SECTION 1 C PROGRAMMING LAB

Structure **Page No.**

1.0 Introduction 5
1.1 Objectives 5
1.2 General Guidelines 5
1.3 Salient Features of C 6
1.4 C Programming Using Borland Compiler 7
1.5 Using C with UNIX 9
1.6 Running C Programs using MS Visual C++ 10
1.7 Program Development Life Cycle 15
1.8 List of Lab Assignments – Session wise 20

1.0 INTRODUCTION

This is the lab course, wherein you will have the hands on experience. You have studied the support course material (MCS-011 Problem solving and programming). In this part, C programming under DOS, UNIX and WINDOWS environments are provided illustratively. A list of programming problems is also provided at the end of each session. Please go through the general guidelines and the program documentation guidelines carefully.

1.1 OBJECTIVES

After completing this lab course you will be able to:

- develop the logic for a given problem ;
- write the algorithm;
- draw a flow chart;
- recognize and understand the syntax and construction of C code;
- gain experience of procedural language programming;
- know the steps involved in compiling, linking and debugging C code;
- understand using header files;
- make use of different data-structures like arrays, pointers, structures and files;
- understand how to access and use library functions;
- understand function declaration and definition;
- feel more confident about writing your own functions;
- be able to write some simple output on the screen as well as in the files;
- be able to write some complex programs;
- be able to apply all the concepts that have been covered in the theory course; and
- know the alternative ways of providing solution to a given problem.

1.2 GENERAL GUIDELINES

- You should attempt all problems/assignments given in the list session wise.

- You may seek assistance in doing the lab exercises from the concerned lab instructor. Since the assignments have credits, the lab instructor is obviously not expected to tell you how to solve these, but you may ask questions concerning the C language or a technical problem.

- For each program you should add comments (i.e. text between /* ... */ delimiters) above each function in the code, including the main function. This should also include a description of the function written, the purpose of the function, meaning of the argument used in the function and the meaning of the return value (if any).

These descriptions should be placed in the comment block immediately above the relevant function source code.

- The comment block above the main function should describe the purpose of the program. Proper comments are to be provide where and when necessary in the programming.
- The program written for the problem given should conform to the ANSI standard for the C language.
- The program should be interactive, general and properly documented with real Input/ Output data.
- If two or more submissions from different students appear to be of the same origin (i.e. are variants of essentially the same program), none of them will be counted. You are strongly advised not to copy somebody else's work.
- It is your responsibility to create a separate directory to store all the programs, so that nobody else can read or copy.
- Observation book and Lab record are compulsory.
- The list of the programs(list of programs given at the end, session-wise) is available to you in this lab manual. For each session, you must come prepare with the algorithms and the programs written in the Observation Book. You should utilize the lab hours for executing the programs, testing for various desired outputs and enhancements of the programs.
- As soon as you have finished a lab exercise, contact one of the lab instructor / incharge in order to get the exercise evaluated and also get the signature from him/her on the Observation book.
- Completed lab assignments should be submitted in the form of a Lab Record in which you have to write the algorithm, program code along with comments and output for various inputs given.
- The total no. of lab sessions (3 hours each) are 10 and the list of assignments is provided session-wise. It is important to observe the deadline given for each assignment.

1.3 SALIENT FEATURES OF C

We briefly list some of C's characteristics that define the language and also have lead to its popularity as a programming language. Naturally, we studied many of these aspects throughout the MCS-011 Problem Solving and Programming course.

- Small size
- Extensive use of function calls
- Structured language
- Low level (BitWise) programming readily available
- Pointer implementation - extensive use of pointers for memory, array, structures and functions.
- It has high-level constructs.
- It can handle low-level activities.
- It produces efficient programs.
- It can be compiled on a variety of computers.

1.4 C PROGRAMMING USING BORLAND COMPILER

C using Borland C/C++ Compiler

Some of you may be using the Borland C/C++ compiler during the lab sessions under MS-DOS connecting through Windows. Whilst C++ is a different programming language to C, it is in fact a superset of C i.e. almost everything that C provides, C++ provides too, and more besides. Therefore we can use Borland C++ to compile our C programs.

To start Borland C/C++

Click the **Start** button in the bottom left hand corner of the screen. The **Start** menu pops up. Select **Programs** from the **Start** menu. Select **Borland C/C++ from** the **Programs** menu. Select **Borland C/C++** from the **Borland C++ menu**. In summary the steps to launching **Borland C/C++** are:

Start--->Programs--->Borland C++--->Borland C++

You should now see the main window for the C/C++ development environment.

Editing a Program

We can create a program by entering text that corresponds to C statements into a file.

Setting Directories

Before you proceed, make sure that the directory settings for Borland C/C++ are correct. This can be done as follows:

Select **Options** from the menu and then select **Project** from the Options pull-down menu. This will display the Project Options dialog box. In the **Topics** area, click on **Directories**. On the right-hand side of the window you will see the **Directories** listed. Ensure that the information in each of the fields is as given below – if it is incorrect, modify it accordingly.

Source Directories
Include c:\ bc5\include
Library c:\ bc5\lib
Source *Leave this field BLANK*
Click on **OK** to continue.

Creating hello.c

- Select **File** from the menu and then select **New** from the file menu. The first thing that you should do is give the program a name, **hello.c**:
- Select **File** from the menu
- Select **Save as** from the File drop-down menu
- In the **Drives** drop-down list box, click on the down arrow to open up the list box.
- Scroll through the list to select the drive and click on it.
- Click on the **File Name:** field and type **hello.c**
- (Make sure the file name has the **.C** extension only. It should not have a **.CPP** extension. If it does change it to **.C**, or it won't run properly)
- Click on the **OK** button to continue.
- Now type the **hello.c** program exactly as you wrote in the lab observation book.
- Remember that it is good practice to save your programs periodically. You can do this as follows:

- Select **File** from the menu.
- Select **Save** from the File drop-down menu.

Compiling a Program

When you have finished typing in the program, you should compile it as follows:

Select **Project** from the menu
Select **Compile** from the project drop-down menu.

An attempt will be made to compile your program. If there are errors, they will be reported in the message window. You should use the information provided to help you fix the problems and then recompile the program.

Running a Program

If you have successfully compiled your program you can now link and run it as follows:

- Select **Debug** from the menu
- Select **Run** from the Debug drop-down menu.

First, an attempt will be made to link your program. If there are errors, they will be reported in the message window. You should use the information provided to help you fix the problems and then recompile and link the program.

Your program now runs. The output from the program will be displayed in a separate window. (If the screen displays a black output window for a split second and the window then disappears it means you did not set the Target Output type before you compiled your program).

To switch between the edit window and the output window, simply click on the window that you want to activate.

To close the output window, point to the icon in the top left-hand corner and double-click on it.

ALT+F9 (Short cut for Compile and Link)

As you become more familiar with the Borland C/C++ development environment, you will realize that it is possible to combine steps such as **compile** and **link** into a single step such as **build all.** There are also key combinations that can be used instead of selecting from menus (e.g. Alt+F9 compiles the program). You should spend some time familiarizing yourself with the Borland C/C++ development environment.

To close the **hello.c** file, double click on the icon in the top left-hand corner of the **hello.c** edit window.

To Quit from Borland C/C++ compiler

To exit from Borland C/C++:

- Select **File** from the menu
- Select **Exit** from the File drop-down menu

Minimum hardware requirements for Borland C/C++ Compiler

This Borland C/C++ compiler has a command line interface. You must run it from the DOS prompt. It has no Graphical User Interface like the version we use in the Labs. What you are getting for free is *not* a good Graphical Interface we use in the labs. You

get just the compiler, which is full spec, but you must drive it from the command line. To run the free C/C++ Compiler, your computer must meet the following specifications:

- PC with a Pentium processor, 90 MHz or higher (P166 recommended)
- Microsoft Windows 95, 98, 2000, or NT 4.0 with Service Pack 3 (or later)
- Microsoft Internet Explorer 4.01 Service Pack 1 (included on CD ROM)
- VGA or higher-resolution monitor; Super VGA recommended
- Microsoft Mouse or compatible pointing device
- 32 MB RAM (64 MB recommended)
- Disk space required for installation: 50 MB.

1.5 USING C WITH UNIX

A little knowledge of UNIX operating system and commands is necessary before you can write and compile programs on the UNIX system. Every programmer goes through the same three-step cycle.

1. Writing the program into a file
2. Compiling the program
3. Running the program.

During program development, the programmer may repeat this cycle many times, refining, testing and debugging a program until a satisfactory result is achieved. The UNIX commands for each step are discussed below.

Writing the Program

UNIX expects you to store your program in a file whose name ends in .c. This identifies it as a C program. The easiest way to enter your text is using a text editor like *vi*, *emacs* or *xedit*. To edit a file called testprog.c using vi type

vi testprog.c

The *editor* is also used to make subsequent changes to the program.

Compiling the Program

There are a number of ways to achieve this, though all of them eventually rely on the compiler (called cc on our system).

The C Compiler (cc)

The simplest method is to type

cc testprog.c

This will try to compile testprog.c, and, if successful, will produce a runnable file called a.out. If you want to give the runnable file a better name you can type

cc testprog.c -o testprog

This will compile testprog.c, creating runnable file testprog.

Running the Program

To run a program under UNIX you simply type in the filename. So to run program testprog, you would type

testprog

or if this fails to work, you could type

./testprog

You will see your prompt again after the program is done.

1.6 RUNNING C PROGRAM USING MS-VISUAL C++

Visual C++ is one of the most powerful and popular general-purpose programming languages. It is an extension of C / C++ programming language. Microsoft's Visual C++ is an Integrated Development Environment, or IDE. Microsoft Visual C++ has always been one of the most comprehensive and sophisticated software development environments available.

It has consistently provided a high level of programming power and convenience, while offering a diverse set of tools designed to suit almost every programming style.

Starting Microsoft Visual C++ 6.0

1. On desktop, click on the **Start** button.

2. Select **Microsoft Visual Studio 6.0** option.

3. Select Microsoft Visual **C++6.0** option.

You will see the main screen (figure 1.1).

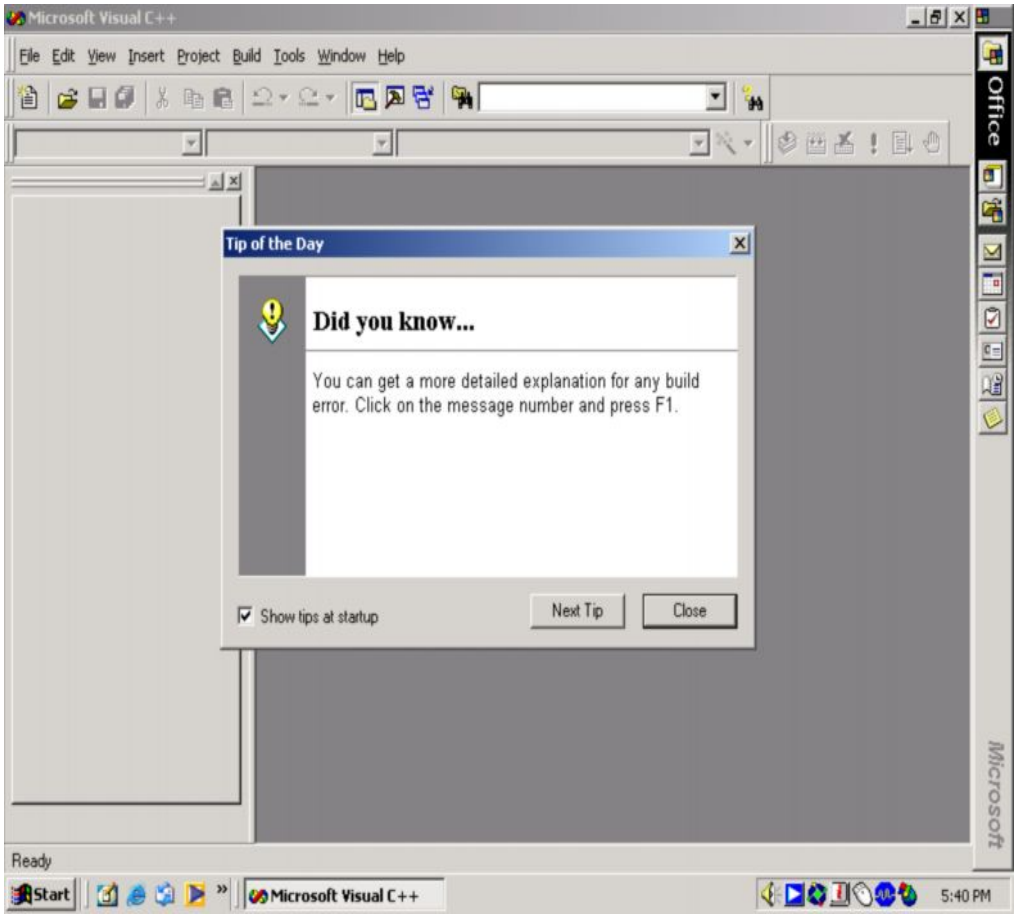


Figure 1.1

4. Click **Close**.

5. After that you can see the blank interface like figure 1.2.

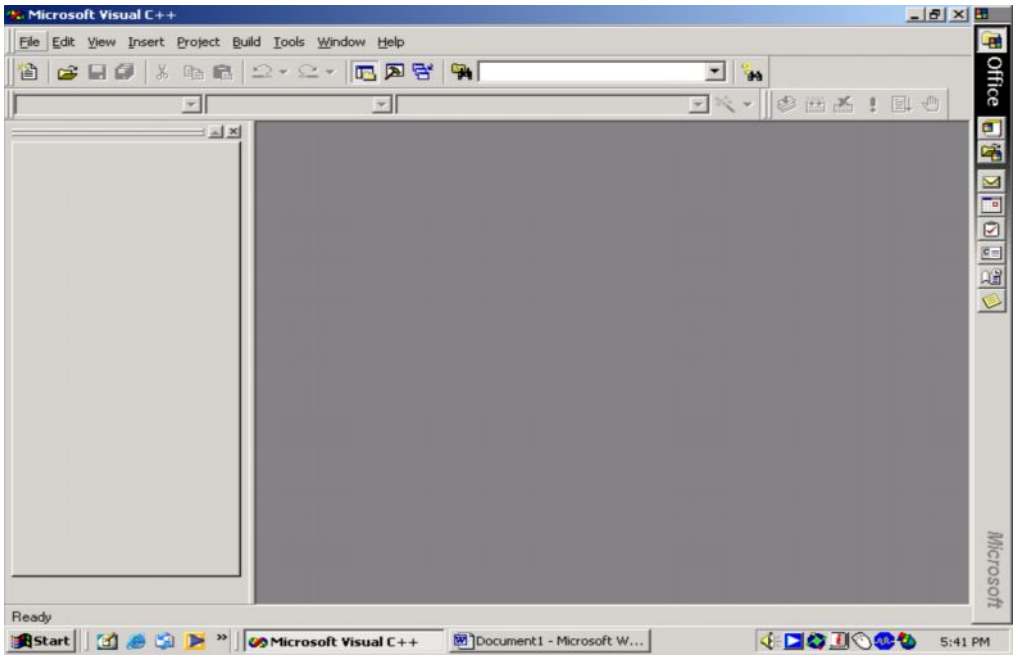


Figure 1.2

6. In the main screen select **File** menu, then click **New**. The **New** dialog box appears.

You can see the screen like figure 1.3

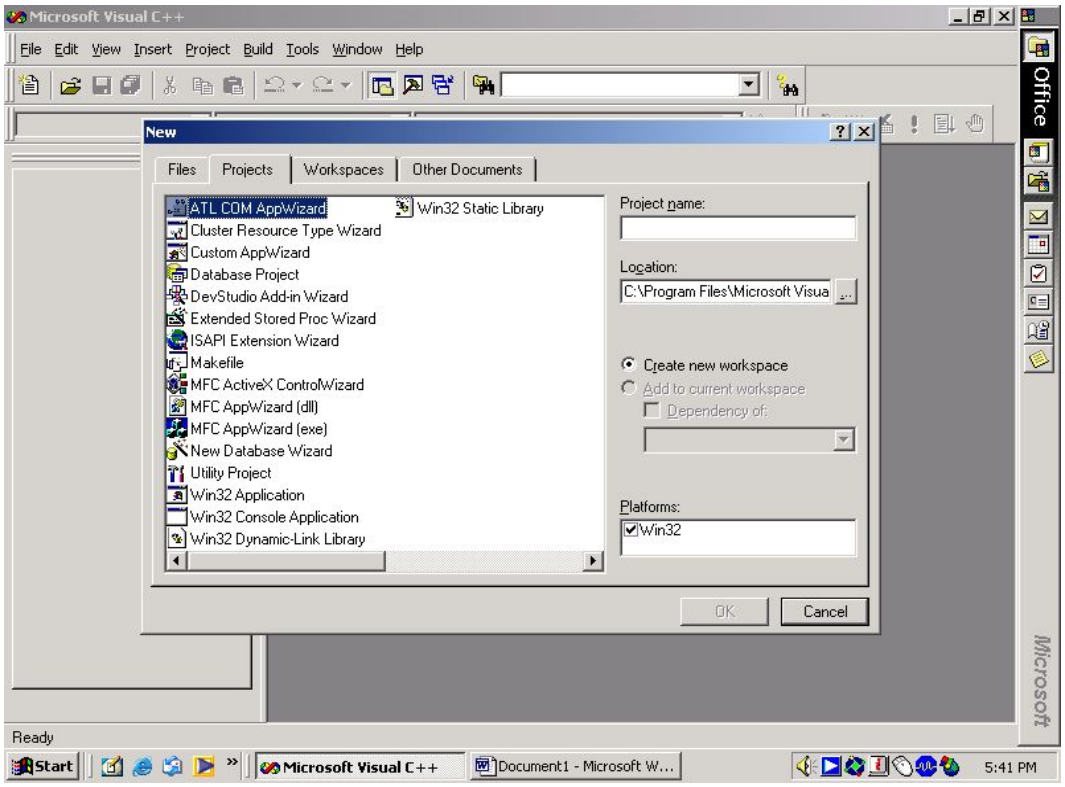


Figure 1.3

7. Select **File** at the *new dialog box*. Then, select **C++ Source File** and click **OK.**
   The screen in figure 1.4 will appear.

A blank text editor window (code window will appears).

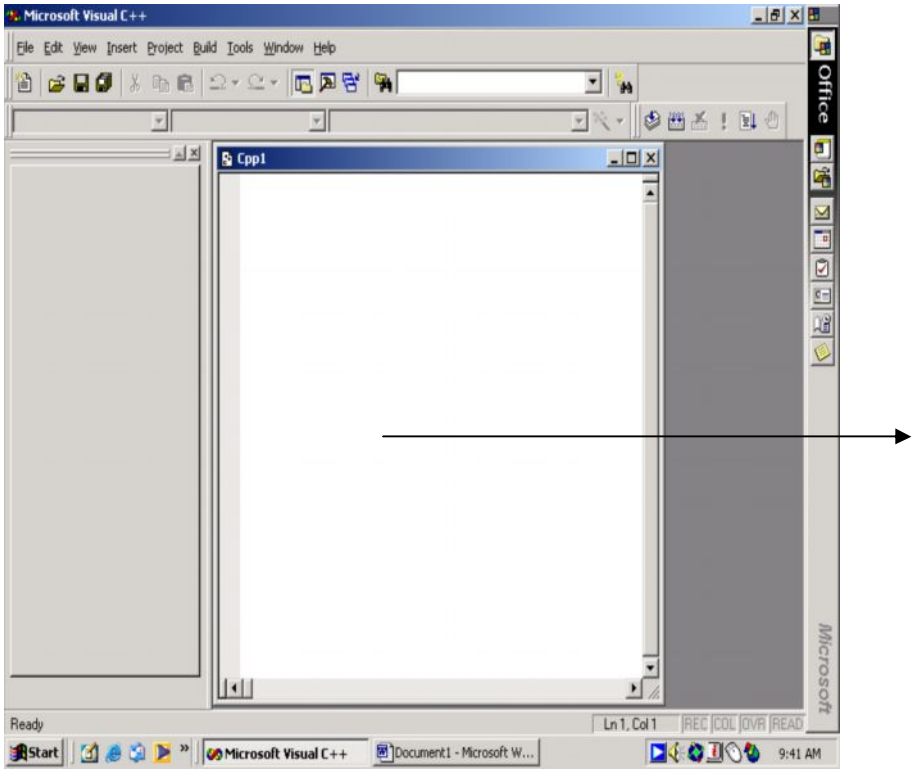


Figure 1.4

Creating a program

1. Type your C source code in the text window as follows.

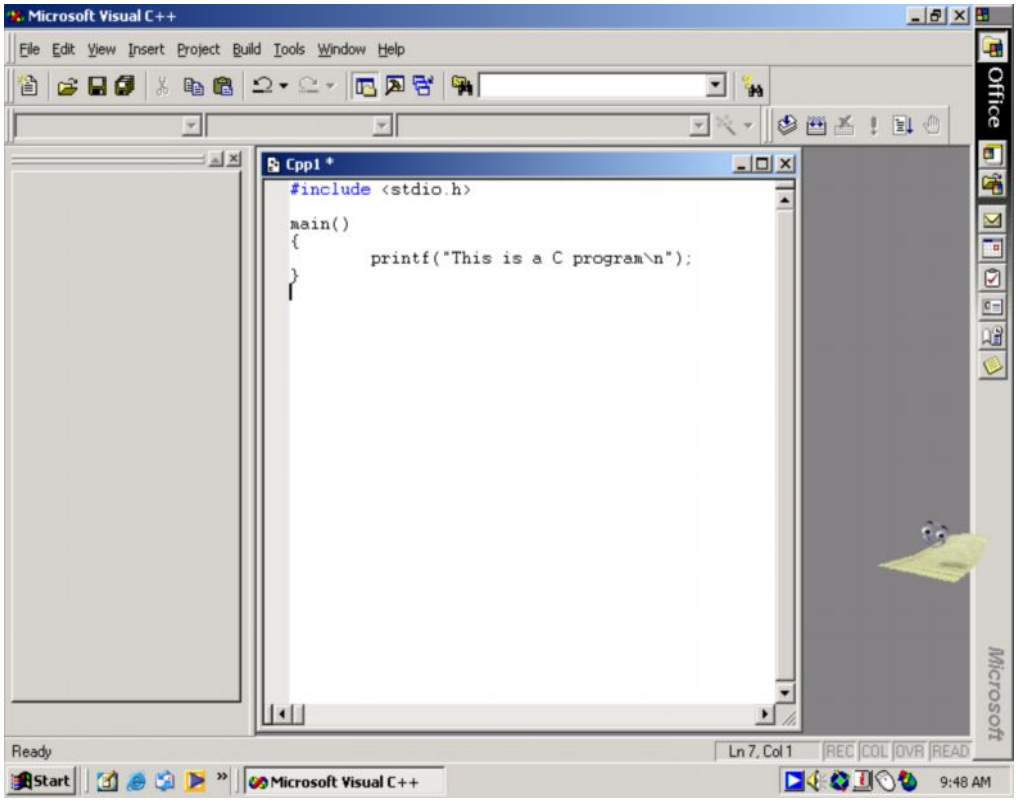


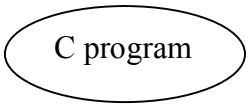
Figure 1.5

Save a program

1. From the menu bar, select **File** and then select **Save As**.
2. Select the appropriate directory .In lab session, we will save all our exercise in Directory desktop. So select save in **Desktop**.

3. Type the name of the program file

Example: program1**.c**



You must save all source code in C extension (means that, all in **.c)**. Click **Save** button.

Compiling a program

1. From the menu bar select **Built** and then select **Compile program1.c or** just click **icon compile.** See below figure1.6

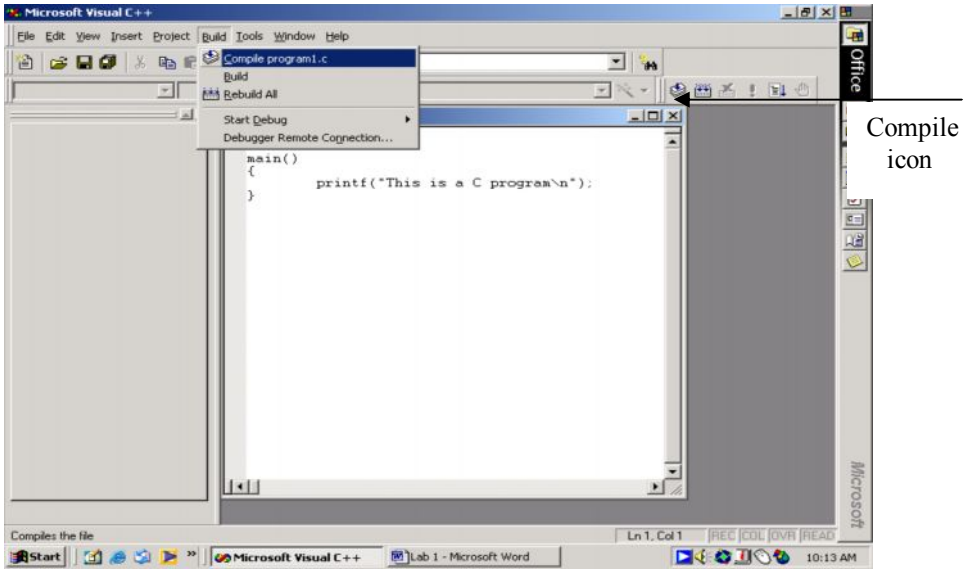


Figure 1.6

You will get message that request you to need a workspace, so just answer **Yes** to the question. Visual C++ will create default workspace and then build your code. This will produce a **.obj** program file. It does not have proper link with the library (built-in library) yet.

If there are any program errors or warning messages, visual C++ will display them in the message window (shown in figure 1.7 below). If there are no errors or warnings, you can execute your program.

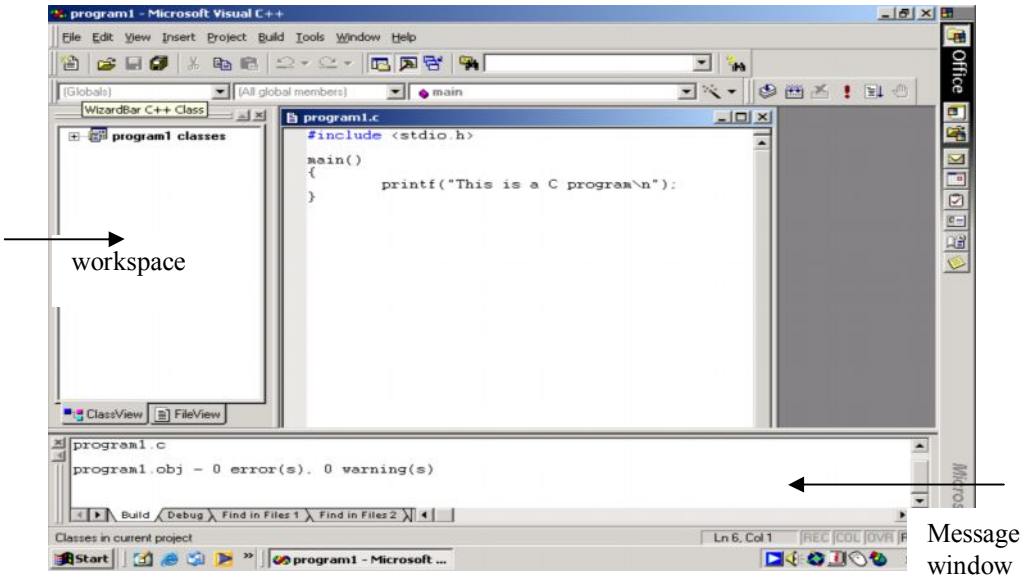


Figure 1.7

Executing (build) a program

1. From the menu bar, select **Build** and then select **Build program1.exe**. Or you can Click **build icon**. See below figure 1.8

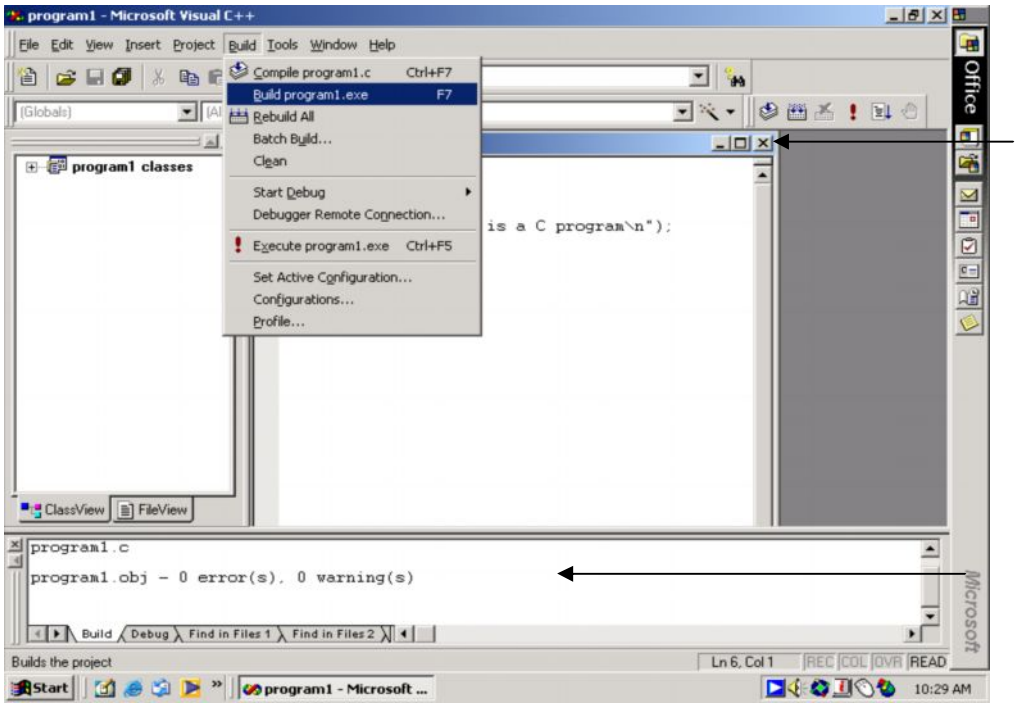


Figure 1.8

2. Now, the program code changes to .exe files extension. This extension you can see at message window as shown in the figure 1.9.

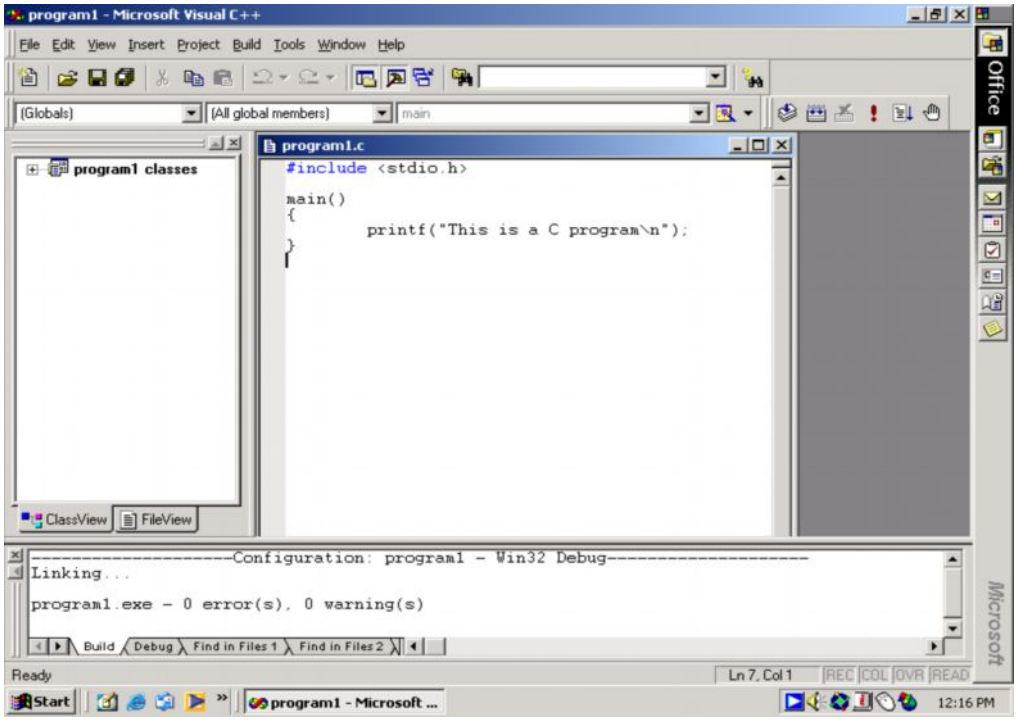


Figure 1.9

Running a program

1. From the menu bar, select **Build** and then select **Execute program.exe**. or you can click **execute program icon** .See below figure 1.10

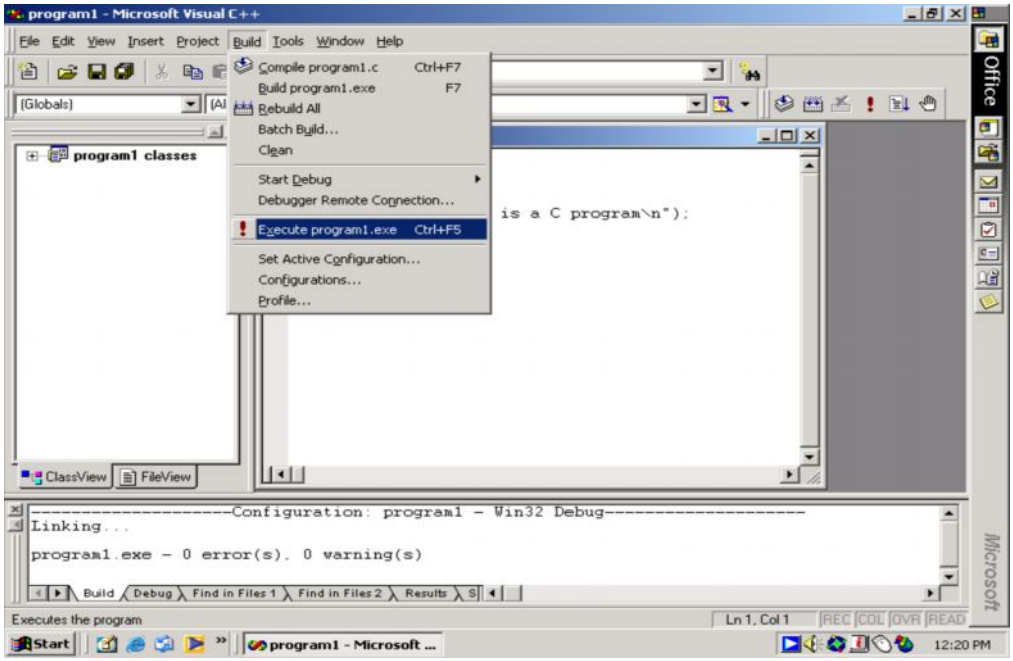


Figure 1.10

2. Now, output will appear as shown in the figure 1.11. The output screen contains the printed results. Press any key to return to the program.

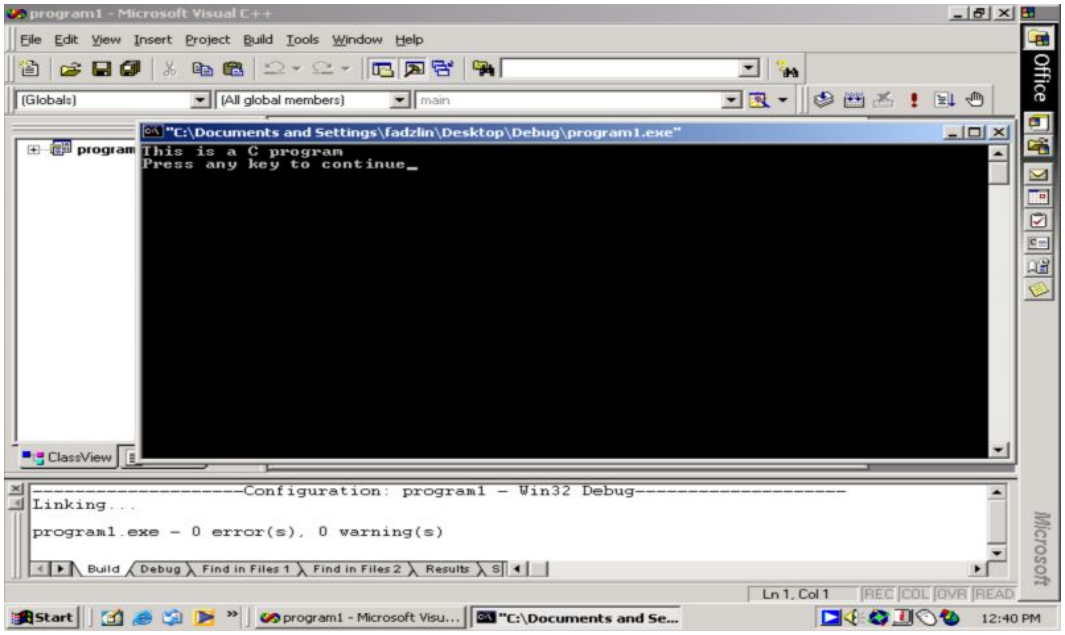


Figure 1.11

Let us see the steps involved in the program development life cycle.

1.7 PROGRAM DEVELOPMENT LIFE CYCLE

The four steps in the program development life cycle:

- Design algorithm
- Draw flowchart
- Program coding
- Testing for various inputs

Example 1

Represent the complete steps in the program development life cycle that reads the number of letter grades A, B, C, D and F for a student. The program will compute and print the student's grade point average. It should then determine and print the student's academic standing (like high honors, honors, satisfactory, or probation) according to the following table:

Grade Point Average	Academic Standing
3.51 - 4.00	High Honors
3.00 - 3.50	Honors
2.00 - 2.99	Satisfactory
Less than 2.00	Probation

Note: In computing grade point average, assume that the weight of letter grade A is 4, B is 3, C is 2, D is 1 and f is 0.

Step 1: Design the algorithm / pseudocode for the given problem

Print "Please enter your number of subject that your taken last semester"

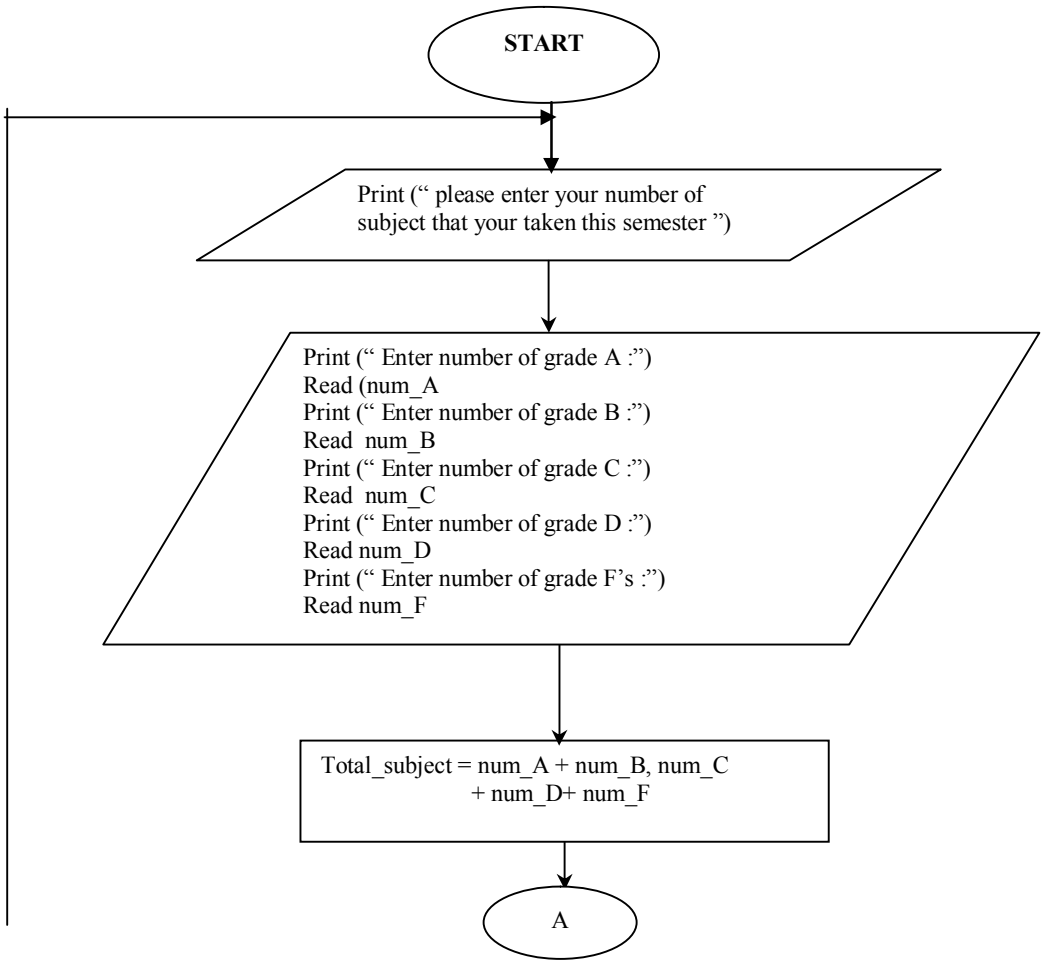
Set CorrectStatusInput "no"
Print "Enter the number of grade A"
Read number_of_ A
Print "Enter the number of grade B"
Read number_ of_B
Print "Enter the number of grade C"
Read number_of_C
Print "Enter the number of grade D"
Read number_of_D
Print "Enter the number of grade F"
Read number_of_F
Compute Total_subject = number of grade A + number of grade B + number of grade C + number of grade D + number of grade F
while CorrectStatusInput "no" and Total_subject < =0
begin
print "Enter number of grade A"
read number of A
print "Enter number of grade B"
read number of B
print "Enter number of grade C"
read number of C
print "Enter number of grade D"
read number of D
print "Enter number of grade F"
read number of F

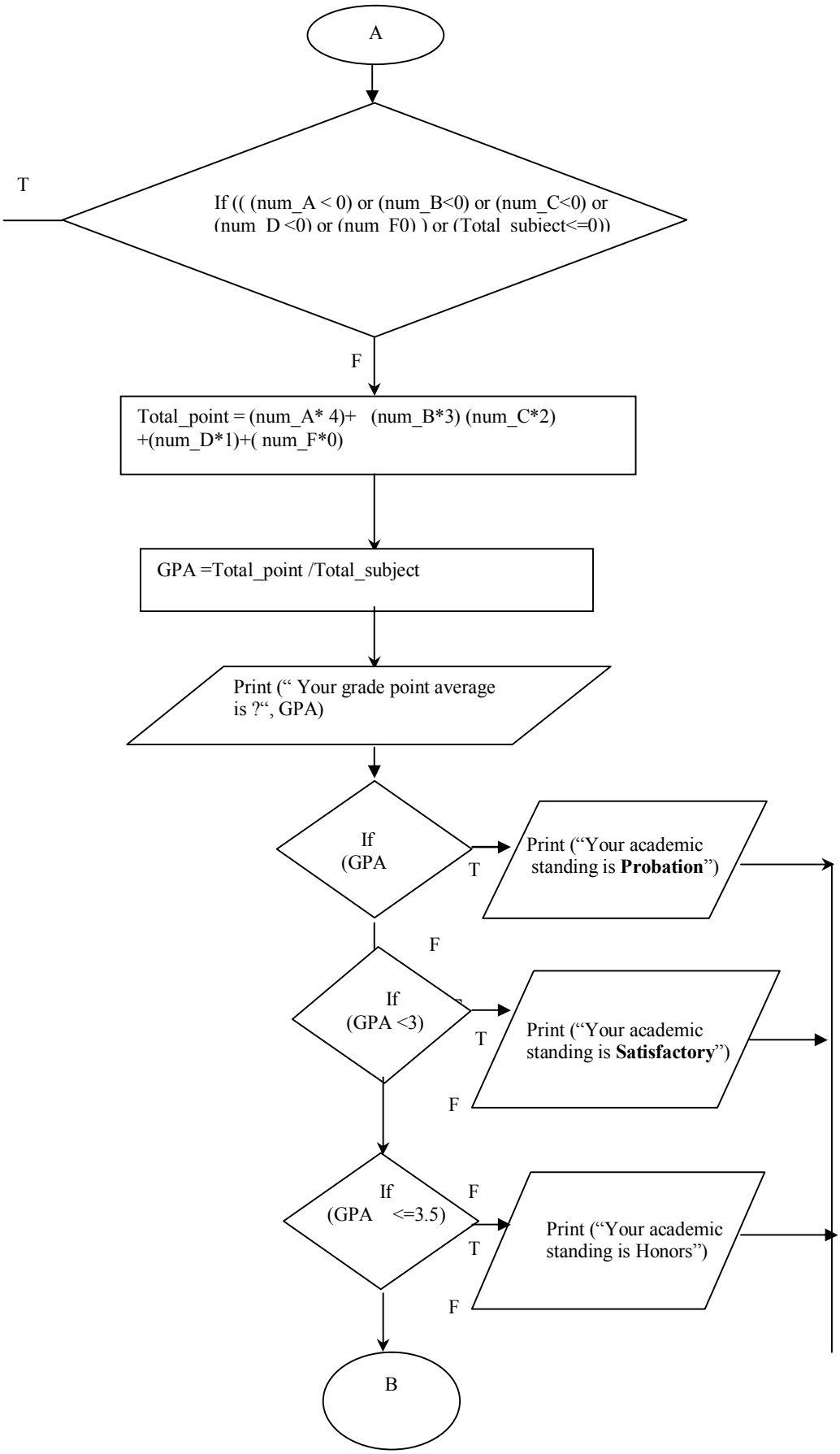
if (**number_of_A** greater than and equal 0 and **number_of_B** greater than and equal 0 and **number_of_C** greater than and equal 0 and **number_of_ D** greater than and equal 0 and **number_of_F** greater than 0) and (Total_subject >0)

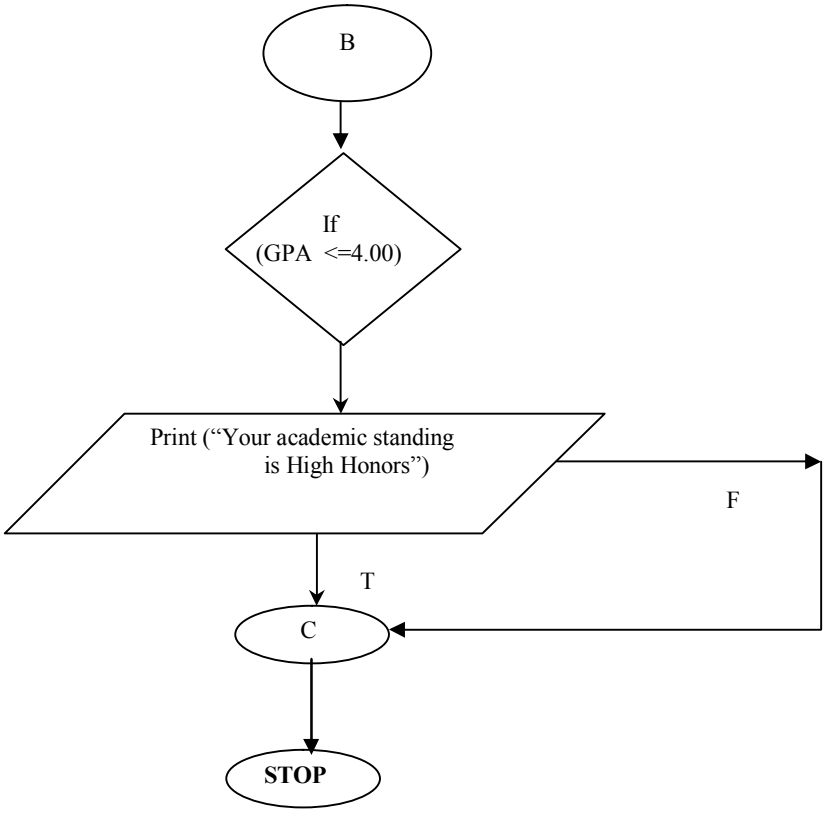
set CorrectStatusInput "yes"

```
            end if
        end
    end while

    compute Total_point = number of grade A * 4 + number of grade B *3 +
                          number of grade C*2  + number of grade D *1 +
                          number of grade F *0

    compute Total_average =  Total point / Total subject

    print  Total_average
    if  Total_ average less than 2
            print " Your academic standing is Probation"
        else
            if Total_average less than 3
                print " Your academic standing is Satisfactory"
              else
                if  Total_average less than or equal 3.5
                        print " Your academic standing is Honors"
                    else
                      if Total_average less than or equal 4.00
                              print " Your academic standing is High Honors"
                         end_if
                end_if
          end_if
  end_if
```

Step 2: Design a flowchart based on the algorithm.




A
T
If (((num_A < 0) or (num_B<0) or (num_C<0) or
(num_D <0) or (num_F0)) or (Total_subject<=0))
F
Total_point = (num_A* 4)+ (num_B*3) (num_C*2)
+(num_D*1)+(num_F*0)
GPA =Total_point /Total_subject
Print (" Your grade point average
is ?", GPA)
If
(GPA
T
Print ("Your academic
standing is **Probation**")
F
If
(GPA <3)
T
Print ("Your academic
standing is **Satisfactory**")
F
If
(GPA <=3.5)
F
T
Print ("Your academic
standing is Honors")
F
B



Step 3: Translate the flowchart to C source code.

```
/* Program to computer and print the grade point average */

#include <stdio.h>

main()

{

        int num_A,num_B,num_C,num_D,num_F;     /* Variables declaration part*/
        int total_subject;
        int total_point ;
        float GPA;


        do {
                printf(" \tThis is for calculate your GPA \n");
                printf("Enter number of grade A : ");
                scanf ("%d", &num_A);

                printf("Enter number of grade B : ");
                scanf ("%d", &num_B);

                printf("Enter number of grade C : ");
                scanf ("%d", &num_C);

                printf("Enter number of grade D : ");
                scanf ("%d", &num_D);

                printf("Enter number of grade F : ");
                scanf ("%d", &num_F);

                /* calculation of the total */
```

```
                total_subject = num_A + num_B + num_C + num_D + num_F;

        {
                if (((num_A <0) || (num_B <0)||(num_C<0)||(num_D <0)||(num_F
                   <0))||(total_subject <=0))
         printf (" you should enter your number that you having take again\n");
                        }
        } while (((num_A <0) || (num_B <0)||(num_C <0)||(num_D <0)||(num_F
                   <0))||(total_subject <=0));

        /* this to calculate and print total_subject, total_point and GPA */

        total_point = num_A *4 + num_B*3 + num_C*2 + num_D*1 + num_F*0;


        GPA =(float) (total_point / total_subject) ;

        /* this to print total subject */

        printf ("\nYour grade point average is %.2f \n",GPA);


                /* this selection to get output the students status */

         if (GPA <2)
         printf ("Your academic standing is Probation\n\n");
            else
                if (GPA <3)
                   printf ("Your academic standing is Satisfactory\n\n");
                     else
                          if (GPA <= 3.5)
                            printf ("Your academic standing is Honors\n\n");
                              else
                                   if (GPA <= 4.00)
                                   printf ("Your academic standing is High
honors\n\n");

        return 0;
}
```

Step 4: Testing

OUTPUT

```
This is for calculate your GPA
Enter number of grade A: 3
Enter number of grade B: 2
Enter number of grade C: 2
Enter number of grade D: 0
Enter number of grade F: 0
Your grade point average is 3.14.
Your academic standing is Honors.
```

1.8 LIST OF LAB ASSIGNMENTS – SESSIONWISE

Session 1:

1. Develop algorithm, flowchart and write an interactive program to calculate simple interest and compound interest.

2. Design a flow chart and write an interactive program for the problem given below:

   Assume that the United States of America uses the following income tax code formula for their annual income:
   First US$ 5000 of income : 0% tax
   Next US$ 10,000 of income : 10% tax
   Next US$ 20,000 of income : 15% tax
   An amount above US$ 35,000 : 20% tax.

   For example, somebody earning US$ 38,000 annually would owe

   US$ 5000 X 0.00 + 10,000 X 0.10 + 20,000 X 0.15 + 3,000 X 0.20,
   which comes to US$ 4600. Write a program that uses a loop to input the income and calculate and report the owed tax amount. Make sure that your calculation is mathematically accurate and that truncation errors are eliminated.

3. Design a flowchart and write an interactive program that reads in integers until a 0 is entered. If it encounters 0 as input, then it should display:

   - the total number of even and odd integers
   - average value of even integers
   - Average value of odd integers.

   Note: Use *switch* statement for selection.

4. Write an interactive program to generate the divisors of a given integer.

Session 2:

5. Write a program to find all Armstrong number in the range of 0 and 999
   *Hint: An Armstrong number of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since 3**3 + 7**3 + 1**3 = 371.*

6. Write a program to check whether a given number is a perfect number or not.
   *Hint: A positive integer n is called a **perfect number** if it is equal to the sum of all of its positive divisors, excluding n itself. For example, 6 is a perfect number, because 1, 2 and 3 are its proper positive divisors and 1 + 2 + 3 = 6. The next perfect number is 28 = 1 + 2 + 4 + 7 + 14. The next perfect numbers are 496 and 8128.*

7. Write a program to check whether given two numbers are amicable numbers or not.
   ***Hint: Amicable numbers** are two numbers so related that the sum of the proper divisors of the one is equal to the other, unity being considered as a proper divisor but not the number itself. Such a pair is (220,284); for the proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 and 110, of which the sum is 284; and the proper divisors of 284 are 1, 2, 4, 71, and 142, of which the sum is 220.*

8. Write a program to find the roots of a quadratic equation.

Session 3:

9. Write a function **invert(x, p, n)** that returns **x** with the **n** bits that begin at position **p** inverted. You can assume that **x**, **p** and **n** are integer variables and that the function will return an integer. As an example, if **x** = **181** [decimal] which is **10110101** in binary, and **p** = **4** and **n** = **2**, then the function will return **101<u>01</u>101** or **173** [decimal]. The underlined bits are the changed bits. Note that

bit positions are counted from the right to the left and that the counts start with a 0. Therefore, position **4** is the **5th** bit from the **right**values.

10. Write a function that calculates the compounded interest amount for a given initial amount, interest rate and number of years. The interest is compounded annually. The return value will be the interest amount. Use the following function definition: **float comp_int_calc(float int_amt, float rate, int years);** Write a program that will accept the initial amount, interest rate and the number of years and call the function with these values to find out the interest amount and display the returned value.

11. Break up the program that you wrote to solve **Problem 10** into two separate source files. The main function should be in one file and the calculation function must be in another file. And modify the program so that the interest rate is a symbolic constant and is no longer input from the keyboard. And put all the C preprocessor directives into a separate header file that is included in the two program source files [**i.e. #include "header.h"**].

12. Define two separate macros, MIN and MAX, to find and return, respectively, the minimum and maximum of two values. Write a sample program that uses these macros.
Hint: Use the ternary operator.

Session 4:

13. Write a program that will take as input a set of integers and find and display the largest and the smallest values within the input data values.

14. Write an interactive program that will take as input a set of 20 integers and store them in an array and using a temporary array of equal length, reverse the order of the integers and display the values.

15. Write a interactive program to do the following computation by providing the option using the *switch* statement:

    - Add two matrices
    - Subtract two matrices
    - Multiply two matrices

Session 5:

16. Write a program to check if the given matrix is magic square or not.

17. Write a program print the upper and lower triangle of the matrix.

18. Write a program to compute transpose of a matrix.

19. Write a program to find the inverse of a matrix.

Session 6:

20. Using recursion,

    (i) Find the factorial of a number
    (ii) Find Greatest Common Divisor (GCD) of two numbers
    (iii) To generate Fibonacci sequence
    (iv) Reverse *'n'* characters.

Session 7:

21. Write a program to convert a given lowercase string to upper case string without using the inbuilt string function.

22. Write a program to count number of vowels, consonants and spaces in a given string.

23. Write a program to input a string and output the reversed string, i.e. if "USF" is input, the program has to output "FSU". You are **not** to use array notation to access the characters, instead please use pointer notation.

Session 8:

24. Write a program to process the students-evaluation records using structures.

25. Define a structure that will hold the data for a complex number. Using this structure, please write a program that will input two complex numbers and output the multiple of the two complex numbers. Use double variables to represent complex number components.

    *Note: A complex number **z** is a number of the form z = a + bi where a and b are real numbers. The term **a** is called the real part of **z** and **b** is called the imaginary part of **z**. The multiplication operation on complex numbers is defined as:*

    (a + bi) * (c + di) = (ac - bd) + (ad + bc)i

26. Modify the above program so that the multiplication is carried out in a function that accepts two complex number structures as input parameters and return a complex number structure with the result.

Session 9:

27. Write a function that will return the length of a character string. You are not allowed to use the strlen C library function.
    Note: Use "Pointers" concept

28. Write a function that returns the minimum and the maximum value in an array of integers. Inputs to the function are the array of integers, an integer variable containing the length of the array and pointers to integer variables that will contain the minimum and the maximum values. The function prototype is:

    void minmax(int array[], int length, int * min, int * max);

29. Write a sample program that uses this function to find and display the minimum and the maximum values of an array of integers. Use an array of 10 integers. You can either use *scanf* to input the values into that array or initialize the array with values in the program itself.

Session 10:

30. Write a program that prompts the user the name of a file and then counts and displays the number of bytes in the file. And create a duplicate file with the word '.backup' appended to the file name. Please check whether file was successfully opened, and display an error message, if not.

31. Write a program to create a file, open it, type-in some characters and count the number of characters in a file.

32. Write a program that will input a person's first name, last name, SSN number and age and write the information to a data file. One person's information should be in a single line. Use the function fprintf to write to the data file. Accept the information and write the data within a loop. Your program should exit the loop when the word 'EXIT' is entered for the first name. Remember to close the file before terminating the program.
Hint: Use the function *strcmp()* to compare two strings.

33. Modify the program no: 23 using file concept.