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

TOSHKENT AXBOROT TEXNOLOGOYALARI UNIVERSITETI QARSHI FILIALI

“AXBOROT TEXNOLOGIYALARI” kafedrasi

”Informatika va axborot texnologiyalari” fanidan

KURS ISHI

Mavzu: ”**C++ vizual dasturlash muhitida virtual funksiyalar ”**

Bajardi : **TT 21-guruh talabasi**
Normurodov E.

Qabul qildi : **ass. Abdullayev. R.**

Qarshi-2014

Reja :

Kirish

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 taňaladi. Funksiyalar dastur yozishda katta qulayliklar beradilar. Lekin mashina saviyasida funksiyani har gal chaqirtirish qo'shimcha ish bajarilishiga olib keladi. Registrlardagi o'zgaruvchilar o'zgartiriladi, lokal yozgaruvchilar quriladi, parametr sifatida berilgan argumentlar funksiya stekiga o'ziladi.Bu, albatta, qo'shimcha vaqt oladi. Umuman aytganda, hech funksiyasiz yozilgan dastur, yani hamma amallari faqat main() funksiyasi ichida bajariladigan monolit programma, bir necha funksiyalarga ega, ayni ishni bajaradigan dasturdan tezroq ishlaydi. Funksiyalarning bu noqulayligini tuzatish uchun inline (satr ichida) ifodasi funksiya e'loni bilan birga qo'llaniladi. inline sifatli funksiyalar tanasi dasturdagi ushbu funksiya chaqirig'I uchragan joyga qo'yiladi. inline deb ayniqsa kichik funksiyalarni belgilash effektivdir. inline ning o'ziga yarasha kamchiligi ham bor, inline qo'llanilganda dastur hajmi oshdi. Agar funksiya katta bo'lsa va dastur ichida ko'p marotaba chaqirilsa, programma hajmi juda kattalashib ketishi mumkin. Oddiy, inline sifati qo'llanilmagan funksiyalar chaqirilish mehanizmi quyidagicha bo'ladi. Dastur kodining ma'lum bir yerida funksiya tanasi bir marotaba aniqlangan bo'ladi. Funksiya chaqirig'i uchragan yerda funksiya joylashgan yerga ko'rsatkich qo'yiladi. Demak, funksiya chaqirig'ida dastur funksiyaga sakrashni bajaradi. Funksiya o'z ishini bajarib bo'lgandan keyin dastur ishlashi yana sakrash joyiga qaytadi. Bu dastur hajmini ihchamlikda saqlaydi, lekin funksiya chaqiriqlari vaqt oladi. Kompilyator inline ifodasini inobatga olmasligi mumkin, yani funksiya oddiy holda kompilyatsiya qilinishi mumkin. Va ko'pincha shunday bo'ladi ham. Amalda faqat juda kichik funksiyalar inline deya kompilyatsiya qilinadi.

Funktsiya ta'rifi.

Funktsiyani C++ tilida quyidagi ikki sifatda qarash mumkin:

1. ***xosila tiplardan biri;***
2. ***dastur bajariluvchi minimal moduli.***

Funktsiya ta'rifi umumiy ko'rinishi quyidagichadir:

<tip> <funktsiya nomi> (<formal_parametrlar_ta'rifi>)

Formal parametrlarga ta'rif berilganda ularninga boshlang'ich qiymatlari xam ko'rsatilishi mumkin.

Funktsiya qaytaruvchi ifoda qiymati funktsiya tanasida return <ifoda> ; operatori orkali ko'rsatiladi.

Misol:

```
float  min(float, float b)
{ if (a<b) return a; return b; }
```

Funktsiyaga murojaat kilish quyidagicha amalga oshiriladi:

<Funktsiya nomi> (<xaqiqiy parametrlar ro'yxati>)

Masalan:

```
int x=5, y=6,z;  z=min(x,y)  yoki  int z=min(5,6)  yoki  int  x=5;  int z=min(x,6)
```

Funktsiya ta'rifida formal parametrlar initsializatsiya qilinishi, ya'ni boshlang'ich qiymatlar ko'rsatilishi mumkin.

Misol uchun:

```
float  min(float a=0.0, float b=0)
{ if (a<b) return a; return b; }
```

Bu funktsiyaga quyidagicha murojaat qilish mumkin:

```
int y=6,z;
z=min(,y)  yoki
int z=min(,6);
```

Agar programmada funktsiya ta'rifi murojaatdan keyin berilsa, yoki funktsiya boshqa faylda joylashgan bo'lsa, murojjatdan oldin shu funktsiyaning prototipi joylashgan bo'lishi kerak. Prototip funktsiya nomi va formal parametrlar tiplaridan iborat bo'ladi. Formal parametrlar nomlarini berish shart emas.

Misol uchun:

float min(float, float);

Funktsiyaga parametrlar qiymat bo'yicha uzatiladi. Funktsiyaga parametrlar qiymat-lari uzatilishi xaqiqiy parametrlar qiymatlarini funktsiya tanasida o'zgartirish imkonini bermaydi. Bu muammoni xal qilish uchun ko'rsatkichlardan foydalanish mumkin.

Primer:

```
void change (int &a, int &b)
{
int r;
r =a;  a = b; b = r;
}
// funktsiya chaqirig'i
change(a, b);
```

Inlayn funktsiyalar.

Dasturda funktsiya ta'riflanganda kompilyator funktsiya kodini bir marta mashina kodiga o'tkazadi va dasturga ma'lumotlarni stekka joylovchi instruktsiyalar qo'shadi. Argumentlarni steka qo'shish, funktsiyaga o'tish va qaytish mashina vaqtini oladi.

C++ kompilyatori inline, so'zini uchratsa bajariluvchi faylga xar bir funktsiyaga murojaat o'rniga funktsiya operatorlarini qo'yadi. Shunday qilib dastur effektivligini oshirish mumkindir.

Masalan:

```
inline float Line(float x1,float y1,float x2=0, float y2=0)
{return sqrt(pow(x1-x2)=pow(y1-y2,2));}//funktsiya ikki nuqta orasidagi masofani qaytaradi.
```

Funktsiyalarni qo'shimcha yuklash.

Funktsiyalarni qo'shimcha yuklashdan maqsad bir xil nomli funktsiyaga xar xil tipli o'zgaruvchilar bilan murojaat qilib qiymat olishdir. Kompilyator xaqiqiy

parametrlar ro’yxati va funktsiya chaqirig’i asosida qaysi funktsiyani chaqirish kerakligini o’zi aniqlaydi.

Misol uchun xar xil o’zgaruvchilarni ko’paytirish uchun quyidagi funktsiyalar kiritilgan bo’lsin:

```
float  min(float a, float b)
 { if (a<b) return a; return b; }
int  min(int a, int b)
{ if (a<b) return a; return b; }
```

quyidagi murojaatlarning xar biri to’g’ri bajariladi:

```
int mqmin(6, x);
float cqmin(5, y);
```

Rekursiv funktsiyalar va funktsiyalar

Rekursiv funktsiya deb uziga uzi murojjat kiluvchi funktsiyaga aytiladi. Misol uchun faktorialni xisoblash funktsiyasini keltiramiz:

```
Long fact(int k)
{if (k==0) return 1;
 return k*fact(k-1);}
```

Biz oldinroq main() funktsiyasi bilan tanishib chiqqan edik. Bu funktsiya odatdagi bo’lmagan, yagona turdagi funktsiyadir. Funktsiyalar dasturning ishlash davrida chaqirilishi kerak. main() funktsiyasi esa dastur tomonidan emas, balki operatsion sistema tomonidan chaqiriladi.

Dastur berilgan matni buyicha satrlarni joylashishiga qarab tartib bilan toki biror bir funktsiya chaqirilguncha bajariladi. Keyin esa boshqaruv birinchi uchragan funktsiyaga beriladi. Funktsiya bajarilgandan so’ng boshqaruv yana dasturning funktsiya chaqirilgan joydan keyingi satriga beriladi. (Chaqirilgan funktsiyadan keyingi satrga beriladi.)

Funktsiyani ishlash jarayoniga mos o’xshashlik mavjud. Masalan, siz rasm chizib turgan vaqtingizda qalamingiz sinib qoldi. Siz rasm chizishni to’xtatasiz va qalamni yo’na boshlaysiz. Keyin esa, rasm chizishni qalamingiz sinib qolgan

joydan boshlab davom ettirasiz. Qachonki, dastur biror bir xizmat ko’rsatuvchi amallarni bajarilishiga ehtiyoj sezsa kerakli funktsiyani chaqiradi. Bu operatsiya bajarilgandan keyin esa dastur o’z ishini funktsiya chaqirilgan joydan boshlab davom ettiradi. Bu g’oya quyidagi misolda namoyish etilgan.

Funktsiyalarning qo’llanilishi.

Funktsiyalar yo void tipidagi, yo boshqa biror bir tipdagi qiymat qaytaradi. Ikkita butun sonni qo’shib, ularning yig’indisini qaytaradigan funktsiya butun qiymat qaytaruvchi deyiladi. Faqatgina qandaydir amallarni bajarib, hech qanday qiymat qaytarmaydigan funktsiyaning qaytaruvchi tipi void deb e'lon qilinadi.
Funktsiya sarlavha va tanadan iboratdir. Funktsiya sarlavhasida uning qaytaradigan tipi, nomi va parametrlari aniqlanadi. Parametrlar funktsiyaga qiymat uzatish uchun ishlatiladi. Masalan, agar funktsiya ikki sonni qo’shishga mo’ljallangan bo’lsa u holda bu sonlarni funktsiyaga parametr qilib berish kerak. Bunday funktsiyaning sarlavhasi quyidagicha bo’ladi:

```
int Sum(int a,int b)
```

Parametr – bu funktsiyaga uzatiladigan qiymat tipini e'lon qilishdir. Funktsiya chaqirilganda unga uzatiladigan qiymat argument deb aytiladi. Ko’pchilik dasturchilar bu ikkala tushunchani sinonim sifatida qarashadi. Ba'zilar esa bu terminlarni aralashtirishni noprofessionallik deb hisoblaydi. Mavzularda ikkala termin bir xil ma'noda kelgan.

Funktsiya tanasi ochiluvchi figurali qavs bilan boshlanadi va u bir necha qatordan iborat bo’lishi mumkin. (Funktsiya tanasida hech qanday satr bo’lmasligi ham mumkin). Satrlardan keyin esa yopiluvchi figurali qavs keladi. Funktsiyaning vazifasi uning satrlarida berilgan dasturiy kodlar bilan aniqlanadi. Funktsiya dasturga return operatori orqali qiymat qaytaradi. Bu operator funktsiyadan chiqish ma'nosini ham anglatadi. Agarda funktsiyaga chiqish operatorini (return) qo’ymasak funktsiya

satrlarini tugashi bilan u avtomatik void tipidagi qiymatni qaytaradi. Funktsiya qaytaradigan tip uning sarlavhasida ko’rsatilgan tip bilan bir xil bo’lishi lozim.

C++ da virtual funktsiyalar

C++ da dasturlashning asosiy bloklaridan biri funksiyalardir. 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 kutubhonasi va boshqa firmalar yozgan kutub-honalar ichidagi funksiyalar bilan birlashtiriladi. Bu esa ishni osonlashtiradi. Ko'p holda dasturda takroran bajariladigan 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.

Funksiyalarning tuzilishi

Funksiyalar dasturchi ishini juda yengillashtiradi. Funksiyalar yordamida programma modullashadi, qismlarga bo'limadi. Bu esa keyinchalik dasturni rivojlantirishni osonlashtiradi. Dastur yozilish davrida hatolarni topishni yengillashtiradi. Bir misolda funksiyaning asosiy qismlarini ko'rib chiqaylik.

```
int foo(int k, int t) {
```

```
  int result;
   result = k * t;
  return (result);
}
```

Yuqoridagi foo funksiyamizning ismi, () qavslar ichidagi parametrlar – int tipidagi k va t lar kirish argument-laridir, ular faqat ushbu funksiya ichida ko'rinadi va qo'llaniladi. Bunday o'zgaruvchilar lokal(local-mahalliy)
deyiladi. result foo() ning ichida e'lon qilinganligi uchun u ham lokaldir. Demak biz funksiya ichida o'zgaruvchilarni va klaslarni (class) e'lon qilishimiz mumkin ekan. Lekin funksiya ichida boshqa funksiyani e'lon qilib bo'lmaydi. foo() funksiyamiz qiymat ham qaytaradi. Qaytish qiymatining tipi foo() ning e'lonida eng boshida kelgan - int tipiga ega. Biz funksiyadan
qaytarmoqchi bo'lgan qiymatning tipi ham funksiya e'lon qilgan qaytish qiymati tipiga mos kelishi kerak - ayni o'sha tipda bo'lishi yoki o'sha tipga keltirilishi mumkin bo'lgan tipga ega bo'lishi shart. Funksiyadan qiymatni
return ifodasi bilan qaytaramiz. Agar funksiya hech narsa qaytarmasa e'londa void tipini yozamiz. Yani:

```
void funk(){
  int g = 10;

  cout << g;
  return; }
```

Bu funksiya void (bo'sh, hech narsasiz) tipidagi qiymatni qaytaradi. Boshqacha qilib aytganda qaytargan qiymati bo'sh to'plamdir. Lekin funksiya hech narsa qaytarmaydi deya olmaymiz. Chunki hech narsa qaytarmaydigan mahsus funksiyalar ham bor. Ularning qaytish qiymati belgilana-digan joyga hech narsa yozilmaydi. Biz unday funksiyalarni keyinroq qo'rib chiqamiz. Bu yerda bir nuqta shuki, agar funksiya mahsus bo'lmasa, lekin oldida qaytish qiymati tipi ko'rsatilmagan bo'lsa, qaytish qiymati int tipiga ega deb qabul qilinadi.

Void qaytish tipli funksiyalardan chiqish uchun return; deb yozsak yetarlidir.Yoki return ni qoldirib ketsak ham bo'ladi. Funksiyaning qismlari bajaradan vazifasiga ko'ra turlicha nomlanadi. Yuqorida korib chiqqanimiz funksiya aniqlanishi (function definition) deyiladi, chunki biz bunda funksiyaning bajaradigan amallarini funksiya nomidan keyin,{} qavslar ichida aniqlab yozib chiqyapmiz. Funksiya aniqlanishida {} qavslardan oldin nuqta-vergul (;) qo'yish hatodir. Bundan tashqari funksiya e'loni, prototipi yoki deklaratsiyasi (function prototype) tushunchasi qo'llaniladi. Bunda funksiyaning nomidan keyin hamon nuqta-vergul qo'yiladi, funksiya tanasi esa berilmaydi. C++ da funksiya qo'llanilishidan oldin uning aniqlanishi yoki hech bo'lmaganda e'loni kompilyatorga uchragan bo'lishi kerak. Agar funksiya e'loni boshqa funksiyalar aniqlanishidan tashqarida berilgan bo'lsa, uning kuchi ushbu fayl ohirigacha boradi. Biror bir funksiya ichida berilgan bo'lsa kuchi faqat o'cha funksiya ichida tarqaladi. E'lon fayllarda aynan shu funksiya e'lonlari berilgan bo'ladi. Funksiya e'loni va funksiya aniqlanishi bir-biriga mos tushishi kerak. Funksiya e'loniga misol:

```
double square(char, bool);
float average(int a, int b, int c);
```

Funksiya e'lonlarda kirish parametrlarining faqat tipi yozish kifoya, huddi square() funksiyasidek. Yoki kiruvchi parametrlarning nomi ham berilishi mumkin, bu nomlar kompilyator tarafidan etiborga olinmaydi, biroq dasturning o'qilishini ancha osonlashtiradi. Bulardan tashqari C++ da funksiya imzosi (function signature) tushunchasi bor. Funksiya imzosiga funksiya nomi, kiruvchi parametrlar tipi, soni, ketma-ketligi kiradi. Funksiyadan qaytuvchi qiymat tipi imzoga kirmaydi.

```
int foo();            //No1
int foo(char, int);   //No2
double foo();         //No3 - No1 funksiya bilan imzolari ayni.
void foo(int, char);  //No4 - No2 bilan imzolari farqli.
char foo(char, int);  //No5 - No2 bilan imzolari ayni.
int foo(void);        //No6 - No1 va No3 bilan imzolari ayni,
                      //      No1 bilan e'lonlari ham ayni.
```

Yuqoridagi misolda kirish parametrlari bo'lmasa biz () qavsning ichiga void deb yozishimiz mumkin (No6 ga qarang). Yoki () qavslarning quruq o'zini yozaversak ham bo'ladi (No1 ga qarang). Yana bir tushuncha - funksiya chaqirig'idir. Dasturda funksiyani chaqirib,qo'llashimiz uchun uning chaqiriq ko'rinishini ishlatamiz. () qavslari funksiya chaqirig'ida qo'llaniladi. Agar funksiyaning kirish argumentlari bo'lmasa, () qavslar bo'sh holda qo'llaniladi. Aslida () qavslar C++ da operatorlardir. Funksiya kirish parametrlarini har birini ayri-ayri yozish kerak, masalan yuqoridagi

```
float average(int a, int b, int c);
```

funksiyasini

```
float average(int a,b,c); // Hato!
```

deb yozishimiz hatodir.

Hali aytib o'tganimizdek, funksiya kirish parametrlari ushbu funksiyaning lokal o'zgaruvchilaridir. Bu o'zgaruvchilarni funksiya tanasida boshqattan e'lon qilish sintaksis hatoga olib keladi. Bir dastur yozaylik.

```
//Funksiya bilan ishlash
# include <iostream.h>
 int foo(int a, int b); //Funksiya prototipi,
                        //argumentlar ismi shart emas.
 int main()
{
   for (int k = 1; k <6; k++){
      for (int l = 5; l>0; l--){
         cout << foo(k,l) << " ";        //Funksiya chaqirig'i.
      }//end for (l...)
      cout << endl;
   }//end for (k...)
return (0);
```

```
} //end main()

//foo() funksiyasining aniqlanishi
int foo(int c, int d)
{ //Funksiya tanasi
  return(c * d);
}
```

Ekranda:

```
 5 4 3 2 1
10 8 6 4 2
15 12 9 6 3
20 16 12 8 4
25 20 15 10 5
```

Bizda ikki sikl ichida foo() funksiyamiz chaqirilmoqda. Funksiyaga k va l o'zgaruvchilarining nushalari uzatil-moqda. Nushalarning qiymati mos ravishda funksiyaning aniqlanishida berilgan c va d o'zgaruvchilarga berilmoqda. k va l ning nushalari deganimizda adashmadik, chunki ushbu o'zgaruvchilarining qiymatlari funksiya chaqirig'idan hech qanday ta'sir ko'rmaydi. C++ dagi funksiyalarning bir noqulay tarafi shundaki, funksiyadan faqat bitta qiymat qaytadi. Undan tashqari yuqorida ko'rganimizdek, funksiyaga berilgan o'zgaruvchilarning faqat nushalari bilan ish ko'rilarkan. Ularning qiymatini normal sharoitda funksiya ichida o'zgartirish mumkin emas. Lekin bu muammolar ko'rsatkichlar yordamida osonlikcha hal etiladi. Funksiya chaqiriqlarida avtomatik ma'lumot tipining konversiyasi bajariladi. Bu amal kompilyator tomonidan bajarilganligi sababli funksiyalarni chaqirganda ehtiyot bo'lish kerak. Javob hato ham bo'lishi mumkin. Shu sababli kirish parametrlar tipi sifatida katta hajmli tiplarni qo'llash maqsadga muofiq bo'ladi. Masalan double tipi har qanday sonli tipdagi qiymatni o'z ichiga olishi mumkin. Lekin bunday qiladigan bo'lsak, biz tezlikdan yutqazishimiz turgan gap. Avtomatik konversiyaga misol keltiraylik.

```
int division(int m, int k){
return (m / k);
}
dasturda chaqirsak:...
float f = 14.7;
double d = 3.6;
int j = division(f,d); //f 14 bo'lib kiradi, d 3 bo'lib kiradi
                    // 14/3 - butun sonli bo'lish esa 4 javobini beradi
cout << j;
```

...

Ekranda:

4

Demak kompilyator f va d o'zgaruvchilarining kasr qismlarini tashlab yuborar ekan. Qiymatlarni pastroq sig'imli tiplarga o'zgartirish hatoga olib keladi.

Matematik kutubhona funksiyalar

Standart kutubhonaning matematik funksiyalari ko'pgina amallarni bajarishga imkon beradi. Biz bu kutubhona misolida funksiyalar bilan ishlashni ko'rib chiqamiz. Masalan bizning dasturimizda quyidagi satr bor bo'lsin:

```
double = k;
int m = 123;
k = sin(m);
```

kompilyator uchbu satrni ko'rganida,standart kutubhonadan sin funksiyasini chaqiradi. Kirish qiymati sifatida m ni berdik. Javob, yani funksiyadan qaytgan qiymat k ga berildi.Funksiya agumentlari o'zgarmas sonlar (konstanta) o'zgaruvchilar, ifodalar va boshqa mos keluvchi qiymat qaytaradigan funksiyalar bo'lishi mumkin. Masalan:

```
int g = 49, k = 100;
cout << "4900 ning ildizi -> "<< sqrt( g * k );
```

Ekranda:

4900 ning ildizi -> 70;

Matematik funksiyalar aksariyat hollarda double tipidagi qiymat qaytarishadi.Kiruvchi argumentning tipi sifatida esa double ga keltirilishi mumkin bo'lgan tip beriladi. Bu funksiyalarni ishlatish uchun math.h (yangi ko'rinishda cmath)e'lon faylini include bilan asosiy dastur tanasiga kiritish kerak.Quyida matematik funksiya-lar kutubhonasining bazi bir a'zolarini beraylik. x va y o'zgaruvchilari double tipiga ega.

Funksiya	**Aniqlanishi**	**Misol**
ceil(x)	x ni x dan katta yoki unga teng b-n eng kichik butun songacha yahlitlaydi	ceil(12.6) = 13.0 ceil(-2.4) = -2.0
cos(x)	x ning trigonometrik kosinusi (x radianda)	cos(0.0) = 1.0
exp(x)	e ning x chi darajasi (eskponetsial f-ya)	exp(1.0) = 2.71828
fabs(x)	x ning absolut qiymati	x>0 => abs(x) = x x=0 => abs(x) = 0.0 x<0 => abs(x) = -x
floor(x)	x ni x dan kichik bo'lgan eng katta butun songacha yahlitlaydi	floor(4.8) = 4.0 floor(-15.9) = -16.0
fmod(x,y)	x/y ning qoldig'ini kasr son tipida beradi	fmod(7.3,1.7) = 0.5
log(x)	x ning natural lagorifmi (e asosiga ko'ra)	log(2.718282) = 1.0
log10(x)	x ning 10 asosiga ko'ra lagorifmi	log10(1000.0) = 3.0
pow(x,y)	x ning y chi darajasini beradi	pow(3,4) = 81.0 pow(16,0.25) = 2
sin(x)	x ning trigonometrik sinusi (x radianda)	sin(0.0) = 0.0
sqrt(x)	x ning kvadrat ildizi	sqrt(625.0) = 25.0
tan(x)	x ning trigonometrik tangensi (x radianda)	tan(0.0) = 0

Foydalanuvchi funksiyalari.

Funktsiyalarni ta'riflash va ularga murojaat kilish. Funktsiya ta'rifida funktsiya nomi, tipi va formal parametrlar ro’yhati ko’rsatiladi. Formal parametrlar nomlaridan

tashqari tiplari ham ko’rsatilishi shart. Formal parametrlar ro’yhati funktsiya signaturasi deb ham ataladi. Funktsiya ta'rifi umumiy kurinishi kuyidagichadir:

Funktsiya tipi funktsiya nomi(formal_parametrlar_ta'rifi)

Formal parametrlarga ta'rif berilganda ularninga boshlang’ich qiymatlari ham ko’rsatilishi mumkin. Funktsiya qaytaruvchi ifoda qiymati funktsiya tanasida return <ifoda> ; operatori orqali ko’rsatiladi.

Misol:

```
Float  min(float, float b)
{ if (a<b) return a;
      return b;
}
```

Funktsiyaga murojaat qilish quyidagicha amalga oshiriladi:

Funktsiya nomi (haqiqiy parametrlar ruyhati)

Haqiqiy parametr ifoda ham bo’lishi mumkin. Haqiqiy parametrlar qiymati hisoblanib mos formal parametrlar o’rnida ishlatiladi.

Misol uchun yuqoridagi funktsiyaga qo’yidagicha murojaat qilish mumkin:

Int x=5,y=6,z; z=min(x,y) eki int z=Min(5,6) eki int x=5; int z=min(x,6)

Funktsiya ta'rifida formal parametrlar initsializatsiya qilinishi, ya'ni boshlang’ich qiymatlar ko’rsatilishi mumkin. Funktsiyaga murojaat qilinganda biror haqiqiy parametr ko’rsatilmasa, uning urniga mos formal parametr ta'rifida ko’rsatilgan boshlang’ich qiymat ishlatiladi.

Misol uchun:

```
Float  min(float a=0.0, float b)
{ if (a<b) return a;
      return b;
}
```

Bu funktsiyaga yuqorida ko’rsatilgan murojaat usullaridan tashqari quyidagicha murojaat qilish mumkin:

Int y=6,z; z=min(,y) eki int z=Min(,6);

Agar funktsiya hech qanday qiymat qaytarmasa uning tipi void deb ko’rsatiladi.

Misol uchun:

```
Void print;
{ Cout<<(“\n Salom!);
};
```

Bu funktsiyaga Print; shaklida murojjat qilish ekranga Salom! Yozilishiga olib keladi.

Qiymat qaytarmaydigan funktsiya formal parametrlarga ega bo’lishi mumkin:

```
Void Pint_Baho(Int baho);
{
Switch(baho)
{case 2:Cout<<(“\n  emon”);break;
case 3:Cout<<(“\n  urta”);break;
case 4:Cout<<(“\n  yahshi”);break;
case 5:Cout<<(“\n  a'lo”);break;
default: Cout<<(“\n  baho notugri kiritilgan”);
};
```

Bu funktsiyaga Print_Baho(5) shaklida murojaat qilish ekranga a'lo so’zi yozilishiga olib keladi.

Agar programmada funktsiya ta'rifi murojaatdan keyin berilsa, yoki funktsiya boshqa faylda joylashgan bo’lsa, murojjatdan oldin shu funktsiyaning prototipi joylashgan bulishi kerak. Prototip funktsiya nomi va formal parametrlar tiplaridan iborat bo’ladi. Formal parametrlar nomlarini berish shart emas.

Misol uchun y=min(a,b)+2*max(c,d) ifodani hisoblashni kuramaz:

```
#Include <iostream.h>
int max(int a,int b)
{if (a<b) return a;else return b};
void main()
{int a,b,c,d,y;
 int min(int ,int);
 Cin>>(“\n %f%f%f%f”,&a,&b,&c,&d);
y=min(a,b)+2*max(c,d);
Cout<<(“\n %f”,y);
};
```

```
int min(int a,int b)
{if (a<b) return b;else return a};
```

Funktsiyaga parametrlar uzatish. Funktsiyaga parametrlar qiymat buyicha uzatiladi va quyidagi bosqichlardan iborat bo'ladi:

1. Funktsiya bajarishga tayyorlanganda formal parametrlar uchun hotiradan joy ajratiladi, ya'ni formal parametrlar funktsiyalarning ichki parametrlariga aylantiriladi. Agar parametr tipi float bo'lsa double tipidagi ob'ektlar hosil buladi, char va shortint bulsa int tipidagi ob'ektlar yaratiladi.
2. Haqiqiy parametrlar sifatida ishlatilgan ifodalar qiymatlari hisoblanadi.
3. Haqiqiy parametrlar ifodalar qiymatlari formal parametrlar uchun ajratilgan hotira qismlariga yoziladi. Bu jarayonda float tipi double tipiga, char va shortint tiplari int tipiga keltiriladi.
4. Funktsiya tanasi ichki ob'ektlar – parametrlar yordamida bajariladi va qiymat chaqirilgan joyga qaytariladi.
5. Haqiqiy parametrlar qiymatlariga funktsiya hech qanday ta'sir o'tkazmaydi.
6. Funktsiyadan chiqishda formal parametrlar uchun ajratilgan hotira qismlari bo'shatiladi.

C ++ tilida chaqirilgan funktsiya chaqiruvchi funktsiyadagi o'zgaruvchi qiymatini uzgartira olmaydi. U faqat o'zining vaqtinchalik nushasini o'zgartirishi mumkin holos.

Qiymat bo'yicha chaqirish qulaylik tug'diradi. Chunki funktsiyalarda kamroq o'zgaruvchilarni ishlatishga imkon beradi. Misol uchun shu hususiyatni aks ettiruvchi POWER funktsiyasi variantini keltiramiz:

```
power(x,n)  /* raise  x  n-th power; n > 0;
version 2 */
int x,n;
int p;
for (p = 1; n > 0; --n)
p = p * x;
return (p);
```

Argument N vaqtinchalik o’zgaruvchi sifatida ishlatiladi. Undan to qiymati 0 bo’lmaguncha bir ayriladi. N funktsiya ichida o’zgarishi funktsiyaga murojjat qilingan boshlang’ich qiymatiga ta'sir qilmaydi.

Rekursiv funktsiyalar. Rekursiv funktsiya deb o’ziga uzi murojjat qiluvchi funktsiyaga aytiladi. Misol uchun faktorialni hisoblash funktsiyasini keltiramiz:

```
Long fact(int k)
{if (k<0) return 0;
if (k==0) return 1;
return k*fact(k-1);
}
```

Manfiy argument uchun funktsiya 0 qiymat qaytaradi. Parametr 0 ga teng bo’lsa funktsiya 1 qiymat qaytaradi. Aks holda parametr qiymat birga kamaytirilgan holda funktsiyaning o’zi chaqiriladi va uzatilgan parametrga ko’paytiriladi. Funktsiyaning uz uzini chaqirish formal parametr qiymati 0 ga teng bo’lganda tuhtatiladi. Keyingi misolimizda ihtiyoriy haqiqiy sonning butun darajasini hisoblash rekursiv funktsiyasini keltiramiz.

```
Double expo(double a, int n)
{ if (n==0) return 1;
if (a==0.0) return 0;
if (n>0) return a*expo(a,n-1);
if(n<0) return expo(a,n+1)/a;
}
```

Misol uchun funktsiyaga expo(2.0,3) shaklda murojaat qilinganda rekursiv ravishda funktsiyaning ikkinchi parametri kamaygan holda murojjatlar hosil buladi: Expo(2.0,3),expo(2.0,2),expo(2.0,1),expo(2.0,0). Bu murojaatlarda quyidaga kupaytma hisoblanadi: 2.0*2.0*2.0*1 va kerakli natija hosil qilinadi.

Shuni kursatib o’tish kerakki bu funktsiyamizda noaniqlik mavjuddir ya'ni 0.0 ga teng sonning 0 chi darajasi 0 ga teng bo’ladi. Matematik nuqtai nazardan bo’lsa bu holda noaniqlik kelib chiqadi. Yuqoridagi sodda misollarda rekursiyasiz iterativ funktsiyalardan foydalanish maqsadga muvofiqdir.

Masalan darajani hisoblash funktsiyani quyidagicha tuzish mumkin:

```
Double expo(double a, int n)
{ if (n==0) return 1;
if (a==0.0) return 0;
int k=(n>0)?n:-n;
for(double s=1.0,int i=0;i<k;i++,s*=a);
if (n>0) return s else return 1/s;
}
```

Rekursiyaga misol sifatida sonni satr shaklida chiqarish masalasini ko'rib chiqamiz. Son raqamlari teskari tartibda hosil buladi. Birinchi usulda raqamlarni massivda saqlab so'ngra teskari tartibda chiqarishdir.

Rekursiv usulda funktsiya har bir chaqiriqda bosh raqamlardan nusha olsih uchun o'z o'ziga murojaat qiladi, so'ngra ohirgi rakamni bosib chiqaradi.

```
printd(n)   /* print n in decimal (recursive)*/
int n;
(
int i;
if (n < 0)
putchar('-');
n = -n;
if ((i = n/10) != 0)

printd(i);
putchar(n % 10 + '0');
)
```

PRINTD(123) chaqiriqda birinchi funktsiya PRINTD N = 123 qiymatga ega. U 12 qiymatni ikkinchi PRINTD ga uzatadi, boshqarish o'ziga qaytganda 3 ni chiqaradi.

Funksiyalar dastur yozishda katta qulayliklar beradilar. Lekin mashina saviyasida funksiyani har gal chaqirtirish qo'shimcha ish bajarilishiga olib keladi.

Registrlardagi o'zgaruvchilar o'zgartiriladi, lokal yozgaruvchilar quriladi, parametr sifatida berilgan argumentlar funksiya stekiga o'ziladi.Bu, albatta, qo'shimcha vaqt oladi. Umuman aytganda, hech funksiyasiz yozilgan dastur, yani hamma amallari faqat main() funksiyasi ichida bajariladigan monolit programma, bir necha funksiyalarga ega, ayni ishni bajaradigan dasturdan tezroq ishlaydi. Funksiyalarning bu noqulayligini tuzatish uchun inline (satr ichida) ifodasi funksiya e'loni bilan birga qo'llaniladi. inline sifatli funksiyalar tanasi dasturdagi ushbu funksiya chaqirig'I uchragan joyga qo'yiladi. inline deb ayniqsa kichik funksiyalarni belgilash effektivdir. inline ning o'ziga yarasha kamchiligi ham bor, inline qo'llanilganda dastur hajmi oshdi. Agar funksiya katta bo'lsa va dastur ichida ko'p marotaba chaqirilsa, programma hajmi juda kattalashib ketishi mumkin. Oddiy, inline sifati qo'llanilmagan funksiyalar chaqirilish mehanizmi quyidagicha bo'ladi. Dastur kodining ma'lum bir yerida funksiya tanasi bir marotaba aniqlangan bo'ladi. Funksiya chaqirig'i uchragan yerda funksiya joylashgan yerga ko'rsatkich qo'yiladi. Demak, funksiya chaqirig'ida dastur funksiyaga sakrashni bajaradi. Funksiya o'z ishini bajarib bo'lgandan keyin dastur ishlashi yana sakrash joyiga qaytadi. Bu dastur hajmini ihchamlikda saqlaydi, lekin funksiya chaqiriqlari vaqt oladi.

Kompilyator inline ifodasini inobatga olmasligi mumkin, yani funksiya oddiy holda kompilyatsiya qilinishi mumkin. Va ko'pincha shunday bo'ladi ham. Amalda faqat juda kichik funksiyalar inline deya kompilyatsiya qilinadi.

inline sifatli funksiyalarga o'zgartirishlar kiritilganda ularni ishlatgan boshqa dastur bloklari ham qaytadan kompilyatsiya qilinishi kerak. Agar katta proyektlar ustida ish bajarilayatgan bo'lsa, bu ko'p vaqt olishi mumkin. inline funksiyalar C da qo'llanilgan # define makrolari o'rnida qo'llanilish uchun mo'ljallangan. Makrolar emas, balki inline funksiyalar qo'llanilishi dastur yozilishini tartibga soladi. Makro funksiyalarni keyinroq o'tamiz.

```
//inline ifodasining qo'llanilishi
 # include <iostream.h>
   inline int sum(int a, int b);//funksiya prototipi
 int main() {
   int j = -356,
```

```
      i =  490;
    cout << "j + i = " << sum(j,i) <<endl;
    return (0);
}
int sum(int a, int b){ //funksiya aniqlanishi
  return( a + b );
}
```

Ekranda:

```
j + i = 134
```

Funksiyalar va unig shablonlari.

Ba'zi bir funksiyalar ko'pincha bir hil qiymatli argumentlar bilan chaqirilishi mumkin. Bu holda, agar biz funksiya argumentlariga ushbu ko'p qo'llaniladigan qiymatlarni bersak, funksiya argumentsiz chaqirilganda bu qiymatlar kompilyator tomonidan chaqiriqqa kiritiladi. Berilgan qiymatlar funksiya prototipida berilsa kifoyadir.

Berilgan qiymatli argumentlar parametrlar ro'hatida eng o'ng tomonda yozilishi kerak. Buning sababi shuki, agar argument qiymati tashlanib o'tilgan bo'lsa, va u o'ng tomonda joylashmagan bo'lsa, biz bo'sh vergullani qo'yishimizga to'g'ri keladi, bu esa mumkin emas. Agar bir necha berilgan qiymatli argumentlar bor bo'lsa, va eng o'ngda joylashmagan argument tushurilib qoldirilsa, undan keyingi argumentlar ham yozilmasligi kerak.

Bir misol keltiraylik.

```
//Berilgan qiymatli parametrlar bilan ishlash
# include <iostream.h>
int square(int = 1, int = 1); // ...(int a=1, int b=1)...
                        // yuqoridagi kabi o'zgaruvchilar otini ham
                        // berishimiz mumkin
int main()
{
  int s = 3, t = 7;
  cout << "Paremetrsiz: " << square()<< endl;
```

```
  cout << "Bitta parametr (ikkinchisi) bilan:" << square(t) << endl;
  cout << "Ikkita parametr bilan:" << square(s,t) << endl;
  return (0);
}
 int square(int k, int g){
  return ( k * g );
}
```

Ekranda:

Parametrsiz: 1

Bitta parametr (ikkinchisi) bilan: 7

Ikkita parametr bilan: 21

Bir xil ismli bir necha funksiya e'lon qilinishi mumkin. Bu C++ dagi juda kuchli tushunchalardandir. Yuklatilgan funksiyalarning faqat kirish parametrlari farqli bo'lishi yetarlidir. Qaytish parametri yuklatilishda ahamiyati yo'qdir. Yuklangan funksiyalar chaqirilganda, qaysi funksiyani chaqirish kirish parametrlarining soniga, ularning tipiga va navbatiga bog'liqdir. Yani ism yuklanishida funksiyaning imzosi rol o'ynidi. Agar kirish parametrlari va ismlari ayni funksiyalarning farqi faqat ularning qaytish qiymatlarida bo'lsa, bu yuklanish bo'lmaydi, kompilyator buni hato deb e'lon qiladi. Funksiya yuklanishi asosan ayni ishni yoki amalni farqli usul bilan farqli ma'lumot tiplari ustida bajarish uchun qo'llaniladi. Masalan bir fazoviy jismning hajmini hisoblash kerak bo'lsin. Har bir jismning hajmi farqli formula yordamida, yani farqli usulda topiladi, bir jismda radius tushunchasi bor bo'lsa, boshqasida asos yoki tomon tushunchasi bor bo'ladi, bu esa farqli ma'lumot tiplariga kiradi. Lekin amal ayni - hajmni hisoblash. Demak, biz funksiya yuklanishi mehanizmini qo'llasak bo'ladi. Bir hil amalni bajaruvchi funksiyalarni ayni nom bilan atashimiz esa, dasturni o'qib tushunishni osonlashtiradi. Kompilaytor biz bergan funksiya imzosidan (imzoga funksiya ismi va kirish parametrlari kiradi, funksiyaning qaytish qiymati esa imzoga kirmaydi) yagona ism tuzadi, dastur ijrosi davruda esa funksiya chaqirig'idagi argumentlarga qarab, kerakli funksiyani chaqiradi. Yangi ismni tuzish operatsiyasi ismlar dekoratsiyasi deb ataladi. Bu tushunchalarni misolda ko'rib chiqaylik. // Yuklatilgan funksiyalarni qo'llash

```
# include <iostream.h>
# include <math.h>
    // Yangi ismlar sohasini aniqladik
   namespace
   mathematics {
   const double Pi = 3.14159265358979;
 double hajm(double radius); // sharning hajmi uchun - 4/3 * Pi * r^3
double hajm(double a, double b, double s) // kubning hajmi uchun - abc
 }
 using namespace mathematics;
 int main()
{
   double d = 5.99; // sharning radiusi
   int x = 7, y = 18, z = 43;
   cout << "Sharninig hajmi: " << hajm(d) << endl;
   cout << "Kubning hajmi: " << hajm(x,y,z) << endl;
   return (0);
}
 double mathematics::hajm(double radius) {
  return (  (Pi * pow(radius,3) * 4.0) / 3.0  );
}
 double mathematics::hajm(double a, double b, double c) {
   return ( a * b * c );
}
```

Ekranda:

Sharning hajmi: 900.2623

Kubning hajmi: 5418

Yuqoridagi dasturda yangi ismlar sohasini aniqladik, unda Pi konstantasini e'lon qildik. shaqning hajmini hisoblashda standart kutubhonadagi pow() funksiyasini ishlatdik, shu sababli <math.h> e'lon faylini # include ifodasi bilan kiritdik. Ismlar sohasida joylashgan funksiyalarni aniqlash uchun, yani

ularning tanasini yozish uchun biz ilarning to'liq ismini berishimiz kerak. Albatta, agar funksiya ismlar sohasining ichida aniqlangan bo'lsa, tashqarida boshqattan yozib o'tirishning hojati yo'q. hajm() funksiyalarining to'liq ismi mathematics::hajm(...) dir. :: operatori sohalarni bog'lovchi operatordir. Yangi ismlar sohasini faqatgina misol tariqasida berdik, uni funksiya yuklanishlari bilan hech qanday aloqasi yo'qdir. Funksiya ismlari yuklanishi, ko'rib turganimizdek, juda qulay narsadir.

Funksiya yuklanishini qo'llaganimizda, funksiyalar argumentlarining berilgan qiymatlarini ehtiyotkorlik bilan qo'llashimiz lozim. Masalan bizda ikkita funksiyamiz bor bo'lsin.

```
foo(int k = 0); // berilgan qiymati 0
foo();
```

Bu ikki funksiya yuklatilgan. Lekin agar biz birinchi funksiyani dasturda argumentsiz chaqirsak, kompilyator qaysi funksiyani chaqirishni bilmaydi, va shu sababli hato beradi. Biroq bu deganimiz funksiya yuklanishi bilan berilgan qiymatlar qo'llanilishi mumkin emas deganimiz emas, eng muhimi funksiya chaqirig'ini ikki hil tushunish bo'lmasligi kerak.

Masalaning qo’yilishi.

Misol. Xaqiqiy qiymatli A={aij}, (i,j=1,2,..,n) matritsa berilgan. Xamma manfiy elementlarni ularning kvadratlari, musbat elementlarni ikkilangan qiymatlari chiqarilsin.

Berilgan masalaning matematik tahlili va yechish usuli.

Bizga haqiqiy qiymatli A={a_{ij}}, (i,j=1,2,..,n) matritsa berilgan bo’lsin.

Ya’ni :

$$A = \begin{matrix} -22 & 39 & 51 & -10 \\ 333 & -6 & -9 & -14 \\ 15 & -28 & 7 & 11 \end{matrix}$$

Masalaning shartiga muvofiq avval manfiy elementlarini ajratib olamiz. Ular :

-22 -10 -6 -9 -14 -28

elementlarninig darajalarini hisoblaymiz. Darajalari :

484 100 36 81 196 784

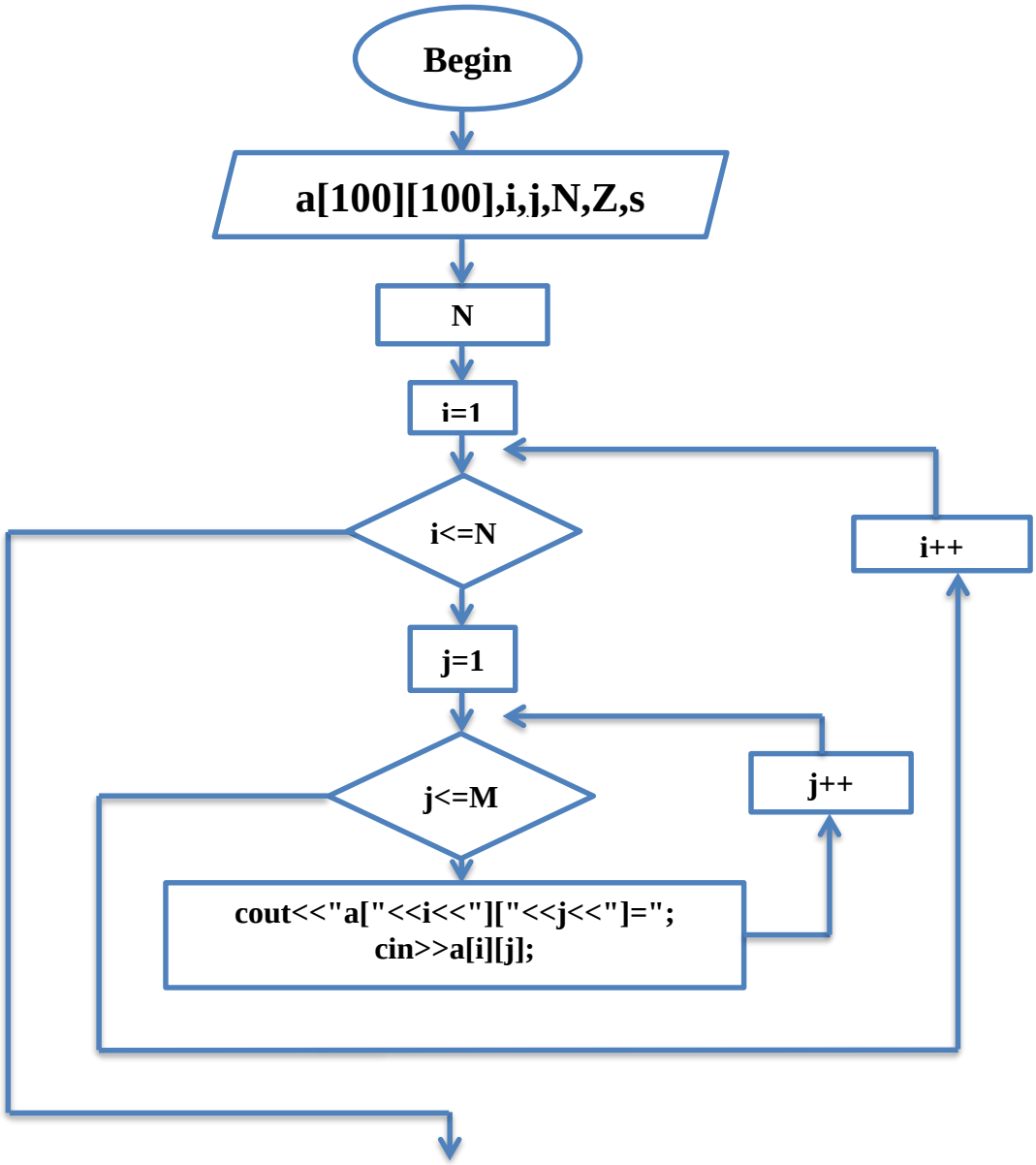
Endi musbat sonlarni ajratib olamiz. Ular :

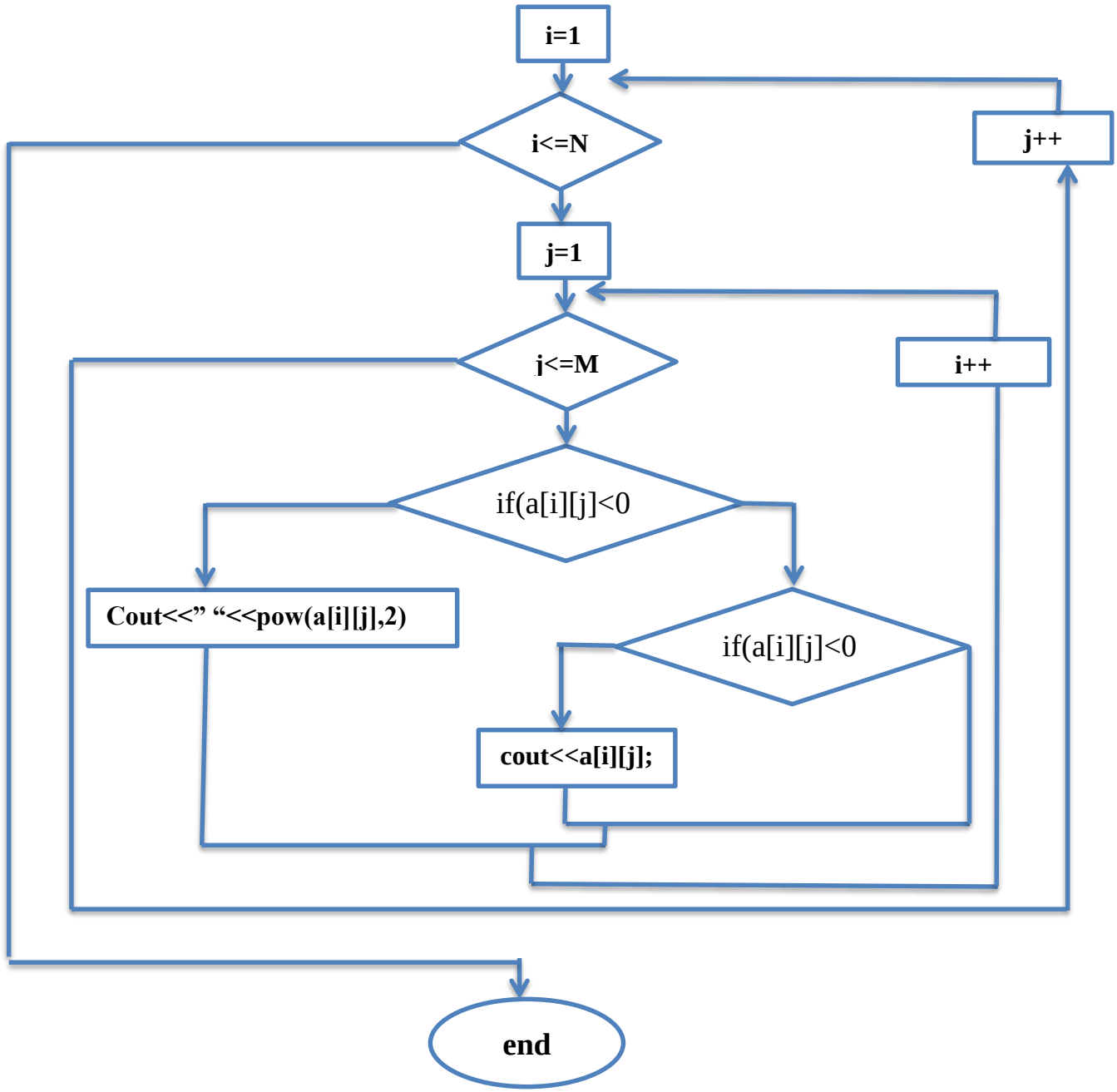
39 51 333 15 7 11

Ushbu musbat elementlarni ikkilangan qiymaini chiqaramiz. Ya’ni ikkilangan deganda musbat sonlarni 2 ga ko’paytirib olamiz:

78 102 666 30 14 22

Masalani yechish algoritmi va blok sxemasi.

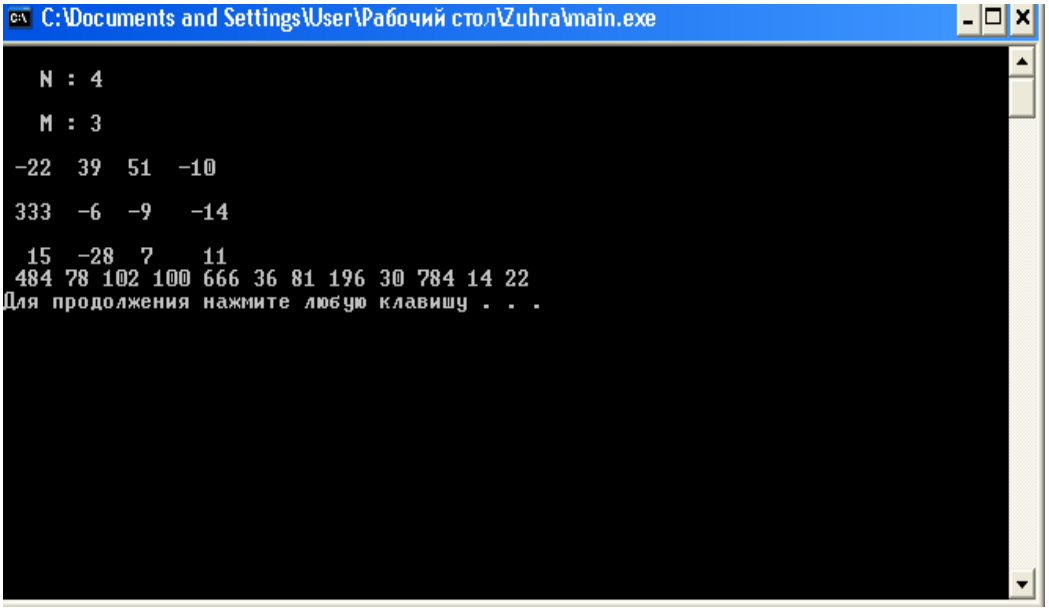



i=1
i<=N
j++
j=1
j<=M
i++
if(a[i][j]<0
Cout<<" "<<pow(a[i][j],2)
if(a[i][j]<0
cout<<a[i][j];
end

Masalaning dasturi.

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
   int a[100][100],i,j,N,M,almash;
     cout<<"N :";
       cin>>N;
     cout<<"M :";
        cin>>M;
         for(i=1; i<=N; i++)
         for(j=1; j<=M; j++)
           {
            cin>>a[i][j];
           }
            for(i=1; i<=N; i++)
            for(j=1; j<=M; j++)
              if(a[i][j]<0)
                 {
                   cout<<" "<<pow(a[i][j],2);
                 }
                     else if (a[i][j]>0)
                       {
                         cout<<" "<<(a[i][j]*2);
                       }
          cout<<endl;
   system("PAUSE");
   return 0;
}
```

Olingan natijalar tahlili.



Xulosa

Xulosa qilib shuni aytish mumkinki tuzilayotgan har bir dasturni funksiyalarsiz tassavvur qilib bo’lmaydi. Bulardan tashqari C++ da funksiya imzosi (function signature) tushunchasi bor. Funksiya imzosiga funksiya nomi, kiruvchi parametrlar tipi, soni, ketma-ketligi kiradi.

Funktsiyalarni ta'riflash va ularga murojaat kilish. Funktsiya ta'rifida funktsiya nomi, tipi va formal parametrlar ro’yhati ko’rsatiladi. Formal parametrlar nomlaridan tashqari tiplari ham ko’rsatilishi shart. Formal parametrlar ro’yhati funktsiya signaturasi deb ham ataladi. Funktsiya ta'rifi umumiy kurinishi kuyidagichadir:

Funktsiya tipi funktsiya nomi(formal_parametrlar_ta'rifi)

Formal parametrlarga ta'rif berilganda ularninga boshlang’ich qiymatlari ham ko’rsatilishi mumkin. Funktsiya qaytaruvchi ifoda qiymati funktsiya tanasida return <ifoda> ; operatori orqali ko’rsatiladi.

Funksiyalar dastur yozishda katta qulayliklar beradilar. Lekin mashina saviyasida funksiyani har gal chaqirtirish qo'shimcha ish bajarilishiga olib keladi. Registrlardagi o'zgaruvchilar o'zgartiriladi, lokal yozgaruvchilar quriladi, parametr sifatida berilgan argumentlar funksiya stekiga o'ziladi.Bu, albatta, qo'shimcha vaqt oladi.

Umuman aytganda, hech funksiyasiz yozilgan dastur, yani hamma amallari faqat main() funksiyasi ichida bajariladigan monolit programma, bir necha funksiyalarga ega, ayni ishni bajaradigan dasturdan tezroq ishlaydi. Funksiyalarning bu noqulayligini tuzatish uchun inline (satr ichida) ifodasi funksiya e'loni bilan birga qo'llaniladi.

Foydalanilgan adabiyotlar

1. O.Shukurov, F.Qorayev “Programmalardan masalalar to’plami” Qarshi-2008
2. Aripov M.M, Muhammadiyev J.O’. “ Informatika, Informatsion texnalogiyalar” Toshkent-2004
3. M. Aripov, A.Xaydarov “Informatika asoslari” O’qituvchi 2002
4. C++ tilida dasturlash asoslari uslubiy qo’llanma Sayfiyev J.F.
5. C++ Tiliga kirish uslubiy qo’llanma Buxoro -2005
6. K.Aslanov C++ dan o’quv qo’llanma Toshkent-2010
7. C++ da dasturlash P.Ya.Xaydarova TATU Aloqachi Tshkent 2008
8. Axmedov.A, Tayloqov N, “Informatika” Toshkent-2002
9. U.Yo’ldoshev, P.Boqiyev, F.Zokirova “Informatika” O’zbekiston-2003
10. Xaljigitov A.A, Madraximov Sh.F, Adambayev U.E, Eshboyev E.A, “ Informatika va Programmalash” Toshkent O’zMU, 2005-148.