## Control Statement in C++

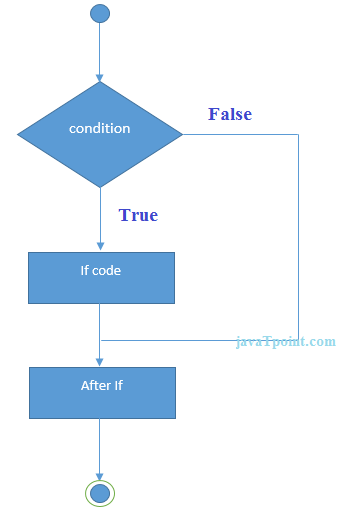
In C++ programming, if statement is used to test the condition. There are various types of if statements in C++.

* if statement
* if-else statement
* nested if statement
* if-else-if ladder

1. **If Statement**

The C++ if statement tests the condition. It is executed if condition is true.

1. **if**(condition){
2. //code to be executed
3. }



**Example**

1. #include <iostream>
2. **using** **namespace** std;
4. **int** main () {
5. **int** num = 10;
6. **if** (num % 2 == 0)
7. {
8. cout<<"It is even number";
9. }
10. **return** 0;
11. }

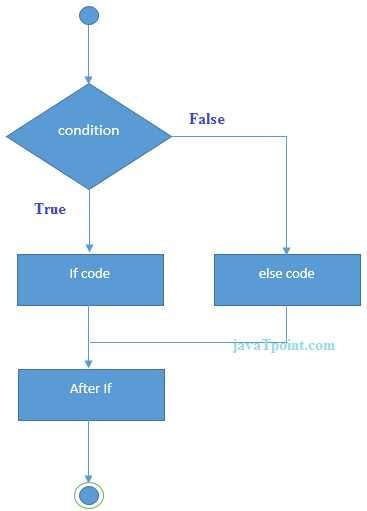
Output:/p>

It is even number

1. **IF-else Statement**

The C++ if-else statement also tests the condition. It executes if block if condition is true otherwise else block is executed.

1. **if**(condition){
2. //code if condition is true
3. }**else**{
4. //code if condition is false
5. }



**Example**

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main () {
4. **int** num = 11;
5. **if** (num % 2 == 0)
6. {
7. cout<<"It is even number";
8. }
9. **else**
10. {
11. cout<<"It is odd number";
12. }
13. **return** 0;
14. }

**Output:**

It is odd number

**If-else Example: with input from user**

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main () {
4. **int** num;
5. cout<<"Enter a Number: ";
6. cin>>num;
7. **if** (num % 2 == 0)
8. {
9. cout<<"It is even number"<<endl;
10. }
11. **else**
12. {
13. cout<<"It is odd number"<<endl;
14. }
15. **return** 0;
16. }

**Output:**

Enter a number:11

It is odd number

**Output:**

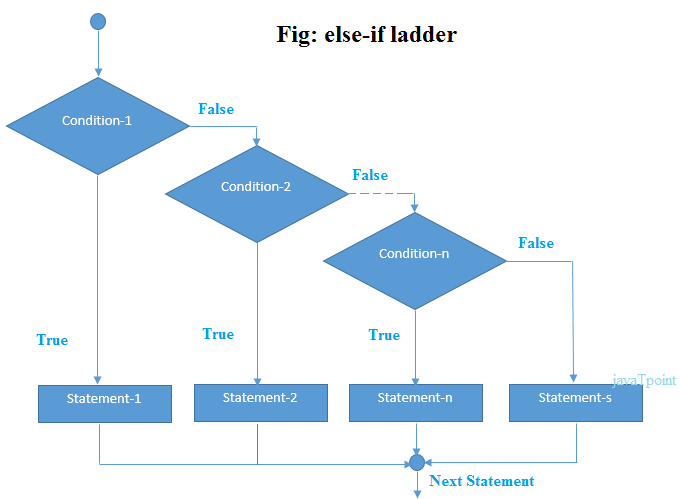
Enter a number:12

It is even number

1. **IF-else-if ladder Statement**

The C++ if-else-if ladder statement executes one condition from multiple statements.

1. **if**(condition1){
2. //code to be executed if condition1 is true
3. }**else** **if**(condition2){
4. //code to be executed if condition2 is true
5. }
6. **else** **if**(condition3){
7. //code to be executed if condition3 is true
8. }
9. ...
10. **else**{
11. //code to be executed if all the conditions are false
12. }



**If else-if Example**

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main () {
4. **int** num;
5. cout<<"Enter a number to check grade:";
6. cin>>num;
7. **if** (num <0 || num >100)
8. {
9. cout<<"wrong number";
10. }
11. **else** **if**(num >= 0 && num < 50){
12. cout<<"Fail";
13. }
14. **else** **if** (num >= 50 && num < 60)
15. {
16. cout<<"D Grade";
17. }
18. **else** **if** (num >= 60 && num < 70)
19. {
20. cout<<"C Grade";
21. }
22. **else** **if** (num >= 70 && num < 80)
23. {
24. cout<<"B Grade";
25. }
26. **else** **if** (num >= 80 && num < 90)
27. {
28. cout<<"A Grade";
29. }
30. **else** **if** (num >= 90 && num <= 100)
31. {
32. cout<<"A+ Grade";
33. }
34. }

**Output:**

Enter a number to check grade:66

C Grade

**Output:**

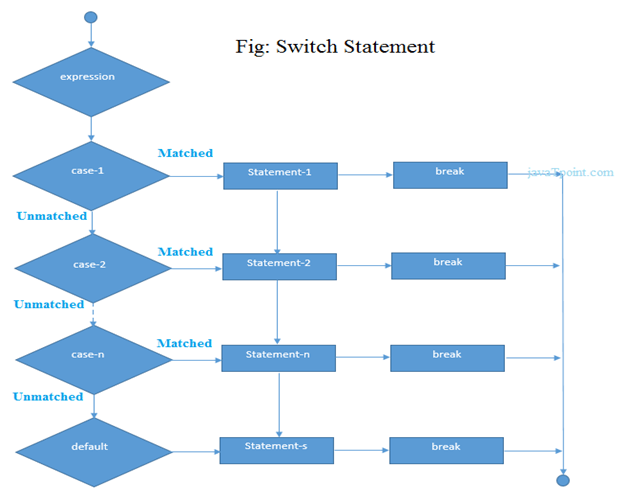
Enter a number to check grade:-2

wrong number

**C++ switch**

The C++ switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement in C++.

1. **switch**(expression){
2. **case** value1:
3. //code to be executed;
4. **break**;
5. **case** value2:
6. //code to be executed;
7. **break**;
8. ......
10. **default**:
11. //code to be executed if all cases are not matched;
12. **break**;
13. }



**C++ Switch Example**

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main () {
4. **int** num;
5. cout<<"Enter a number to check grade:";
6. cin>>num;
7. **switch** (num)
8. {
9. **case** 10: cout<<"It is 10"; **break**;
10. **case** 20: cout<<"It is 20"; **break**;
11. **case** 30: cout<<"It is 30"; **break**;
12. **default**: cout<<"Not 10, 20 or 30"; **break**;
13. }
14. }

Output:

Enter a number:

10

It is 10

Output:

Enter a number:

55

Not 10, 20 or 30

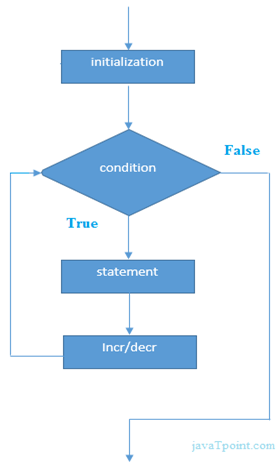
**C++ For Loop**

The C++ for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop than while or do-while loops.

The C++ for loop is same as C/C#. We can initialize variable, check condition and increment/decrement value.

1. **for**(initialization; condition; incr/decr){
2. //code to be executed
3. }

**Flowchart:**



**For Loop Example**

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main() {
4. **for**(**int** i=1;i<=10;i++){
5. cout<<i <<"\n";
6. }
7. }

Output:

1

2

3

4

5

6

7

8

9

10

**C++ Nested For Loop**

In C++, we can use for loop inside another for loop, it is known as nested for loop. The inner loop is executed fully when outer loop is executed one time. So if outer loop and inner loop are executed 4 times, inner loop will be executed 4 times for each outer loop i.e. total 16 times.

**C++ Nested For Loop Example**

Let's see a simple example of nested for loop in C++.

1. #include <iostream>
2. **using** **namespace** std;
4. **int** main () {
5. **for**(**int** i=1;i<=3;i++){
6. **for**(**int** j=1;j<=3;j++){
7. cout<<i<<" "<<j<<"\n";
8. }
9. }
10. }

Output:

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

**Infinite For Loop**

If we use double semicolon in for loop, it will be executed infinite times. Let's see a simple example of infinite for loop in C++.

1. #include <iostream>
2. **using** **namespace** std;
4. **int** main () {
5. **for** (; ;)
6. {
7. cout<<"Infinitive For Loop";
8. }
9. }

Output:

Infinitive For Loop

Infinitive For Loop

Infinitive For Loop

Infinitive For Loop

Infinitive For Loop

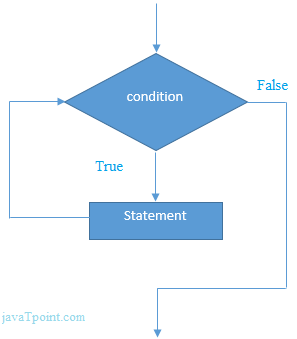
ctrl+c

**C++ While loop**

In C++, while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop than for loop.

1. **while**(condition){
2. //code to be executed
3. }

**Flowchart:**



**While Loop Example**

Let's see a simple example of while loop to print table of 1.

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main() {
4. **int** i=1;
5. **while**(i<=10)
6. {
7. cout<<i <<"\n";
8. i++;
9. }
10. }

Output:

1

2

3

4

5

6

7

8

9

10

**Nested While Loop Example**

In C++, we can use while loop inside another while loop, it is known as nested while loop. The nested while loop is executed fully when outer loop is executed once.

Let's see a simple example of nested while loop in C++ programming language.

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main () {
4. **int** i=1;
5. **while**(i<=3)
6. {
7. **int** j = 1;
8. **while** (j <= 3)
9. {
10. cout<<i<<" "<<j<<"\n";
11. j++;
12. }
13. i++;
14. }
15. }

Output:

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

**Infinitive While Loop Example:**

We can also create infinite while loop by passing true as the test condition.

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main () {
4. **while**(**true**)
5. {
6. cout<<"Infinitive While Loop";
7. }
8. }

Output:

Infinitive While Loop

Infinitive While Loop

Infinitive While Loop

Infinitive While Loop

Infinitive While Loop

ctrl+c

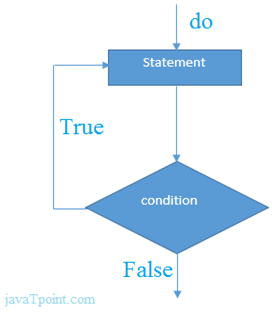
**C++ Do-While Loop**

The C++ do-while loop is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.

The C++ do-while loop is executed at least once because condition is checked after loop body.

1. **do**{
2. //code to be executed
3. }**while**(condition);

**Flowchart:**



**do-while Loop Example**

Let's see a simple example of C++ do-while loop to print the table of 1.

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main() {
4. **int** i = 1;
5. **do**{
6. cout<<i<<"\n";
7. i++;
8. } **while** (i <= 10) ;
9. }

Output:

1

2

3

4

5

6

7

8

9

10

**Nested do-while Loop**

In C++, if you use do-while loop inside another do-while loop, it is known as nested do-while loop. The nested do-while loop is executed fully for each outer do-while loop.

Let's see a simple example of nested do-while loop in C++.

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main() {
4. **int** i = 1;
5. **do**{
6. **int** j = 1;
7. **do**{
8. cout<<i<<"\n";
9. j++;
10. } **while** (j <= 3) ;
11. i++;
12. } **while** (i <= 3) ;
13. }

Output:

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

**Infinitive do-while Loop**

In C++, if you pass **true** in the do-while loop, it will be infinitive do-while loop.

1. **do**{
2. //code to be executed
3. }**while**(**true**);

**Infinitive do-while Loop Example**

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main() {
4. **do**{
5. cout<<"Infinitive do-while Loop";
6. } **while**(**true**);
7. }

Output:

Infinitive do-while Loop

Infinitive do-while Loop

Infinitive do-while Loop

Infinitive do-while Loop

Infinitive do-while Loop

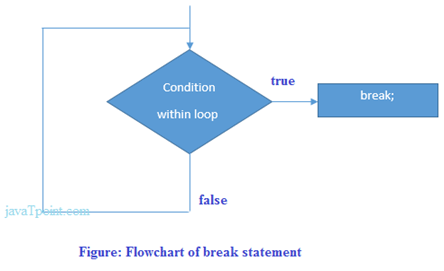
ctrl+c

**C++ Break Statement**

The C++ break is used to break loop or switch statement. It breaks the current flow of the program at the given condition. In case of inner loop, it breaks only inner loop.

1. jump-statement;
2. **break**;

**Flowchart:**



**Break Statement Example**

Let's see a simple example of C++ break statement which is used inside the loop.

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main() {
4. **for** (**int** i = 1; i <= 10; i++)
5. {
6. **if** (i == 5)
7. {
8. **break**;
9. }
10. cout<<i<<"\n";
11. }
12. }

Output:

1

2

3

4

**C++ Break Statement with Inner Loop**

The C++ break statement breaks inner loop only if you use break statement inside the inner loop.

Let's see the example code:

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main()
4. {
5. **for**(**int** i=1;i<=3;i++){
6. **for**(**int** j=1;j<=3;j++){
7. **if**(i==2&&j==2){
8. **break**;
9. }
10. cout<<i<<" "<<j<<"\n";
11. }
12. }
13. }

Output:

1 1

1 2

1 3

2 1

3 1

3 2

3 3

**C++ Continue Statement**

The C++ continue statement is used to continue loop. It continues the current flow of the program and skips the remaining code at specified condition. In case of inner loop, it continues only inner loop.

1. jump-statement;
2. **continue**;

**Continue Statement Example**

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main()
4. {
5. **for**(**int** i=1;i<=10;i++){
6. **if**(i==5){
7. **continue**;
8. }
9. cout<<i<<"\n";
10. }
11. }

Output:

1

2

3

4

6

7

8

9

10

**Continue Statement with Inner Loop**

C++ Continue Statement continues inner loop only if you use continue statement inside the inner loop.

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main()
4. {
5. **for**(**int** i=1;i<=3;i++){
6. **for**(**int** j=1;j<=3;j++){
7. **if**(i==2&&j==2){
8. **continue**;
9. }
10. cout<<i<<" "<<j<<"\n";
11. }
12. }
13. }

Output:

1 1

1 2

1 3

2 1

2 3

3 1

3 2

3 3

**C++ Goto Statement**

The C++ goto statement is also known as jump statement. It is used to transfer control to the other part of the program. It unconditionally jumps to the specified label.

It can be used to transfer control from deeply nested loop or switch case label.

**Goto Statement Example**

Let's see the simple example of goto statement in C++.

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main()
4. {
5. ineligible:
6. cout<<"You are not eligible to vote!\n";
7. cout<<"Enter your age:\n";
8. **int** age;
9. cin>>age;
10. **if** (age < 18){
11. **goto** ineligible;
12. }
13. **else**
14. {
15. cout<<"You are eligible to vote!";
16. }
17. }

Output:

You are not eligible to vote!

Enter your age:

16

You are not eligible to vote!

Enter your age:

7

You are not eligible to vote!

Enter your age:

22

You are eligible to vote!