C - switch statement

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 **switch case**.

Syntax

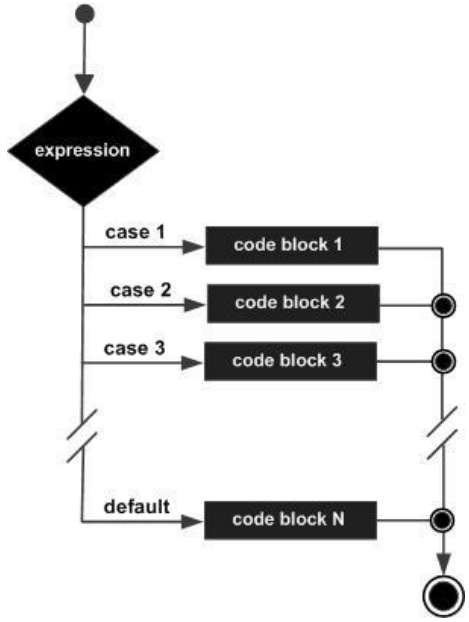
The syntax for a **switch** statement in C programming language is as follows −

```
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);
}
```

The following rules apply to a **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



Example

Let's try to understand it by the examples. We are assuming that there are following variables.

1. int x,y,z;
2. char a,b;
3. float f;

Valid Switch	Invalid Switch	Valid Case	Invalid Case
switch(x)	switch(f)	case 3;	case 2.5;
switch(x>y)	switch(x+2.5)	case 'a';	case x;
switch(a+b-2)		case 1+2;	case x+2;
switch(func(x,y))		case 'x'>'y';	case 1,2,3;

```
#include <stdio.h>

int main () {

   /* local variable definition */
   char grade = 'B';

   switch(grade) {
      case 'A' :
         printf("Excellent!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("Well done\n" );
         break;
      case 'D' :
```

```
         printf("You passed\n" );
         break;
      case 'F' :
         printf("Better try again\n" );
         break;
      default :
         printf("Invalid grade\n" );
   }

   printf("Your grade is  %c\n", grade );

   return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
Well done
Your grade is B
```

nested switch statements

It is possible to have a switch as a 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.

Syntax

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

```
switch(ch1) {

   case 'A':
      printf("This A is part of outer switch" );

      switch(ch2) {
         case 'A':
            printf("This A is part of inner switch" );
            break;
         case 'B': /* case code */
      }

      break;
   case 'B': /* case code */
}
```

Example

```
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 100;
   int b = 200;
```

```
   switch(a) {

      case 100:
         printf("This is part of outer switch\n", a );

         switch(b) {
            case 200:
               printf("This is part of inner switch\n", a );
         }
   }

   printf("Exact value of a is : %d\n", a );
   printf("Exact value of b is : %d\n", b );

   return 0;
}
```

When the above code is compiled and executed, it produces 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
```

Example-1:

```
1. #include <stdio.h>
2. int main()
3. {
4.     int x = 10, y = 5;
5.     switch(x>y && x+y>0)
6.     {
7.        case 1:
8.        printf("hi");
9.        break;
10.       case 0:
11.       printf("bye");
12.       break;
13.       default:
14.       printf(" Hello bye ");
15.    }
16.
17. }
```

Output

```
hi
```

```
Example-2
#include <stdio.h>
```

```
int main() {
        int ID = 500;
        int password = 000;
        printf("Plese Enter Your ID:\n ");
        scanf("%d", & ID);
        switch (ID) {
            case 500:
                printf("Enter your password:\n ");
                scanf("%d", & password);
                switch (password) {
                    case 000:
                        printf("Welcome Dear Programmer\n");
                        break;
                    default:
                        printf("incorrect password");
                        break;
                }
                break;
            default:
                printf("incorrect ID");
                break;
        }
}
```

OUTPUT:

```
Plese Enter Your ID:
 500
Enter your password:
 000
Welcome Dear Programmer
```

Example-3: Program to create a simple calculator

```
#include <stdio.h>
  int main() {
  char operator;
  double n1, n2;
      printf("Enter an operator (+, -, *, /): ");
      scanf("%c", &operator);
      printf("Enter two operands: ");
      scanf("%lf %lf",&n1, &n2);
      switch(operator)
      {
          case '+':
              printf("%.1lf + %.1lf = %.1lf",n1, n2, n1+n2);
              break;
          case '-':
              printf("%.1lf - %.1lf = %.1lf",n1, n2, n1-n2);
              break;
          case '*':
              printf("%.1lf * %.1lf = %.1lf",n1, n2, n1*n2);
              break;
          case '/':
              printf("%.1lf / %.1lf = %.1lf",n1, n2, n1/n2);
              break;
```

```
            // operator doesn't match any case constant +, -, *, /
            default:
                printf("Error! operator is not correct");
        }
        return 0;
    }
```

Output

```
Enter an operator (+, -, *,): -
Enter two operands: 32.5
12.4
32.5 - 12.4 = 20.1
```

Example-4: program to check whether number is EVEN or ODD using switch.

```
#include <stdio.h>

int main()
{
    int number;

    printf("Enter a positive integer number: ");
    scanf("%d",&number);

    switch(number%2) //this will return either 0 or 1
    {
        case 0:
            printf("%d is an EVEN number.\n",number);
            break;
        case 1:
            printf("%d is an ODD number.\n",number);
            break;
    }

    return 0;
}
```

Output

```
First run:
Enter a positive integer number: 10
10 is an EVEN number.

Second run:
Enter a positive integer number: 11
11 is an ODD number.
```

Example-5: Program to find number of days in a month using C

```
#include <stdio.h>

int main()
{
        int month;
        int days;
```

```
        printf("Enter month: ");
        scanf("%d",&month);

        switch(month)
        {
                case  4:
                case  6:
                case  9:
                case 11:
                        days=30;
                        break;
                case  1:
                case  3:
                case  5:
                case  7:
                case  8:
                case 10:
                case 12:
                        days=31;
                        break;

                case 2:
                        days=28;
                        break;

                default:
                        days=0;
                        break;
        }

        if(days)
                printf("Number of days in %d month is: %d\n",month,days);
        else
                printf("You have entered an invalid month!!!\n");

        return 0;

}
```

Output

```
First run:
Enter month: 3
Number of days in 3 month is: 31

Second run:
Enter month: 2
Number of days in 2 month is: 28

Third run:
Enter month: 11
Number of days in 11 month is: 30

Fourth run:
Enter month: 13
You have entered an invalid month!!!
```

Conditional Operator

We have covered **conditional operator ? :** in the previous chapter which can be used to replace **if...else** statements. It has the following general form −

```
Exp1 ? Exp2 : Exp3;
```

Where Exp1, Exp2, and Exp3 are expressions. Notice the use and placement of the colon.

The value of a ? expression is determined like this −

- Exp1 is evaluated. If it is true, then Exp2 is evaluated and becomes the value of the entire ? expression.
- If Exp1 is false, then Exp3 is evaluated and its value becomes the value of the expression.

```
Example-1: program to find largest among two numbers using ternary operator
#include <stdio.h>

int main()
{
    // variable declaration
    int n1 = 5, n2 = 10, max;

    // Largest among n1 and n2
    max = (n1 > n2) ? n1 : n2;

    // Print the largest number
    printf("Largest number between"
           " %d and %d is %d. ",
           n1, n2, max);

    return 0;
}
```

Output:

```
Largest number between 5 and 10 is 10.
```

Example-2: