Lab # 8: Loops

EC-102 – Computer Systems and Programming

Usama Wajhi

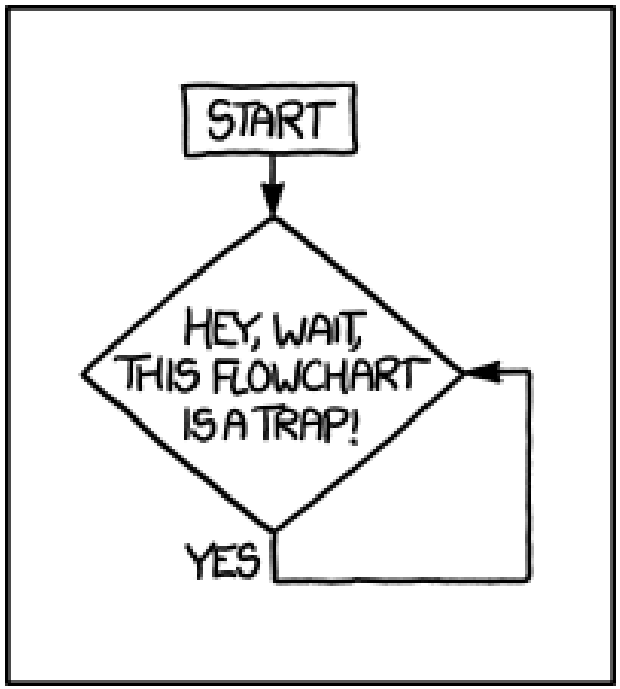
School of Mechanical and Manufacturing Engineering (SMME),
National University of Sciences and Technology (NUST)

December 15, 2016

Outline I

Introduction to Loops

- Cause a section of your program to be repeated a certain number of times
- The repetition continues while a condition is true
- As soon as the condition becomes false, the loop ends and passes the control to the statements following the loop



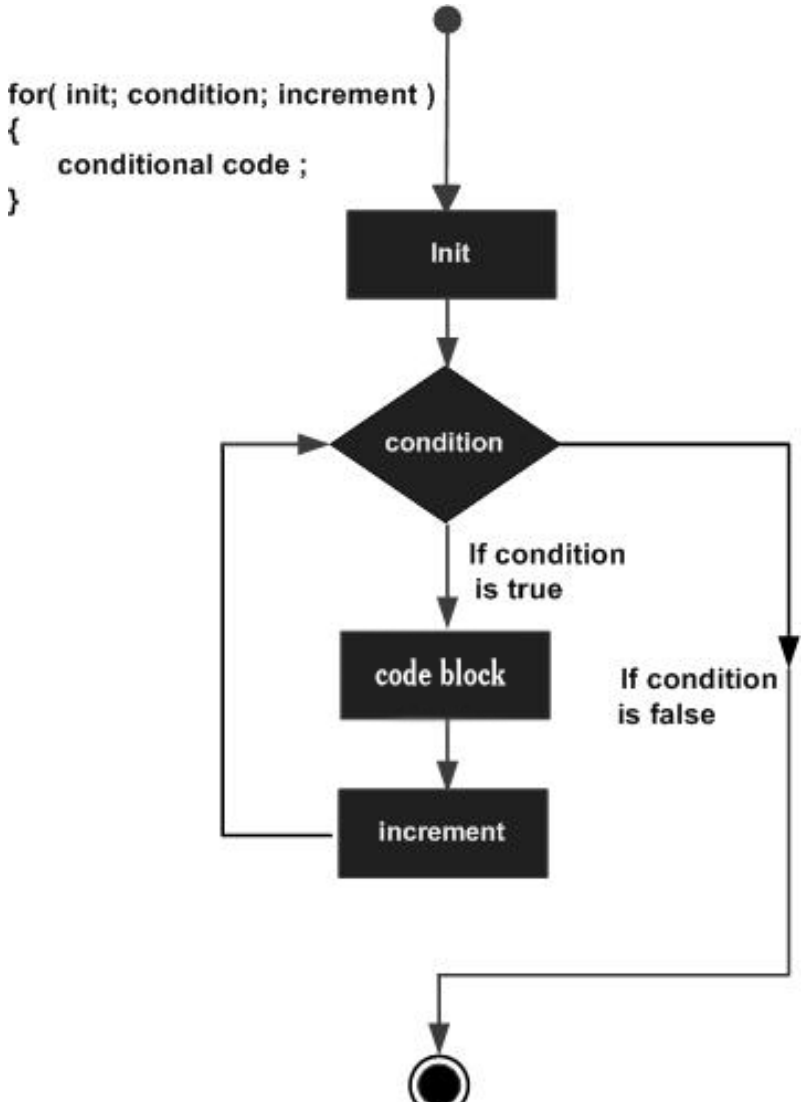
Loops in C++

There are three types of loops in C++:

- the `for` loop,
- the `while` loop, and
- the `do` loop

The `for` Loop

- Easiest to understand because all its loop control elements are gathered in one place
- Executes a section of a code a fixed number of times



The `for` Loop – Syntax

```
1 for(init; test;
      update)
2     {
3         statement;
4         statement;
5         statement;
6         statement;
7     }
```

- Keyword `for` followed by parantheses that contain three expressions separated by semicolons
  1. the initialization expression,
  2. the test expression, and
  3. the update expression
- These three expressions usually involve the same variable, also known as the loop variable
- The body of the loop, delimited by the left and right braces, is the code to be executed each time through the loop

The for Loop – Solved Example 1

```
1  // demonstrates simple FOR loop
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7      int j;
8      for(j = 0; j < 15; j++)
9      {
10         cout << j * j << endl;
11     }
12     return 0;
13 }
```

The for Loop – Solved Example 2

```
1  // lists cubes from 1 to 10
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5
6  int main()
7  {
8      int num;
9
10     for(num = 1; num <= 10; num++)
11     {
12         cout << setw(4) << num;
13         int cube = num * num * num;
14         cout << setw(6) << cube << endl;
15     }
16     return 0;
17 }
```

Variation in `for` Loop

- Initialization can also be performed before loop expression

```
int i = 1;
for(; i <= 5; i++)
```

- Update expression can also be placed within a loop body

```
int i = 1;
for(; i <= 5;)
{
    cout << "i = " << i << endl;
    i++;
}
```

- If test expression is omitted, then the loop will run forever

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

The break Statement

- Immediate exit from the loop
- Program continues with first statement after the loop block
- Used to escape early from a loop

```
for (x = 1; x <= 10; x++)
    {
        if (x == 5)
            break;
        cout << x << endl;
    }
    cout << "\n Out when x became " << x;
```

The continue Statement

- Skips remainder of the loop body
- Proceeds with the next iteration of loop

```
1 for (x = 1; x <= 10; x++)
2     {
3         if (x == 5)
4             continue;
5         cout << x << " ";
6       }
7     cout << "\n Skipped value 5" <<endl;
```

Home Tasks

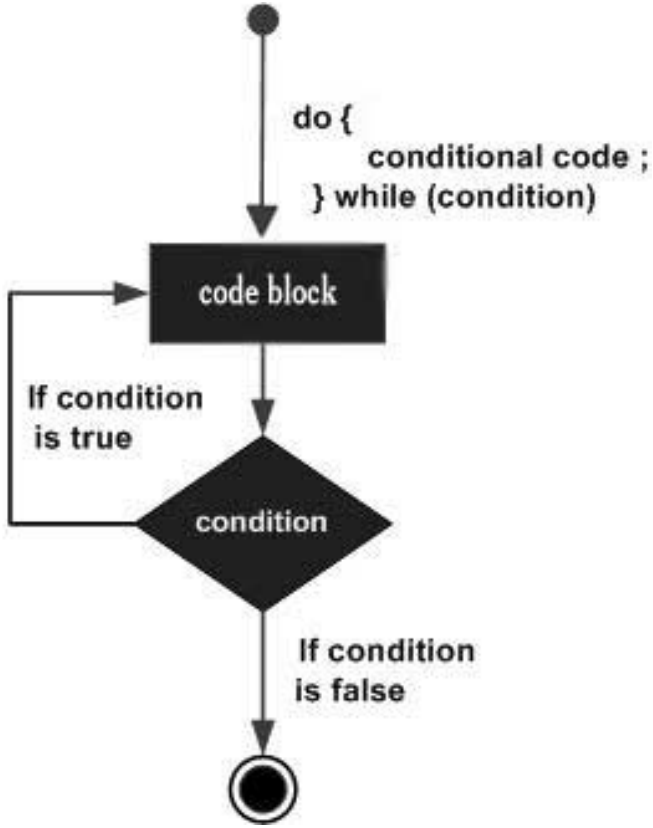
- Write a program using `for` loop which displays the following shape

```
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

- Write a program using `for` loop which displays all the even numbers from a minimum number entered by the user to a maximum number entered by the user

The do Loop

- In some situations, you want the test expression to be evaluated at the beginning of the loop
- But sometimes you want to guarantee that the loop is executed atleast once, no matter what the initial state of the test expression
- When such is the case, do loop should be used



The do Loop – Syntax

```
1 do
2     {
3         statement;
4         statement;
5         statement;
6         statement;
7     }
8 while(test);
```

- Keyword do marks the beginning of the loop
- As with other loops, braces delimit the body of the loop
- Finally, a `while` statement provides the test expression and terminates the loop

The do Loop – Solved Example

```
1  int main()
2  {
3      int dividend, divisor;
4      char ch;
5      do{
6              cout << "Enter dividend: ";
7              cin >> dividend;
8              cout << "Enter divisor: ";
9              cin >> divisor;
10             cout << "Quotient is " << dividend /
       divisor;
11             cout << ", remainder is " << dividend %
       divisor;
12             cout << "\nDo another? (y/n): ";
13             cin >> ch;
14         }
15     while(ch != 'n');
16 return 0;
17 }
```

Exercise

Write a program using do loop that repeatedly asks for a number and calculates its factorial until the user enters 0, at which point it terminates.

Here goes a sample interaction with the program:

```
Enter a number: 7
The factorial of the number is: 5040
Enter a number: 6
The factorial of the number is: 720
Enter a number: 5
The factorial of the number is: 120
Enter a number: 0
The factorial of the number is: 1
```