ШАГ компьютерная академия

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

Dərs №2

C
Proqramlaşdırma dili

Mündəricat

1. Operator anlayışı

Keçən dərsdə siz dəyişkən və verilənlərin tipi anlayışları ilə tanış oldunuz. Bundan başqa, sizinlə dərs nümunələrində və ev tapşırığında dəyişkənlər üzərində müəyyən əməliyyatar apardıq, yəni verilənləri idarə etdik. Aydındır ki, operator və əməliyyat aparmaq eyni mənşəlidir, buna görə də sadə məntiqə əsasən:

Operator - verilənlər üzərində müəyyən nəticəyə gətirən əməliyyatları yerinə yetirməyə imkan verən dil quruluşudur.

Bütün operatorları onların gördüyü işə görə qruplara ayırmaq qəbul olunub. Məsələn, riyazi əməliyyatlar - verilənlər üzərində riyazi əməliyyatları yerinə yetriməyə imkan verən əməliyyatlardır (vurma, çıxma və s.) C dilində təqdim olunan bütün bu cür qruplar barədə sonra danışacağıq. Hal-hazırda, dəyişkənlərin tərkibinə təsir etməyindən asılı olmayaraq, bütün operatorların geniş miqyaslı təsnifatını müzakirə etməyə dəyər. Beləliklə, bütün operatorlar bunlara bölünür:

1. **Unar -** yalnız bir operand (üzərində əməliyyat aparılan verilənlər) tələb edən operatorlardır. Unar operatora nümunə kimi məktəb riyaziyyat kursundan tanışsız - unar minus, bu ədədi mənfi ədədə çevirir (3 və -3) və yaxud müsbət (-(-3). Yəni unar operatorun ümum sintaksisi belədir:

```
operator operand ;   və ya  operand operator;
```

2.Binar - operatorun sağında və solunda iki operand tələb edən operatorlar. Bu cür çoxsaylı operator tanıyırsız, bun-lar - +, -, * və s.dir. Onların ümumi sintaksisini növbəti şəkildə göstərmək olar:

```
operand operator operand
```

3. Ternar - operand tələb edən operatorlar. C dilində bu cür operand yalnız bir dənədir və onun sintaksisi ilə bir qədər sonra tanış olacağıq.

Prioritet

Bütün operatorların prioriteti vardır. Aşağıda operatorlar prioritetlərinə uyğun göstərilmişdir. Bəziləri ilə bu günki dərsimizdə tanış olacağıq, bəzilərini isə gələn dərslərdə öyrənəcəksiniz. Təbii ki, bu cədvəldə dilin bütün operatorları göstərilməmişdir, hələ ki, bizim üçün ən aktual olanlarıdır.

Əməliyyatın simvollu işarəsi	
Yüksək prioritet	Yüksək prioritet
() [] . ->	^
! *(un) – (un) ~ ++ ––	|
% * /	&&
+ –	||
<< >>	?:
< > <= >=	= += –= *= /= %= &= |= ^= >>= <<=
!= ==	
&	Aşağı prioritet

İndi, operatorlar sahəsində biliklərin əsası qoyulduqdan sonra, siz onların daha ətraflı öyrənilməsinə keçə bilərsiniz, daha dəqiq, dərsin növbəti bölməsinə.

2. Rəqəmlər ilə riyazi əməliyyatlar

Yaxşı unudulmuş keçmiş....

Beləliklə, başlayaq. Əvvəldə qeyd olunduğu kimi, riyazi əməliyyatlar - verilənlər üzərində riyazi əməliyyatların aparılmasına imkan verən əməliyyatlardır. Onların çoxu sizə uşaq vaxtından tanışdır, ancaq, yenə də gəlin, biliklərimizi aşağıda göstərilən cədvəlin köməyilə sistemləşdirək.

Əməlyyatın adı	C dilində işarəsi üçün istifadə olunan simvol	Qısa təsvir. Nümunə
Toplama	+	İki qiyməti bir yerə toplayır, nəticəsi operandların cəmidir: 5+18 nəticə 23İİ
Çıxma	–	Operatordan solda olan qiymətdən sağda olanı çıxır. Nəticə, operandların fərqidir: 20 -15 nəticə 5OO
Vurma	*	İki qiyməti vurur, nəticə operandların hasılıdır: 5*10 nəticə 50
Bolmə	/	Operatordan solda olan qiyməti sağda olana bölür. Məsələn, 20/4 nəticə 5
Mdulca bölmə	%	Bu əməliyyatın nəticəsi, tamqiymətli bölmədən alınan qalıqdır, məsələn, əgər biz 11-i 3-ə bölürüksə, onda tam hissədə 3 ədədi alınır (belə ki, 3*3=9), qalıqda isə 2 olacaq, bu ədəd modulca bölmənin nəticəsidir: 11/3 4 tam 2 qalıqdır. 11%3=2 (qalıq)

Qeyd:

1. Modulca bölmə əməliyyatını yalnız tamqiymətli verilənlərə tətbiq etmək olar. Bu qaydanı pozmağa edilən cəhd, kompilyasiya mərhələsində səhvə gətirib çıxaracaq.

2. Əgər kiçik ədəd, böyük ədədə % operatoru ilə bölünürsə, onda nəticə kiçik ədədin özü olacaqdır. 3%10 = 3

3. Modulca sıfra bölmək olmaz, bu icra mərhələsində proqramın yanlış işləməsinə gətirib çıxaracaq.

İnkrement və dekrement.

Yuxarıda göstərilən əməliyyatar binar idi, lakin həmçinin, unar riyazi əməliyyatlar da vardır ki, bu cür əməliyyatlar məktəb kursunda yoxdur, ancaq, əslində onlar çox sadədirlər:

1. İkrement - ++ kimi işarələnir. Bu operator istənilən dəyişkənin qiymətini bir vahid artırır və qiyməti həmin dəyişkənə yazır.

Məsələn:

```
int a=8;
cout<<a; // ekrana 8 a++;
cout<<a; // ekrana 9
```

2. Dekrement - -- kimi işarələnir. Bu operator istənilən dəyişkənin qiymətini bir vahid azaldır və qiyməti həmin dəyişkənə yazır.

```
int a=8;
cout<<a; // ekrana 8 a--;
cout<<a; // ekrana 7
```

Olduqca asantdır, elə deyilmi?! Bu cür ifadələr belə də təsvir oluna bilər: a=a+1 və ya a=a-1. Qeyd etmək lazımdır ki, literallar üçün nə inkrement, nə də dekrement istifadə olunmur, çünki bü cür etmək məntiqi deyildir: 5=5+1. Bu açıq-aydın səhvdir. Lakin, burada inkrement və dekrement ilə tanışlığımız bitmir. Dərsin keçən bölməsində biz təyin etdik ki, unar operatorun sintaksisi yalnız belə olmur:

```
operand operator;
```

həm də belə olur

```
operator operand;
```

Bu cür yazı formaları postfiks (operator qiymətdən sonra yerləşir) və prefiks (operator qiymətdən öncə yerləşir) adlanır. İnkrement və dekrement hər iki formaya malikdirlər. Gəlin baxaq görək, bu formalar arasında hansı fərqlər var və bu hansı hallarda bu vərqlər məna kəsb edir.

Nümunə №1

```
int a=8;
cout<<a; // ekrana 8 a++;
cout<<a; // ekrana 9++a;
cout<<a; // ekrana 10
```

Bu nümunədə prefiks və postfiks formalar arasında heç bir fərq yoxdur. Həm birinci, həm də ikinci halda a dəyişkəninin qiyməti bir vahid artır.

Operatorun müxtəlif formalarından istifadənin mənası yalnız o zaman olur ki, sətirdə operatorun özündən başqa hər-hansı bir əmr olsun.

Nümunə 2.

```
int a=8;
cout<<++a; // ekrana 9
cout<<a++; // ekrana 9
cout<<a; // ekrana 10
```

Misalı şərh etməzdən əvvəl, gəlin üç qayda qoyaq:

1. C dilində əmrlərin icra prinsipi birmənalı deyil. Buna görə də bəzi operatorların icra istiqaməti aşağıdakı cədvəldə göstərilir:
2. Əgər inkrement və ya dekrementin postfiks formasından başqa, sətirdə başqa bir əmr də olarsa, onda əvvəlcə bu əmr və məhz bundan sonra, əmrlərin sətirdə yerindən asılı olmayaraq inkerement və dekrement yerinə yetirilir.
3. Əgər inkrement və ya dekrementin prefiks formasından başqa, sətirdə başqa bir əmr də olarsa, onda bu sətirdəki bütün əmrlər operatorların prioritetinə uyğun olaraq sağdan sola icra olunur.

Operator	İstiqamət
() [] . ->	Soldan sağa
* / % + –	
<< >> & ^ |	
<< = >> = ==! =	
&& ||	
Unar – unar + ! ++ ––	Sağdan sola
?:	
+ = – = / = * = % = & = ^ = | =	

İndi isə nümunə barədə daha ətraflı:

- İlk olaraq dəyişkənin qiyməti 8 ədədinə bərabərdir.
- cout<<++a; əmrində inkrement operatorunun prefiks forması mövcuddur, beləliklə, yuxarıda qeyd olunmuş üçüncü qaydaya riayət edərək, a dəyişkəninin qiymətini bir vahid artırırıq, sonra onu cout<< əmri vasitəsilə ekranda göstəririk.
- cout<<a++; əmrində inkrement operatorunun postfiks forması mövcuddur, beləliklə, yuxarıda qeyd olunmuş ikinci qaydaya riayət edərək, biz əvvəlcə dəyişkənin qiymətini (onun qiyməti hələki 9-dur) cout<< əmri vasitəsilə ekranda göstəririk, sonra dəyişkənin qiymətini bir vahid artırırıq.
- Növbəti cout<<a; əmrinin icrası zamanı artıq qiyməti dəyişmiş, yəni bir vahid artırılmış qiymət - 10 göstəriləcəkdir.

Dərsin bu bölməsindəki əvvəlki mövzulara əsaslanaraq, biz artıq bilirik ki, x=x+1 və ya x=x-1 kimi narahat və "çirkin" yazıları x++ və ya x - - kimi necə sadələşdirmək olar

Lakin, bu üsulla biz dəyişkənin qiymətini yalnız bir vahid artıra və ya azalda bilirik.
Bəs digər sayda artırıb/azaltmaq üçün necə etmək olar?
Məsələn, X=X+12 yazılığını necə sadələşdirək?
Bu halda da sadə həll vardır - birləşdirilmiş operatorlar və ya qisaldılmış riyazı formalardan istifadə etmək.

Onlar növbəti şəkildə görünür:

Formanın adı	Kombinasiya	Standar yazı	Qısaldılmış yazı
Vurmaqla mənimsətmə	*=	A=A*N	A*=N
Bölməklə mənimsətmə	/=	A=A/N	A/=N
Modulca bölməklə mənimsətmə	%=	A=A%N	A%=N
Çıxmaqla mənimsətmə	– =	A=A–N	A– =N
Toplamaqla mənimsətmə	+=	A=A+N	A+=N

Biz sizə gələcəkdə qısaldılmış formalardan istifadə etməyi tövsiyyə edirik, belə ki, bu proqramlaşdırmada nəinki yaxşı üslubdur, həm də proqram kodunun oxunaqlığını nəzərəçarpacaq dərəcədə artırır. Bundan başqa bəzi mənbələrdə göstərilir ki, qısaldılmış formalar kompüterlə tez emal edilir, bu da proqramın icra olunma sürətini artırır. İndi isə bütün bu yuxarıda deyilənlərin təcrübədə istiadə etməyin zamanı gəldi. Necə deyərlər, yüz dəfə eşitməkdənsə, bir dəfə görmək daha yaxşıdır.

Siz artıq layihə yaratmağı və onlara faylların əlavə edilməsini bacarırsınız, indi sizdən elə bu tələb olunur. Növbəti olaraq sizə, riyazi əməliyyatların təcrübədə necə icra edilməsini görmək üçün yazmalı olduğunuz bir neçə proqram təqdim olunmuşdur.
Game adlı layihədən başlayaq.

3.Riyazi əməliyyatların tədbiq olunması

Nümunə №1. Oyun.

```
// Uşaqlar üçün sadə oyun
#include <iostream>
using namespace std;
void main()
{
    int buddies;
    int afterBattle;


    cout<<"You the pirate. How many the person in your command, without you?\n\n";
    cin>>buddies;


    cout<<"Suddenly you are attacked by 10 musketeers \n\n";


    cout<<"10 musketeers and 10 pirates perish in fight.\n\n";


    afterBattle=1+buddies-10;


    cout<<»Remains only "<<afterBattle<<» pirates\n\n";


    cout<<"The condition killed totals 107 gold coins \n\n";


    cout<<"It on "<<(107/afterBattle)<<"coins on everyone";


    cout<<"Pirates arrange greater fight because of remained\n\n";

    cout<<(107%afterBattle)<<"coins \n\n";
}
```

Bu nümunədə tamın tama bölünməsi qaydasından istifadə nolunur - bu cür bölmədə kəsr hissəsi alınsa belə o kəsilib atılır. Bu barədə daha ətraflı - “Tiplərin çevrilməsi” dərsində danışacağıq. (107/afterBattle) ifadəsində sikkələri bərabər böldükdə, hər dəniz qulduruna neçə sikkə çatacağını öyrənmiş oluruq. Bundan başqa modulca bölmə operatoru, bizə bölünməsi mümkün olmayan neçə sikkənin qaldığını müəyyən etməyə kömək edir, yəni biz 107-nin sağ qalan dəniz quldurlarının sayına bölünməsindən alınan qaldığı əldə edəcəyik. Bu da nümunənin bütün xüsusiyyətləridir.

Nümunə №2. Çevrə.

Bu nümunədə, proqramlarda riyazi hesablamalar aparan riyazi operatorların istifadəsi nümayiş olunacaq. Layihənin adı **Circle.**

Biz əmin olacayıq ki, riyazi operatorları bilmək, sadə məsələləri həll etməyə imkan verir. Lakin, yalnız operatorlardan istifadə etmək azdır, onlardan istifadə etmənin nəticələrinin nə olduğunu başa düşmək lazımdır. Növbəti bölmədən bundan danışacağıq.

```
// Çevrənin parametrlərinin təyin edilməsi üçün proqram
#include <iostream>
using namespace std;
void main()
{
      const float PI=3.141592;


      float radius, circumference, area;


      cout<<"Welcome to program of work with rounds\n\n";
      cout<<"Put the radius from rounds\n\n";
      cin>>radius;
      cout<<"\n\n";

      area=PI*radius*radius;
      circumference=PI*(radius*2);


      cout<<"Square of round: "<<area<<"\n\n";
      cout<<"length of round: "<<circumference<<"\n\n";
      cout<<"THANKS!!!  BYE!!!\n\n";

}
```

4. Tiplərin çevrilməsi

Biz bir şeyi edərkən, şübhəsiz ki, nəticənin necə olacağını bilmək vacibdir. Məlumdur ki, məsələn xarço şorbası üçün hazırladığımız qatışığından, təbii ki, qaymaqlı tort hazırlaya bilmərik. Sözsüz ki, nəticə qarışığın tərkib hissələrindən asılıdır. Eyni şey də dəyişkənlərlə baş verir. Məsələn, əgər int tipli iki ədəd toplanırsa, tam aydındır ki, nəticə də int tipli olacaqdır. Bəs verilənlər ayrı-ayrı tiplərdə olduğu zaman necə edək? Dərsimizin bu bölümündə məhz bundan söhbət açacağıq.

Beləliklə, hər şeydən əvvəl, hansı verilənlər tipləri bir-biri ilə qarşılıqlı əlaqədə ola bilirlər. Belə bir ierarxiya tipi mövcuddur ki, burada bütün tiplər böyüklüyünə görə yerləşdirilmişlər. Tiplərin çevrilməsini anlamaq üçün, bu ierarxiyadakı tiplərin ardıcıllığını həmişə xatırlamaq lazımdır.

```
bool, char, short-int-unsigned int-long-unsigned
long-float-double-long double
```

Bəzi tiplərin eyni ölçülərə malik olmağına baxmayaraq, onların daxilində qiymətlərin müxtəlif diapazonları yerləşə bilir, məsələn, unsigned int - int dən fərqli olaraq özünə iki dəfə çox müsbət qiymətləri yerləşdirə bilər, ona görə ierarxiya üzrə böyükdür, baxmayaraq ki, hər iki tip eyni ölçüdədir - 4 bayt. Bundan əlavə, burada vacib xüsusiyyəti qeyd etmək lazımdır, əgər tiplərin çevrilməsində bool, char, short kimi tiplər iştirak edirsə, onlar avtomatik olaraq int tipinə çevrilirlər.

İndi isə çevrilmələrin müxtəlif təsniflənməsinə nəzər salaq.

Tərkibdə olan qiymətlərin diapasonuna görə təsnifləndirmə.

Bütün çevrilmələri çevrilmədə iştirak edən tiplərin ierarxiyadakı yerinə görə iki qrupa ayırmaq olar.

1. Məhdudlaşdırıcı çevrilmələr - bu cür çevrilmədə, ierarxiyada böyük verilənlər tipi kiçiyə çevrilir, sözsüz, bu halda verilənlərin itirilməsi baş verə bilər, buna görə məhdudlaşdırıcı çevrilmələr ilə aehtiyyatlı olmaq lazımdır.

Məsələn:

```
int A=23.5;
cout<<A; // ekranda 23
```

2. Genişləndirici çevrilmələr - Bu çevrilmə növü verilənlər tiplərinin kiçik diapazondan böyük diapazona genişlənməsinə gətirib çıxarır.

Nümunə kimi göstərək:

```
unsigned int a=3000000000;
cout<<a; // ekranda 3000000000
```

Bu halda 3000000000 - int tipli literaldır, o unsigned int tipinə qədər genişlənir, bu da bizə ekranda başqa şey yox, məhz 3000000000 qiymətini görməyə imkan verir. Baxmayaraq ki, adi int tipinə bu ədəd yerləşməz.

Çevrilmə üsuluna görə təsnifləndirmə.

Çevrilmənin istiqamətindən asılı olmayaraq, o iki üsuldan biri ilə həyata keçirilə bilər. Bu cür çevrilmə növünü həm də avtomatik adlandırırlar.

1. Qeyri-aşkar çevrilmələr.

Bütün yuxarıda şərh edilən nümunələr bu tip çevrilmələrə aid idi. Bu növ çevrilməni həm də avtomatik adlandırırlar, belə ki, o, proqramçının müdaxiləsi olmadan baş verir, bir sözlə, biz onun baş tuması üçün heç nə etmirik.

```
float A=23,5; - double  float oldu
```

2. **Aşkar çevrilmə** (ikinci adı tiplərə gətirmə). Bu halda, çevrilməni proqramçı ehtiyac olduqda özü edir.

Belə əməliyyata aid sadə bir misala baxaq:

```
double z=37.4;
float y=(int) z;
cout<<z<<"*** "<<y; // на экране 37.4 ***37
```

(int)z - double tipindən int tipinə aydın məqudlaşdırıcı çevrilmədir. Elə həmin bu sətirdə də alınmış int tipindən float tipinə qeyri-aşkar çevrilmə baş verir. Yadda saxlamaq lazımdır ki, istənilən çevrilmə müvəqqəti xarakter daşıyır və göstərilmiş sətirdə qüvvədə qalır. Yəni z dəyişkəni necə ki, double idi, bütün proqram boyu da belə qalacaqır, onun int tipinə çevrilməsi isə müvəqqəti xarakter daşıyır.

İfadələrdə tiplərin çevrilməsi.

İndi isə biz, dərsin lap əvvəlində danışdığımız bölməyə gəldik çatdıq, hansısa ifadənin nəticəsinin hansı tip olacağını necə müəyyən etmək olar. Gəlin əldə etdiyimiz biliklərə əsasən bunu aydınlaşdıraq. Hesab edək ki, bizdə aşağıdakı dəyişkənlər var:

```
int I=27;
short S=2;
float F=22.3;
bool B=false;
```

Bu dəyişkənlərdən istifadə edərək, biz belə bir ifadə tərtib etməyə hazırlasırıq:

```
I-F+S*B
```

Hansı verilənlər tipi dəyişkəninə nəticəni yazmaq lazımdır? Bunu həll etmək sadədir, əgər, ifadəni verilənlər tipi şəklində təsvir etsək:

```
int-float+short*bool
```

Yadınıza salırıq ki, short və bool dərhal int tipini alacaq, ona görə nəticə belə olacaq:

```
int-float+int*int,  və false 0 çevriləcək
```

int-i int-ə vurmaq bizə şübhəsiz int olan nəticə verəcəkdir. Lakin, float-un int-lə toplanması nəticədə float verəcəkdir, belə ki, burda belə bir qayda dövrəyə girir:

Əgər, hər-hansı bir ifadədə müxtəlif tip məlumatlardan istifadə olunursa, onda nəticə bu tiplərdən böyük olana gəlir.

Nəhayət sonda, qeyd olunan qaydaya əsasən, int tipindən float tipi çıxılarsa, nəticə float olar.

Beləliklə də ifadənin nəticəsi float olacaqdır.

```
float res= I-F+S*B; // 27-22.3+2*0
cout<<res; // 4.7
```

Artıq qayda ilə tanış olduqdan sonra sizin ifadəni ətraflı araşdırmağınıza ehtiyaz yoxdur, ən böyük tipi tapmağınız kifayətdir, məhz o nəticənin tipi olacaqdır.

Qeyd: Həm də, eyni məlumat tipli dəyişkənlərin istifadəsi zamanı da diqqətli olun. Məsələn, tam tama bölünürsə, nəticədə də tam alınacaqdır. Yəni int A=3; int B=2; cout<<A/B; // ekranda 1, belə ki, nəticəsi int tipindədir və kəsr hissə atılmışdır. cout<<(float)A/B; //ekranda 1.5, belə ki, nəticə float-dur.

Tiplərin çevrilməsininə aid nümunə.

İnd isə gəlin biliklərimizi təcrübədə gücləndirək. Yeni layihə yaradaq və aşağıdakı kodu daxil edək.

```
#include <iostream>
using namespace std;
void main(){
     // объявление переменных и запрос на ввод данных
     float digit;
     cout<<"Enter digit:";
     cin>>digit;

     /* Даже, если пользователь ввел число с вещественной частью,
     результат выражения запишется в int и вещественная часть будет утеряна,
     разделив число на 100 мы получим количество сотен в нем. */
     int res=digit/100;
     cout<<res<<"hundred in you digit!!!\n\n";
}
```

İndi, nümunələrə baxandan sonra siz, əlbəttə ki, əmin oldunuz ki, tiplərin çevrilməsinin köməyilə bir tipdən digərinə müvəqqəti keçidi təşkil etmək olur, həm də sadə məntiqi məsələni həll etmək olar. Buna görə də siz bu mövzuya diqqətlə yanaşmalısınız. Çevrilmənin başa düşülməsi, gələcəkdə sizə maraqlı məsələləri həll etməyə və lazımsız səhvlərdən yan keçməyə kömək edəcək.

Vahid şəklə salınmış inisializasiya.

C++ 11 -ə vahid şəklə salınmış mexanizm əlavə edilmişdir, bu müxtəlif proqram quruluşlarında (dəyişkənlərə, massivlərə, obyektlərə) qiymətin vahid üsulla verilməsinə imkan yaradır. Dəyişkənlərin inisializasiyasına aid nümunəyə baxaq.

```
int a = {11};  //  a-ya 11 yazılır
int b{33};     // b-yə 33 yazılır
```

Dəyişkənlərə qiymət vermək üçün biz {} istifadə edirik. Nümundən göründüyü kimi bunu iki üsulla etmək olar. Bu cür formada inisializasiya həmçinin **siyahılı inisializasiya** adlanır.

Mədudlaşdırma və siyahılı inisializasiya

Aşağıdakı kodun icrası zamanı nə baş verəcək?

```
int x = 2.88;
cout<<x; // ekrana 2 çıxacaq
```

Öyrəndiyiniz materialdan bildiyiniz kimi, verilmiş nümunədə qeyri-aşkar məhdudlaşdırıcı çevrilmə baş verir, belə ki, biz tam tipli x dəyişkəninə **double** tipli qiymət mənimsədirik. Ancaq, əgər siyahılı inisializasiyadan istifadə etsək, kompilyator kompilyasiya mərhələsində səhv emal edir, belə ki, bu inisializasiya forması məhdudlaşdırmadan qoruyur. O, böyük həcmli qiyməti, bu qiymət diapazonunu dəstəkləməyən tipə yazmağa imkan vermir.

```
int x = { 2.88 };  // kompilyasiya etapında səhv. 2.88 – double,
a x tam tipə aiddir
char ch = { 777 };  // kompilyasiya etapında səhv. 777 – int,
a ch simvollu tipə aiddir
// 777 char diapazonuna daxil deyil
```

Digər tərəfdən

```
char ch2 = { 23 };    // doğrudur
double x = { 333 };  // doğrudur
```

Əgər siz, kompilyasiya mərhələsində verilənlərin itirilməsi ilə bağlı potensial problemləri üzə çıxarmaq istəyirsinizsə, onda siyahılı inisializasiyadan istifadə edə bilərsiniz.

5. Məntiqi əməliyyatlar

Proqramlaşdırmada nəinki hansısa hesablamalar aparmaq, həm də kəmiyyətləri öz aralarında müqayisə etmək lazımdır. Bunun üçün məntiqi əməliyyatlardan istifadə edilir. Məntiqi əməliyyatların nəticəsi həmişə ya true (doğru) və ya false (yalan) olur.

Məntiqi əməliyyatlar 3 altqrupa ayrılır:
1. Müqayisə operatorları
2. Bərababərlik operatorları
3. Məntiqi birləşdirmə operatorları və mənfi inversiya.

Indi isə, hər qrup operatorlara ayrılıqda baxaq.

Müqayisə operatorları.

İki kəmiyyətin bir-birinə necə aid olmasını aydınlaşdırmaq lazım gəldiyində istifadə olunur.

Operatoru bildir'n simvol	Müddəa
<	Sol operand sağ operanddan kiçikdir
>	Sol operand sağ operanddan böyükdür
<=	Sol operand sağ operandan kiçik və ya bərabərdir
>=	Sol operand sağ operanddan böyük və ya barəbərdir

Müqayisə əməliyyatlarının (ikinci adı münasibət əməliyyatlarıdır) mənası bundadır ki, operatorun köməyilə verilən müddəa doğrudursa, onda onun istifadə olunduğu ifadə true qiymətini alacaq, yalandırsa - false. Məsələn:

```
cout<<(5>3); // 1
cout<<(3<2); // 0
```

Qeyd: false və true əvəzinə ekrana 0 və 1 çıxır, belə ki, onlar yalan və doğruya ekvivalentdirlər. C dilində doğru rolunda 1-dən fərqli olan hər hansı bir başqa ədəd də çıxış edə bilər, həm müsbət, həm mənfi.

Bərabərlik operatorları

İki kəmiyyətin bir-birinə tam uyğunluğunu və ya uyğun olmadığını yoxlamaq üçün istifadə olunur. Bu operatorların tədbiq olunması, bundan öncəki qrupun prinsiplərilə uyğun gəlir, yəni, çıxışda məna, müddəadan asılı olaraq doğru və ya yalanla əvəz olunur.

Символ, обозначающий оператор	Утверждение
==	Левый операнд равен правому
!=	Левый операнд не равен правому

Bu operatorların tədbiqi bundan öncəki qrupun tədbiq prinsipiylə uyğun gəlir, yəni, çıxışda məna, müddəadan asılı olaraq doğru və ya yalanla əvəz olunur.

```
cout<<(5!=3);

cout<<(3==2);
```

Məntiqi birləşdirmə əməliyyatları və mənfi inversiya.

Bir çox hallarda yalnız bir müddəa ilə keçinmək olmur. Tez-tez müddəaları bu və ya digər üsulla birləşdirmək lazımdır. Məsələn, ədədin 1 və 10 arası diapazonda olduğunu yoxlamaq üçün, iki müddəanı yoxlamaq lazımdır: ədəd eyni zamanda > = 1 və <= 10 olmalıdır. Bu cür birləşməni reallaşdırmaq üçün etmək üçün, əlavə operatorlar daxil etmək lazımdır.

Əməliiyat	Adı
&&	Və
||	Və ya
!=	YOX

Məntiqi VƏ (&&)

Məntiqi VƏ iki müddəanı bir yerdə birləşdirir və həm sol, həm də sağ müddəa doğru olduqda, doğru qiymətini qaytarır. Əgər bunlardan heç olmazsa biri və ya hər ikisi yalan olarsa, birləşmiş müddəa yalan ilə əvəz olunur. Məntiqi VƏ qısaldılmış sxem üzrə işləyir, yəni, əgər birinci müddəa yalandırsa, ikincisi artıq yoxlanmır.

Müddəa 1	Müddəa2	Müddəa1&&Müddəa2
true	true	true
true	false	false
false	true	false
false	false	false

İndi isə gəlin, proqramın ədəd almasına və bu ədədin 1 və 10 diapazonu arasına düşüb-düşməməyini necə yoxlamağına nümunədə nəzər salaq.

```
#include <iostream>
using namespace std;
void main()
{
	int N;
	cout<<"Enter digit:\n";
	cin>>N;
	cout<<((N>=1)&&(N<=10));
	cout<<"\n\nIf you see 1 your digit is in diapazone\n\n";
	cout<<"\n\nIf you see 0 your digit is not in diapazone\n\n";
}
```

Bu nümunədə, əgər hər iki müddəa doğrudursa, ifadənin qiyməti 1, əks halda 0 olacaq. Uyğun olaraq, istifadəçi proqramın əmrlərindən istifadə edərək, yaranmış vəziyyəti analiz edə bilər.

Məntiqi VƏ YA (||)

Məntiqi VƏ YA iki müddəanı bir yerdə birləşdirir və heç olmazsa bu müddəalardan biri doğru olarsa true, hər ikisi yalan olarsa, false qiymətini qaytarır. Məntiqi VƏ YA qısaldılmış sxem üzrə işləyir, yəni, əgər birinci müddəa doğrudursa, ikincisi artıq yoxlanılmır.

Müddəa 1	Müddəa 2	Müddəa1&&Müddəa 2
true	true	true
true	false	true
false	true	true
false	false	false

Proqramın ədəd qəbul etməsinə və bu ədədin 1-dən 10-ə qədər diapazonun daxilinə düşüb-düşmədiyini necə yoxlamağına nümunədə nəzər salaq. Ancaq, indi VƏ YA diapazonunu istifadə edək.

```
#include <iostream>
using namespace std;
void main()
{
      int N;
      cout<<"Enter digit:\n";
      cin>>N;
      cout<<((N<1)||(N>10));
      cout<<"\n\nIf you see 0 your digit is in diapazone\n\n";
      cout<<"\n\nIf you see 1 your digit is not in diapazone\n\n";
}
```

Bu nümunədə, əgər hər iki müddəa yalan olarsa, (yəni, ədəd 1-dən kiçik və 10-dan böyük olmazsa) onda ifadənin qiyməti 0 olacaq, əksa halda 1 olacaq. Uyğun olaraq, bundan əvvəlki nümunədə olduğu kimi, istifadəçi yaranmış vəziyyəti analiz edə və nəticə çıxara bilər.

Mətiqi YOX (!)

Məntiqi YOX unar operatordur və bununla əlaqədar olaraq birləşdirmə operatoru adlana bilməz. Müddəanın yoxlama nəticəsini əks hala dəyişdirmək lazım gələn zaman istiadə olunur.

Müddəa	! Müddəa
true	false
false	true

```
//ekranda 1 olacaq, belə ki, (5==3) yalandır və onun inversiyası isə doğrudur.
cout<<!(5==3);
//ekranda 0 olacaq, belə ki, (3!=2) doğrudur və onun inversiyası isə yalandır.
cout<<!(3!=2);
```

Məntiqi inkar əgər sonuncu doğrudursa yalan qiymətini verir və əksinə olarsa, doğru qiymətini verir. Bu operatoru, şərtin qoyulmasını qısaltmaq üçün tədbiq etmək olar. Məsələn:

```
b==0
```

inversiyanın köməyilə qısa da yazmaq olar:

```
!b
```

Əgər b sıfra bərabərdirsə, hər iki yazı çıxışda doğrunu verir. Bu bölmədə biz, hər növ məntiqi əməliyyatları nəzərdən keçirdik, bunlar sizə istənilən müddəanın doğru olmasını yoxlamağa kömək edəcək. Lakin, burada təsvir olunmuş nümunələr sıravi istifadəçi üçün narahatdırlar, belə ki, nəticələrin analizini o yox, kompüter etməlidir. Bundan əlavə, müddəadan asılı olaraq onun yoxlanmasının nəticəsini ekrana nəinki vermək, hər hansı bir əməliyyat etmək lazımdırsa, burada istifadəçi artıq tamamilə gücsüzdür. Bununla əlaqədar olaraq, məntiqi əməliyyatları bilərək, şərtdən asılı olan bu və ya digər hərəkətin reallaşdırılmasın üçün mümkün olan əlavə informasiyanı almaq lazımdır. Dərsimizin növbəti bölməsində məhz bu barədə danışacayıq.

6. Məntiqi if seçiminin quruluşu

İndi biz sizinlə, adi xətti proqramı, “düşünən” proqrama çevirə bilən operatorla tanış olacayıq. Bu operator hansısa müddəanın (ifadənin) doğruluğunu yoxlayır və alınmış nəticədən asılı olaraq bu və digər əməli həyata keçirir. Başlanğıc olaraq, operatorun ümumi sintaksisinə nəzər salaq:

```
if (ifadə)
{
  hərəkət 1;

else
{
  hərəkət 2;
}
```

if operatorunun işinin əsas prinsipləri.

1. Müddəa və ya ifadə kimi məntiqi operatorlar və ya riyazi ifadələr ehtiva edən hər-hansı bir quruluş çıxış edə bilər.

- If(X>Y) - adi müddəa o zaman doğru olacaq ki, X həqiqətən Y-dən böyük olsun.

```
int X=10,Y=5;
if(X>Y){ // истина
cout<<"Test!!!";// на экране Test
}
```

- if(A>B&&A<C) - birləşdirilmiş müddəa iki hissədən ibarətdir, hər iki hissəsi doğru olarsa, doğru olacaqdır.

```
int A=10,B=5,C=12;
if(A>B&&A<C){ // true
      cout<<"A between B and C";// ekranda A between B and C
}
```

- if(A-B) - riyazi ifadə o zaman doğru olacaq ki, A B-yə bərabər olmasın, əks halda (onlar bərabər olarsa) onların fərqi 0 verəcək, 0 isə yalandır.

int A=

```
int A=10,B=15;
if(A-B){ // -5 true
      cout<<"A != B";// ekranda A != B
}
```

- if(++A) - riyazi ifadə A -1-ə bərabər olmazsa doğru olacaq, belə ki, əgər A -1-ə bərabərdirsə, 1 vahid artırıldığı üçün 0 verəcək, 0 isə yalandır.

```
int A=0;
if(++A){ // 1 это истина
      cout<<"Best test!!";// на экране Best test!!
}
```

- if(A++) - riyazi ifadəsi o zaman doğru olar ki, A 0-a bərabər olmasın, bu halda, inkrementin postfiks forması istifadə olunur. Əvvəlcə şərtin yoxlanılması həyata keçirilir və 0 tapılacaq, sonra isə bir vahid artirilacaq.

```
int A=0;
if(A++){ // 0 false
      cout<<"Best test!!";// if icra olunmadıqından bunu görməyəcəyik
}
```

- if(A==Z) -adi müddəa A Z-ə bərabər olarsa, doğrudur.
- if(A=Z) - mənimsətmə əməliyyatı Z 0-a bərabər olmazsa, ifadə doğru olacaq

Qeyd: tipik səhv. Tez-tez diqqətsizlik üzündən, bərabərliyin == yoxlanması əməliyyatı zamanı, = mənimsətmə operatoru göstərilir və ifadənin mənası radikal olaraq dəyişir. Bu cür səhv bütün proqramın yanlış işləməsinə səbəb ola bilər. İki nümunəyə nəzər salaq.

Düzgün nümunə.

```
#include <iostream>
using namespace std;
void main(){
  int A,B;


    cout<<"Enter first digit:\n";
    cin>>A;
    cout<<"Enter second digit:\n";
    cin>>B;

    if(B==0){
      cout<<"You can't divide by zero!!!";
    }
    else{
  cout<<"Result A/B="<<A<<"/"<<B<<"="<<A/B;
    }
    cout<<"\n The end. \n";
}
```

Səhv nümunə

```
#include <iostream>
using namespace std;
void main(){
          int A,B;

           cout<<"Enter first digit:\n";
           cin>>A;
           cout<<"Enter second digit:\n";
           cin>>B;


           if(B=0){
                  cout<<"You can't divide by zero!!!";
           // сообщаем об ошибке
             }
else{


 cout<<"Result A/B="<<A<<"/"<<B<<"="<<A/B;
              }
              cout<<"\n The end. \n";
}
```

2. Siz diqət etdiyiniz kimi, əgər dairəvi mötərizələrin içindəkilər doğrudursa, if strukturunun fiqurlu mötərizələrlə əhatə olunmuş 1 əməli icra olunacaq, bu halda else blokunun 2 əməli ləğv ediləcək.

3. Yox, əgər əgər dairəvi mötərizələrin içindəkilər səhvdirsə, fiqurlu mötərizələrlə əhatə olunmuş 2 əməli icra olunacaq, bu halda 1 əməli ləğv ediləcək.

4. else strukturu vacib deyildir. Bu o deməkdir ki, müddəa yalan olduğu zaman nə isə bir şey etmək lazım deyil, bu strukturu verməmək də olar. Məsələn, sıfra bölməyə cəhdin qarşısını almaq üçün proqramı belə yazmaq olar:

```
#include <iostream>
using namespace std;
void main(){
         int A,B;

         cout<<"Enter first digit:\n";
         cin>>A;
         cout<<"Enter second digit:\n";
         cin>>B;
         if(B!=0){
                 cout<<"Result A/B="<<A<<"/"<<B<<"="<<A/B; // производим вычисления
         }

         cout<<"\nThe end.\n";
}
```

5. Əgər if və ya else blokuna bir əmr aid olarsa, onda fiqurlu mötərizələri göstərilmək vacib deyil. Bu qaydanın köməyi ilə proqramı daha qısa edək:

```
#include <iostream>
using namespace std;
void main(){
       int A,B;


        cout<<"Enter first digit:\n";
        cin>>A;
        cout<<"Enter second digit:\n";
        cin>>B;

       if(B!=0)
           cout<<"Result A/B="<<A<<"/"<<B<<"="<<A/B; // производим вычисления

       cout<<"\nThe end.\n";
}
```

Biz indi if şərti operatoru ilə tanış olduq və onun fuksiyasının əsas prinsiplərini müzakirə etdik. if operatorunun özünəxas xüsusiyyətlərinə və praktik nümunələrə keçməzdən əvvəl, sadə şərt qurmağa imkan verən daha bir operatora baxaq.

Qeyd: Diqqətli olun: if operator və else ayrılmazdırlar! onların arasına kod sətrini yazmaq cəhdi, kompilyasiya mərhələsində səhvə gətirəcək.

Səhv olan kod nümunəsi.

```
...
    if(B==0){
        cout<<"You can't divide by zero!!!";// сообщаем об ошибке
    }
    cout<<"Hello";
    else{
              cout<<"Result = "<<A/B;// выдаем результат деления A на B
    }
            ....
```

Ternar operator

Bəzi şərtlər çox sadədirlər. Məsələn, bizim iki ədədi bölmə proqramını götürək. O əməl cəhətdən və kod nöqteyi nəzərindən sadədir. if operatoru və else kodun bir sətri kimi qəbul edilir. Bu cür proqramı ternar operator istifadə edərək daha da sadələşdirmək olar.

Başlanğıc üçün, onun sintaksisinə nəzər salaq:

```
MÜDDƏA VƏ YA İFADƏ? ƏMƏL1: ƏMƏL2;
```

İş prinsipi sadədir - əgər, MÜDDƏA VƏ YA İFADƏ - doğrudursa, onda ƏMƏL1 icra olinur, yalandırsa - ƏMƏL2 .

Gəlin bu operatorun işinə aşağıdakı nümunədə baxaq:

```
#include <iostream>
using namespace std;
void main(){
    int A,B;


    cout<<"Enter first digit:\n";
    cin>>A;
    cout<<"Enter second digit:\n"
    cin>>B;

/*Bu halda əgər B sıfıra bərabər deyilsə, onda sual işarəsindən sonra gələn əmr icra
olunacaq və ekranda bölmənin nəticəsi göstəriləcək. Əks halda iki nöqtə işarəsindən
sonrakı əmr icra olunacaq və ekranda sıfıra bölmənin səhv olduğu ismarşı görünəcək.*/

 (B!=0)?cout<<"Result A/B="<<A<<"/!"<<B<<"="<<A/B:cout<<"You can't divide by zero!!!";

    //конец программы
    cout<<"\n The end. \n";
}
```

Kod daha optimal oldu, elə deyilmi? Əldə etdiyiniz informasiyanı möhkəmləndirmək üçün, daha bir nisbətən çətin nümunəyə baxaq. Proqram istifadəçinin daxil etdiyi iki ədəddən hansının böyük, hansının kiçik olduğunu təyin edəcək.

```
#include <iostream>
using namespace std;
void main(){
    int a,b;


    cout<<"Enter first digit:\n";
    cin>>a;
    cout<<"Enter second digit:\n";
    cin>>b;




    в/* əgər, (b>a), onda ?: operatorunun yerinə, b gələcək,
əks halda operatorun yerinə a gələcək,
 beləliklə max dəyişkəninə böyük olan ədəd yazılacaq. */
int max=(b>a)?b: a;
 /* əgər, (b<a), onda ?: operatorunun yerinə, b gələcək,
əks halda operatorun yerinə a gələcək,
 beləliklə min dəyişkəninə daha böyük olan rəqəm yazılacaq. */


    cout<<"\n Maximum is \n"<<max;
    cout<<"\n Minimum is \n"<<min<<"\n";
}
```

Beləliklə gəlin, bunları aydınlaşdıraq:

Əgər şərt və ondan asılı olan əməllər kifayət qədər sadədirsə, onda ternar operatordan istifadə edəcəyik. Yox, əgər bizə mürəkəb struktur lazımdırsa, şübhəsiz if operotunu istifadə edəcəyik.

7. if – else if pilləkəni

Dərsin keçən bölümlərindən şərt operatorları haqqında biliklər əldə etdik. İndi isə onların özünəməxsus xüsusiyyətləri haqqında informasiya almaq pis olmazdı. Məsələn, hesab edək ki, bizə cəmindən asılı olaraq, qiymət endiriminin hesablanması üçün proqram yazmaq lazımdır. Məsələn, əgər alıcı 100 manatdan çox məbləğdə mal alıbsa, onda o 5% endirim alır. 500 manatdan çox olarsa - 10% və nəhayət 1000 manatdan çoxdursa - 25%. Proqram əgər alıcı endirim əldə edibsə, hansı məbləği ödəyəcəyini hesablamalıdır. İndi isə məsələnin həlli üçün optimal variantı tapmaq lazımdır. Layihənin adı Discount.

Həll variantı - № 1.

```
#include <iostream>
using namespace std;
void main(){

	int summa;


	cout<<"Enter item of summa:\n";
	cin>>summa;

	if(summa>100){
	cout<<"You have 5% discount!!!\n";
		cout<<"You must pay - "<<summa-summa/100*5<<"\n";
	}
	if(summa>500){
	cout<<"You have 10% discount!!!\n";
	cout<<"You must pay - "<<summa-summa/100*10<<"\n";
```

```
        }
        if(summa>1000){
        cout<<"You have 25% discount!!!\n";
        cout<<"You must pay - "<<summa-summa/100*25<<"\n";
        }
        else{
        cout<<"You have not discount!!!\n";
        cout<<"You must pay - "<<summa<<"\n";
        }

}
```

Bu nümunə, ilk baxışda yenicə başlayan proqramçı üçün, heç bir sual yaratmır. Ancaq gəlin proqramın tamamilə səhv iş görəcəyi hala baxaq. Klaviaturadan daxil edilən məbləğ 5000-ə bərabərdir, bu ədəd 1000-dən çoxdur, buna əsasən biz 25% endirim almalıyıq. Lakin, tamamilə başqa şey baş verəcək.

1. Hər if operatoru sərbəstdir, digər if operatorlarından asılı deyil və hansı if operatorunun icra olunacağından asılı olmayaraq, şərtin yoxlanılması bütün operatorlar üçün icra olunacaq.

2. Əvvəlcə if(summa>100) şərtinin yoxlanılması icra olunacaq. 5000, əlbəttə ki, 100-dən böyükdür, şərt doğrudur və if operatorunun məzmunu icra olunacaq. Ekranda isə biz görürük:

```
You have 5% discount!!!
You must pay - 4750
```

3. Ancaq, proqram bunun üzərində dayanmayacaq - sonra if(summa>500) şərti analiz olunacaq. 5000, 500-dən böyükdür, şərt doğrudur və if operatorunun məzmunu icra olunur. Ekranda biz alırıq:

```
You have 10% discount!!!
You must pay - 4500
```

4. Və sonda proqram if(summa>1000) şərtini yoxlayacaq. Bu da doğru olacaq, çünki, 5000 1000-dən böyükdür. if operatorunun məzmunu icra olunur. Ekrana çıxır:

```
You have 25% discount!!!
You must pay - 3750
```

Bununla da, bir informasiya əvəzinə 3-nü alırıq. Bu cür məsələ həlli əlverişli deyil. Onu optimallaşdırmaq lazımdır. Layihənin adı Discount2

Həll variantı № 2.

```
#include <iostream>
using namespace std;
void main(){

        int summa;


        cout<<"Enter item of summa:\n";
        cin>>summa;


скидка 5%
        if(summa>100&&summa<=500){
                cout<<"You have 5% discount!!!\n";
                cout<<"You must pay - "<<summa-summa/100*5<<"\n";
        }

        if(summa>500&&summa<=1000)
```

```
{
                cout<<"You have 10% discount!!!\n";
                cout<<"You must pay - "<<summa-summa/100*10<<"\n";
        }
        if(summa>1000){
                cout<<"You have 25% discount!!!\n";
                cout<<"You must pay - "<<summa-summa/100*25<<"\n";
        }
        else{
                cout<<"You have not discount!!!\n";
                cout<<"You must pay - "<<summa<<"\n";
        }
}
```

Başlanğıc üçün, yenə hesab edək ki, istifadəçi 5000 manat dəyərində məbləği daxil edib.

1. Əvvəlcə, if(summa>100 && summa<=500) şərti yoxlanılacaq. 5000 verilmiş edilmiş diapazonda deyil, şərt yalandır və if operatorunun məsmunu icra olunmayacaq.

2. Sonra if(summa>500 && summa<=1000) şərti analiz ediləcək. 5000 bu diapazonda da deyil, şərt yenə yalandır və if operatoru icra olunmayacaq.

3. Və sonda, proqram if(summa>1000) şərtini yoxlayacaq, bu isə doğru olacaq, belə ki, 5000 1000- dən çoxdur. if ilə bağlı əməl icra olunacaq. Ekranda çıxacaq:

```
You have 25% discount!!!
You must pay - 3750
```

Elə gəlir ki, bu mərhələ artıq dayanmaq olar. Lakin, gəlin daha bir variantı yoxlayaq. Məsələn, istifadəçi 600 qiymətini daxil edir. Ekranda isə bu məlumatlar çıxır.

```
Enter item of summa:
600
You have 10% discount!!!
You must pay - 540
You have not discount!!!
You must pay - 600
Press any key to continue
```

Hadisələrin bu cür gedişatı asanlıqla şərh oluna bilər:

1. Əvvəlcə if(summa>100&&summa<=500) şərtinin yoxlanılması başlayacaq, şərt səhvdir, if operatorunun məzmunu icra olunmayacaq. 5000 verilmiş diapazonda deyil.

2. Sonra if(summa>500&&summa<=1000) analiz ediləcək, 5000 bu diapazona daxil ola bilər, şərt doğrudur, if operatorunun məzmunu icra olunacaq. Ekrana 10% endirim barədə ismarış çıxacaq.

3. Sonda isə proqram if(summa>1000) şərtini yoxlayacaq, bu yalan çıxacaq, if icra olunmayacaq, lakin, bu sərbəst if operatorunun, öz else-i vardır. Bizim halda o işə düşür, ekrana endirim olmadığı barədə ismarış çıxır.

Nəticə: birincisi, biz ayırd etdik ki, else operatoru sonuncu if-ə aiddir, ikincisi, belə nəticəyə gəldik ki, proqramın bu cür reallaşdırılması da bizi qane etmir. Bir varianta da baxaq. Proektin adı Discount 3.

Həll variantı № 3.

```
#include <iostream>
using namespace std;
void main(){
// İlkin məbləğin saxlanılması üçün dəyişkənin elan edilməsi
int summa;
// məbləğin klaviaturadan daxil edilməsinə sorğusu
cout<<”Enter item of summa:\n”;
cin>>summa;
if(summa>1000){ // əgər məbləq 1000 manatdan çoxdursa, 25% endirim
cout<<”You have 25% discount!!!\n”;
cout<<”You must pay - “<<summa-summa/100*25<<”\n”;
else{ // əgər məbləğ 1000 manatdan böyük deyilsə, analizi davam etdiririk
if(summa>500){ // məbləğ 500 manatdan çoxdursa, 10% endirim
cout<<”You have 10% discount!!!\n”;
cout<<”You must pay - <<summa-summa/100*10<<”\n”;
}
else{ // əgər məbləğ 500 manatdan böyük deyilsə, analizi davam etdiririk
if(summa>100){ // əgər cəm 100 manatdan çoxdursa, 5% endirim
cout<<”You have 5% discount!!!\n”;
cout<<”You must pay - “<< summa-summa/100*5<<”\n”;
else{ // əgər məbləğ 100 manatdan çox deyilsə, endirim yoxdur
cout<<”You have not discount!!!\n”;
cout<<”You must pay - “<< summa<<”\n”;
                        }
                        }
                        }
}
```

Bu nümunəni diqqətlə analiz etdikdən sonra siz görəcəksiniz ki, hər sonra gələn if, o halda icra oluna bilər ki, ondan əvvəlki icra olunmasın, belə ki, sonuncunun strukturu daxilində else vardır. Və bununla da biz reallaşdırmanın optimal kodunu tapdıq. Bu indicə yaratdığımız struktur “if else if pilləkəni” adlanır, çünki, içindəki şərtlər pilləkən şəkillidir. İndi biz bilirik bu necə faydalı strukturdur. Sonuncu hal qaldı:

Kodun optimallaşdırılması.

Bundan əvvəlki bölmədə belə bir qayda səsləndi: Əgər if və ya else blokuna yalnız bir əmr aiddirsə, onda fiqurlu mötərizələri göstərməsək də olar. İş burasındadır ki, if else strukturu, yeganə əmrli struktur hesab olunur. Bununla da, əgər bəzi else-lərin içində daxil edilmiş strukturlardan başqa heç nə yoxdursa, bu cür else-lərin fiqurlu mötərizələrini aşağı salmaq olar:

```
#include <iostream>
using namespace std;
void main(){
// İlkin məbləğin saxlanılması üçün dəyişən elan edilir
int summa;


cout<<"Enter item of summa:\n";
cin>>summa;

if(summa>1000){
        cout<<"You have 25% discount!!!\n";
        cout<<"You must pay - "<<summa-summa/100*25<<"\n";
 }


else if(summa>500){
                cout<<"You have 10% discount!!!\n";
                cout<<"You must pay - "<<summa-summa/100*10<<"\n";
 }


else if(summa>100){
                        cout<<"You have 5% discount!!!\n";
                        cout<<"You must pay - "<<summa-summa/100*5<<"\n";
 }
else{
                        cout<<"You have not discount!!!\n";
                        cout<<"You must pay - "<<summa<<"\n";
 }
}
```

Bu da son! Məsələ həll olundu. Biz çoxsaylı seçimin tam strukturunu əldə etdik, hansı ki, qarşılıqlı bağlılığı olan ayrıca şərtlərdən ibarətdir. İndi isə dərsin növbəti bölümlərinə keçə bilərik, burada biz if else istifadəsinin nümunələrinə daha geniş nəzər salacağıq.

8. Praktik misal: mətn kvestinin yaradılması

Əlbəttə siz kvest oyun janrı ilə tanışsınız. Belə oyunun qəhramanı müxtəlif tapşırıqlar yerinə yetirməli, suallara cavab verməlidir, oyun nəticəsini həll edən qərarlar verməlidir. Biz sizinlə mətnli kvest deyilən (qrafikasız kvest) oyun yaratmaqa çalışacayıq. Bizim işimiz qəhramana hərəkər yolları təklif edib, onun addımların asılı olaraq situasiyanı qurmaqdır. Lahiyənin adı **Quest.**

Rlizasiya kodu.

```
#include <iostream>
using namespace std;
void main()
{
    // Xoş gəlmisən. Üç şərəf sınaması. Cadügər şahzadəni qaçırdıb
    //və onun talehi sənin əlindədir. O
    //labirintində 3 şərəf sınaması keçməyi təklif edir.
    cout<<"Welcome. Three tests of honour. The malicious magician has stolen\n\n";
    cout<<"\nprincess and its destiny in your hands. It suggests you\n";
    cout<<"\nto pass 3 tests of honour in its labyrinth.\n";


    bool goldTaken, diamondsTaken, killByDragon;

    cout<<"You enter into the first room, here it is a lot of gold.\n\n";

    cout<<"Whether you will take it?(1-yes, 0-no)\n\n";
    cin>>goldTaken;
    if(goldTaken)
    {
```

```
        cout<<"Gold remains to you, but you have ruined test. GAME is over!!!\n\n";
    }
    else
    {

        cout<<"I congratulate, you have passed the first test abuse!\n\n";

        cout<<"You pass in a following room. It is full of brilliants \n\n";

        cout<<"Whether you will take brilliants? (1-yes,0-no)\n\n";
        cin>>diamondsTaken;
        if(diamondsTaken)//
        {

                cout<<"Brilliants remain to you, but you have ruined the second test\n\n";

                cout<<"GAME is over!!!\n\n";
        }
        else //
        {

                cout<<"I congratulate, you have passed the second test abuse!!!\n\n";

                cout<<"You enter into the third room. \n\n";

                cout<<"The person was attacked by a dragon! To move further \n\n";

                cout<<"Not paying to them of attention (1-yes,0-no)?\n\n";
                cin>>killByDragon;
                if(killByDragon)//
                {

                        cout<<"You try to pass past, but a dragon \n\n";

                        cout<<"notices your presence\n\n";

                        cout<<"It transforms you into ashes. You are dead!!!\n\n";

                        cout<<"GAME is over!!!\n\n";
                }
                else//
                {

                        cout<<"I congratulate, you with honour have was tested all!!! \n\n";

                        cout<<"Princess gets to you!!!\n\n";
                }
        }
    }
}
```

Nümunənin primitivliyinə baxmayaq siz özünüz görə bilərsiniz ki, artıq indi minimal biliklərə malik olaraq biz adi uşaqı əyləndirə biləcək proqram yaza bilərik. Bunun səbəbi bizim əlimizdə şərti operatorların - güclü vasitənin olmasıdır.

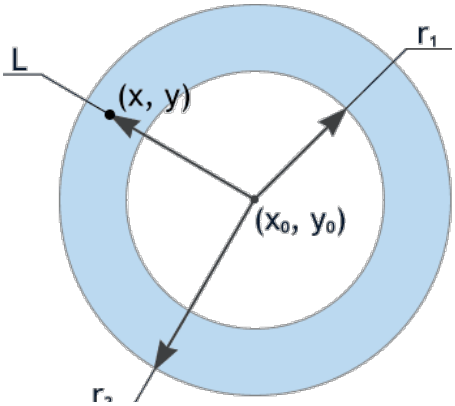
9. Nöqtənin üzüyə aidiyyatının praktiki nümunəsi

Məsələ

Düzlükdə (x0,y0) nöqtəsində mərkəzlə üzük çəkilib. Və r1<r2 hədd radiusları ilə. Bundan başqa həmin düzlükdə (x, y) koordinatlı nöqtə verilib. Bu nöqtənin üzüyə aid olub ya olmadığını təyin etmək lazımdır. Lahiyənin adı **Circle2.**

Məsələnin həlli

Məsələnin həlli üçün üzüyün ortasından nöqtəyə qədər olan məsafəni hesablıyıb onu radiuslar ilə müqayisə etmək lazımdır:



1. Əgər kəsikin uzunluqu xarici çevrə radiusundan az və daxili çevrənin radiusundan çox olarsa, nöqtə üzüyə aiddir.

if(L>=r1&&L<=r2)

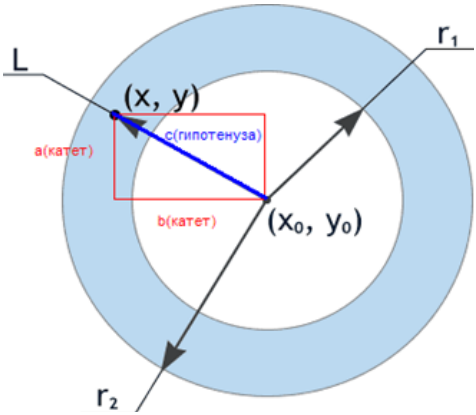
Əks təqdirdə nöqtə üzüyə aid deyil.

2. Mərkəzdən nöqtəyə qədər olan məsafəni öyrənmək üçün biz Pifaqor nəzəriyyəsindən istifadə edəcəyik - Hipotenuzun kvadratı katetlərin kvadratının cəminə bərabərdir. Следовательно — Bundan çıxır ki, hipotenuzun uzunluğu katetlərin kvadratının cəminin kvadrat kökünə barəbərdir.

Qeyd: Bizə dərəcənin və kvadrat kökün tapılması üçün əlavə biliklər lazım olacaq.

1. Proqrama riyazi funksiyalardan istifadə etmək üçün math.h adlı kitabxananı qoşmaq lazımdır.

2. Dərəcəyə yüksəltmək üçün pow (double num,double exp) funksiyasından istifadə olunur, və burda num dərəcəyə yüksəldidən ədəd exp isə dərəcənin özüdür.



3. Kvadrat kökün tapılması üçün isə sqrt(double num) funksiyasından istifadə olunur, və burda num kökü tapılan ədəddir. c=sqrt(pow(a,2)+pow(b, 2));

L=c;

3. Katetlərin uzunluğunu tapmaq qalır. Şəkildə bunun necə olunduqu görünür.

a=y-y0;

b=x-x0;

```
#include <iostream>
#include <math.h>
using namespace std;
void main() {

  int x0, y0, r1, r2, x, y;
  float L;


cout<<"Input coordinates of circle's center (X0, Y0):";
  cin>>x0>>y0;
  cout<<"Input circle radiuses R1 and R2:";
  cin>>r1>>r2;
  cout<<"Input point coordinates (X, Y): ";
  cin>>x>>y;

// düstürün daxil edilməsi
  L = sqrt(pow(x - x0, 2) + pow(y - y0, 2));

//nəticələrin analizi
  if ((r1 < L)&& (L < r2 )) {
       cout<<"This point is situated inside the circle.\n";
     }
 else {
       cout<<"This point is not situated inside the circle.\n";
     }
}
```

10. switch çoxluqlu seçim strukturu.

Biz artıq şərtləri analiz edən konstruksiya ilə - if konstruksiyası və həmçinin ternar operator ilə tanışıq. Bir dənədə seçim operatoru switch operatorudur. Təsəvvür edin ki, elə bir proqram yazmaq lazımdır ki, menüsu 5 punktdan ibarətdir. Misal üçün uşaqlar üçün vurma, çıxma və s. bacaran proqram. Seçim işini if else if pilləkəni köməkliyi ilə realizasiya etmək olar, bax belə:

```
# include <iostream>
using namespace std;
void main(){

float A,B,RES;
cout<<"Enter first digit:\n\";
cin>>A;
cout<<"Enter second digit:\n":;
cin>>B;

// proqram menüsunun realizasiyası
char key;
cout<<"\nSelect operator:\n»;
cout<<"\n + - if you want to see SUM.\n";
cout<<"\n - - if you want to see DIFFERENCE.\n";
cout<<"\n * - if you want to see PRODUCT.\n0";
cout<<"\n / - if you want to see QUOTIENT.\n";


cin>>key;

if(key=='+') { // istifadəçi üstəgəl seçdidkə
 .      RES=A+B;
 .      cout<<"\nAnswer: "<<RES<<"\n";
  }
else if(key=='-'){ // istifadəçi çıxmanı seçikdə
```

```
.       RES=A-B;
.       cout<<"\nAnswer: "<<RES<<"\n";
 }
else if(key=='*'){ // istifadəçi vurmanı seçdikdə
.       RES=A*B;
.       cout<<"\nAnswer: "<<RES<<"\n";
 }
else if(key=='/'){ // İstifadəçi bölməni seçdikdə
.       if(B){ // bölücü sıfra bərabər olmadıqda
.               RES=A/B;
.               cout<<"\nAnswer: "<<RES<<"\n";
.       }
.       else{ // sıfra bölündükdə
.               cout<<"\nError!!! Divide by null!!!!\n";
.       }
 }
else{ // daxil edilən operator səhv olduqda
.       cout<<"\nError!!! This operator isn't correct\n";
 }
}
```

Yuxarıda göstərilən misal tam düzgündür, sadəcə bir az böyük görsənir. Bu kodu məs switch istifadə edərək xeyli asanlaşdırmaq olar. O, dəyişkənin qiymətini bir sıra qiymətlə müqayisə edib uyğunluq tapdıqda müəyyən hərəkəti etmək imkanı verir.

Ümumi sintaksis və iş prinsipi.

İlkin olaraq operatorun ümumi sintaksisinə nəzər yetirək:

```
switch(İfadə){
case Qiymət1:
          Hərəkət1;
              break;
case qiymə2:
          Hərəkət2;
              break;
case qiymət2:
          Hərəkət3;
              break;
 ....
default:

              break;
}
```

Gəlin bu yazı formasını analiz edək:

1. İfadə - uyğunluğunu yoxlamamız lazım olan məlumatlar. Burda dəyişkən göstərilə bilər (amma yalnız char tipli və ya tamədədli), və ya nəticəsi tamədədli məlumat olan ifadə.

2. case qiymət1, case qiymət2, case qiymət3 — ifadənin yoxlanıldığı tamədədli və simvollu daimi qiymətlər.

3. hərəkət1, hərəkət2, hərəkət3 — ifadənin qiyməti case-in qiyməti ilə üst üstə düşdükdə yerinə yetirilməli olan hərəkətlər.

4. Əgər təsadüf oldusa və təsadüf edən case ilə əlaqəli hərək uğurla yerinə yetirildi isə, switch öz işinə sona çatdırır və proqram switch operatorunun fiqur mötərəzəsindən sonra olan sətrə keçir. Bu funksiyaya break operatoru cavabdehdir məs o switch-i dayandırır.

5. Əgər analiz zamanı uyğunluq tapılmasa default seksiyası işə düşür və standart hərəkətlər yerinə yetirilir. default operatoru else operatorunun analoqudur.

İndisə isə gəlin baxaq mövzunun əvvəlində verilən nümunəni necə asanlaşdırmaq olar.

Nümunənin optimizasiyası

```
# include <iostream>
using namespace std;
void main(){

// dəyişkənin elanı və klaviaturadan qiym\tin daxil edilməsi
float A,B,RES;
cout<<"Enter first digit:\n";
cin>>A;
cout<<"Enter second digit:\n";
cin>>B;

// proqram menüsunun realizasiyası
char key;
cout<<"\nSelect operator:\n";
cout<<"\n + - if you want to see SUM.\n";
cout<<"\n - - if you want to see DIFFERENCE.\n";
cout<<"\n * - if you want to see PRODUCT.\n";
cout<<"\n / - if you want to see QUOTIENT.\n";


cin>>key;

// key dəyişkəninin qiyməti yoxlanlır
switch(key){
case  '+':    // istifadəçi üstəgəl seçdikdə
       RES=A+B;
       cout<<"\nAnswer: "<<RES<<"\n";
       break; // остановка switch
case  '-':    // istifadəçi çıxma seçdikdə
       RES=A-B;
       cout<<"\nAnswer: "<<RES<<"\n";
       break; // остановка switch
case  '*':    // istifadəçi vurma seçdikdə
       RES=A*B;
       cout<<"\nAnswer: "<<RES<<"\n";
       break; // остановка switch  case  '/':
       // istifadəçi bölmə seçdikdə
       if(B){ // bölücü sıfra bərabər deyilə
                RES=A-B;
                cout<<"\nAnswer: "<<RES<<"\n";
       }
       else{ // bölücü sıfra bərabərdirsə
                cout<<"\nError!!! Divide by null!!!!\n";
       }
       break; // switch dayanacaq
default:    // əgər daxil edilən simvol səhvdirsə
       cout<<"\nError!!! This operator isn't correct\n";
       break; // switch dayanacaq
  }
}
```

Gördüyünüz kimi kod indi əvvəlkindən sadə görünür və onu oxumaq daha rahatdır.

switch operaturunun istifadəsi yetəri qədər sadədir amma onun işinin bəzi xüsusiyyətlərinin bilmək lazımdır:

1. Əgər case-də simvollu ifadələrdən istifadə olunursa onlar tək dırnaqda göstərilməlidir, tam ədələr isə dırnaqsız.

2. Оператор default operatoru switch sisteminin istənilən yerində saxlanıla bilər, istənilən halda heç bir uyğunluq olmadıqda yerinə yetiriləcək. Amma default-u bütün konstruksiyanın sonuna qoymaq daha düzgün hesab edilir

```
switch(ifadə){
case qiymət1:
      hərəkət1;
      break;
case qiymət2:
      hərəkət2;
      break;
default:
```

```
      действие_по_умолчанию;
      break;
case значение3:
      действие3;
      break;
}
```

3. Siyahıda ən son operatordan sonra (case və ya default olsa da) break operatorunu göstərməmək olar.

```
switch(ifadə){
case qiymət1:
      hərəkət1;
      break;
case qiymət2:
      hərəkət2;
      break;
default:
      defaul hərəkət;
      break;
case qiymət3:
      hərəkət3;
}  switch(ifadə){
case qiymət1:
      hərəkət1;
      break;
case qiymət2:
      hərəkət2;
      break;
case qiymət3:
      hərəkət3;
      break;
default:
        default hərəkət; }
```

4. default operatorunu ümumiyyətlə göstərməmək olar. Uyğunluq tapılmadıqda sadəcə heçnə baş vermiyəcək.

```
switch(ifadə){
case qiymət1:
      hərəkət1;
       break;
case qiymət2:
      hərəkət2;
       break;
case qiymət3:
      hərəkət3;
       break;
}
```

5. Əgər yoxlanılan ifadənin müxtəlif qiymətləri üçün eyni hərəkətləri icra etmək lazım olarsa, bir neçə işarəni ardıcıl yazmaq olar. Hərfi qiymətlər sistemini rəqəmsala çevirən proqram nümunəsinə baxaq.

```
# include <iostream>
using namespace std;
void main(){
// hərfi qiymətin saxlanılması üçün dəyişkənin elanı
char cRate;

// Hərfi qiyməti daxil etmə xahişi
cout<<"Input your char-rate\n";
cin>>cRate;

//daxil edilmiş qiymətin analizi
switch (cRate) {
case 'A':
case 'a':
        // A qiyməti və ya a barəbərdi 5
        cout<<"Your rate is 5\n";
        break;
case 'B':
case 'b':
        // B qiyməti və ya b barəbərdi 4
        cout<<"Your rate is 4\n";
        break;
case 'C':
case 'c':
        // C qiyməti və ya c barəbərdi 3
        cout<<"Your rate is 3\n";
        break;
case 'D':
case 'd':
        // D qiyməti və ya d barəbərdi 2
        cout<<"Your rate is 2\n";
        break;
default:
        // qalan simvollar səhvdir
        cout<<"This rate isn't correct\n";
    }
}
```

Nümunənin xüsusiyyəti ardıcıl gələn case-lərin köməkliyi ilə registr-müstıqilliyinə nail olmaq olur. Yani istifadəçinin böyük və ya balaca hərif yazacaqının fərqi yoxdu.

Geniş yayılmış səhvРаспространенная ошибка.

Switch operatoru barədə əsas hər şey deyildi. İndi isə proqrammistin bu operatordan istifadə zamanı ortaya çıxa biləcək bəzi problemləri müzakirə edək.

Break-i sonuncudan başqa case-in istənilın blokunda təsadüfən yaddan çıxarsaz və bu blok nəticədə işləsə, switch-in icrası dayanmayacaq. Case operatorunun artıq icra olunmuşun ardı ilə gedən blokuda yoxlanmasız icra olunacaq.

Səhvin nümunəsi.

```
# include <iostream>
using namespace std;
void main(){

// proqram menüsunun realizasiyası
int action;
cout<<"\nSelect action:\n";
cout<<"\n 1 - if you want to see course of dollar.\n";
cout<<"\n 2 - if you want to see course of euro.\n";
cout<<"\n 3 - if you want to see course of rub.\n";

// istifadəçinin seçiminin gözlənilməsi
cin>>action;

//action dəyişkəninin qiyməti yoxlanılır
switch(action){
case  1:    // istifadəçi dollar seçdikdə
        cout<<"\nCource: 5.2 gr.\n";
        break; // switch dayanacaqı
case  2:     // istifadəçi avro seçdikdə
        cout<<"\nCource: 6.2 gr.\n";
         //break;  закомментированна остановка switch
case  3:    // istifadəçi rubl seçdikdə
        cout<<"\nCource: 0.18 gr.\n";
        break; // switch dayanacaqı
default:    // seçim düzgün olmadıqda
        cout<<"\nError!!! This operator isn't correct\n";
        break; // switch dayanacaqı
 }
}
```

Səhv menünun ikinci punktu seçildikdə olacaq. 2 qiymətli case-də break dayanmaq operatoru şərh edilib. Ekranda belə səhvin nəticəsi aşağıda ki kimi görsənəcək:

```
Select action:

 1 - if you want to see course of dollar.

 2 - if you want to see course of euro.

 3 - if you want to see course of rub.
2

Cource: 6.2 gr.

Cource: 0.18 gr.
Press any key to continue
```

Lazımi məlumatdan başqa ekranda case blokunda olan səhv konstruksiyadan sonra yerləşən göründü. Belə səhvərlə ehtiyatlı olmaq lazımdır çün ki onlar kompilyasiya etapında səhvlərə gətirir.

Bu günkü dərsdə biz sizinlə hər hansı məlumatların analizini eləməyi öyrəndik. İndi ev tapşırıqının həllinə keçə bilərsiniz. Uğurlar!

11. Ev tapşırıqı

1. Klaviaturada daxil edilmiş ədədin cütlüyünü yoxlayan proqram yazın.

2. a (a<100) natural ədədi verilib. Bu ədəddə olan rəqəmlərin sayını və bu rəqəmlərin cəmini göstərən proqram yazın.

3. Məlumdur ki, 1 düym 2.54 sm-ə barəbərdir. Düymları santimetrə və əksinə çevirə proqram yazın. İstifadəçi ilə dialoqu menü sistemi ilə realizasiya edin.

4. Populyar Milyonçu" oyunu realizasiya edən proqram yazın.