Conditional Statements in c++

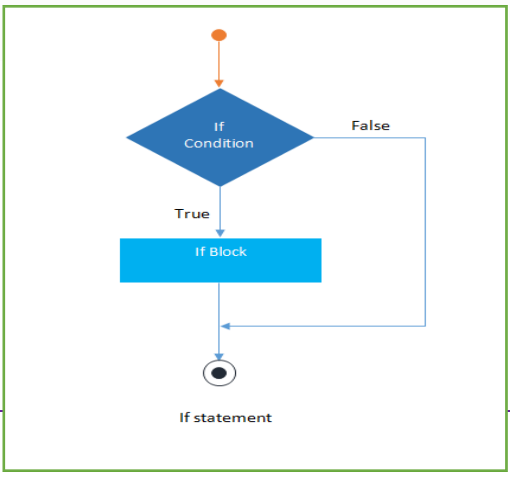
- if statement
- If-Else statement
- Nested If
- If-Else if statement
- Switch statement

If statement in C++

The single if statement is used to execute the code if a condition is true. It is also called one-way selection statement.

Syntax

```
if(condition)
{
            Statement(s);
}
```



Example of if statement

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

int main()
{
    int a,b ;
     a=50;
     b=50;
    if ( a == b)
        cout << " a is equal to b"<<end1;

    return 0;
}
```

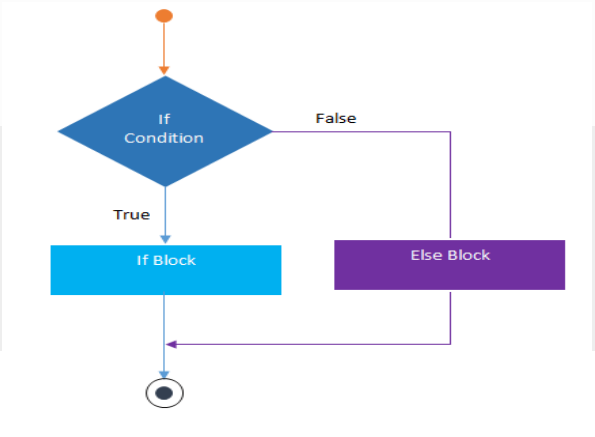
- To execute only one statement in if or else (as in previous example), you can skip curly brackets { }. If you want to execute multiple statements in if, then enclose the statements within curly brackets { }

If-else statement in c++

The if-else statement is used to execute the code if condition is true or false. It is also called two-way selection statement.

Syntax

```
if(condition) {
   Statement(s1);
}
else {
   Statement(s2);
}
```



If-Else statement

Example of if-else statement

```
#include <iostream>
using namespace std;
int main(){
   int num=66;
   if( num < 50 ){
      //This would run if above condition is true
      cout<<"num is less than 50";
   }
   else {
      //This would run if above condition is false
      cout<<"num is greater than or equal 50";
   }
   return 0;
```

Nested if statement in C++

When there is an if statement inside another if statement then it is called the **nested if statement**.
The structure of nested if looks like this:

```
if(condition_1) {
   Statement1(s);

   if(condition_2) {
      Statement2(s);
   }
}
```

Node: Statement1 would execute if the condition_1 is true. Statement2 would only execute if both the conditions(condition_1 and condition_2) are true.

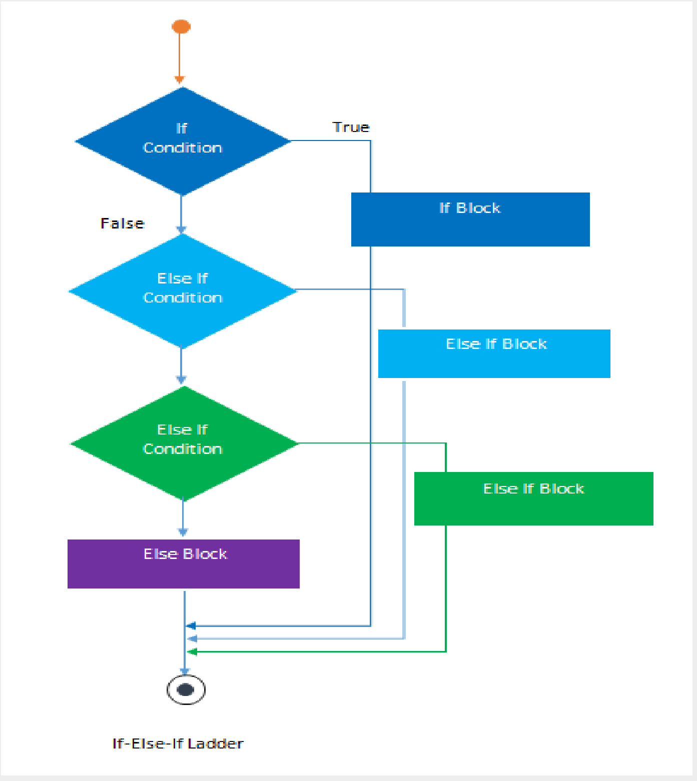
Example of Nested if statement

```
#include <iostream>
using namespace std;
int main(){
   int num=90;
   /* Nested if statement. An if statement
    * inside another if body
    */
   if( num < 100 ){
      cout<<"number is less than 100"<<endl;
      if(num > 50){
         cout<<"number is greater than 50";
      }
   }
   return 0;
```

if-else-if Statement in C++

if-else-if statement is used when we need to check multiple conditions. In this control structure we have only one “if” and one “else”, however we can have multiple “else if” blocks. This is how it looks:

```
if(condition_1) {
   /*if condition_1 is true execute this*/
   statement(s1);
}
else if(condition_2) {
   /* execute this if condition_1 is not met and
    * condition_2 is met
    */
   statement(s);
}
else if(condition_3) {
   /* execute this if condition_1 & condition_2 are
    * not met and condition_3 is met
    */
   statement(s);
}
.
.
.
else {
   /* if none of the condition is true
    * then these statements gets executed
    */
   statement(s);
}
```



C++ switch...case

C++ switch...case syntax

```
switch (n)

{

    case constant1:

        // code to be executed if n is equal to constant1;

        break;

    case constant2:

        // code to be executed if n is equal to constant2;

        break;

        .

        .

        .

    default:

        // code to be executed if n doesn't match any constant
```

Example : C++ Program to Check Whether Number is Even or Odd

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

int main()
{
    int n;
    cout << "Enter an integer: ";
    cin >> n;

    if ( n % 2 == 0)
        cout << n << " is even.";
    else
        cout << n << " is odd.";

    return 0;
}
```

Example: C++ Program to Find All Roots of a Quadratic Equation

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$D = b^2 - 4ac$$

If determinant > 0,

$$root1 = \frac{-b + \sqrt{(b^2 - 4ac)}}{2a}$$

$$root2 = \frac{-b - \sqrt{(b^2 - 4ac)}}{2a}$$

If determinant = 0,

$$root1 = root2 = \frac{-b}{2a}$$

If determinant < 0,

$$root1 = \frac{-b}{2a} + i\frac{\sqrt{-(b^2 - 4ac)}}{2a}$$

$$root2 = \frac{-b}{2a} - i\frac{\sqrt{-(b^2 - 4ac)}}{2a}$$

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

int main() {

    float a, b, c, x1, x2, D;
    cout << "Enter coefficients a, b and c: ";
    cin >> a >> b >> c;
    D = b * b - 4 * a * c;

    if (D > 0) {
        x1 = (-b + sqrt(D)) / (2 * a); //sqrt() library function is used to find the square root of a number
        x2 = (-b - sqrt(D)) / (2 * a);
        cout << "Roots are real and different." << endl;
        cout << "x1 = " << x1 << endl;
        cout << "x2 = " << x2 << endl;
    }

    else if (D == 0) {
        cout << "Roots are real and same." << endl;
        x1 = (-b ) / (2 * a);
        cout << "x1 = x2 =" << x1 << endl;
    }

    else {

        cout << "Roots are complex and different." << endl;
    }

    return 0;
}
```

Example: Find maximum value of two variables

```
#include <iostream>
using namespace std;
int main() {
	cout << "Insert two numbers to find the maximum one\n";
	cout << "-------------------------------------------\n";
	double FNum, SNum;
	cout << "First Number= ";
	cin >> FNum;
		cout << "Second Number= ";
		cin >> SNum;
		cout << "-------------------------------------------\n";
	if (FNum > SNum)
		cout << "First Number= " << FNum << " Is the maximum Number\n";
	else if  (FNum < SNum) cout << "Second Number= " << SNum << " Is the maximum
Number\n";
	else cout << "First Number = Second Number";
	return 0;
}
```

Example: C++ Program to Find Largest Number Among Three Numbers

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

int main()
{
    float n1, n2, n3;

    cout << "Enter three numbers: ";
    cin >> n1 >> n2 >> n3;
    if ((n1 >= n2) && (n1 >= n3))
        cout << "Largest number: " << n1;
    else if ((n2 >= n1) && (n2 >= n3))
        cout << "Largest number: " << n2;
    else
        cout << "Largest number: " << n3;

    return 0;
```

Example: C++ Program to read the numbers from 1 to 7 and display their correspondence day of week using if statement.

```
#include <iostream>
using namespace std;
 int main()
{
	int Day_Number;
	cin >> Day_Number;
	if (Day_Number == 1) {
		cout << " saturday ";
	}
	else if(Day_Number == 2){
		cout << " sunday ";
	}
	else if (Day_Number == 3) {
		cout << " monday ";
```

```
        }
        else if (Day_Number == 4) {
               cout << " Tuerday ";
        }
        else if (Day_Number == 5) {
               cout << " wednesday ";
        }
        else if (Day_Number == 6) {
               cout << " thursday ";
        }
        else if (Day_Number == 7) {
               cout << " friday ";
        }
        else{
               cout << "error";
        }
        return 0;
}
```

Example: C++ Program to read the numbers from 1 to 7 and display their correspondence day of week using case statement.

```
#include <iostream>
using namespace std;
int main()
 {
     int Day_Number;
     cout << "inter the number of day";
     cin>> Day_Number;
     switch (Day_Number) {
     case 1:cout << " saturday ";
         break;
     case 2: cout << " sunday ";
         break;
     case 3: cout << " monday ";
         break;
     case 4: cout << " Tuerday ";
         break;
     case 5: cout << " wednesday ";
         break;
     case 6: cout << " thursday ";
         break;
     case 7: cout << " friday ";
         break;
     default: cout << "error";
     }
 }
```

Task

Example:C++ Program to read student's mark as integer then print the equivalent grade depends on the following table:

0≤Mark<60	60≤Mark<65	65≤Mark<75	75≤Mark<85	85≤Mark<100
Fail	OK	Good	Very Good	Excellent

Example:C++ Program to:

a) Read an employee name (NAME), overtime hours worked (OVERTIME), hours absent (ABSENT)

b) Determine the bonus payment (PAYMENT).

Bonus Schedule	
OVERTIME – (2/3)*ABSENT	Bonus Paid
>40 hours	$50
>30 but ≤ 40 hours	$40
>20 but ≤ 30 hours	$30
>10 but ≤ 20 hours	$20
≤ 10 hours	$10

C++ loops

Loops are used in programming to repeat a specific block until some end condition is met. There are three type of loops in C++ programming:

1. for loop
2. while loop
3. do...while loop

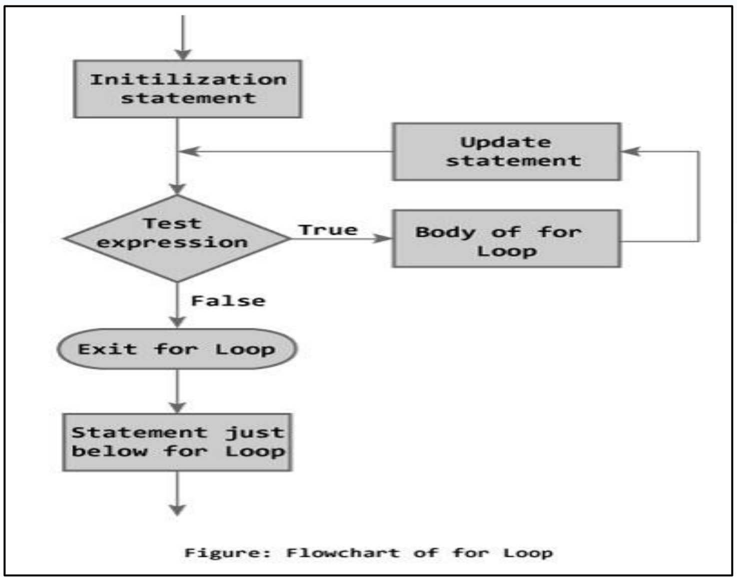
C++ for Loop Syntax

```
for(initialization Statement(counter); testExpression; updateStatement)

 {

    // code

}
```

How for loop works?

1. The initialization statement is executed only once at the beginning.
2. Then, the test expression is evaluated.
3. If the test expression is false, for loop is terminated. But if the test expression is true, codes inside body of `for` loop is executed and update expression is updated.
4. Again, the test expression is evaluated and this process repeats until the test expression is false.

Flowchart of for Loop in C++



Example 1: C++ for Loop

```
// C++ Program to find factorial of a number
// Factorial on n = 1*2*3*...*n

#include <iostream>
using namespace std;

int main()
{
    int i, n, factorial = 1;

    cout << "Enter a positive integer: ";
    cin >> n;

    for (i = 1; i <= n; ++i) {
        factorial *= i;   // factorial = factorial * i;
    }

    cout<< "Factorial of "<<n<<" = "<<factorial;
    return 0;
}
```

C++ while and do...while Loop

C++ while Loop

The syntax of a while loop is:

```
while (testExpression)

{

     // codes

}
```

How while loop works?

- The while loop evaluates the test expression.
- If the test expression is true, codes inside the body of while loop is evaluated.
- Then, the test expression is evaluated again. This process goes on until the test expression is false.
- When the test expression is false, while loop is terminated.

Flowchart of while Loop

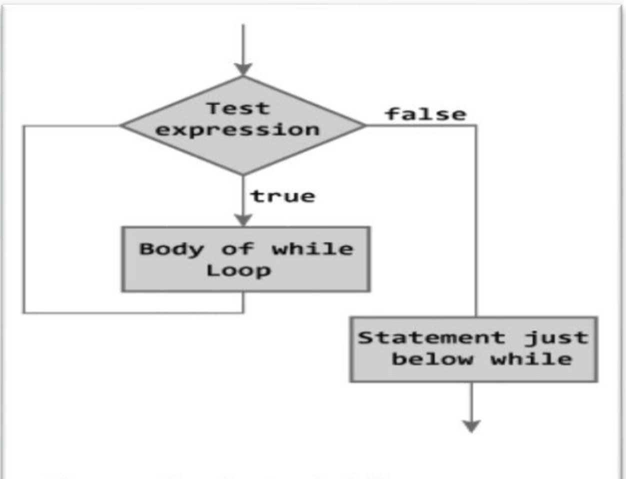


Figure: Flowchart of while Loop

Example 1: C++ while Loop

```
// C++ Program to compute factorial of a number
// Factorial of n = 1*2*3...*n

#include <iostream>
using namespace std;

int main()
{
    int number, i = 1, factorial = 1;

    cout << "Enter a positive integer: ";
    cin >> number;

    while ( i <= number) {
        factorial *= i;      //factorial = factorial * i;
        ++i;
    }

    cout<<"Factorial of "<< number <<" = "<< factorial;
    return 0;
}
```

C++ do...while Loop

The do...while loop is a variant of the while loop with one important difference. The body of do...while loop is executed once before the test expression is checked.

The syntax of do..while loop is:

```
do {

   // codes;

}

while (testExpression);
```

How do...while loop works?

- The codes inside the body of loop is executed at least once. Then, only the test expression is checked.
- If the test expression is true, the body of loop is executed. This process continues until the test expression becomes false.
- When the test expression is false, do...while loop is terminated

Flowchart of do...while Loop

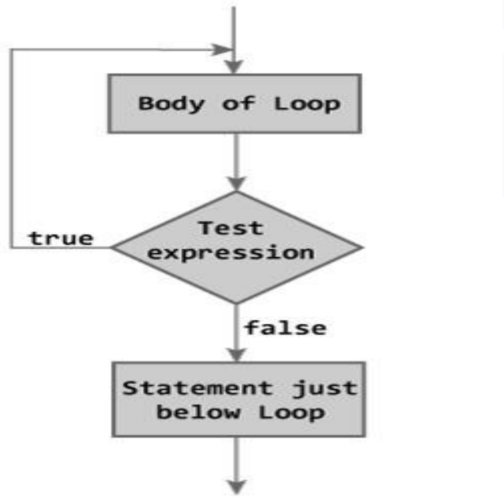


Figure: Flowchart of do...while Loop

Example 2: C++ do...while Loop

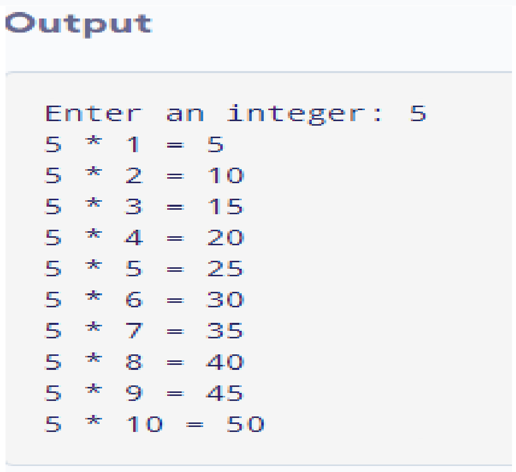
```
// C++ program to add numbers until user enters 0

#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;
```

Example: C++ Program to Calculate Sum of Natural Numbers

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

int main()
{
    int n, sum = 0;

    cout << "Enter a positive integer: ";
    cin >> n;

    for (int i = 1; i <= n; ++i) {
        sum += i;
    }

    cout << "Sum = " << sum;
    return 0;
```

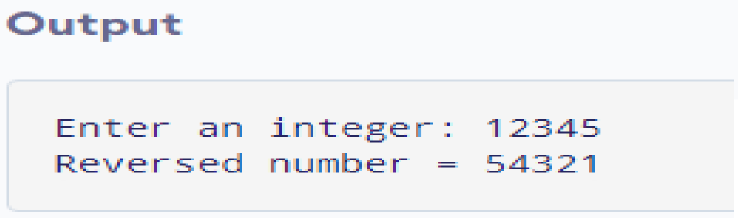
Task

1)C++ Program to Generate Multiplication Table


Output

```
Enter an integer: 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

2)C++ Program to Reverse an Integer


Output

```
Enter an integer: 12345
Reversed number = 54321
```

Send to email	cpp.programing1@gmail.com