O’ZBEKISTON RESPUBLIKASI

OLIY VA O’RTA MAXSUS TA’LIM VAZIRLIGI

A.NAVOI NOMIDAGI SAMARQAND DAVLAT UNIVERSITETI

MEXANIKA-MATEMATIKA FAKULTETI

INFORMATIKA O’QITISH METODIKASI YO’NALISHI

ALGORITIM VA DASTURLASH TILLARI FANIDAN

# KURS ISHI

***Mavzu:*** C++ tilida Satrlar va funksiyalarga doir masalalar yechish


**Talaba:Jaxparova Sh**

**Guruh raxbari:Nazarov F**

**SAMARKAND – 2016**

## Mundarija

## Kirish

Vaqt o‘tishi bilan dasturchilar oldiga quyilgan masalalar o‘zgarib boryapti. Bundan yigirma yil oldin dasturlar katta hajmdagi ma’lumotlarni qayta ishlash uchun tuzilar edi. Bunda dasturni yozuvchi ham, uning foydalanuvchisi ham kompyuter sohasidagi bilimlar bo‘yicha professional bo‘lishi talab etilardi. Hozirda esa ko‘pgina o‘zgarishlar ro‘y berdi. Kompyuter bilan ko‘proq uning apparat va dasturiy ta’minoti, haqida tushunchalarga ega bo‘lmagan kishilar ishlashyapti. Kompyuter odamlar tomonidan uni, chuqur o‘rganish vositasi emas, ko‘proq o‘zlarining oldilariga qo‘yilgan, o‘zlarining ishlariga tegishli bo‘lgan muammolarini echish instrumenti bo‘lib qoldi.

Foydalanuvchilarning ushbu yangi avlodini dasturlar bilan ishlashlarini osonlashtirilishi bilan bu dasturlarning o‘zini murakkabligi darajasi oshadi. Zamonaviy dasturlar - foydalanuvchi bilan do‘stona munosabatni yuqori darajada tashkil qiladigan ko‘p sondagi oynalar, menyu, muloqot oynalari va vizual grafikaviy muhitlardan tarkib topgan interfeysga ega bo‘lishi lozim.Dastur so‘zi ham komandalarning alohida blokini (berilgan kodini) aniqlovchi so‘z, ham yaxlit holdagi bajariluvchi dasturiy mahsulotni belgilovchi so‘z sifatida ishlatiladi. Dasturlashga talabni o‘zgarishi nafaqat tillarning o‘zgarishiga balki uni yozish texnologiyasini ham o‘zgarishiga olib keldi. Dasturlash evolyusiyasi tarixida ko‘pgina bosqichlar bo‘lishiga qaramay biz bu kursimizda protsedurali dasturlashdan ob’ektlarga mo‘ljallangan dasturlashga o‘tishni qaraymiz.Keyingi yillarda amaliy dasturchilarga juda ko‘p integratsion dastur tuzish muhitlari taklif etilayapti. Bu muhitlar u yoki bu imkoniyatlari bilan bir–biridan farq qiladi. Aksariyat dasturlashtirish muhitlarining fundamental asosi C++ tiliga borib taqaladi.

## I Bob. C++ da Satrlar va funksiyalarga doir masalalar yechish.

### 1.1 Funksiyalar tasnifi.

C++ tilida tuzilgan dastur ob’ektlar, funksiyalar, o‘zgaruvchilar va boshqa elementlardan tashkil topadi. Ushbu mavzuning asosiy qismi ularning har birini to‘liq tavsiflashga bag‘ishlangan. Lekin bu elementlarni uyg‘unlashgan holda qarash uchun biror bir tugallangan ishchi dasturni qarab chiqish kerak.Dasturda birinchi funta (#) belgisi joylashgan. U prepotsessorga signal uzatadi. Kompilyatorning har safar ishga tushirilishida preprotsessor ham ishga tushiriladi. U dasturdagi funta (#) belgisi bilan boshlanuvchi qatorlarni o‘qiydi. include - preprotsessorning koman`dasi bo‘lib, u quyidagicha tarjima qilinadi: «Bu komandani ortidan fayl nomi keladi. Ushbu nomdagi faylni topish va fayldagi mazmunni dasturning joriy kismiga yozish lozim». Burchakli qavs ichidagi faylni mos fayllar joylashtirilgan barcha papkalardan izlash lozimligini ko‘rsatadi. Agarda kompilyator to‘g‘ri sozlangan bo‘lsa burchakli kavslar iostream.h faylini sizning kompilyatoringiz uchun muljallangan .h kengaytmali fayllarni o‘zida saqlovchi papkadan izlashi kerakligini ko‘rsatadi. iostream.h ( input – output stream – kiritish–chiqarish oqimi) faylida ekranga ma’lumotlarni chiqarish jarayonini ta’minlaydigan cout ob’ekti aniqlangan. Birinchi qator bajarilgandan so‘ng iostream.h fayli joriy dasturga xuddi uning mazmunini qo‘l bilan yozganimizdek biriktiriladi. Preprotsessor kompilyatordan keyin yuklanadi va funt (#) belgii bilan boshlanuvchi barcha qatorlarni bajaradi, dastur kodlarini kompilyasiyaga tayyorlaydi.Dasturning asosiy kodi main() funksiyasini chaqirish bilan boshlanadi. ‘C++ tilidagi har bir dastur main() fuksiyasini o‘zida saqlaydi. Funksiya bu bir yoki bir necha amalni bajaruvchi dastur blokidir. Odatda funksiyalar boshqa funksiyalar orqali chaqiriladi, lekin main() funksiyasi alohida xususiyatga ega bo‘lib u dastur ishga tushirilishi bilan avtomatik tarzda chaqiriladi.main() funksiyasini boshqa funksiyalar kabi qaytaradigan qiymati tipini e’lon qilish lozim. SALOM.cpp dasturida main() funksiyasi int (integer – butun so‘zidan olingan) tipli qiymat

qaytaradi, ya’ni bu funksiya ishini tugatgandan so‘ng operatsion sistemaga butun sonli qiymat qaytaradi. Operatsion sistemaga qiymat qaytarish unchalik muhim emas, umuman sistema bu qiymatdan foydalanmaydi, lekin ‘C++ tili standarti main() funksiyasi barcha qoidalarga muvofiq e’lon qilinishini talab qiladi.Ishorali va ishorasiz tiplar. Dasturda qo‘llaniladigan butun sonli tiplar ishorali va ishorasiz bo‘lishi mumkin. Ba’zan o‘zgaruvchi uchun faqatgina musbat sonni qo‘llash foydali bo‘ladi. **Unsingned** kalitli so‘ziciz keltirilgan butun sonli tiplar (**short** va **long**) ishorali hisoblanadi. Ishorali butun sonlar manfiy va musbat bo‘lishi mumkin. Ishorasiz sonlar esa doimo musbat bo‘ladi.O‘zgaruvchilarning tayanch tiplari.

‘C++ tilida boshqa berilganlar tiplari ham qaralgan. Ular butun sonli, haqiqiy va belgili bo‘lishi mumkin. Haqiqiy o‘zgaruvchilar kasr ko‘rinishda ifodalanuvchi qiymatlarni ham o‘zida saqlaydi. Belgili o‘zgaruvchilar bir bayt joy egallaydi va 266 ta belgi hamda ASCII belgilarni saqlash uchun ishlatiladi.ASCII belgilari deganda kompyuterlarda qo‘llaniladigan standart belgilar to‘plami tushuniladi. ASCII - bu American Standard Code for Information Interchange ( Amerikaning axborot almashinishi uchun standart kodi) degan ma’noni anglatadi.Kalit so‘zlar. ‘C++ tilida ayrim so‘zlar oldindan zahiralanadi. Bular kalitli so‘zlar deb aytiladi. Bunday so‘zlarni o‘zgaruvchilarni nomlashda ishlatish mumkin emas. Ularga if, while, for va main kabi so‘zlar kiradi. Kompilyatorning texnik dokumentatsiyasida barcha zahiralangan so‘zlarning ruyxati turadi.O‘zgaruvchiga qiymat berish.

O‘zgaruvchilarga qiymat berish uchun o‘zlashtirish operatori qo‘llaniladi. Masalan, Width o‘zgaruvchisiga 5 qiymatni berish uchun quyidagilarni yozish lozim:

```
unsigned short Width;
Width = 5;
```

Bu ikkala satrni Width o‘zgaruvchisini aniqlash jarayonida birgalikda yozish mumkin.

```
unsigned short Wigth = 5;
```

Bir necha  o‘zgaruvchilarni aniqlash vaqtida ham ularga qiymat berish mumkin:

Long width = 5,  length = 7;

O‘zgarmaslar. O‘zgaruvchilar kabi o‘zgarmaslar ham ma’lumotlarni saqlash uchun mo‘ljallangan xotira yacheykalarini o‘zida ifodalaydi. O‘zgaruvchilardan farqli ravishda ular dasturni bajarilishi jarayonida qiymati o‘zgarmaydi. O‘zgarmas e’lon qilinishi bilan unga qiymat berish lozim,  keyinchalik bu qiymatni o‘zgartirib bo‘lmaydi.

‘C++ tilida ikki turdagi, literal va belgili o‘zgarmaslar aniqlangan.

Literal o‘zgarmaslar.  Literalli o‘zgarmaslar to‘g‘ridan-to‘g‘ri dasturga kiritiladi. Masalan:

Int myAge =39;

Bu ifodada MyAge int tipidagi o‘zgaruvchi,  39 soni esa literal o‘zgarmasdir.Belgili o‘zgarmas – bu nomga ega bo‘lgan o‘zgarmasdir. ‘C++ tilida belgili o‘zgarmasni aniqlashning ikki usuli mavjud:

1. # define  direktivasi yordamida o‘zgarmasni aniqlash.
2. const kalitli so‘zi orqali o‘zgarmasni aniqlash.

An’anaviy usul hisoblangan #define direktivasi orqali o‘zgarmasni aniqlashni quyidagi misolda ko‘rishimiz mumkin.

#define StudentsPerClass 15

Bu holda StudentsPerClass o‘zgarmas hech qanday tipga tegishli bo‘lmaydi.

Preprotsessor StudentsPerClass so‘ziga duch kelganida uni 15 literaliga almashtiradi.

‘C++ tilida  #define direktivasidan tashqari  o‘zgarmasni aniqlashning nisbatan qulayroq bo‘lgan yangi usuli ham mavjud:

const unsigned short int StudentsPerClass=15

Bu misolda ham belgili konstanta StudentsPerClass nomi bilan aniqlanayapti va unga unsigned short int tipi berilyapti. Bu usul bir qancha imkoniyatlarga ega bo‘lib u sizning dasturingizni keyingi himoyasini engillashtiradi. Bu o‘zgarmasni oldingisidan eng muhim afzalligi uning tipga egaligidir.

Belgili o‘zgarmaslarni literal o‘zgarmaslarga nisbatan ishlatish qulayroqdir. CHunki agarda bir xil nomli literalli o‘zgaruvchini qiymatini o‘zgartirmoqchi bo‘lsangiz butun dastur bo‘yicha uni o‘zgartirishga to‘g‘ri keladi, belgili o‘zgarmaslarni esa faqatgina birining qiymatini o‘zgartirish etarli.

**1.2.Simvollar va satrlar bilan ishlash.**

Belgilar. Belgili o‘zgaruvchilar odatda bir bayt joyni egallaydi va bu 256 xil belgini saqlash uchun etarlidir. Char tipi qiymatlarini 0..255 sonlar to‘plamiga yoki ASCII belgilar to‘plamiga interpretatsiya qilish mumkin. <string.h> C uslubida satrlar bilan ishlovchi funksiyalar e’loni berilgan.

Maxsus belgilar. ‘C++ kompilyatori tekstlarni formatlovchi bir nechta maxsus belgilardan tashkil topgan. (Ulardan eng ko‘p tarqalgani 3.2. - jadvalda keltirilgan). Bu belgilarni dasturda ishlatishda «teskari slesh»dan foydalanamiz. Teskari sleshdan keyin boshqaruvchi belgi yoziladi. Masalan, tabulyasiya belgiini dasturga qo‘yish uchun quyidagicha yozuvni yozish kerak.

Char tab =‘\t’;

Bu misoldagi char tipidagi o‘zgaruvchi \t qiymatini qabul qiladi. Maxsus belgilar axborotlarni ekranga, faylga va boshqa chiqarish qurilmalariga chiqarishda formatlash uchun qo‘llaniladi.

‘C++ da belgili ma’lumotlar uchun char turi qabul qilingan. Beligili axborotni taqdim etishda belgilar, simvolli o‘zgaruvchilar va matniy konstantalar qabul qilingan. C++ da satrlar (string) qo‘shtirnoqlar (") orasida bo‘ladi. Bitta harfli literalar

esa bitta tirnoq - apostrof (‘) ichiga olinadi. Misol uchun: 'A', '$'. Bitta harf yoki belgini qo‘shtirnoq ichiga olsa u satr kabi qabul qilinadi.

Misollar:

**sonst char c=’c’;** //belgi - bir baytni egallaydi, uning qiymati o‘zgarmaydi.

**char a,b;** //belgili o‘zgaruvchilar, bir baytdan joy egallaydi, qiymatlari o‘zgaradi.

**const char *s= ‘‘\n satrining misoli’’;** //matniy konstanta

‘C++ dagi satr - bu nul-belgi - ‘\0’ (nul-terminator) - bilan tugallanuvchi belgilar massivi. Nul-terminatorning holatiga qarab satrning amaldagi uzunligi aniqlanadi. Bunday massivdagi elementlar soni, satr tasviriga qaraganda, bittaga ko‘p.

Qiymat berish operatori yordamida satrga qiymat berish mumkin emas. Satrni massivga yoki kiritish paytida yoki nomlantirish yordamida joylashtirish mumkin.

**Misol:**

```
void main()
{
    char s1[10]=’’string1’’;
    int k=sizeof (s1);
    cout<<s1<<’’\t’’<<k<<endl;
    char s2[]=’’string2’’;
    k=sizeof(s2);
    cout<<s2<<’’\t’’<<k<<endl;
```

```
    char s3[]={‘s’,’t’,’r’,’i’,’n’,’g’,’3’};
    k=sizeof(s3);
    cout<<s3<<’’\t’’<<k<<endl;
    char *s4=’’string4’’;//satr ko‘rsatkichi, uni o‘zgartirib bo‘lmaydi
    k=sizeof(s4);
    cout<<s4<<’’\t’’<<k<<endl;
}
```

**Natijalar:**

**string1 10 - 10 bayt ajratilgan, shu jumladan \0 ga**

**string2 8 - 8 bayt ajratilgan (7+1 bayt /0 ga)**

**string3 8 - 8 bayt ajratilgan (7+1 bayt /0 ga)**

**string4 4 - ko‘rsatkichning o‘lchamlari**

Satrlar, yani harflar ketma-ketligi ("Toshkent", "Yangi yilingiz bilan!"...) C/C++ da char tipidagi massivlar yordamida beriladi. Bunday satrlar bilan islovlar juda tez bajariladi. CHunki ortiqcha tekshirishlar bajarilmaydi. Bundan tashqari C++ da ancha rivojlangan String klasi mavjuddir, u oddiy char bilan berilgan satrlardan ko‘ra qulayroqdir. Lekin ushbu klas ko‘proq joy egallaydi va massivli satrlardan ko‘ra sekinroq ishlaydi. String klasini keyingi qismlarda o‘tamiz. Qolaversa, satrlar bilan ishlash uchun biz o‘zimiz klas yoki struktura yozishimiz mumkin. C dan meros bo‘lib qolgan satrlar ustida amallar bajarish uchun biz dasturimizga **<string.h>** (yangi ismi <cstring>) e’lon faylini kiritishimiz kerak. Ushbu e’lon faylida berilgan funksiyalar bilan keyingi bo‘limda ishlaymiz. Harflar, yani literalar, aytib o‘tganimizdek, C++ da

char tipi orqali beriladi. Literalar apostroflarga ('S', '*' ...) olinadi. Satrlar esa qo‘shtirnoqlarga olinadi. Satrlar e’loniga misol beraylik.

**char string[] = "Malibu";**

**char *p = "Ikkinchi!?!";**

Satrlarni yuqoridagi ikkita yo‘l bilan initsalizatsiya qilsa bo‘ladi. Satrlar ikkinchi uslubda e’lon qilinganda, yani pointer mehanizmi qo‘llanilganda, ba’zi bir kompilyatorlar satrlarni hotiraning konstantalar saqlanadigan qismiga qo‘yadi. YAni ushbu satrlarga o‘zgartirish kiritilishi ta’qiqlanadi. SHu sababli satrlarni hardoim o‘zgartirish imkoni bo‘lishi uchun ularni char tipidagi massivlar ko‘rinishida e’lon qilish afzaldir. Satrlar massiv yordamida berilganda, ularning uzunligi noma’lumdir. SHu sababli satrning tugaganligini bildirish uchun satr ohiriga mahsus belgi nol literasi qo‘yiladi. Uning dastursa belgilanishi '\0' ko‘rinishga ega. Demak, yuqorida berilgan "Malibu" satriga ohiriga '\0' belgisi qo‘shiladi, yani massivning umumiy uzunligi "Malibu":6 + '\0':1 = 7 = 7 ta char tipidagi elementga teng bo‘ladi. Satrlarni massiv initsalizatsiya ro‘yhati ko‘rinishida ham bersak bo‘ladi:

**char c[6] = {'A', 'B', 'C', 'D', 'E' , '\0'};**

**...**

**cout << c;**

**...**

***Ekranda:***

**ABCDE**

**Funksiyaga ko‘p o‘lchamli massivlarni uzatish.** Ko‘p o‘lchamli massivlarni funksiyaga uzatishda barcha o‘lchamlar parametrlar sifatida uzatilishi kerak. C va C++ da ko‘p o‘lchamli massivlar aniqlanishi bo‘yicha mavjud emas. Agar biz bir

nechta indeksga ega bo‘lgan massivni tavsiflasak (masalan, int mas[3][4]), bu degani, biz bir o‘lchamli mas massivini tavsifladik, bir o‘lchamli int [4] massivlarining ko‘rsatkichlari esa uning elementlaridir

Misol: Kvadrat matritsani uzatish (transportirovka qilish)

Agar void transp(int a[ ][ ],int n){.....} funksiyasining sarlavhasini aniqlasak, bu holda biz funksiyaga noma’lum o‘lchamdagi massivni uzatishni xohlagan bo‘lib qolamiz. Aniqlanishiga ko‘ra massiv bir o‘lchamli bo‘lishi kerak, hamda uning elementlari bir xil uzunlikda bo‘lishi kerak. Massivni uzatishda uning eelementlarining o‘lchamlari haqida ham biron narsa deyilmagan, shuning uchun kompilyator xato chiqarib beradi.

**C++ da funktsiyalar**

Dastur xajmining ko‘payishi bilan uning xotirasida hamma detallarni saqlab turish imkoni qiyinlashadi. Dasturni soddalashtirish uchun, u qismlarga bo‘linadi. C++ da masala funktsiyalar yordamida soddaroq masalachalarga bo‘linishi mumkin. SHuningdek masalaning funktsiyalarga bo‘linishi kodning ortiqchaliligini bartaraf etish imkonini ham beradi, chunki funktsiya bir marta yoziladi, ko‘p marta chaqiriladi. Tarkibida funktsiya bo‘lgan dasturni sozlash oson bo‘ladi.

Ko‘pincha qo‘llanayotgan funktsiyalarni kutubxonalarga joylashtirish mumkin. Shunday qilib sozlashda va kuzatib borishda ancha sodda dasturlar yaratiladi.

## II.C++ da Satrlar va funksiyalarga doir masalalar yechish dasturiy ta’minoti;

### 2.1.Funktsiyalarni e’lon qilish va aniqlash.

Funktsiya - bu tavsiflar va operatorlarning nomlangan ketma-ketligi bo‘lib, tugallangan xatti-harakatlarni, masalan, massivni shakllantirish, massivni bosib chiqarish va h.k. larni bajaradi.

Funktsiya, birinchidan, C++ ning xosila turlaridan biri, ikkinchidan esa, minimal bajarilayotgan dastur moduli hisoblanadi.

Har qanday funktsiya e’lon qilinishi va aniqlanishi kerak.

Funktsiyani e’lon qilishda (prototip, sarlavha) unga nom, qaytarilayotgan qiymat turi va uzatilayotgan parametrlar ro‘yxati beriladi.

Funktsiyaning aniqlanishi, e’londan tashqari, yana tavsiflar va operatorlar ketma-ketligidan iborat funktsiya tanasini bildiradi.

Funktsiyaning__tur nomi([formal__parmetrlar__ro‘yxati])

(funktsiya__tanasi)

Funktsiya__ tanasi bu blok yoki tarkibli operatordir. Funktsiya ichida boshqa funktsiyani aniqlash mumkin emas. Funktsiya tanasida funktsiyaning olingan

qiymatini chaqiirilish nuqtasiga qaytaradigan operator bo‘lishi lozim. U ikkita shaklga ega bo‘ladi:

1) return qiymat;
2) return

Birinchi shakl natijani qaytarish uchun qo‘llanadi, shuning uchun aniqlashdagi funktsiya qanday turga ega bo‘lsa, ifoda ham shunday turga ega bo‘lishi kerak. Agar funktsiya qiymatni qaytarmasa, ya’ni tur void ga ega bo‘lsa, ikkinchi shakl qo‘llanadi. Dasturchining o‘zi bu operatorni funktsiya tanasida qo‘llamasligi mumkin, kompilyator uni funktsiya oxiriga avtomatik tarzda qo‘shib qo‘yadi.Qaytarilayotgan turning qiymati, massiv va funktsiyadan tashqari, har qanday bo‘lishi mumkin, ammo massiv yoki funktsiyaga ko‘rsatkich bo‘lishi mumkin.Formal parametrlar ro‘yxati - bu funktsiyaga uzatilishi lozim bo‘lgan qiymatlar. Ro‘yxat elementlari vergullar bilan ajratiladi. Har bir parametr uchun tur va nom ko‘rsatiladi. E’londa nomlarni ko‘rsatmasa ham bo‘ladi.Funktsiya tanasida yozilgan operatorlar bajarilishi uchun, funktsiyani chaqirib olish lozim. Chaqirishda funktsiyaning nomi va faktik parametrlari ko‘rsatiladi. Funktsiya tanasi operatorlarini bajarishda faktik parametrlar formal parametrlarning o‘rnini egallaydi. Faktik va formal parametrlar miqdori va turiga to‘ra bir-biriga mos kelishi kerak.Kompilyator chaqirilishning to‘g‘riligini tekshirish imkoniga ega bo‘lishi uchun, funktsiyani e’lon qilish funktsiya chaqirilishdan oldin matnda bo‘lmog‘i lozim. Agar funktsiya void bo‘lmagan turga ega bo‘lsa, u holda uning chaqirilishi ifodaning operatsiya bajarilayotgan elementi bo‘lishi mumkin.

***Prototip fuktsiyalar***

Funktsiyaga murojaat qilish mumkin bo‘lsin uchun, xuddi shu faylning o‘zida funktsiya aniqlovchisi yoki tavsifi (prototipi) bo‘lmog‘i lozim.

**double line(double x1, double y1, double x2 double y2);**

**double square(double a, double b, double c);**

**double triangle(double a, double b, double c);**

**double line(double, double, double double);**

**double square(double, double, double);**

**double triangle(double, double, double);**

Bu yuqorida tavsiflari keltirilgan funktsiyalarning prototiplaridir.

Protiplar bo‘lganda, chaqirilayotgan funktsiyalar chaqirayotgan funktsiyalar bilan bitta faylda bo‘lishlari shart emas, balki ular alohida modullar ko‘rinishida rasmiylashtirilishi hamda ko‘chirilgan holda ob’ektlar modullari kutubxonasida saqlanishlari mumkin. Xuddi shu narsa standart modullardagi funktsiyalarga ham tegishli. Bu holda ob’ekt modullari sifatida translyatsiya qilinib, rasmiylashtirilib bo‘lingan kutubxona funktsiyalarining aniqlovchilari kompilyator kutubxonasida bo‘ladi, funktsiyalar tavsiflarini esa dasturga qo‘shimcha ravishda kiritish lozim bo‘ladi. Bu ish include<fayl nomi> protsessor komandalari yordamida amalga oshiriladi.

Fayl_nomi sarlavhaviy faylni aniqlaydi. Sarlavhaviy fayl esa berilgan funktsiyalar kompilyatori uchun standart bo‘lgan guruhlar prototipiga ega bo‘ladi. Masalan, deyarli barcha dasturlarda biz kiritish-chiqarish ob’ektlar oqimining tavsifi uchun #include <iostream.h> komandasidan hamda ularga mos operatsiyalardan foydalandik.

Katta miqdordagi funktsiyalardan iborat bo‘lgan hamda turli modullarda joylashtirilgan dasturlarni ishlab chiqishda, funktsiyalar prototiplari va tashqi ob’ektlarning tavsiflari (konstantalar, o‘zgaruvchilar, massivlar) alohida faylga

joylashtiriladi. Bu fayl esa include “fayl_nomi’’ direktivasi yordamida har bir modulning boshiga kiritiladi.

***Funktsiya parametrlari***

Chaqirilayotgan va chaqirayotgan funktsiyalar o‘rtasida axborot almashinishning asosiy usuli bu mexanizm parametridir. Parametrlarni funktsiyaga uzatishning ikkita usuli mavjud: manzil bo‘yicha va qiymati bo‘yicha.

*Qiymati bo‘yicha uzatish*da quyidagi xatti-harakatlar bajariladi:

12) faktik parametrlar o‘rnida turgan ifodalar qiymatlari hisoblanadi;

13) funktsiyaning formal parametrlari uchun stekda xotira ajratiladi;

14) har bir faktik parametrga formal parametr qiymati beriladi, bunda turlarning o‘zaro muvofiqligi tekshiriladi hamda, zarurat tug‘ilganda, ular qayta o‘zgartiriladi.

**Misol:**

```
double square(double a, double b, double c);

{ //funktsiya a, b, c uzunlikdagi tomonlarga ega bo‘lgan uchburchak maydonini qaytarib beradi.

double s, r=(a+b+c)/2;

return s=sqtr(p*(p-a)*(p-b)*(p-c));  //Geron formulasi

}
```

Shunday qilib, stekka faktik parametrlarning nushalari kiritiladi va funktsiya operatorlari ushbu nushalar bilan ish olib boradi. Faktik parametrlarning o‘ziga funktsiyaning kirish huquqi yo‘q, demak ularni o‘zgartirish imkoni ham yo‘q.

*Manzil bo‘yicha uzatish*da stekka parametrlar manzillarining nushalari kiritiladi, demakki, funktsiyada faktik parametr joylashtirilgan xotira uyasiga kirish huquqi paydo bo‘ladi va funktsiya bu parmetrni o‘zgartirishi mumkin.

```
void Change(int*a,int*b)//manzil bo‘yicha uzatish
{int r=*a; *a=*b;*b=r;}
int x=1,y=5;
Change(&x,&y);
cout<<’’x=’’<<x<<’’y=’’<<y;
```

**x=5y=1 kelib chiqadi.**

Manzil bo‘yicha uzatish uchun iqtiboslar ham qo‘llanishi mumkin. Iqtibos bo‘yicha uzatishda funktsiyaga chaqirish paytida ko‘rsatilgan parametr manzili uzatiladi, funktsiya ichida esa parametrga barcha murojaatlarning sezilmagan holda nomlari bekor qilinadi.

```
void Change(int &a,int &b)
{int r=a; a=b;b=r;}
int x=1,y=5;
Change(x,y);
cout<<’’x=’’<<x<<’’y=’’<<y;
```

**x=5y=1 kelib chiqadi.**

Ko‘rsatkichlar o‘rniga iqtiboslardan foydalanish dasturning o‘qilishini yaxshilaydi, chunki bu o‘rinda nomni bekor qilish operatsiyasini qo‘llamasa ham bo‘ladi. Qiymat bo‘yicha uzatish o‘rniga iqtiboslardan foydalanish samaraliroq hamdir, chunki parametrlarni nushalashni talab qilmaydi. Agar funktsiya ichida parametrning o‘zgarishini taqiqlash lozim bo‘lib qolsa, bu holda const modifikatori qo‘llanadi. Sonst modifikatorini funktsiyada o‘zgarishi ko‘zda tutilmagan barcha parametrlar oldidan qo‘yish tafsiya qilinadi (undagi qaysi parametrlar o‘zgaradi-yu, kaysilari o‘zgarmasligi sarlavhadan ko‘rinib turadi).

***Lokal va global o‘zgaruvchilar***

Berilgan funktsiya ichida qo‘llanadigan o‘zgaruvchilar lokal deb ataladi. Ular uchun stekda xotira ajratilmaydi, shuning uchun, ish tugagach, funktsiyalar xotiradan chiqarib tashlanmaydi. Ko‘rsatkichni lokal o‘zgaruvchiga qaytarish mumkin emas, chunki bunday o‘zgaruvchi ajratib bergan xotira bo‘shatila boshlaydi.

```
Int*f()
{
int a;
....
return&a;//NOT¤o‘RI
}
```

Global o‘zgarvchilar - bu funktsiyadan tashqarida tavsiflangan funktsiyalar. Ular shunday nomli lokal funktsiyalar bo‘lmagan barcha funktsiyalarda ko‘rinadi.

**Misol:**

```
int a,b;//global o‘zgaruvchilar
void xhange()
{ int r;//lokal o‘zgaruvchi
r=a;a=b;b=r;
}
void main()
{ cin>>a,b;
change();
cout<<’’a=’’<<a<<’’b=’’<<b;
}
```

Funktsiyalar o‘rtasida ma’lumotlarni uzatish uchun global o‘zgaruvchilardan ham foydalanish mumkin, lekin bunday qilish tafsiya etilmaydi, chunki bu dasturni sozlashni qiyinlashtiradi hamda funktsiyalarni kutubxonaga joylashga to‘sqinlik qiladi. Funktsiyalar maksimal mustaqil bo‘lishiga, funktsiya prototipi esa ularning interfeysini to‘liQicha aniqlashiga intilish kerak.

***Dastlabki (yashirilgan) parametrlar qiymatiga***

***ega bo‘lgan funktsiyalar***

Funktsiyani aniqlashda dastlabki (yashirilgan) parametr qiymati bo‘lishi mumkin. Agar funktsiyani chaqirishda tegishli parametr tushirib qoldirilgan bo‘lsa, mana shu qiymat qo‘llanadi. Bunday parametrning o‘ng tomonida tavsiflangan parametrlar ham yashirilgan bo‘lishi lozim.

**Misol:**

**void print(int value=1)**

**{cout<<’’\n’’<<’’Uy raqami:’’<<value:}**

CHaqirishlar:

1. print();

Xulosa: Uy raqami: 1

2. print(15);

Xulosa: Uy raqami: 15

***Taqdim etiladigan (inline) funktsiyalar***

C++ dagi ayrim funktsiyalarni inline rasmiy so‘zini qo‘llagan holda aniqlash mumkin. Bunday funktsiya taqdim etilayotgan yoki o‘rnatilayotgan funktsiya deb ataladi.

Masalan:

inline float Line(floa x1, floa y1, floa x2=0, floa y2=0)

{return sqrt(pow(x1-x2)+pow(y1-y2,2));}// funktsiya (x1,u1) koordinatali nuqtadan (x2,u2) koordinatali nuqtagacha bo‘lgan masofani orqaga qaytaradi.

Qo‘yilgan funktsiyaning har bir chaqirishiga ishlov berar ekan, kompilyator dastur matniga dastur tanasi operatorlari kodini joylashtirishga urinadi. Inline spetsifikatori funktsiya uchun ichki boQlashni aniqlaydi. Ichki bog‘lash shundan iboratki, bunda kompilyator funktsiyani chaqirish o‘rniga funktsiya kodining komandalarini qo‘yadi. Bunda dastur xajmi kattalashishi mumkin, ammo chaqirilayotgan funktsiya boshqaruvni uzatish va undan qaytishga ketadigan sarflar bo‘lmaydi. Agar funktsiya tanasi bir necha operatorlardan iborat bo‘lsa, o‘rniga o‘rin qo‘yiladigan funktsiyalar qo‘llanadi.

Quyidagi funktsiyalar o‘rniga o‘rin qo‘yiladigan bo‘la olmaydi:

1. rekursiv funktsiyalar;
2. chaqirilishi funktsiyalarning aniqlanishidan oldin joylashtiriladigan funktsiyalar;
3. ifodada bittadan ortiq marta chaqiriladigan funktsiyalar;
4. davrlar, ulagichlar va uzatish diapazonlariga ega bo‘lgan funktsiyalar;
5. o‘rniga o‘rin qo‘yishni amalga oshirish uchun o‘ta katta xajmga ega bo‘lgan funktsiyalar.

***Funktsiyalarni ortiqcha yuklash***

Ortiqcha yuklashning maqsadi shundan iboratki, bunda bitta nomga ega bo‘lgan funktsiya turlicha bajarilishi kerak hamda unga murojaat qilinganda har xil turlarga va har xil sondagi faktik parametrlarga ega bo‘lgan har xil qiymatlarni qaytarib berishi kerak. Ortiqcha yuklashni ta’minlash uchun har bir ortiqcha yuklangan funktsiya uchun qaytarib berilayotgan qiymatlar va uzatilayotgan parametrlarni aniqlash kerak. Bu ish shunday amalga oshirilishi kerakki, bunda har bir ortiqcha yuklangan funktsiya xuddi shu nomli boshqa funktsiyadan ajralib tursin. Kompilyator faktik parametrlar turi bo‘icha qanday funktsiyani tanlab olishni aniqlab beradi.

**Misol:**

```
#include<iostream.h>

#include<string.h>

int max(int a, int b)
```

```
{ if (a>b) return a;
else return b;
}
float max(float a, float b)
{  if(a>b)return a;
else return b;
}
void main()
{ int a1,b1;
float a2, b2;
cout<<''\nfor int:\n'';
cout<<''a=?''; cin>>a1;
cout<<''b=?''; cin>>b1;
cout<<''\nMAX=''<<max(a1,b1)<<''\n'';
cout<<''\nfor float:\n;
cout<<''a=?''; cin>>a2;
cout<<''b=?''; cin>>b2;
cout<<''\nMAX=''<<max(a2,b2)<<''\n'';
```

Ortiqcha yuklangan funktsiyalarni tavsiflash qoidalari:

1) Ortiqcha yuklangan funktsiyalar bitta ko‘rish sohasida joylashtirilgan bo‘lishi kerak.

2) Ortiqcha yuklangan funktsiyalar yashiringan parametrlarga ega bo‘lishi mumkin, bundaturli funktsiyalardagi bitta parametrning qiymatlari o‘zaro mos bo‘lishi kerak. Ortiqcha yuklangan funktsiyalarning turli variantlarida turli miqdordagi yashiringan parametrlar bo‘lishi mumkin.

3) Agar funktsiyalar parametrlarining tavsifi faqat const modifikatori bilan yoki iqtibosning mavjudligi bilan farqlansa, funktsiyalar ortiqcha yuklangan bo‘lolmaydi.

Masalan, int&f1(int&,const int&){...} va int f1(int,int){...} funktsiyalari ortiqcha yuklangan emas, chunki funktsiyalarning qaysi biri chaqirilayotganini kompilyator bila olmaydi: parametrni qiymat bo‘yicha uzatayotgan hamda parametrni manzil bo‘yicha uzatayotgan funktsiyalarning chaqirilishi o‘rtasida sintaktik (ma’no bo‘yicha) farq yo‘q.

## 2.2 Funktsiyalar qiymatini hisoblash va ularga oid misollar.

***Bir o‘lchamli massivlarni funktsiya parametrlari sifatida uzatish***

Massivdan funktsiya parametri sifatida foylalanganda, funktsiyaning birinchi elementiga ko‘rsatkich uzatiladi, ya’ni massiv hamma vaqt adres bo‘yicha uzatiladi. Bunda massivdagi elementlarning miqdori haqidagi axborot yo‘qotiladi, shuning uchun massivning o‘lchamlari haqidagi ma’lumotni alohida parametr sifatida uzatish kerak. Funktsiyaga massiv boshlanishi uchun ko‘rsatkich uzatilgani tufayli (adres bo‘yicha uzatish), funktsiya tanasining operatorlari hisobiga massiv o‘zgarishi mumkin.

**Misol: Massivdan barcha juft elementlar chiqarilsin.**

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

```
#include <stdlib.h>
int form(int a[100])
{  int n;    cout<<"\nEnter n";          cin>>n;
        for (int i=0;i<n;i++)  a[i]=rand()%100;
        return n;
}
void print(int a[100],int n)
{       for(int i=0;i<n;i++)  cout<<a[i]<<" ";       cout<<"\n";
}
void Dell(int a[100],int&n)
{       int j=0,i,b[100];
        for(i=0;i<n;i++)                  if(a[i]%2!=0)
                {  b[j]=a[i];j++;
                }
                n=j;
for(i=0;i<n;i++)a[i]=b[i];
}
void main()
{       int a[100];   int n;
        n=form(a);          print(a,n);
        Dell(a,n);    print(a,n);
}
```

***Satrlarni funktsiyalar parametrlari sifatida uzatish***

Satrlar funktsiyaga char turidagi bir o‘lchamli massivlar sifatida yoki char* turidagi ko‘rsatkichlar sifatida uzatilishi mumkin. Oddiy massivlardan farqli o‘laroq, funktsiyada satr uzunligi ko‘rsatilmaydi, chunki satr oxirida satr oxiri /0 belgisi bor.

**Misol: Berilgan belgini satrda qidirish funktsiyasi.**

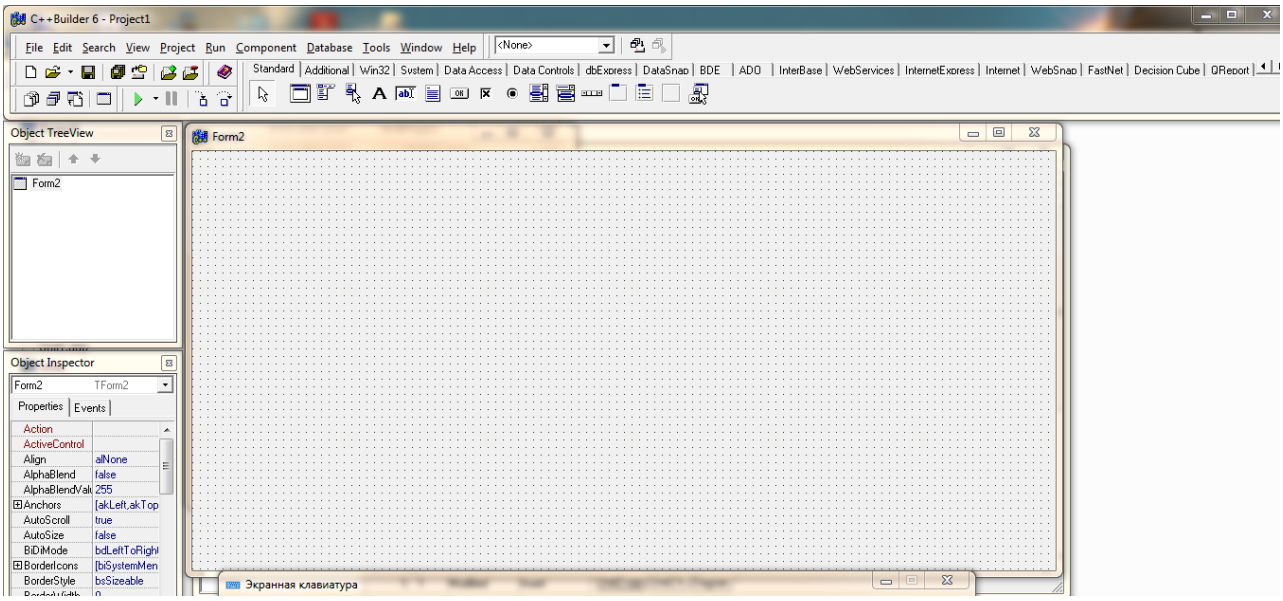
```
int  find(char *s,char c)
{ for (int I=0;I<strlen(s);I++)
if(s[I]==c) return I;
return –1
}
```

***Funktsiyaga ko‘p o‘lchamli massivlarni uzatish***

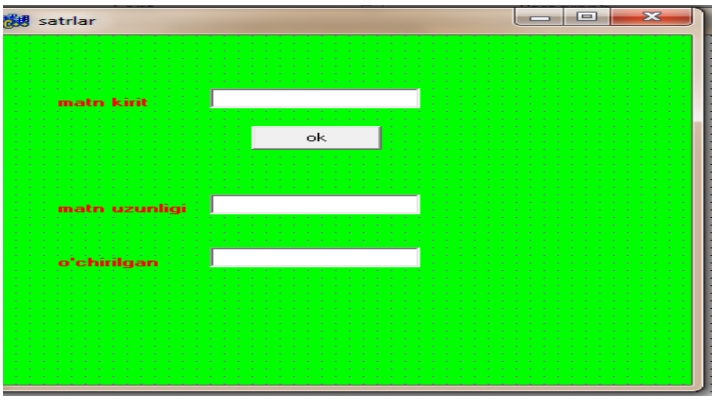
Ko‘p o‘lchamli massivlarni funktsiyaga uzatishda barcha o‘lchamlar parametrlar sifatida uzatilishi kerak. Si va SI++ da ko‘p o‘lchamli massivlar aniqlanishi bo‘yicha mavjud emas. Agar biz bir nechta indeksga ega bo‘lgan massivni tavsiflasak (masalan, int mas[3][4]), bu degani, biz bir o‘lchamli mas massivini tavsifladik, bir o‘lchamli int [4] massivlarining ko‘rsatkichlari esa uning elementlaridir

**<u>Misol:</u> satr uzunligini va harflarni o’chirish dasturini tuzish.**

buning uchun C++ builderni ishga tushiramiz u quydagicha ko’rinishga ega bo’ladi




form onasiga quydagi kampanentalarni tashlaymiz 3 ta edit, 3ta label va buttonlar tashlaymiz. Keyin ularning xossalarini o'zimizga kerakli qilib to'g'irlaymiz. Form oynasi quydagi ko'rinishga ega bo'ladi.

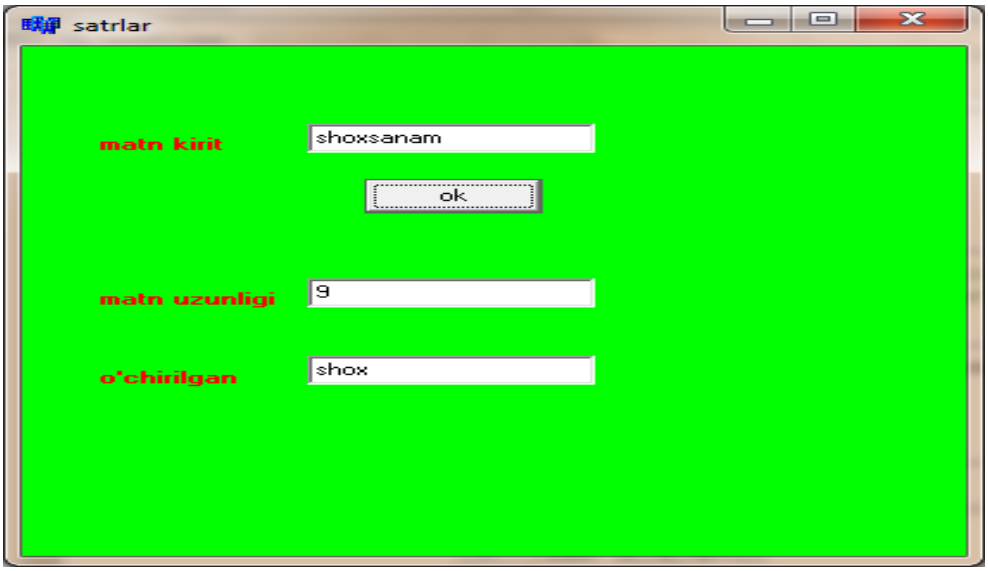



forma oynasini tayyor holga keltirgach ok tugmasiga sichqonchani tugmasini ikkimarta tez bosib kod oynasiga quydagigilarni kiritamiz

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{  String s,h;
int b,a,k=0;
s=Edit1->Text;
```

```
b=s.Length();

h=s.Delete(5,15) ;

Edit4->Text=(h);

Edit3->Text=IntToStr(b);

}
```

Dasturni ishlatamiz




**<u>Misol:</u> Aytaylik, uchburchak tomonlarining koordinatalari berilgan. Agar shunday uchburchak mavjud bo‘lsa, uning maydoni topilsin. Ushbu masalani yechuvchi dastur tuzing va unda funktsyani hisoblash algoritmlaridan foydalaning.**

1. **Matematik model:**

1) **I=sqrt(pow(x1-x2,2)+pow(y1-y2,2));//uchburchak tomonining uzunligi**

2) **p=(a+b+c)/2;**

**s=sqtr(p*(p-a)*(p-b)*(p-c));  //Geron formulasi**

**3) uchburchak mavjudligini tekshirish**

**(a+b>c&&a+c>b&&c+b>a)**

**2. Algoritm:**

1) **(x1,u1), (ch2,u2), (ch3,u3) uchburchagi tomonlarining koordinatalari kiritilsin;**

2) **ab, bc, ca tomonlarining uzunligi hisoblansin;**

3) **Shunday tomonlarga ega bo‘lgan uchburchakning mavjudligi tekshirilsin. Agar mavjud bo‘lsa, unda uning maydoni hisoblansin va natijasi chiqarilsin;**

4) **Agar mavjud bo‘lmasa, xabar chiqarilsin;**

5) **Agar hamma koordinatlar 0 ga teng bo‘lsa, unda tamom, aks holda 1-bandga qaytiladi.**

```
#include<iostream.h>

#include<math.h>

double line(double x1, double y1, double x2, double y2)

{ //Funktsiya x1,y1 x2, y2 return sqtr(pow(x1-x2,2)+pow(y1-y2,2)) koordinatalariga ega bo‘lgan kesim uzunligini qaytarib beradi:

}

double square(double a, double b, double c);

{ //funktsiya a, b, c uzunlikdagi tomonlarga ega bo‘lgan uchburchak maydonini qaytarib beradi.

double s, r=(a+b+c)/2;

return s=sqrt(p*(p-a)*(p-b)*(p-c));//Geron formulasi
```

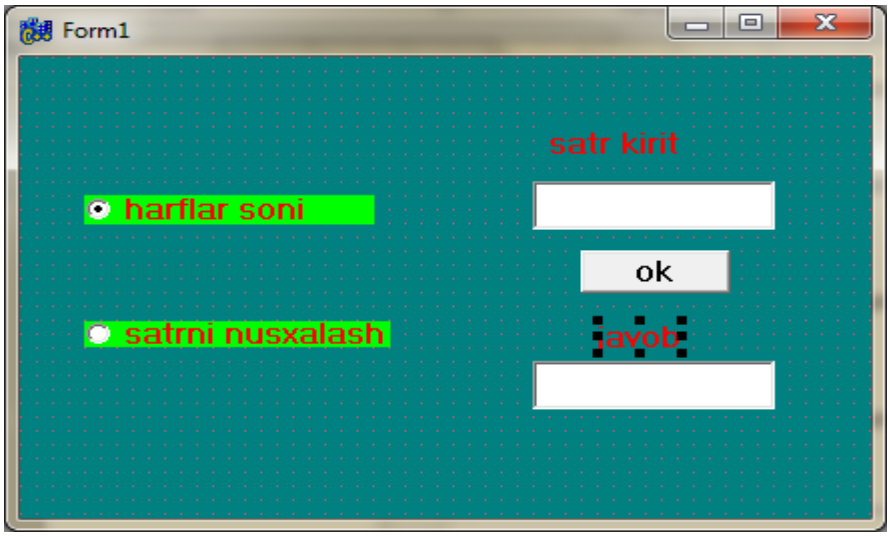
```
}
bool triangle(double a, double b, double c);
{  //agar uchburchak mavjud bo‘lsa, true ni qaytarib beradi
if(a+b>&&a+c>b&&c+b>a)return true;
else return false;
}
void main()
{ double x1=1,y1,x2,y2,x3,y3;
double point1_2,point1_3,point2_3;
do
{
cout<<’’\nEnter koordinats of triangle:’’;
cin>>x1>>y1>>x2>>y2>>x3>>y3;
point1_2=line(x1,y1,x2,y2);
point1_3=line(x1,y1,x3,y3);
point2_3=line(x2,y2,x3,y3);
if(triangle(point1_2,point1_3,point2_3)==true)
cout<<’’S=’’<<square(point1_2,point2_3,point1_3)<<’’\n’’;
else cout<<’’\n Triagle doesnt exist’’;
}
```

**while(!(x1==0&&y1==0&&x2==0&&y2==0&&x3==0&&y3==0));**

**}**

**Misol: satr va funksiya qatnashgan radiobutton kampanentasi ishtirokida dastur tuzish;**

**Buning uchun C++ builderni ishga tushiramiz va form oynasiga quydagi kampanentalarni tashlaymiz: 1 ta button, 2ta radiobutton,2ta label, 2ta edit larni tshlab ularni xossalarini to'g'irlaymiz.**




**Kod oynasiga quydagilarni kiritamiz**

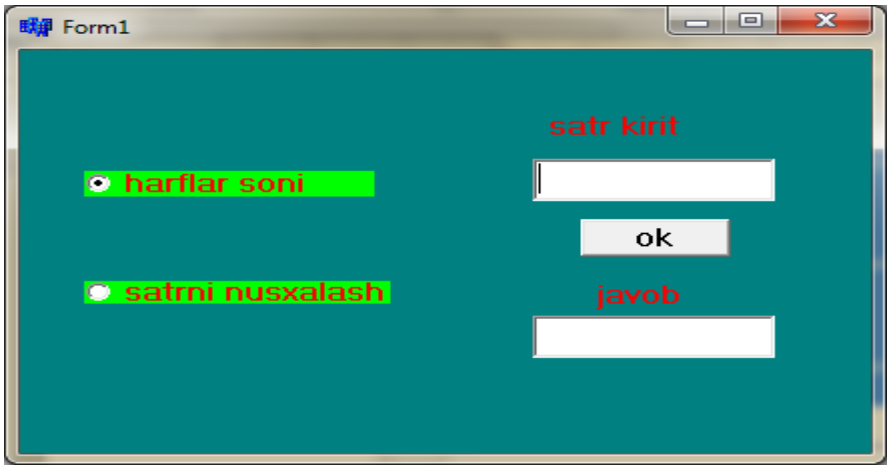
```
void __fastcall TForm1::Button1Click(TObject *Sender)
{ String s;
  int n,b,k=0;
  s=Edit1->Text;
  if (RadioButton1->Checked==true)
  n=s.Length();
```

**Edit2->Text=IntToStr(n);**

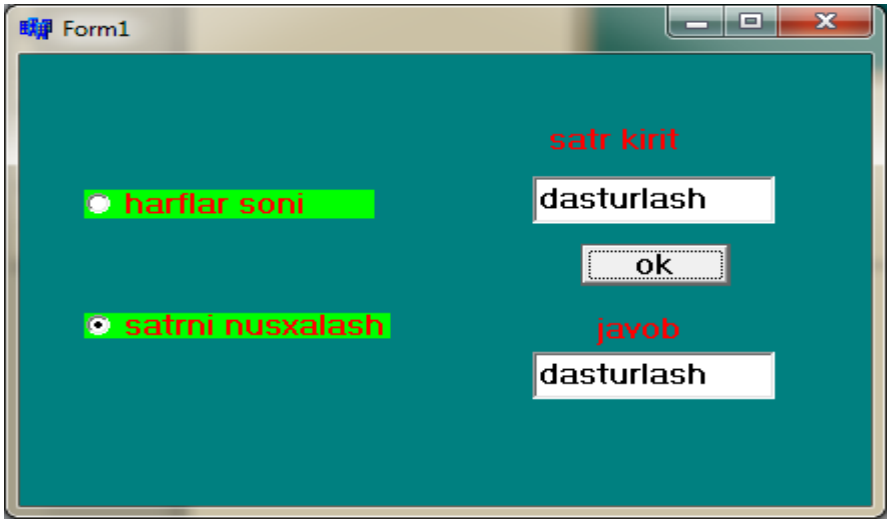
**if (RadioButton2->Checked==true)**

**Edit2->Text=s;**

**}**




**Dasturni ishlatib ko’ramiz**



## Xulosa.

Biz Algoritmlash dasturlash tillari fani texnologiyalaridan biri bulgan S++ ob’ektga yunaltirilgan tili bilan yakindan tanishdik. Ma’ruza darslarida olgan nazariy bilimlarimizni amaliyot darslarida mustaxkamlab oldik. Bu fan orqali ko‘lga kiritgan bilim va kunikmalarimiz ushbu kurs ishini tayyorlash jarayonida o‘zining ijobiy samaralarini berdi.Kurs ishim mavzusidan kelib chikib funksiya va satrlar bilan ishlash buyicha izlanishlar olib bordim. Amin buldimki, dasturlashda matematik masalalari bilan bir katorda simvollar, satrlar va katta xajmdagi ma’lumotlar bilan ishlash kabi biroz murakkab, ammo dolzarb masalalar mavjud. Kuyilgan masala buyicha algoritm ishlab chikdim. Tuzilgan algoritm asosida S++ ob’ektga yunaltirilgan dasturlash tilida kuyilgan masalani xal kiluvchi dastur tuzdim. Tuzgan dasturimni testdan utkazib, uning tug‘ri ishlayotganligiga amin buldim.Bugungi kunda aksariyat dasturchilar S++ tilida dastur tuzishadi, chunki bu dastulash muxiti bu imkoniyatlari bilan boshka dasturlash tillaridan farq qiladi. Bundan тащkari kupgina dasturlashtirish muhitlarining fundamental asosi xam S++ tiliga borib taqaladi. Nazariyada o‘rganilgan shu turdagi masalalarni echish kelajakda yukori saviyali dasturlar tuzishda asos bo‘lib qoladi desak mubolag‘a bo‘lmaydi. Aytish mumkinki, C++ tili zamonaviy dasturlash texnologiyalarining takomillashgan kurinishlaridan biridir.

## Foydalanilgan adabiyotlar


1. Jess Liberti, “Osvoy samostoyatelno S++ za 21 den”, Sankt Peterburg 2000, 815 s.
2. Liberti D. Osvoy samostoyatelno C++: 10 minut na urok. Per s angl. Vilyams, 374 str, 2004 g.
3. SHmidskiy YA.K. Prorammirovanie na yazыke S++: Samouchitel. Uchebnoe posobie. Dialektika. 361 str, 2004 g.
4. Kimmel P., «Borland C++5» . – SPb.: BHV, 1997 g.
5. Sayfiyev J. F., «S++ tiliga kirish», Buxoro 2004 y.
6. Nazirov Sh. A., Qobulov R. V., «Ob’yektga mo`ljallangan dasturlash», Toshkent 2006 y.
7. Boltayev Sh. J., Elov B. B., «Zmonaviy dasturlash tillari», Buxoro 2004 y.
8. Elov B. E., Boltayev Sh. J., «Dasturlash texnologiyalari», Toshkent - 2002 y.