The Case Control Structure

- The control statement that allows us to make a decision from the number of choices is called a **switch**, or more correctly a **switch-case-default**, since these three keywords go together to make up the control statement.
- They most often appear as follows:

```
switch ( integer expression )
{
  case constant 1 :
  do this ;
  case constant 2 :
  do this ;
  case constant 3 :
  do this ;
  default :
  do this ;
 }
```

Note: 1. when we run a program containing a **switch**, first, the integer expression following the keyword **switch** is evaluated.
2. The value it gives is then matched, one by one, against the constant values that follow the **case** statements.
3. When a match is found, the program executes the statements following that **case**, and all subsequent **case** and **default** statements as well.
4. If no match is found with any of the **case** statements, only the statements following the **default** are executed.

Note: To stop the execution of all the subsequent **cases (**after the case where a match is found) **a**nd the **default** use a **break** statement (except after the **default**, since the control comes out of the **switch** anyway).

Example-
```
void main( )
{
 int num;
 printf("Enter either 1 or 2");
 scanf("%d", &num);
 switch ( num )
 {
   case 1 : printf ( "You have entered 1 \n" ) ;
            break ;
   case  2 : printf ( " You have entered 2 \n" ) ;
             break ;
   default : printf ( " \nWrong input!!!" ) ;
 }
}
```

❖ **Remember –**

1. You are also allowed to use **char** values in **case** and **switch.** When we use these, they are actually replaced by the ASCII values of these character constants.

2. The multiple statements in each **case** should not be enclosed within a pair of braces (unlike **if**, and **else**) since once a **case** is satisfied the control simply falls through the **case** till it doesn't encounter a **break** statement.

3. Every statement in a **switch** must belong to some **case** or the other. If a statement doesn't belong to any **case** the compiler won't report an error. However, the statement would never get executed.

4. If we have no **default** case, then the program simply falls through the entire **switch** and continues with the next instruction (if any,) that follows the closing brace of **switch**.

5. We can check the value of any expression in a **switch**. Thus the following **switch** statements are legal-

```
switch ( i + j - k )
switch ( 2 + 48 % 8 * k )
switch ( a < 10 && b > 10)
```

6. The **break** statement when used in a **switch** takes the control outside the **switch**. However, use of **continue** will not take the control to the beginning of **switch** as one is likely to believe.
7. In principle, a **switch** may be nested, but in practice it is rarely done.
8. The **switch** statement is very useful while writing menu driven programs.
9. An expression can also be used in **case** provide it is constant expression. Thus case 3+7 is correct but case a+b is incorrect. That is, all that we can have after the **case** is an **int** constant or a **char** constant or an expression that evaluates to one of these constants. Even a **float** is not allowed.
10. Multiple cases cannot use the same expression. Thus the following is illegal-

```
switch(a)
 {
    Case 4: printf("Fun");
            break;
    Case 2+2: printf("wrong!!!");
              break;
 }
```

❖ ***Switch* Versus *if-else* Ladder-**

1. A float expression cannot be tested using a **switch** which can be done with **if-else**.
2. **cases** can never have variable expressions (for example it is wrong to say **case a +3 :**) but a if-else can have the same.
3. **switch** works faster than an equivalent **if-else** ladder.

(Note: This is because the compiler generates a jump table for a **switch** during compilation. As a result, during execution it simply refers the jump table to decide which case should be executed, rather than actually checking which case is satisfied. As against this, **if-else**s are slower because they are evaluated at execution time.
A **switch** with 10 cases would work faster than an equivalent **if-else** ladder. Also, a **switch** with 2 **case**s would work slower than **if-else** ladder. If the 10^{th} **case** is satisfied then jump table would be referred and statements for the 10^{th} **case** would be executed. As against this, in an **if-else** ladder 10 conditions would be evaluated at execution time, which makes it slow. A lookup in the jump table is faster than evaluation of a condition, especially if the condition is complex.)