Decision making using switch-case-default

Many times in our daily lives, we face conditions where we are required to choose between a number of alternatives rather than just two or three. For example, which school to go to, which food to have in a restaurant, which game to play, etc. Similarly, in programming languages, we sometimes face problems where we have to make the program user-friendly by giving them more than two alternatives to choose from rather than just one or two.

In such cases, it becomes a convoluted problem if we use a series of if-else statements. Therefore, C provides us a discrete control statement which is **"switch"** to handle such issues effectively. Let us learn **how to use a switch, case and default keywords**?

The general form of the three keywords is:

```
switch (integral_expression)
{
      case constant_1:
      code;
      [break;]
      case constant_2:
      code;
      [break;]

      .
      .
      .

      case constant_n:
      code;
      [break;]

      default:
      code;
}
```

Points to be noted:

1. The integer expression after the `switch` keyword is any valid C statement that yields an integer value. Example, integer constants like `1`, `2`, `100` etc.

2. The values of `constant_1`, `constant_2` after the `case` keyword can be an integer or character. But all these constants must be different from each other.
3. The `code` mentioned above is the code that we want to execute. It can be anything ranging from `printf` to another switch-case ladder.
4. Now, when we run our program, what happens first is the integer expression gets evaluated.
5. The control then goes inside the switch and the value received from the integer expression is compared with the case constants.
6. If the match is found with any case constant, that particular case will be executed along with the following case and default statements.
7. If the match is not found, only the statements after the `default` get executed.

Example:

```c
#include <stdio.h>

int main( )
{
	int i = 2 ;

	switch ( i )
	{
		case 1:
		printf ( "1 \n" ) ;

		case 2:
		printf ( "2 \n" ) ;

		case 3:
		printf ( "3 \n" ) ;

		default:
		printf ( "No match \n" ) ;
	}

	return 0;
}
```

Output

```
2
3
```

```
No match
```

In the program above, most of us expected the output to be only `2` since the value of constant `i` is `2`. But that does not happen since all the `case` statements and the default gets executed after the match is found, as mentioned earlier.

To prevent this from happening, we use another statement called the break statement to get the output from that particular case only. Note that `break` need not be written after the default statement since the control comes out of the switch loop anyway.

Example:

```c
#include <stdio.h>

int main( )
{
	int i = 2 ;

	switch ( i )
	{
		case 1:
		printf ( "1 \n" ) ;
		break;

		case 2:
		printf ( "2 \n" ) ;
		break;

		case 3:
		printf ( "3 \n" ) ;
		break;

		default:
		printf ( "No match \n" ) ;
	}

	return 0;
}
```

Output

```
2
```

More about switch (some useful points and some disadvantages)

1) The above program need not be made in an ascending order only. It can be made in other order.

Example:

```c
#include<stdio.h>

int main( )
{
	int i = 2 ;

	switch ( i )
	{
		case 34 :
			printf ( "1 \n" ) ;
			break;

		case 2 :
			printf ( "2 \n" ) ;
			break;

		case 121 :
			printf ( "3 \n" ) ;
			break;

		default :
		printf ( "No match \n" ) ;
	}

	return 0;
}
```

Output

```
    2
```

2) We can also use "character values" in "case and switch".

Example:

```c
#include<stdio.h>

int main()
{

	char ch = 's';

	switch (ch)
	{
		case 'a':
			printf("The letter is 'a'");
			break;

		case 'b':
			printf("The letter is 'b'");
			break;

		case 's':
			printf("The letter is 's'");
			break;

		default:
		printf("No match");
	}

	return 0;
}
```

Output

```
    The letter is 's'
```

The characters are in reality replaced by their ASCII values by the compiler to make them act like integer constants.

3) If we want to write multiple statements within a particular case, we need not enclose them within a pair of braces.

4) If there is a statement inside the switch statement but it does not belong to any of the cases then that particular statement will not be executed or in other words will be skipped by the compiler.

5) It is not compulsory to add the default statement at the end of the switch. Even if we don't write it, the program would run just the same.

6) Nested switches exist in reality but rarely used. The switch statements are mostly used for writing menu driven programs.

7) Many times, we want to execute the same set of statements under different cases. This can be done as shown below.

Example:

```
#include <stdio.h>

int main()
{

	char ch;

	printf("Enter alphabets a, b or c:\n");
	scanf("%c",&ch);

	switch(ch)
	{
		case 'a':
		case 'A':
			printf("The letter is 'a'");
			break;

		case 'b':
		case 'B':
			printf("The letter is 'b'");
			break;
```

```
            case 'c':
            case 'C':
                  printf("The letter is 's'");
                  break;

            default:
            printf("No match");
      }

      return 0;
}
```

Output

```
    Enter alphabets a, b or c: b
    The letter is 'b'
```

Here, what happens is that the cases keep executing until a `break` statement is found. Therefore, if for example if alphabet `a` is entered the case `'a'` is satisfied and because there are no statements after that, the control goes to the next `case` i.e. `case 'A'` and executes the statements underneath that.

8) The switch statement is often compared with the `if statement`. It is better to use the `switch` in many cases due to the advantages listed above but also, the disadvantage of the switch is that we can't have a case which contains conditionals like: `case i > 10:`

Switch Statements (features, disadvantages and difference with if else)

switch : A compound statement

This statement is used when we have to select one choice from multiple choices on the basis of the value of some variable.

Syntax for switch statement:

```
    switch(expression/condition)
    {
          case 1:
                statements;
                [break;]
          case 2:
                statements;
                [break;]
          default:
                statements;
    }
```

Here, `expression` can be anything that returns integer value or character value but not a floating value.

Example:

```c
#include <stdio.h>

int main()
{
      int a = 2 ;

      switch(a)
      {
            case 1:
                  printf("abc");
                  break;

            case 2:
                  printf("xyz");
                  break;

            case 3:
                  printf("pqr");
                  break;

            default:
                  printf("stu");
      }

      return 0;
```

```
}
```

Output

```
xyz
```

Here, `a` is an expression which returning an integer value since it's value is 2 `case 2` will be executed. Note that as we have used `break` statement here only `case 2` will be executed and after that control will come out of the closing brace of the switch.

Note: Also keep in mind that `switch`, `case`, `default` are all keywords.

Features of switch

- To execute multiple statements, in any case, there is no requirement of braces as in if else.
- The default may or may not use the `break` statement.
- We can use case number in ascending or descending order or in any random order.
- In both `switch` and `case`, we use `int` or `char` constant.
- We can use `default` anywhere in the program because every time `default` is executed only when there is no match of any case.
- We can execute a common set of statement for multiple cases.
- If there is any statement in `switch` then it must be in any case.

Executing a common set of statements for multiple cases

```c
#include <stdio.h>

int main()
{
	int a = 2 ;

	switch(a)
	{
		case 1:
		case 2:
			printf("xyz");
			break;

		case 3:
			printf("pqr");
			break;

		default:
			printf("stu");
	}
	return 0;
}
```

Output

```
xyz
```

Here, if even case 1 is encountered xyz is printed.

Disadvantages of switch statements

1. float constant cannot be used in the switch as well as in the case.
2. You can not use the variable expression in case.
3. You cannot use the same constant in two different cases.
4. We cannot use the relational expression in case.

Difference between switch case and if-else

Sr No	The switch case	The if – else
1	In case of a switch, we create jump table on compile time only selected case is executed on runtime.	In this case, we do not create a jump table and all cases are executed at runtime.
2	If a program is large we use the switch statement.	If a program is large then program structure will be more complex.
3	If a program is large, then switch case gives the more structured program.	If a program is small we use if else.

In C Programming Language, ladder/multiple if can be replaced by the **switch case statement**, if value to be tested is **integral type**.

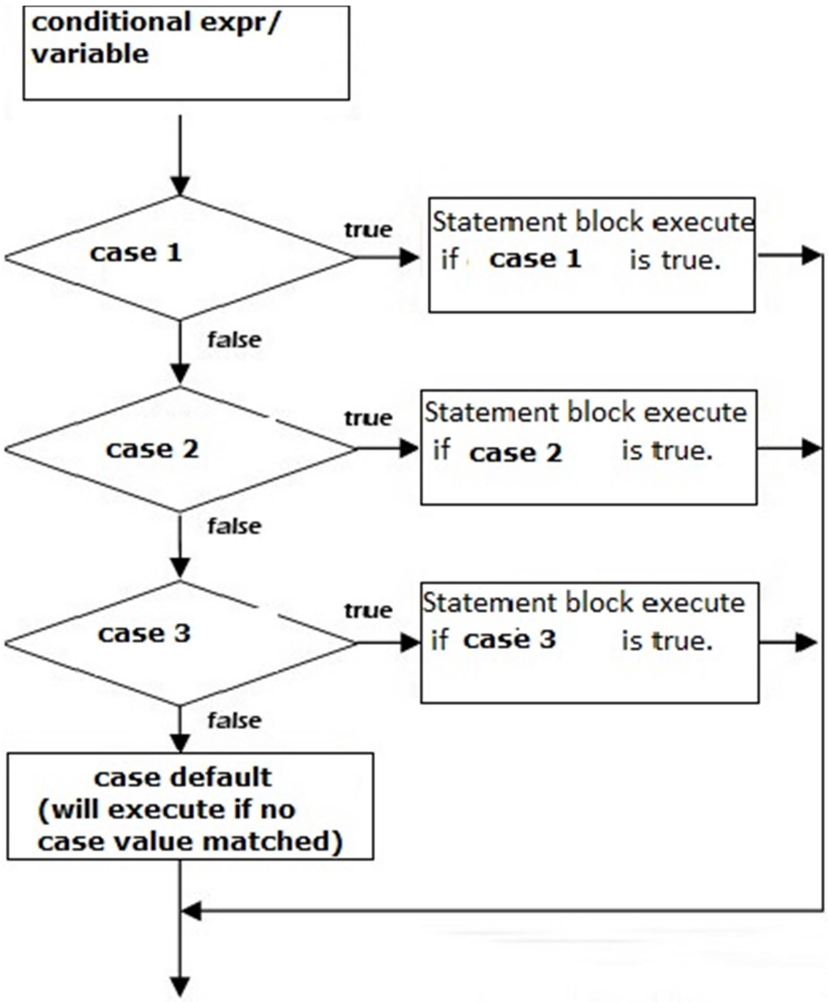
Switch statement is used to check a conditional expression or variable with the given multiple choices of integral types. These integral values are called case values.

In switch statement these is one condition (variable), which is checked with its multiple case values. This statement is very useful in programming language.

Remember following points:

- There may be multiple values to be checked.
- Program's control moves to the matched **case value**, if there is no **case value** found it moves to the **default case**.
- **case value** must be **integral type literal**, variable can not be use as a **case value**.
- **default case** statement is an option, but it should be use.
- **default case** can be placed anywhere in the switch body, but should be use at the end of the switch body.

- Multiple **case value** can have single body i.e. you can check more than one case values for single set of statements.
- Do not forget to put **break** after the case body.
- **switch statement** has **fall-down property**, if **break** is not found, program's execution moves to the next case body whether it is matched or not.



Syntax

```
switch(test-condition/variable)
{
      case case-value:
            block1;
      [break];

      case case-value:
            block2;
```

```
        [break];

        case case-value:
                block3;
        [break];

        case case-value:
                block4;
        [break];

        default:
                block-default;
}
```

Here, `break` is an optional may be ignored if you are checking multiple case values for single block (set of statements).

Consider the examples:

```
/* program to print number from 0 to 5 using switch*/
#include <stdio.h>
int main()
{
        int num;
        printf("Enter an integer number: ");
        scanf("%d",&num);
        switch(num){
                case 0:
                        printf("Zero");
                        break;
                case 1:
                        printf("One");
                        break;
                case 2:
                        printf("Two");
                        break;
                case 3:
                        printf("Three");
                        break;
                case 4:
                        printf("Four");
                        break;
                case 5:
```

```
                printf("Five");
                break;
            default:
                printf("Invalid Number");
        }
        return 0;
}
```

Output

```
First run:
Enter an integer number: 0
Zero

Second run:
Enter an integer number: 4
Four

Third run:
Enter an integer number: 8
Invalid Number
```

Consider the example to check whether entered character is VOWEL or CONSONANT using switch statement having multiple case values without using break, because here we are checking multiple case values for single block.

```
/*program to check entered character is VOWEL or CONSONANT*/

#include < stdio.h >
int main()
{

        char ch;
        printf("Enter a character: ");
        scanf("%c",&ch);

        if( (ch>='A' && ch<='Z') || (ch>='a' && ch<='z') )
        {
            switch(ch)
            {
                case 'A':
                case 'a':
```

```
                    case 'E':
                    case 'e':
                    case 'I':
                    case 'i':
                    case 'O':
                    case 'o':
                    case 'U':
                    case 'u':
                          printf("%c is a VOWEL.",ch);
                          break;
                    default:
                          printf("%c is a CONSONANT.",ch);
              }
       }
       else
             printf("Entered character is INVALID.");
       return 0;
}
```

Output

```
First run:
Enter a character: I
I is a VOWEL.

Second run:
Enter a character: t
I is a CONSONANT.

Third run:
Enter a character: 9
Entered character is INVALID.
```