Tutorial Link https://codequotient.com/tutorials/Switch statement/5a22a1b6c66cfe38f296225c

TUTORIAL

Switch statement

Chapter

1. Switch statement

Sometimes multiple conditions are so intuitive that writing if-else ladder becomes much complicated in code. C also has another construct called switch statement. It successively tests the value of an expression against a list of integer or character constants. When a match is found, the statements associated with that constant are executed. The general form of the switch statement is:

```
switch (expression)
{

  case constant1:
    statement sequence
    break;

  case constant2:
    statement sequence
    break;

  case constant3:
    statement sequence
    break;
    .
    .
    .
  default:
```

```
    statement sequence
}
```

The expression must evaluate to an integer type. Thus, you can use character or integer values, but floating-point expressions, for example, are not allowed. The value of expression is tested against the values, one after another, of the constants specified in the case statements. When a match is found, the statement sequence associated with that case is executed until the break statement or the end of the switch statement is reached. The default statement is executed if no matches are found.

The default is optional, and if it is not present, no action takes place if all matches fail. For example, following program will check for a number and print corresponding weekday:

```c
#include <stdio.h>

int main()
{
  int day = 2;

  switch (day)
  {
    case 1:
      printf("Monday. \n");
      break;
    case 2:
      printf("Tuesday. \n");
      break;
    case 3:
      printf("Wednesday. \n");
      break;
    case 4:
      printf("Thursday. \n");
```

```
      break;
    case 5:
      printf("Friday. \n");
      break;
    case 6:
      printf("Saturday. \n");
      break;
    case 7:
      printf("Sunday. \n");
      break;
    default:
      printf("Wrong number entered. \n");
  }

  return 0;
}

```

The break statement is one of C's jump statements. You can use it in loops as well as in the switch statement (see the section "Iteration Statements"). When break is encountered in a switch, program execution "jumps" to the line of code following the switch statement.

There are some important things to know about the switch statement:

- The switch differs from the if in that switch can only test for equality, whereas if can evaluate any type of relational or logical expression.
- No two case constants in the same switch can have identical values. Of course, a switch statement enclosed by an outer switch may have case constants that are in common.
- If character constants are used in the switch statement, they are automatically converted to integers.
- Even if there are multiple statements to be executed in each case there is no need to enclose them within a pair of braces (unlike if, and else).

The break statements inside the case is optional, it make the control fall-through. In other words, what ever case starts execution, all the statements after that case will be executed will the end of switch block. Hence we place the break statement to separate the case from each other. For example,

```c
#include <stdio.h>

int main()
{
  int day = 2;

  switch (day)
  {
    case 1:
      printf("Monday. \n");
      break;
    case 2:
      printf("Tuesday. \n");
    case 3:
      printf("Wednesday. \n");
    case 4:
      printf("Thursday. \n");
      break;
    case 5:
      printf("Friday. \n");
      break;
    case 6:
      printf("Saturday. \n");
      break;
    case 7:
      printf("Sunday. \n");
      break;
    default:
      printf("Wrong number entered. \n");
  }

```

```
  return 0;
}

```

The above program will print

```
Tuesday
Wednesday
Thursday
```

As the number is 2, which match with second case and from there onwards it execute all statements till the break statement. Switch blocks can be nested just like the nesting of if blocks. Another thing in switch is, we can club some case constants if they needs to be handled in same way, for example, following program will test the character for vowel or not: -

```c
#include <stdio.h>

int main()
{
  char ch = 'u';
  switch (ch)
  {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
      printf("ch is a small case vowel \n");
      break;
    case 'A':
    case 'E':
    case 'I':
    case 'O':
```

```
    case 'U':
      printf("ch is a UPPER case vowel \n");
      break;
    default:
      printf("ch is not a vowel. \n");
  }
  return 0;
}

```

So all first 5 case needs same execution, so they can be clubbed as above.

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. For example, in the following program the printf() never goes to work: -

```c
#include <stdio.h>

int main()
{
  int i;
  i = 2;
  switch (i)
  {
      printf ("Hello") ;        // No syntax error, but never executes.
    case 1:
      printf("Case 1. \n");
      break ;
    case 2 :
      printf("Case 2. \n");
      break ;
```

```
16      }
17  }
18
```