Оператори циклу

- for
- while
- do while

Оператори циклу

Оператори циклу використовують для здійснення багаторазового повторення деякої послідовності дій.

Кожен цикл складається з тіла циклу, тобто операторів, що виконуються декілька разів.

Один прохід циклу називається **ітерацією**.

У мові C/C++ існують три оператори циклу: **for, while, do while**.

Вони поділяються на дві групи:

- оператори циклу с передумовою: **for, while**
- оператор циклу с післяумовою do while

Зауваження: Оператори **for** і **while** C/C++ повністю взаємозаміняємі.

Рекомендується, якщо відома кількість ітерацій то застосовувати for, наприклад, для проходження елементів масиву. У противному випадку застосовувати while. Наприклад, поки не кінець файлу, виконувати читання з нього.

Оператор for

Загальний синтаксис оператора цику for

```
for ( [блок1] ; [блок2-умова];  [блок3-дія] )
    оператор;
```

де **- блок 1** — це створення чи оновлення змінних (зазвичай так званих лічильників) циклу, що зазвичай використовується для встановлення початкового значення (необов'язковий параметр);

- **блок 2** — це вираз умови, що визначає, за якої умови цикл буде повторюватися (необов'язковий параметр);

- блок 3 — це вираз, який зазвичай (хоча не обов’язково) задає зміну лічильника циклу (необов'язковий параметр).

Алгоритм виконання оператору **for:**

- Одноразово виконується блок1, в якому створюються або оновлюються початковим значенням змінні-лічильники циклу. Тобто цей блок виконується тільки один раз перед першою ітерацією циклу.
- Виконується блок 2. Тобто перевіряється умова (чи являється вираз істинний). Умова істина, якщо результат виразу не дорівнює нулю.
- Якщо блок 2 істинний, то виконується оператор (тіло циклу) і блок 3. Після чого знову виконується блок 2. Це повторюється до тих пір поки вираз блоку 2 буде істинний.
- Якщо блок 2 прийме значення «неправда» (вираз-умова дорівнює нулю) тоді буде реалізований вихід із циклу.
- Оскільки перевірка умови виконується перед циклом, то цикл може не виконатися жодного разу, якщо умова відразу буде помилковою.

Оператор for

рекомендований синтаксис

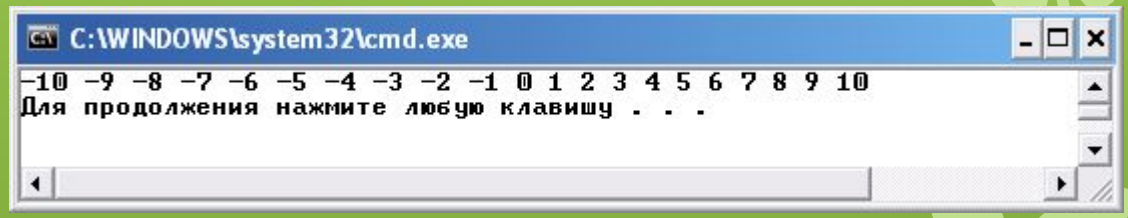
```
for ( [ініціалізація] ; [вираз-умова];  [вираз-дія] )
{
    деякий код/оператори;
}
```

де – ініціалізація – створення змінних циклу;
- вираз-умова – логічна вираз, який задає умову виконання тіла циклу;
- зміна лічильника циклу.

Приклад 1

Вивести усі цілі числа від -10 до 10

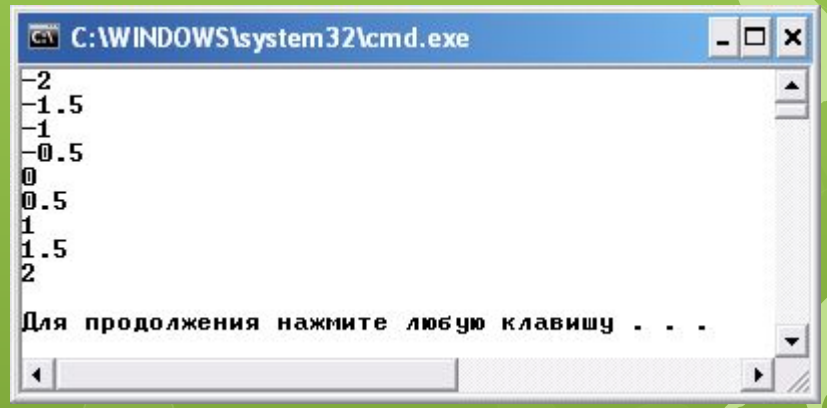
```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main()
{
    for (int i=-10; i<=10; ++i)
    {
        cout<<i<<" " ;
    }
    cout<<endl;
    system("pause");
    return 0;
}
```



Приклад 2

Вивести усі числа від -2 до 2 з кроком 0.5

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main()
{
    const double E=0.0001;
    for (double x=-2; x<2+E; x+=0.5)
    {
        cout<<x<<"\n" ;
    }
    cout<<endl;
    system("pause");
    return 0;
}
```



Зауваження

Не слід порівнювати дійсні числа операціями

==, <=, >=

тому що вони в памяті зберігаються з певною точністю.

Рекомендуеється їх порівнювати з деякою точістю.

Наприклад, змінні a і b рівні, якщо

$$|a-b|<E,$$

де E допуск порівняння

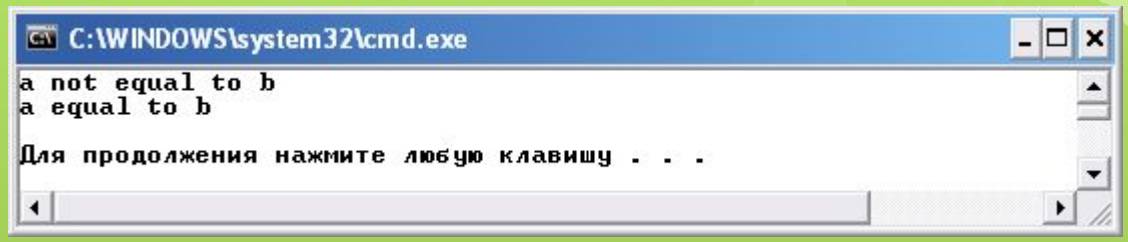
Мовою C, це виглядає

```
const E=0.0001;
fabs(a-b)<E
```

Приклад 3

Порівняння дійсних чисел

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
    double a = 2.0/3.0;
    float b  = 2.0/3.0;
    if (a == b)
    {
        cout<<"a equal to b"<<endl;
    }
    else
    {
        cout<<"a not equal to b"<<endl;
    }
    const double E=0.0001;
    if ( fabs(a- b) <E)
    {
        cout<<"a equal to b"<<endl;
    }
    else
    {
        cout<<"a not equal to b"<<endl;
    }
    cout<<endl;
    system("pause");
    return 0;
}
```

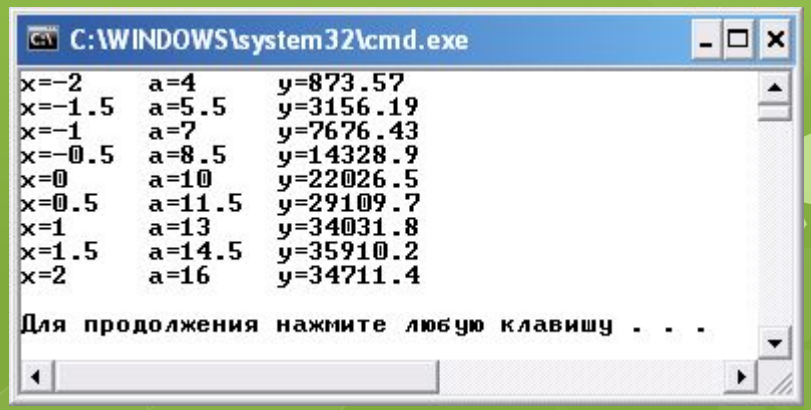


Приклад 4

Обчислити значення виразу $y=\frac{e^a}{a^x}$, де x змінюється на інтервалі $x=[-2:2]$, з кроком $h_x=0.5$. А а отримує початкове значення 4 і змінюється одночасно з x на величину 1,5.

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
   const double E=0.0001;

   for (double x=-2, a=4.0; x<2+E; x+=0.5, a+=1.5)
   {
      double y = exp(a)/pow(a,x);
      cout<<"x="<<x<<"\ta="<<a<<"\ty="<<y<<endl;
   }
   cout<<endl;
   system("pause");
   return 0;
}
```



Приклад 5

Обчислити значення виразу $y=\frac{e^a}{a^x}$, де x змінюється на інтервалі $x=[-2:2]$, з кроком $h_x=0.5$. А а мінюється на інтервалі $x=[4:10]$, з кроком $h_a=1.5$.

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
   const double E=0.0001;
   const double hx=0.5;
   const double ha=0.5;
   for (double x=-2; x<2+E; x+=hx)
      for (double a=4; a<10+E; a+=ha)
      {
         double y = exp(a)/pow(a,x);
         cout<<"x="<<x<<"\ta="<<a<<"\ty="<<y<<endl;
      }
   cout<<endl;
   system("pause");
   return 0;
}
```

C:\WINDOWS\system32\cmd.exe

```
x=-2     a=4      y=873.57
x=-2     a=4.5    y=1822.85
x=-2     a=5      y=3710.33
x=-2     a=5.5    y=7401.93
x=-2     a=6      y=14523.4
x=-2     a=6.5    y=28102.2
x=-2     a=7      y=53735
x=-2     a=7.5    y=101702
x=-2     a=8      y=190781
x=-2     a=8.5    y=355092
x=-2     a=9      y=656350
x=-2     a=9.5    y=1.20572e+006
x=-2     a=10     y=2.20265e+006
x=-1.5   a=4      y=436.785
x=-1.5   a=4.5    y=859.298
x=-1.5   a=5      y=1659.31
x=-1.5   a=5.5    y=3156.19
x=-1.5   a=6      y=5929.17
x=-1.5   a=6.5    y=11022.6
x=-1.5   a=7      y=20309.9
x=-1.5   a=7.5    y=37136.5
x=-1.5   a=8      y=67451.4
x=-1.5   a=8.5    y=121796
x=-1.5   a=9      y=218783
x=-1.5   a=9.5    y=391186
x=-1.5   a=10     y=696538
x=-1     a=4      y=218.393
x=-1     a=4.5    y=405.077
x=-1     a=5      y=742.066
x=-1     a=5.5    y=1345.81
x=-1     a=6      y=2420.57
x=-1     a=6.5    y=4323.42
x=-1     a=7      y=7676.43
x=-1     a=7.5    y=13560.3
x=-1     a=8      y=23847.7
x=-1     a=8.5    y=41775.5
x=-1     a=9      y=72927.8
x=-1     a=9.5    y=126917
x=-1     a=10     y=220265
x=-0.5   a=4      y=109.196
x=-0.5   a=4.5    y=190.955
x=-0.5   a=5      y=331.862
x=-0.5   a=5.5    y=573.853
x=-0.5   a=6      y=988.195
x=-0.5   a=6.5    y=1695.79
x=-0.5   a=7      y=2901.42
x=-0.5   a=7.5    y=4951.53
x=-0.5   a=8      y=8431.42
x=-0.5   a=8.5    y=14328.9
x=-0.5   a=9      y=24309.3
x=-0.5   a=9.5    y=41177.4
x=-0.5   a=10     y=69653.8
x=0      a=4      y=54.5982
x=0      a=4.5    y=90.0171
x=0      a=5      y=148.413
x=0      a=5.5    y=244.692
x=0      a=6      y=403.429
x=0      a=6.5    y=665.142
x=0      a=7      y=1096.63
x=0      a=7.5    y=1808.04
x=0      a=8      y=2980.96
x=0      a=8.5    y=4914.77
x=0      a=9      y=8103.08
x=0      a=9.5    y=13359.7
x=0      a=10     y=22026.5
x=0.5    a=4      y=27.2991
x=0.5    a=4.5    y=42.4345
x=0.5    a=5      y=66.3724
x=0.5    a=5.5    y=104.337
```

Оператор while

Загальний синтаксис оператора цику while

```
while (вираз-умова)
    оператор;
```

- де **вираз-умова** — це умова, яка визначає, за якої умови цикл буде повторюватися;
- де **оператор** — це тіло циклу (оператор буде повторюватися до тих пір поки істинний вираз-умова);

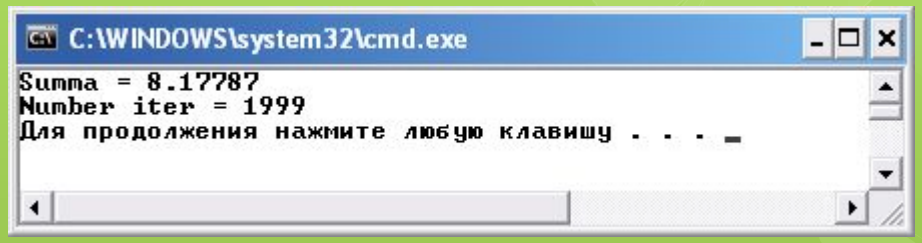
Рекомендований синтаксис

```
while (вираз-умова)
{
   операторі
}
```

Приклад 4

Обчислити суму ряду $S = \sum_{x=1}^{\infty} 1/x, \quad x = \overline{1,2,3,....\infty}$ з точністю E=0,0005

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
    const double E=0.0005;
    double S=0.0;
    double x=1.0;
    while(1.0/x >E)
    {
        S+=1.0/x;
        x+=1.0;
    }
    cout<<"Summa = "<<S<<endl;
    cout<<"Number iter = "<<x-1<<endl;
    system("pause");
    return 0;
}
```



Оператор циклу **do while**

Оператор циклу **do while** звичайно використовується в тих випадках, коли тіло циклу повинне виконуватися хоча б один раз, і має наступний загальний синтаксис

```
do
  оператор
while (вираз умова);
```

Рекомендований синтаксис

```
do
{
  оператори
}
while (вираз умова);
```

Виконується оператор **do** таким чином:

спочатку здійснюється вхід у тіло циклу і виконуються оператори (їх може буде декілька, після цього перевіряється умова і, якщо вона виконується, тобто "істина" true (не дорівнює нулю), то цикл повторюється, а якщо "неправда" false — здійснюється вихід з циклу.

Приклад 5

Обчислити вираз $y=\sin^{x}(x)$, для деякого x введеного з клавіатури.
Забезпечити повторне обчислення виразу для різних x не виходячи з програми

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
    double x,y;
    char ch;
    do
    {
        cout<<"Input x = ";
        cin>>x;
        y=pow(sin(x),x);
        cout<<"Result y= = "<<y<<endl;
        cout<<"Continue (y/n)? ";
        cin>>ch;
    }while(ch == 'y' || ch == 'Y');
    return 0;
}
```