20CS1101 –PROGRAMMING FOR PROBLEM SOLVING

(Common to all branches)

UNIT – III

DECISION STATEMENTS: Introduction, Types of If statements, switch statement, break, continue, goto.

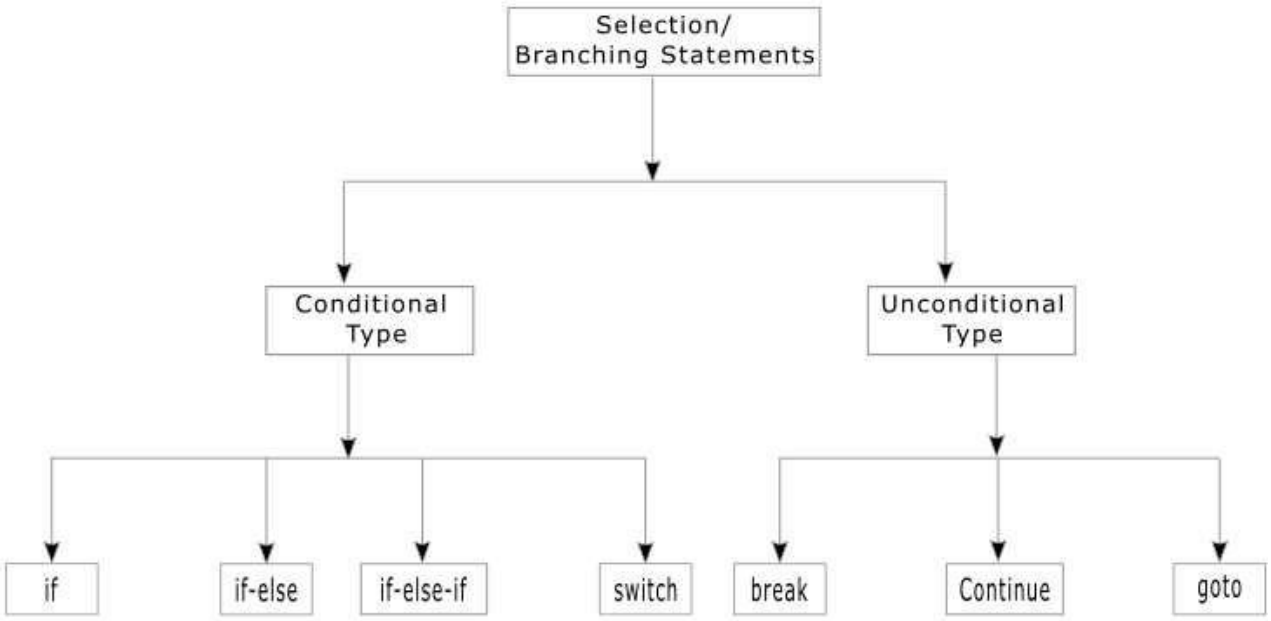
ITERATIVE STATEMENTS: while, do-while and for loops.

→Decision Statements (Control Statements)

Introduction: In any program statements are normally executed. We have number of situations repeat a group of statements until certain specified condition or change the order of execution of statements. The C language supports the control statements as listed below

1. The simple if statement
2. The if-else statement
3. Nested if-else statement
4. The if-else-if ladder statement
5. The switch( ) case statement
6. The nested switch( ) case statement

Flowchart Control Statements

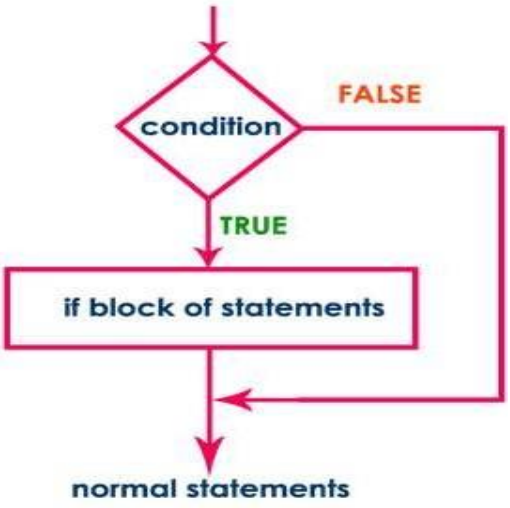


1. The simple if statement: The if statement is used to specify conditional execution of program statements or a group of statements enclosed in braces.

Syntax

```
if ( condition )
{
   ....
   block of statements;
   ....
}
```

Execution flow diagram



The statement is executed only when the condition is true. In case the condition is false the compiler skips the lines within a pair of parenthesis. The conditional statements should not be terminated with semi colons.

Example 1: Write a c program to check whether the entered number is less than 10? If yes, display the same.

```
 #include<stdio.h>
#include<conio.h>
void main( )
  {
                int n;
               clrscr();
                printf("Enter the number");
               scanf("%d",&n);
               if(n<10)
               printf("\n number is less than 10");
               sleep(2);
               getch();
              }
```

Example 2: Write a c program to check whether the candidates age is greater than 17 or not. If yes display message "Eligible for voting".

```
 #include<stdio.h>
#include<conio.h>
void main( )
  {
                int  age;
```

```
clrscr();
 printf("Enter the age");
scanf("%d",&age);
if(age>17)
printf("\nEligible for voting");
getch();
}
```

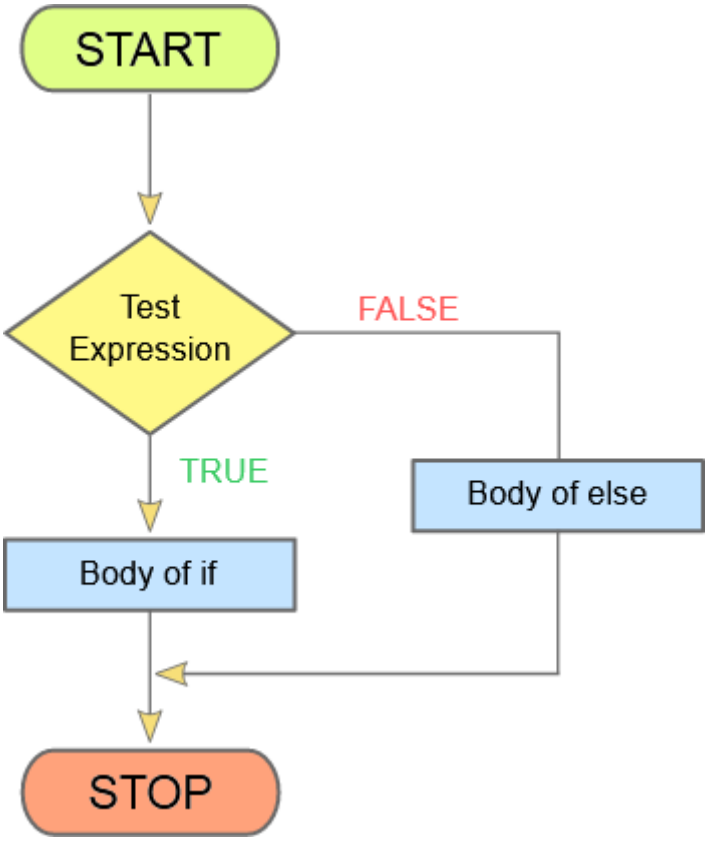
Example 3: Write a c program to use curly brace in the if block. Enter only the three numbers and calculate their sum and multiplication.

```
#include<stdio.h>
#include<conio.h>
void main( )
  {
    int a,b,c,x;
   clrscr();
    printf("Enter three numbers");
   scanf("%d%d%d",&a,&b,&c);
   if(x==3)
   printf("\n sum=%d", a+b+c);
   printf("\n multiplication=%d", a*b*c);
   getch();
  }
```

2. The if-else statement: The **if-else** statement takes care of true as well as false conditions. It has two blocks. One block is for **if** and it is executed when the condition is **true.** The other block is of **else** and it is executed when the condition is **false.** The **else** statement cannot be used without **if**. No multiple else statements are allowed with one if.

Syntax: if (condition)

```
{
    Statement1;
    Statment2; If block (True block)
}
else
 {
    Statement3;
    Statement4; else block (False block)
  }
```



Example 1: Read the values of a, b,c through the keyboard add them and after addition check if it is in the range of 100 and 200 or not. Print separate message for each

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a,b,c,d;
  clrsr();
printf("Enter the three numbers");
scanf("%d%d%d",&a,&b,&c);
d=a+b+c;
if(d<=200&&d>=100)
printf("\n sum is %d which is in between 100 and 200", d);
else
printf("\n sum is %d which is outof range ",d);
getch();
}
```

Example 2:Write a c program to calculate the salary of medical representative bases on the sales. Bonus and incentive to be offered to him will be based on total sales. If the sales exceeds Rs 100000/- follow the particulars of table(1) otherwise table(2).

Table 1	**Table 2**
Basic= 3000	**Basic=3000**
H.R.A=20% of basic	**H.R.A =20% of basic**
D.A=110% of basic	**D.A=110% of basic**
Conveyance=500	**conveyance=500**
Incentive=10% of sales	**Incentive=5% of sales**
Bonus=500	**Bonus=200**

```
#include<stdio.h>
#include<conio.h>
void main()
{
  float bs,hra,da,cv,incentive,bonus,sales,ts
  clrsr();
printf("enter the toatal sales");
scanf("%d",&sales)l
if(sales>=100000)
 {
   bs=3000;
   hra=20*bs/100;
   da=110*bs/100;
   cv=500;
   incentive=sales*10/100;
   bonus=500;
 }
else
{
     bs=3000;
   hra=20*bs/100;
   da=110*bs/100;
   cv=500;
   incentive=sales*5/100;
   bonus=200;
}
ts=bs+hra+da+cv+incentive+bonus;
```

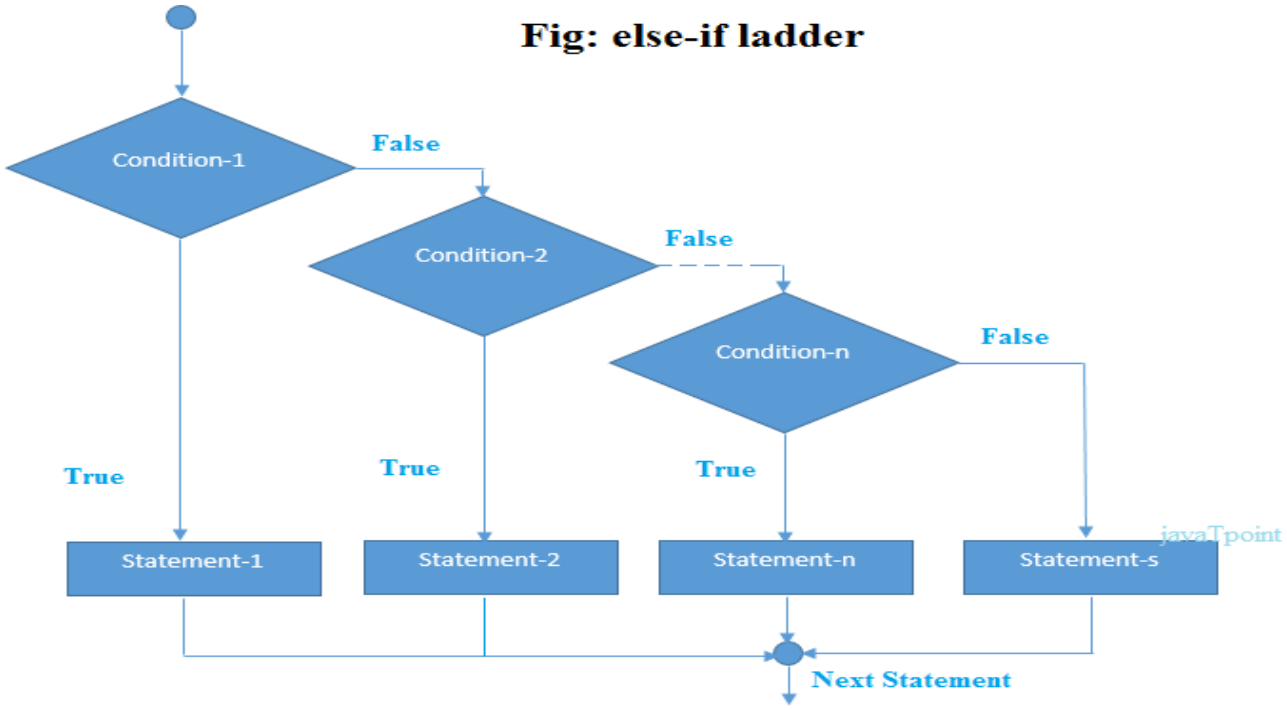
```
printf("\n total sales=%f",sales);
printf("\n basic salary=%f",bs);
printf("\n H.R.A=%f",hra);
printf("\n D.A=%f",da);
printf("\n conveyanvce=%f",cv);
printf("\n Bonus=%f",bonus);
printf("\n grass salary=%f",ts);
getch();
}
```

3.The if-else-if ladder statement: In this kind of statements number of logical conditions are executing various statements here, if any logical condition is true the compiler executes the block followed by if condition otherwise it skips and executes else block. In if-else statement else block is executed by default after failure of condition.

Syntax:

```
 if(condtion)
 {
  statement1;
  statment2;
 }
else if(condtion)
 {
  statement3;
  statement4;
 }
 else
  {
   statement5;
   statement6;
  }
```

Fig: else-if ladder



Example 1: Write a c program to calculate electricity bill. Read the starting and ending meter reading. The charges are as follows.

No. of units consumed	Rate in (RS)
200-500	3.50
100-200	2.50
Less than 100	1.50

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int intial, final, units;
 float price;
 clrscr();
 printf("\n enter the intial and final readings");
 scanf("%d%d",&initial, &final);
 untis=final-initial;
 if(units>=200&&units<=500)
  price=uints*3.50;
else if(units>=100&&units<=199)
  price=uints*2.50;
else if(units<100)
  price=uints*1.50;
```

```
printf("units=%d price=%f", uints,price);
getch();
}
```

Example : Write a c program to find the average of six subjects and display the result as follows.

Average	Result
>34 and <50	Third class
>49 and <60	second class
>60 and <75	First class

If marks in any subject less than 35 Fail

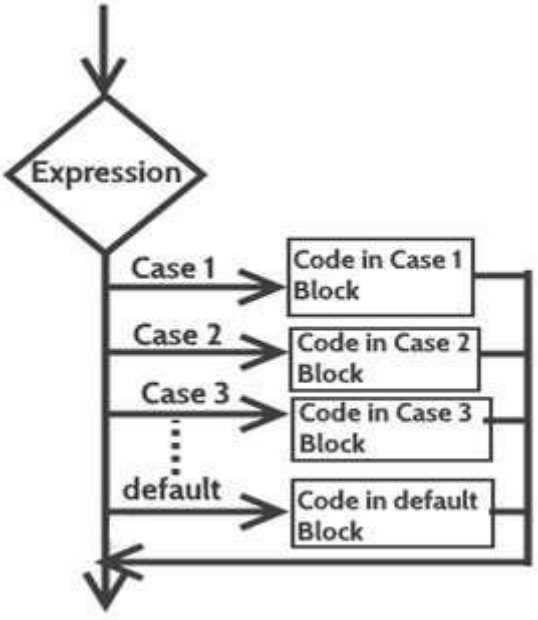
```
#include<stdio.h>
#inlclude<conio.h>
void main()
{
  int sum=0,a,b,c,d,e,f;
  float avg;
  clrscr();
  printf("Enter the six subjects marks");
  scanf("%d%d%d%d%d%d",&a,&b,&c,&d,&e,&f);
  sum=a+b+c+d+e+f;
  avg=sum/6;
  printf("sum=%d avg=%f",sum,avg);
 if(a<||35b<35||c<35||d<35||e<35||f<35)
  {
    printf("\n rerult is fail");
    exit(0);
  }
 if(avg>34&&avg<50)
printf("\n result is third class");
else if(avg>49&&avg<60)
printf("\n result is second class");
else if(avf>60&&avg<75)
printf("\n result is first class");
else if(avg>75&&avg<100)
printf("\n result is distinction");
getch();
}
```

4. The switch ( ) case statement: The switch statement is a multi-way branch statement. In the program if there is a possibility to **make a choice from a number of options, this structured selection is useful.** The switch statement requires only one argument of any data type. Which is checked with number of case options. The switch statement evaluates expressions and then looks for its value among the case constants. If the value matches with case constants, this particular case statement is executed. If not default is executed.

Here switch, case and break, default are keywords. Every case statement is terminated with semicolon. The break statement is used to exit from the current case structure.

Syntax:

```
switch(variable or expression)
 {
  case  constant1:
                statement;
                break;
  case constant2:
                statement;
                break;
   .
   .
   .
   .
 default:  statement;
 }
```



Example: Write a c program to provide multiple functions such as 1.addition 2.substraction 3. Multiplication 4. Division

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a,b,c,ch;
 clrscr();
 printf("----------");
 printf("\n\t Menu");
 printf("----------");
 printf("\n\t 1.Addtion");
 printf("\n\t 2.substraction");
printf("\n\t 3.multiplication");
printf("\n\t 4.division");
printf("enter the a,b values");
scanf("%d%d",&a,&b);
printf("enter the choice");
scanf("%d",&ch);
switch(ch)
{
 case 1: c=a+b;
        printf("sum=%d",c);
        break;
case 2: c=a-b;
        printf("sub=%d",c);
        break;

case 3: c=a*b;
        printf("multi=%d",c);
        break;
case 4: c=a+b;
        printf("division=%d",c);
        break;
default: printf("invalid choice");
}
getch();
```

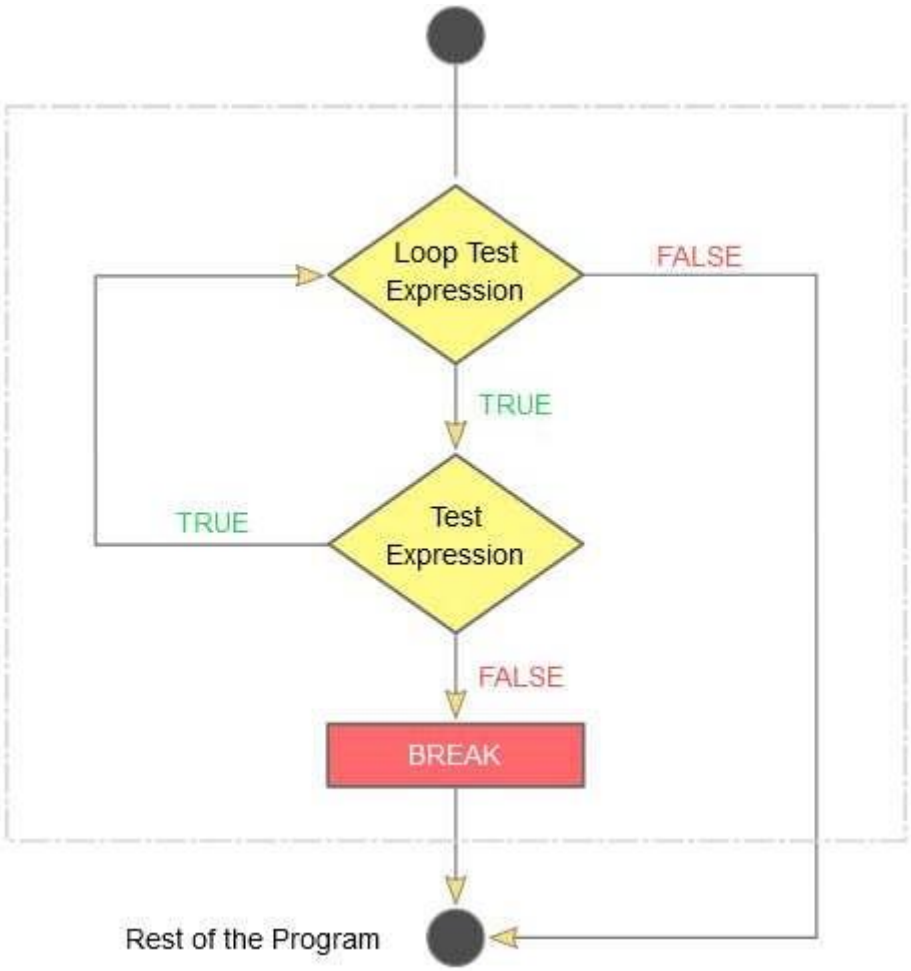
}

5. Nested switch ( ) case: C supports the nesting of switch ( ) case. The inner switch can be part of an outer switch. The inner and the outer switch case constants may be the same. No conflict arises even if they are same.

Example: Write a c program to demonstrated nested switch( ) case statement

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int x;
 clrscr();
 printf("enter a number");
 scanf("%d",&x);
 switch(x)
  {
    case 0:printf("number is even");
          break;
    case 1:printf("number is odd");
          break;
    default:
           y=x%2;
          switch(y)
           {
             case 0: printf("even");
                    break;
             default: printf("odd");
           }
  }
getch();
}
```

The break statement: The keyword break allows the programmers to terminate the loop. The break skips from the loop or block in which it is defined.



Break Statement Program

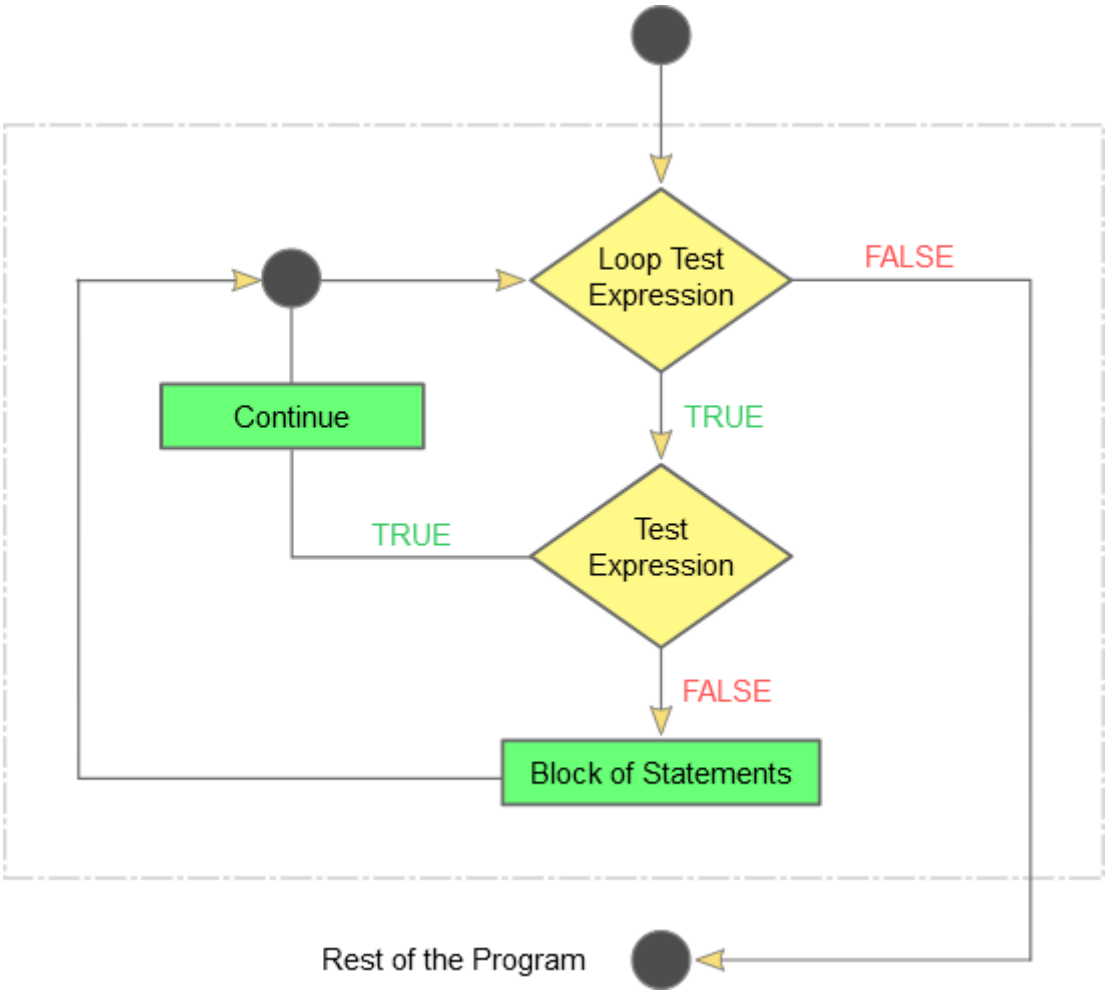
Let us write a C program to demonstrate break statement**.**

```
#include <stdio.h>
int main()
{
int a;
for(a = 1;a <= 5;a++)
{
if(a == 4)
{
break;
}
printf("%d ", a);
}
return 0;
}
```

Output

```
1 2 3
```

The continue statement: The continue statement is exactly opposite to break. The continue statement is used for continuing next iteration of loop statements. When it occurs in the loop it does not terminate, but it skips the statements after this statement.

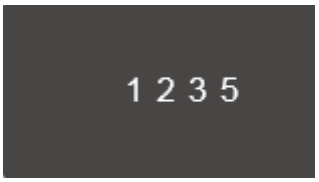


Continue Statement Program

Let us write a C program to demonstrate continue statement

```
#include <stdio.h>
int main()
{
int a;
for(a = 1;a <= 5; a++)
{
if(a == 4)
{
continue;
}
printf("%d", a);
}
return 0;
}
```

Output



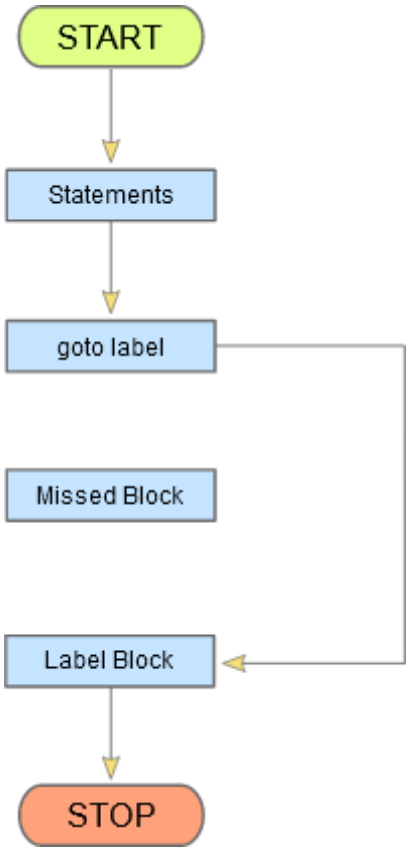
The goto statement: This statement does not require any condition. This statement passes control anywhere in the program. i.e; control is transferred to another part of the program without testing any condition.

Why goto statement

C programming introduces a goto statement by which you can jump directly to a line of code within the same file.

Syntax: goto label;

Where the label name must start with any character.



Example 1: write a c program to find the entered number is even or odd using goto statement.

```
#include<stdio.h>
#includ<conio.h>
void main()
{
  int x;
  clrscr();
  printf("enter a number");
  scanf("%d",&x);
  if(x%2==0)
```

```
  goto even;
  else
  goto odd;

even: prinf("\n %d is a even number");
      return;

odd: printf("\n %d is odd number");
     getch();
}
```

Example2: write a c program to check whether the entered year is a leap year or not use goto statement.

```
#include<stdio.h>
#includ<conio.h>
void main()
{
  int leap;
  clrscr();
  printf("enter the year");
  scanf("%d",&leap);
  if(leap%4==0)
  goto leap;
  else
  goto nonleap;

leap: prinf("\n %d is a lep year");
      return;
nonleap: printf("\n %d is not a leap year");
     getch();
}
```

Difference between break and continue statement

Break	**Continue**
1. Exist from current block or loop	1. Loop takes next iteration
2. Control passes to next statement	2. Control passes at the beginning of loop
3. Terminates the program	3.Never terminates the program.

→Iterative Statements (Loop control statements)

Introduction: Many tasks are needed to be done with the help of a computer and they are repetitive in nature. For example , the salary of laborers of a factory is calculated by the formula (No.of hours* wage rate). This calculation will be performed by an accountant for each worker every month. Such type of repetitive actions can be easily done using a program that has a loop into the solution of the problem.

Loop: A loop is defined as a block of statements which are repeatedly executed for certain number of times.

Steps in loop:

1) Loop variable: It is a variable used in the loop.

2) Initialization: It is the first step in which starting and final value is assigned to the loop variable. Each time the updated value is checked by the loop itself.

3) Increment/decrement: It is the numerical value added or subtracted to the variable in each round of the loop.

The loop statements are

1. for loop
2. Nested for loop
3. The while loop
4. The do-while loop

1. The for loop: This is used when the statements are to be executed more than once. This is the most widely used iteration construct. The for loop supported by C is much more powerful than its counterpart in other high level language.

Syntax:

```
for(initialization; condition; increment/decrement)
 {
    Statement1;
    Statement2;
 }
```

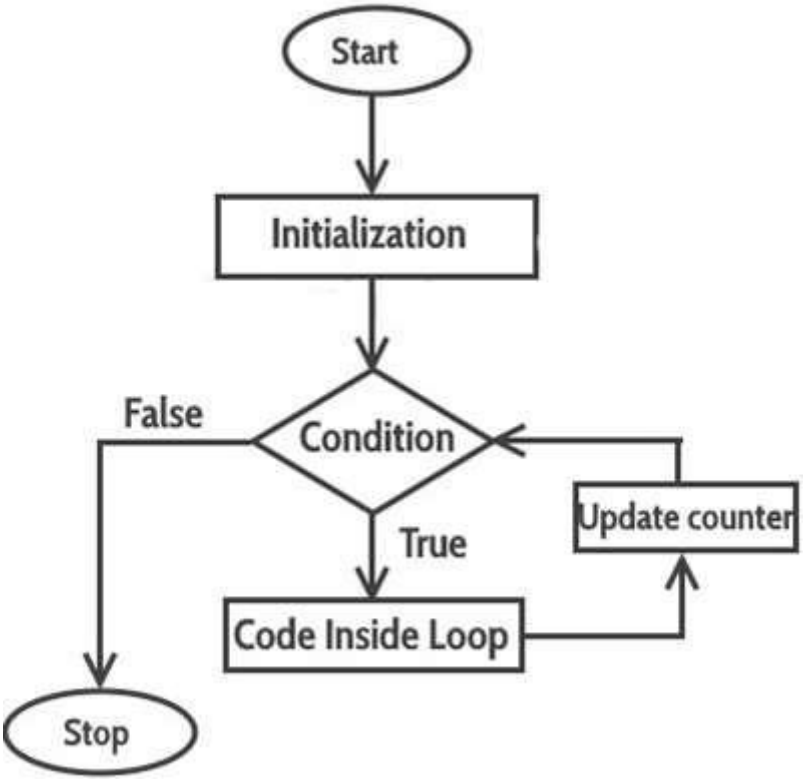
Initialization: The initial value is executed only once.

Condition: The condition is a relational expression that determines the number of iterations desired or it determines when to exit from the loop. The for loop continues to execute as long as conditional test is satisfied. When the condition becomes false the control of the program exists from the body of the for loop and executes next statement after the body of the loop.

Various formats of for loop:

Syntax	Output	Remarks
for (; ;)	Infinite loop	No arguments
for(a=0;a<=20)	Infinite loop	„a" is neither incremented nor decrement.
for(a=0;a<=10;a++)	Display value from 0 to 10	„a" is increased from 0 to 10.
for(a=10;a>=0;a--)	Display value from 10 to 0	„a" is decreased from 10 to 0

Syntax: for(exp1;exp2;exp3)
Statement;



Example1: write a c program to find the factorial of a given number

```
#include<stdio.h>
#include<condio.h>
void main()
{
  int n, i, fact=1;
  printf("Enter the number");
  scanf("%d",&n);
  for(i=1;i<=n;i++)
   {
     fact=fact*i;
    }
   printf("The factorial of a given number is%d", fact);
getch();
}
```

Example 2: write a c program to display numbers from 1to 10

```
#include<stdio.h>
#include<condio.h>
void main()
{
  clrscr();
  for(i=1;i<11;i++)
   {
     printf("%d", i);
   }
getch();
}
```

Example3: Write a c program to find the distance travelled by a vehicle(ut+1/2at2)

```
#include<stdio.h>
#include<condio.h>
void main()
{
  int n,i,t;
  float s;
  clrscr();
  printf("how many times do you want repeat the process");
  scanf("%d",&n);
  for(i=1;i<=n;i++)
   {
     printf("Enter the u,a,t values");
     scanf("%d%d%d",&u,&a,&t);
     s=u*t+a*t*t/2.0
     printf("The distance travelled by a vehicle is %f", s);
   }
getch();
}
```

2. Nested for loop: In nested for loops one or more statements are included in the body of the loop. The number of interactions in this type of structure will be equal to the number of interactions in the outer loop multiplied by the number of interactions in the inner loop.

Example: Write a C program to illustrate an example based on nested for loops

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i,j;
  clrscr();
  for(i=1;i<=3;i++)
   {
      for(j=1;j<=2;j++)
```

```
    {
    printf("%d", i*j);
    }
  }
getch();
}
```

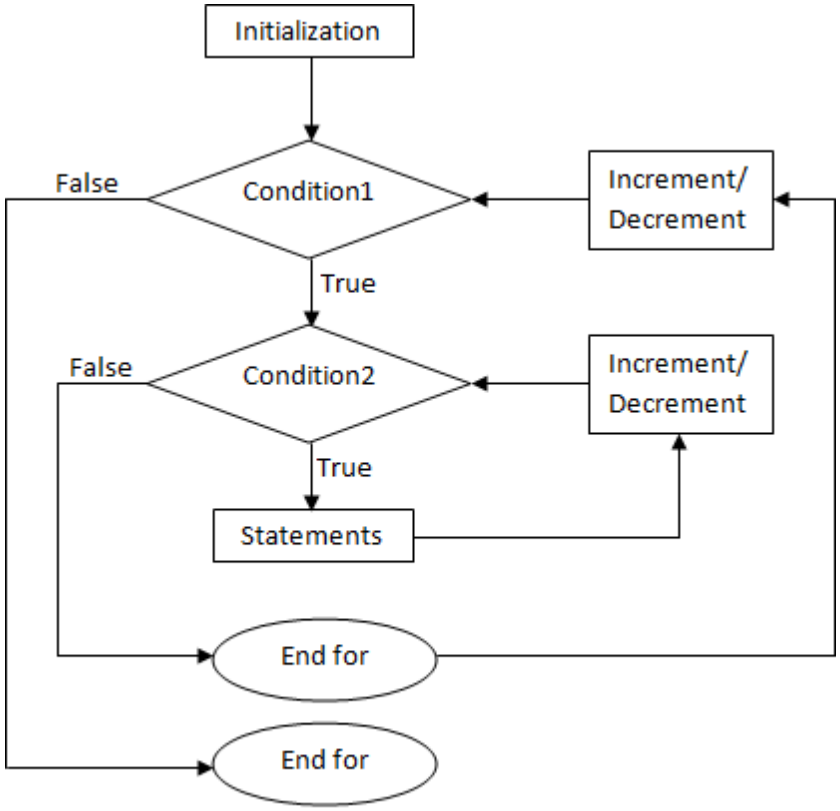


Fig: Flowchart for nested for loop

3. The while loop

Syntax: while (condition)

```
        {
          Body of the loop;
        }
```

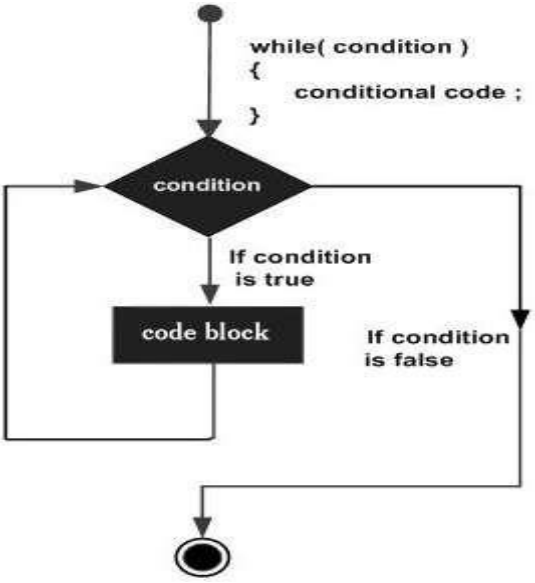
1. The test condition is evaluated and if it is true, the body of the loop is executed.

2. On execution of the body, test condition is repetitively checked and if it is true the body is executed.

3. The process of execution of the body will be continuing till the test condition becomes false.

4. The control is transferred out of the loop.

The block of the loop may contain a single statement or a number of statements. The same block can be repeated.



Example1: Write a c program to print the string "welcome" 9 times using while loop

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int x=1;
 while(x<10)
  {
    printf("\n welcome");
    x++;
   }
getch();
}
```

Example2: Write a c program to add 10 consecutive numbers starting from 1 use the while loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a=1,sum=0;
 clrscr();
 while(a<=10)
  {
    printf("%2d",a);
    sum=sum+a;
    a++;
   }
 printf("sum of the 10 numbers is %d", sum);
getch();
}
```

Example 3:Write a c program to find the factorial of a given number by using while loop

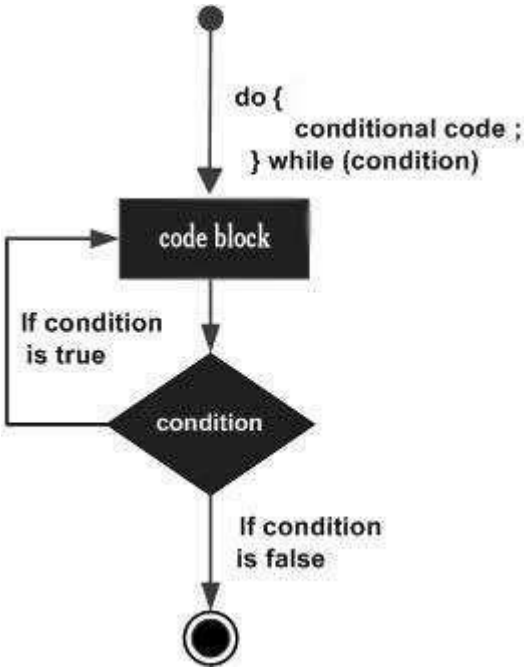
```
#include<stdio.h>
#include<condio.h>
void main()
{
  int n, i=1, fact=1;
  printf("Enter the number");
  scanf("%d",&n);
  while(i<=n)
   {
     fact=fact*i;
     i++;
   }
   printf("the factorial of a given number is%d",fact);
getch();
}
```

4. do-while loop

A do...while loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

Syntax:

```
do
{
   Statements;
 }
While (condition);
```



Example 1: Write a c program to find the factorial of a given number by using do-while

```
#include<stdio.h>
#include<condio.h>
void main()
{
  int n, i=1, fact=1;
  printf("Enter the number");
  scanf("%d",&n);
   do
   {
     fact=fact*i;
     i++;
   }
    while(i<=n);
   printf("the factorial of a given number is%d",fact);
getch();
}
```

Difference between while and do-while

While	Do-while
In the while loops the condition is tested following the statement and then the body gets executed.	The do-while the condition is checked at the end of the loop.
	The do-while loop will execute at least one time even if the condition is false.

5. The while loop within the do-while loop

Syntax:

```
do while(condition)
{
   Statements;
}
while(condition);
```

Example: Write a C program to use while statement in do-while and print values from 1 to 5

```
#include<stdio.h>
#include<conio.h>
void main()
{
      int x=0;
     clrscr();
   do  while(x<5)
        {
          x++;
          printf("%d",x);
        }
         while(x<1);
getch();  }
```

Output: 1 2 3 4 5