O’zbekiston Respublikasi Aloqa Axborotlashtirish va Telekomunikatsiya Texnologiyalari Davlat Qo’mitasi Toshkent Axborot Texnalogiyalari Universiteti Qarshi Filiali

“Axborot texnologiyalari”

kafedrasi

“Informatika va axborot texnologiyalari”

fanidan

KURS ISHI

Mavzu:**C++ tilida ifodalar va operatorlar**

Guruh:IKT-22

Talaba: Ibragimova Dilafruz

2014- Qarshi

MUNDARIJA

Kirish

C++ tilining asosiy tushunchalaridan biri nomlangan xotira qismi – ob'ekt tushunchasidir. Ob'ektning xususiy holi bu o'zgaruvchidir. O'zgaruvchiga qiymat berilganda unga ajratilgan xotira qismiga shu qiymat kodi yoziladi. O'zgaruvchi qiymatiga nomi orqali murojaat qilish mumkin, xotira qismiga esa faqat addresi orqali murojat qilinadi. O'zgaruvchi nomi bu erkin kiritiladigan identifikatordir. O'zgaruvchi nomi sifatida xizmatchi so'zlarni ishlatish mumkin emas.

Har qanday dastur funktsiyalar ketma -ketligidan iborat bo'ladi.Funksiyalar sarlavha va funktsiya tanasidan iborat bo'ladi. Funktsiya sarlavhasiga void main() ifoda misol bo'la oladi. Funktsiya tanasi ob'ektlar ta'riflari va operatorlardan iborat bo'ladi.

Har qanday operator (;) nuqtali vergul belgisi bilan tugashi lozim. Quyidagi ifodalar X=0, yoki i++ operatorga aylanadi. Agar ulardan so'ng nuqtali vergul kelsa,

X = 0; i++;

Operatorlar bajariluvchi va bajarilmaydigan operatorlarga ajratiladi. Bajarilmaydigan operator bu izoh operatoridir.

Izoh operatori /* belgisi bilan boshlanib */ belgisi bilan tugaydi. Bu ikki simvol orasida ixtiyoriy jumla yozish mumkin. Kompilyatsiya bu jumlani tekshirib o'tirmaydi. Izoh operatoridan dasturni tushunarli qilish maqsadida izohlar kiritish uchun foydalaniladi.

Bajariluvchi operatorlar o'z navbatida ma'lumotlarni o'zgartiruvchi va boshqaruvchi operatorlarga ajratiladi.

Men bu kurs ishimda C++ dasturlashtirish tilida C++ tilida ifodalar va operatorlar haqida va matematik masalalarni yechish haqida ma'lumot berishga harakat qildim.

Ifodalar va operatorlar

Ifoda.

C++ tilida ifodalar biror bir hisoblash natijasini qaytaruvchi boshqa ifodalar ketma-ketligini boshqaradi yoki hech nima qilmaydi (nol ifodalar).

C++ tilida barcha ifodalar nuqtali vergul bilan yakunlanadi. Ifodaga misol qilib o‘zlashtirish amalini olish mumkin.

```
x=a+b;
```

Algebradan farqli ravishda bu ifoda x a+v ga teng ekanligini anglatmaydi. Bu ifodani quyidagicha tushinish kerak:

a va v o‘zgaruvchilarni qiymatlarini yig‘ib natijasini x o‘zgaruvchiga beramiz yoki x o‘zgaruvchiga a+v qiymatni o‘zlashtiramiz. Bu ifoda birdaniga ikkita amalni bajaradi, yig‘indini hisoblaydi va natijani o‘zlashtiradi. Ifodadan so‘ng nuqtali vergul qo‘yiladi. (=) operatori o‘zidan chap tomondagi operandga o‘ng tomondagi operandlar ustida bajarilgan amallar natijasini o‘zlashtiradi.

Bo‘sh joy (probel) belgisi.

Bo‘sh joy belgilariga nafaqat probel, balki yangi satrga o‘tish va tabulyatsiya belgilari ham kiradi. Yuqorida keltirilgan ifodani quyidagicha ham yozish mumkin:

```
x     = a
+       b   ;
```

Bu variantda keltirilgan ifoda ko‘rimsiz va tushunarsiz bo‘lsa ham to‘g‘ridir.

Bo‘sh joy belgilari dasturning o‘qilishliligini ta’minlaydi.

Bloklar va kompleks ifodalar.

Ba’zan dastur tushunarli bo‘lishi uchun o‘zaro mantiqiy bog‘langan ifodalarni blok deb ataluvchi komplekslarga birlashtirish qulaydir. Blok ochiluvchi figurali qavs ({) bilan boshlanadi va yopiluvchi figurali qavs (}) bilan tugaydi. Blok ochilganda va yopilganda nuqtali vergul qo‘yilmaydi.

```
{
  temp= a;
  a = b;
  b = temp;
}
```

Bu blok xuddi bir ifodadek bajariladi, u a va b o‘zgaruvchilar qiymatlarini almashtiradi.

Amallar.

Bajarilishi natijasida biror bir qiymat qaytaradigan barcha ifodalar C++ tilida amallar deyiladi. Amallar albatta biror bir qiymat qaytaradi. Masalan, 3+2 amali 5 qiymatni qaytaradi.

Operatorlar.

Operator - bu qandaydir amalni bajarish tug‘risida kompilyatorga uzatiladigan literaldir. Operatorlar operandlarga ta’sir qiladi. C++ da operandlar deb alohida literallar va butun ifodalar tushuniladi.

C++ tilida ikki ko‘rinishdagi operatorlar bor:

o‘zlashtirish operatorlari

matematik operatorlar

O‘zlashtirish operatori.

O‘zlashtirish operatori (=) o‘zidan chap tomonda turgan operand qiymatini tenglik belgisidan o‘ng tomondagilarni hisoblangan qiymatiga almashtiradi. Masalan,

x = a+b;

ifodasi x operandga a va b o‘zgaruvchilarni qiymatlarini qo‘shishdan hosil bo‘lgan natijani o‘zlashtiradi.

O‘zlashtirish operatoridan chapda joylashgan operand adresli operand yoki l–kiymat (chap-chap so‘zidan olingan) deyiladi. O‘zlashtirish operatoridan o‘ngda joylashgan operand operatsion operand yoki r–kiymat deyiladi.

O‘zgarmaslar faqatgina r–qiymat bo‘lishi mumkin va hech qachon adresli operand bo‘la olmaydi, chunki dasturning bajarilishi jarayonida o‘zgarmas qiymatini o‘zgartirib bo‘lmaydi.

35 = x // noto’g’ri!

l–qiymat esa r–qiymat bo‘lishi mumkin.

Matematik operatorlar.

C++ tilida 5 ta asosiy matematik operatorlar qo‘llaniladi: qo‘shish (+), ayirish (-), ko‘paytirish (*), butun songa bo‘lish (\) va modul bo‘yicha bo‘lish (%)(qoldiqni olish).

Ishorasiz butun sonlarni ayirishda, agarda natija manfiy son bo‘lsa g‘ayrioddiy natija beradi. Buni misoldan ko‘rishimiz mumkin.

Ayirish natijasida butun sonni to‘lib qolishiga misol

```
# include < iostream.h >
int main()
{
```

```
unsiegned int ayirma
unsisgned int kattaSon = 100;
unsigned int kichikSon = 50;
ayirma = kattaSon – kichikSon;
cout << “Ayirma“:<< ayirma<< “ ga teng\n”;
ayirma = kichikSon - kattaSon ;
cout << “Ayirma“:<< ayirma<< “ ga teng\n”;
endl;
return 0;
}
```

HATIJA:

Ayirma: 50 ga teng

I.C++ dasturlash tilida operatorlar.

Birinchi elektron hisoblash mashinalari paydo bo`lishi bilan dasturlash tillari evolyutsiyasi boshlanadi. Dastlabki kompyuterlar ikkinchi jahon urushi vaqtida artilleriya snaryadlarining harakat traektoriyasini hisob-kitob qilish maqsadida qurilgan edi. Oldin dasturchilar eng sodda mashina tilini o`zida ifodalovchi kompyuter komandalari bilan ishlaganlar. Bu komandalar nol va birlardan tashkil topgan uzun qatorlardan iborat bo`lar edi. Keyinchalik, insonlar uchun tushunarli bo`lgan mashina komandalarini o`zida saqlovchi (masalan, ADD va MOV komandalari) assembler tili yaratildi. Shu vaqtlarda BASIC va COBOL singari yuqori sathli tillar ham paydo bo`ldiki, bu tillar tufayli so`z va gaplarning mantiqiy konstruktsiyasidan foydalanib dasturlash imkoniyati yaratildi. Bu komandalarni mashina tiliga interpretatorlar va kompilyatorlar ko`chirar edi. Interpretator dasturni o`qish jarayonida uning komandalarini ketma - ket mashina tiliga o`tkazadi. Kompilyator esa yaxlit programma kodini biror bir oraliq forma - obyekt fayliga o`tkazadi. Bu bosqich kompilyatsiya bosqichi deyiladi. Bundan so`ng kompilyator obyektli faylni bajariluvchi faylga aylantiradigan kompanovka dasturini chaqiradi.

Interpretatorlar bilan ishlash osonroq, chunki dastur komandalari qanday ketma - ketlikda yozilgan bo`lsa shu tarzda bajariladi. Bu esa dastur

bajarilishini nazorat qilishni osonlashtiradi. Faqatgina kompilyatsiya qilingan fayl tezroq bajariladi, chunki bundagi komandalar kompilyatsiya jarayonida mashina tiliga o`tkazilgan bo`ladi.

C++ kabi kompilyatsiya qiluvchi dasturlash tillarini yana bir afzalligi hosil bo`lgan dastur kompyuterda kompilyatorsiz ham bajarilaveradi. Interpretatsiya qiluvchi tillarda esa tayyor dasturni ishlatish uchun albatta mos interpretator dasturi talab qilinadi.

Ayrim tillarda (masalan, VISUAL BASIC) interpretator rolini dinamik kutibxonalar bajaradi. Java tilining interpretatori esa virtual mashinadir (Virtual Machine, yoki VM). Virtual mashinalar sifatida odatda brouzer (Internet Explorer yoki Netscape) lar qo`llaniladi.

Ko`p yillar davomida dasturlarning asosiy imkoniyati uning qisqaligi va tez bajarilishi bilan belgilanar edi. Dasturni kichikroqqilishga intilish kompyuter xotirasini juda qimmatligi bilan bog’liq bo`lsa, uning tez bajarilishiga qiziqish protsessor vaqtining qimmatbaholigiga bog’liq edi. Lekin kompyuterlarning narxi tushishi bilan dastur imkoniyatini baholash mezoni o`zgardi.Ekspluatatsiyaning oddiyligi, konkret masalani yechish bilan bog’liq bo`lgan talabni ozroq o`zgarishiga, dasturni ortiqcha chiqimlarsiz oson moslashtirish bilan izohlanadi.

C++ tilida operatorlar

Shartli operator. Shartli operator ikki ko'rinishda ishlatilishi mumkin:

```
If (ifoda)
1- operator
else
2- operator
```

Shartli operator bajarilganda avval ifoda hisoblanadi ; agar qiymat rost ya'ni nol dan farqli bo'lsa 1- operator bajariladi. Agar qiymat yolg'on ya'ni nol bo'lsa va else ishlatilsa 2-operator bajariladi. else qism har doim eng yaqin if ga mos qo'yiladi.

```
if( n>0)
if(a>b)
      Z=a;
else
     Z=b;
```

Agar else qismni yuqori if ga mos qo'yish lozim bo'lsa, figurali qavslar ishlatish lozim.

```
if( n>0) {
if(a>b)
      z=a;
}
else
     z=b;
```

Misol tariqasida uchta berilgan sonning eng kattasini aniqlash dasturini ko'ramiz:

```
#include <iostream.h>
void( )
 {   float a,b,c,max);
     Cout <<"\n a="; Cin>>a;
```

```
        Cout <<“\n b=”; Cin>>b;
        Cout <<“\n c=”; Cin>>c;
    if (a>b)
    if (a>c)  max=a else max=c;
    else
    if b>c then max=b else max=c;
       Cout <<“\n” <<max;
     }
```

Keyingi misolda kiritilgan ball va maksimal ball asosida baho aniqlanadi:

```
#include <iostream.h>
void main( )
 {    float ball,max_ball,baho);
      Cout<< “\n ball=”; Cin>>(“%f”,&ball);
      Cout<<“\n max_ball=”; Cin>>max_ball;
      d=ball/max_ball;
if (d>0.85) baho=5 else
if (d>75) baho=4 else
if (d>0.55) then baho=3 else baho=2;
    Cout<<“\n  baho;
 }
```

Kalit bo’yicha tanlash operatori. Kalit bo’yicha o’tish switch operatori umumiy ko’rinishi quyidagicha

```
Switch(<ifoda>)  {
        Case <1-qiymat>:<1-operator>
                        …
break;
                        …
default: <operator>
                        …
case: <n-operator>;
```

}

Oldin qavs ichidagi butun ifoda hisoblanadi va uning qiymati hamma variantlar bilan solishtiriladi. Biror variantga qiymat mos kelsa shu variantda ko'rsatilgan operator bajariladi. Agar biror variant mos kelmasa default orqali ko'rsatilgan operator bajariladi.Break operatori ishlatilmasa shartga mos kelgan variantdan tashqari keyingi variantdagi operatorlar ham avtomatik bajariladi. Default; breakva belgilangan variantlar ixtiyoriy tartibda kelishi mumkin. Default yoki break operatorlarini ishlatish shart emas. Belgilangan operatorlar bo'sh bo'lishi ham mumkin.

Misol tariqasida bahoni son miqdoriga qarab aniqlash dasturini ko'ramiz.

```
Include <iostream.h>
Int baho;
Cin>> 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";
};
}
```

Keyingi misolimizda kiritilgan simvol unli harf ekanligi aniqlanadi:

```
Include <iostream.h>
Int baho; Char c; Cin >> c;
Switch(c)
{case 'a':
case 'u':
case 'o':
case 'i':
Cout <<"\n Kiritilgan simvol unli harf";break;
```

```
default: Cout <<“\n Kiritilgan simvol unli harf emas”;
};
}
```

Break operatori. Ba'zihollarda sikl bajarilishini ixtiyoriy joyda to’xtatishga to’g’ri keladi. Bu vazifani break operatori bajarishga imkon beradi. Bu operator darhol sikl bajarilishini to’xtatadi va boshqaruvni sikldan keyingi operatorlarga uzatadi.

Misol uchun o’quvchining n ta olgan baholariga qarab uning o’qish sifatini aniqlovchi dasturini ko’ramiz. Buning uchun dasturda o’quvchining olgan minimal bahosi aniqlanadi

```
# include <iostream.h>
void main()
{
int i,n,min,p;
while (1)
{Cout<<“Baholar soni=”; Cin>>n;};
if (n>0) break;
Cout<<(“Hato! n>0 bulishi lozim ! \n”);
for (i=1,min=5; i<=n; i++)
{ cin>>p;
if (p<2)||(p>5) {min=0; break};
if (min>p) min=p;
}
if (p<2)||(p>5) cout break;
switch(min)
case 0:cout<<”Baho noto’g’ri kiritilgan”;break;
case 2:cout<<”Talaba yomon o’qiydi”;break;
case 3:cout<<”Talaba o’rtacha o’qiydi”;break;
case 4:cout<<”Talaba yaxshi o’qiydi”;break;
case 5:cout<<”Talaba a'lo o’qiydi”;break;
```

```
}
```

Biz misolda xato kiritilgan n qiymatdan saqlanish uchun while(1)sikl kiritilgan. Agar n>0 bo'lsa Break operatori siklni tuxtatadi va dastur bajarilishi davom etadi. Agar kiritilayotgan baholar chegarada yotmasa min ga 0 qiymat berilib darhol sikldan chiqiladi.

O'tish operatori GO TO.

O'tish operatorining ko'rinishi:

Go to <identifikator>. Bu operator identifikator bilan belgilangan operatorga o'tish kerakligini ko'rsatadi.

Misol uchun goto A1;…;A1:y=5;

Strukturali dasturlashda Go to operatoridan foydalanmaslik maslahat beriladi. Lekin ba'zi hollarda o'tish operatoridan foydalanish dasturlashni osonlashtiradi.

Misol uchun bir necha sikldan birdan chiqish kerak bo'lib qolganda , to`g'ridan-to`gri break operatorini qo'llab bo'lmaydi, chunki u faqat eng ichki sikldan chiqishga imkon beradi.

Quyidagi misolda n ta qatorga n tadan musbat son kiritiladi. Agar n yoki sonlardan biri manfiy bo'lsa, kiritish qaytariladi:

```
# include <iostream.h>
int n, I, j, k;
M1: Cout<<"\n n="; Cin>>n;
If (n<=0) { Cout<<"\n xato! n>0 bo`lishi kerak";
Go to M1;} ;
  M: Cout<<"x sonlarni kiriting \n";
For (I=1; I<=10; I++) {Cout<<"\n I="<< i;
For (j=1 ; j<=10; j++) {Cin>> k;
if (k<=0) goto M;}
    }
```

Bu masalani GOTO operatorisiz hal qilish uchun qo'shimcha o'zgaruvchi kiritish lozimdir.

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

```
   int n, I, j, k;
while 1 {
   Cout<<“\n n=”; Cin>>n;
if (n>0) break;
   Cout<<“\n hato! n>0 bo`lishi kerak”;
   } ;
int M=0;
    While M
{ M=0;
     Cout<<“x sonlarni kiriting \n”;
   For (I=1; I<=10; I++) {
    If (M) break;
    Cout<<(“\n I=%, i);
   For (j=1 ; j<=10; j++) {Cin>>(“%f”, k);
if (k<=0)  {M=1;break;}
   }
```

FOR TAKRORLASH OPERATORI

for strukturasi sanovchi (counter) bilan bajariladigan takrorlashni bajaradi.Boshqa takrorlash bloklarida (while, do/while) takrorlash sonini control qilish uchun ham sanovchini qo'llasa bo'lardi, bu holda takrorlanish sonini oldindan bilsa bo'lardi, ham boshqa bir holatning vujudga kelish-kelmasligi orqali boshqarish mumkin edi. Ikkinchi holda ehtimol miqdori katta bo'ladi. Masalan qo'llanuvchi belgilangan sonni kiritmaguncha takrorlashni bajarish kerak bo'lsa biz while li ifodalar-ni ishlatamiz. for da esa sanovchi ifodaning qiymati oshirilib (kamaytirilib) borilaveradi va chegaraviy qiymatni olganda takrorlanish tugatiladi. for ifodasidan keyingi bitta ifoda qaytariladi. Agar bir necha ifoda takrorlanishi kerak bo'lsa, ifodalar bloki {} qavs ichiga olinadi.

```
 //Ekranda o'zgaruvching qiymatini yozuvchi dastur, for ni ishlatadi.
# include <iostream.h>
int main()
```

```
{
for (int i = 0; i < 5; i++){
cout<< i << endl;
 }
return (0);
}
```

Ekranda:

0

1

2

3

4

for strukturasi uch qismdan iboratdir. Ular nuqtali vergul bilan bir-biridan ajratiladi.for ning ko'rinishi:

```
for( 1. qism ; 2. qism ; 3. qism ){
takror etiladigan blok}
```

1. qism - e'lon va initsalizatsiya.

2. qism - shartni tekshirish (o`zgaruvchini chegaraviy qiymat bilan solishtirish).

3.qism - o'zgaruvchining qiymatini bajarilish ketma-ketligi quyidagichadir:

1. qism bajariladi (faqat bir marta), keyin

2. qismdagi shart tekshiriladi va agar u true bo'lsa takrorlanish bloki ijro ko'radi

3. qismda o'zgaruvchilar o'zgartiriladi, keyin yana ikkinchi qismga o'tiladi.for strukturamizni while struktura bilan almashtirib ko'raylik:

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

Ekranga 10 marta Hello! so'zi bosib chiqariladi. i o'zgaruvchisi 0 dan 9 gacha o'zgaradi. i 10 ga teng bo'lganda esa i < 10 sharti noto'g'ri (false) bo'lib chiqadi va for strukturasi nihoyasiga yetadi. Buni while bilan yozsak:

```
int i = 0;
while ( i<10 ){
cout<< "Hello!" <<endl;
i++;
}
```

Endi for ni tashkil etuvchi uchta qismning har birini alohida ko'rib chiqsak.Birinchi qismda asosan takrorlashni boshqaradigan sanovchi (counter) o'zgaruvchilar e'lon qilinadi va ularga boshlang`ich qiymatlar beriladi (initsalizatsiya). Yuqoridagi dastur misolida buni int i = 0; deb berganmiz. Ushbu qismda bir necha o'zgaruvchilarni e'lon qilishimiz mumkin, ular vergul bilan ajratiladi. Ayni shu kabi uchinchi qismda ham bir nechta o'zgaruvchilarning qiymatini o'zgartirishimiz mumkin. Undan tashqari birinchi qismda for dan oldin e'lon qilingan o'zgaruvchilarni qo'llasak bo'ladi.

Masalan:

```
int k = 10;
int l;
for (int m = 2, l = 0 ; k <= 30 ; k++, l++, ++m) {
cout<< k + m + l;
}
```

Albatta bu ancha sun'iy misol, lekin u bizga for ifodasining naqadar moslashuvchanligi ko'rsatadi.for ning qismlari tushrib qoldirilishi mumkin.

Masalan:

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

ifodasi cheksiz marta qaytariladi. Bu for dan chiqish uchun break operatorini beramiz. Yoki agar sanovchi sonni takrorlanish bloki ichida o'zgartirsak,for ning 3-qismi kerak emas. Misol:

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

Yana qo'shimcha misollar beraylik.

```
for (int y = 100; y >= 0; y-=5){
...
ifoda(lar);
...
}
```

Bu yerda 100 dan 0 gacha 5 lik qadam bilan tushiladi.

```
for(int d = -30; d<=30; d++){
...
ifoda(lar);
...
}
```

60 marta qaytariladi.

for strukrurasi bilan dasturlarimizda yanada yaqinroq tanishamiz. Endi

1-qismda e'lon qilinadigan o'zgaruvchilarning xususiyati haqida bir og'iz aytib o'taylik. Standartga ko'ra bu qismda e'lon qilingan o'zgaruvchilarning qo'llanilish sohasi faqat o'sha for strukturasi bilan chegaralanadi. Ya'ni bitta blokda joylashgan for strukturalari mavjud bo'lsa, ular ayni ismli o'zgaruvchilarni qo'llay olmaydilar. Masalan quyidagilar xatodir:

```
for(int j = 0; j<20 ; j++){...}
...
for(int j = 1; j<10 ; j++){...} //hato!
```

j o'zgaruvchisi birinchi for da e'lon qilinib bo'lindi. Ikkinchi for da ishlatish mumkin emas.Bu masalani yechish uchun ikki xil yo'l tutish mumkin.

Birinchisi bitta blokda berilgan for larning har birida farqli o'zgaruvchilarni qo'llashdir. Ikkinchi yo'l for lar guruhidan oldin sanovchi vazifasini bajaruvchi bir o'zgaruvchini e'lon qilishdir. Va for larda bu o'zgaruvchiga faqat kerakli boshlang'ich qiymat beriladi xolos.

for ning ko'rinishlaridan biri bo'sh tanali for dir.

```
for(int i = 0 ; i < 1000 ; i++);
```

Buning yordamida biz dastur ishlashini sekinlashtirishimiz mumkin.

switch operatori

if-else-if yordami bilan bir necha shartni test qilishimiz mumkin. Lekin bunday yozuv nisbatan o'qishga qiyin va ko'rinishi qo'pol bo'ladi. Agar shart ifoda butun son tipida bo'lsa yoki bu tipga keltirilishi mumkin bo'lsa, biz switch (tanlash) ifodalarini ishlata olamiz.

switch strukturasi bir necha case etiketlaridan (label) va majburiy bo'lmagan default etiketidan iboratdir. Etiket bu bir nomdir.U dasturnig bir nuqtasidaga qo'yiladi.Programmaning boshqa yeridan ushbu etiketga o'tishni bajarish mumkin. O'tish yoki sakrash goto bilan amalga oshiriladi, switch blokida ham qo'llaniladi.

5 lik sistemadagi bahoni so'zlik bahoga o'tqizadigan blokni yozaylik.

```
int baho;
baho = 4;
 switch (baho) {
case 5: cout << "A'lo";
break;
case 4: cout << "Yahshi";
break;
case 3: cout << "Qoniqarli";
break;
case 2:
case 1: cout << "A'lo";
break;
default: cout << "Baho hato kiritildi!";
break;
}
```

switch ga kirgan o'zgaruvchi (yuqorigi misolda baho) har bir case etiketlarining qiymatlari bilan solishtirilib chiqiladi. Solishtirish yuqoridan pastga bajariladi. Shartdagi qiymat etiketdagi qiymat bilan teng bo'lib chiqqanda ushbu case ga tegishli ifoda yoki ifodalar bloki bajariladi. So'ng breaksakrash buyrug'i

bilan switch ning tanasidan chiqiladi. Agar break qo'yilmasa, keyingi etiketlar qiymatlari bilan solishtirish bajarilmasdan ularga tegishli ifodalar ijro ko'raveradi.Bu albatta biz istamaydigan narsa.default etiketi majburiy emas. Lekin shart chegaradan tashqarida bo'lgan qiymatda ega bo'lgan hollarni diagnostika qilish uchun kerak bo'ladi.case va etiket orasida bo'sh joy qoldirish shartdir. Chunki, masalan, case 4: ni case4: deb yozish oddiy etiketni vujudga keltiradi, bunda sharti test qilinayotgan ifoda 4 bilan solishtirilmay o'tiladi.

Do while takrorlash operatori

Do while ifodasi while strukturasiga o'xshashdir. Bitta farqi shundaki while da shart boshiga tekshiriladi. Do while da esa takrorlanish tanasi eng kamida bir marta ijro ko'radi va shart strukturaning so'ngida test qilinadi. Shart true bo'lsa blok yana takrorlanadi. Shart false bo'lsa do while ifodasidan chiqiladi. Agar do while ichida qaytarilishi kerak bo'lgan ifoda bir dona bo'lsa {} qavslarning keragi yo'qdir. Quyidagicha bo'ladi:

```
do
ifoda;
while (shart);
```

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

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

Bu blokda 1000 dan kichik yoki teng bo'lgan eng katta 5 ga karrali son topilmoqda.while shartini ozroq o'zgartirib berdik, ! (not - inkor) operatorining ishlashini misolda ko'rsatish uchun. Agar oddiy qilib yozadigan

bo'lsak, while shartining ko'rinishi bunday bo'lardi: while (k<=1000); Cheksiz takrorlanishni oldini olish uchun shart ifodasining ko'rinishiga katta e'tibor berish kerak. Bir nuqtaga kelib shart true dan false qiymatiga o'tishi shart.

Mantiqiy operatorlar

Operator	Belgi	Misol
VA	&&	1ifoda && 2ifoda
YOKI	||	1ifoda||2ifoda
INKOR	!	!ifoda

Mantiqiy ko‘paytirish operatori

Mantiqiy ko‘paytirish operatori ikkita ifodani hisoblaydi, agar ikkala ifoda true qiyma tqaytarsa VA operatori ham true qiymat qaytardi.Agarda sizning qorningiz ochligi rost bo‘lsa VA sizda pul borligi ham rost bo‘lsa siz supermarketga borishingiz va u yerdan o‘zingizga tushlik qilish uchun biror bir narsa xarid qilishingiz mumkin. Yoki yana bir misol, masalan,

if(x==5)&&(y==5)

mantiqiy ifodasi agarda x va y o‘zgaruvchilarini ikkalasining ham qiymatlari 5 ga teng bo‘lsagina true qiymat qaytaradi. Bu ifoda agarda o‘zgaruvchilardan birortasi 5 ga teng bo‘lmagan qiymat qabul qilsa false qiymatini qaytaradi. Mantiqiy ko‘paytirish operatori faqatgina o‘zining ikkala ifodasi ham rost bo‘lsagina true qiymat qaytaradi.

Mantiqiy ko‘paytirish operatori && belgi orqali belgilanadi.

Mantiqiy qo‘shish operatori

Mantiqiy qo‘shish operatori ham ikkita ifoda orqali hisoblanadi. Agarda ulardan birortasi rost bo‘lsa mantiqiy qo‘shish operatori true qiymat qaytaradi. Agarda sizda pul YOKI kredit kartochkasi bo‘lsa, siz schyotni to‘lay olasiz. Bu holda ikkita shartning birdaniga bajarilishi: pulga ham va kredit kartochkasiga ham ega bo‘lishingiz shart emas. Sizga ulardan birini bajarilishi yetarli. Bu operatorga oid yana bir misolni qaraymiz. Masalan,

if(x==5)||(y==5)

ifodasi yoki x o‘zgaruvchi qiymati, yoki u o‘zgaruvchi qiymati, yoki ikkala o‘zgaruvchining qiymati ham 5 ga teng bo‘lsa rost qiymat qaytaradi.

Mantiqiy inkor operatori

Mantiqiy inkor operatori tekshirilayotgan ifoda yolg‘on bo‘lsa true qiymat qaytaradi. Agarda tekshirilayotgan ifoda rost bo‘lsa inkor operatori falseqiymat qaytaradi. Masalan,

(if!(x==5))

ifodasining qiymati, agarda x o‘zgaruvchisi 5 ga teng bo‘lmasa trueqiymat qaytaradi. Bu ifodani boshqacha ham yozish mumkin:

if(x!=5)

Qo`yilgan masalaning C++ dasturlash tilidagi algoritm, Berilgan masalani yechish tartibi

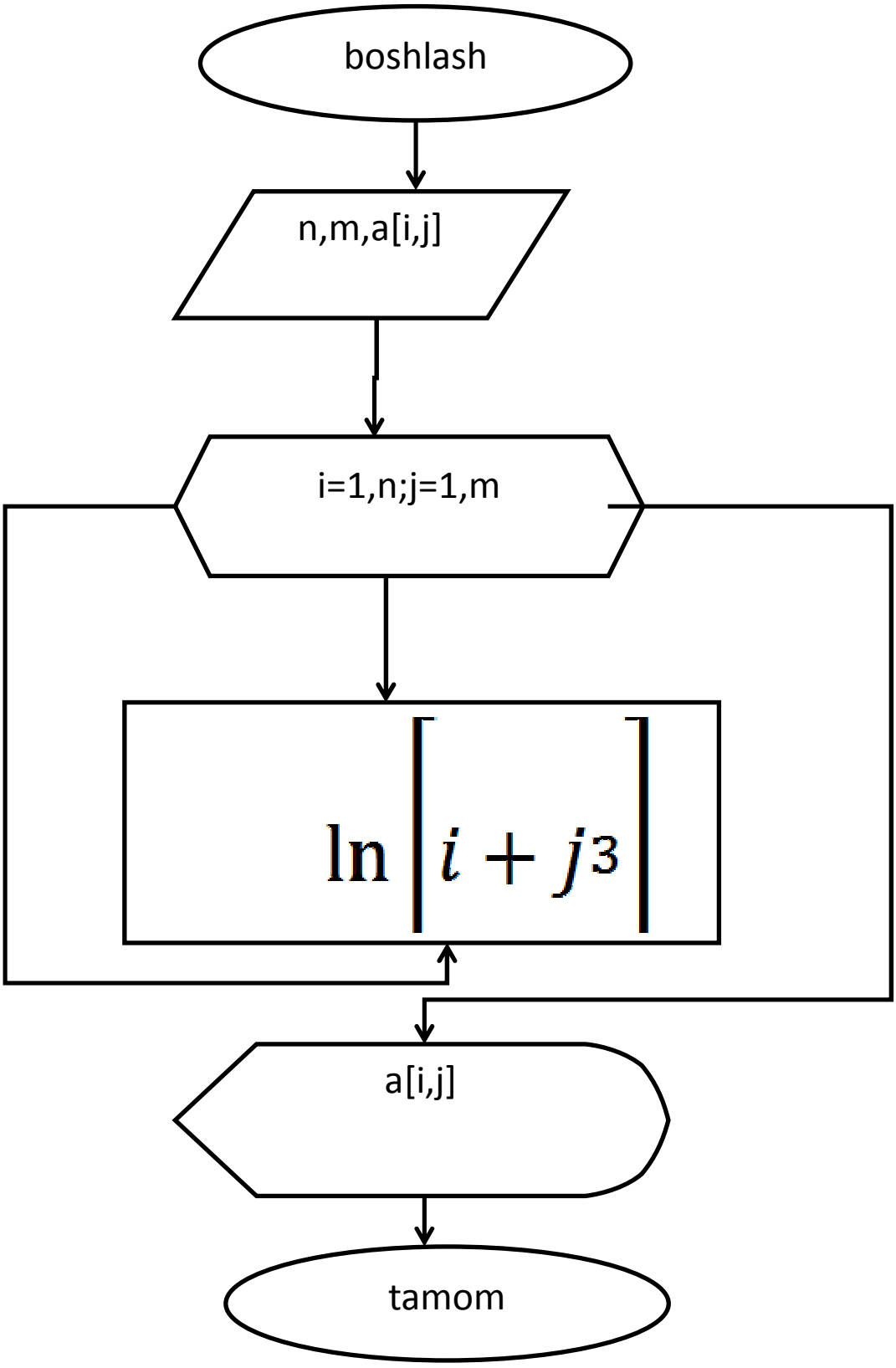
Masalaning berilishi. Haqiqiy qiymatli $A=\{a_{ij}\}$, (i,j=1,2,..,n) matritsa berilgan. Manfiy va musbat elementlar yig`indisini hisoblang.

Masalaning matematik tahlili.

*m ta satrli va n ta ustunli ushbu to`g`ri burchakli jadval shaklida yozilgan m*n ta son berilgan bo`lsin.*

$$A = \begin{Bmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{Bmatrix}.$$

Blok sxemasi

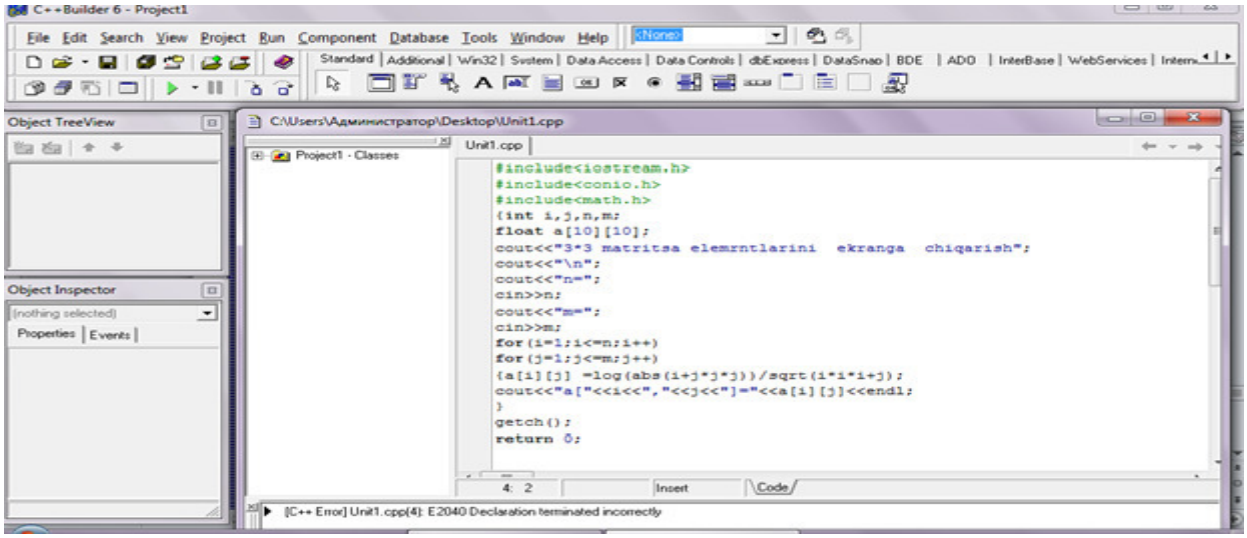


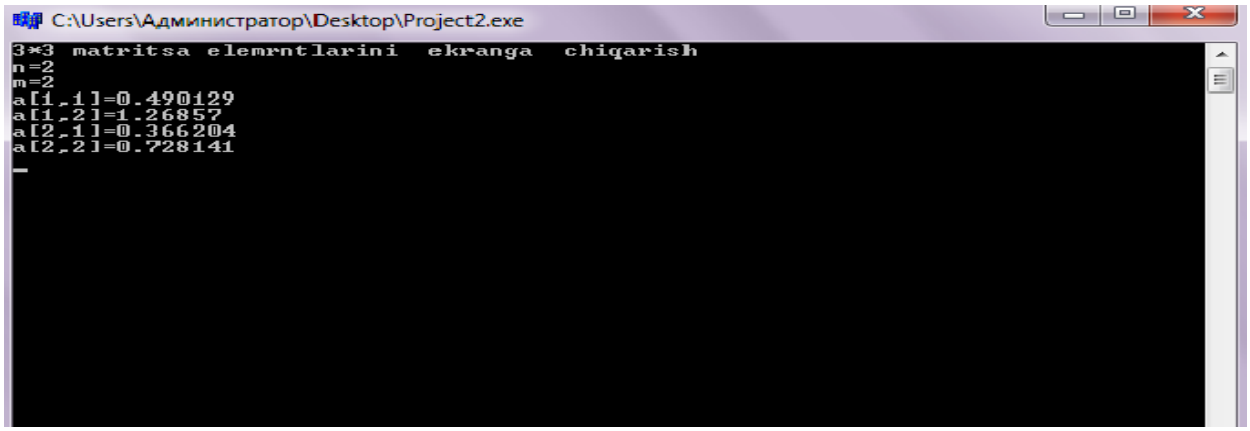
Masalaning C++ tilidagi dasturi

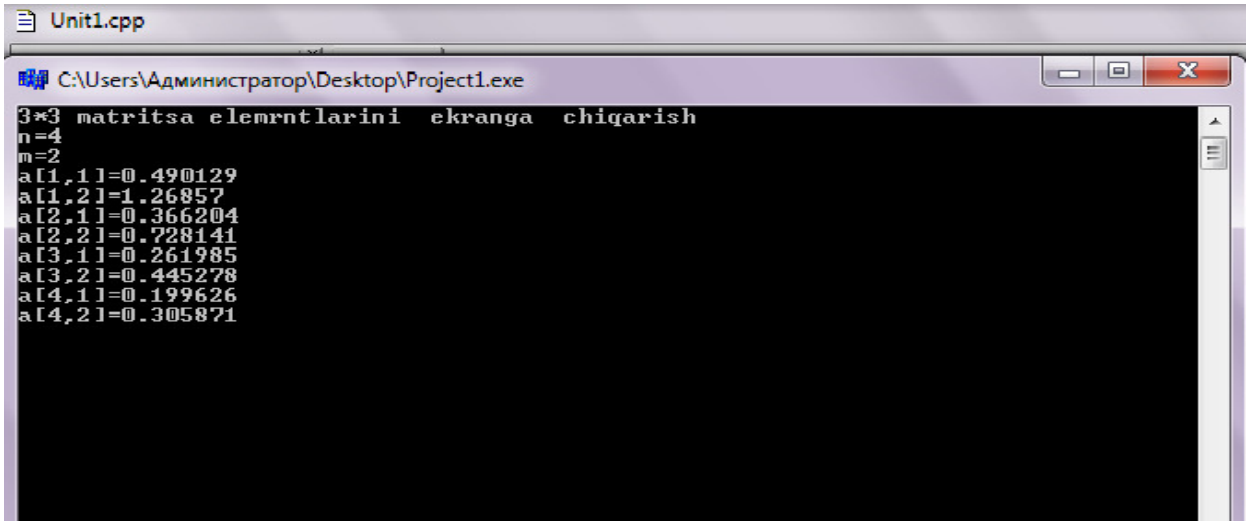
```
#include<iostream.h>
#include<conio.h>
#include<math.h>
int main()
{inti,j,n,m;
float a[10][10];
cout<<"matritsaelementlariniekrangachiqarish";
cout<<"\n";
cout<<"n=";
cin>>n;
cout<<"m=";
cin>>m;
for(i=1;i<=n;i++)
for(j=1;j<=m;j++)
{


getch();
return 0;}
```

Olingan natijalar tahlili







XULOSA

Xulosa qilib aytganda

Men tayyorlagan kurs ishimda masalani C++ dasturlash tilidagi dasturini, algoritm, blok sxemasini va metematik tahlilini batafsilroq yoritishga harakat qildim.

FOYDALANILGAN ADABIYOTLAR RO`YHATI:

1. Nazirov Sh. C++ da dasturlash asoslari.
2. Mardanova N.S. C++ tilida dasturlash
3. J.Liberti. Osvoy samostoyatelno C++ za 21 den.-SPb.2003.-815 s.
4. Informatika. Bazaviykurs. 2-izdanie. Uchebnikdlya VUZOV/Pod red. S.V.Simonovicha.-SPb.: Piter, 2009.-640 s.
5. Рахмокулова С.И. IBMPCшахсийкомпьютердаишлаш. 1999.
6. Бобровский С. , Delphi 7. Учебный курс. 2003, - 736 с.
7. О.А.Акулов, Н.В.Медведев Информатика базовый курс. Учебник.Москва 2007г. 555 стр.

Internet resurslari

1. **http://www.ziyonet.uz** - milliy axborot ta`lim tarmog`i
2. **http://www.mail.uz** - milliy elektron pochta xizmati
4. **http://www.edu.uz** - O`zbekiston ta`lim portali
5. **http://www.aci.uz** - O`zbekiston aloqa va axborotlashtirish agentligi portali
6. **http://www.uza.uz** - O`zbekiston milliy axborot agentligi
7. **http://www.tuit.uz** -TATU Veb portali