



ПРОГРАММИРОВАНИЕ  
**НА ЯЗЫКЕ C**

# Урок №4

## Цикл for

### Содержание

<b>1. Конструкция for.....</b>	<b>3</b>
Изменение управляющей переменной .....	6
Оператор break .....	7
Оператор continue.....	8
<b>2. Практические примеры.....</b>	<b>11</b>
Пример 1 .....	11
Пример 2 .....	12
Пример 3 .....	13
<b>3. Домашнее задание .....</b>	<b>15</b>

# 1. Конструкция for

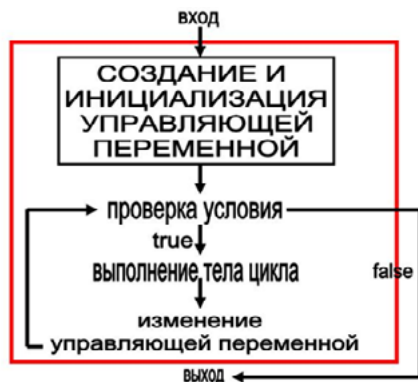
В прошлом уроке мы с вами познакомились с таким понятием как цикл и рассмотрели некоторые из конструкций, представляющих циклы в языке С. А, именно — while и do while. Сейчас мы рассмотрим еще одну разновидность цикла — оператор for. Данный оператор теоретически является полной аналогией while, а практически позволяет организовать цикл с более удобным управлением.

Общий синтаксис и принцип работы конструкции for

```
for (инициализация переменной;  
    проверка условия;изменение переменной)  
{  
    действие;  
}
```

Принцип выполнения цикла:

1. Инициализация переменной.
2. Проверка условия .
3. Выполнение действия, если условие истинно.
4. Если условие ложно, выполнение следующего за циклом оператору.
5. Если условие было истинно – изменение управляющей переменной.
6. Проверка условия. Далее снова пункт 3 или 4.



### Пример использования

Рассмотрим простой уже знакомый пример: с помощью цикла показать на экран цифры от 1 до 5 включительно. Только сделаем это с помощью оператора `for`.

```

#include <iostream>
void main()
{
    for(int i=1;i<=5;i++)
    {
        cout<<i;
    }
}
  
```

### Комментарий к примеру

1. Внутри цикла объявляется переменная `i` равная 1. Это и будет управляющая переменная.
2. Затем, осуществляется проверка значения этой переменной с помощью условия `i<=5`;
3. Если условие истинно (а так будет, пока `i` не достигнет значения 6) выполняется показ значения `i` на экран

(cout<<i;) и изменение управляющей переменной i на 1 (i++). Затем, снова проверяется условие.

4. Если условие ложно (то есть значение i стало равно 6), то программа переходит на следующую строчку за закрывающейся фигурной скобкой цикла.

**Примечание:** Обратите внимания, что первый шаг — СОЗДАНИЕ И ИНИЦИАЛИЗАЦИЯ ПЕРЕМЕННОЙ — всегда выполняется только один раз.

### **Некоторые особенности синтаксиса for**

Несмотря на простоту работы оператора, он обладает некоторыми особенностями форм записи.

#### **Инициализация управляющей переменной**

1. Инициализация и создание переменной производится в цикле.

```
for(int x=1;x<=100;x++)
{
    cout<<x;
}
```

2. Создание переменной производится до цикла, а инициализация в цикле.

```
int x;
for (x=1;x<=100;x++)
{
    cout<<x;
}
```

3. Инициализация и создание переменной производится до цикла.

```
int x=1;
for (;x<=100;x++)
{
    cout<<x;
}
```

Все три примера являются абсолютно функционирующими и равновесными.

## Изменение управляющей переменной

Изменение управляющей переменной можно перенести внутрь тела цикла, как это происходит в while и do while.

```
for(int x=1;x<=100;)
{
    cout<<x;
    x++;
}
```

## Условие

Условие конструкции также можно пропустить, однако в этом случае оно будет считаться по умолчанию истинным. Таким образом, мы получаем постоянно истинное условие и, как следствие — ВЕЧНЫЙ ЦИКЛ.

```
for(int x=1;;x++)
{
    cout<<x;
}
```

**Примечание:** Если хотите узнать, как пропустить условие и избежать вечного цикла — читайте следующий раздел урока.

Исходя из вышеописанного, мы можем сделать следующий вывод: ***Ни одна из частей цикла for не является обязательной.***

Как видите, работа for проста и аналогична работе while. Что выбрать?! Это зависит от поставленной задачи и от вашего решения.

## Оператор break

Нередко при работе с циклами, возникает необходимость искусственно прервать выполнение цикла. Для этого используется, уже знакомый вам (по изучению switch), оператор break. Этот оператор должен находиться в теле цикла, в том месте где необходимо сделать остановку. Например, именно с помощью этого оператора, мы можем решить проблему вечного цикла, в ситуации, когда условие в цикле for не указывается. Рассмотрим пример:

```
#include <iostream>
using namespace std;

void main()
{
    for(int x=1;;x++)
    {
        if(x==4) break; // если x стал равен 4 -
                        // остановить цикл

        cout<<x;

    }
    cout<<"Bye!";
}
```

## Комментарии к примеру

1. Согласно правилу, условие цикла всегда истинно, так как его просто нет.
2. При значениях 1, 2 и 3 переменной `x` условие оператора `if` выполняться не будет. `break`, естественно не сработает, так как находится в теле `if`. Между тем, на экран последовательно будут выводиться числа 1, 2, 3.
3. Когда `x` станет равно 4, программа попадет в тело `if` и выполнится `break`. Цикл сразу же будет остановлен, а выполнение программы перейдет на следующую строчку за закрывающейся фигурной скобкой оператора `for`.
4. На экране появится надпись Bye!
5. Цифра 4 на экране никогда не появится, так как, если сработал `break`, все что находится в цикле ниже него уже не выполнится.

**Примечание:** *`break` может быть использован либо в цикле, либо в операторе `switch`. Любое другое размещение приводит к ошибке на этапе компиляции.*

## Оператор `continue`

Оператор `continue` используется для прерывания текущей итерации цикла и осуществления перехода на следующий шаг. В ряде случаев, такие действия являются необходимыми. Если выполняется оператор `continue`, то в зависимости от вида цикла происходит следующее:

Циклы `while` и `do while` останавливают выполнение шага и переходят к проверке условия.



Цикл `for` также останавливает выполнение шага. Но, сначала переходит к изменению управляющей переменной, а потом уже к проверке условия.

Рассмотрим пример: показать на экран все нечетные целые числа, в диапазоне от нуля до 25 включительно. Название проекта `Odd`.

```
#include <iostream>
using namespace std;

void main()
{
    for(int i=0;i<26;i++)
    {
        if(i%2==0)    // если число делится на два без
                     // остатка
        {
            continue; // остановить итерацию цикла
                     // и перейти к i++
        }
        cout<<i<<»\n»;
    }
}
```

### **Комментарии к примеру**

1. Цикл начинает свое движение с нуля и проходит итерации до 25 включительно.
2. Внутри цикла предусмотрено условие: если число `i` — четное, нужно остановить текущий шаг цикла (`continue`;) и перейти к конструкции `i++`.
3. То, что располагается ниже сработавшего оператора `continue` на текущем шаге уже не выполнится.

4. Если условие `if` не выполняется, значит число `i` нечетное, `if` будет проигнорирован, а число — отображено на экран.

Теперь, когда мы познакомились с теоретическими материалами урока, давайте перейдем к следующему разделу, где будет рассмотрено несколько практических задач.

## 2. Практические примеры

### Пример 1

#### Постановка задачи

Часы бьют каждый час, столько раз, сколько времени. Написать программу, которая подсчитает, сколько раз пробьют часы за 12 часов. Название проекта Time.

#### Код реализации

```
#include <iostream>
using namespace std;
void main(){
    int sum=0;
    for(int bom=1; bom<=12;bom++){
        sum+=bom; // накопление суммы ударов
    }
    // Часы проббили 78 раз.
    cout<<> Hours have punched <<<sum<<>times.\n\n»;
}
```

#### Комментарии к коду

1. Изначально объявляется переменная `sum` равная нулю.
2. Цикл формируется из трех конструкций `int bom=1;` — начальная инициализация, `bom<=12;` — условие, `bom++` — изменение управляющей переменной.
3. Внутри тела цикла накапливается сумма ударов путем прибавления управляющей переменной к значению общей суммы.
4. Когда `i` достигнет значения 13, цикл остановится и на экран покажется результат.

## Пример 2

### Постановка задачи

Пользователь с клавиатуры последовательно вводит целые числа. Как только пользователь ввел 0, необходимо показать на экран сумму всех введенных чисел. Название проекта Amount.

### Код реализации

```
#include <iostream>
using namespace std;
void main(){
    int digit, sum=0;
    for(;;){ // реализация бесконечного цикла
        cout<<»Enter digit:»»;
        cin>>digit; // ввод числа
        if(digit==0) // если введен 0
            break; // остановить цикл
        sum+=digit; // накопление суммы
    }
    // показ результата
    cout<<»Sum of digits»<<sum<<»\n\n»»;
}
```

### Комментарии к коду

1. В программе реализован условно бесконечный цикл. То есть остановка цикла происходит искусственным путем (break).
2. На каждой итерации пользователь вводит число.
3. Осуществляется проверка, если это число — 0, значит пора остановить цикл, если не 0, необходимо прибавить число к общей сумме.

- После того, как отработает `break` и цикл прекратит работу, на экран покажется сумма всех введенных с клавиатуры чисел.

## Пример 3

### Постановка задачи

Написать программу, которая показывает все числа, которым кратно число, введенное с клавиатуры. Название проекта `Number`.

### Код реализации

```
#include <iostream>
using namespace std;

void main(){
    int digit;
    cout<<»Enter digit:»;
    cin>>digit;

    // цикл перебирает числа от 2 до введенного числа
    for(int i=2;i<digit;i++){

        // если число не делится на текущее
        // значение i без остатка остановить
        // данный шаг и перейти к
        // следующему
        if(digit%i!=0)
            continue;
        // показать i на экран
        cout<<i<<»\n»;
    }
}
```

### **Комментарии к коду**

1. Пользователь вводит число для анализа.
2. Цикл последовательно перебирает все числа от 2 до исходного.
3. Осуществляется проверка: если искомое число на текущее без остатка не делится, необходимо прервать данный шаг цикла и перейти к части `i++`. (`continue`).
4. Если искомое число на текущее без остатка делится, то на экран показывается текущее число.

## 3. Домашнее задание

Во втором уроке, вы уже научились разбивать число на цифры. Сегодняшнее домашнее задание основано именно на этом принципе, однако, вам придется использовать еще и циклы.

1. Пользователь вводит с клавиатуры число — программа должна показать сколько в данном числе цифр. Число вводится целиком в одну переменную.

**Примечание:** Например, пользователь ввел число 12345. На экране должно появиться сообщение о том, что в числе 5 цифр.

2. Пользователь вводит с клавиатуры число, необходимо перевернуть его (число) и вывести на экран.

**Примечание:** Например, пользователь ввел число 12345. На экране должно появиться число наоборот — 54321.

3. Пользователь вводит с клавиатуры число, необходимо показать на экран сумму его цифр.

**Примечание:** Например, пользователь ввел число 12345. На экране должно появиться сообщение о том, что сумма цифр числа 15.



## Урок №4 Цикл for

© Компьютерная Академия «Шаг»  
[www.itstep.org](http://www.itstep.org)

Все права на охраняемые авторским правом фото-, аудио- и видеопроизведения, фрагменты которых использованы в материале, принадлежат их законным владельцам. Фрагменты произведений используются в иллюстративных целях в объеме, оправданном поставленной задачей, в рамках учебного процесса и в учебных целях, в соответствии со ст. 1274 ч. 4 ГК РФ и ст. 21 и 23 Закона Украины «Про авторське право і суміжні права». Объем и способ цитируемых произведений соответствует принятым нормам, не наносит ущерба нормальному использованию объектов авторского права и не ущемляет законные интересы автора и правообладателей. Цитируемые фрагменты произведений на момент использования не могут быть заменены альтернативными, не охраняемыми авторским правом аналогами, и как таковые соответствуют критериям добросовестного использования и честного использования.

Все права защищены. Полное или частичное копирование материалов запрещено. Согласование использования произведений или их фрагментов производится с авторами и правообладателями. Согласованное использование материалов возможно только при указании источника.

Ответственность за несанкционированное копирование и коммерческое использование материалов определяется действующим законодательством Украины.