# More on Alternative Selection:  Switch Statement.

**Videos:**

Watch the two following Videos:

http://www.youtube.com/watch?v=_CelY_ZBXb4

http://www.youtube.com/watch?v=kmDiNFfV-9k

**Explanation and Code examples:**

The switch Statement in C++ allows selection among multiple sections of code, depending on the value of an integral expression.

**Syntax:**

```
Switch ( expression ) {
        case constant-expression : statement     // two or more case statements
         [default   : statement]                  // optional
}
```

Remarks

The expression must be of an integer type (int or char)  or of a class type for which there is an unambiguous conversion to integral type. Integral promotion is performed as described in Integral Promotions.

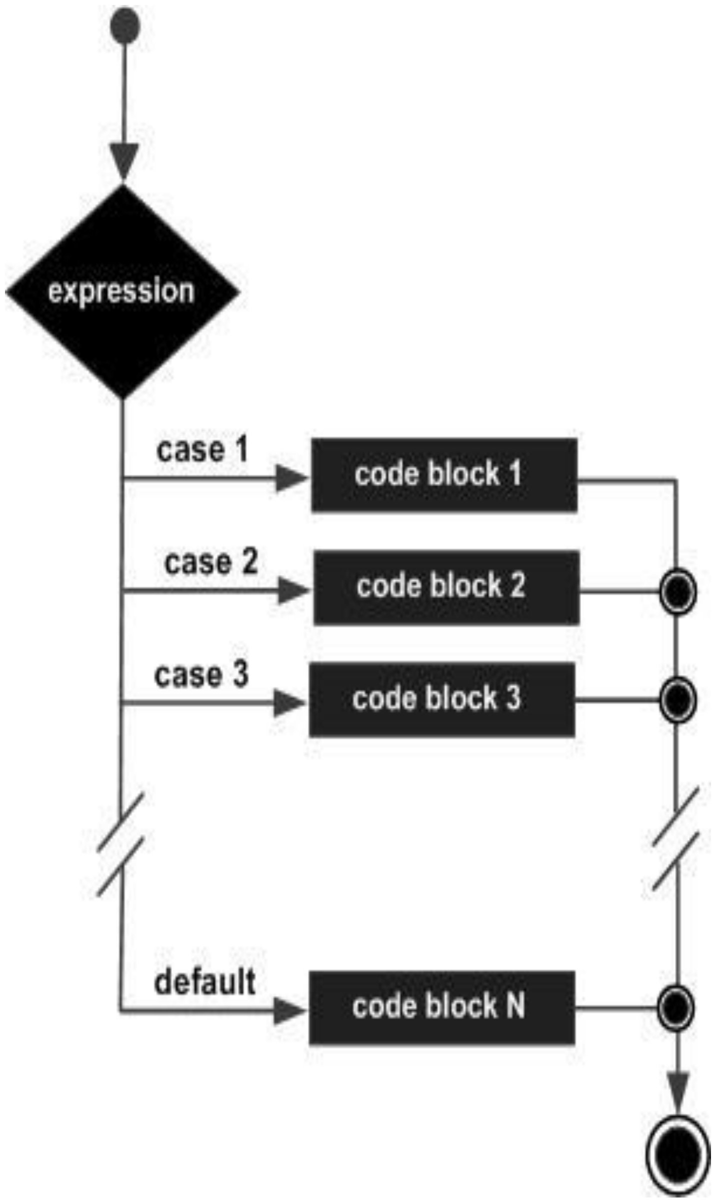
The switch statement body consists of:

- Cases
- An optional default label.

No two constant expressions in case statements can evaluate to the same value. The default label can appear only once. The labeled statements are not syntactic requirements, but the switch statement is meaningless without them. The default statement need not come at the end; it can appear anywhere in the body of the switch statement. A case or default label can only appear inside a switch statement.

The constant-expression in each case label is converted to the type of expression and compared with expression for equality.  Control passes to the statement whose case constant-expression matches the value of expression.

If a matching expression is found, control is not impeded by subsequent case or default labels. The break statement is used to stop execution and transfer control to the statement after the switch statement. Without a break statement, every statement from the matched case label to the end of the switch, including the default, is executed.



For example: **USING BREAKS**

```
// switch_statement1.cpp
#include <stdio.h>

int main() {
  char *buffer = "Any character stream";
  int aCap, aLetter, nota;
  char c;
  aCap = aLetter = nota = 0;

  while ( c = *buffer++ )   // Walks buffer until NULL
  {
    switch ( c )
    {
      case 'A':
        aCap++;
        break;
      case 'a':
        aLetter++;
        break;
      default:
        nota++;
    }
  }
  printf_s( "\nUppercase a: %d\nLowercase a: %d\nTotal:
%d\n",
    aCap, aLetter, (aCap + aLetter + nota) );
}
```

In the above example, aCap is incremented if c is an uppercase A. The break statement after aCap++ terminates execution of the switch statement body and control passes to the while loop. Without the break statement, aLetter and nota would also be incremented. A similar purpose is served by the break statement for case 'a'. If c is a lowercase a, aLetter is incremented and the break statement terminates the switch statement body. If c is not an a or A, the default statement is executed.

An inner block of a switch statement can contain definitions with initializations as long as they are reachable — that is, not bypassed by all possible execution paths. Names introduced using these declarations have local scope. For example:

**KEY concept about Switch:**

It is not the same as the ‘if’ or ‘if else’.

IT is a SWITCH..   (OFF/ON)

By default it is OFF
When it gets set to ON… it stays ON.

If there are no BREAK statements.. it will execute ALL cases After the FIRST case that is true..
(Because the Switch is now set to ON)

```
// switch_statement2.cpp    // NO BREAK
// C2360 expected
#include <iostream>
using namespace std;
int main( )
{
  switch( aChar )
  {
  case 'a':
     cout << aChar << endl;

  case 'b':
     cout << aChar << endl;

  case 'e':
     cout << aChar << endl;

  default:
     cout << aChar << endl;
  }
}
```

```
If aChar = ‘f’.. results are    f
If aChar = ‘e’.. results are   e
If aChar = ‘b’.. results are   b e
If aChar = ‘a’ .. results are , a b e
```