan ancha boshqacharoq boʻlib, u oʻz ichiga beshta oynani oladi:

- bosh oyna - **Builder 6** Project1;
- forma oynasi - Form1;
- ob'ekt xossalarini taxrirlash oynasi-Object Inspector;
- ob'ektlar roʻyxatini koʻrish oynasi - Object tree View;
- dastur kodlarini tahrirlash oynasi - Unit.cpp.

Bosh oyna ekranning yuqori qismida joylashgan boʻlib, uning birinchi qatorida sarlovha, ya'ni proektning nomi joylashgan. Ikkinchi qatorda buyruqlar menyusi gorizantal koʻrinishda joylashgan. Keyingi qatorning chap tarafida uskunalar paneli va oʻng tarafida komponentalar politrasi joylashgan.

Buyruqlar menyusi quyidagilarni oʻz ichiga olgan:

- -File (fayl) boʻlimi fayllar ustida ish bajarish uchun kerakli buyruqlarni oʻz ichiga olgan;
- -Edit (taxrir) boʻlimi fayl ichidagi ma'lumotlarni taxrirlash uchun kerakli buyruqlarni oʻz ichiga olgan;
- -Seerch
- -View
- -Compile
- -Run formani ishga tushirish.
- -Options
- -Tols servis xizmatidan foydalanish.
- -Help yordam chaqirish.

Forma oynasida ilovalar yaratiladi. Object Inspector oynasi ob'ekt xossalarini taxrirlash uchun xizmat qiladi. Ob'ekt xossalari bu - ob'ektga berilgan xarakteristika boʻlib, uning koʻrinish, joylashishi va holatidir. Masalan, Width va Height xossalari forma oʻlchamini, top va Lift esa formaning ekrandagi holati, Caption - sarlovha matnini aniqlaydi.

Vizual dasturlash texnologiyasida ob’ekt deganda muloqat oynasi va boshqarish elementlari (kiritish va chiqarish maydoni, buyruq tugmalari, pereklyuchatellar va boshqa) tushuniladi.

Builder 6 forma komponentalari

Forma komponentalari bu dasturni boshqarish uchun maxsus tugmachalar bo‘lib uni formaga joylashtirishdan oldin bosh oynadan kerakli komponentalar palitrasi tanlanadi. Masalan, Standart (Standart) komponentalar palitrasida quyidagi piktogrammalar (tugmachalar) majmuasi mavjud:

MainMenu - dastur bosh menyusi. Komponenta murakkab ierarxik strukturali menyu yaratish uchun xizmat qiladi.

PopupMenu - yordamchi yoki lokal menyusi. Bu menyu oynada sichqoncha o‘ng tugmasini bosish bilan chiqadi.

Label - metka (belgi). Bu komponenta forma oynasiga uncha uzun bo‘lmagan bir qatorli yozuvni chiqarishda ishlatiladi va uning piktogrammasi panelda “A” ko‘rinishda berilgan.

Edit - kiritish qatori. Forma oynasida matnli qator kiritish va taxrirlashda ishlatiladi.

Memo - ko‘pqatorli matn muxarriri. Kupqatorli matnlarni kiritish yoki chiqarishda ishlatiladi.

Button - buyruq tugmasi (Obrabotchik sobitiya OnClick). Bu komponenta dasturchi tamonidan berilgan bir necha buyruqlarni bajarishda ishlatiladi.

CheckBox - bog‘liq bo‘lmagan tanlash tugmasi (pereklyuchatel). Dasturda bu komponenta asosiy mantiqiy xossasi (Checked) o‘zgartiriladi.

RadioButton - bog‘liq bo‘lgan tanlash tugmasi (pereklyuchatel). YAngi tutanlash tugmasi bosilganda, oldin tanlangan tugma avtomatik ravishda ozod etadi.

ListBox - ro‘yxatdan tanlash. Ro‘yxat variantlarini taqdim etadi va tanlash imkonini yaratadi.

ComboBox – kiritish qatoriga ega (kombinirovannыy) ro‘yxatdan tanlash. Ro‘yxatdan kombinatsiya qilib tanlash

ScrollBar - yo‘lchali boshqarish. Windows oynasi chetlarida gorizontal yoki vertikal yo‘lcha tashkil etadi.

GroupBox - elementlar guruhi. Ma’no bo‘yicha bir necha bog‘lik komponentalarni gruhlashda ishlatiladi.

RadioGroup - bog‘liq guruhlangan tanlash tugmalari (o‘chirib yoquvchi tugmalar). Bir necha bog‘liq tanlash tugmalari xossalarini saqlaydi.

Panel - panel. Bu komponenta, xuddi GroupBoxga o‘xshab bir necha komponentalarni birlashtirish uchun xizmat qiladi.

Actionlist - ta’sir qilish ro‘yxatlari. Foydalanuvchi dasturga markazlashgan holda ta’sir qilishi uchun ishlatiladi.

§2.2. C++ dasturlash tilida hisoblash kalkulyatori yaratish dasturi

C++ dasturida hisoblash kalkulyatori yaratish uchun Form1 oynasiga BitBtn tugmasi, va StaticText komponentalaridan foydalanamiz. BitBtn , va StaticText komponentalarini Panelda joylashtirib kalkulyatorimizni ko’rinishini tuzamiz.

Birinchi novbatda Form1 oynasiga Panel yaratib olamiz.

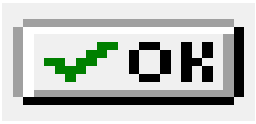
Panel Standart menyusining komponentasi hisoblanadi.

Panelni ichida StaticText va BitBtn komponentalarini joylashtiramiz.

StaticText tugmasi. Edit kiritish qatori matnni bir qatordan kiritish va uni tahrirlash uchun ishlatiladi.

Matn qatoriga kiritilgan ma’lumot faqat matn, ya’ni String (qator) bo‘lib hisoblanadi.

BitBtn komponentasining umumiy ko’rinishi:

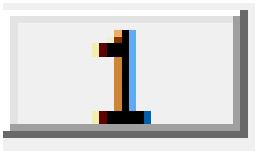


BitBtn komponentasi Button komponentasiga o’xshash buyruqni bajaradi, Buttondan farqi bunda BitBtnni ustiga har xil rasmlar keltirib qo’yishimiz mumkin.

Kalkulatorning “0” tugmasiga quyidagi dastur kiritiladi:

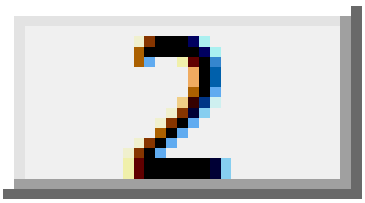
```
//  "0"tugmasiga
void __fastcall TForm1::Btn0Click(TObject *Sender)
{
   if ( f != 0)
        StaticText1->Caption = StaticText1->Caption + "0";
}
```

Kalkulatorning “1” tugmasiga quyidagi dastur kiritiladi:



```
// "1"tugmasiga
void __fastcall TForm1::Btn1Click(TObject *Sender)
{
 if ( f == 0)
   {
     StaticText1->Caption = "1";
     f = 1;
   }
   else
      StaticText1->Caption = StaticText1->Caption + "1";
}
```

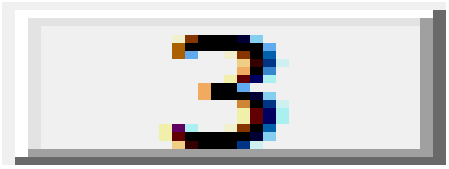
Kalkulatorning “2” tugmasiga quyidagi dastur kiritiladi:



```
// "2"tugmasiga
void __fastcall TForm1::Btn2Click(TObject *Sender)
{
 if ( f == 0)
   {
     StaticText1->Caption = "2";
     f = 1;
   }
```

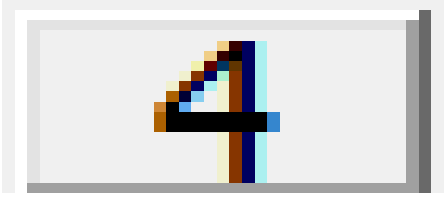
```
    else
        StaticText1->Caption = StaticText1->Caption + "2";
}
```

Kalkulatorning “3” tugmasiga quyidagi dastur kiritiladi:



```
//  "3"tugmasiga
void __fastcall TForm1::Btn3Click(TObject *Sender)
{
    if ( f == 0)
    {
      StaticText1->Caption = "3";
      f = 1;
    }
    else
        StaticText1->Caption = StaticText1->Caption + "3";
}
```

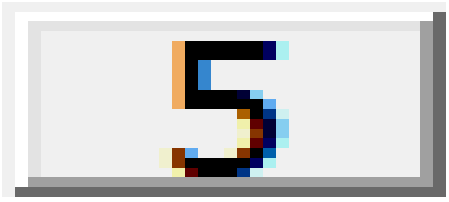
Kalkulatorning “4” tugmasiga quyidagi dastur kiritiladi:



```
//  "4" tugmasiga
void __fastcall TForm1::Btn4Click(TObject *Sender)
{
if ( f == 0)
    {
```

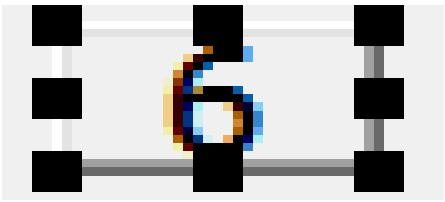
```
      StaticText1->Caption = "4";
      f = 1;
   }
   else
      StaticText1->Caption = StaticText1->Caption + "4";
}
```

Kalkulatorning “5” tugmasiga quyidagi dastur kiritiladi:



```
//  "5" tugmasiga
void __fastcall TForm1::Btn5Click(TObject *Sender)
{
   if ( f == 0)
   {
      StaticText1->Caption = "5";
      f = 1;
   }
   else
      StaticText1->Caption = StaticText1->Caption + "5";
}
```

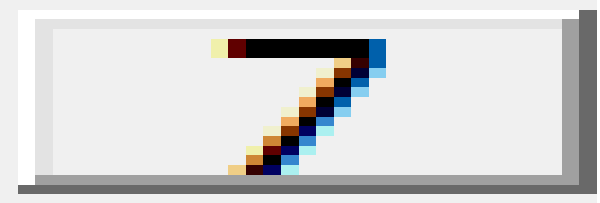
Kalkulatorning “6” tugmasiga quyidagi dastur kiritiladi:



```
//  "6"tugmasiga
```

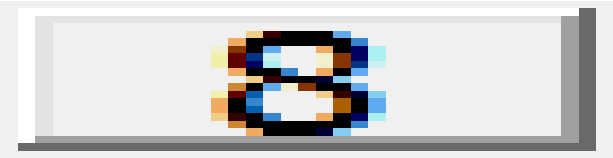
```
void __fastcall TForm1::Btn6Click(TObject *Sender)
{
   if ( f == 0)
   {
      StaticText1->Caption = "6";
      f = 1;
   }
   else
       StaticText1->Caption = StaticText1->Caption + "6";
}
```

Kalkulatorning “7” tugmasiga quyidagi dastur kiritiladi:



```
//  "7" tugmasiga
void __fastcall TForm1::Btn7Click(TObject *Sender)
{
   if ( f == 0)
   {
      StaticText1->Caption = "7";
      f = 1;
   }
   else
       StaticText1->Caption = StaticText1->Caption + "7";
}
```

Kalkulatorning “8” tugmasiga quyidagi dastur kiritiladi:



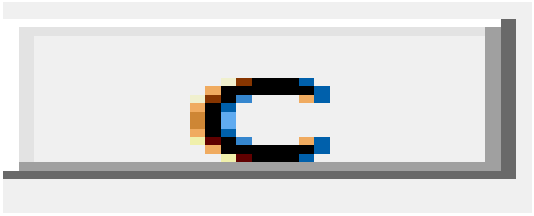
```
//  "8"tugmasiga
void __fastcall TForm1::Btn8Click(TObject *Sender)
{
   if ( f == 0)
   {
      StaticText1->Caption = "8";
      f = 1;
   }
   else
       StaticText1->Caption = StaticText1->Caption + "8";
}
```

Kalkulatorning “9” tugmasiga quyidagi dastur kiritiladi:

```
//  "9" tugmasiga
void __fastcall TForm1::Btn9Click(TObject *Sender)
{
   if ( f == 0)
   {
      StaticText1->Caption = "9";
      f = 1;
   }
   else
```

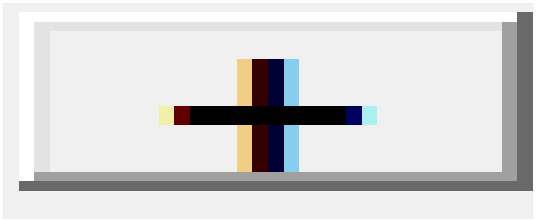
```
        StaticText1->Caption = StaticText1->Caption + "9";
}
```

Kalkulatorning “c” tugmasiga quyidagi dastur kiritiladi:



```
//  "c" tugmasiga
void __fastcall TForm1::BtnCClick(TObject *Sender)
{
    StaticText1->Caption = "0";
    accum = 0;
    op = 0;
    f = 0;
}
```

Kalkulatorning “+” tugmasiga quyidagi dastur kiritiladi:



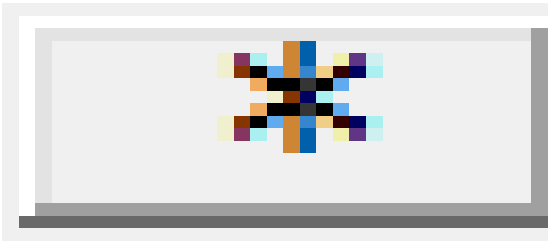
```
//  "+" tugmasiga
void __fastcall TForm1::BtnPClick(TObject *Sender)
{
if ( f != 0)
    {
DoOp();
        f = 0;
    }
```

```
    op =1;
}
```

Kalkulatorning “-” tugmasiga quyidagi dastur kiritiladi:

```
//  "-" tugmasiga
void __fastcall TForm1::BtnMClick(TObject *Sender)
{
    if ( f != 0)
    {
DoOp();
        f = 0;
    }
    op = 2;
}
```

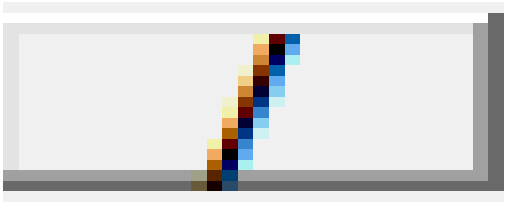
Kalkulatorning “*” tugmasiga quyidagi dastur kiritiladi:



```
// "*" tugmasi
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
 if ( f != 0)
    {
DoOp();
```

```
        f = 0;
    }
    op =3;
}
```

Kalkulatorning “/” tugmasiga quyidagi dastur kiritiladi:



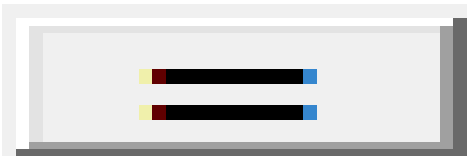
```
// "/" tugmasi
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
 if ( f != 0)
    {
DoOp();
        f = 0;
    }
    op =4;
}
```

Kalkulatorning “,” tugmasiga quyidagi dastur kiritiladi:

```
//  "," tugmasiga
void __fastcall TForm1::BtnkClick(TObject *Sender)
{
    if ( f == 0)
```

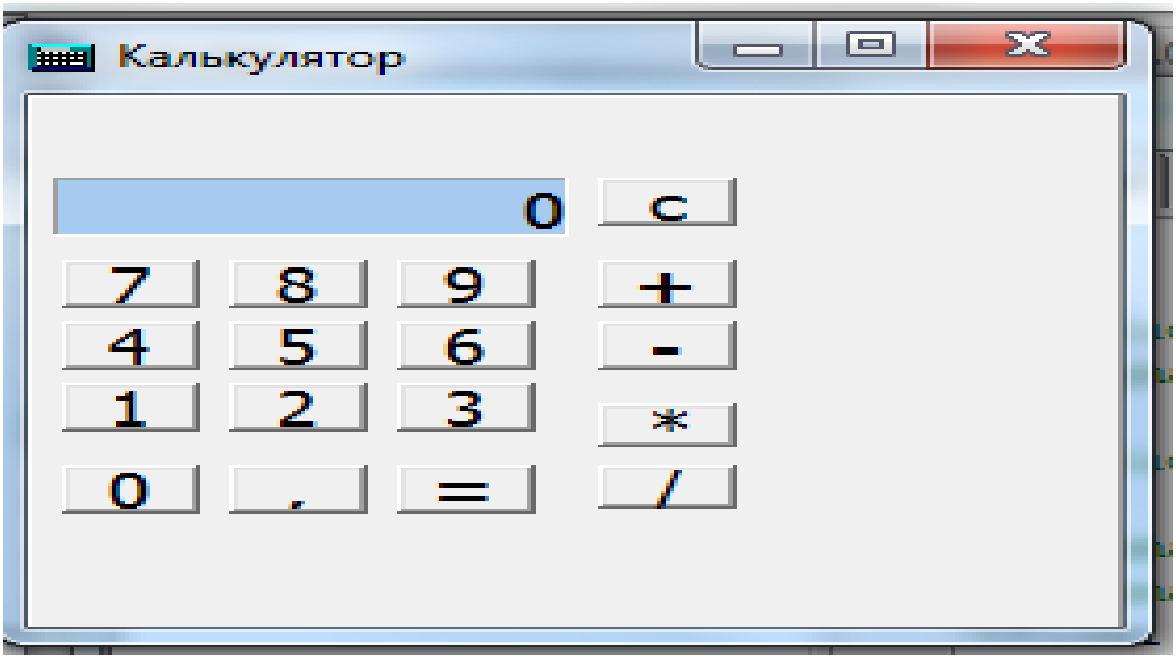
```
    {      StaticText1->Caption = "0,";
      f = 1;    }
    else
    {
       if ( StaticText1->Caption.Pos(",") == 0)
          StaticText1->Caption = StaticText1->Caption + ",";
    }
}
```

Kalkulatorning “=” tugmasiga quyidagi dastur kiritiladi:



```
//  "="tugmasiga
void __fastcall TForm1::BtnEClick(TObject *Sender)
{
    if ( f != 0)
    {
       DoOp();
       f = 0;
    }
    op = 0;
}
```

Shundan so’ng Formaning umumiy ko’rinishi quyidagicha bo’ladi:



Dasturning umumiy kodi quyidagicha bo'ladi:

```
#include <vcl.h>
#pragma hdrstop
#include "CalcForm.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
float accum;
int op;
int f;
_fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    f =0;
    op = 0;
    StaticText1->Caption = 0;
}
//  "0"tugmasiga
```

```
void __fastcall TForm1::Btn0Click(TObject *Sender)
{
   if ( f != 0)
        StaticText1->Caption = StaticText1->Caption + "0";
}
// "1"tugmasiga
void __fastcall TForm1::Btn1Click(TObject *Sender)
{
 if ( f == 0)
   {
     StaticText1->Caption = "1";
     f = 1;
   }
   else
      StaticText1->Caption = StaticText1->Caption + "1";
}
// "2"tugmasiga
void __fastcall TForm1::Btn2Click(TObject *Sender)
{
 if ( f == 0)
   {
     StaticText1->Caption = "2";
     f = 1;
   }
   else
      StaticText1->Caption = StaticText1->Caption + "2";
}
//  "3"tugmasiga
```

```
void __fastcall TForm1::Btn3Click(TObject *Sender)
{
   if ( f == 0)
   {
      StaticText1->Caption = "3";
      f = 1;
   }
   else
       StaticText1->Caption = StaticText1->Caption + "3";
}
//  "4" tugmasiga
void __fastcall TForm1::Btn4Click(TObject *Sender)
{
if ( f == 0)
   {
      StaticText1->Caption = "4";
      f = 1;
   }
   else
       StaticText1->Caption = StaticText1->Caption + "4";
}
//  "5" tugmasiga
void __fastcall TForm1::Btn5Click(TObject *Sender)
{
   if ( f == 0)
   {
      StaticText1->Caption = "5";
      f = 1;
```

```
   }
   else
      StaticText1->Caption = StaticText1->Caption + "5";
}
//  "6"tugmasiga
void __fastcall TForm1::Btn6Click(TObject *Sender)
{
   if ( f == 0)
   {
     StaticText1->Caption = "6";
     f = 1;
   }
   else
      StaticText1->Caption = StaticText1->Caption + "6";
}
//  "7" tugmasiga
void __fastcall TForm1::Btn7Click(TObject *Sender)
{
   if ( f == 0)
   {
     StaticText1->Caption = "7";
     f = 1;
   }
   else
      StaticText1->Caption = StaticText1->Caption + "7";
}
//  "8"tugmasiga
void __fastcall TForm1::Btn8Click(TObject *Sender)
```

```
{
   if ( f == 0)
   {
      StaticText1->Caption = "8";
      f = 1;
   }
   else
       StaticText1->Caption = StaticText1->Caption + "8";
}
//  "9" tugmasiga
void __fastcall TForm1::Btn9Click(TObject *Sender)
{
   if ( f == 0)
   {
      StaticText1->Caption = "9";
      f = 1;
   }
   else
       StaticText1->Caption = StaticText1->Caption + "9";
}
//  "," tugmasiga
void __fastcall TForm1::BtnkClick(TObject *Sender)
{
   if ( f == 0)
   {
      StaticText1->Caption = "0,";
      f = 1;
   }
```

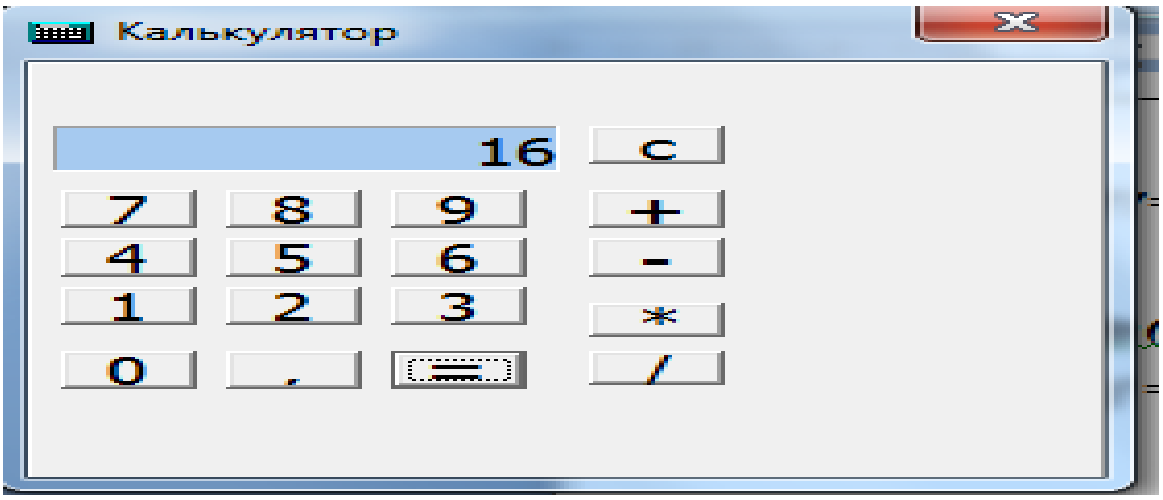
```
    else
    {
        if ( StaticText1->Caption.Pos(",") == 0)
            StaticText1->Caption = StaticText1->Caption + ",";
    }
}
//  "Ñ" tugmasiga
void __fastcall TForm1::BtnCClick(TObject *Sender)
{
    StaticText1->Caption = "0";
    accum = 0;
    op = 0;
    f = 0;
}
void __fastcall TForm1::DoOp(void)
{
float op4 =  StrToFloat(StaticText1->Caption);
    switch ( op )
    {
        case 0 : accum = op4;  break;
        case 1 : accum += op4; break;
        case 2 : accum -= op4; break;
         case 3 : accum *= op4; break;
          case 4 : accum /= op4; break;
    }
    StaticText1->Caption = FloatToStrF(accum,ffGeneral,6,3);
}
//  "+" tugmasiga
```

```
void __fastcall TForm1::BtnPClick(TObject *Sender)
{
   if ( f != 0)
   {
 DoOp();
      f = 0;
   }
   op =1;
}
//  "-" tugmasiga
void __fastcall TForm1::BtnMClick(TObject *Sender)
{
   if ( f != 0)
   {
DoOp();
      f = 0;
   }
   op = 2;
}
//  "="tugmasiga
void __fastcall TForm1::BtnEClick(TObject *Sender)
{
   if ( f != 0)
   {
      DoOp();
      f = 0;
   }
   op = 0;
```

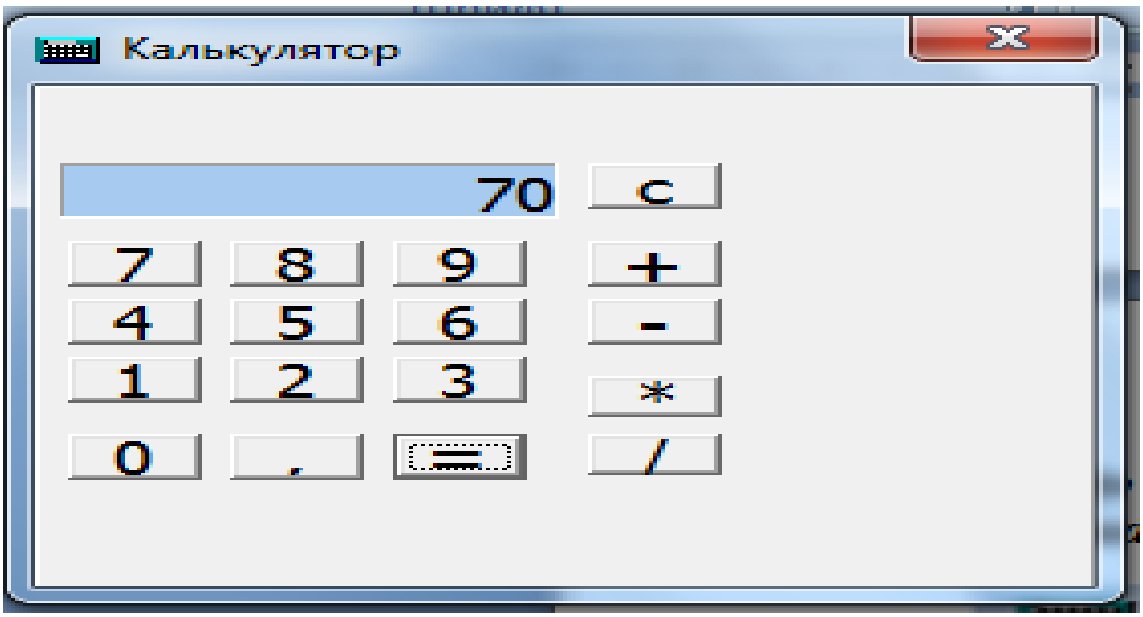
```
}
    // "*" tugmasi
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
 if ( f != 0)
   {
DoOp();
       f = 0;
   }
   op =3;
}
   // "/" tugmasi
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
 if ( f != 0)
   {
DoOp();
       f = 0;
   }
   op =4;
}
```

Kalkulyatorimizni ishlashini tekshirib olamiz:

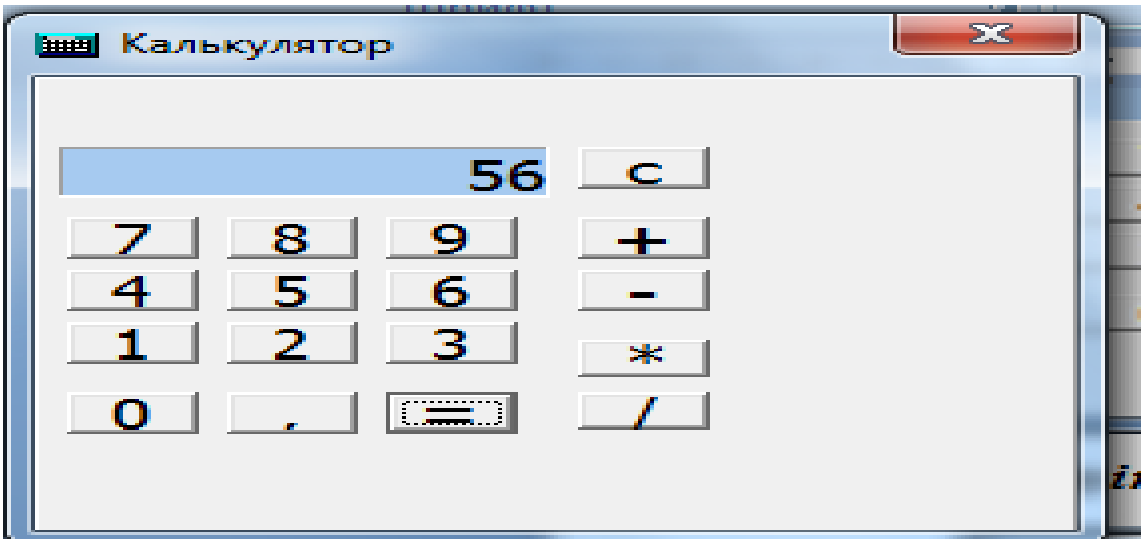
1). Qo’shish amaliga 7 + 9 sonini kiritganimizda quyidagicha oyna hosil bo’ladi



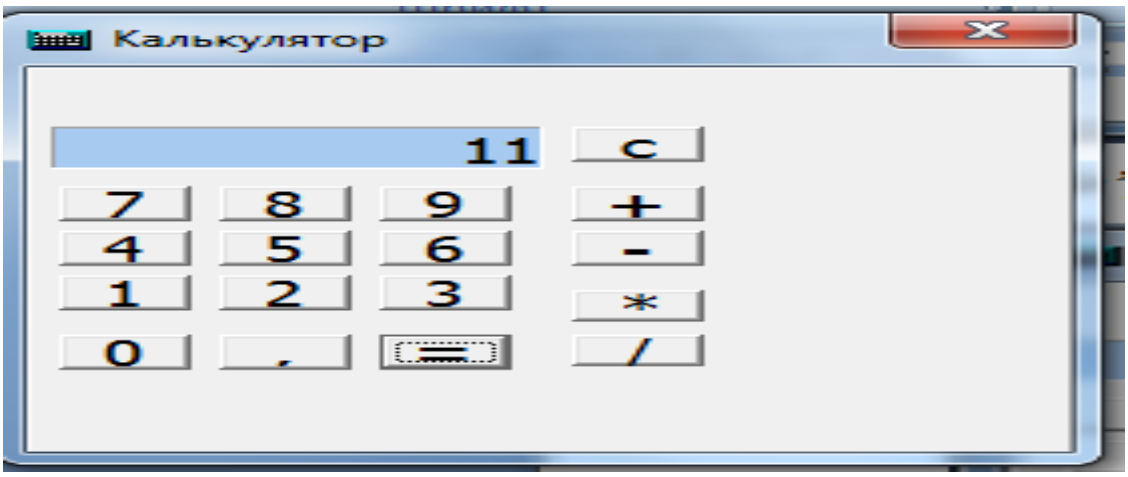
2). Ayirish amaliga 77-7 kiritganimizda quyidagi oyna hosil bo'ladi



*3). Ko'paytirish amaliga 7, * va 8 tugmalarini bosganimizda quyidagi oyna hosil bo'ldi*



4). Bo'lish amaliga 77/7 kiritganimizda quyidagi natija hosil bo'ldi


Калькулятор
11
C
7
8
9
+
4
5
6
-
1
2
3
*
0
,
=
/

Xulosa

Xulosa qilib aytganda men ushbu “**kurs ishi**” tayyorlash davomida quyidagi xulosaga keldim, kalkulyator yaratish uchun C++ Builder muhiti haqida yetarli ma’lumotga ega bo’ldim. Formaga kalkulyator bilan ishlashda kerak bo’ladigan komponentalarni joylashni va har bir komponentaning tegishli kodlarini kiritib chiqishni o’rgandim.

Bu dasturda nafaqat kalkulyatorda balki hisoblashni amalga oshirish dasturini bilmaydigan foydalanuvchi ham bir ko’rinishda tushinadigan tushunarli ko’rinishga ega dastur tuzdim.

Men bu dasturni tuzib kalkulyator yaratishni va ular orqali shunga o’xshash yana boshqa hisoblash tizimlarini ham yaratishni maqsad qildim, bu dasturda ishlashimga yana bir sabab boshqa obektlarga mo’ljallanmagan dasturlash tillarida faqat masalani matematik dasturi tuzib natija olinadi bunda ya’ni C++ Builderda esa boshqa imkoniyatlar va komponentalar bilan ham ishlashni o’rgandim.

Foydalanilgan adabiyotlar ro’yhati

1. Страуструп Б. Язык программирования C++. Третье издание, M.:

Бином, 1999.

2. Шмидский Я.К.Программирование на языке C++: Самоучитель. Учебное пособие. Диалектика. 361 стр, 2004 г.

3. Q.Abduraximov C++ Dasturlash asoslari 2014

4. Ашарина Н.А. Основы программирования на языках Си,C++. Учебный

курс.М.: 2002 г.

5. Подбельский В.В. Язык C++ М.: Финансы и статистика, 1996.

6. www.tammi.uz

7. www.ziyonet.uz

8. www.google.uz qidiruv sayti