1.2 Algorithm and Flow Chart

1.2.1 Algorithm: Algorithm is a step by step procedural description of the programming logic where each step is numbered in a hierarchical order. In algorithm, number of steps must be finite and every step must be complete and error free. The steps must specify the input and output as per the requirement of the user.

Characteristics of Algorithm:

a. **Finiteness**: An algorithm must terminate after a finite number of steps and further each step must be executable in finite amount of time.
b. **Definiteness**: Each steps in algorithm must be precisely defined, the action to be carried out must be rigorously and unambiguously specified for each case.
c. **Inputs:** An algorithm must have zero or more but only finite number of inputs.
d. **Output**: An algorithm must have one or more outputs.
e. **Effectiveness:** An algorithm should be effective. This means that each of the operation performed in an algorithm must be sufficiently basic that it can, in principle be done exactly and in a finite length of time, using pencil or pen and paper. It may be noted that the finiteness condition is a special case of effectiveness. If a sequence of steps is not finite, then it cannot be effective also.

Examples: **Write an algorithm for the following**

1. **To read the percentage marks of the student and display PASS or FAIL.**

   Algorithm:

   Step 0: Start
   Step 1: Read the percentage of Marks obtained by the student
   Step 2: if (Marks>35)
   Print PASS
   else
   Print FAIL
   Step 3: Stop

2. **To determine whether the given number is odd or even.**

   Algorithm:
   Step 0: Start
   Step 1: Read the number say n.
   Step 2: if (n%2==0)
   Print EVEN
   else
   Print ODD
   Step 3: Stop

3. **To compute the Volume and Surface area of the Sphere.**

   Algorithm:

   Step 0: Start
   Step 1: Read the radius of the sphere say r.
   Step 2: Define the Constant PI = **3.14159**
   Step 3: Compute the Volume V = 4/3*PI*(r*r*r)
   Step 4: Compute the Surface Area S = 4*PI*(r*r)
   Step 5: Stop

4. **To find the sum of first n natural numbers.**

   Algorithm:

   Step 0: Start

Step 1: Initialize sum = 0
Step 2: Read the value of n
Step 3: Set counter i =1
Step 4: Compute the sum = sum + i**;**
Step 5: Increment i = i+1
Step 6: Repeat step 4 and 5 until i<=n
Step 7: Print Sum
Step 8: Stop

5. **To calculate the factorial of a given number.**

Algorithm:

Step 0: Start
Step 1: Initialize product, p=1 as special case (0!=1)
Step 2: Read the value of n whose factorial is to be calculated
Step 3: Set counter i =1
Step 4: Compute the product p = p*i**;**
Step 5: Increment i = i+1
Step 6: Repeat step 4 and 5 until i<=n
Step 7: Print the product p as factorial of n
Step 8: Stop

6. **Algorithm to swap the contents of two variables using third variable**.

Algorithm:

Step 0**:** Start
Step 1: Read a, b;
Step 2: [Exchange a & b]
temp =a;
a =b;
b=temp;
Step 3: [Output the result]
Display a b.
Step 4: Stop.

7. **Algorithm to check whether a given number is positive or negative**.

Algorithm:

Step 0**:** Start
Step 1: Read n.
Step 2: [Check whether the number is +ve or –ve].
if(n<0)
Display “The number is negative”
else
Display “The number is positive”.
Step 3: Stop.

8. **Algorithm to find the area of the triangle**.

Algorithm:

Step 0**:** Start
Step 1: [Input 3 sides of the triangle] i.e. Read a,b,c.
Step 2: [Compute the value of s].
s = (a+b+c)/2.
Step 3: [Compute the area of the triangle].
area=sqrt(s*(s-a)*(s-b)*(s-c))
Step 4: Display area.
Step 5: Stop.

9. **Algorithm to find the sum of first N natural numbers**.

Algorithm:

Step 0**:** Start
Step 1: Read n.
Step 2: Initialize sum = 0
Step 3: Find the sum of all terms
for i = 1 to n in steps of 1 do
sum = sum + i
end for.
Step 4: Display sum.
Step 5: Stop.

10. **Algorithm to find the biggest of 3 numbers**.

Algorithm:

Step 0**:** Start
Step 1: Read a,b,c.
Step 2: [Comparison]
if((a>b) and (a>c)) then
display a is greatest.
else if((b>a) and (b>c))then
display b is greatest.
else display c is greatest.
Step 3: Stop.

11. Algorithm to swap the two integers without using third variable

Algorithm:

Step 0**:** Start
Step 1: Read a, b;
Step 2: [Exchange a & b]
a = a + b;
b = a - b;
a = a – b;
Step 3: [Output the result]
Display a b.
Step 4: Stop.

12. Algorithm to find the area and circumference of the circle.

Algorithm :

Step 0 : Read the radius of circle r
Step 1 : Initialize the Pi Value
Step 2 : Compute Area = Pi*r*r ;
Step 3: Compute Circumference = 2*Pi*r;
Step 4 : Display Area and Circumference

1.2.2 **Flowchart:** Flowchart is a graphical representation of an algorithm**.** Flowcharts use special shapes to represent different types of actions or steps in a process. Lines and arrows show the sequence of the steps, and the relationships among them. In flowchart symbols or shapes are used to represent the data flow, operations, process and recourses used to complete a given task. Table1.1 below shows the commonly used flowchart symbols and their purpose .

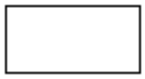
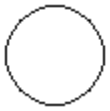
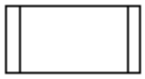
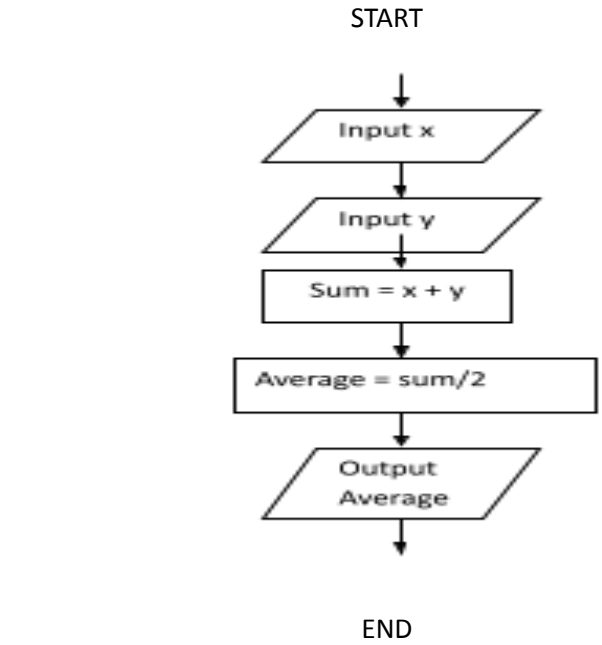
Flowchart Symbol	Purpose
	**Start/End:** The terminator (Ellipse like) symbol is used to represent the starting or ending point. It usually contains the word "Start" or "End."
	**Action or Process :** A rectangular box is used to represent a process defined in an algorithm
	**Decision:** A diamond symbol is used to represent the decision or branching point.
	**Input/output:** Parallelogram symbol is used to represent an input or output operation.
	**Connector:** Circle symbol is used to connect one point of flow to another point.
	**Flow Line:** Used to show the direction of a control flow and to connect the various flowchart symbols.
	**Subroutine:** The symbol is used to represent the sub functions in a given program.
	**Looping:** The symbol is used to represent the looping or repetition of a given set of statements.

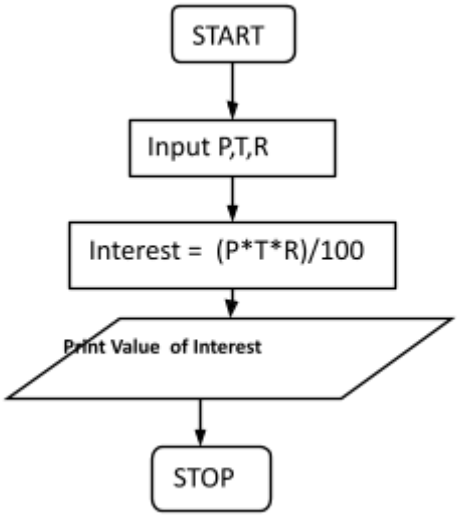
Table 1.1 : Flowchart Symbols

Examples: Write a flowchart for the following

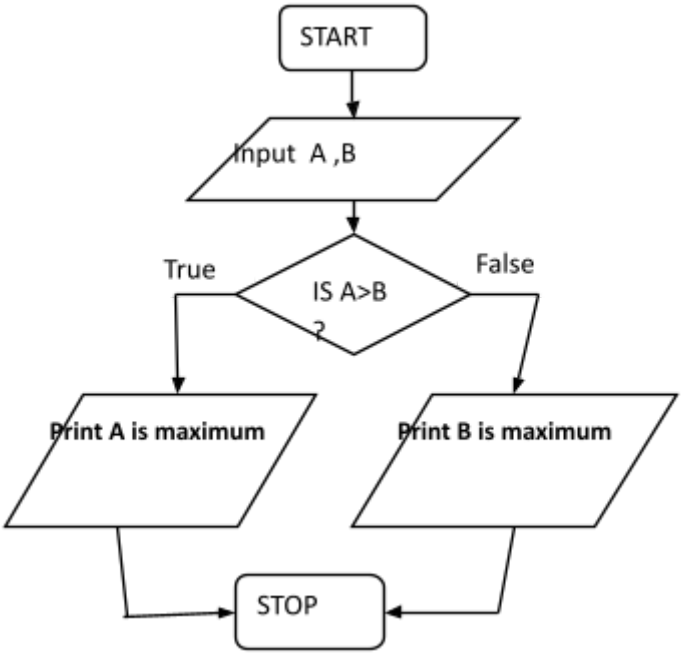
1. **To find the average of two numbers**



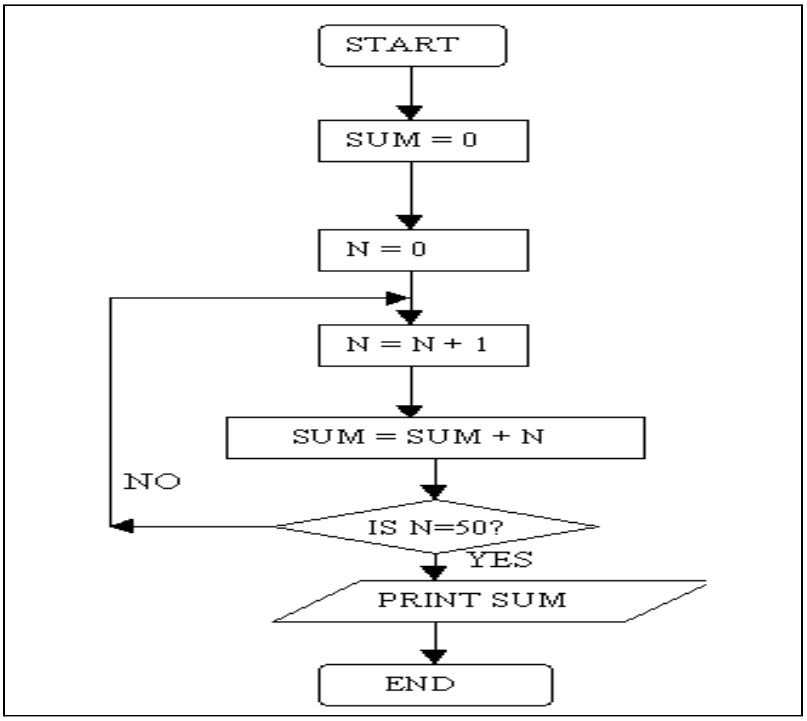
2. **To find the simple interest or interest given the value of P,T and R.**



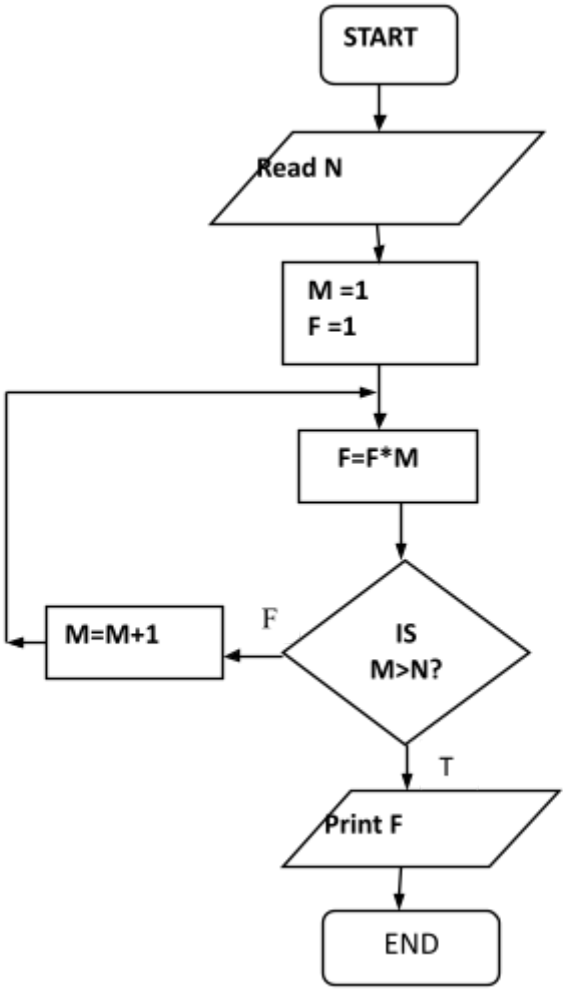
3. **To find the maximum of two numbers**



4. **To find the sum of first 50 natural numbers.**



5. To find the factorial of a given number N.



1.3: Pseudocode Solution to Problem: Pseudocode is a problem solving tool which consists of statements written in English and C language. It is a series of steps which describes what must be done to solve a given problem. It is used for developing a program and it is constructed before writing a program.

It is an informal high level description of program mixed with natural descriptions and mathematical notation. It does not have any standard syntax. Pseudocode is used to solve a problem and develop a C program as illustrated below:

Consider a problem of *"Displaying a list of numbers and its squares from 4 to 9".*

Method: The method to be used to solve the above problem is: *Start with the first number 4 and compute its square i.e. 16. Print the number and its square. Then do the same for remaining numbers (5, 6, 7, 8 and 9).* The above method to solve the problem can be put in the form of pseudo code as given below:

PseudoCode:

1. **Start** with the number 4
2. Compute its square (i.e. 4*4=16)
3. Print the number and its square (i.e. 4 and 16)
4. Do the same for each of the other number from 5 to 9.
5. **End**

Program: The above pseudo code can be utilized to develop the following C program

```
#include<stdio.h>
#include<conio.h>
main()
{   /*n takes  the values from 4 to 9 and s for storing squares of those values */
                int    n,s;
                 clrscr();
                 n = 4;
                 s = n*n;  /* s is the square of the number */
                 printf(" %d %d\n",n,s);
                 n = 5; s=n*n;
                 printf(" %d %d \n",n,s);
                 n = 6; s=n*n;
                 printf(" %d %d \n",n,s);
                 n = 7; s = n*n;
                 printf(" %d %d \n",n,s);
                 n=8; s =n*n;
                 printf(" %d %d \n",n,s);
                 n=9; s =n*n;
                 printf(" %d %d \n",n,s);
                 getch();
}
```

Characteristics of Pseudocode

- Composed of a sequence of statements or steps
- Statements are written in simple English and C statements.
- Each statement is written on a separate line
- Each statement in pseudocode expresses just one action for the computer.
- There is no fixed syntax.

Examples: Write a Pseudocode for the following:

1. **To read the percentage marks of a student and determine whether the student has passed or not.**

   Pseudocode :

   Begin
   Read Marks
   if marks>=40
   print result: PASS
   else
   print result : FAIL
   End

2. **To read the number and determine whether the number is even or odd number.**

   Pseudocode:

   Begin
   Read the number say **n**
   if (n%2= =0)
   print : “The number is Even”.
   else
   print : “The number is ODD”.
   End

3. **To calculate the volume and surface area of the sphere**.

Pseudocode :

Begin
Read the Radius of the sphere say R
Define PI = 3.142
Compute Volume = (4/3)*PI*(R*R*R)
Compute Surface_Area = 4*PI*(R*R)
PRINT Volume and Surface_Area
End

4. **To find the sum of two numbers**

Pseudocode :

Begin
Read two numbers a,b
Compute Sum = a + b
Print Sum
End

5. **To swap two numbers using temp variable.**

Pseudocode

Begin
Read a,b
Print a and b before swapping
temp = a;
a = b;
b = temp;
Print a and b
End

Program : Program is a set of instructions arranged in sequence used to guide the computer to solve a given problem. C program is a set of programming instructions written in C programming language (ANSI C Version) .

1.4 Basic Concepts of a C program: The lists of basic concepts of a C program are as follows:

1. The Basic Structure of a C program
2. Comments and Programming Style
3. The Program Header
4. The body or Action portion of the program

1.4.1 The Structure of a C program: The general basic structure of a C program is shown below:

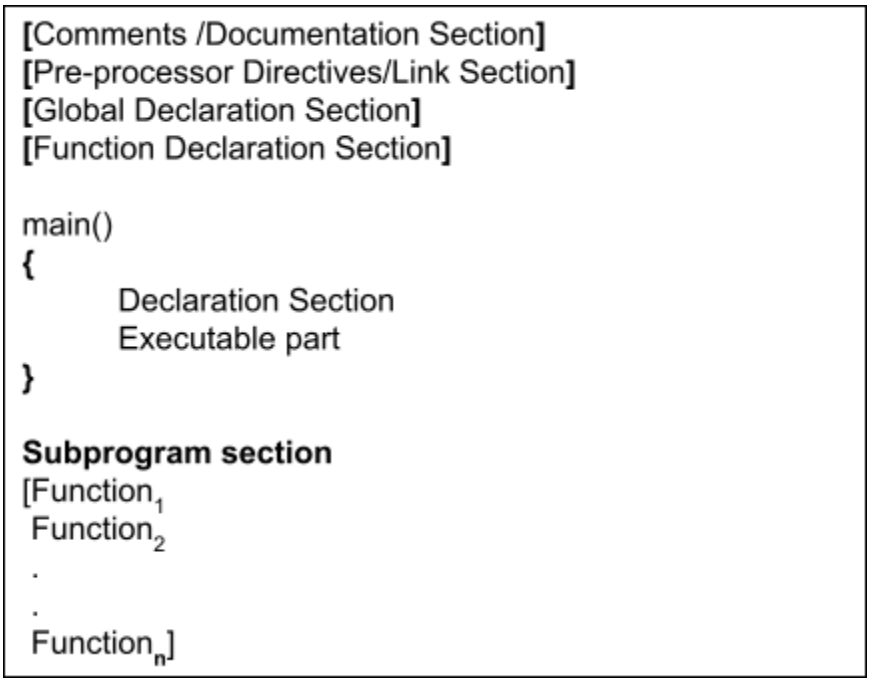


Fig 1.1: Generic Structure of a C program

The Documentation /Comments Section consists of a set of comment lines giving the short description /purpose of the program or any statement of the program and other details.

The Link/Pre-processor Section provides instructions to the compiler to link functions from the system library.

The Definition Section defines all symbolic constants.

The Global Declaration Section: In this section variables are declared outside the main function and these variables can be accessed by all functions.

The Function Declaration Section: Here the prototype declarations of sub functions are declared.

The main( ) function: Every C program must have one main function section. This section contains two parts, declaration and executable part.

Declaration Part declares all the variables used in the executable part.

There should be at least one statement in the **executable part** which contains instructions to perform certain task. The declaration and executable part must appear between the opening and closing braces. All statements in the declaration part should end with the semicolon.

The **Subprogram Section** contains all the user defined functions that are called in the main function. The example given below illustrates the structure of C program.

Example :

```
1  /*Documentation Section: program to find the area of circle*/
2  #include <stdio.h>/*link section*/
3  #include <conio.h>/*link section*/
4
5  #define PI 3.14     /*definition section*/
6  float area;         /*global declaration  section*/
7  void main()
8  {
9
10          float     r;        /*declaration part*/
11          clrscr();
12          printf("Enter the radius of the circle\n");   /*executable part starts here*/
13          scanf("%f",&r);
14          area=PI*r*r;
15
16          printf("Area of the circle=%f",area);
17          getch();
18 }
```

1.4.2 Comments and Programming Style: Comments are programming constructs which are used to describe briefly the purpose of each statement, set of statements or entire program.

Uses of Comments: Comments (documentation/ short descriptions) are written along with program for the following reasons:

a. To read and understand the logic of the program

b. To understand the semantics of the program

c. To maintain and modify the program

Comments or documentation begins with /* and ends with */. The symbols /* and */ are called *comment delimiters* because they delimit or mark the beginning and end of the comment.

Examples:

```
/* Program Written  by Palguni , USN : 2SD12345 on 1-12-2015 */
/*The function avg(t1,t2) determines the average two tests */
// To Read the Value of n
// Variable strength represents the strength of class
/*To Compute the perimeter of the rectangle*/
```

Programming Style: The comments can be used *anywhere inside* the program file. It can be used inside the main function or outside the main function. In order to describe the entire program the comments are written at the top of the program before header files. In order to describe the statement or set of statements the comments are included either at the top or at the right side of the statements.

Following example illustrates the usage of comments in a C program:

```
/*Program to accept the number and display the number */
#include<stdio.h>  /*Header Files*/
#include<conio.h>
main() /* Beginning of the Program */
 {       int    n; /* Declaration */
         clrscr();
          printf("\n Enter the number\n");
          scanf("%d",&n) ; /* Reading the number from the keyboard */
          /*Displaying the number entered */
         printf("\n The Entered number is %d\n",n);
         getch();
}  /* End of the Program */
```

1.4.3 The program header: The program header consists of the following sections

a) Preprocessor directives
b) Global declaration and definitions
c) Main program header

a) Preprocessor directives: The preprocessor directives are the statements which start with symbol #. These statements instruct the compiler to include some of the files in the beginning of the program. For example, #include<stdio.h>, #include<math.h> and #include<conio.h> are some of the files that the compiler includes in the beginning of the program. The #include<stdio.h> statement tells the compiler to include the standard input/output library or header file. This file has the function definition for built in functions like printf (), scanf() ,etc.

Using the preprocessor directives the user can define the constants also. For example

```
#define   MAX   1024
#define   PI      3.1417
```

Here MAX and PI are called symbolic constants.

b) Global Declaration and function definitions: The variables that are declared before all the functions are called global variables. The global variables can be accessed by all the functions including main function. As we declare the global variables, the functions are also declared here as illustrated below:

Example:

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

/* Demonstrating Global variables and function */

add_numbers (  );              /* Function declaration */
int  value1,value2, value3;     /* Declaration of  global variables */

add_numbers( )
{
      int    result2;    /* Declaration of local variable*/
      value1=5;
      value2=3;
      value3=2;
      result2 = value1 + value2 + value3;
      printf (" Result 2 = %d \n",result2);
}
  main()
{
        int    result1;
        clrscr ( ) ;
        value1 = 10;
        value2 = 20;
        value3 = 30;
        result1 = value1 + value2 + value3;
        printf("Result 1 = %d\n",result1);
        add_numbers();
        getch();
}
```

Output :

```
Result 1 =  60
Result 2 = 10
```

c) main program header: In C program the execution always starts from the main( ) function and it can be written as follows :

```
#include<stdio.h>
#include<conio.h>
main()                          /* the main statement*/
{   --------
}
```

The main statement instructs the compiler that execution always starts from function main and it is called *main program header*. The pair of brackets denoted by **( )** must follow after the main. Every C program must have only one main function.

1.4.4 The Body or Action Portion of the Program: Inside every C main program after the header or top line is a set of braces **{** and **}** containing a series of C statements which comprise the body. This is called the action portion of the program.

The body of the program contains two essential parts:

- Declaration Section
- Execution Section

Declaration Section: The variables that are used inside the main function or sub function should be declared in the declaration section. The variables can also be initialized during declaration. For example consider the declarations shown below:

```
main ()
{
  int   sum = 0;
  int        a;
  float      b;

   ----------
}
```

Declaration of the variables

Here the variable sum is declared as an integer variable and it is also initialized to zero. The variable **a** is declared as an integer variable whereas the variable **b** is declared as a floating point variable.

Executable Section: This section includes the instructions given to the computer to perform a specific task. An instruction may represent an expression to be evaluated, input/output statement etc. Each executable statement ends with "**;**". The instructions can *be input or output statements, simple statements such as if statement, for statement etc*.

Example:

```
main()
{
 -------

        printf("\n Enter the values of a and b\n");
        scanf ("%d %d",&a,&b);
        sum =a+b;
        printf(" \n The sum =%d \n", sum);
}
```

Executable Statements

Compiling and Running a C program

1. Compiling and Running a C Program on Ubuntu Linux :

Step 1: Open or create file using the command :

gedit filename.c

Step 2: Write a program and save the file.

Step 3: Compile the simple programs using the command :

cc filename.c

(Note : one can also use gcc filename.c)

If the program contains math functions use the following command

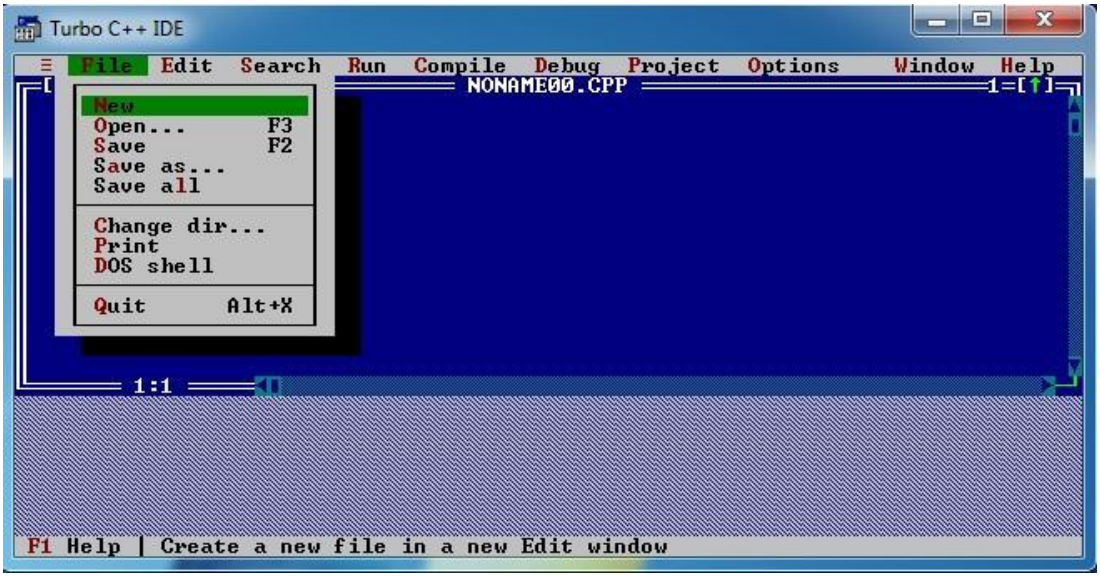
cc filename.c -lm

(Note : once can also use gcc filename.c –lm)

Step 4: Execute the program using the command : **./a.out**

2. Compiling and Running a C program on Windows Turbo C:

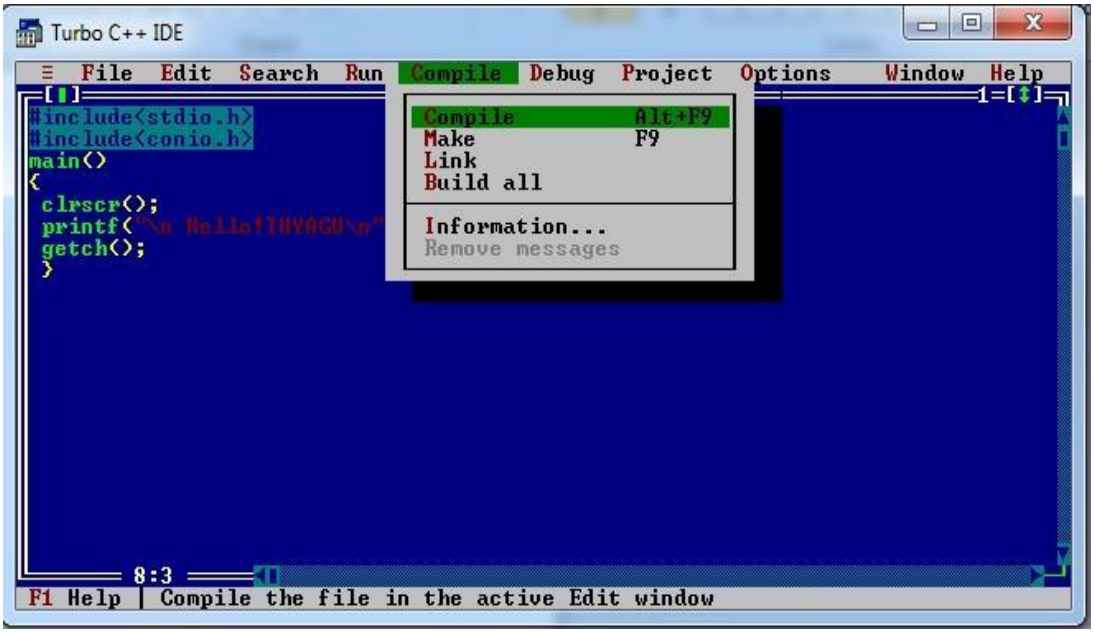
Step 1: Locate the TC.exe file and open it. You will find it at location C:\TC\BIN\.



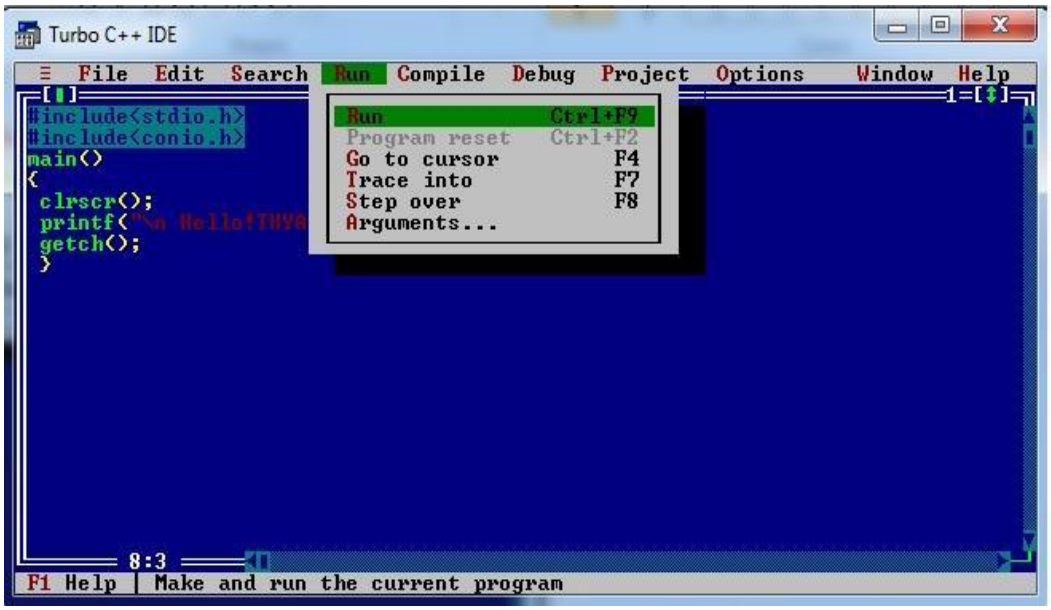
Step 2: File-> New (as shown in above picture) and then write your C program as given below

Step 3: Save the program using F2 (OR File-> Save as), remember the extension should be ".c". In the below screenshot I have given the name as Hello.c.

Step 4: Compile the program using Alt + F9 **OR** Compile-> Compile (as shown in the below screen shot).



Step 5: Press Ctrl + F9 to Run (or select Run-> Run in menu bar) the C program.



Step 6: Alt + F5 to view the output of the program at the output screen.