C++ Programming Language

Presented By:

T.A. Asmaa Hamad El-saied

E-mail: eng.asmaa134@gmail.com

Contents

1. Iteration statements (loops)
   - ✓ While…Statement
   - ✓ Do-while...Statement
   - ✓ For…Statement
2. Loop Control Statements

Repetition/Iteration

- **A loop** is a control structure that causes a statement or group of statement to repeat
- **C++** has three looping control structures:
  - The while loop
  - The do-while loop and
  - The for loop
- the difference between each of these is how they control the repetition

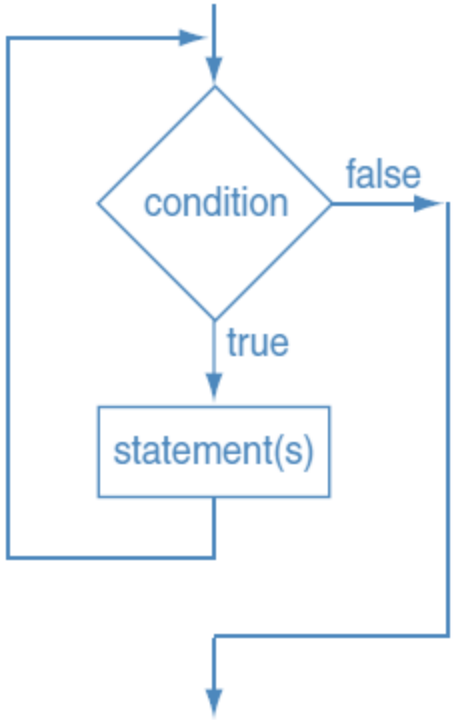
While..Statement

While Loop

- The **while** loop has two important parts:
  (1) an expression that is tested for a true or false value, and
  (2) a statement or block that is repeated as long as the expression is true.

The syntax of a while loop in C++ is:

```
while (expression)
{
Statement(s);
}
```



While Loop(cont.)

- **Example 1**

```
#include <iostream>
using namespace std;
int main()
{      int number = 0;
       cout << "This program will let you enter number after\n";
       cout << "number. Enter 99 when you want to quit the ";
       cout << "program.\n";
         while (number != 99)
                cin >> number;
       cout << "Done\n";
       return 0;
}
```

While Loop(cont.)

- **Output**

  This program will let you enter number after number. Enter 99 when you want to quit the program.

  1

  2

  30

  75

  99

  - Done

- **Note** :while Is a Pretest Loop

While Loop(cont.)

- **Example 2**

```
#include <iostream>
using namespace std;
int main()
{     int num = 1; // Initialize counter
      cout << "Number      Number Squared\n";
      cout << "-----------------------\n";
while (num <= 10)
 {
      cout << num << "\t\t" << (num * num) << endl;
      num++;     // Increment counter
   }
return 0;
}
```

While Loop(cont.)

- **Outpute**

```
Number      Number Squared
-----------------------
  1              1
  2              4
  3              9
  4             16
  5             25
  6             36
  7             49
  8             64
  9             81
 10            100
```

While Loop(cont.)

Example 3: A C++ program to calculate the summation of 10 numbers**.**

```
#include <iostream>
using namespace std;
int main()
{    int Sum =0;
     int Count =0;
 While(count<10)
 {
        Cout<<“Enter an integer:”;
        Cin>>x;
        Sum=sum+x;
        Count=count+1;
 }
Cout<<“Summation =“<<sum;
Return 0;}
```

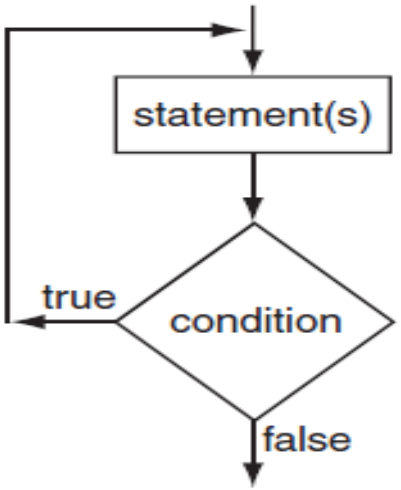
DO-While Statement

do-while Loop

- Unlike **for** and **while** loops, which test the loop condition at the top of the loop, the **do...while** loop checks its condition at the bottom of the loop.

- A **do...while** loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time.
- The syntax of a do...while loop in C++ is:

```
do
   {
      statement(s);
   }while( condition );
```



do-while Loop(cont.)

- The do-while loop must be terminated with a semicolon after the closing parenthesis of the test expression.
- the difference between the do-while loop and the while loop is that do-while is a *post test* loop.
- It tests its expression after each iteration is complete. This means do-while always performs at least one iteration, even if the test expression is false from the start

do-while Loop(cont.)

- For example, in the following while loop the cout statement will not execute at all:

```
int x = 1;
while (x < 0)
    cout << x << endl;
```

- But the cout statement in the following do-while loop will execute once because the do-while loop does not evaluate the expression x < 0 until the end of the iteration.

```
int x = 1;
 do
     cout << x << endl;
while (x < 0);
```

do-while Loop(cont.)

Example 1

```
#include <iostream>
 using namespace std;
 int main ()
 {int a = 10;
   do
   {
    cout << "value of a: " << a
    << endl;
     a = a + 1;
   }while( a < 20 );
 return 0; }
```

Output

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

do-while Loop(cont.)

- **Example 2:**This program averages 3 test scores. It repeats as many times as the user wishes

```
#include <iostream>
using namespace std;
int main()
{ int score1, score2, score3;double average; char again;
  do {  cout << "Enter 3 scores and I will average them: ";
        cin >> score1 >> score2 >> score3;
        average = (score1 + score2 + score3) / 3.0;
        cout << "The average is " << average << ".\n";
       cout << "Do you want to average another set? (Y/N) ";
       cin >> again;
 } while (again == 'Y' || again == 'y');
 return 0; }
```

do-while Loop(cont.)

- **Output**
  - Enter 3 scores and I will average them: 80 90 70
  - The average is 80.
  - Do you want to average another set? (Y/N) y
  - Enter 3 scores and I will average them: 60 75 88
  - The average is 74.333336.
  - Do you want to average another set? (Y/N) n

For.. Statement

For Loop

- A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
- The syntax of a for loop in C++ is:

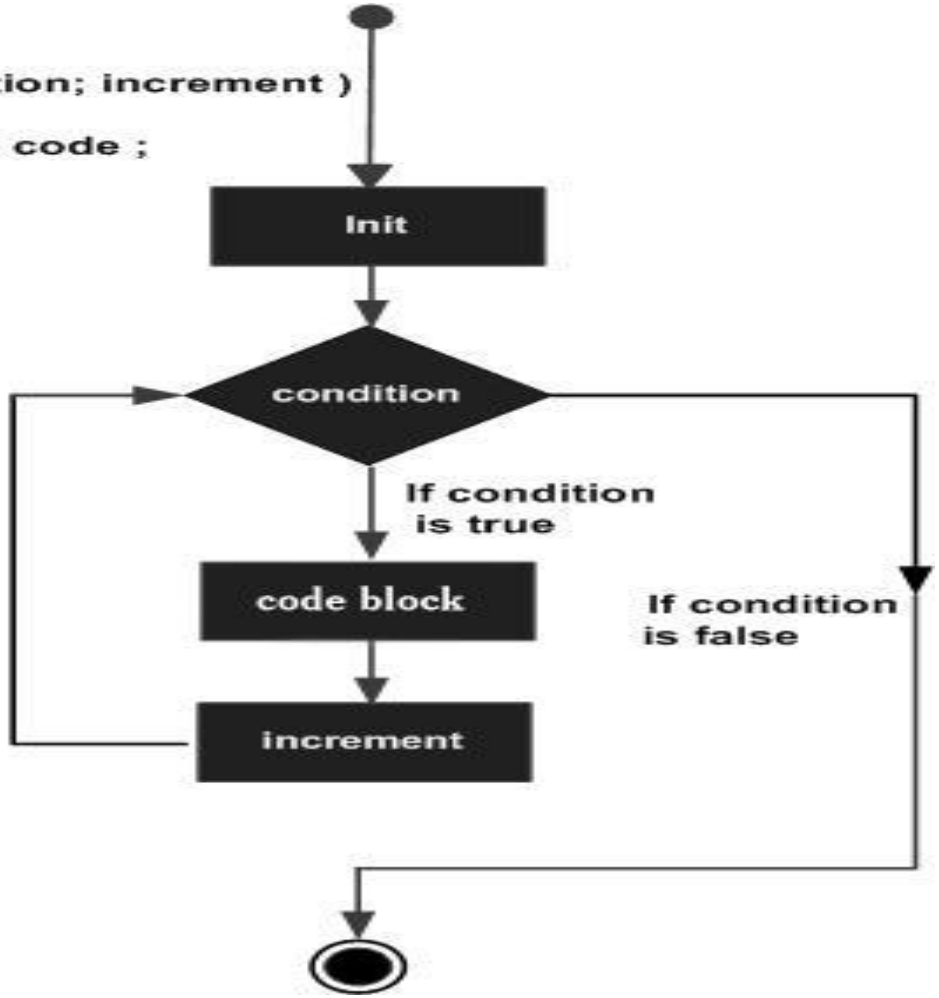
```
for ( init; condition; increment )
        {
           statement(s);
        }
```

For Loop(cont.)

- **Here is the flow of control in a for loop:**
  - The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
  - Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the for loop.
  - After the body of the for loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.
  - The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the for loop terminates.

For Loop(cont.)

```
for( init; condition; increment )
{
    conditional code ;
}
```



For Loop(cont.)

Example 1:

```
#include <iostream>
using namespace std;
int main ()
{
 for( int a = 10; a < 20; a = a + 1 )
{
 cout << "value of a: " << a <<
endl;
}
return 0; }
```

Output

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 15

value of a: 16

value of a: 17

value of a: 18

value of a: 19

For Loop(cont.)

Example 2: A program calculates the average of 10 numbers.

```
#include <iostream>
using namespace std;
int main ()
{int x;
  int Sum=0;
    For (int i=1;i<=10;i++)
    {
       cout<<“Enter a number:”;
       cin>>x;
       sum=sum+x;
    }
   cout<<”Average = “<<sum/10;
   return 0;
 }
```

For Loop(cont.)

Example 4: A program calculates the summation of odd numbers from 1 to 100.

```
#include <iostream>
using namespace std;
int main ()
{
  int Sum=0;
   For (int i=1 ;i<= 100;i++)
     If((i%2)!=0)
         sum = sum + i;

  cout<<“Summation = “<<sum;
  return 0;
 }
```

For Loop(cont.)

Example 5: A program calculates the factorial of an integer number.

```
#include <iostream>
using namespace std;
int main ()
{int n, Fact;
 cout<<“Enter an integer: “;
   cin>>n;
     Fact = n;
    For (int i= n-1; i>= 2;i--)
              Fact = Fact *i;
    cout<<n<<“! = “<<Fact;
    return 0;
 }
```

Nested Loops

Nested Loops

- **A loop** can be nested inside of another loop. C++ allows at least 256 levels of nesting.
- The syntax for a **nested for loop** statement in C++ is as follows:

```
for ( init; condition; increment )
{
   for ( init; condition; increment )
         { statement(s); }
  statement(s); // you can put more statements.
}
```

Nested Loops(cont.)

- The syntax for a **nested while loop** statement in C++ is as follows:

```
while(condition)
 {
     while(condition)
     {
          statement(s);
     }
     statement(s); // you can put more statements.
}
```

Nested Loops(cont.)

- The syntax for a **nested do-while loop** statement in C++ is as follows:

```
do {
    statement(s); // you can put more statements.
    do {
        statement(s);
    }while( condition );
}while( condition );
```

Nested Loops(cont.)

Example : **the following program uses a nested for loop to find the prime numbers from 2 to 100:**

```
#include <iostream>
using namespace std;
int main ()
{    int i, j;
   for(i=2; i<100; i++)
   {
      for(j=2; j <= i; j++)
        {
         i f(i%j==0)
                  break;}
  if(i==j)      cout << i << " is prime\n“;
}
 return 0; }
```

Nested Loops(cont.)

- **Output**

2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
53 is prime
59 is prime
61 is prime
67 is prime
71 is prime
73 is prime
79 is prime
83 is prime
89 is prime
97 is prime

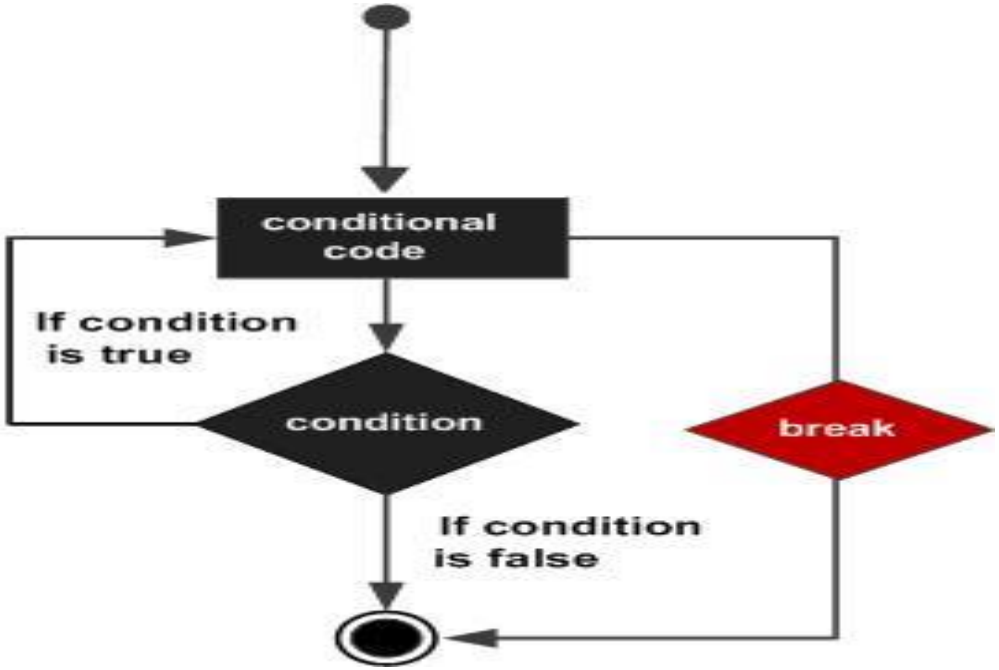
Control Statements

- The **break** statement has the following two usages in C++:
  - When the break statement is encountered inside a loop, the loop is immediately terminated and program control resumes at the next statement following the loop.
  - It can be used to terminate a case in the switch statement
- If you are using nested loops (i.e., one loop inside another loop), the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

Control Statements(cont.)

- The syntax of a break statement in C++ is:

```
break;
```



Control Statements(cont.)

- **Example:**

```
#include <iostream>
 using namespace std;
int main ()
{int a = 10;
        do {
             cout << "value of a: " << a << endl;
            a = a + 1;
            if( a > 15)
                {break; }
           }while( a < 20 );
return 0; }
```

Control Statements (cont.)

- **Output**
  - ✓value of a: 10
  - ✓value of a: 11
  - ✓value of a: 12
  - ✓ value of a: 13
  - ✓value of a: 14
  - ✓value of a: 15

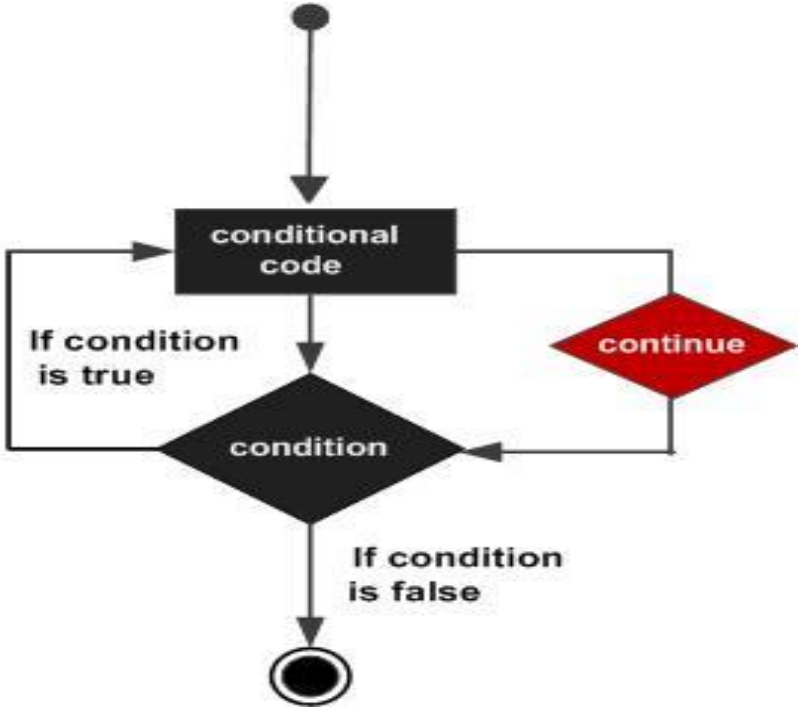
Control Statements(cont.)

- The **continue** statement works somewhat like the break statement. Instead of forcing termination, however, continue forces the next iteration of the loop to take place, skipping any code in between.
- For the **for** loop, continue causes the conditional test and increment portions of the loop to execute.
- For the **while** and **do...while** loops, program control passes to the conditional tests.

Control Statements(cont.)

- The syntax of a continue statement in C++ is:

```
continue;
```



Control Statements(cont.)

❖ **Example**:

```
#include <iostream>
using namespace std;
int main ()
{
    int a = 10;
    do {
        if( a == 15)
            { a = a + 1; continue; }
        cout << "value of a: " << a << endl;
        a = a + 1;
    } while( a < 20 );
    return 0;
}
```

Output

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Control Statements(cont.)

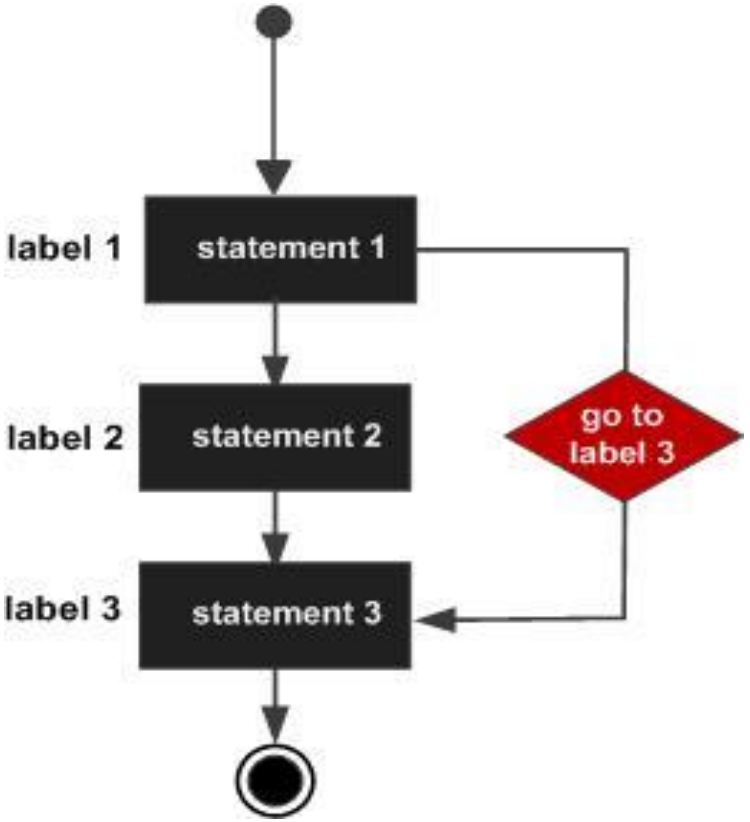
- A **goto** statement provides an unconditional jump from the goto to a labeled statement in the same function.

- **NOTE:** Use of **goto** statement is highly discouraged because it makes difficult to trace the control flow of a program, making the program hard to understand and hard to modify. Any program that uses a goto can be rewritten so that it doesn't need the goto.

Control Statements(cont.)

- The syntax of a goto statement in C++ is:

```
goto label;
..
.
label: statement;
```



Control Statements(cont.)

- **Example:**

```
#include <iostream>
  using namespace std;
  int main ()
{       int a = 10;
          LOOP:do
        {  if( a == 15)
         { // skip the iteration.
              a = a + 1;
              goto LOOP; }
        cout << "value of a: " << a << endl;
        a = a + 1;
        }while( a < 20 );
return 0; }
```

Output

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

some Examples

- Write For loop That calculates the total of the following series of numbers

  1/30 + 2/29 + 3/28 +…..+30/1

- Write the program that output

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

Thanks