STEP computer ACADEMY

PROGRAMMING C

Lesson No. 4

Programming

C

Contents

1. Construction for

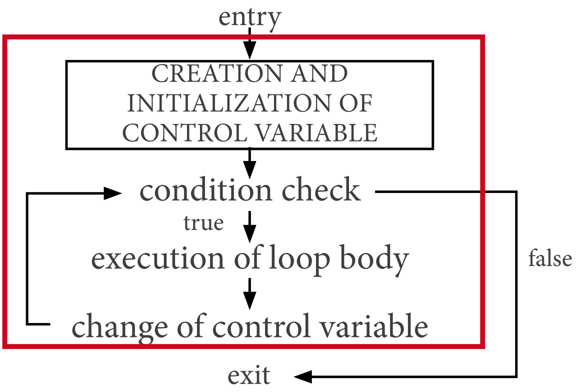
In the previous lesson we have acquainted with such a concept as loop and have studied several constructions representing loops in language C. In particular — while and do while. Now we are going to review one more loop type — for operator. Theoretically this operator is a complete analogy to «while», and practically it allows arranging the loop with a more convenient control.

General syntax and principle of operation of construction for

```
for(variable initialization; condition check; change of variable)
{
               action;
}
```

Principle of loop execution:

```
1. Variable initialization.
2. Check of a condition.
3. Action execution if a condition is true.
4. If a condition is false next to the loop operator is executed.
5. If a condition was true - change of control variable.
6. Check of a condition. And then step 3 or 4 again.
```


entry
CREATION AND INITIALIZATION OF CONTROL VARIABLE
condition check
true
execution of loop body
change of control variable
false
exit

Use case

Let’s review simple and familiar example: show the numbers from 1 to 5 included using a loop. But we will do it using for operator.

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

Comment to the example

1. A variable equal to 1 is declared within the loop. It will be a control variable.

2. Afterwards, this variable value is checked by means of condition i<=5;

3. If condition is true (and it will continue until i reaches value 6) value i will be displayed on a screen (cout<<i;) and change of control variable i by 1 (i++). Afterwards, the condition is checked.

4. If condition is false (meaning i value became equal to 6), the program switches to the next string behind a closing curly brace of the loop.

Note: pay attention that the first step is — CREATION AND INITIALIZATION OF VARIABLE — is always executed once.

Some features of for syntax

In spite of simplicity of operator functioning, it possesses some features of record forms.

Initializing control variable

1. Initialization and creation of a variable takes place within the loop.

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

2. Creating variable takes place before the loop and initializing happens within the loop.

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

3. Initialization and creation of a variable takes place before the loop.

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

All three examples are absolutely equally functional.

Change of control variable.

Change of control variable can be moved inside the loop body as it happens inside while and do while.

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

Condition.

Condition of a construction can also be omitted, though in this case it will be regarded a default true. Therefore, we obtain a constantly true condition and as a consequence an ETERNAL LOOP.

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

Note: if you want to know how to skip the condition and avoid an eternal loop — read the next lesson unit.

Proceeding from the above we can make a conclusion: **neither of loop for parts is obligatory.**

As you can see operation of for is simple and analogical to while operation. Then what to choose? It depends on the assigned task and your decision.

Break operator

Quite often when working with the loops there arises a necessity to force the termination of loop execution. With this purpose we use already familiar (in our studies - switch), break operator. This operator should be within the loop body in the place where it is necessary to make a stop. For example, using this operator we can solve the problem of the eternal loop in a case when a condition in for loop is not indicated. Let's review an example:

```
#include <iostream>
using namespace std;
void main()
{
              for(int x=1;;x++)
              {
                  if(x==4) break;// if x became equal to 4 - stop the loop
                  cout<<x;

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

Comments to the example

1. Following the rule a loop condition is always true, because it simply does not exist.

2. Upon the values 1,2 and 3 of x variable a condition of if operator will not be executed. Break of course will not be actuated because it is situated within the if body. Meanwhile numbers 1, 2, 3 will be displayed on a screen.

3. When x becomes equal to 4 a program will get within the if body and the break will be executed. A loop will be terminated at once and program execution will be shifted to the next string behind a closing curly brace of for operator.

4. A sign Bye will be displayed!

5. There will never be a number 4 on a screen because if break was actuated, then everything within the loop below 4 will never be executed.

Note: break can be used either within a loop, or within the switch operator. Any other positioning leads to an error at compile stage.

Continue operator

Continue operator is used for termination of the running loop iteration and exercising a transition to the next step. In a range of cases such actions are necessary. If continue operator is executed then depending on the loop type the following happens:

Cycles while and do while stop execution of the step and shift to condition check.

For loop also stops a step execution. But at first it shifts to changing a control element and then to condition check.

Review an example: to display on a screen all uneven integers in a range from zero to 25 included. Project name is Odd.

```
#include <iostream>
using namespace std;
void main()
{
        for(int i=0;i<26;i++)

        {
          if(i%2==0)// if number can be divided by two without excess
          {
                      continue;// to stop loop iteration and move to i++
          }
          cout<<i<<"\n";

        }
}
```

Comments to the example

1. . Loop starts its moving from zero and passes through iterations till 25 included.

2. Inside a cycle there is a condition provided: if number i is even we have to stop a current step of the loop (continue;) and shift to the construction i++.

3. Everything is below an actuated continue operator at the current step will not be executed.

4. If condition if is not executed a number i is uneven, if will be ignored and the number displayed on a screen.

Now when we got acquainted with theoretical materials of the lesson, let's proceed to the next unit where we will review a few practical tasks.

2. Use cases

Example 1

Problem statement

Clock strike every hour, number of strikes is equal to the hour number. Write a program that calculates how many times the clock will strike during 12 hours. Project name is Time.

Realization code

```
#include <iostream>
using namespace std;
void main(){
      int sum=0;
      for(int bom=1; bom<=12;bom++){
        sum+=bom;// accumulation of strike sum
      }

      // Clock strikes 78 times.
      cout<<" Hours have punched "<<sum<<"times.\n\n";
}
```

Comments to the code

1. At the beginning a sum equal to zero is declared.

2. The loop forms out of three constructions int bom=1; — entry initialization, bom<=12; — condition, bom++ — change of control variable.

3. Inside the loop body a sum of strikes accumulates by means of adding a control variable and value of general sum.

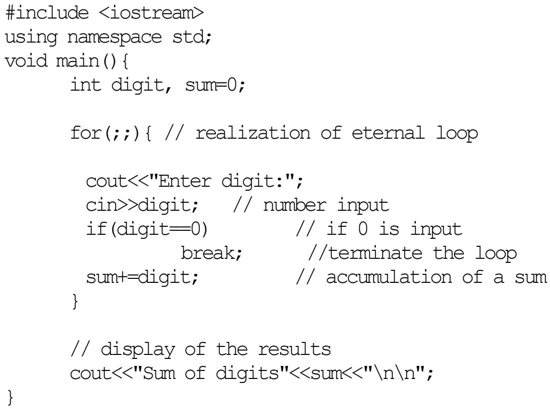
4. When i reaches a value 13 the loop terminates its operation and the result will be displayed.

Example 2

Problem statement:

User successively inputs integer numbers from a keyboard. As soon as user inputs 0, it is necessary to display a sum of all input numbers. Project name is Amount.

Realization code



```
#include <iostream>
using namespace std;
void main(){
      int digit, sum=0;

      for(;;){ // realization of eternal loop

        cout<<"Enter digit:";
        cin>>digit;   // number input
        if(digit==0)        // if 0 is input
                break;       //terminate the loop
        sum+=digit;          // accumulation of a sum
      }

      // display of the results
      cout<<"Sum of digits"<<sum<<"\n\n";
}
```

Comments to the code

1. An eternal loop is realized within a program. Meaning that loop stop is forceful (break).

2. Within each iteration a user inputs a number.

3. A check runs and if this number is 0, we should stop the loop, if not 0, and then we have to add a number to the general sum.

4. After that when break finishes off and the loop will finish its operation, then there will be on a screen a sum of all введенных с клавиатуры чисел.

Example 3.

Problem statement.

Write a program, which shows all numbers to which an input number is multiple of. Project name is Number.

Realization code

```
#include <iostream>
using namespace std;
void main(){
        int digit;
        cout<<"Enter digit:";
        cin>>digit;

        // a loop searches the numbers from 2 till an input one
        for(int i=2;i<digit;i++){

        /// if a number cannot be divided by the current one
        // value i without an excess, then it is necessary to terminate
        // this step and move to
        // the next one
        if(digit%i!=0)
                  continue;

        // display i on a screen
        cout<<i<<"\n";
        }
}
```

Comments to the code

1. User inputs a number for analysis.

2. Loop successively searches all numbers from 2 till initial one.

3. Check: if search number cannot be divided by the current one without the excess, it is necessary to terminate this loop step and move to the part i++. (continue).

4. If a search number cannot be divided by the current one without an excess, then the current number is displayed on a screen.

3. Home assignment

In the second lesson you have already learned how to split a number into digits. Current home assignment is based on this very principle, though you will have to use also loops.

1. User input a number from a keyboard and program should show how many digits there are in a number. A number is input totally into one variable.

Note: for example user input a number 12345. There should a message appear on a screen notifying that there are 5 digits within a number.

2. User inputs a number from a keyboard, we need to turn it (number) and display on a screen.

Note: for example, user inputs a number 12345. There should be an opposite number displayed on a screen like — 54321.

3. User inputs from a keyboard, and we should display on a screen a sum of its digits.

Note: For example, user inputs a number 12345. There should a message appear on a screen notifying that a sum of its digits is 15.