ШАГ компьютерная академия

# ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

# Dərs №3

# C

dilində
proqramlaşdırma

**Mündəricat**

# 1. Silsilə anlayışı

Tez-tez həyatda və proqram yazdıqda hər hansı bir hərəkətin təkrarlanması tələbi yaranır. Məsələn, qabların yuyulması alqoritmini yazaq.

```
0. Tasdan boşqabı götürmək.
1. Qab şampunu ilə boşqabı sabunlamaq.
2. Boşqabı lif ilə sürtmək.
3. Sabunu boşqabdan yumaq.
4. Boşqabı silmək.
5. Boşqabı rəfə qoymaq.
6. Proqramın sonu.
```

Bu birinci baxışdan məntiqli alqoritmdə balaca bir səhv var — boşqab bir dənədən çox olsada yalnız biri yuyulacaq. Bu proqramın bütün əməliyyatları bir xəttlə eləməyi ilə əlaqədardır — yuxarıdan aşağı, ardıcıl olaraq. Deməli, biz fikirləşməliyik proqramı necə müəyyən əməliiyatları yerinə yetirməyə məcbur edə və bununla belə həmçinin təkrarların sayını mü\yy\n edə bilərik. Düzgün alqoritm belə olacaq:

```
0. Qabı götürmək.

1. Qabı qabyuyanla sabunlamaq.

2. Qabı lif ilə silmək.

3. Sabunu qabın üzərindən yumaq

4. Qabı qurulamaq.

5. Qabı rəfə qoymaq.

6. Yenə çirkli qab qaldıqda 0 mərhələyə keçmək.

7. Proqramın sonu
```

Nəzərinizə çatdıraq ki, əməliyyatı təkrar edib etmiyəcəyimizi təyin etmək üçün ilk öncə "Hələ də çirkli qab varmı" şərtindən istifadə olunur. əgər bu şərt həqiqidirsə əməliyyat icra olunur, əks halda isə, yalan olduqda, alqoritmin sonrakı, 7-ci bəndi icra olunur.

Beləliklə biz qərara gəldik ki, bizə özündə təkrar üçün lazımlı olan əməliiyatları saxlayan konstruksiya lazımdır. Bununla belə təkrarların sayı həmin bu konstruksiyada saxlanınan şərt ilə təyin edilməlidir

Özümüzdəbilmədən biz DÖVRÜN anlayışını verdik. Bir daha təkrarlayaq!!!

**Dövr — proqramlaşdırma dilinin şərtindən asılı olaraq bir və digər əməliyyatın lazım olan qədər təkrarlanmasını təşkil etməyə kömək edən xüsusi operatoru.**

Qeyd: Yeri gəlmişkən dövrün başqa adı silsilə - təkrarlanma konstruksiyasıdır. əməliiyatın hər təkrarı isə DÖVRÜN ADDIMI və ya İTERASİYA.

C dilində dövr kimi formanın bir neçə realizasiyası var.Bu dərsdə, söhbət iki realizasiyadan gedəcək — **while** и **do while**.

# 2. *while silsiləsi*

while dövrünün ümumi sintaksisi və icra ardıcıllıqı

```
while(утверждение)
{
                        təkrarlanma üçün əməliyyət
}
```

1. İlk öncə müddəanın yoxlanması keçirilir.
2. əgər yumru mötərəzədəki ifadə həqiqidirsə fiqur mörəzənin içində olan əməliyyat icra olunur.
3. əgər yumru mötərəzədəki ifadə yalandırsa proqram bağlanan fiqur mötərəzədən sonrakı sətrə keçir
4. əgər yumru mötərəzədəki ifadə həqiqi idirsə və əməliyyat icra olundusa, yenidən ifadənin yoxlanması keçirilir.

Gördüyünüz kimi ifadənin yoxlanması hər dəfə dövr icra olunduqda keçirilir. O, həqiqi olmayan kimi dövrün icrası dayandırılır. Fikir verin ki, ifadə əvvəldən yalan olduqda, dövrün içində ki əməliyyat heç vaxt icra olunmur

## Nümunəyə baxaq

Tutaq ki, kimsə dünyanın 7 möcüzəsi haqqında inşa yazmalıdır. Bunu etməzdən öncə gedib hər bir möcüzəni görmək lazımdır. Və, ancaq bundan sonra bu haqqda yazmaq. Lahiyənin adıa **Miracles**.

```
#include <iostream>
using namespace std;
void main()
{
 //idarə edən dəyişkənin elanı
 int counter=0;
 while(counter<7)// ifadənin
yoxlanması    {
                counter++;// idarə edən dəyişkənin dəyişdirilməsi

                //təkrarlanma üçün əməliyyət

                // siz ... dünya möcüzəsini gördüz

                cout<<"You seen»<<counter<<" miracle of world!!!\n";

             }
cout<<"Now, you can start your work.\n";
}
```

İndi bizim misalın necə işləyəcəyinə nəzər yetirək

1. əvvəldən 0 bərabər olan dəyişkəni elan edək

2. Sonra silsilənin şərtində dəyişkənimizin qiymətini yoxlayırıq. Silsilənin icra olunub olunmayacaqı məs bu qiymətdən asılı olduğundan, bu dəyişkən silsiləni idarə edən adlanır.

3. Silsilənin qiymətini bir vahid artırırıq

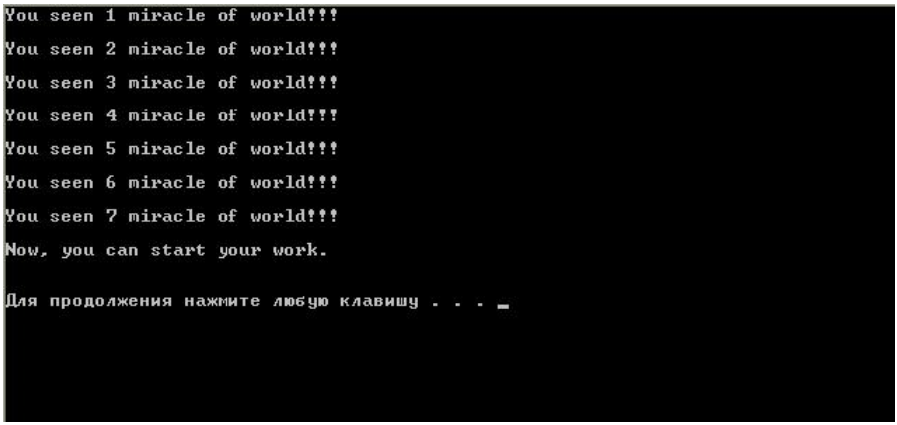
Qeyd: Bu əməliyyat mütləqdir, çün ki silsiləni idarə edən dəyişkənin qiyməti dəyişdirilməsə, ifadənin yoxlama nəticəsidə heç vaxt dəyişməyəcək. Bu əbədi silsilə adlanın çox geniş yayılmış bir səhvdir. əgər silsilənin ifadəsi

həqiqi, idarəedən dəyişkən isə həmişə eyni qiymətə malikdirsə, ifadə həmişə həqiqidir. Təsəvvür edin, çirkli boşqablar heç vaxt bitmir - onların sayıdaim artır. Qabyuyan nə qədər işləyə biləcək?! O qədərdə çox yox, düzdü? Bax proqramda beləcə çox dözə bilməyəcək və əbədi silsilə qoşulduqdan bir az sonra icra mərhələsində səhv çıxaracaq. Bələ səhvlərin olmaması üçün nəzarət etmək lazımdır ki, silsilənin içində silsilənin bədəninin içində idarə edən dəyişkən dəyişsin.

4. Sonra isə ekrana dəyişkənin cari qiyməti baxılmış dünya mözüsənin nömrəsi barədə məktub kimi çıxarılacaq
5. Yenədə şərtə qayıdıb idarəedən dəyişkənin qiymətini yoxlayırıq.

Silsilə öz işini dəyişkənin qiyməti yeddiyə bərabər olanadək davam edəcək. Bu halda ekrana «You seen 7 miracle of world!!!» çıxarılacaq, sonra proqram şərtin yoxlanılmasına keçəcək. 7<7 — yalandır. Proqram artıq silsiləyə daxil olmayacaq və «Now, you can start your work.» keçəcək.

İcra prosesində ekranda növbəti yazını görəcəyik:




İndi biz sizinlə C dilində silsilələrdən biri ilə tanış olduq. Ümid edirik ki, çətin olmadı. Dərsin növbəti bölməsində while alternativ konstruksiyanı öyrənəcəyik.

# 3. do while konstruksiyası

do while ümumi sintaksisi və iş prinsipi:

```
do
{
                    действие;
}
while(условие);
```

do while silsiləsi while silsiləsinə oxşayır. Fərq ondan ibarətdir ki, while-da yoxlama dərhal silsiləyı girdikdə sonra keçirilir və ancaq bundan sonra şərt həqiqi olduqda əməliyyət ica olunur. Do while-da hər halda əməliyyat icra olunur və ancaq sonra şərt yoxlanılır. Şərt həqiqi olduqda icra davam edir, əks halda while-dan sonra ələn operatora keçir. Digər sözlə while-dan fərqli olaraq dp while-da əməliyyat ən azından bir dəfə icra olunur. Gəlin buna sxemda baxaq:

## do while-dan praktikada istifadə

Tutaq ki, bizə hər hansı əməliyyatın bir neçə dəfə ardıcıl təkrarlanması seçimi verilir. əvvəlcə bu məsələni while ilə həll edək, sonra isə do while ilə. Lahiyənin adı Calc.

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

  // əməliyyatın seçimi sorğusu:
cout<<"\nSelect operation:\n";
cout<<"\n 1 - if you want to see SUM.\n";
cout<<"\n 2 - if you want to see DIFFERENCE.\n";
cout<<"\n 3 - if you want to exit.\n";
cin>>answer;

  while(answer!=3){ // ışərtin yoxlanması
        switch(answer){
                case  1:    // istifadəçi toplama seçdikdə
                cout<<"Enter first digit:\n";
                cin>>A;
                cout<<"Enter second digit:\n";
                cin>>B;
                RES=A+B;
                cout<<"\nAnswer: "<<RES<<"\n";
                break; // switch dayanacaq
                case  2:     // istifadəçi çıxma seçdikdə
                cout<<"Enter first digit:\n";
                cin>>A;
                cout<<"Enter second digit:\n";
                cin>>B;
                RES=A-B;
                cout<<"\nAnswer: "<<RES<<"\n";
                break; // switch dayancaq
        case  3: // çıxmaqı seçdikdə
                cout<<"\nEXIT!!!\n";
                break;
           default:   seçilmiş əməliyyat səhvdirsə
                cout<<"\nError!!! This operator isn't correct\n";
               }
```

```
// əməliyyatın seçim sorğusu
       cout<<"\nSelect operation:\n";
       cout<<"\n 1 - if you want to see SUM.\n";
       cout<<"\n 2 - if you want to see DIFFERENCE.\n";
       cout<<"\n 3 - if you want to exit.\n";
                cin>>answer;
       }
      cout<<»\nBye....\n»;
}
```

bu nümunədə istifadəçiyə əməliyyat seçmək təklif edilir. Sonra daxil etdikdən yoxlama başlayır: əgər bu əməliyyat — proqramnan çıxmadırsa — proqram bağlanır, əks halda silsiləyə giriş baş verir, əməliyyatın analizi və riyazi əməliyyatın icrası. Sonra proqram istifadəçidən yenə onun nə etmək istədiyini soruşur

Bu kod optimal həll deyil. Gördüyünüz fraqment

```
// запрос на выбор операции
   cout<<»\nSelect operation:\n»;
   cout<<»\n 1 - if you want to see SUM.\n»;
   cout<<»\n 2 - if you want to see DIFFERENCE.\n»;
   cout<<»\n 3 - if you want to exit.\n»;
   cin>>answer;
```

bir neçə dəfə təkrarlanır. Bu halda do while-dan istifadə etməyinə dəyər. Bu konstruksiya kodu lazımi hala gətirəcək. Lahiyının adı CalcDoWhile.

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

 do{ // silsiləyə giriş

        // əməliyyatın seçim sorğusu
        cout<<"\nSelect operation:\n";
        cout<<"\n 1 - if you want to see SUM.\n";
      cout<<"\n 2 - if you want to see DIFFERENCE.\n";
      cout<<"\n 3 - if you want to exit.\n";
                                cin>>answer;

        // əməliyyatın analizi
        switch(answer){
         case 1: // istifadəçi toplamanı seçdikdə
                  cout<<"Enter first digit:\n";
                  cin>>A;
                  cout<<"Enter second digit:\n";
                  cin>>B;
                  RES=A+B;
                  cout<<"\nAnswer: "<<RES<<"\n";
                  break; // остановка switch
        case 2: // istifadəçi çıxma seçdikdə
                  cout<<"Enter first digit:\n";
                  cin>>A;
                  cout<<"Enter second digit:\n";
                  cin>>B;
                  RES=A-B;
                  cout<<"\nAnswer: "<<RES<<"\n";
                  break; // остановка switch
        case 3: // istifadəçi çıxmaqı seçdikdə
                  cout<<"\nEXIT!!!\n";
                  break;
           default:   seçilmiş əməliyyat düzgün deyilsə
                  cout<<"\nError!!! This operator isn't correct\n";
   }

        } while(answer!=3);
cout<<"\nBye....\n";

}
```

Yuxarıda yazılanlardan siz anlamalısız ki, bu günkü dərsdə göstərilən konstruksiyanın hər ikisi çox xeyirlidir. Siz sadəcə məqsəddən asılı olaraq bir və ya digərini seçməyi öyrənməlisiniz.

İndi artıq silsilələr ilə tanış olduqdan sonra siz dərsin növbəti bölməsinə keçə bilərsiniz. Biz sizin üçün bu günkü mövzuya aid bir neçə misal hazırlamışıq.

# 4. Dərsə nümunələr

**Nümunə 1.**

**Məsələ.**

1-dən daxil 5-dək bütün tam ədədlərin toplamını tapan proqram yazmaq. Lahiyənin adı Summ. **Realizasiya kodu.**

```
#include <iostream>
using namespace std;
void main(){
    int BEGIN=1; // toplanan ədələrin diapazonunun əvvəli
    int END=5; // toplanan ədələrin diapazonunun sonu
    int SUMM=0; // toplamın yığılması üçün dəyişkən
    int i=BEGIN; // silsiləni idarə edən dəyişkən

     // şərtin yoxlanması
while(i<=END){ //(idarə edən dəyişkənin dipazonun sonu ilə müqaisəsi)
SUMM+=i;// toplamın yığılması
i++;// idarə edən dəyişkənin dəyişilməsi
     }

     // nəticənin göstərilməsi
     cout<<"Result - "<<SUMM<<"\n\n";
}
```

**Koda şərhlər.**

Koda şərh kimi biz silsilənin hər iterasiyanı xırdalıqına kimi təsvir edən cədvəl təqdim edirik:

| ВХОДНЫЕ ДАННЫЕ | | | |
|---|---|---|---|
| BEGIN=1 | | END=5 | |
| РАБОТА ЦИКЛА | | | |
| номер итерации | i | условие | SUMM |
| 1 | 1 | 1<=5 - true | 0+1=1 |
| 2 | 2 | 2<=5 - true | 1+2=3 |
| 3 | 3 | 3<=5 - true | 3+3=6 |
| 4 | 4 | 4<=5 - true | 6+4=10 |
| 5 | 5 | 5<=5 - true | 10+5=15 |
| 6 | 6 | 6<=5 - false | X |
| SUMM=15 | | | |

Cədvələ baxdıqda idarə edən dəyişkənin həmçinin toplama üçün qiymətləri ardıcıl seçən dəyişkən rolunu oynamasını başa düşmək o qədərdə çətin deyil.

**Qeyd:** İdarə edən dəyişkənin yalız bir vahid dəyişə bilməsi fikri - geniş yayılmış səhvdir. Bu belə deyil. əsas dəyişkənin istənilən məntiqi üsul ilə dəyişməsidir.

Nümunə 2.
Məsələ.
Ekrana 6 ulduzdan ibarət xətt çıxaran proqram yazmaq. Lahiyənin adı Line.
Realizasiya kodu

```
#include <iostream>
using namespace std;
void main(){
	int COUNT=5; // ulduzların sayı (xəttin uzunluqu)
	int i=0; // silsiləni idarə edən dəyişkən

	while(i<=COUNT){ // şərtin yoxlanması

	cout<<"*";// ulduzun çıxarılması
	 i++;// idarə edən dəyişkənin dəyişdirilməsi
}
cout<<"\n\n";
}
```

## Koda şərh.

1. İdarə edən dəyişkən şərtin yoxlanması zamanı artıq çəkilmiş ulduzların sayına bərabərdir. i dəyişkəninin hər * çıxarıldıqdan sonra bir vahid artmasına görə belə olur.

2. Silsilə i=5 olduqda dayancaq, və bu çəkilmiş * sayına bərabər olacaq.

İndi isə ev tapşırıqına keçmə vaxtıdır!

# 5. Ev tapşırıqı

1. Ekrana simvollardan üfüqi xətt çıxaran proqram yazmaq. Simvolların sayı, hansı simvoldan istifadə etmək və hansı xəttin olacaqını — üfüqi və ya şaquli — istifadəçi seçir.

2. İstifadəçi tərəfindən göstərilən diapazonda bütün tam tək ədədlərin toplamını hesablayan proqram yazmaq

3. n natural ədədi verilib.

n-nin qeyri-mənfi tam ədədlərinin faktorialını hesablayan proqram yazmaq (yani tam 0 böyük ədədlər). Faktorialın tapılma düstüru aşağıda gətirilib

```
n! = 1*2*3*....*n, (n ədədinin faktorialının hesablanma düsturu)
0! = 1 (0 faktorialı 1 bərabərdir (faktorialın qaydalarına görə))
```