O’ZBEKISTON RESPUBLIKASI ALOQA, AXBOROTLASHTIRISH VA TELEKOMMUNIKATSIYA TEXNOLOGIYALARI DAVLAT QO’MITASI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
NUKUS FILIALI

“Kompyuter injiniringi“ fakulteti

“Kompyuter injiniringi” yo’nalishi

C++ dasturlash

fanidan tayyorlangan

Kurs ishi

Qabul qilgan: _____

Bajargan: ***301-13 guruhi*** _____

NUKUS 2015-YIL

C++ TILIDA FUNKTSIYALAR QIYMATLARINI HISOBLOVCHI DASTURLARNI TUZISH

REJA:

1. **KIRISH**
2. **ASOSIY QISM**
   - ***A) FUNKTSIYANI E'LON QILISH***
   - ***B) FUNKTSIYA PROTOTIPLARI***
   - ***C) FUNKTSIYANING ANIQLANISHI***
3. **XULOSA**
4. **AMALIY ISH**
5. **FOYDALANGAN ADABIYOTLAR**

Kirish

C++ da dasturlashning asosiy bloklaridan biri funksiya-lardir. Funksiyalarning foydasi shundaki, katta masala bir necha kichik bo'laklarga bo'linib, har biriga alohida funksiya yozilganda, masala yechish algoritmi ancha soddalashadi. Bunda dasturchi yozgan funksiyalar C++ ning standart kutubxonasi va boshqa firmalar yozgan kutubxonalar ichidagi funksiyalar bilan birlashtiriladi. Bu esa ishni osonlashtiradi. Ko'p holda dasturda takroran bejariladigan amalni funksiya sifatida yozish va kerakli joyda ushbu funksiyani chaqirish mumkin. Funksiyani programma tanasida ishlatish uchun u chaqiriladi, yani uning ismi yoziladi va unga kerakli argumentlar beriladi.
() qavslar ushbu funksiya chaqirig'ini ifodalaydi. Masalan:

```
foo();

k = square(l);
```

Demak, agar funksiya argumentlar olsa, ular () qavs ichida yoziladi. Argumentsiz funksiyadan keyin esa () qavslarning o'zi qo'yiladi.

Funktsiyalar

C/C++ dasturlash tillari dastur kodining asosiy qismi funktsiyalar bo’lib hisoblanadi.Ular dasturni bir necha bloklarga bo’lish imkonini beradi. Bizga ma’lumki bu tillardagi ixtiyoriy dastur main ( ) funktsiyasini o’zida mujassamlashtiradi. Funktsiyalarni yaxshi tayyorlashi, dasturning effektli va ishonchli ishlashini ta’minlaydi. Ushbu ma’ruzada, funktsiyalar yaratish bilan bog’liq bir nechta dasturlarni keltirib o’tamiz. Shuningdek ma’ruzada C/C ++ tillarining ko’plab standart funktsiyalari haqida ham to’xtalib ketamiz.

Obiektlarga ixtisoslashgan dasturlashda asosiy e’tibor funktsiyaga emas, balki obiektga qaratilgan bo’lsada dasturlarda funktsiya markaziy komponentligicha qoldi. Biz shu mavzuda quydagilar bilan tanishamiz:

- Funksiya nima va u qanday qismlardan iborat?
- Funksiya qanday e’lon qilinadi va aniqlanadi?
- Funksiyaga qanday qilib parametrlar uzatiladi?

1. Funksiya qanday qiymat qaytaradi?

Funksiya nima?

Funksiya bu ma’nosiga ko‘ra dastur osti bo‘lib, u ma’lumotlarni o‘zgartirishi va biror bir qiymat qaytarishi mumkin. S++ da har bir dastur hech bo‘lmaganda bitta main() funksiyasiga ega bo‘ladi. main() funksiyasi dastur ishga tushirilishi bilan operatsion sistema tomonidan avtomatik chaqiriladi. Boshqa funksiyalar esa u tomonidan chaqirilishi mumkin. Har bir funksiya o‘zining nomiga egadir. Qachonki, dasturda bu nom uchrasa boshqaruv shu funksiya tanasiga o‘tadi. Bu jarayon funksiyani chaqirilishi (yoki funksiyaga murojaat qilish) deb aytiladi. Funksiya ishini tugatgandan so‘ng dastur o‘z ishini funksiya chaqirilgan qatorning keyingisidan boshlab davom ettiradi. Dastur bajarilishining bunday sxemasi

Qaytariladigan qiymatlar, parametrlar va argumentlar.

Funksiya biror bir qiymat qaytarishi mumkin. Funksiyaga murojaat qilingandan so‘ng u qandaydir amallarni bajaradi, keyin esa u o‘z ishining natijasi sifatida biror bir qiymat qaytaradi. Bu kaytariladigan kiymat deb ataladi va bu qiymatning tipi oldindan e’lon qilinishi lozim. Quyidagi yozuvda myFunction funksiyasi butun sonli qiymat qaytaradi.

```
int myFunction()
```

Funksiyaga ham o‘z navbatida biror bir qiymat uzatish mumkin. Uzatiladigan qiymatlar funksiyaning parametrlari deb aytiladi.

```
int myFunction (int Par, float  ParFloat);
```

Bu funksiya nafaqat butun son qaytaradi, balki parametr sifatida butun va haqiqiy sonli qiymatlarni qabul qiladi.

Parametrda funksiya chaqirilganda unga uzatiladigan qiymat tipi aniqlanishi lozim. Funksiyaga uzatiladigan haqiqiy qiymatlar argumentlar deb aytiladi.

```
int theValueReturned=myFunction(5,6,7);
```

Bu yerda theValueReturned nomli butun sonli o‘zgaruvchiga argument sifatida 5, 6 va 7 qiymatlar berilgan myFunction funksiyasining qaytaradigan qiymati o‘zlashtirilayapti. Argument tiplari e’lon qilingan parametr tiplari bilan mos kelishi lozim.

Funksiyani e’lon qilish va aniqlash.

Dasturda funksiyani qo‘llash uchun, oldin uni e’lon qilish, keyin esa aniqlash lozim. Funksiyani e’lon qilishda kompilyatorga uning nomi, qaytaradigan qiymatlari va parametrlari haqida xabar beriladi. Funksiyani aniqlanishidan kompilyator uning qanday ishlashi haqida ma’lumot oladi. Dasturdagi biror funksiyani oldindan e’lon qilmasdan turib chaqirish mumkin emas. Funksiyani e’lon qilinishi uning prototipini (timsolini) hosil qilish deb ataladi.

Funksiyani e’lon qilish.

Funksiyani e’lon qilishning uch xil usuli mavjud:

- Funksiya prototipi faylga yoziladi, keyin esa u #include ifodasi qo‘llanilib kerakli dasturga qo‘shib qo‘yiladi.
- Funksiya ishlatiladigan faylga uning prototiplari yoziladi.
- Funksiya uni chaqiruvchi ixtiyoriy funksiyadan oldin yoziladi va bu holda funksiya e’lon qilinishi bilan bir vaqtda aniqlanadi.

Funksiyani prototipini tuzmasdan turib ham uni ishlatishdan oldin e’lon kilish mumkin. Lekin, dasturlashning bunday uslubi quyidagi uchta sababga ko‘ra yaxshi hisoblanmaydi.

Birinchidan, funksiyani faylda ko‘rsatilgan tartibda yozish, uni dastur ishlatilishida o‘zgartirish jarayonini murakkablashtiradi.

Ikkinchidan, quyidagi ko‘p uchraydigan holatni amalga oshirish imkoniyati mavjud emas.

A() funksiya V() funksiyani chaqirsin. Xuddi shuningdek, dasturning biror bir qismida V()funksiya A() funksiyani chaqirsin. U holda biz A()funksiyani V() funksiya aniqlanmasdan turib ishlata olmaymiz.

Bu holda hech bo‘lmaganda bitta funksiya oldindan e’lon qilinishi lozim.

Uchinchidan, funksiyaning prototiplari dasturni tekshirish jarayonida juda yaxshi ishlatiladi. Agarda funksiya prototipi aniqlangan bo‘lsa unga muvofiq funksiya aniqlangan parametrini qabul qiladi yoki aniqlangan biror bir qiymat qaytaradi. Dasturda e’lon qilingan prototipga muvofiq bo‘lmagan funksiyani ishlatishga urinsak kompilyator bu xatolikni kompilyatsiya jarayonini o‘zidayoq aniqlaydi va dastur ishlashida turli noxush xatoliklarni ro‘y berishining oldini oladi.

Funksiya prototiplari.

Ko‘pgina ichki qurilgan funksiyalarning prototiplari dasturga #include kalit so‘zi yordamida qo‘shiladigan fayl-sarlavhasida yoziladi. Foydalanuvchi tomonidan tuziladigan funksiyalar uchun esa ularning mos prototiplarini dasturga qo‘shish dasturchi tomonidan bajarilishi lozim.

Funksiyaning prototipi nuqtali vergul orqali tugaydigan funksiyani qaytaradigan qiymati va signaturasidan iboratdir. Funksiyani signaturasi deb uning nomi va parametrlar ro‘yxati tushiniladi.

Formal parametrlar ro‘yxati barcha parametrlar va ularning tiplarini ifodalaydi.

unsigned short int FindArea (int length, int width) ;

Funksiyaning prototipi hamda aniqlanishidagi uning qaytaradigan qiymati tipi va signaturasi mos bo‘lishi lozim. Agarda bunday mutanosiblik bo‘lmasa kompilyator xatolik haqida xabar beradi. Funksiya prototipida parametr nomlarisiz tiplarni ko‘rsatilishi yetarlidir. Masalan, quyida keltirilgan misol tug‘ridir:

long Area(int, int)

Bu prototip ikkita butun sonli parametrni qabul qilib, long tipidagi qiymat qaytaradigan Area()nomli funksiyani e’lon qiladi. Prototipning bunday yozilishi unchalik yaxshi variant emas. Prototipga parametrlarning nomlarini qo‘shilishi uni tushunarliroq bo‘lishini ta’minlaydi.

Har bir funksiyaning qaytaradigan qiymati tipi aniqlangan bo‘ladi. Agarda u ochiq aniqlanmagan bo‘lsa avtomatik ravishda int tipini qabul qiladi.

1–misol. Funksiyani e’lon qilinishi, aniqlanishi va ishlatilishi.

Funksiyaning aniqlanishi.

Funksiyaning aniqlanishi ikki qismdan – uning sarlavhasi va tanasidan iboratdir. Funksiyaning sarlavhasi uning prototipiga o‘xshash aniqlanadi, faqatgina bu holda parametrlar nomlangan bo‘lishi shart va sarlavha oxirida nuqtali vergul qo‘yilmaydi. Funksiya tanasi figurali qavsga olingan ifodalar to‘plamidan iborat.

int Yuza (int uzunlik,int kenglik)

{ – ochiluvchi figurali kavs.

// funktsiya tanasi

*return (uzunlik*kenglik);*

} – yopiluvchi figurali kavs.

Funksiyaning sarlavxasi va tanasi

Funksiyaning bajarilishi.

Funksiya chaqirilganda unda ko‘rsatilgan amallar ochiluvchi figurali qavsdan ({) keyingi birinchi ifodadan boshlab bajariladi. Funksiya tanasida if shartli operatoridan foydalanib tarmoqlanishni ham amalga oshirish mumkin.

Funksiya o‘z tanasida boshqa funksiyalarni va hatto o‘z – o‘zini ham chaqirishi mumkin.

Lokal o‘zgaruvchilar.

Funksiyaga qiymatlar uzatish bilan birga uning tanasida o‘zgaruvchilarni e’lon qilish ham mumkin. Bu lokal o‘zgaruvchilar orqali amalga oshiriladi. Qachonki dasturni bajarilishi funksiyadan asosiy qismga qaytsa, bu funksiyadagi lokal o‘zgaruvchilar xotiradan o‘chiriladi.

Lokal o‘zgaruvchilar xuddi boshqa o‘zgaruvchilar kabi aniqlanadi. Funksiyaga beriladigan parametrlarni ham lokal o‘zgaruvchilar deb atash mumkin va ularni funksiya tanasida aniqlangan o‘zgaruvchilar kabi ishlatish mumkin. 2 – misolda funksiya parametrlari va funksiya ichida aniqlangan lokal o‘zgaruvchilarni qo‘llashga oid misol keltirilgan.

Global o‘zgaruvchilar.

main() funksiyasida aniqlangan o‘zgaruvchilar dasturdagi barcha funksiyalar uchun murojaat qilishga imkonli va ko‘rinish sohasiga ega hisoblanadi. Bunday o‘zgaruvchilar dasturdagi funksiyalar uchun global o‘zgaruvchilar deyiladi.

Global o‘zgaruvchi nomi bilan funksiya ichida nomlari ustma-ust tushadigan lokal o‘zgaruvchilar faqatgina joriy funksiyaning ichidagina global o‘zgaruvchining qiymatini o‘zgartiradi. Lekin global o‘zgaruvchi funksiya o‘z ishini tugatgach u

chaqirilishidan oldingi qiymatini saqlab qoladi, ya’ni funksiya tanasida e’lon qilingan lokal o‘zgaruvchi funksiyaning ichida global o‘zgaruvchini yashiradi xolos. Bunda lokal o‘zgaruvchi alohida hosil qilinadi va funksiya ishlash vaqtida global va lokal o‘zgaruvchilarning nomlari bir xil bo‘lsa faqatgina lokal o‘zgaruvchi ustida amallar bajariladi. Global o‘zgaruvchi esa funksiyaning bajarilishi davomida oldingi qiymatini saqlab turadi. Quyidagi 3-misolda main() va sum() funksiyalarida bir xil nomdagi o‘zgaruvchilarni ishlatish ko‘rsatilgan. Programmada ikkita sonning yig‘indisi hisoblanadi va chop etiladi:

3-misolda keltirilgan programmada kompilyatsiya xatosi ro‘y beradi, chunki f1() funksiya uchun x o‘zgaruvchisi noma’lum hisoblanadi.

Programma matnida global o‘zgaruvchilarni ular e’lonidan keyin yozilgan ixtiyoriy funksiyadan ishlatish mumkin. Shu sababli, global o‘zgaruvchilar programma matnining boshida yoziladi. Funksiya ichidan global o‘zgaruvchiga murojat qilish uchun funksiyada uning nomi bilan mos tushadigan lokal o‘zgaruvchilar bo‘lmasligi kerak. Agar global o’zgaruvchi e’lonida unga boshlang‘ich qiymat berilmagan bo‘lsa, ularning qiymati 0 hisoblanadi. Global o‘zgaruvchining amal qilish sohasi uning ko‘rinish sohasi bilan ustma-ust tushadi.

Lokal o‘zgaruvchilar haqida batafsilroq ma’lumot.

Funksiyani ichida aniqlangan o‘zgaruvchilar lokal ko‘rinish sohasiga ega deyiladi. Yuqorida aytib o‘tilganidek, bu o‘zgaruvchilarni faqatgina funksiyaning ichidagina qo‘llash mumkinligini anglatadi. C++da o‘zgaruvchilarni nafaqat dasturning boshida balki ixtiyoriy joyda aniqlash mumkin. Agarda o‘zgaruvchi funksiya tanasidagi biror bir blok ichida aniqlangan bo‘lsa, bu o‘zgaruvchi faqatgina shu blok ichidagina ta’sirga ega bo‘lib butun funksiyaning ichida ko‘rinish sohasiga ega bo‘lmaydi.

Funksiyada ishlatiladigan operatorlar.

Funksiyada ishlatiladigan operatorlar soni va turiga hech qanday chegara yo‘q. Funksiya ichida istalgancha boshqa funksiyalarni chaqirish mumkin bo‘lsada, lekin funksiyalarni aniqlash mumkin emas.

C++ tilida funksiyaning hajmiga hech qanday talab qo‘yilmasa ham uni kichikroq tarzda tuzgan ma’quldir. Har bir funksiya tushunish oson bo‘ladigan bitta masalani bajarishi lozim. Agarda funksiya hajmi kattalashayotganligini sezsangiz, bu funksiyani bir nechta kichikroq funksiyalarga ajratish haqida o‘ylashingiz kerak.

Funksiya argumentlari.

Funksiyaning argumentlari turli tipda bo‘lishi mumkin. Shuningdek, argument sifatida C++ tilidagi biror bir qiymat qaytaradigan o‘zgarmaslar, matematik va mantiqiy ifodalar va boshqa ixtiyoriy funksiyalarni berish mumkin.

Misol sifatida bizda biror bir qiymat qaytaruvchi double(), triple(), square() va cube() funksiyalari berilgan bo‘lsin. Biz quyidagi ifodani yozishimiz mumkin:

Answer=(double(triple(square(my Value))))

Bu ifodada myValueo‘zgaruvchisini qabul qilinadi va u cube()funksiyasiga argument sifatida uzatiladi.
cube() funksiyasi qaytargan qiymat esa square() funksiyasiga argument cifatida uzatiladi. square() funksiyasi qiymat qaytargandan keyin, buning qiymati o‘z navbatida triple()fuknsiyasiga argument sifatida beriladi. triple() funksiyasining qiymati esa double()funksiyasiga argument qilib beriladi va u Answer o‘zgaruvchisiga o‘zlashtiriladi.

Parametrlar bu lokal o‘zgaruvchilardir.

Funksiyaga uzatilgan har bir argument, bu funksiyaga nisbatan lokal munosabatda bo‘ladi. Funksiyani bajarilishi jarayonida argumentlar ustida bajarilgan o‘zgartirishlar funksiyaga qiymat sifatida berilgan o‘zgaruvchilarga ta’sir qilmaydi. Bu fikr 5 – misolda namoyish qilingan.

5–misol. Funksiyaga parametrlarni qiymat sifatida uzatish.

```
 // Funksiyaga parametrlarni kiymat sifatida
// uzatish
#include <iostream.h>
void Almashtirish(int x, int y);
int main()
{ int x = 5,  y =10;
cout << “Main(). Almashtirishdan oldin,x:”
<<x <<”  y:”  << y <<“\n”;
Almashtirish(x,y);
```

```
cout<<“Main(). Almashtirishdan keyin, x:”
<<x<< “y:” <<y<< “\n”;
return 0;
}
void Almashtirish(int x, int y)
{
int temp;
cout<<“Almashtirish().Almashtirishdan ”<<
“oldin, x: ”<<x<<“ y: ”<<y<< “\n”;
temp = x ;
x = y;
y = temp;
cout<<“Almashtirish().Almashtirishdan ”<<
“keyin, x: ”<<x<<“ y: ”<<y<< “\n”;
}
```

HATIJA:

Main(). Almashtirishdan oldin,x: 5 y:10

Almashtirish().Almashtirishdan oldin, x:5 y:10

Almashtirish().Almashtirishdan keyin, x:10 y:5

Main(). Almashtirishdan keyin, x:5 y:10 x:

Funksiyaning qaytaradigan qiymatlari.

Funksiya yo biror bir real qiymatni, yo kompilyatorga hech qanday qiymat qaytarilmasligi haqida xabar beruvchi voidtipidagi qiymatni qaytaradi.

Funksiyani qiymat qaytarishi uchun returnkalitli so‘zidan foydalaniladi. Bunda oldin returnkalitli so‘zi, keyin esa qaytariladigan qiymat yoziladi. Qiymat sifatida esa o‘zgarmaslar kabi butun bir ifodalarni ham berish mumkin. Masalan:

return 5 ;

return (x > 5) ;

return (MyFunction()) ;

MyFunction()funksiyasi biror bir qiymat qaytarishidan kelib chiqsak, yuqoridagi barcha ifodalar to‘g‘ri keltirilgan. return(x>5)ifodasi esa x 5dan katta bo‘lsa true, kichik yoki teng bo‘lsa false mantiqiy qiymatini qaytaradi.

Agarda funksiyada returnkalit so‘zi uchrasa undan keyingi ifoda bajariladi va uning natijasi funksiya chaqirilgan joyga uzatiladi. returnoperatori bajarilgandan keyin dastur funksiya chaqirilgan satrdan keyingi ifodaga o‘tadi. return kalitli so‘zidan keyingi funksiya tanasidagi operatorlar bajarilmaydi.

Funksiya bir nechta return operatorlarini o‘zida saqlashi mumkin. Bu g‘oya 6 – misolda namoyish qilingan.

6 – misol. Bir nechta returnoperatorini qo‘llanilishi

```
// 6 – misol.
# include < iostream.h>
int IkkigaKupaytirish(int KupaytSon);

int main()
{
int natija=0;
int input;
cout << “Ikkiga ko`paytiriladigan sonni”
<< “kiriting(0 dan 10000 gacha):”;
cin >> input;

cout << “\n IkkigaKupaytirish() funktsiyasi”
<< “chaqirilishidan oldin\n”;
cout<<“Kiritilgan qiymat:” <<input
<<“Ikkilangani:”<<natija<< “\n”;
```

```
result = IkkigaKupaytirish(input);
cout<<“\nIkkigaKupaytirish() funktsiyasidan”
<<“qaytgandan so`ng…\n”;

cout<<“Kiritilgan qiymat:” <<input
<<“Ikkilangani:”<<natija<< “\n”;

return 0;
}
int IkkigaKupaytirish(int original)
{
if (original <= 10000)
return original*2;
else
return –1;
cout<< “ Siz bu satrga o`ta olmaysiz!\n”;
}
```

NATIJA:

Ikkiga ko`paytiriladigan sonni kiriting (0 dan 10000 gacha): 9000

IkkigaKupaytirish() funktsiyasi chaqirilishidan oldin

Kiritilgan qiymat: 9000 Ikkilangani: 0

Ikkiga_kupaytirish() funktsiyasidan qaytgandan so`ng

Kiritilgan qiymat:9000 Ikkilangani: 18000

Ikkiga ko`paytiriladigan sonni kiriting (0 dan 10000 gacha): 11000

IkkigaKupaytirish() funktsiyasi chaqirilishidan oldin

Kiritilgan qiymat: 11000 Ikkilangani: 0

Ikkiga_kupaytirish() funktsiyasidan qaytgandan so`ng

Kiritilgan qiymat:11000 Ikkilangani: -1

Bir xil nomli har xil funksiyalar

C++ tilida bir nomdagi bir nechta funksiyalarni tuzish imkoniyati mavjuddir. Bu *bir xil nomdagi xar xil funksiyalar*deyiladi. Qayta yuklanuvchi funksiyalar bir – birlari bilan parametrlari ro‘yxati bilan farq qilishi lozim. Bunda parametrlar yoki turli sonda aniqlanishi, yoki tiplari bilan farqlanishi kerak. Quyidagi misollarni ko‘rib chiqamiz:

int myFunction(int, int)

int myFunction(long, long)

int myFunction(long)

myFunction()funksiyasi uchta turli parametrlar ro‘yxati bilan xar xil nomda aniqlanyapti. Funksiyaning birinchi va ikkinchi versiyalari parametrlar tipi bilan, uchinchisi esa parametrlar soni bilan farq qilayapti.

Ushbu funksiyalarning qaytaradigan qiymatlarining tiplari bir xil yoki turlicha bo‘lishi mumkin. Lekin, parametrlari ro‘yxati bir xil bo‘lgan ikkita funksiya turli tipdagi qiymat qaytaradigan qilib aniqlansa dasturni kompilyatsiya qilish jarayonida xatolik yuz beradi.

Funksiyalarni bir xil nom bilan aniqlanishi ularning polimorfizmi deb ham ataladi. Polimorfizm so‘zi poli*(*gr.poly*)* – ko‘p, morfe *(*gr. morphyo*)* - shakl co‘zidan olingan bo‘lib, ko‘pshakllilik degan ma’noni anglatadi.

Funksiyaning polimorfizmi deganda dasturda bir nechta turli vazifalarni bajaruvchi bir xil nomdagi funksiyalar bo‘lishi tushuniladi. Parametrlari soni va tipini o‘zgartirish orqali bir nechta bir xil nomdagi funksiyalarni aniqlash mumkin. Bunday holda funksiyani chaqirishda hech qanday noaniqlik bo‘lmaydi, kerakli funksiya parametriga muvofiq tarzda aniqlanadi.

Faraz qilamiz, siz ixtiyoriy berilgan qiymatni ikkiga ko‘paytiradigan funksiya yozmoqchisiz. Bu funksiyaga int, long, float yoki doubletipidagi qiymatlarni uzatish imkoniyati bo‘lishini hohladingiz. Bir xil nomli xar xil funksiyalarsiz buni bajarish uchun to‘rtta turli nomdagi funksiyalarni hosil qilishingiz kerak bo‘ladi:

int DoubleInt(int);

long DoubleLong(long);

float DoubleFloat(float);

double DoubleDouble(double);

Bir xil nomli funksiyalar yordamida esa ularni o‘rniga ushbu funksiyalarni aniqlashimiz mumkin.

int Double(int);

long Double(long);

float Double (float);

double Double(Double);

Qayta yuklangan funksiyalar chaqirilganda kompilyator aynan qaysi variantdagi funksiyani ishlatish kerakligini avtomatik tarzda uzatilayotgan parametrlarning tiplariga muvofiq aniqlaydi. Bir xil nomli funksiyalarning aniqlanishi 5.8.– misolda ko‘rsatilgan.

Funksiyalar haqida qo‘shimcha ma’lumot. Rekursiya.

Funksiya o‘z – o‘zini chaqirishi rekursiya deyiladi. U ikki xil – to‘g‘ri rekursiya va bilvosita rekursiya bo‘lishi mumkin. Agarda funksiya o‘zini o‘zi chaqirsa bu to‘g‘ri rekursiya deyiladi. Agarda funksiya boshqa bir funksiyani chaqirsa va u funksiya o‘z navbatida birinchisini chaqirishi esa bilvosita rekursiya deyiladi.

Ayrim muammolar rekursiya yordamida osonroq yechiladi. Agarda ma’lumotlar ustida biror protsedura ishlatilsa va uning natijasi ustida ham xuddi shu protsedura bajarilsa, bunday hollarda rekursiyani ishlatish lozim. Rekursiya ikki xil natija bilan yakunlanadi: u yoki oxir – oqibat albatta tugaydi va biror natija qaytaradi, ikkinchisi hech qachon tugallanmaydi va xatolik qaytaradi.

Agarda funksiya o‘zini o‘zi chaqirsa, bu funksiyani nusxasi chaqirilishini aytib o‘tish lozim. Rekursiya yordamida yechiladigan muammo sifatida Fibonachchi qatorini ko‘rsatish mumkin.

1, 1, 2, 3, 5, 8, 13, 21, 34 …

Bu qatorning har bir soni (ikkinchisidan keyin) o‘zidan oldingi ikki sonning yig‘indisiga teng. Masala quyidagidan iborat: Fibonachchi qatorini 12 – a’zosi nechaga tengligini aniqlang. Bu muammoning yechishning usullaridan biri qatorni batafsil tahlil qilishdan iboratdir. Qatorning oldingi ikki soni birga teng. Har bir navbatdagi son oldingi ikkita sonning yig‘indisiga teng. Ya’ni, o‘n yettinchi son o‘n oltinchi va o‘n beshinchi sonlar yig‘indisiga teng. Umumiy holda n – sonning yig‘indisi (n-2)- va (n-1) – sonlarning yig‘indisiga teng (n>2 hol uchun).
Rekursiv funksiyalar uchun rekursiyani to‘xtatish shartini berish zarur. Fibonachchi qatori uchun to‘xtatish sharti n<3 shartidir.
Bunda quyidagi algoritm qo‘llaniladi:

1. Foydalanuvchidan Fibonachchi qatorining qaysi a’zosini hisoblashini ko‘rsatishini so‘raymiz.

2. fib()funksiyasini chaqiramiz va unga foydalanuvchi tomonidan berilgan Fibonachchi qatori a’zosi tartib nomerini argument sifatida uzatamiz.

3. fib() funksiyasi (n) argumentni tahlil qiladi. Agarda n<3 bo‘lsa, funksiya 1 qiymat qaytaradi; aks holda fib() funksiyasi o‘zini o‘zi chaqiradi va unga argument sifatida n–2 qiymatni beradi, keyin yana o‘z–o‘zini chaqiradi va bu funksiyaga n–1ni qiymat sifatida beradi. Bundan keyin esa ularning yig‘indisini qiymat sifatida uzatadi.

Agarda fib(1) funksiyasi chaqirilsa u 1 qiymatni qaytaradi. Agarda fib(2) funksiyasi chaqirilsa u ham 1 qiymatni qaytaradi. Agarda fib(3)funksiyasi chaqirilsa u fib(1) va fib(2)funksiyalarini yig‘indisini qaytaradi. fib(2)va fib(1)funksiyalari 1 qiymatni qaytarganligi sababli fib(3)funksiyasi qiymati 2 ga teng bo‘ladi.
Agarda fib(4)funksiyasi chaqirilsa, bunda u fib(3)va fib(2) funksiyalari yig‘indisini qiymat sifatida qaytaradi. fib(3) funksiyasi 2 va fib(2) funksiyasi 1 qiymat qaytarishidan fib(4)funksiyasi 3 ni qiymat sifatida qaytarishi kelib chiqadi. Demak, Fibonachchi qatorining to‘rtinchi a’zosi 3 ga teng ekan.

Yuqorida, tavsiflangan usul bu masalani yechishda unchalik samarali usul emas. fib(20)funksiyasi chaqirilsa, u tomonidan fib()funksiyasi 13529 marta chaqiriladi. Bunday hollarda ehtiyot bo‘lish lozim. Agarda Fibonachchi qatorining juda katta nomerini bersak xotira yetishmasligi mumkin. Fib() funksiyasini har safar chaqirilishida xotiradan ma’lum bir soha zahiralanadi. Funksiyadan qaytish bilan xotira bo‘shatiladi. Lekin, rekursiv tarzda funksiya chaqirilsa

xotiradan uning uchun yangi joy ajratiladi va toki ichki funksiya o‘z ishini tugatmaguncha uni chaqirgan funksiya egallagan joy bo‘shatilmaydi. Shuning uchun xotirani yetishmay qolish xavfi bor. Fib() funksiyasining ishlatilishi 4 – misolda ko‘rsatilgan.

IZOH

Ayrim kompilyatorlar sout ob’ekti qatnashgan ifodalarda operatorlarni qullashda ba’zi qiyinchiliklar paydo bo‘ladi. Agarda kompilyator 28 satrdagi ifoda haqida ogohlantirish bersa ayirish operatsiyasini qavs ichiga oling, bu qator quyidagi ko‘rinishga ega bo‘lsin.
cout << “Call fib(“ <<(n-2)<<”) and fib(“ <<(n-2)<< “).\n”;

Funksiyaning ishlash tamoyili haqida

Funksiya chaqirilganda dastur boshqaruvi chaqirilgan funksiyaga uzatiladi, ya’ni funksiya tanasidagi operatorlarning bajarilishi boshlanadi. Funksiya operatorlari bajarilgach, u biror qiymatni natija sifatida qaytaradi (agarda u aniqlanmagan bo‘lsa funksiya voidtipidagi qiymatni qaytaradi) va boshqaruv funksiya chaqirilgan joydan davom ettiriladi.
Bu masala qanday amalga oshiriladi? Dasturga uning boshqaruvi qaysi blokka o‘tishi lozimligi qanday ma’lum bo‘ladi? Argument sifatida aniqlangan o‘zgaruvchi qaerda saqlanadi? Funksiya tanasida aniqlangan o‘zgaruvchilar qaerda saqlanadi? Qaytariladigan qiymat qanday orqaga uzatiladi? Dasturga u qaysi joydan ish boshlashi kerakliligi qaerdan ma’lum?
Ko‘pgina adabiyotlarda bu savollarga javob berilmaydi. Lekin, funksiyani ishlash prinsipini bilmasak bizga kompilyatorning ishlash prinsipi juda mavhum bo‘lib ko‘rinadi. Shuning uchun ushbu mavzuda kompyuterning xotirasi bilan bog‘liq savollar ko‘rib chiqiladi.

Abstraksiya darajasi

Kompyuterlar, albatta, oddiy elektron mashinalardir. Ular oynalar va menyular, programmalar va komandalarni bilishmaydi. Ular hatto 0 va 1 larni ham tushunishmaydi. Haqiqatda esa bo‘ladigan barcha jarayonlar integral mikrosxemaning turli joylarida elektr tokiga bog‘liqdir. Va hattoki bu ham abstraksiyadir. Elektrning o‘zi ham elementar chastotalar harakatini umumlashtiruvchi konsepsiyadir.
Ayrim dasturchilar operativ xotira (OZU) da saqlanadigan qiymatlar darajasiga tushishgacha detalizatsiya qilishadi. Lekin mashinani boshqarish yoki to‘pni tepish uchun fizika fanidagi elementar chastotalarni o‘rganish zarur bo‘lmaganidek,

dasturlashda ham bu darajadagi detalizatsiya talab etilmaydi. Demak, kompyuter elektronikasini tushunmasdan turib ham dasturlashtirish mumkin ekan.

Lekin siz kompyuter xotirasi qanday tashkil etilganligini tushunishingiz kerak. Sizning o‘zgaruvchilaringiz tuzilgandan keyin qaerda joylashishi va ularga funksiyalar orqali qiymat qanday berilishini haqida aniq tasavvurga ega bo‘lmasangiz, dasturlash siz uchun sirli bo‘lib qolaveradi.

Xotiraning taqsimlanishi

Siz o‘zingizni dasturingiz bilan ishlay boshlashingiz bilan operatsion sistema (Dos yoki Microsoft Windows) kompilyatorning talabiga muvofiq xotira sohasidan joy ajratadi. S++ dasturchisi sifatida siz global nomlar fazosi, erkin taqsimlanuvchi xotira, registr, segmentli xotira va stek tushunchalarini bilishingiz lozim.

Global nomlar fazosida global o‘zgaruvchilar saqlanadi. Global nomlar fazosi va erkin taqsimlanuvchi xotira haqida keyingi mavzularda batafsilroq tanishib chiqamiz. Hozir esa registr, dastur segmenti va stek haqida to‘xtalib o‘tamiz.

Registr xotiraning maxsus sohasi bo‘lib, uning asosiy vazifasi ichki yordamchi funksiyalarni ishlashini tashkil etishdir.

Dasturning o‘zi dastur operatorlari ikkilik formatda saqlanishi uchun ajratilgan kompyuter xotirasida saqlanadi.

Stek – bu sizning dasturingizda funksiya chaqirilganda undagi ma’lumotlar uchun talab qilinadigan xotiraning maxsus sohasidir. Uning stek deb atalishiga sabab «oxirgi kelgan – birinchi ketadi» prinsipi asosida ishlashidir. Haqiqatan ham, funksiyalarni chaqirilishi xuddi shu prinsip asosida amalga oshiriladi. Agarda bir funksiya ikkinchisini chaqirsa, ikkinchi funksiya o‘z ishini tugatgandan so‘ng birinchi funksiya o‘z ishini yakunlaydi, ya’ni oxirgi chaqirilgan funksiya birinchi ishini tugatadi.

Amaliy ish

1 – Misol. Funksiya prototipini

```
#include <iostream>
#include <cmath>
using namespace std;
float S;
int uchburchakning_maydoni (int a, int b, int c)
{float p;
   p=(a+b+c)/2;
   S=sqrt(p*(p-a)*(p-b)*(p-c));
   return S;
}
int main()
{
   uchburchakning_maydoni(5,6,7);
   cout <<" uchburchakning_maydoni ="<<S;
   return 0;
}
```

2 – misol. Funksiya lokal o‘zgaruvchilari va parametrlarining qo‘llanilishi.

```
# include <iostream.h>
float Almashtirish(float);
int main()
{
  float TempFer;
  float TempCel;
```

```
  cout << “Feringait bo`yicha temperaturani
  << “kiriting:”;
  cin >> TempFer;
TempCel = Almashtirish(TempFer);
cout << “\n Bu temperatura selziy shkalasi
<< “bo`yicha: ”;
cout << TempCel << endl;
return 0 ;
}
float Almashtirish(float TempFer)
{
float TempCel;
tempCel=((TempFer-32)*5)/9;
return TempCel;
};
```

3–misol. Global va lokal o‘zgaruvchilarning qo‘llanishi.

```
#include <iostream.h>
// funksiya prototipi
int sum (int a; int b);
int main()
{
// lokal  o’zgaruvchi
int x=r;
cout<<sum(x,y);
return 0;
}
```

```
int sum(int a, int b)
{
// lokal o'zgaruvchi
int x=a+b;
return x;
}
```

4 – misol. Fibonachchi qatori a'zosining qiymatini topish uchun rekursiyani qo'llanilishiga misol.

```
#include <iostream.h>
int fib(int n);
int main( )
{
int n, javob;
cout << "Izlanayotgan nomerni kiriting:";
cin >> n;
        cout << "\n\n";

        javob=fib(n);
        cout<< "Fibonachchi qatorining"<< n
         <<"nomeri qiymati " <<javob<<" ga teng \n";
        return 0;
        }
        int fib(int n)
        {
        cout << "fib("<< n << ") jarayoni…";
          if (n <3)
```

```
 {
  cout<< "1 qiymatni qaytarayapti!"\n;
  return (1);
}
else
{
 cout<< "fib(" << n-2 << ") va fib(" <<n-1
 cout<< ") funktsiyalari chaqirayapti. \n";
return(fib(n-2)+fib(n-1));
 }
}
```

NATIJA:

Izlanayotgan nomerni kiriting:: 6

fib(6) … fib(4) va fib(5) funktsiyalarini chaqirayapti.

fib(4) … fib(2) va fib(3) funktsiyalarini chaqirayapti.

fib(2) … 1 qiymatni qaytarayapti!

fib(3) … fib(2) va fib(1) funktsiyalarini chaqirayapti.

fib(1) … 1 qiymatni qaytarayapti!

fib(2) … 1 qiymatni qaytarayapti

fib(5) … fib(3) va fib(4) funktsiyalarini chaqirayapti.

fib(3) … fib(2) va fib(1) funktsiyalarini chaqirayapti.

fib(1) … 1 qiymatni qaytarayapti!

fib(2) … 1 qiymatni qaytarayapti

fib(4) … fib(2) va fib(3) funktsiyalarini chaqirayapti.

fib(2) … 1 qiymatni qaytarayapti

fib(3) … fib(2) va fib(1) funktsiyalarini chaqirayapti.

fib(1) … 1 qiymatni qaytarayapti!

fib(2) … 1 qiymatni qaytarayapti

Fibonachchi qatorining 6 nomeri qiymati 8 ga teng

XULOSA

Xulosa qilib aytganda, C++ dasturlash tili va unda funktsiyalar, funktsiylarni e’lon qilish, funktsiyaning prototiplari, funktsiyaning aniqlanishi, qaytariladigan qiymatlari xaqida umumiy ma’lumotlarga ega bo`ldim.

Biz funktsiyani e’lon qilishda kompilyatorga uning nomi, qaytaradigan qiymatlari va parametrlari haqida xabar beriladi.

Funksiyaning aniqlanishi esa ikki qismdan – uning sarlavhasi va tanasidan iboratdir. Funksiyaning sarlavhasi uning prototipiga o‘xshash aniqlanadi, faqatgina bu holda parametrlar nomlangan bo‘lishi shart va sarlavha oxirida nuqtali vergul qo‘yilmaydi. Funksiya tanasi figurali qavsga olingan ifodalar to‘plamidan iborat.

Ko‘pgina ichki qurilgan funksiyalarning prototiplari dasturga #includekalit so‘ziyordamida qo‘shiladigan fayl-sarlavhasida yoziladi. Foydalanuvchi tomonidan tuziladigan funksiyalar uchun esa ularning mos prototiplarini dasturga qo‘shish dasturchi tomonidan bajarilishi lozim.

Funksiyaning prototipi nuqtali vergul orqali tugaydigan funksiyani qaytaradigan qiymati va signaturasidan iboratdir. Funksiyani signaturasi deb uning nomi va parametrlar ro‘yxati tushiniladi.

Men C++ dasturi strukturasi haqida, belgilar bayoni, algoritm va dastur tushunchasi, ma’lumotlarni kiritish va chiqarish operatorlari hamda dasturda ishlatiladigan toifalar, ifodalar va operatorlar hamda sinflar va funksiya, funktsiyaning aniqlanishi, prototipi, e’lon qilinishi haqida bilim va ko’nikmalarga ega bo`ldim.

FOYDALANGAN ADABIYOTLAR.

1. Маркушевич А. И. Теория аналитических функций. В 2-х т. – М.: Наука, 1968. Т.2. – 624с
2. Голузин Г.М. Геометрическая теория функции комплексного переменного. – М. : Наука, 1976.– 540 с.
3. Б. В. Шабат. Введение в комплексный анализ. 1–част. М.Н. 1989.
4. Г. Худайберганов, А. Ворисов, Х. Мансуров. Комплекс анализ. Тошкент, «Университет», 1998.
5. Г. Худайберганов, А. Ворисов, Х. Мансуров. Комплекс анализ.Карши. «Насаф», 2003.
6. http://old.ziyone.uz
7. http://ziyonet.uz