**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. When we have multiple conditions and we need to execute a block of statements when a particular condition is satisfied. In such case either we can use lengthy [if..else-if statement](https://beginnersbook.com/2017/08/cpp-if-else-statement/) or switch case. The problem with lengthy if..else-if is that it becomes complex when we have several conditions. The switch case is a clean and efficient method of handling such scenarios.

The **syntax of Switch case** statement:

switch (variable or an integer expression)

{

case constant:

//C++ code

;

case constant:

//C++ code

;

default:

//C++ code

;

}

Switch Case statement is mostly used with break statement even though the break statement is optional. We will first see an example without break statement and then we will discuss switch case with break

**Example of 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;

}

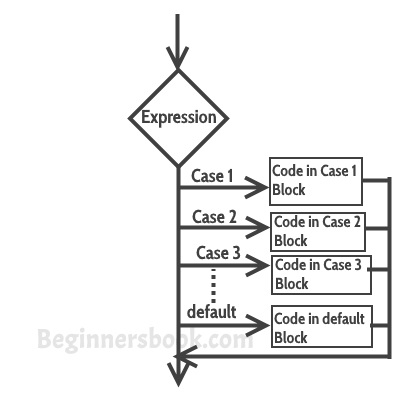
return 0;

}

**Output:**

Default: Value is: 5

## 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. See the example below:

#include <iostream>

using namespace std;

int main(){

int i=2;

switch(i) {

case 1: cout<<"Case1 "<<endl;

case 2: cout<<"Case2 "<<endl;

case 3: cout<<"Case3 "<<endl;

case 4: cout<<"Case4 "<<endl;

default: cout<<"Default "<<endl;

}

return 0;

}

**Output:**

Case2

Case3

Case4

Default

# Nested loops in C++ Programming

A loop inside another loop is called a nested loop. The number of loops depend on the complexity of a problem. Suppose, a loop, outer loop, running n number of times consists of another loop inside it, inner loop, running m number of times. Then, for each execution of the outer loop from 1...n, the inner loop runs maximum of m times.

## Types of nested loops

There can be many types of nested loops in C++ but the mostly used nested loops are

* [Nested while loop](https://www.programtopia.net/cplusplus/docs/nested-loops#nested-while)
* [Nested do-while loop](https://www.programtopia.net/cplusplus/docs/nested-loops#nested-do-while)
* [Nested for loop](https://www.programtopia.net/cplusplus/docs/nested-loops#nested-for)

**Note**: There can also be very variation of nested loops where a while loop can be inside a for loop, a for loop can be inside a do-while loop and many more.

## Nested while loop

A while loop inside another while loop is called nested while loop.

### Syntax of Nested while loop

while (condition1)

{

  statement(s);

  while (condition2)

{

  statement(s);

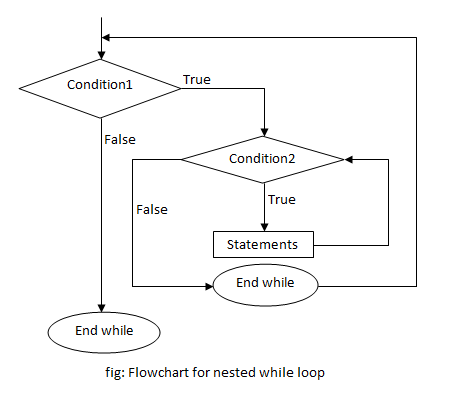
  ... ... ...

}

  ... ... ...

}

### Flowchart for Nested while loop



### Example of Nested while loop

**C++ program to print the number pattern.**

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

int i=1,j;

while (i <= 5)

{

j=1;

while (j <= i )

{

cout <<j;

j++;

}

cout << endl;

i++;

}

getch();

return 0;

}

## Nested do-while loop

A do-while loop inside another do-while loop is called nested do-while loop.

### Syntax of Nested do-while loop

do

{

  statement(s);

  do

{

  statement(s);

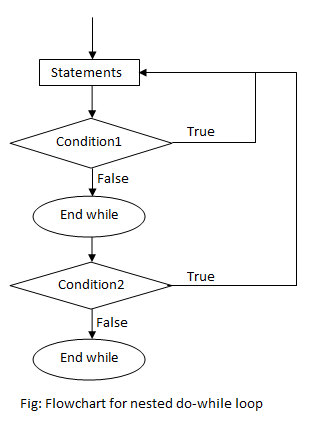
  ... ... ...

}while (condition2);

... ... ...

}while (condition1);

### Flowchart of Nested do-while loop



### Example of do-while loop

**C++ program to print the given star pattern.**

**\***

**\*\***

**\*\*\***

**\*\*\*\***

**\*\*\*\*\***

#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

int i=1,j;

do

{

j=1;

do

{

cout << "\*";

j++;

}while(j <= i);

i++;

cout << endl;

}while(i <= 5);

getch();

return 0;

}

In this program, nested do-while loop is used to print the star pattern. The outermost loop runs 5 times and for every loop, the innermost loop runs i times which is 1 at first, meaning only one "\*" is printed, then on the next loop it's 2 printing two stars and so on till 5 iterations of the loop executes, printing five stars. This way, the given star pattern is printed.

## Nested for loop

A for loop inside another for loop is called nested for loop.

### Syntax of Nested for loop

for (initialization; condition; increment/decrement)

{

  statement(s);

  for (initialization; condition; increment/decrement)

{

  statement(s);

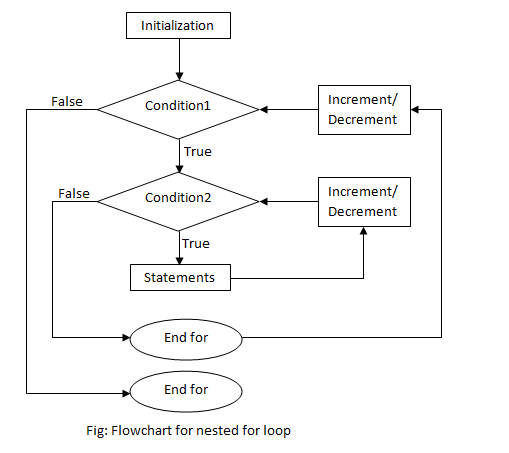
  ... ... ...

}

  ... ... ...

}

### Flowchart of Nested for loop



### Example of Nested for loop

**C++ program to find the sum of 2 matrices.**

#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

int a[10][10], b[10][10], s[10][10];

int i,j,row, column;

cout <<"Enter size of row:";

cin >> row;

cout <<"Enter size of column:";

cin >> column;

cout <<"Enter elements of matrix A" << endl;

for(i=0;i<row;i++)

{

for(j=0;j<column;j++)

{

cin >> a[i][j];

}

}

cout <<"Enter elements of matrix B" << endl;

for(i=0;i<row;i++)

{

for(j=0;j<column;j++)

{

cin >> b[i][j];

}

}

cout << "Sum of A and B" << endl;

for(i=0;i<row;i++)

{

for(j=0;j<column;j++)

{

cout << a[i][j] + b[i][j] << " ";

}

cout << endl;

}

getch();

return 0;

}

Nested for loop is used to calculate the sum of two 2-dimensional matrices. The program consists of three for nested loops where the outer loop runs equal to size of row and inner loop runs equal to size of column. The first and second are used for entering the values of elements for Matrix A and B, while the third is used for displaying the sum of the elements of the two matrices. Matrix A and B are stored in 2-dimensional arrays a and b respectively. In the final nested loop, each element of a and b is traversed and the sum is printed.

**Output**

Enter size of row:2

Enter size of column:3

Enter elements of matrix A

2 7 0

3 -1 7

Enter elements of matrix B

4 9 2

0 1 -8

Sum of A and B

6 16 2

3 0 -1