Chapter 2 Introduction to C Programming

1. **Features of C language:**
    a. **Robust Language**: C is a robust language whose rich set of built-in functions and operators can be used to write any complex program. The C compiler combines the capabilities of an assembly language with the features of a high-level language and therefore it is well suited for writing both system and business packages. In fact, many of the C compilers available in the market are written in C.
    b. **Efficient and fast**: The programs written in C are efficient and fast. This is due to its variety of data types and powerful operator. It is many time faster than BASIC. For example a program to increment a variable from 0 to 15,000 takes about one second in C while it takes more than 50 seconds in an interpreter BASIC.
    c. **Portable**: C is highly portable. This means that C programs written for one computer can be run on another with little or no modification. Portability is important if we plan to use a new computer with a different operating system.
    d. **Structured Language**: C language is well suited for structured programming. Thus requiring the user to think of a problem in term of function modules or blocks. A proper collection of these modules would make a complete program. This modular structure makes program debugging, testing and maintenance easier.
    e. **Extended itself**: Another important feature of C is its ability to extend itself. A C program is basically a collection of functions that are supported by the C library. We can continuously add our own functions to C library. With the availability of a large number of functions, the programming task becomes simple.
2. **Structure of C program**:
    ⇨ A group of building blocks called function. A function is a subroutine that may include one or more statements designed to perform a specific task.
    a. **Documentation Section:** The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.

```
/*Documentation Section:
  Program Name: program to find the area of circle
  Author: Rumman Ansari
  Date : 12/01/2013
*/

#include"stdio.h" //Link section
#include"conio.h" //Link section

#define PI 3.14 //Definition section

float area; //Global declaration section
void message(); //function prototype declaration section

void main()
{
  float r;                          //Declaration part
  printf("Enter the radius \n");    //Executable part
  scanf("%f",&r);
  area=PI*r*r;                      // Calculation Part
  printf("Area of the circle=%f \n",area);
  message();                        // Function Calling
}

// Sub function
void message()
{
  printf("This Sub Function \n");
  printf("we can take more Sub Function \n");
}
```


b. **Link section:** The link section provide instruction to the compiler to link functions from the system library.

c. **Definition section:** The definition section defines all symbolic constant.

d. **Global declaration section:** There are some variable that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also declares all the user defined functions.

e. **main() function section:** Every C program must have one main() function section. This section contains two parts:
   - **Declaration part:** The declaration part declares all the variables used in the executable part.
   - **Executable part:** There is at least one statement in the executable part

   Thus two parts must appear between the opening and the closing braces. The program execution begins at the opening brace and ends at the closing brace. The closing brace of the main function section is e logical end of the program. All statements in the declaration and executable part end with semi colon (;).

f. **Subprogram section:** The subprogram section contains all the user defined functions that are called in the **main** function. User defined functions are generally placed immediately after the **main** function, although they may appear in any order.

   All sections, except main function section may be absent when they are not required.

3. **Development of Program:**

A programmer, while writing program, should follow certain rules:

a. All All statements should be written in lower case letters. Upper case letters are only used for symbolic constants.

b. Blank spaces may be inserted between the words. This improves the readability of the statements. However, it is not used while declaring a variable, keyword, constant and function.

c. It is not necessary to fix the position of statement in the program, that is, the programmer can write the statement anywhere between the two braces following the declaration part. The user can also write one or more statements in one line by separating them with a semicolon (;). Hence, it is often called a free-form language. The following examples of statements are valid.

```
a=b+c;
d=b*c;
```

or

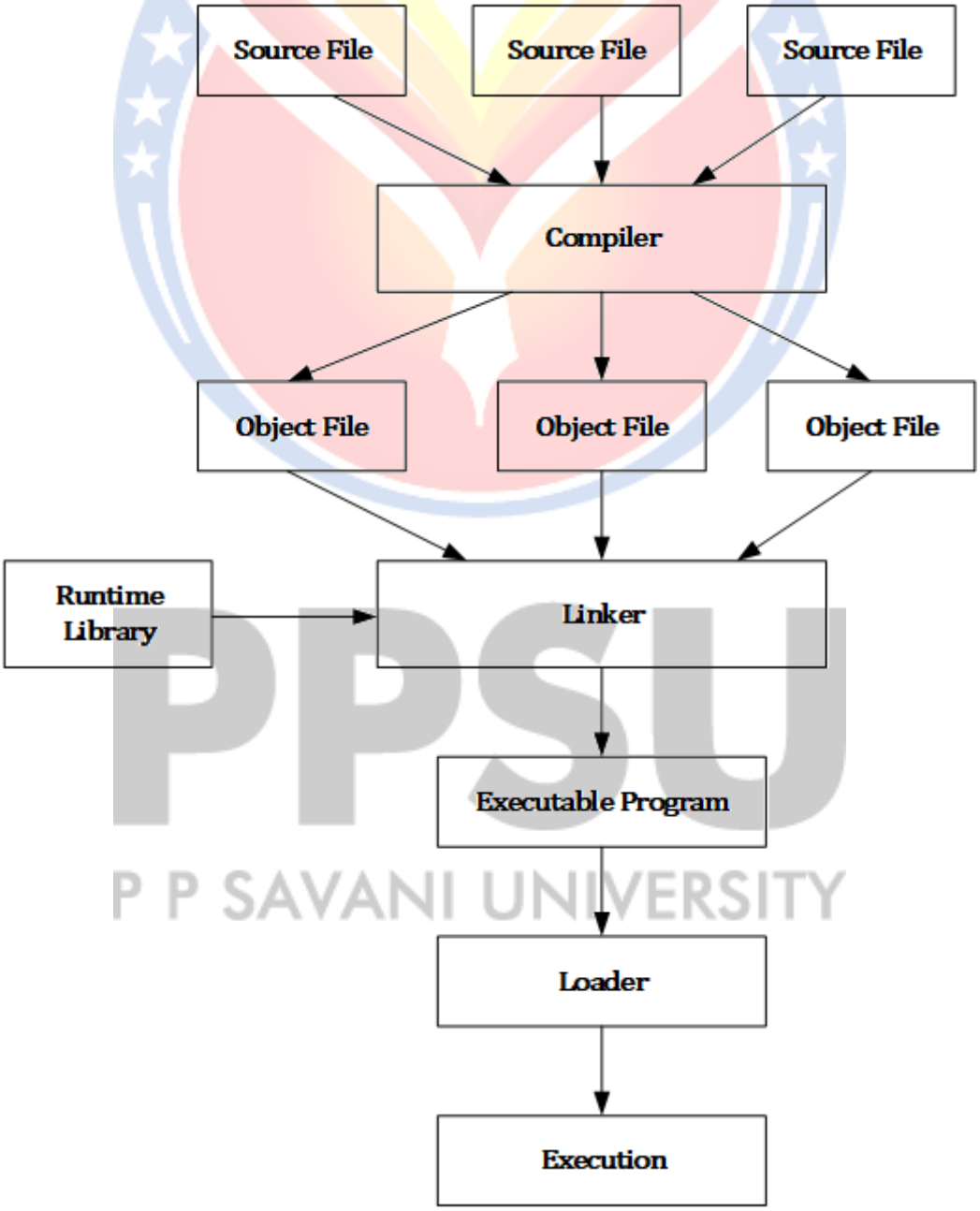
```
a=b+c; d=b*c;
```

d. The opening and closing braces should be balanced, for example, if opening braces are four, then closing braces should also be four.

4. **Flow chart and algorithm:** Refer from chapter 1 Lecture notes.
5. **Types of Errors:** There are 5 types of error in C :
    a. **Syntax Errors:** These are also referred to as compile-time errors. These errors have occurred when the rule of C writing techniques or syntaxes has been broken. These types of errors are typically flagged by the compiler prior to compilation.
    b. **Runtime Errors:** This type of error occurs while the program is running. Because this is not a compilation error, the compilation will be completed successfully. These errors occur due to segmentation fault when a number is divided by division operator or modulo division operator.
    c. **Logical Errors:** Even if the syntax and other factors are correct, we may not get the desired results due to logical issues. These are referred to as logical errors. We sometimes put a semicolon after a loop, which is syntactically correct but results in one blank loop. In that case, it will display the desired output.
    d. **Linker Errors:** When the program is successfully compiled and attempting to link the different object files with the main object file, errors will occur. When this error occurs, the executable is not generated. This could be due to incorrect function prototyping, an incorrect header file, or other factors. If main() is written as Main(), a linked error will be generated.
    e. **Semantic Errors:** When a sentence is syntactically correct but has no meaning, semantic errors occur. This is similar to grammatical errors. If an expression is entered on the left side of the assignment operator, a semantic error may occur.
6. **Debugging:**
    a. **Creation of program:** Programs should be written in C editor. The file name does not necessarily include extension .C. The default extension is .C. The users can also specify their own extensions.
    b. **Compilation and linking of a program:** The source program statements should be translated into object programs which is required for execution by the computer. The translation is done after correcting each statement. If there is no error, compilation proceeds and translated program are stored in another file with the same file name with extension ".obj". If there is any error, then the programmer should correct it. Linking is also an essential process. It keeps all

program files and functions required by the program together. For example, if the programmer is using pow() function, then the object code of this function should be brought from math.h library of the system and linked to the main() program.

c. **Executing the program:** After the compilation, the executable object code will be loaded in the computer's main memory and the program is executed. In case of logic or data errors in the program, the source program is altered and all the earlier initiated steps such as compilation, linking and execution to be repeated. All these steps can be performed using menu options of the editor.



Flow chart for Execution of C program