UNIT: 3
Decision Statements and Control Structure

- **Decision Statements**
- **Conditional branching statements:**

Simple if statement:

Syntax:

```
if (condition)
{
  Statement (s)
}
```

If the given condition is true then the statements inside the body of "if" will be executed. If the condition is false then the statements inside the body of "if" will be skipped.

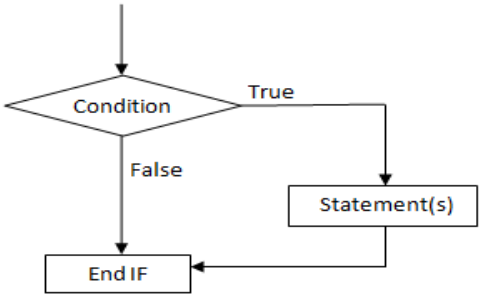


fig: Flowchart for if statement

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int marks=45;
    if( marks < 35 )
    {
    printf("Student is fail");
    }
    getch();
    }
```

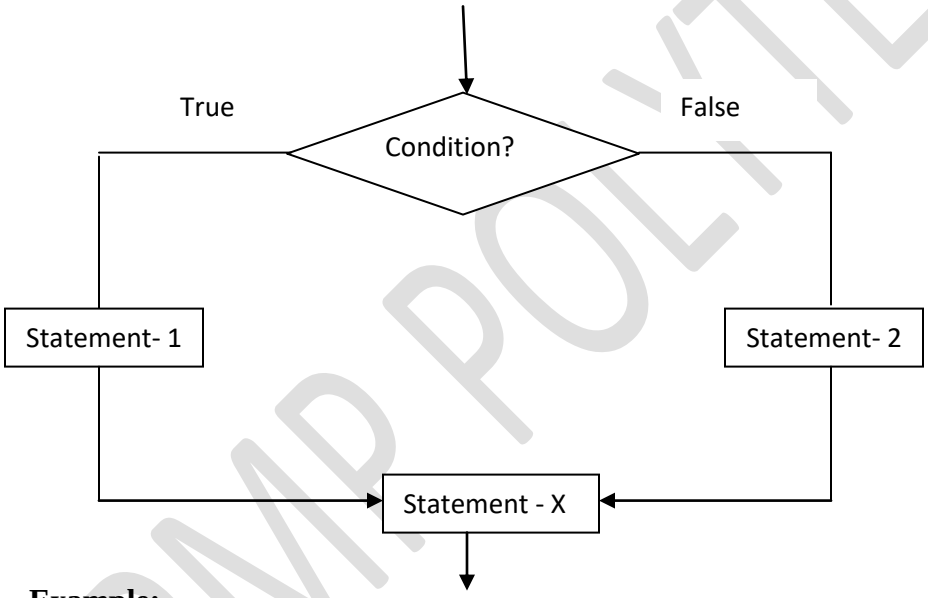
If-else statement:

Syntax:

```
if(condition)
{
Statement 1;
}
else
{
Statement  2;
}
```

The if..else statement is executed in the following order:

1. first the condition is checked.
2. if condition is true the statement -1 is executed.
3. if condition is false the statement - 2 is executed.



Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int marks=45;
    if( marks < 35 )
    {
    printf("Student is fail");
```

```
    }
    else
    {
    printf( "Student is pass" );
    }
    getch();
}
```

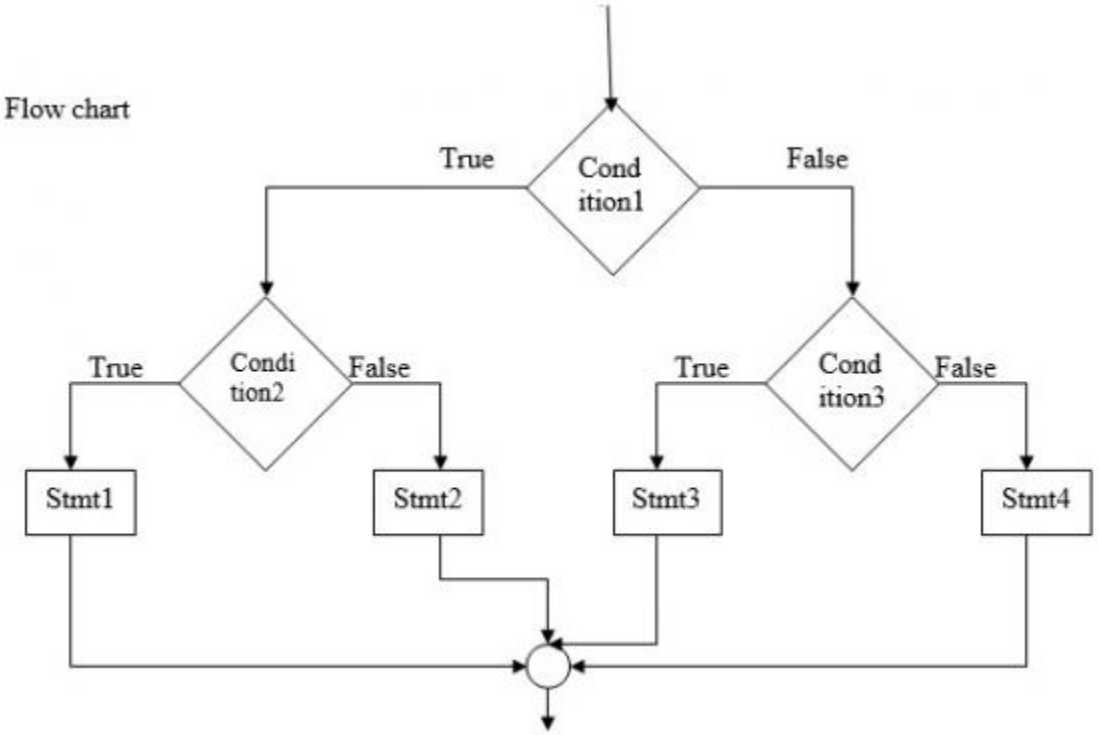
Nested If-else statement:

Nested If-else statement means if-else statement within another if-else statement.

Syntax:

```
if (condition1)
{
  if (condition2)
    stmt1;
  else
    stmt2;
}
else
{
  if (condition3)
    stmt3;
  else
    stmt4;
  }
```

Flow chart



Example:

```
#include<stdio.h>
#include<conio.h>
void main (){
  int a,b,c;
  printf("Enter the values of a,b,c: ");
  scanf("%d,%d,%d",&a,&b,&c);
  if((a>b)&&(a>c))
{
    if(a>c)
   {
      printf("%d is the largest",a);
    }
  else
   {
      printf("%d is the largest",c);
    }
 }
else
 {
    if(b>c)
     {
      printf("%d is the largest",b);
    }
```

```
else
   {
      printf("%d is the largest",c);
    }
  }
```

if else if…ladder

It is used **for multiple choice**.

Syntax:

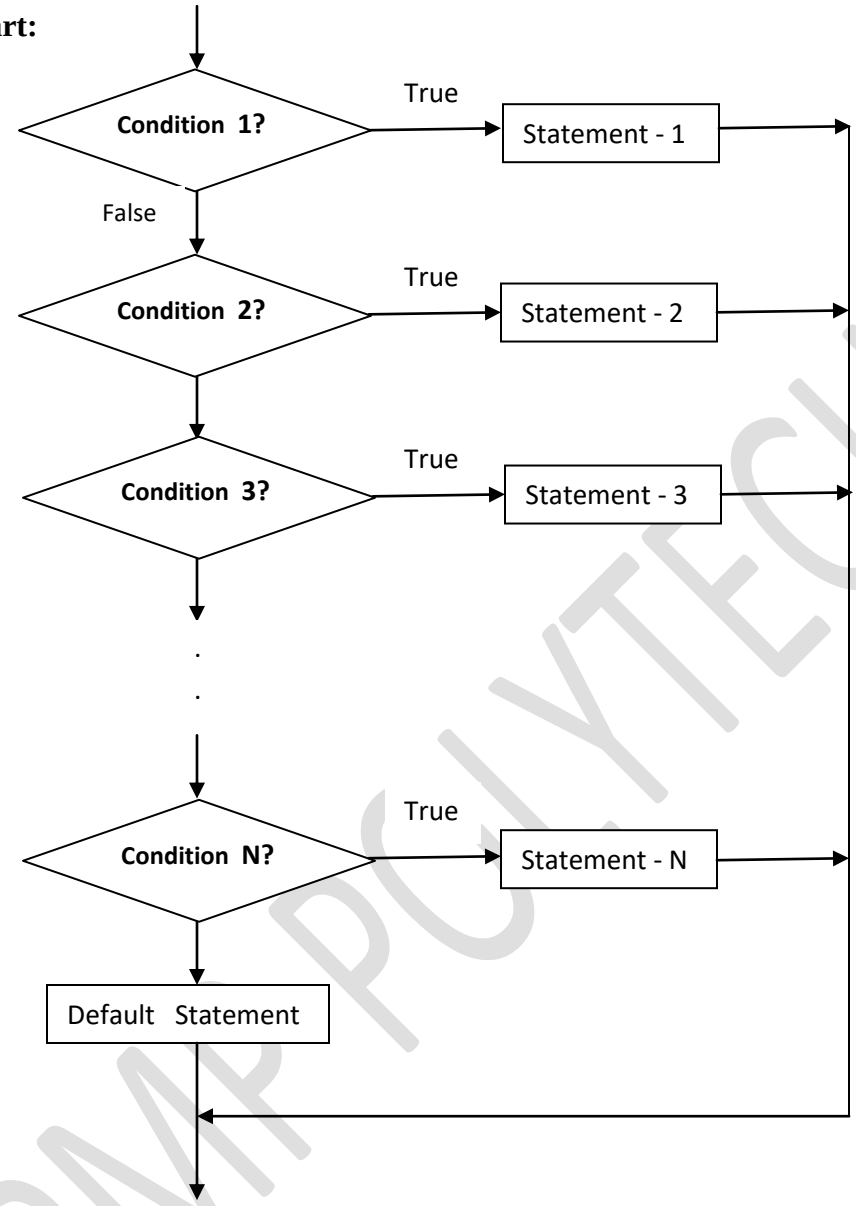
```
if(condition-1)
{
        Statement - 1;
}
else if(condition-2)
{
        Statement - 2;
}
else if(condition-3)
{
        Statement - 3;
}
        .
        .
        .
else if(condition-N)
{
        Statement -N;
}
else
{
        Default statement;
}
```

if-else-if ladder is executed in the following order:

1. First condition-1 is executed, if the condition-1 is true then the statement-1 is executed.
2. if the condition-1 is false then condion-2 is checked. If condition-2 is true then statement-2 is executed.

3. This procedure repeated until all the condition is checked. if all the condition became false then the default statement is executed.

Flowchart:



Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int no;
    printf("enter number\n");
    scanf("%d",&no);
    if( no > 0 )
    {
    printf("Number is Positive");
    }
    else if (no  < 0)
    {
    printf( "Number is Negative" );
    }
    else
    {
    printf( "Number is Zero" );
    }
    getch();
}
```

Switch statement:

- It is used to select one among multiple decisions.
- The switch statement is also known as multi-choice or multi decision statement.
- 'switch' successively tests a value against a list of integers or character constant.
- When a match is found, the statement(s) associated with that value is executed.
- Break statement send the control to the next statement after switch statement.
- If the case not found then the statement associated with the default case is executed.

Syntax:

```
switch(variable name or expression)
{
case  label1:
        statement(s) 1;
        break;
case  label2:
        statement(s) 2;
        break;
                .
                .
case  label N:
        statement(s) N;
        break;
default   :
        default statement(s);
}
```

Example:

```
void main()
{
int a=2;
clrscr();

switch(a)
{

case 1:
        printf("One \n");
        break;
case 2:
        printf("Two");
        break;
case 3:
        printf("Three");
        break;
default:
        printf("Wrong choice");

getch();
}
```

O/p: Two

❖ Unconditional branching statement: goto

- The go to statement in the c programming is used to transfer the control unconditionally from one part of the program to other part of the program.
- 'c' language provide a **unconditional branching mechanism** called as go to statement.

Syntax:

```
goto label ;
```

- Here , label is the label to the statement to which go to transfer control.
- Label must be a valid identifier.

- There are two possible use of goto statement.
  1. **Forward Reference**
  2. **Backward Reference**

Forward reference	Backward reference
goto label; ……. ……. Label: Statement;	Label: Statement; …… …….. Goto label;
Target statement comes after the goto.	Target statement comes before the goto.

❖ Control Statements:

When we want to repeatedly run a particular statement until a particular condition is met, then in that case we use Loop Statements.

There are three types of loop statements in C language:

- While loop
- Do while loop
- For loop

While loop:

- It is entry control loop.
- In the while loop, first the condition is checked and if the condition is true then the statement inside the while loop is run.
- As soon as the condition is False, the control is moved out of the while loop, and another statement is executed.
- At first time, condition is false, then body is not executed.

Syntax:

```
while (Condition)
{
Statements;
}
```

Example:

```
i =1;

while (i < = 10)
{
 printf( "% d \n " , i );
 i =  i+1;
 }
```

Do-While loop:

- It is exit control loop.
- In do-while loop, first body is executed and then condition is checked and if the condition is true, then again body of the loop will be executed, otherwise next statement after the loop will be executed.
- At first time, condition is false, at least once the body is executed.

Syntax:

```
do
{
// statements inside the loop
}
While (condition);
```

Example:

```
i =1;
do
{
```

```
        printf( "% d \n " , i );
        i =  i+1;
    } while (i < = 10);
```

For loop :

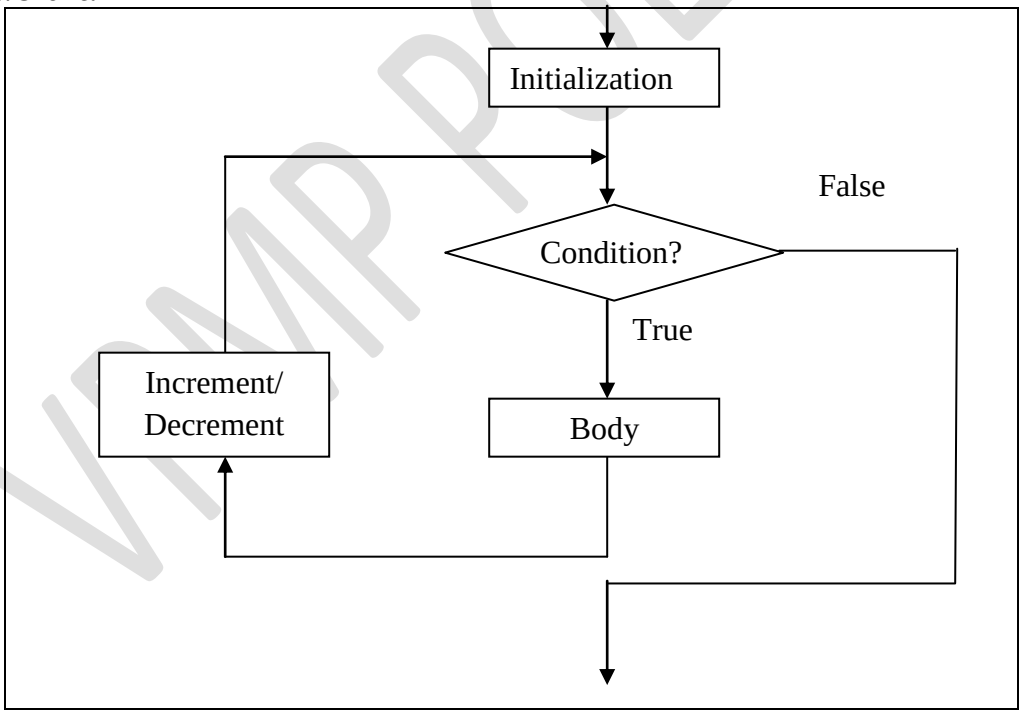
- In For Loop, we write both the initialization and control condition of the variable together inside the parentheses "()".
- If the condition of For Loop is true, then the statement inside it is run, or else the statement is not run.

Syntax:

```
for  (Initialization ; Condition ; Increment or decrement)
{
 body of the loop;
}
```

- **Initialization:** The control variable is initialized in the initialization expression. It executes only once.
- **Condition:** The condition statement checks the value of the control variable. If the condition statement is true, the body of the loop is executed.
- **Increment/Decrement:** The increment/decrement of the control variable is done in this part. After the incrementing/decrementing the value of control variable, it is tested using condition if condition is true than again the body of loop is executed and this process is repeated until the condition become false.

Flowchart:



Example: Print 1 to 5 numbers using for loop.

```
for(i = 1; i < =5 ; i++)
        {
           printf(" %d  \n ", i ) ;
        }
```

Nested for loop:

Nested for loop means for loop within another for loop.

Syntax:

```
for (Initialization; condition; increment ) {

   for (Initialization; condition; increment ) {
     statement(s);
   }
   statement(s);
}
```

Example:

```
for(i=1;i<=5;i++)
{
   for(j=1;j<=i;j++)
  {
    printf("%d",i);
  }
  printf("\n");
}
```

❖ Break and continue statements:

Break statement:

- It terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch.
- Keyword used is "**break**".
- Break from loop:

```
for(     ;     ;     )
{
    for(     ;     ;     )
    {
```

```
                    if(condition)
                        break;
                }
                --------------------
            }
```

Example;

```
for(i=1; i<=5 ; i++)
 {
 if(i==3)
    break;
printf("%d\n",i);
 }
```

Output: 1 2

Continue statements :

- It causes the loop to skip the remainder of its body and starts the next iteration.
- Keyword used is "**continue**".

```
for(     ;     ;     )
    {

            --------------
            if( condition)
                continue;
            ------------

    }
```

Example:

Print 1 to 5 number except 3.

```
for( i=1 ; i <= 5 ; i++)
{
        if ( i ==3)
            continue;
        printf("%d " ,i);
}
```

Output: 1 2 4 5