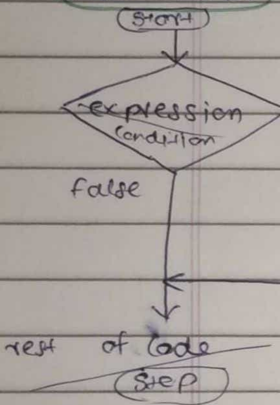
Module-3

Braching : to control the flow of execution of the program according to the requirements

(1) **if statement** : It is very frequently used in decision making, allow the flow of program execution

Syntax :-
```
if (condition){
    statement ;
}
```

flowchart :-



program :
```
#include<stdio.h>
int main(){
   int n ;
 printf("Enter the Integer" number :");
 scanf("%d", &n)
 If (n>0){
   Printf("Given number" is ~~+ve~~ Number
 }
 return 0;
}
```

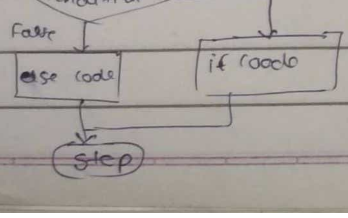
output:
```
Enter the Integer
    number : -20
Given number is (-ve)
    number
```

(2) **If-else statement** : - The if else statement in C is like another if condition

- it's used in program when if statement having several decision.

Syntax :
```
if (condition){
  /* statement 1 will excute if the condition */
                               is true
}
else {
/* statement 2 will excute if the condition */
                               is false
}
```

Flowchart :



Program :
```
#include<stdio.h>
int main()
 int n;
printf("Enter the Enumber :");
scanf("%d", &n)
If (n%2==0){
Printf("Given number is even
          number : \n" );
}
else
{
 printf("Given number is odd
          number : \n" );
```

if else aur if else

multiway decision statment

(3) Nesting of if - else statement

↳ when many possible decisions are involved, more than one if-else is used

~~Syntax:~~

```
~~if (condition)
{
    if (condition)
    {
        statemen 1;
    }
    else
    {
        statement 2;
    }
}
else
{
    statement 3;
}~~
```

~~Syntax:~~

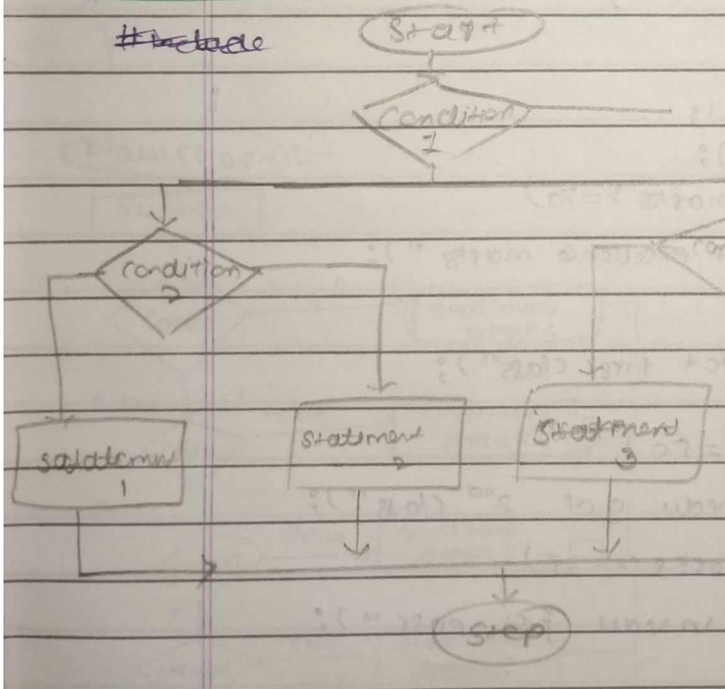
```
~~if (condition)
{
    statement 1;
}
else if (condition)
{
    statement 2;
}
else {
    statement 3;
}~~
```

Syntax:

```
if (condition)
{
    if (condition)
    {
        statement 1;
    }
    else
    {
        statement 2;
    }
}
else
{
    statement 3;
}
```

Flowchart:

~~#include~~



Program:

```
#include <stdio.h>
int mounc()
int marks;
printf()
scanf()
if (marks >=250)
{   if (attendenco >=24)
    {
    printf("Pass")
    }
    else
    print ("your Attendence
            are low");
}
else
{ Printf(" fail ");
}
```

(4) The else if ladder

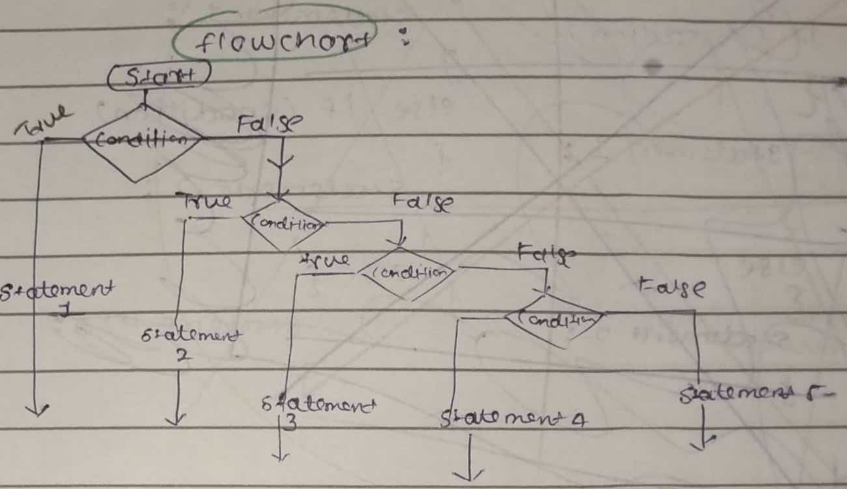
↳ There is another way of putting it together when multi path decision are involved

↳ It used in multi-way decision on several condition

Syntax :-

```
if (condition 1)
  {
   statement 1;
  }
  else if  (condition)
     {
        statement 2;
     }
     else
     {
      statement 3 ;
     }
```

flowchart :



~~Flowchart~~ :

Program :

```
#include<stdio.h>
void main() {
int  marks ;
printf("Enter  marks:");
scanf(" %d" ,&marks );
if (marks <=100 && marks >=70)
   printf("\n you got excellence marks ");
else  if (marks >=60)
     printf("\n you got first class");
     else if ( marks >= 50)
       printf("\n you got  2nd class");
       else if (marks >=35)
         printf("\n you just pass ");
         else
           printf(" Sorry! You are Fail");
}
```

→ Break statement is used to terminate the each case.

(5) Switch statement :

↳ The switch statement is used as a substitute of Nested-if-else statement

It is used when multiple choices are given and one choice is to be selected

Syntax :-

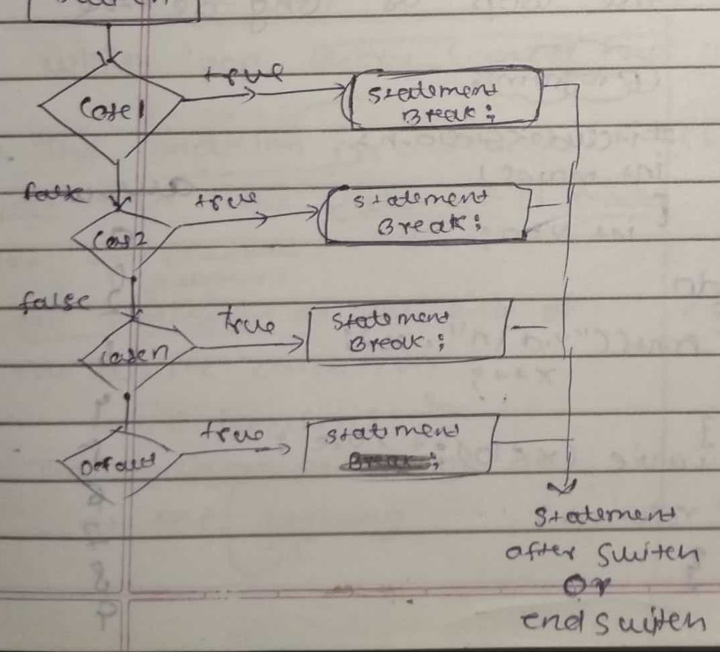
```
switch (condition )
{
case  value 1 :
        statement_1 ;
        Break;
~~case  value 2 :~~
        Statement_2;
  :       :
  :       :
case  value n:  statement_n;
        Break ;
   default  :    default_statement;
}
```

Flow Program : e.g switch (number) ← keyword

```
#include<stdio.h>
  int main() {
    int day ;
  Printf ("Enter day (1-7):");
  Scanf("%d" , &day );

   switch (day){
  case 1 : Printf("monday \n");
           Break ;
  case 2 : Printf ("tuesday \n");
           Break ;
  ~~case 3 : Printf ("wednesday \n");~~
           break ;
   case 4 : Printf (" Thursday \n");
           break ;
   case 5 : Printf (" Friday \n");
           break ;
   ~~case 6 : Printf (" saturday \n");~~
           break ;
   case 7 : Printf ("sunday \n");
           break ;
   default :
      Printf("not a valid \n" day!
   }
 return 0 ;
}
```

Flow chart :-



Loops → any repeated task more than one time is called loop

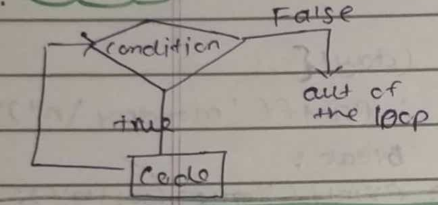
(1) While loop : → also know as entry of the loop
↳ while loop is used to
repeat → Specific block of code
an unknown number of times,
until a condition is met.

Syntax :-
```
while (condition)
{
  Statement ;
}
```
not use ;

Program :-
```
#include<stdio.h>
int main()
{
  int x=0;
while (x<10)
{
  Printf("%d\n", x );
      x++ ;

  }
return 0
  }
```
output :-
```
0
1
2
3
4
5
6
7
8
9
```

Flow chart :-



(2) do while loop : → also know as ~~exact~~ exit of control loop
↳ The do while loop is a variant of the while loop.
→ This loop will execute the code block once.
→ before checking if the condition is true,
→ then. it will repeat the loop as long as the condition is true.

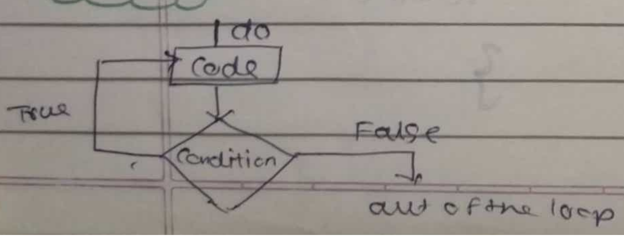
Syntax :-
```
do
{
}
while (condition);
```

Program :
```
#include<stdio.h>
int main()
{
  int x=0;
do
{
  Printf("%d\n", x );
      x++;

}
while (x<10);
return 0;
}
```
> use ;

output
```
0
2
3
4
5
6
7
8
9
```

Flow chart :-



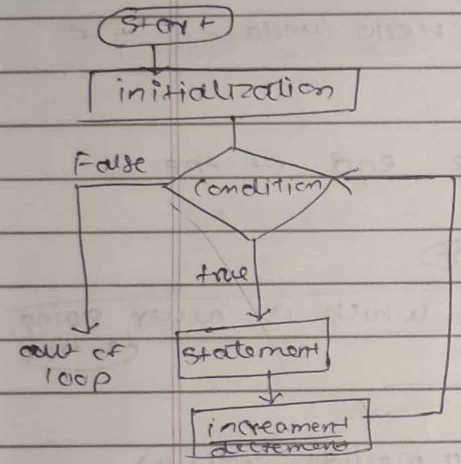
(3) For loop :

↳ For loop execute the statement of program for several time repeating until a given condition return false.

syntax :-

```
for (initialization ; condition ; increament/decrement )
  {
      statement;
  }
```

Flow chart



program :

```
#include<stdio.h>
 int main()
 {
For ( int x=0; x<5 ; x++)
  {
   printf(" %d\n", x )
  }
return 0;
}
```

output : 0
1
2
3
4
5

while loop	do while loop
(1) while loop entry control loop	(1) do while loop exit control loop
(2) The condition is checked First then the statement is excuted	(2) The statement is executed ~~at least one~~ then after checked condition
syntax while (condition) { statement; }	~~do~~ { statment; } while (condition);
(3) No semicolon at the end of the while condition while (condition)	(3) semicolon use at the end of while condition while (condition);
(4) The condition statement is at the begining	(4) The condition statement is at the end of the loop.

Infinite loop → A loops that repeats indefinitely and never terminates is called infinite loop

e.g

```
#include<stdio.h>
void main()
{
for (k=0 ; k<10 ; k--)
{                         infinite goes
   printf ("Hello");
     k++;
 }
return 0;
}
```

output :

```
Hello Hello Hello ----- ∞
```

empty loop → A semicolon at the end of the while

for statement

the condition which is never going to true

```
#include<stdio.h>
 int main(){
 int i;
while (i<5);
  printf("Hello maam");
} return 0;
  }
```

```
#include <stdio.h>
  int main(){
 int i;
for (i=0 ; i<10 ; i++);
  printf ("Hello maam");
    return 0;
}
```

Nested loop :

↳ one loop inside another loop

Syntax :-

```
for (intilization ; condition ; increment
                                decreem )
{
   For (intilization ; condition ; increm
                                   decrement )
   {
     statement 1 ;
   }
 Statement 2 ;
}
```

example :-

```
#include<stdio.h>
void main()
{
 int i , j ;
for (i=1;   i<=2   ; i++ )
{
     for (j=1;   i<=3  ; j++ )
     {
      printf ("j=%d ,
     }
}
 printf (
```

```
#include<stdio.h>
void main()
(I) {
     int  i , j , row ;
     printf(" Enter the no. of raw);
     scanf(" %d", & raw );

   for (i=1; i<=row ; ++i)
   (1) {
     for(j=1 ; j<=i ; ++j)
    (2) { printf(" *")

    (2) }
     print f ("\n)"
   (1) }
     return 0;
   } (I)
```

row=4

```
*
* *
* * *
* * * *
```

Break	Continue
(1) Break is used to Break the loop or iteration	(1) Continue is used to Continue the loop or iteratio
~~(2) Break keywor used~~ (2) keyword used is "break"	(2) keyword used is "continue"
(3) Break keyword is used in switch case	(3) Continue ~~Break~~ keyword is not used in switch cas
(4) control is transferred outside the loop	(4) control is transfered in the same loop

(5) Syntax :
 while (condition){
 //code
 if (condition to break)
 {
 Break;
 }
 //code
 }
	(5) Syntax :
	 while (condition){
	 //code
	 if (condition)
	 {
	 Continue;
	 }
	 //code
	 }

goto program:-

```
#include<stdio.h>
 int main() {
 int a;
printf("\n Enter No.");
scanf("%d",&a);
if (a%2 ==0)
  goto even;
  else
  goto odd;
```

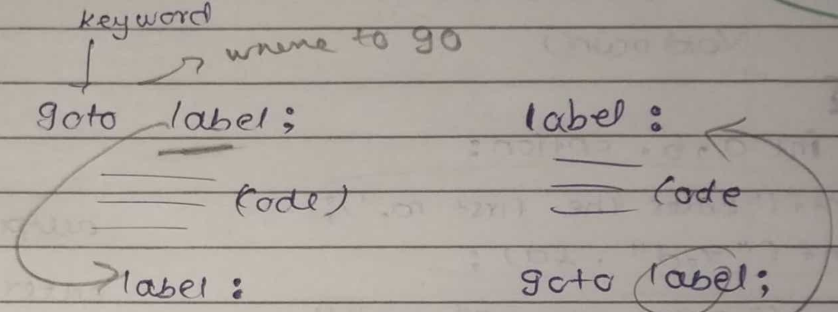
```
even :
printf(" \n No. is even);
return 0;

odd :
Printf("\n No. is odd);
 return 0;
```

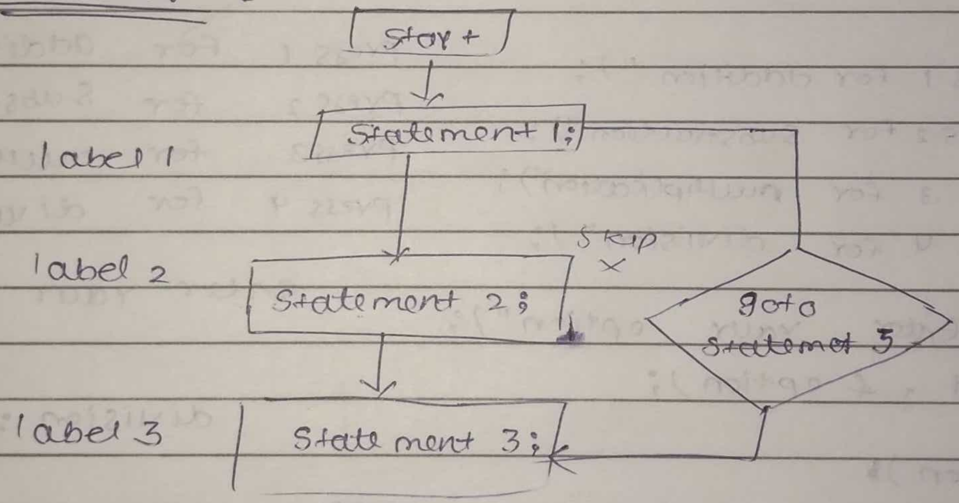
Goto statement

→ goto statement is a jump statement
→ It is used to transfer → the program control → From one location to another location

Syntax :



Flowchart :



Program :

```
# include<stdio.h>
int main() {
   int n=1;
   Jump:
      printf("%d  ",n);
      n++;
      if (n<=10)
   goto Jump;
   return 0;
```

output
1 2 3 4 5 6 7 8 9 10

100% Fix ask for 10 marks

Menu- Driven Program :

write a menu- driven program using
function (1) Addition of two number
(2) subtraction of —11—
(3) multiplication of —11—
(4) division of —11—

```
#include<stdio.h>
Void main()
{
    int a, b, option;
①  Printf("Enter the first no.");
    scanf("%d", &a);
②  Printf(" Enter the 2nd no.");
    scanf("%d", &b);

    Printf("\npress 1 for addition ");
    Printf("\npress 2 for substraction");
    Printf("\npress 3 for multiplication");
    Printf("\npress 4 for division");

    printf("\n Enter your option");
    scanf("%d , & option);
    switch (option)
    {
    case 1 : Printf (" addition %d", a+b);
             Break;
    case 2 : Printf ("Substraction =%d",a-b );
             Break;
    case 3 : Print f (" multiplication =%d", a*b);
             Break;
    case 4 : Printf (" division =%d", a/b);
             Break;
    default : printf(" You Entered invalid number");
    }
}
```

output :

```
enter 1st number : 5
enter 2nd number : 5

Press 1 for addition
Press 2 for Substraction
Press3 for multiplication
Press 4 for division

enter your option : 4

division : 1
```