## ➢ Selection Statements

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

**Syntax:**

| Lab | |
|---|---|
| **Contents Covered:** | **Conditional statements and execution flow for conditional statements** <br>• Switch statement <br>• Nested Switch |

The syntax for a switch statement in C++ is as follows −

```
switch(expression) {
  case constant-expression  :
    statement(s);
    break; //optional
  case constant-expression  :
    statement(s);
    break; //optional
  // you can have any number of case statements.
  default : //Optional
    statement(s);
}
```

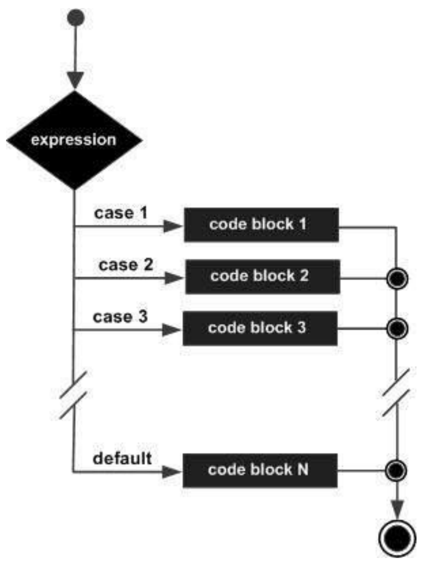
### Rules of switch statement:

The expression used in a switch statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type.
You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
The constant-expression for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.

When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached.
When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached.

A switch statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

➢ **Flow Diagram**



**<u>Switch case :</u>**

switch...case is a branching statement used to perform action based on available choices, instead of making decisions based on conditions. Using switch...case you can write more clean and optimal code than if...else statement. switch...case only works with integer, character and enumeration constants.

## <u>C++ switch statement</u>

**Example 1:Write a C++ program that uses a switch statement to evaluate a grade (A, B, C, D, or F). The program should print a message based on the grade entered, such as "Excellent!" for an A, "Well done" for B or C, "You passed" for a D, and "Better try again" for an F. If the input is not one of these grades, it should print "Invalid grade". Finally, display the grade entered by the user**

**Code:**

```
#include <iostream>
using namespace std;
int main () {
  // local variable declaration:
  char grade = 'D';
  switch(grade) {
    case 'A' :
      cout << "Excellent!" << endl;
break;
    case 'B' :
    case 'C' :
      cout << "Well done" << endl;
      break;
    case 'D' :
      cout << "You passed" << endl;
      break;
    case 'F' :
      cout << "Better try again" << endl;
      break;
    default :
      cout << "Invalid grade" << endl;
  }

  cout << "Your grade is " << grade << endl;

  return 0;
}
```

**Output:**

```
You passed
Your grade is D
```

**Example 2:Write a C++ program that asks the user to enter their grade (A, B, C, D, or F) and prints a message based on the grade. Ensure the program handles invalid inputs and displays the entered grade at the end.**

**Code**:

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

int main() {
   char grade;

   // Prompt the user to enter their grade
   cout << "Enter your grade (A, B, C, D, F): ";
   cin >> grade;

   // Switch statement to evaluate the grade
   switch(grade) {
      case 'A':
         cout << "Excellent!" << endl;
         break;
      case 'B':
      case 'C':
         cout << "Well done" << endl;
         break;
      case 'D':
         cout << "You passed" << endl;
```

```
            break;
        case 'F':
            cout << "Better try again" << endl;
            break;
        default:
            cout << "Invalid grade" << endl;
    }

    // Display the entered grade
    cout << "Your grade is " << grade << endl;

    return 0;
}
```

**OUTPUT:**

```
Enter your grade
(A, B, C, D, F): B
Well done
Your grade is B
```

# Lab Tasks:

**Task 1:**Program to build a simple calculator using switch Statement
**Task 2:**Program to check vowel and consonant using switch statement

**Task 3:** Write a program to use weekday numbers to calculate weekday name using switch statement.

**Task 4:** Write a program to check positive, negative or zero using switch statement.

- **List of switch case programming exercises**

1-Write a C++ program print total number of days in a month using switch case.

2-Write a C program to check whether a number is even or odd using switch case.

## ➢ Nested Switch

It is possible to have a switch as part of the statement sequence of an outer switch. Even if the case constants of the inner and outer switch contain common values, no conflicts will arise.

C++ specifies that at least 256 levels of nesting be allowed for switch statements.

**Syntax**

The syntax for a **nested switch** statement is as follows −

```
switch(ch1) {
case 'A':
cout << "This A is part of outer switch";
switch(ch2) {
case 'A':
cout << "This A is part of inner switch";
break;
case 'B': // ...
}
break;
case 'B': // ...
}
```

**Example**

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

int main () {
  // local variable declaration:
  int a = 100;
  int b = 200;

  switch(a) {
    case 100:
      cout << "This is part of outer switch" << endl;
      switch(b) {
        case 200:
          cout << "This is part of inner switch" << endl;
      }
  }
  cout << "Exact value of a is : " << a << endl;
  cout << "Exact value of b is : " << b << endl;

  return 0;
}
```

This would produce the following result −

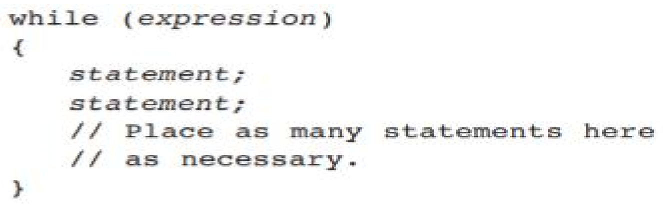
```
This is part of outer switch
This is part of inner switch
Exact value of a is : 100
Exact value of b is : 200
```

| Lab | |
|---|---|
| **Contents Covered:** | Basic loop structures<br>for loops<br>while Loops<br>do-while Loop |

**Objective:** To have better understanding regarding loops.

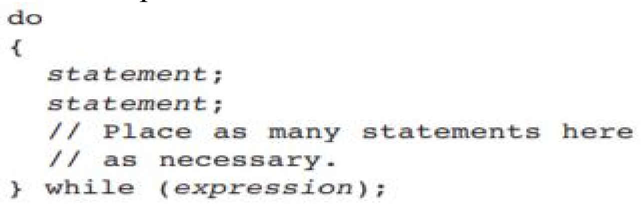
**LOOP:**

A loop is part of a program that repeats. 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.



```
while (expression)
{
    statement;
    statement;
    // Place as many statements here
    // as necessary.
}
```

The while Loop Is a Pretest Loop ,which means it tests its expression before each iteration whereas the do-while loop is a post-test loop, which means its expression is tested after each iteration.



```
do
{
   statement;
   statement;
   // Place as many statements here
   // as necessary.
} while (expression);
```

The for loop executes a section of code a fixed number of times. It consists of three parts:Initialization (optional),Test condition Increment/decrement expression (optional).

In computer programming, loops are used to repeat a block of code.

For example, let's say we want to show a message 100 times. Then instead of writing the print statement 100 times, we can use a loop.

That was just a simple example; we can achieve much more efficiency and sophistication in our programs by making effective use of loops.

There are 3 types of loops in C++.

- for loop
- while loop
- do...while loop

**C++ for loop**

The syntax of for-loop is:

```
for (initialization; condition; update) {
   // body of-loop
}
```

Here,

initialization - initializes variables and is executed only once

condition - if true, the body of for loop is executed

if false, the for loop is terminated

update - updates the value of initialized variables and again checks the condition

```
#include <iostream>
using namespace std;
int main() {
      for (int i = 1; i <= 5; ++i) {
      cout << i << " ";
   }
   return 0;
}
```

## ➢ C++ while and do...while Loop

The syntax of the while loop is:

```
while (condition) {
   // body of the loop
}
```

Here,

- A while loop evaluates the condition
- If the condition evaluates to true, the code inside the while loop is executed.
- The condition is evaluated again.
- This process continues until the condition is false.
- When the condition evaluates to false, the loop terminates.

**Example1: C++ Program to print numbers from 1 to 5**

```
#include <iostream>
using namespace std;
int main() {
    int i = 1;
    // while loop from 1 to 5
    while (i <= 5) {
        cout << i << " ";
        ++i;
    }
    return 0;
}
```

**Output**

```
1 2 3 4 5
```

## ➢ 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 condition is checked.

Its syntax is:

```
do {
   // body of loop;
}
while (condition);
```

Here,

- The body of the loop is executed at first. Then the condition is evaluated.
- If the condition evaluates to true, the body of the loop inside the do statement is executed again.
- The condition is evaluated once again.
- If the condition evaluates to true, the body of the loop inside the do statement is executed again.
- This process continues until the condition evaluates to false. Then the loop stops.

**Example 2: Display Numbers from 1 to 5**

// C++ Program to print numbers from 1 to 5

```
#include <iostream>
using namespace std;
int main() {
    int i = 1;
    // do...while loop from 1 to 5
    do {
```

**Output**

```
1 2 3 4 5
```

```
        cout << i << " ";
        ++i;
    }
    while (i <= 5);
    return 0;
}
```

## Infinite Loops:

If a loop does not have a way of stopping, it is called an infinite loop. An infinite loop continues to repeat until the program is interrupted. Here is an example of an infinite loop:

```
int number = 0;
while (number < 5)
{
    cout << "Hello\n";
}
```

We can make this loop finite by adding a line as shown below

```
while (number < 5)
{
    cout << "Hello\n";
    number++;
}
```

**Practice Problem**

**Example 1: Display Multiplication table up to 10**

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

int main() {
    int num;

    // Prompt the user to enter a number
    cout << "Enter a number to display its multiplication table: ";
    cin >> num;

    // Display multiplication table up to 10
    for (int i = 1; i <= 10; ++i) {
        cout << num << " x " << i << " = " << num * i << endl;
    }
```

**Output:**

```
Enter a number to display
its multiplication table: 2

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
    return 0;
}
```

**Example: Display multiplication table up to a given range**

```
#include <iostream>
using namespace std;
int main()
{
    int n, range;

    cout << "Enter an integer: ";
    cin >> n;
  cout << "Enter range: ";
    cin >> range;

    for (int i = 1; i <= range; ++i) {
        cout << n << " * " << i << " = " << n * i << endl;
    }

    return 0;
}
```

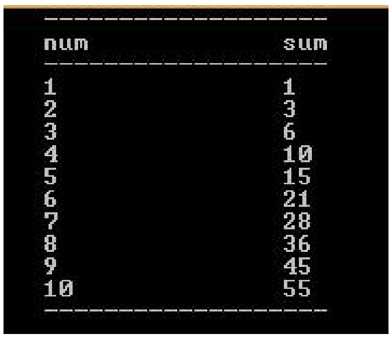
## Lab 5 Tasks 1

**Task 1:Write a C++ Program to Display Fibonacci Series.**

**Task 2:Write a program to print the sum of first 10 natural numbers**

**TASK#3: write a program to print first 20 natural numbers using for, while and do-while loop**
**TASK#4: Write a program to find factorial of any number input by a user?**
**TASK#5: Write a program that print sum of first ten natural numbers?**
**TASK#6: Write a program that adds numbers until user enters zero?**
**TASK#7: Write a program that display the sum of the following series?**
**1,2,4,8,16,32,64**
**TASK#8: Program to print natural numbers in reverse from n to 1 using while loop.**

**TASK#9: Write a program to display the following format using while loop**



```
------------------
num          sum
------------------
1            1
2            3
3            6
4            10
5            15
6            21
7            28
8            36
9            45
10           55
------------------
```

**TASK#10: Write a program using do while that inputs a number from user and displays n Fibonacci terms. In Fibonacci sequence, sum of two successive terms gives the third term.**
**TASK#11: Write a program using do while that inputs a number from the user and displays its factorial. It asks the user whether he wants to calculate another factorial or not. If the user inputs 1, it again inputs number and calculates factorial. If user inputs 0, program terminates.**

## ➢ Break, Continue, go-to Statements

**C++ break Statement**

In C++, the break statement terminates the loop when it is encountered.

The syntax of the break statement is:

break;

**Example : break with for loop**

```
// program to print the value of i
#include <iostream>
using namespace std;
int main() {
   for (int i = 1; i <= 5; i++) {
      // break condition
      if (i == 3) {
         break;
      }
      cout << i << endl;
   }
return 0;

}
```

```
Output

1

2
```

In the above program, the for loop is used to print the value of i in each iteration. Here, notice the code:

```
if (i == 3) {

   break;

}
```

This means, when i is equal to 3, the break statement terminates the loop. Hence, the output doesn't include values greater than or equal to 3.

## Continue Statement

In computer programming, the continue statement is used to skip the current iteration of the loop and the control of the program goes to the next iteration.

The syntax of the continue statement is:

```
continue;
```

**Example: continue with for loop**

```
// program to print the value of i

#include <iostream>

using namespace std;


int main() {

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

      // condition to continue

      if (i == 3) {

         continue;

      }

      cout << i << endl;

   }
```

```
  return 0;
}
```

This means

- When i is equal to 3, the continue statement skips the current iteration and starts the next iteration
- Then, i becomes 4, and the condition is evaluated again.
- Hence, 4 and 5 are printed in the next two iterations.

## C++ Switch Statement

The switch statement allows us to execute a block of code among many alternatives.

The syntax of the switch statement in C++ is:

```
switch (expression)  {
  case constant1:
    // code to be executed if
    // expression is equal to constant1;
    break;

  case constant2:
    // code to be executed if
    // expression is equal to constant2;
    break;
    .
    .
    .
  default:
    // code to be executed if
    // expression doesn't match any constant
}
```

## C++ go-to Statement

In C++ programming, go-to statement is used for altering the normal sequence of program execution by transferring control to some other part of the program.

Syntax of go-to Statement

```
Go-to label;
... .. ...
... .. ...
... .. ...
label:
statement;
```

## Lab Tasks

**Task 1: Write a program to calculate positive numbers till 50 only if the user enters a negative number, that number is skipped from the calculation.**

**Task 2: Calculate the average of  positive numbers entered by user, and ignores the negative number.**