# Decision Making Control Structure

**(if, if….else, if…else if(ladder if…else or multiple if….else),nested if..else, switch, goto)**

Decision-making structures requires when the programmer specifies one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

There are many decision making structure are used that is discussed as follow.

## 1 Explain *if* with example and draw flowchart.

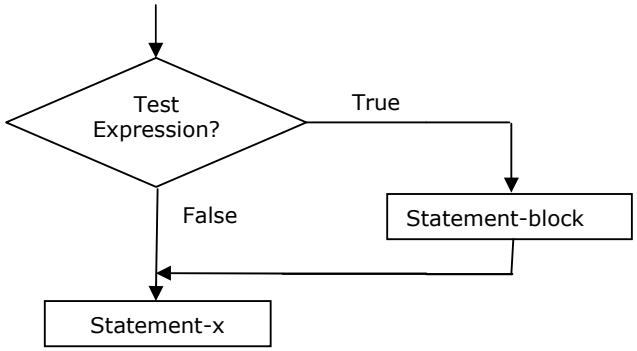
- *if* is used to control the flow of execution of statements.
- It is two way decision making statements.
- It evaluates expression first and based on its result, the control is transferred to the particular statement.
- The general form of simple *if* statement is or its syntax is:

```
if(test condition)
{
        Statement-block;
}

statement-x
```

- If the test expression is true, the statement-block will be executed; otherwise the statement- block will be skipped and the execution will jump to the statement-x.

Flowchart Strucutre:




Example:

```
#include<stdio.h>
#include<conio.h>

void main ()
{
        /* local variable definition */
        int a ;
        clrscr();
        printf("Enter the value of  a:");
        scanf("%d",&a);
         /* check the boolean condition using if statement */
        if( a < 20 )
        {
                /* if condition is true then print the following */
```

```
            printf("a is less than 20\n" );
        }
        printf("value of a is : %d\n", a);
        getch();
}
```

**2 Explain if…else… with example and draw flowchart.**

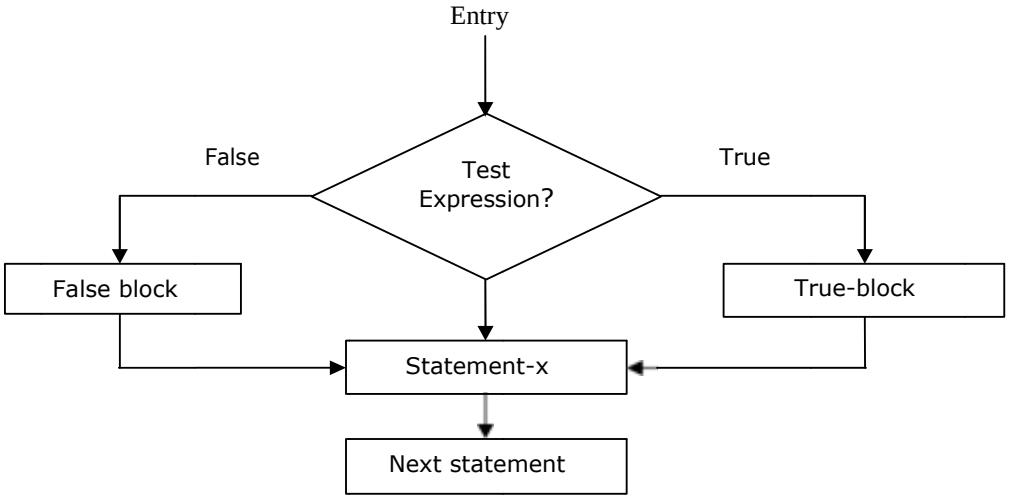
- If supports statements only for true part
- If…else supports statements for true part and false part. if statement is followed by else part which is executed only when condition is false.
- It is used for either or type situation when we need have only two option and among then we only one will be executed depending upon condition is true or false.

The general format is given below.

```
if (test-expression)
{
    True-block statement;
}
else
{
    False-block statement;
}
Statement x;
```

- If test-expressionis true, then true-block statement is executed otherwise false-block statement is executed.
- So when condition is checked at that time, either true block or false block will be executed.
- In both the cases, statement-xwill be executed immediately after that block.

- Flowchart Structure:



Example:

Check whether the given number is even or odd.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        pritnf("enter the value of n: ");
'       scanf("%d",&n);
        if(n%2==0)
        {
                printf("% d is even",n);
        }
        else
        {
                printf("%d is odd",n);
        }
        getch();
}
```

**3 Explain if…else if (ladder if….else or multiple if …..else) with example and draw flowchart.**

An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement. It is used when we test multiple condition and want that any one condition is true. If any single condition is true then its block of code will be executed and if all conditions are false then at last default else part will be executed. So we can say

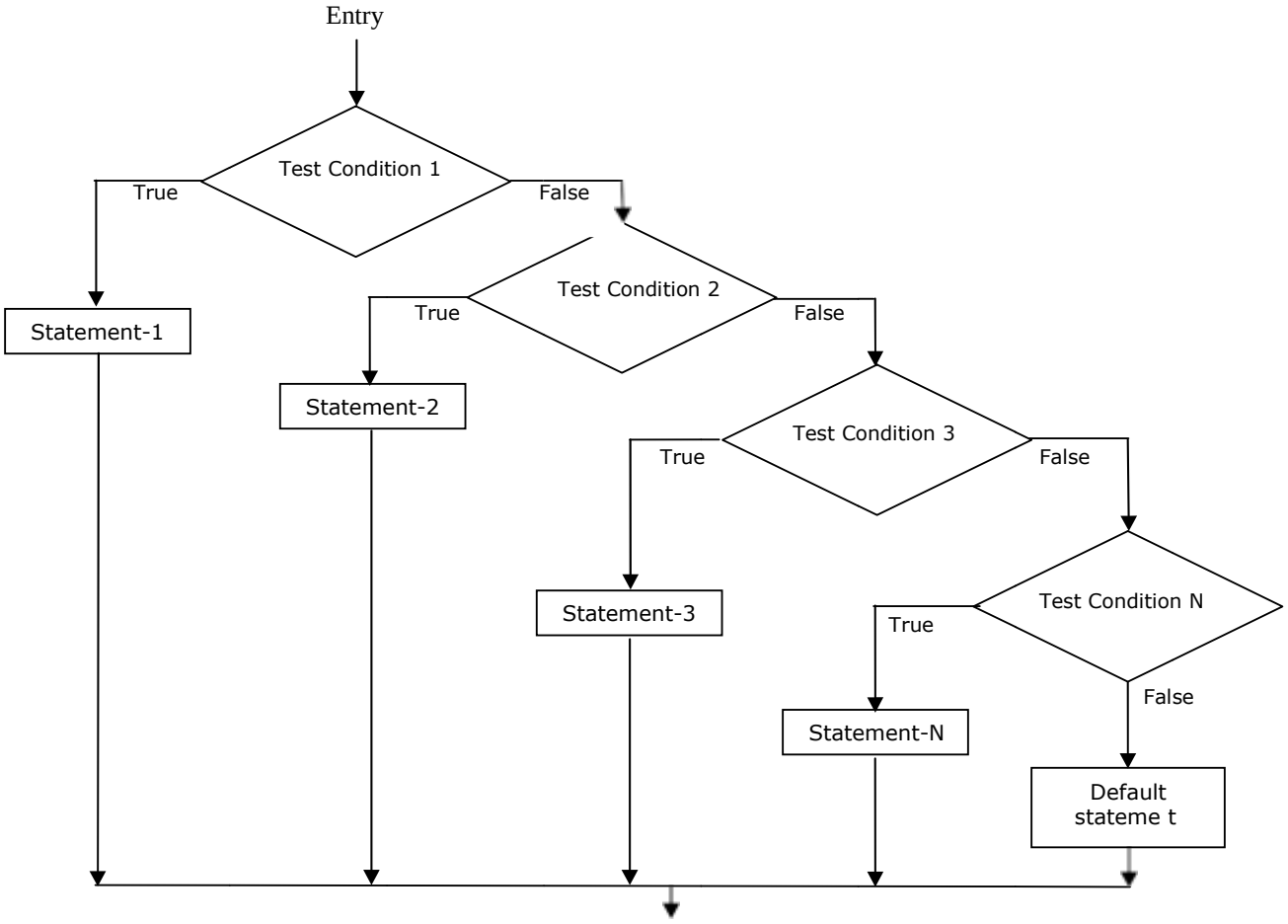
When multipath decisions are involved, we may use if…else…if ladder,
which takes the following general form.

```
if(condition-1)
        statement-1;
                else if(condition-2)
                        statement-2;
                                else if(condition-3)
                                        statement-3;
                                        ……..
                                        ……..
                                        ……..
                                        else if(condition –N)
                                                statement-3;
                                                        default—statement;

                                                statement-x
```

- First condition-1 is checked and if it is true, then statement-1 will be executed and control goes to statement-x.
- If condition-1 is false, then condition-2 is checked and if it is true then statement-2 will be executed and control goes to statement-x.

- If condition-2 is false, then condition-3 is checked and process repeats
- This process is repeated until either it finds one of the conditions is true or all the conditions are false.
- If all the conditions are false, then default-statement will be executed.
- Flowchart Structure:




Example: WAP to calculate avg of 5 subject and display result of student according to their average. Ex. If avg >= 60 display first class.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int m1,m2,m3,m4,m5;
        float avg;
        clrscr();
        printf("enter the marks of 5 different subject: ");
        scanf("%d %d %d %d %d",&m1,&m2,&m3,&m4,&m5);
        avg=(float) (m1+m2+m3+m4+m5)/5.0;
        printf("\naverage is %f= ",avg);
        if(avg>=60)
        {
                printf("first class");
        }
        else if (avg<60 && avg>=50)
                {
                        printf("second class");
                }
```

```
                else if(avg<50 && avg>=40)
                        {
                                printf("third class");
                        }
                        else
                        {
                                printf("fail");
                        }
        getch();
}
```

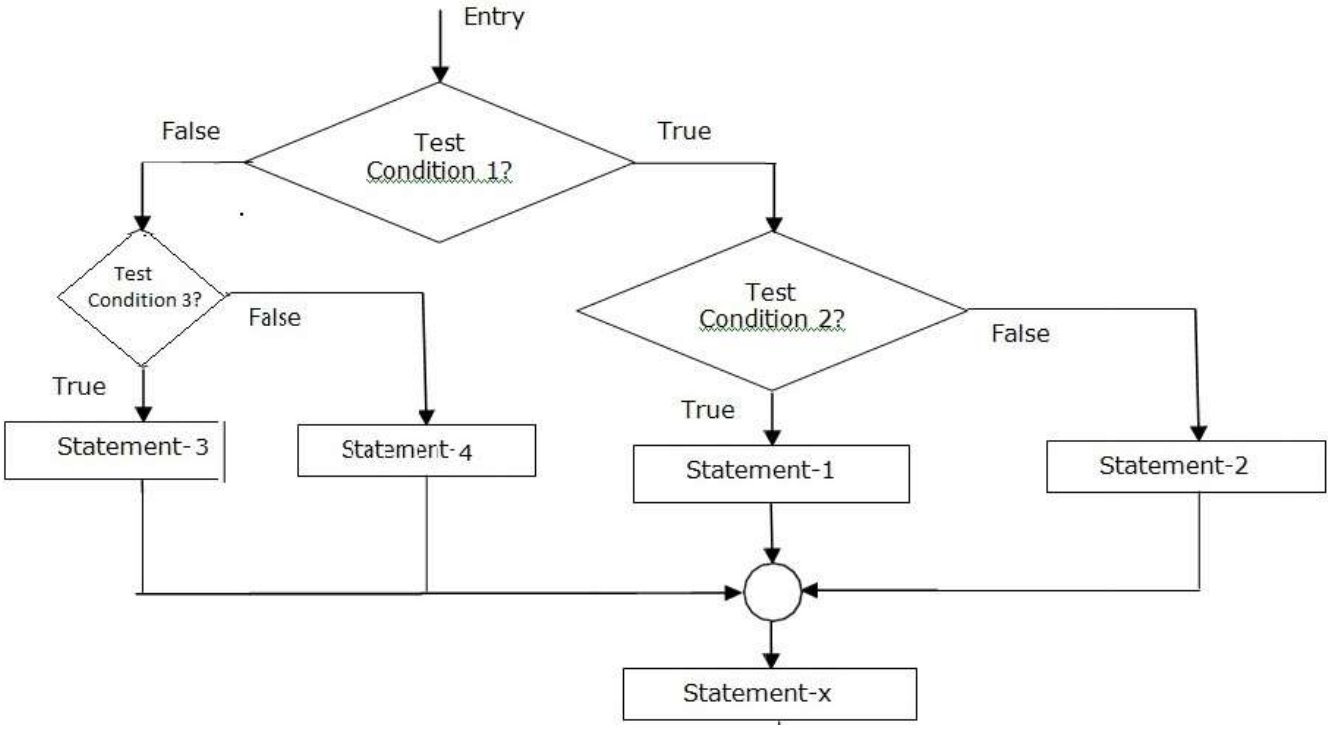
**4 Explain nested if with example and draw flowchart.**

Some time it may happen that one condition is depend on result of other condition. So if 1st condition is true 2nd condition will be executed otherwise control will move to else block and again check for other condition.
So we can say when we are using condition inside other condition, means we use if block inside other if block then it is called nested if.
The general form or syntax of nested if is as follow.

```
if(test-condition-1)
{
        if(test-condition-2)
        {
                statement-1;
        }
        else
        {
                statement-2;
        }
}
 else
{
        if(test condition 3)
        {
                statement-3
        }
        else
        {
                statement-4
        }
}
statement-x
```

- If *test-condition-1* is true then *test-condition-2* is evaluated. If it is true then *Statement-1* will be executed, if it is false then *Statement-2* will be executed.
- If *test-condition-1* is false *Test condition 3* will is evaluated, and if it is true then statement-3 will be executed and if it is false then statement 4 will be executed.

Flow chart structure of nested if else is shown is fig.




Example : Write a Program to find maximum number among three nos.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,c;
        clrscr();
        printf("enter the value of a,b and c?");
        scanf("%d %d %d",&a,&b,&c);
        if(a>b)
        {
                if(a>c)
                {
                        printf("%d is maximum",a);
                }
                else
                {
                        printf("%d is maximum",c);
                }
        }
        else
        {
                if(b>c)
                {
                        printf("%d is maximum",b);
                }
                else
                {
                        printf("%d is maximum",c);
```

```
            }
        }
        getch();
}
```

## 5 Explain switch-case statement with example.(Menu Driven)

- The switch statement is a multi-way decision making uses menu a list of option and we need to select one as a choice.
- It is alternative of multiple if….else or ladder if…else.
- When there are too many conditions in ladder if else or multiple if else it is always better to use switch case in that condition.
- In side switch case we need to enter a choice and the value we entered as a choice, that case should be executed. If we provide a choice other than list of menu than default case will be executed where we can define any user define message.
- It tests whether an expression matches any one of the constant values or not.
- Switch requires a break key word to break a case it is optional but if we don't write a break key word than its consecutive cases will be executed automatically.
- General form of switch-case is as below syntax,

```
switch(choice)
{
        case 1:         statement 1;
                        break;
        case 2:         statement 2;
                        break;
        case 3:         statement 3;
                        break;
                    ……..
                    ……..
        default:
                            statements
}
```

- *Choice* in switch should be integer or character expression. Float or any other data type is not allowed.
- Each case is labeled by one valued constant.
- If a case matches the choice value then execution starts at that case.
- Value of all case expressions must be different.
- If none of the cases are matched then default case is executed. Default case is optional.
- The break statement causes an immediate exit from the switch.
- The switch expression must be integral type. Float or other data types are not allowed.
- Case labels must be constant or constant expression.
- Case labels must be unique.
- Case labels must end with colons.
- The break statement is optional.
- The default case statement is optional.

Example:
Write a Program to perform arithmetic operation using switch case.

```
#include<stdio.h>
#include<conio.h>
 void main()
{
        int a,b,ch;
        clrscr();
        printf("\n\n1. addiditon\n\n 2. subtraction\n\n 3. division \n\n4. multiplication");
        printf("enter value of a and b ");
       scanf("%d  %d",&a,&b);
        printf("enter your choice: ");
```

```
        scanf("%d",&ch);
        switch(ch)
        {
                case 1:  c=a+b;
                         printf("\n addition = %d",c);
                         break;
                case 2:  c=a-b;
                         printf("\n subtraction = %d",c);
                         break;
                case 3:  c=a/b;
                         printf("\n division= %d",c);
                         break;
                case 4:  c=a*b;
                         printf("\n multiplication = %d",c);
                         break;
               default : printf("\n please enter valid choice");
                         break;
      }
     getch();
}
```

## 6 State difference between if-else and switch-case

| if-else | switch |
|---|---|
| *If* statement is used to select among two alternatives. | The *switch* statement is used to select among multiple alternatives. |
| *If* can have values based on constraints. | *Switch* can have values based on user choice. |
| Float, double, char, int and other data types can be used in *if* condition. | Only int and char data types can be used in *switch* Block as a choice |

## 7 Explain goto statement with example. Explain forward reference and backward reference or forward jump and backward jump with the help of example.

A **goto** statement in C programming provides an unconditional jump from the 'goto' to a labeled statement in the same function. So goto is also called **unconditional branching statement.**

**Syntax**

The syntax for a **goto** statement in C is as follows:

1)

```
goto label;
..
..
label:
statement;
```

or

2)

```
label:
..
..
goto label;
statement;
```

In goto unconditional branching jump is divided into two categories.
1 forward jump or forward reference (As per syntax 1)
2 backward jump or backward reference ( As per syntax 2)

Forward reference or forward jump is used to skip certain line of code and backward reference or backward jump is used to repeat certain line of code.
In forward jump label is written after goto statement and in backward reference label is written before goto statement.

Example.

```
#include <stdio.h>
#include<conio.h>

void main ()
{
        /* local variable definition */
        int a;
        abc:                        // label abc created for backward reference
        printf("enter value of a");
        scanf("%d",&a);
                if( a == 99)
                {
                        goto xyz;
                }
        printf("value of a =%d",a);
        goto abc;                    // control goes to again abc label to repeat statement
        xyz:                        // label for forward reference
        printf("You have entered 99");
        getch();
}
```