Decision Making and Branching

A program is nothing but the execution of one or more instructions sequentially in the order in which they come into sight. Quite often, it is desirable in a program to alter the sequence of the statements depending upon certain circumstances. In real time applications there are a number of situations where one has to change the order of the execution of statements based on the conditions.

Decision making statements in a programming language help programmer to transfer the control from one part to other parts of the program. Thus, these decision making statements facilitate the programmer in determining the flow of control. The decision making statements checks the given condition and then executes its sub-block or set of statements, based on success or failure of the condition.

C programming also provide decision making constructs and are if-statement, switch statement, Conditional operator, and goto statement. Here, in this section we shall discuss about if- statement and switch. if statement can be implemented in different forms and are:

i) Simple if statement.

ii) if-else statement.

iii) Nested if-else statement.

iv) else if ladder.

simple if statement:

The general form or syntax of a simple if statement is

```
if (expression/condition)
{
  // statements-block i.e, statements to be executed expression1 is true
}
next-statements;        //i.e, statements after the if block
```

The statements-block may be single statement or set of statements. If the expression/condition is true, the statement-block will be executed and followed by statements after the if block; otherwise the statement -block will be skipped and the execution will be jumped directly to the next-statements.

For example, c program to find the greater value, using if statement as

```
#include<stdio.h>
int main()
{
        int a=20;
        int b=10;
        if (a>b)
        {
                printf("a is greater ");
        }
        printf(“value of a is %d\n ”,a);
        printf(“value of b is %d\n ”,b);
}
```

output: a is greater

In this example, a is greater than b condition is checked, it is true then printf inside the if-block is executed and followed by the statements after it. Consider, in case If it false then only the printf statements after if- block is executed.

if-else statement:

The syntax or general for of if-else statement is given below as:

```
if (expression)
        {
                // statements to be executed when expression is true
        }
        else
         {
                //statements to be executed when expression is false
```

```
}
next-statements;        //i.e, statements after the if -else block
```

If the expression/condition is true, then statements inside the 1st bock will be executed and followed by statements after the if block; otherwise the statements inside the 2nd bock will be executed, followed by statements after the if block.

For example, c program to find the greatest value, using if else statement as

```
#include<stdio.h>
    int main()
    {
        int a=20;
        int b=30;
            if(a>b)
            {
                printf("a is greater %d",a);
            }
            else
            {
                printf("b is greater %f",b);
            }
}
```

output: b is greater

In this example, a is greater than b condition is checked, it fails then it does not execute the first block of statements. It executes the else block and printf inside the else block is executed and followed by the statements after if-else .

Nested if-else statement:

The syntax or general for of nested if-else statement is given below as:

```
if(expression)
    {
```

```
        //statements to be executed when expression is true
        if(expression1)
        {
                // statements to be executed when expression1 is true
        }
        else
        {
                // statements to be executed when expression1 is false
        }
}
else
{
        //statements to be executed when expression is false
        if(expression2)
        {
                // statements to be executed when expression2 is true
        }
        else
        {
                // statements to be executed when expression2 is false
        }
}
next-statements;        //i.e, statements after the nested if-else block
```

Nested if else statement is same like if else statement, where new block of if else statement is defined in existing if or else block statement.

For example, c program to find the greater value, using nested if else statement as

```
#include<stdio.h>
int main()
{
    int a,b,c;
    printf("Enter 3 values\n");
    scanf("%d%d%d",&a,&b,&c);
    printf("\nLargest value is");
    if(a>b)
    {
        if(a>c)
        {
            printf("a is greater %d\n",a);
        }
        else
        {
            printf("b is greater %d\n",c);
        }
    }
    else
    {
        if(c>b)
        {
            printf("c is greater %d\n",c);
        }
```

```
                else
                {
                        printf("b is greater %d\n",b);
                }
        }
}
```

else if ladder:

The syntax or general form of else if ladder statement is given below as:

```
if (expression1)
{
        // statements to be executed when expression1 is true
}
else if(expression2)
{
        // statements to be executed when expression1 is false and expression2 is true
}
else if (expression 3)
{
        // statements to be executed when expression1 and expression2 is false and expression3 is true
}…...
else
{
        // statements to be executed when all expressions are false
}
next-statements;        //i.e, statements after the else -if blocks
```

Useful when you need to check several conditions within the program, nesting of if-else blocks can be avoided using else-if. Last else part is handled when condition fails.

For example, c program to find the highest student marks, using nested else if ladder statement as

```
#include<stdio.h>
int main()
{
        int marks;
        printf("Enter student marks\n");
        scanf("%d",&marks);
        if(marks<=39)
                printf("Grade : F\n");
        else if(marks<=49)
                printf("Grade : E\n");
        else if(marks<=59)
                printf("Grade : D\n");
        else if(marks<=69)
                printf("Grade : C\n");
        else if(marks<=79)
                printf("Grade : B\n");
        else if(marks<=89)
                printf("Grade : A\n");
        else
                printf("Grade : O\n");
}
```

switch statement:

When one of the many alternatives is to be selected, we use an if -else statement to control the selection. However, the complexity of such a program increases dramatically when the number of alternatives increases. The program becomes difficult to read and follow. Instead of else -if ladder, C has a built-in multi way decision statement known as switch.

The switch statement tests the expression against a list of case values (constant-expression) and when a match is found, set of statements associated with that case is executed. The general form of the switch statement is as shown below:

Syntax of switch:

```
switch(expression)
{
  case constant-expression:
        statement(s);
        break;       // Optional statement

  case constant-expression:
        statement(s);
        break;       // Optional statement
 ....
       // any number of case statements

  default:           // Optional case
       statement(s);
}
statements-x;  //i.e, statements followed by switch
```

The expression is an integer expression or characters. Constant-expression (evaluable to integral constant) are known as case-labels (or also called as case-value). Case labels cannot be floating point value and double value. Each of the cases may contain zero or more statements and there is no need to put braces around these statements. Note that case labels end with a colon (:).

When the switch is executed, the expression is successfully compared against the case labels, if the value of the expression matches with a case, then the set of statements follows the case are executed.

The break statement at the end of each block is optional and it indicates the end of a particular case and causes an exit from switch statement, transferring the control to the statement-x following the switch.

The default case is an optional case. When present, it will be executed if the value of the expression does not match with any of the cases. If not present, no action takes place if all matches fail and control goes to the statements-x.

Variations of switch with examples:

```
int main()
{
    int i;
    printf("Enter the value of i\n");
    scanf("%d",&i);
    switch(i)
    {
        case 1:
            printf("case 1 is matched\n");
            break;
        case 2:
            printf("case 2 is matched\n");
            break;
        default:
            printf("No match found\n");
    }
}
```

When the value of i entered is 1, case 1 is matched and it executes the set of statement's follows the case i.e, printf("case 1 is matched\n"); and switch terminates after that.

When the value of i entered is 2, case 2 is matched and it executes the set of statement's follows the case i.e, printf("case 2 is matched\n"); and switch terminates after that.

When the value of i entered is other than 1 and 2, None of the case is matched. Now it executes the default case and execute those set of statement's after it. i.e, printf("No match found\n");

```
int main()
{
        char ch;
        printf("Enter the ch\n");
        scanf("%c",&ch);
        switch(ch)
        {
                case 'a':
                        printf("case 'a' is matched\n");
                        break;
                case 'b':
                        printf("case 'b' is matched\n");
                        break;
                default:
                        printf("No match found\n");
        }
}
```

When ch entered is 'a', case 'a' is matched and it executes the set of statements follows the case i.e, printf("case 'a' is matched\n"); and switch terminates after that.

When ch entered is 'b', case 'b' is matched and it executes the set of statements follows the case i.e, printf("case 'b' is matched\n"); and switch terminates after that.

When ch entered is other than 'a' and 'b', none of the case is matched. Now it executes the default case and execute those set of statement's after it. i.e, printf("No match found\n");

As entered value is character in this example, internally ASCII comparison with case is done.

Refer to the example programs for more variations on switch.

We can use one switch-case statement inside the other switch-case statement. The inner out and outer switch case constant may be same. The general form of the nested switch statement is as shown below:

Syntax of nested switch:

```
switch(expression)
{
    case constant-expression:
            statement(s);
            break;
    case constant-expression:
            statement(s);
            break;
    default:
            statement(s);          // or expression
        switch(expression)
        {
            case constant-expression:
                    statement(s);
                    break;
            case constant-expression:
                    statement(s);
                    break;
            default:
                statement(s);

        }
}
```