Introduction to C++ Programming: Lecture 3

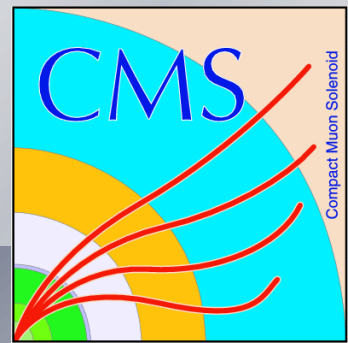
Presented by

DR. MOHAMMED ATTIA MAHMOUD

-PhD, Fayoum University, Egypt and Antwerp University, Belgium.

-Researcher in ENHEP, ASRT, Fayoum Uni, and BUE.

-FSQ Gen-Contact, CMS experiment, CERN, Geneva, Switzerland.



Conditional ternary operator (?)

The conditional operator evaluates an expression, returning one value if that expression evaluates to true, and a different one if the expression evaluates as false. Its syntax is:

condition ? result1 : result2

If condition is true, the entire expression evaluates to result1, and otherwise to result2.

```
// conditional operator
#include <iostream>
using namespace std;
int main ()
{
int a,b,c;
 a=2;
 b=7;
 c = (a>b) ? a : b;
cout << c << '\n';
}
```

7

Selection statements: if and else

The if keyword is used to execute a statement or block, if, and only if, a condition is fulfilled. Its syntax is:

if (condition) statement;

```
1  if (x == 100)
2   cout << "x is 100";
```

```
if (x == 100) { cout << "x is "; cout << x; }
```

If x is not exactly 100, this statement is ignored, and nothing is printed.

if (condition) statement1
else statement2;

```
1  if (x == 100)
2   cout << "x is 100";
3  else
4   cout << "x is not 100";
```

Selection statements: if and else

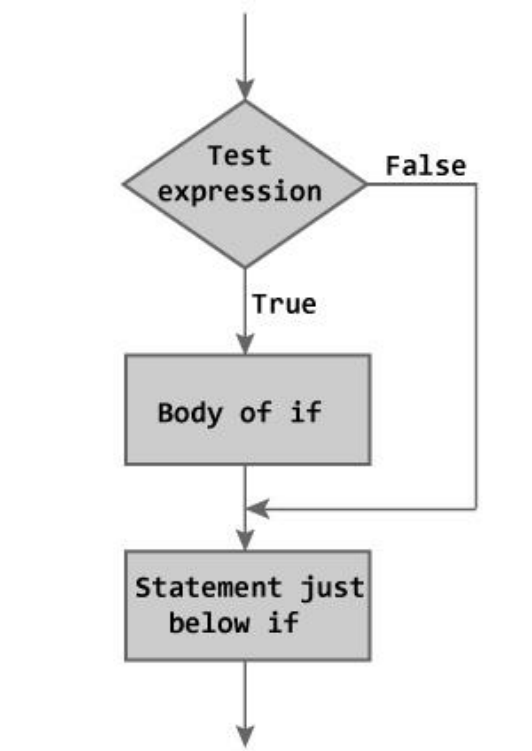


Figure: Flowchart of if Statement

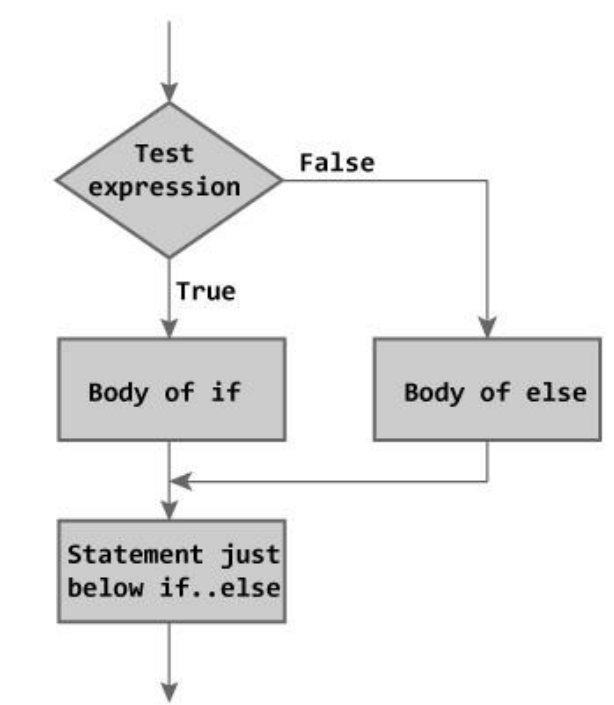


Figure: Flowchart of if...else Statement

```
1  if (x > 0)
2   cout << "x is positive";
3   else if (x < 0)
4  cout << "x is negative";
5   else
6   cout << "x is 0";
```

Simples examples

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

int main () {
   // local variable declaration:
   int a = 100;

   // check the boolean condition
   if( a == 10 ) {
      // if condition is true then print the following
      cout << "Value of a is 10" << endl;
   } else if( a == 20 ) {
      // if else if condition is true
      cout << "Value of a is 20" << endl;
   } else if( a == 30 ) {
      // if else if condition is true
      cout << "Value of a is 30" << endl;
   } else {
      // if none of the conditions is true
      cout << "Value of a is not matching" << endl;
   }
   cout << "Exact value of a is : " << a << endl;

   return 0;
}
```

Iteration statements (loops)

Loops repeat a statement a certain number of times, or while a condition is fulfilled.

They are introduced by the keywords ***while, do, and for.***

The while loop

The simplest kind of loop is the while-loop. Its syntax is:

while (expression) statement

The while-loop simply repeats statement while expression is true. If, after any execution of statement, expression is no longer true, the loop ends, and the program continues right after the loop. For example, let's have a look at a countdown using a while-loop:

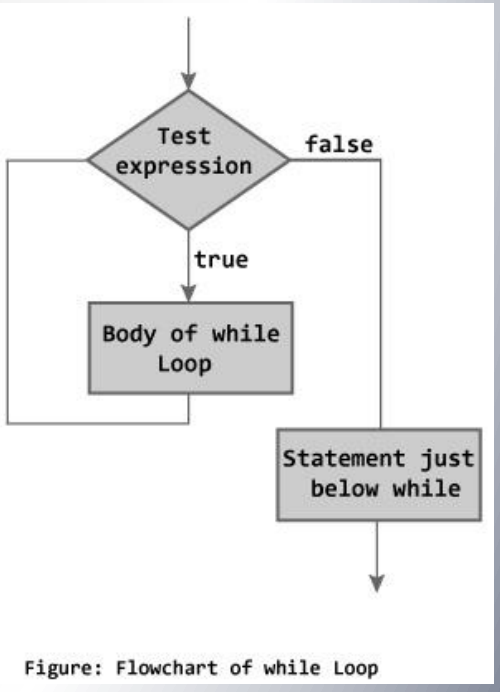


Figure: Flowchart of while Loop

```
1  // custom countdown using while
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int n = 10;
8
9    while (n>0) {
10     cout << n << ", ";
11     --n;
12   }
13
14   cout << "liftoff!\n";
15 }
```

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, liftoff!
```

(prints the value of n and decreases n by 1)

do-while loop

The do-while loop is also called exit control loop because, in do-while loop, compiler will 1st execute the statements, then check the condition, whether it is true or false.

```cpp
// C++ program to add numbers until user enters 0

#include <iostream>
using namespace std;

int main()
{
    float number, sum = 0.0;

    do {
        cout<<"Enter a number: ";
        cin>>number;
        sum += number;
    }
    while(number != 0.0);

    cout<<"Total sum = "<<sum;

    return 0;
}
```

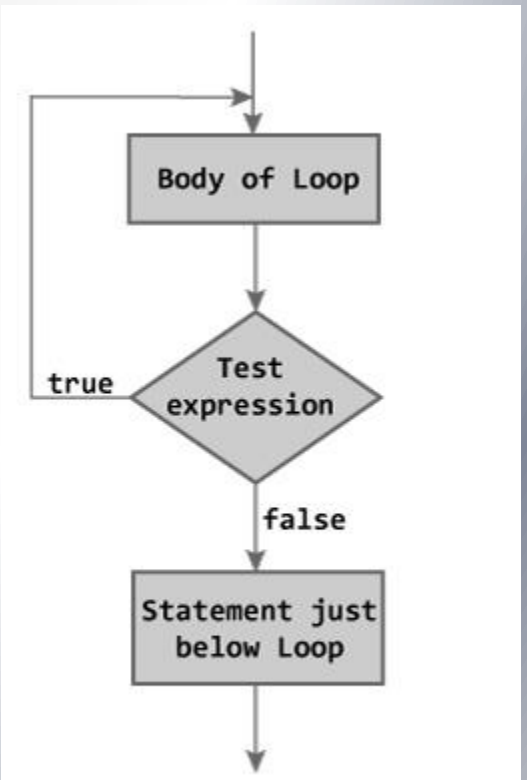


Figure: Flowchart of do...while Loop

The for loop

The for loop is designed to iterate a number of times. Its syntax is:

for (initialization; condition; increase) statement;

1.initialization is executed. Generally, this declares a counter variable, and sets it to some initial value. This is executed a single time, at the beginning of the loop.

2.condition is checked. If it is true, the loop continues; otherwise, the loop ends, and statement is skipped, going directly to step 3.

3.statement is executed. As usual, it can be either a single statement or a block enclosed in curly braces { }.

4.increase is executed, and the loop gets back to step 2.

5.the loop ends: execution continues by the next statement after it.

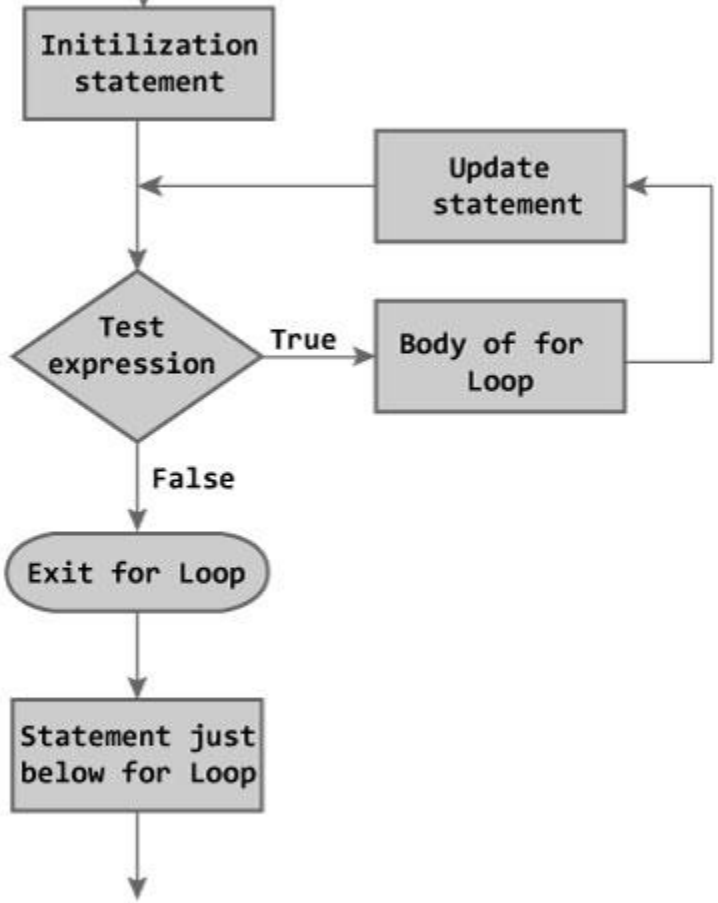


Figure: Flowchart of for Loop

Examples

```
1  // countdown using a for loop
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    for (int n=10; n>0; n--) {
8      cout << n << ", ";
9    }
10   cout << "liftoff!\n";
11 }
```

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, liftoff!
```

```
int main()
{
   int i=0,j;

   cout << i << endl;
   for (i = 1; i < 5; i++)
   {
        cout << "loop in :: "  << i << endl;

        for (j = i; j < 3; j++)
        {
         cout << "\t inner for loop j :: " << j
              <<  endl;
        }
        cout << "loop end :: " << i << endl;
   }
   cout << i << endl;
   return 0;
}
```


Thanks!