Flow control

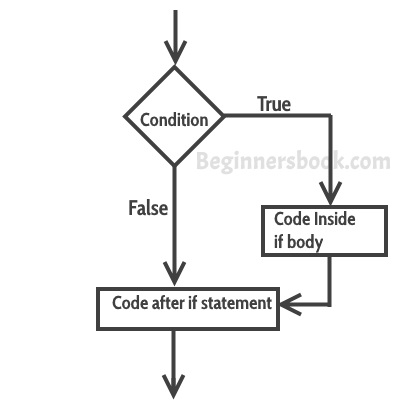
**If statement in C++**

If statement consists a condition, followed by statement or a set of statements as shown below:

if(condition){

Statement(s);

}

**Flow diagram of If statement**  


### Example of if statement

#include <iostream>

using namespace std;

int main(){

int num=70;

if( num < 100 ){

/\* This cout statement will only execute,

\* if the above condition is true

\*/

cout<<"number is less than 100";

}

if(num > 100){

/\* This cout statement will only execute,

\* if the above condition is true

\*/

cout<<"number is greater than 100";

}

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);

}

}

### 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 statement in C++

Sometimes you have a condition and you want to execute a block of code if condition is true and execute another piece of code if the same condition is false. This can be achieved in C++ using if-else statement.

This is how an if-else statement looks:

if(condition) {

Statement(s);

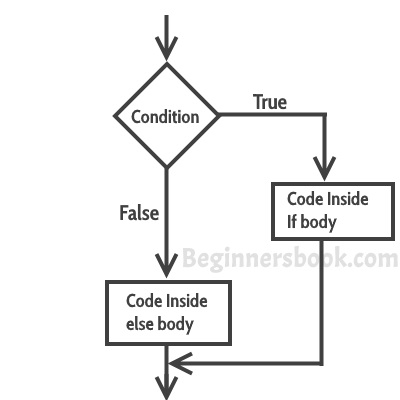
}

else {

Statement(s);

}

The statements inside “if” would execute if the condition is true, and the statements inside “else” would execute if the condition is false.

**Flow diagram of if-else**  


### 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;

}

### Example of if-else-if

#include <iostream>

using namespace std;

int main(){

int num;

cout<<"Enter an integer number between 1 & 99999: ";

cin>>num;

if(num <100 && num>=1) {

cout<<"Its a two digit number";

}

else if(num <1000 && num>=100) {

cout<<"Its a three digit number";

}

else if(num <10000 && num>=1000) {

cout<<"Its a four digit number";

}

else if(num <100000 && num>=10000) {

cout<<"Its a five digit number";

}

else {

cout<<"number is not between 1 & 99999";

}

return 0;

}

if(num <100 && num>=1) {

cout<<"Its a two digit number";

}

if(num <1000 && num>=100) {

cout<<"Its a three digit number";

}

if(num <10000 && num>=1000) {

cout<<"Its a four digit number";

}

if(num <100000 && num>=10000) {

cout<<"Its a five digit number";

}

If (num>=100000){

cout<<"number is not between 1 & 99999";

}

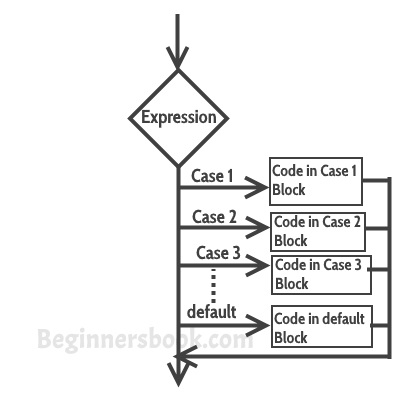
return 0;

}

# Switch Case statement in C++ with example

Switch case statement is used when we have multiple conditions and we need to perform different action based on the condition.

## Switch Case Flow Diagram

It evaluates the value of expression or variable (based on whatever is given inside switch braces), then based on the outcome it executes the corresponding case.  


## Break statement in Switch Case

Before we discuss about break statement, Let’s see what happens when we don’t use break statement in switch case.

#include <iostream>

using namespace std;

int main(){

int num=5;

switch(num+2) {

case 1:

cout<<"Case1: Value is: "<<num<<endl;

case 2:

cout<<"Case2: Value is: "<<num<<endl;

case 3:

cout<<"Case3: Value is: "<<num<<endl;

default:

cout<<"Default: Value is: "<<num<<endl;

}

return 0;

}

Output 5

You can also use characters in switch case. for example –

#include <iostream>

using namespace std;

int main(){

char ch='b';

switch(ch) {

case 'd': cout<<"Case1 ";

break;

case 'b': cout<<"Case2 ";

break;

case 'x': cout<<"Case3 ";

break;

case 'y': cout<<"Case4 ";

break;

default: cout<<"Default ";

}

return 0;

}

#### Syntax of for loop

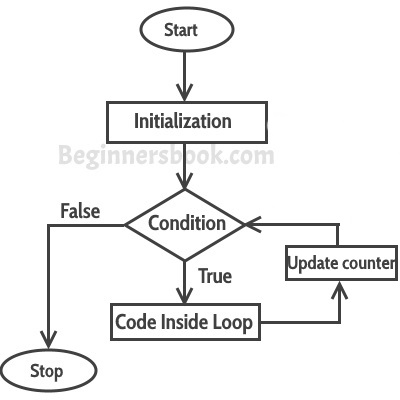
for(initialization; condition ; increment/decrement)

{

C++ statement(s);

}

## Flow of Execution of the for Loop

As a program executes, the interpreter always keeps track of which statement is about to be executed. We call this the control flow, or the flow of execution of the program.  


#include <iostream>

using namespace std;

int main(){

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

{

cout<<"Value of variable i is: "<<i<<endl;

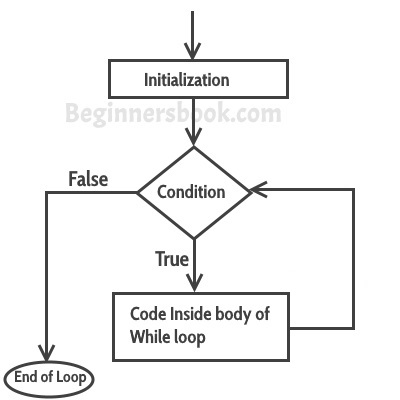
}

return 0;

}

# While loop

### Flow Diagram of While loop



#include <iostream>

using namespace std;

int main(){

int i=1;

while(i<=6)

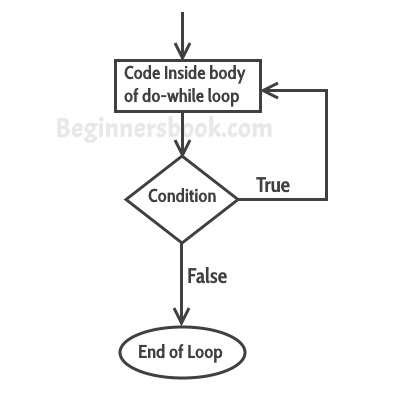
{

cout<<"Value of variable i is: "<<i<<endl; i++;

}

}

# do-while loop



## do-while loop example in C++

#include <iostream>

using namespace std;

int main(){

int num=1;

do{

cout<<"Value of num: "<<num<<endl;

num++;

}while(num<=6);

return 0;

}

# Continue Statement

Continue statement is used inside loops. Whenever a continue statement is encountered inside a loop, control directly jumps to the beginning of the loop for next iteration, skipping the execution of statements inside loop’s body for the current iteration

#include <iostream>

using namespace std;

int main(){

for (int num=0; num<=6; num++) {

if (num==3) {

continue;

}

cout<<num<<" ";

}

return 0;

}

**Output:**

0 1 2 4 5 6

## Example: Use of continue in While loop

#include <iostream>

using namespace std;

int main(){

int j=6;

while (j >=0) {

if (j==4) {

j--;

continue;

}

cout<<"Value of j: "<<j<<endl;

j--;

}

return 0;

}

**Output:**

Value of j: 6

Value of j: 5

Value of j: 3

Value of j: 2

Value of j: 1

Value of j: 0

# Break statement

The **break statement** is used in following two scenarios:

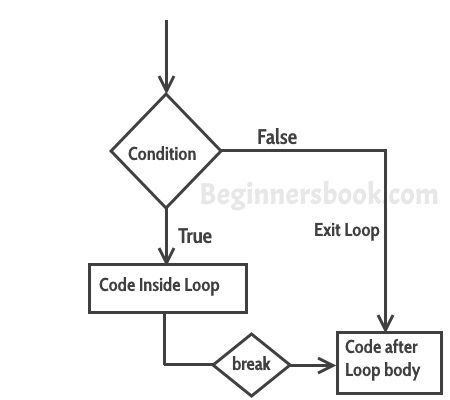
a) Use break statement to come out of the loop instantly. Whenever a break statement is encountered inside a loop, the control directly comes out of loop terminating it. It is used along with if statement, whenever used inside loop(see the example below) so that it occurs only for a particular condition.

b) It is used in switch case control structure after the case blocks. Generally all cases in switch case are followed by a break statement to avoid the subsequent cases (see the example below) execution. Whenever it is encountered in switch-case block, the control comes out of the switch-case body.

#### Syntax of break statement

break;

## break statement flow diagram



#include <iostream>

using namespace std;

int main(){

int num =10;

while(num<=200) {

cout<<"Value of num is: "<<num<<endl;

if (num==12) {

break;

}

num++;

}

cout<<"Hey, I'm out of the loop";

return 0;

}

**Output:**

Value of num is: 10

Value of num is: 11

Value of num is: 12

Hey, I'm out of the loop

# goto statement

The goto statement is used for transferring the control of a program to a given label. The syntax of goto statement looks like this:

goto label\_name;

## Example of goto statement in C++

#include <iostream>

using namespace std;

int main(){

int num; cout<<"Enter a number: ";

cin>>num;

if (num % 2==0){

goto raj;

}

else {

cout<<"Odd Number";

}

raj:

cout<<"Even Number";

return 0;

}

**Output:**

Enter a number: 42

Even Number

1 – Mon

2 – Tue

3 – Wed

….

7 – Sun

Out of range