Decision control statements:

Control statements are the statements which are used to control the flow of execution of the instructions. The programs are basically executed sequentially and they are known as sequence control structure. It does not use decision making process. But a program may require to change the order of execution of statements or repetition of a group of statements.

The statements are:

1. Decision or selection control structure or Branching.
2. Case control structure.
3. Looping control structure or repetition.
4. Jumping statements.

Decision or selection control structure or Branching: it involves an execution of a group of instructions depending upon the result of a decision.

Simple If statement: most simple decision control statement.

```
Syntax : if(condition)
            {
              Statement blk
              }
```

The condition is placed in parenthesis. The statement block is executed if the condition is true otherwise the statement block is ignored.

Eg.

```
#
main()
{
 int a;
 printf("enter your age");
 scanf("%d", &a);
 if(a>=18)
 {
  printf("you are eligible to vote");
 }
 getch();
}
```

If-else statement: The if statement is executed when the statement block is true otherwise it does nothing. But often the statement block has to be executed when the condition is true or false. This can be solved by using if-else statement.

```
Syntax:  if(condition)
         {
           True stmt blk
         }
         Else
         {
           False stmt blk
         }
```

If condition is true the true statement blk is executed else false statement blk will be executed

Eg :

```
// program to check the entered number is +ve or -ve
main()
{
 int n;
 clrscr();
 printf("enter any number");
 scanf("%d", &n);
 if(n<0)
 {
  printf("the number is -ve");
 }
 else
 {
  printf("the number is +ve");
 }
 getch();
}
```

Nested if-else statement: if an entire if-else construct is within the if statement or within the else statement, this construct is called an nesting of if statement.

Syntax:

```
 If(cond1)
{
                 If(cond2)
                    Statement blk1
                 Else
                    Statement blk2
}
Else
  Statement blk3
```

Eg:

```
// print the biggest number of three numbers
#include<stdio.h>
#include<conio.h>
Main()
{
 Int a,b,c;
 Clrscr();
 Printf("\n enter three numbers");
 Scanf("%d%d%d", &a,&b, &c);
 If(a>b)
 {
   If(a>c)
     printf("\n biggest no = %d", a);
   else
     printf("biggest no = %d", c);
 }
 Else
 {
   If(b>c)
     printf("\n biggest no = %d", b);
   else
     printf("biggest no = %d", c);
 }
 Getch();
}

// biggest of three numbers using logical operators
#include<stdio.h>
#include<conio.h>
```

```
Main()
{
  Int a,b,c,;
  Clrscr();
  Printf("\n enter three numbers");
 Scanf("%d %d %d", &a, &b,&c);
 If((a>b) && (a>c) )
  Printf("\n biggest no = %d", a );
 If((b>a) && (b>c))
  Printf("\n biggest no = %d", b );
 If((c>a) && (c>b))
  Printf("\n biggest no = %d", c );
 Getch();
}
```

WAP to read a year and determine whether it is a leap year or not using nested if

Else –if ladder: The nested if-else statement becomes complex and an alternative is else-if ladder statement or multiple alternative if statement, and it is executed from top to bottom. Each conditional expression is tested and if found true only then its corresponding statement is executed and remaining ladder is then bypassed. If none of the condition is found true then the else part is executed.

Syntax

```
If(cond1)
        Stmt blk1
                Else if(cond2)
```

```
                    Stmy blk2
                            Else if(cond3)
                                    Stmt blk3
 Else
   Stmt of else
```

Eg:

```
//using else if ladder print the day of the week
#include<stdio.h>
#include<conio.h>
void main()
{
 int d;
 clrscr();
 printf("\n enter the day ofthe week");
 scanf("%d",&d);   3
 if(d==1)
  printf("\nDay is monday");
 else if(d==2)
  printf("\nDay is tuesday");
 else if(d==3)
  printf("\nDay is wednesday");
 else if(d==4)
  printf("\nDay is thursday");
 else if(d==5)
  printf("\nDay is friday");
 else if(d==6)
  printf("\nDay is saturday");
 else if(d==7)
  printf("\nDay is sunday");
 else
  printf("\nwrong choice");
 getch();
}
```

Case control structure: it is used to select one of several different courses of action. It is also called switch-case-default control structure. It select one out of multiple alternatives present in a program. It is a better construct than if-else if ladder. It has more flexibility and a clearer format than if- else if ladder. It can also be called as the replacement of multiple ifs.

Syntax:

```
switch (expression)
{
 case constant 1 :
        statement-block-1;
        break;
 case constant 2 :
        statement-block-2;
        break;
 case constant n :
        statement-block-n;
        break;
 default :
       statement-block-default;
```

where switch, case, default are the reserved words or keywords;
expression --> it must evaluate to an integer or character value constant-f it must be an int or char compatible value.
statement-block—> It is_a simple (only one) or compound statement (combination of more than one statements)
break—> is a reserved word that stops the execution with in the switch and control comes out of the switch construct

//using case control print the day of the week

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int d;
 clrscr();
 printf("\n enter the day of the week");
 scanf("%d",&d);
 switch(d)
 {
   case 1:
    printf("\n monday");
    break;
   case 2:
```

```
   printf("\n tuesday");
   break;
  case 3:
   printf("\n wednesday");
   break;
  case 4:
   printf("\n thursday");
   break;
  case 5:
   printf("\n friday");
   break;
  case 6:
   printf("\n saturday");
   break;
  case 7:
   printf("\n sunday");
   break;
  default:
   printf("\n wrong input");
 }
 getch();
}
```

WAP to read the values of three coefficients of a quadratic equation $ax^2 + bx + c = 0$. Find the roots of the equation and display them on your screen, specifying the nature of roots.

The roots of the quadratic equation can be computed by the following formulas

The roots x1 and x2 are :
(i) real and equal roots for b^2 -4ac = 0
(ii) real and unequal roots for b^2-4ac > 0
(iii) imaginary or complex roots for b2-4ac < 0

/* Roots of the quadratic equation */

```
#include <stdio.h>
#include <conio.h>
#include <math.h> /* for sqrt function */
void main( )
{
  int xl, x2, d, t, a, b, c, set;
  clrscr( );
  printf (" \n Enter coefficients : a, b, c");
  scanf ("%d%d%d", &a, &b, &c);
  d=b*b-4*a*c;
  t = 2 * a;
  if (d == 0)
   set = 0;
  else
  {
    if (d > 0)
     set = 1;
   else
     set = 2;
  }
  switch (set)
   {
    case 0 : printf (" \n Roots are real and equal");
      xl = -1* b/t ;
      x2 = xl;
      printf ( " \n Root 1 = % d \t Root 2 = %d", xl, x2);
      break;
    case 1 :
      print (" \n Roots are real and unequal");
      d = sqrt (d);
      xl = (-1* b + d)/t;
      x2 = (-1*b-d)/t;
      printf ("\n Rootl = % d \t Root 2 = % d", xl, x2);
      break;
    case 2 :
      printf ("\n Roots are imaginary or complex");
  }
  getch( );
}
```

Conditional Operator: C provides us a conditional operator (? :) which help in performing simple conditional operations.. It is also called as ternary operator because it operates on three values. It is used to do the operation almost similar to if-else construct.

Syntax: El ? E2 : E3

where El, E2 and E3 are expressions.

In the conditional operation, the expression El is tested, if El is true then E2 is evaluated otherwise the expression E3 is evaluated.

Eg: to find the greatest number of two numbers

c = (a > b) ? a : b   a=20   b=30

The expression a > b is evaluated. If a is greater than b then the value of a is assigned to c otherwise the value of b is assigned to c.

```
 (i) y = (x > = 20) ? 0 : 20;
(ii) result = ((i > j) ? (sum + i) : (sum + j)) ;
(iii) y = (((a > = 97) && (a < = 122)) ? 1 : 0);
(iv) (i = =1? prinft ("Monday") : printf ("Other day"));
```

WAP to demonstrate conditional operator to find the biggest of two given numbers.

```
/* Find the bigger of two numbers */
#include <stdio.h>
#include <conio.h>
void main( )
{
    int a, b, big;
   clrscr ( );
   printf("\n Enter two numbers");
   scanf ("%d %d", &a, &b);
   big=a>b?a:b;
   printf(" \n Bigger number = %d", big);
   getch( );
}
```

Output

```
Enter two numbers 10 20
Bigger number =20
```

WAP to find the biggest of three given numbers

```
/* Biggest of three numbers using conditional operator */
#include <stdio.h>
#include <conio.h>
void main( )
{
    int a, b, c big;
   cIrscr( );
   printf (" \n Enter three numbers");
   scanf ("%d %d %d", &a, &b, &c);                         E1? E2: E3
   big = a>b? (a > c ? a: c) : (b > c ? b : c);             a>b?a:b
   printf (" \.n Biggest number = %d", big);
   getch( );
}
```

WAP to check if entered character is **uppercase alphabet**

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char ch;
    clrscr( );
    printf "\n Enter a character");
    scanf ("%c", &ch);
    if (ch >= 'A' && ch <='Z')     ch>=65 && ch<=91
printf ("\n  character %c is uppercase alphabet", ch);
getch( );
```

Write a program to find whether the character entered is a capital letter, a small case letter, a digit or a special symbol.

Solution

```
#include <stdio.h>
#include <conio.h>
void main( )
{
   char ch;
   clrscr( );
   printf \n Enter a character");
   scanf ("%c '", &ch);
   if (ch > = 'A' && ch <='Z')
     printf ("An character  is a capital letter", ch);
   else
   {
       if (ch >= 'a' && ch <= 'z')
            printf (" \n character %c is small case letter", ch);
      else
      {
            if (ch >= '0' && ch <='9')
                printf (" \n character %c is a digit", ch);
           else
               printf ("\n character %c is a special symbol", ch);
         }
      }
  getch( );
}
```

Write a program to calculate percentage of marks of a student in five different subjects. If his percentage is greater than equal to 60 he got first division, if greater than equal to 50 second division, if greater than equal to 40 third division, otherwise fail.

```
#include <.stdio.h>
#include <conio.h>
void main( )
{
    int m1, m2, m3, m4, m5;
    float p;
    clrscr( );
    printf (“ \n Enter marks in 5 subjects");
    scanf ("%d %d%d %d %d", &m1, &m2, &m3, &m4, &m5);
    p= (m1 + m2 + m3 + m4 + m5)/ 5.0;
    if (p >= 60)
        printf (" \n Student got first division = %f” , p);
    else
    {
      if (p >= 50)
          printf (" \n Student got second division = %f', p);
     else if (p >= 40)
          printf (" \n Third division = %f’, p);
     else
          printf ("\n fail = %f’, p);
    }
   getch ( );
}
```

Write a menu driven program with the following options

1. Biggest of two numbers
2. Positive or negative
3. Factorial of a number
4. Exit

```
/* Menu driven program */
#include <stdio.h>
#include <conio.h>
void main( )
{
   int ch, a, b, n, fact, i,big;
```

```
clrscr( );
while( 1)
{
  clrscr();
  printf (" \n 1. Biggest of two nos.");
  printf (" \n 2. Positive or negative");
  printf (" \n 3. Factorial of a no.");
  printf (" \n 4. Exit");
  printf (" \n Enter your choice");
  scanf ("%d", &ch);
  switch (ch)
  {
     case 1 :
        clrscr( );
        printf (" \n Enter two numbers");
        scanf ("%d %d", &a, &b);
        big=a>b?a:b;
        printf ( " \n Bigger number of %d %d = %d", a, b, big);
        getch ( );
        break;
    case 2 :
        clrscr( );
        printf ( " \n Enter a number");
        scanf ("%d", &a);
        if (a >= 0)
              printf (" \n Number %d is positive", a);
       else
              printf (" \n Number %d is negative", a);
       getch( );
       break;
   case 3 :
          cIrscr( );
          printf (" \n Enter a number");
          scanf ("%d", &n);
          fact = 1;
          for (i = 1 ; n; i ++)
            fact = fact * i;
          printf (" \n Factorial of %d = %d", n, fact);
          getch( );
          break;
        case 4 :
          exit( );
        }
  }
  getch ();
}
```

Looping / Repetition/ Iteration Statement: Many problems require that a set of statements should be executed more than one time, each time changing the values of one or more variables, so that every execution is different from the previous one. This kind of repetitive execution of a set of statements in a program is known as interactive Loop. Looping involves repeating some portion of the program either a specified number of times or until a particular condition is being satisfied. There are three methods by which looping can be done in a program.

They are:

1. while statement
2. for statement
3. do-while statement

Flow of Control

Usually the flow of control is sequential i.e. one after the other or selective i.e. depending upon the result of the condition, a block of statements get executed. It also require to repeat a statement or a group of statements many times and this can be done by loop construct.

Counters

Loop control can be controlled by a special variable that keep track of the number of times the loop is executed. This special variable is known as a counter. The counter is increased by a fixed amount each time the loop is executed (usually 1). The counter is known as a loop control variable.

To set a counter user has to follow the following steps.

1. Initialize the counter by a starting value.
2. Increment the counter each time the loop gets executed.
3. Test the counter each time the loop gets executed, to see if the loop has been repeated a proper number of times.

The while loop: It is a repetitive control structure used in C. While loop is suited for the problems where it is not known in advance that how many times a statement or a set of statements *i.e.* statement block will be executed.

Syntax:

```
while (condition)
   statement - block
```

where while -> is a reserved word of C

condition --> is a constant, a variable or an expression

Statement block -> can be a simple or compound statement

The statement block is executed repeatedly till the condition evaluates true value otherwise statement will terminate and control will be passed to the statement following while loop.

The sequence of operations in a while loop is as follows

1. Test the condition.

2. If the condition is true then execute the statement block and repeat step 1.
3. If the condition is false, leave the loop and go on with the rest of the program.

Program to display the consecutive digits from 0 to *n using while loop.*

```
/* To display digits from 0 to n */
#include <stdio.h>
#include <conio.h>
void main( )
{
  int n, d = 0;
  clrscr( );
  printf ("\n Enter last digit");
  scanf ("%d", &n);   10
  while (d< = n)     0<=10
  {
    printf ("%d \t", d);   0     1     2     3 4 5 6,,,10
   d ++;
  }
   getch( );
}
```

Output.

```
Enter the last digit 10
0 1 2 3 4 5 6 7 8 9 10
```

Write a program which computes a^b where a and b are of type real and integer respectively.

Solution

Eg $3^5 = 3 * 3 * 3 * 3 * 3$

```
/* compute ab * /
#include <stdio.h>
#include <conio.h>
void main( )
{
  float a, pow;
  int b, bl;
  clrscr( );
  printf ("\n Enter the values");
  scanf ("%f %d", &a, &b);
  pow = 1.0;
  bl = b
 while (b >= 1)
 {
   pow = pow * a;
   b - -;
 }
   printf ("\n %f raise to power %d is = %f', a, b1, pow);
 getch( );
}
```

1234=10

4321

Program to find the factorial of a number entered through keyboard using while loop.

Factorial of a number can be calculated as

5! = 5 * 4 * 3 * 2 * 1 = 120

n! = n * (n-1) * (n-2) * 1

```
/* Factorial of a number */
#include <stdio.h>
#include <conio.h>
void main( )
{
   int n, i = 1 ;
   long fact = 1;
   cIrscr( );
   printf ("\n Enter a number");
   scanf ("%d", & n) ;
   while (i < = n)
   {
     fact = fact * i;
    i ++;
   }
   printf ("\n Factorial of %d = %ld", n, fact);
 getch( );

}
```

Write a program to accept an integer and reverse the integer. Display both the numbers on your screen.

```
/* Program to accept an integer and reverse it */
#include <stdio.h>
#include <conio.h>
void main( )
{
  long rev = 0, num, rem,  n;
  clrscr( );
  printf ("\n Enter the number");
  scanf ("%ld", &n);    1234
  num = n;
  while (num > = 0)
  {
   rem = num %10;
   num = num /10;
   rev = rev * 10 + rem;
  }
  printf ("\n The actual number = %ld", n);
  printf ("\n The reverse number = %ld", rev);
  getch( );
}
```

Output.

```
Enter the number 1234
The actual number = 1234
The reverse number = 4321
```

for loop: It is a count controlled loop in which the program knows in advance how many times the loop is to be executed.

The statement of the loop is specified in a single line:

(a) Setting a loop counter to an initial value.
(b) Testing the condition about loop counter.
(c) Increasing or decreasing the loop counter's value.

Syntax

```
for (initialization; expression ; increment or decrement)
   statement - block
```

where for --> is a reserved word
initialization -> is usually an assignment expression where in a loop control variable is initialized.
expression --> is a conditional expression required to determine whether the loop should continue or be terminated.
increment or decrement —> it modifies the value of the loop control variable by a certain amount.
statement block —> can be a simple or compound statement.

Program to print first 10 multiples of an integer n where n is to be entered by the user.

```
/* Print first 10 multiples of an integer n */
#include <stdio.h>
#include <conio.h>
void main( )
{
   int n, i;
  clrscr( );
  printf ("\n Enter the integer number");
  scanf ("%d", &n);                                    2*1=2
  for (i = 1; i <= 10; i ++)                                   2*2=4
      printf ("\n %d * %d= %d", n, i, n * i);
  getch( );
}
```

Output.

```
Enter the integer number 9
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9* 10 = 90
```

Write a program to calculate the average of n given numbers.

```
/* Calculate the average of n given numbers */
#include <stdio.h>
#include <conio.h>
void main( )
{
   int i, n;
   float x, sum = 0, avg;
  clrscr( );
  printf ("\n How many numbers");
  scanf ("%d", &n);          5
  for (i = 1; i < = n; i++)
  {
    printf ("\n Enter a number");
    scanf ("%f", &x);
    sum + = x;   // sum=sum+x
  }
    avg = sum/n;
    printf ("\n The average = %f", avg);
   getch( );
}
```

Write a program to find and print the n terms of Fibonacci series.
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,

```
/* Fibonacci series */
#include <stdio.h>
#include <conio.h>
void main( )
{
  int a, b, c, n, i;
  a = 0;
  b = 1;
  clrscr( );
  printf ("\n Enter the number of terms");
  scanf ("%d", &n);
  if (n == 1)
    printf ("\n %d \t", a);
  else
    if (n >= 2)
    {
      printf ("\n %d \t %d", a, b);
      for (i = 3; i < = n; i ++)
      {
         c = a + b;
         printf (" \ t %d", c);
```

```
        a = b;
        b = c;
     }
    }
  getch( );
}
```

Output

```
Enter the number of terms 10
0 1 1 2 3 5 8 13 21 34
```

do-while loop: do-while loop is repetitive control structure provided by C. It is almost same as that of while loop and is used for the problems where it is not known in advance that how many times a statement block will be executed. Since the condition is tested after the execution of the loop, it is also known as post-test loop. This loop is executed at least once.

Syntax

```
do
{
   statement - block
}
while (condition);
```

where
 do is a reserved word
 statement - block can be a simple or a compound statement
 while is a reserved word
 condition -› is a constant, a variable or an expression.

The statement block is executed as long as the condition remains true. As soon as condition becomes false, the control leaves the loop and goes to the next immediate statement after the while.

Program to calculate factorial of a given positive integer *n using do while.*

```
/* Factorial of a number */
#include <stdio.h>
#include <conio.h>
void main( )
{
  int n, i = 1;
  long fact = 1;
  clrscr( );
  printf ("\n Enter a number");
  scanf ("%d", &n);   -1
  do
{
   fact * = i ;  //fact=fact*I;
   i ++,
 }
 while (i <= n);     2<=-1
 printf ("\n Factorial of %d = %d is", n,fact);
 getch();
}
```

Calculate the sum of first n numbers using do-while loop.

```
/* sum of first n numbers */
#include <stdio.h>
```

```
#include <conio. h>
void main( )
{
  int sum, n, i = 0;
  sum = 0;
  clrscr( );
  printf ("\n Enter the value of n");
  scanf ("%d", &n);
  do
{
   sum + = i;
   i++;
}
while (i < = n);
printf (" \ n Sum of first %d numbers = %d", n, sum);
getch( );
}
```

Output.

```
Enter the value of n 10
Sum of first 10 numbers = 55
```

Comparison of while and do-while construct

While	**do-while**
The statements of loop may not be executed at all when the conditions is false.	The statement of do- while loop must be executed at least once even though the condition is false
While loop is a pre test loop.	It is a post test loop.

The loop terminates when the loop condition becomes false	As lons as the condtion is true, the computer keep executing the loop

```
4. The syntax is                    4. The syntax is
while (condition)                    do
{                             {
 statement block;                      statement block;
}                             }
                              while (condition) ;
```

Jumping statements
Jumping statements are used to transfer the control from one part of the program to another part.
Jumping statements are:
1. The break statement
2. The continue statement
3. The goto statement

The break statement
The break statement is always used inside the body of the switch statement. It can also be used in any of the loop statement for transfer of control from the loop to the statement immediately following the loop.
In switch statement, it is used as the last statement of the statement block of every case except the last one. It transfers the control out of switch statement and the program continues its execution from the statement following switch.
In while, for, do-while loops, it is always used in conjuction with if statement. When executed it transfers the control out of the particular

loop and the execution of the program continues after looping block.

Demonstration of break statement

```
#include <stdio .h>
#include <conio.h>
void main( )
{
 int i;
 clrscr( );
 for (i = 1; i < = 10; i ++) //  i=i+2
 {
  printf ("\n %d", i);
  if (i == 3)
  {
   printf ("\n OK Bye");
   break;
  }
   printf ("\n Next");
 }
 getch( );
}
```

Output.

```
1
Next
2
Next
3
OK Bye
```

Write a program that reads in a number and a single digit. It determines whether the number contains the digit or not.

```
/* A digit is entered in a number or not */
#include <stdio.h>
#include <conio.h>
void main( )
{
 long n, n1;
 int d, temp, flag;
 clrscr( );
 printf ("\n Enter the number");
 scanf ("%ld", &n);    1234
 n1 = n;
 printf ("\n Enter the digit");      3
 scanf ("%d", &d);
 flag = 0;
 do
{
   temp = n % 10;
   if (d == temp)
  {
     flag = 1; /* digit is found */
     break;
   }
  n = n/10;

 }
 while (n! = 0);
 if (flag == 1)
     printf (" \ n Digit %d is present in %ld", d, n1);
  else
     printf (" \ n Digit %d' is not present in %ld", d, n1);
 }
 getch( );
}
```

Output. First run

```
Enter the number 1234
Enter the digit 5
Digit 5 is not present in 1234
```

Second run

```
Enter the number 7256
Enter the digit 2
Digit 2 is present in 7256
```

continue statement: The continue statement is used inside the body of the loop statement. The break statement terminates the loop, while the continue statement does not terminate the loop but transfers the control back to the first statement. The continue statement is associated with if statement. The continue statement transfers the control to the beginning of the loop, bypassing the statements which are not yet executed.

Demonstration of continue statement.

```
for (i = 1; i < = 3; i ++)
{
  printf ("%d \n", i);
  if (i == 2)
    continue;
  printf ("Bottom of loop \n");
}
printf ("Out of loop");
```

Output.

```
1
```

```
Bottom of loop
2
3
Bottom of loop
Out of loop
```

Write a program that reads n integers and computes the sum of positive integers out of them.

```
/* Sum of positive integers only out of n integers */
#include <stdio.h>
#include <conio.h>
void main( )
{
  int num, n, i, sum = 0;
  clrscr( );
  printf ("\n Enter the number of integers");
  scanf ("%d", &n); 5
  for (i = 1; i <= n; i++)
  {
    printf ("\n Enter a number");
    scanf ("%d", &num);
    if (num < 0)
      continue;
   sum = sum + num;
  }
  printf (" \ n The sum of positive integers = %d", sum);
  getch( );
}
```

Output.

```
Enter the number of integers 5
Enter a number 10
Enter a number - 50
Enter a number 20
Enter a number 30
Enter a number - 20
The sum of positive integers = 60
```

goto statement: It is an unconditional transfer of control which means that the sequence of execution will be broken without performing any test and the control will be transferred to some statement other than the immediate next one.

Syntax

```
    goto <statement label>
```

where goto -> is a reserved word
<statement label> -> is an identifier to label the target statement
The goto statement causes the control to be transferred to the statement whose label is specified in the goto statement.
E.g.

```
goto next;
next :
```

Statement label A statement label is an identifier which can be placed before any C statement followed by a colon : as shown below

```
      next : C = A + B;
```

Thus next is a label attached to the statement C = A + B and the control of execution can be transferred to this statement by the following

```
goto next ;
```

Demonstration of goto statement

```
scanf ("%f", &x);
while (x < = 50)
```

```
{
 if (x < 0)
   goto error;
   scanf ("%f", &x);
error :
printf ("Negative value error");
```

Demonstration of goto statement

```
void main( )
{
   int i, j;
   for (i = 1; i <= 2; i ++)
   {
     for j= 1; j <= 3; j++)
     {
       if (i == 2 && j == 3)
            goto out;
      else
            printf (" \ n %d %d", i, j);
     }
    out : printf (" \ n I am out of loops now");
 }
```

Write a program to calculate average for several different lists of numbers.

```
/* Calculate average for different lists of numbers */
#include <stdio.h>
#include <conio.h>
void main( )
```

```
{
    int n, i, j, lists;
    float x, avg, sum;
    clrscr( );
    printf ("\n How many lists");
    scanf ("%d", &lists);
    for (i = 1; i <= lists; i ++) /* Outer loop for number of lists */
    {
      sum = 0;
      printf ("\n List number %d \n How many numbers ?", i);
      scanf ("%d", &n);
      printf ("\n Enter the value of %d numbers", n);
      for (j = 1; j < = n; j ++) /* Inner loop calculating sum of n numbers
for all lists */
      {
         scanf ("%f', &x);
         sum + = x;
      }
      avg = sum/n;
      printf ("\n The average is % f\n", avg);
   }
getch( );
}
```
Output.

```
How many lists 2
List number 1
How many numbers ? 5
Enter the value of 5 numbers 5 10 15 20 25
The average is 15.000000
List number 2
How many numbers ? 3
```

```
Enter the value of 3 numbers 1 3 5
The average is 3.000000
```

Write a program to add first n terms of the following series
1/ 1!+2/ 2!+3/ 3!+ …………**n/n!**

```
/* Sum of a series */
#include <stdio.h>
#include <conio.h>
void main( )
{
   int i, n, j, fact;
   float sum = 0;
   clrscr( );
   printf ("\n Enter number of terms");
   scanf ("%d", &n);   5
   for (i = 1; i < = n; i ++)
    {
      fact = 1;
      for (j = 1; j < = i; j ++)
           fact * = j;
      sum = sum + (float) i/fact;
  }
  printf ("\n Sum of series = %f", sum);
  getch( );
}
```
Output.

```
  Enter number of terms 3
  Sum of series = 2.5000000
```

Write a program to print the figure

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

```
/* Print the figure of stars */
#includc <stdio.h>
#include <conio.h>
void main( )
{
   int i, j, n;
   clrscr( );
   printf ("\n How many rows");
   scanf ("%d", &n);
   for (i = 1; i <= n; i ++)
  {
    for (j = 1; j < = i; j ++)
          printf ("*");
    printf ("\n");
  }
getch( );
}
```

Write a program to print the structure.

```
         1
       232
      34543
      4567654

#incldue <stdio.h>
#include <conio.h>
void main( )
{
   int i, j, k, 1, n, a, b;
   clrscr( );
   printf ("\n Enter the number of rows");
   scanf ("%d", &n);
   for (i = 1; i <= n; i ++)
   {
     a = i;
     b = 2 * i - 2;
     for (j = 1; j < = 2* (n - i); j++)
        printf (" ");
     for (k = 1; k< = i; k++)
     {
        printf (" %d", a);
        a ++;
     }
     for (1= 1; 1 < i; 1 ++)
     {
          printf (" %d", b);
          b - -;
  }
     printf ("\n");
   }
  getch();
}
```