Unit - 2

chapter 3

Introduction to C Programming

Chapter Outline

- Introduction
- What is a Program? What is a Program Language?
- History and Evolution of C
- Features or Characteristics of C Language
- Applications of C Programming Language
- Sample 'C' Programs
- Basic Structure of C Program
- Creating and Executing a 'C' – Program
- Programming Style
- Review Questions

3.1 Introduction

Computers can be utilised efficiently only when the communications with them are very well defined. We know that communication between human beings normally comes through a language. Similarly we need a language to communicate with computers, which they understand. We can not think of English for communicating with computers but it is possible to use English-like language to work with computers. C is a English-like language because there is a similarity in learning English and in learning C.

When we want to develop a meaning for paragraph in English, First we need alphabets, alphabets are grouped together to form words; words are grouped together to form sentences; Finally sentences are grouped together to form the paragraph ! Similarly, before we develop a simple program in C, we need alphabets (this include digits and special symbols); when we group alphabets together it generates constants, variables and keywords etc; Assembling these words will lead to formation of instructions (sentences) ; finally group of instructions forms a simple C program. In this book, we will study how to form words, instructions and programs using C programming language.

3.2 What is a Program? What is a Program Language?

Computers are really dumb machines because they do only what they are asked to do. Many computers perform operations on a very basic level like addition/subtraction of numbers etc. These basic operations are decided by the instruction set of the computer. To solve a problem using a computer, it is necessary to express the method to solve the problem using the primitive instructions of the computer. A sequence of instructions given to the computer is known as program. The method or strategy used for solving the problem is known as algorithm. With the algorithm in hand, it is possible to write the instructions necessary to implement the algorithm. Programs are developed to solve some problem. These programs can be written using some computer language like Visual Basic, C, C++, Java. C#, Python etc.

Definition : Program and Programming Language

- **Programming Language** - Is a formal language, which comprises a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement specific algorithms. Most programming languages consist of instructions for computers.
- **Computer Program** - Is a collection of instructions that performs a specific task when executed by a computer. Most computer devices require programs to function properly. A computer program is usually written by a computer programmer and can be can be written in either high or low-level languages, depending on the task and the hardware being used.

The programming language mainly refers to high-level languages such as C, C++, Java, C#, COBOL, Visual Basic, Python etc. The high level languages use the common English language to help make the code more understandable and to speed up the process of writing and debugging programs.

Each programming language contains a unique set of keywords and syntax, which are used to create a set of instructions. Thousands of programming languages have been developed till now, but each language has its specific purpose.

Programming languages can be divided into two different levels:

1. **High Level Languages :**
   - When we talk about high level languages, these are programming languages. The important feature about such high level languages is that they allow the programmer to write programs for all types of computers and systems. Every instruction in high level language is converted to machine language for the computer to comprehend.

     The highlights of **high-level languages** are faster program development, readable, well structured. The debugging is very easy in high-level languages. But, the execution speed of high-level languages is much less compared to low-level languages.
   - **Example** : Python, Visual Basic, Java, C, C++, SQL and many more.
2. **Low Level Languages :**

   Hardware/Processor-specific assembly language and machine language. The Low level Languages are designed to give a better machine efficiency i.e., programs can be executed much faster compared to high level languages. But, developing a program is much difficult and debugging is very difficult.
   - **Machine Language :** This is one of the most basic low level languages. The language was first developed to interact with the first generation computers. It is written in binary code or machine code, which means it basically comprises of only two digits – 1 and 0.

     The machine languages need no translators. It is because they are already present in machine-understandable form. The process of execution is very fast in the case of machine languages. It is because they already contain their data in a binary format. The machine language is the lowest level of all the programming languages. It executes all the instructions directly through the CPU (Central Processing System) of the system.
   - **Assembly Language:** This is the second generation programming language. It is a development on the machine language, where instead of using only numbers, we use English words, names, and symbols.

     Assembly language falls between a high-level programming language and Machine language. It has syntaxes similar to English, but more difficult than high-level programming languages. To program in assembly language, one should have understood at hardware level like computer architecture, registers, etc. This kind of programming is mostly seen in the embedded systems. **Example**: ADD R1, R2

     Assembly languages need translators (also known as assemblers) for converting the mnemonics (english like words and symbols) into a machine-understandable form. These languages have a slower execution than that of any machine language. The assembly language is a low-level language for programming that requires an assembler to convert the instructions into a final object or machine code.

3.3 History and Evolution of C

- The base or father of programming languages is 'ALGOL.' It was first introduced in 1960 'ALGOL' was used on a large basis in European countries. 'ALGOL' introduced the concept o structured programming to the developer community.

- CPL was developed jointly between the Mathematical Laboratory at the University of Cambridg and the University of London Computer Unit in 1960s. **CPL (Combined Programmin Language)** was developed with the purpose of creating a language that was capable of bot machine independent programming and would allow the programmer to control the behavio of individual bits of information. But the CPL was too large for use in many applications.
- In 1967, a new computer programming language was announced called as 'BCPL' which stand for **Basic Combined Programming Language (BCPL)**. BCPL was designed and develope by Martin Richards, especially for writing system software. This was the era of programmin languages. BCPL was scaled down version of CPL while still retaining its basic features.
- In 1970, a new programming language called **'B'** was introduced by Ken Thompson tha contained multiple features of 'BCPL.' This programming language was created using UNI operating system at AT&T and Bell Laboratories. Both the 'BCPL' and 'B' were syste programming languages. B Language was a scaled down version of BCPL. B Language wa written for the systems programming.
- In 1972, a great computer scientist **Dennis Ritchie** created a new programming languag called 'C' at the Bell Laboratories. It was created from 'ALGOL', 'BCPL' and 'B' programmin languages. 'C' programming language contains all the features of these languages and man more additional concepts that make it unique from other languages.
- C was developed by **Dennis Ritchie** at Bell Laboratories in 1972. Most of its principles an ideas were taken from the earlier language ALGOL, B, BCPL and CPL.
- In 1973, C language had become powerful enough that most of the Unix kernel was rewritte in C. This was one of the first operating system kernels implemented in a language other tha assembly.
- During the rest of the 1970's, C spread throughout many colleges and universities because o its close ties to UNIX and the availability of C compilers. Soon, many different organization began using their own versions of C Language. This was causing great compatibility problems. In 1983, the American National Standards Institute (ANSI) formed a committee to establish a standard definition of C Language. That is known as ANSI Standard C. Today C is the most widely used System Programming Language.
- The language C uses many features from B, BCPL and CPL languages and added the concept of data types and other powerful features. This programming language began to gain widespread popularity and support only at late 1970s. This was because, C compilers were not readily available for commercial use outside of Bell Laboratories.

3.4 Features or Characteristics of C Language

C is a general-purpose high level language that was originally developed for the UNIX operating system. The most of UNIX operating system are written in C language. C has become a widely used professional language for various reasons. The below figure indicates feature of C at a glance.

- **Simple and Efficient :**

  The C Language is a simple language that is easy to learn even for a beginner and is super efficient to use both in terms of the time of development and time of execution.

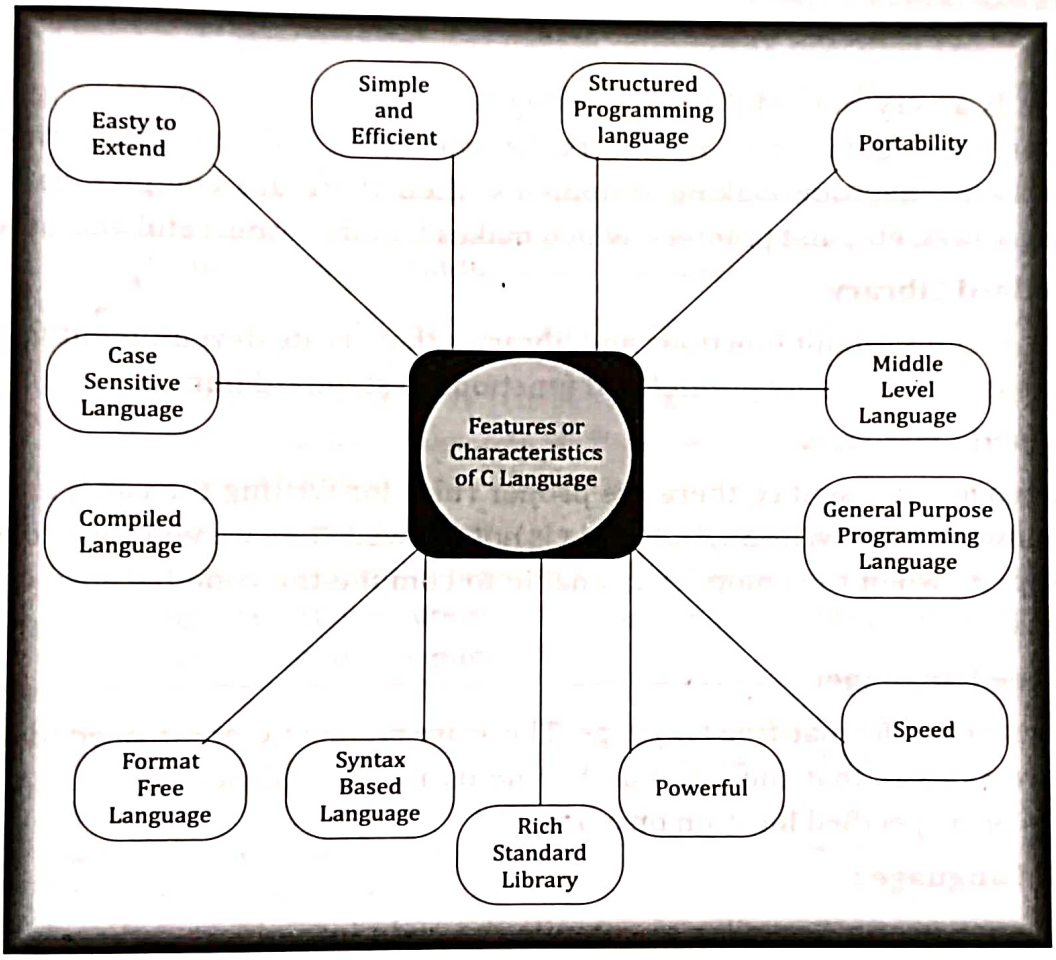
Modularity/Structured Programming Language :

This feature of C language allows the program to be broken into smaller units and run individually with the help of functions. For example, if we want to find the area of a square, a rectangle and, a triangle. Instead of writing the code as a whole, we can divide it into separate functions, one for finding the area of a square, a rectangle, and triangle respectively. It guarantees fewer chances of errors and makes it visually appealing and more organized.

C is a structured language, so it typically support several loop constructs, such as while, do-while, and for. As it is a structured language, the statements can be placed any where on a line and does not require a strict field concept.

Structured Languages: C, Pascal, C++, Java

Non-Structured Languages : Cobol, Basic, Fortran.



Portability :

It refers to the usability of the same fragment of code in different environments. C programs are capable of being written on one platform and being run on another with or without any modification. It means if we have written a simple C program on Windows OS laptop or computer, then same program can be executed on any other operating system or machine, like, Linux or macOS, etc. with minimal mofications or without any modifuications.

C' language offers highest degree of portability i.e., percentage of changes to be made to the sources code is at minimum when the software is to be loaded in another platform. Percentage of changes to the source code is minimum.

- **Middle Level Language:**

  The C language combines the features of both high-level languages and low-level languages resulting in faster program development and faster program execution. So, C is often called Middle Level Programming Language.

- **Genral Purpose Programming Language:**

  C is a general purpose programming language. We can develop games, business software, utilities, mathematical models, word processors, spreadsheets and other kinds of software.

- **Speed :**

  The compilation and execution time of C language is fast than other high level programming languages like Java or Python.

- **Powerful :**

  C language is a very powerful programming language. It has a broad range of features like support for many data types, operators, keywords, etc., allows structuring of code using functions, loops, decision-making statements, then there are complex data-structures like structures, arrays, etc., and pointers, which makes C quite resourceful and powerful.

- **Rich Standard Library:**

  C supports various inbuilt functions and libraries that create development fast. We can create programs easily by utilizing pre-defined functions in standard library.

- **Syntax Based Language:**

  The C language has a syntax, there are proper rules for writing the code, and the C language strictly follows it. If we write anything that is not allowed, then we will get a compile-time error, which happens when the compiler is unable to compile the code because of some incorrect code syntax.

- **Format Free Language:**

  The C language is a format-free language. There are no line numbers needed in the C language code, or we can say that the line number holds no significance. There is no need to place statements on a specified location on a line.

- **Compiled Language :**

  The C language uses a Compiler to compile the code into object code, which is nothing but machine code that the computer understands. Hence to run a C language program we have to install a C language compiler first.

- **Case Sensitive Language :**

  The C language is case sensitive. The uppercase and lowercase characters are different. For example,if is not the same as IF in C language.

- **Easy to Extend :**

  Programs written in C language can be extended means when a program is already written in it then some more features and operations can be added to it.

What are the main features of the C language?

The main features of the C language are, it is simple(easy to learn), efficient, Powerful, Portable, Middle-level language, and Structured Language. The C language also has a rich standard library with the support of extensive 3rd party libraries.

Why C Language is So Popular and Powerful?

C language is the most popular computer language today because it is a structured, high-level, and machine independent language. C was basically designed for the UNIX operating system, 93% of UNIX operating system is written in C language.

Initially 'C' programming was limited to the UNIX operating system, but as it started spreading around the world, it became commercial, and many compilers were released for cross-platform systems. Today 'C' runs under a variety of operating systems and hardware platforms.

Why C often called as Middle Level Language?

C is often termed as a middle level programming language because it combines the power of a high level language with the flexibility of a low level language. High-level languages have lot of built-in features and facilities, which result in high programming efficiency and productivity. Low-level languages, on the other hand, are designed to give more efficient programs and better machine efficiency.

C has the feature of high level programming language as well as the low-level programming. It works as a bridging gap between machine language and the more conventional high-level languages. This feature of C Language made it most popular for system programming as well as application programming

What do you mean by C programming language standard?

In 1970s the C language became widely popular, with many universities and organizations begins to create their own variations of the language. At start of 1980s compatibility problems between the various C implementations became apparent.

To assure that 'C' language will remain standard, **American National Standards Institute (ANSI)** defined a commercial standard for 'C' language in 1989. Later, it was approved by the International Standards Organization (ISO) in 1990. 'C' programming language is also called as 'ANSI C'. The version of C defined by the 1989 standard is commonly referred to as C89. The next version of the standard C99 was published in 1999 that introduced new futures like advanced data types and other changes.

The latest C standard was released in June 2018 which is ISO/IEC 9899:2018 also known as the C11.

The compiler creators follow the standards defined by ANSI to create the compilers. This ensures that all compilers follow the same standards.

3.5 Applications of C Programming Language

C language is used in wide variety of applications. Some application of C language is given below.

- To develop the System Software like Operating System, Compiler.
- To develop Application Software like Database and Spread Sheets.
- Development of Graphical related application like Computer Simulators and Mobile Games.
- To design Network Devices Drivers.
- Development of Embedded System Software.
- Development of Text editors and Language Interpreters.
- Creating Video Games, 3D Animations, Multimedia Applications, Image Processing, Modelling and Simulations etc.

QUICK FACTS

- The first operating system to be developed using a high-level programming language was UNIX, which was designed in the C programming language. Later on, Microsoft Windows and various Android applications were scripted in C..
- Adobe Photoshop, one of the most popularly used photo editors since olden times, was created with the help of C. Later on, Adobe Premiere and Illustrator were also created using C.
- Several popular compilers were designed using C such as Bloodshed Dev-C, Clang C, MINGW, and Apple C.
- MySQL is the most popular database software which is built using 'C'.

3.6 Sample 'C' Programs

Example

Let us begin by considering a simple C program. The following program prints a line of text. The program and program's output are shown below.

```
1  #include <stdio.h>
2  void main( )
3  {
4          printf ("Welcome to C language \n");
5  }
```

Output

```
Welcome to C language
```

Even though the above program is simple, it illustrates several important features of the C language.

Line 1 **#include <stdio. h>**

Is a preprocessor directive. Lines begin with # are processed by preprocessor before the program is compiled. The above line tells the preprocessor to include the contents of the **standard input/output header file** (stdio.h) in the program. This header file contains information and declarations used by the compiler when compiling standard input / output library functions such as **printf, scanf** etc.

Line 2 **main( )**

All C code runs inside functions. The most important function we find in any C program is called the main() function. The main() function is the starting point for all of the code in the program.

C program contains one or more functions, one of which must be **main().** Every program in C begins executing from the function **main().**

Line 3 **{**

The left brace, {, must begin the body of every function. A corresponding right brace must end each **function.** The portion of the program between the left brace and the right brace is also called a **block.** The block is an important program unit in C.

Line 4 **printf ("Welcome to C language \ n");**

The above statement instructs the computer to perform some action, here the action is to print on the screen the string of characters marked by the double quotes. So, it prints the message **"Welcome to C language"** on the screen. Notice that the characters '\n' were not printed on the screen. The backslash (\) is called on **escape character** and it indicates that printf is supposed to do something out of the ordinary properties. When printf looks ahead at the next character and combines it with the backslash to form an **escape sequence.** The escape sequence **\n** means newline and it causes the cursor to move to the beginning of next line.

Line 5 **}**

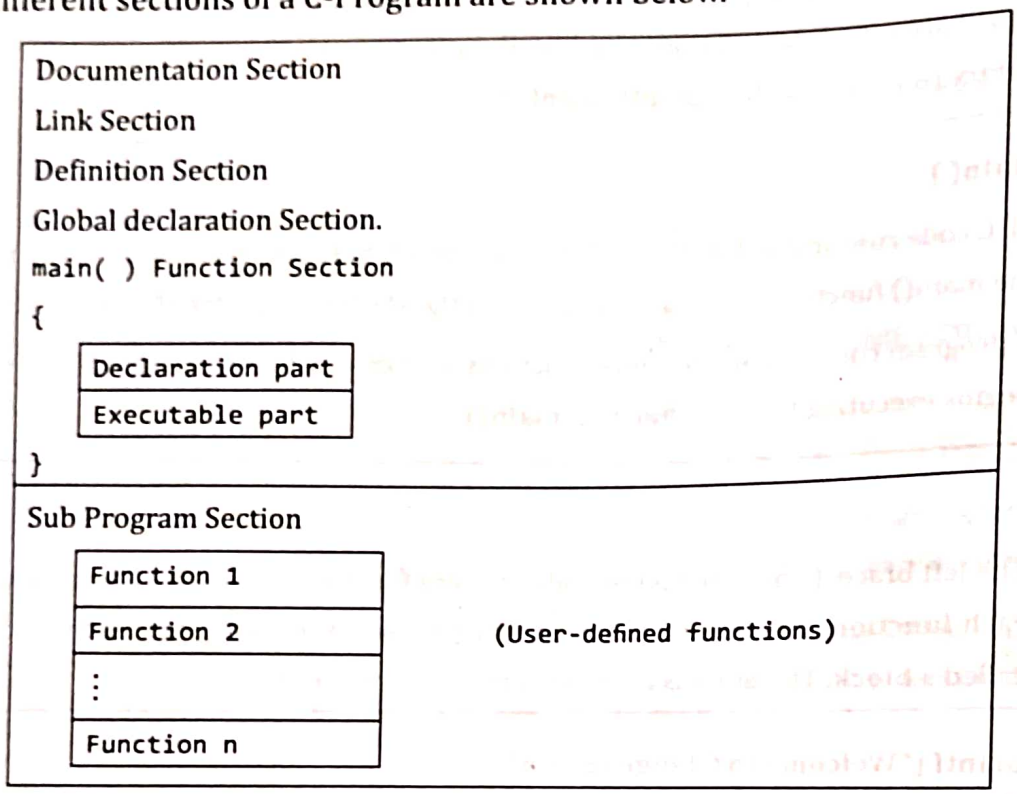
The right brace, }, indicates that end of the main has been reached.

Header File - stdio.h

C is a very, very small language and it can do almost nothing without the use of external libraries. We need to tell the compiler what external code to use by including header files for the relevant libraries. For example, the stdio library <stdio.h> contains code that allows to read and write data from and to the terminal.

3.7 Basic Structure of C Program

The C program usually contains one or more sections. To write a C program, we first create a function **main** to accomplish a specific task. A **function** is a subroutine that contains the statements, enclosing them within { and } to perform the required task. Except the function **main( )**, rest of the sections are optional. The different sections of a C-Program are shown below.



Structure of C – Program.

1. Documentation Section

Which contains comments such as program description, programmer's name, date on which it is written and other details required by the program.

A comment is one in C, when used in the program should be ignored by the compiler. In C '/*' is used to mark the start of a comment and it terminate with'*/'. There must be no gap between the asterisk (*) and slash (/). Comments may be placed any where in a program, as long as they do not appear in the middle of a keyword or identifier. Comments are used to enhance program's readability and understandability.

```
/ *
        Program Name     :  Demo Program
        Programmer       :  Srikanth . S.
        Purpose          :  To demonstrate the Structure of a program.
        Created on       :  01-11-2021 at 6 P.M.
* /
```

Documentation Section

2. Link Section

The Link Section helps to include the external file functions from the system library. We often use some mathematical functions like **cos, sin, exp** etc. The standard mathematical functions are defined and kept as a part of C math library. If we want to use any of these mathematical functions, we must add an **#include** instruction in the program, which instructs the compiler to link the specified function from the library. All the mathematical functions are available in **<math.h>**. Similarly to include some standard input and output functions, we should use **stdio.h**, which refers to the **standard input/output header file** containing standard input and output functions.

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

Link Section

The files **stdio.h** and **math.h** are called as **header files**. When we compile our C Program the header files are also compiled with our original program.

What are header files? Why it is important?

Th header files contain the set of predefined standard library functions. We use a header file in a program by including it with the C preprocessing directive "#include". All the header file have a '.h' an extension. By including a header file, we can use its contents in our program. Preprocessor directives are processed before a source program is handed over to the compiler. These should be placed in a source program before the main().

3. Definition Section

The definition section defines all the symbolic constants. To define any symbolic constant to our program we should use **#define** instruction. This #define instruction defines value to a symbolic constant for use in the program.

Example

```
#define   MONTHS   02
#define   AMOUNT   10000
```

Definition Section

Whenever a symbolic name is encountered, the compiler substitutes the value associated with the name automatically. In the above example, we have defined two symbolic constants MONTHS and AMOUNT and assigned values 12 and 10000 respectively. The values remains constant throughout the execution of the program. Since they are constants we may not change their values in our program.

#define is a preprocessor directive and **#define** lines should not end with a semicolon. Generally symbolic constants are written in upper case.

4. Global Declaration Section

The global declaration section contains variables that are used in more than one function. Any variable which is declared in this section is available to all functions.

Example

```
#include <stdio.h>
#define MONTHS 10
int    a = 10;          } → Global declaration
float  b = 3.14;
main( )
{
   ..............
}
```

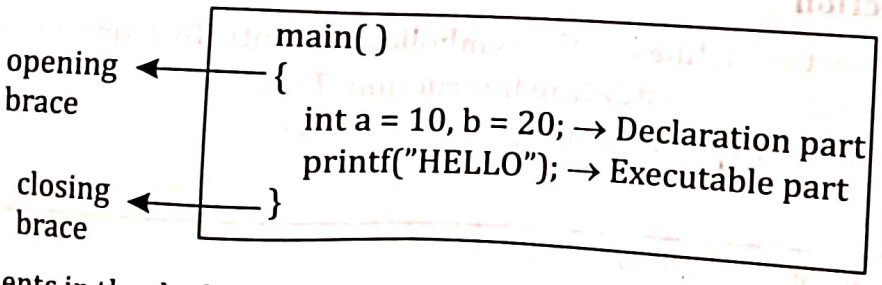
5. Main Function Section

All the C Programs consists of one (or) more functions. But there must always be a function called **main( )**. Every C Program should begin with this function main( ).

This section contains two parts.

- Declaration Part
- Executable Part

The declaration part declares all the variables used in the executable part. These two parts must be within the opening and closing braces({ and }). The program execution starts from the opening brace '{' and ends at the closing brace '}'.



All the statements in the declaration part and executable part must end with a semicolon.

Note:

All the sections may be an optional but the **main( )** function is always required.

6. Sub-Program Section

The **Sub-Program Section** contains user-defined functions. These functions are called in the **main( )** function. We can create any number of our-own functions. These functions are generally placed after the main function, although they may appear in any order.

Example

```
MyFunction( )
{
      printf ("HELLO" );
}
YourFunction( )
{
      printf ("HAI");
}
```

User – defined functions (Sub-program section)

In the above example, we have created two functions of our own. First function is **MyFunction( )** and second one is **YourFunction( )**. Like this we can create any number of functions. These functions are executed only when we call these functions from the **main( )** function.

Program To demonstrate all the Sections of a C-Program

```
/ *    Program Name      :  Demo Program
       Programmer        :  Srikanth .S
       Date              :  11-01-2021
       Time              :  7 A.M.    */
#include <stdio.h>
#include <math.h>

#define MONTHS 12
#define AMOUNT 10000

int a = 10, b = 20;

main( )
{
       int x = 30, y ;
       printf (“%d”, a) :
       y = MONTHS - 2;
       printf (“%d”, y);
       MyFunction( );
}
MyFunction( )
{
       printf ("%d %d”, a, b);
}
```

} → Documentation Section.

} → Link Section

} → Definition Section

} → Global declaration section

} → Main function section

} → Declaration part

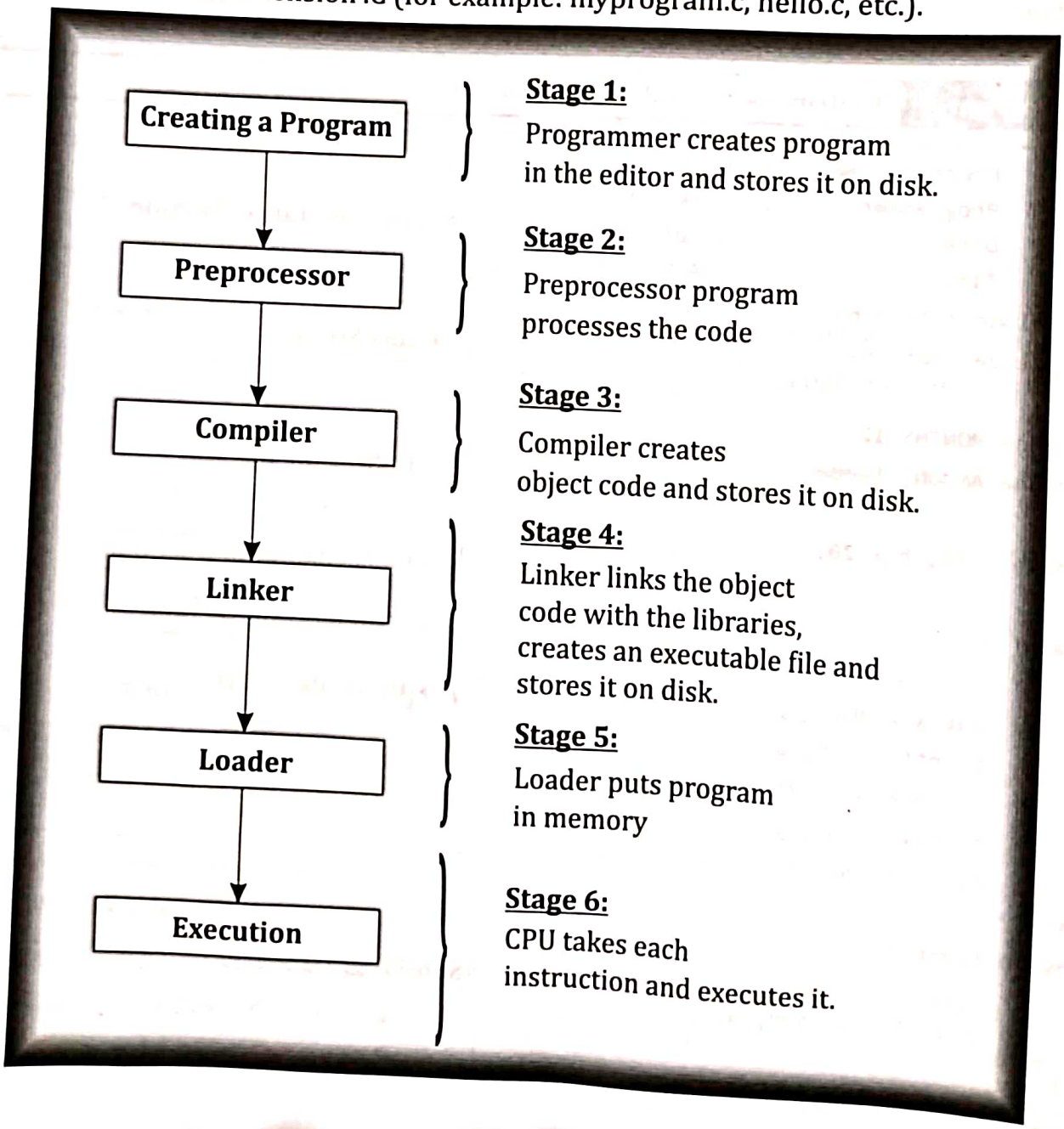
Sub-Program Section

3.8 Creating and Executing a 'C' – Program

Generally, the programs created using programming languages like are written using a high-level language like English. But, the computer cannot understand the high-level language. It can understand only low-level language. So, the program written in the high-level language needs to be converted into the low-level language to make it understandable for the computer. C programs typically go through six stages to be executed.

Stage 1: Creating a C Program

- The first step in creating programs is, writing or editing the program.
- A program can be written in any text editor like notepad. After writing a program, the program must be saved,
- In C language, the program is saved in disk with the extension ".c". This is the source program written in a high-level language. The program is also called as source code.
- C program file has extension .C (for example: myprogram.c, hello.c, etc.).



Stage 2 : Preprocessing

- After creating or editing source code we need to compile it by using compiler. In this stage preprocessor program processes the program before compilation phase begins.
- ***The C preprocessor obeys special commands called preprocessor directives, that begin with # (known as directives).***
- The preprocessor removes comments, text replacements, expands macros and expands included files. It will look for the header files used in program such as #include <stdio.h> and copy the header file into the source code file. The preprocessor also generates macro code and replaces symbolic constants defined using #define with their values. At the end of preprocessing stage, the program or source code will be expanded.

Stage 3: Compiling

- In this stage generated output file after preprocessing (expanded source code) will be passed to the compiler for compilation.
- The compiler translates the C program into machine language-code (also referred to as object code or object file).
- If any syntax errors in the source program, then compiler checks for it. Once all the errors are corrected then compiler converts the source program into its equivalent language code called as the Object program (or) Object code. (for example: myprogram.obj, hello.obj, etc.). If compiler detects error in the program, then we need to return to stage 1 to make correction in source code.

Stage 4: Linking

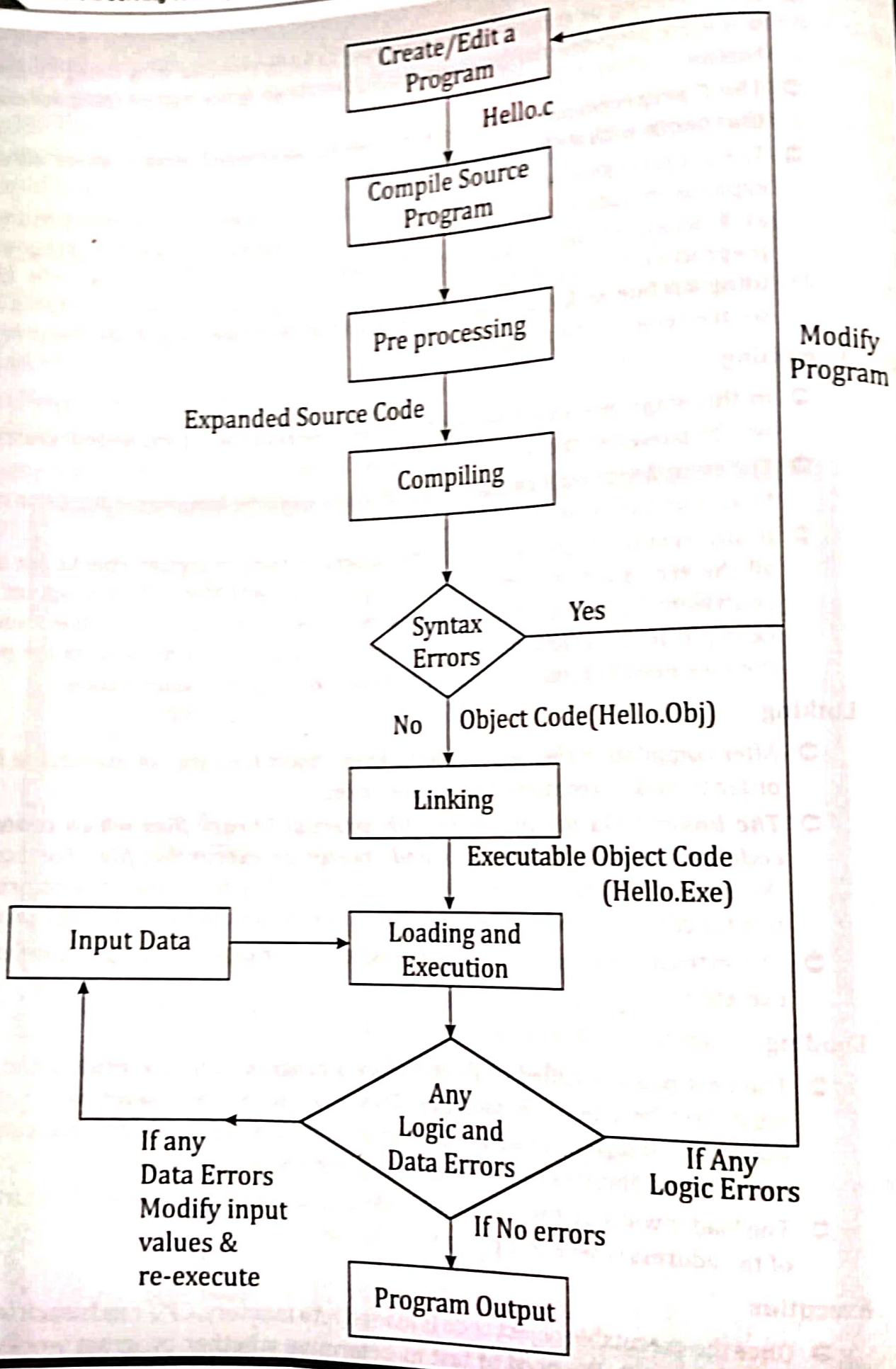
- After compilation the next step is linking. Object files are not executable file so in order to make executable file we use linker.
- ***The linker links the program with external library files which contains the code for predefined functions and creates an executable file.*** For example, if the program contains sqrt() function (built-in-function), then the object code of this function should be brought from math.h library and linked to the program.
- The extension of the executable file is ".exe" (for example: myprogram.exe, hello. exe, etc.).

Stage 5: Loading

- The next phase is called loading. Before a program can be executed, the program must first be placed in memory. This is done by the loader, which takes the executable image from disk and transfers it to memory. Additional components from shared libraries that support the program are also loaded.
- The loader will load the .exe file in RAM and inform the CPU with the starting point of the address where this program is loaded.

Stage 6: Execution

- Once the executable object code is loaded into memory, CPU reads each instruction and executes it. We need to test to determine whether program works properly or not. If program does not work properly we need to return to stage 1 for modifications.



Process of Compiling and Running a Program

3.9 Programming Style

- All statements should be written in lowercase letters. Uppercase letters are only used for symbolic constants.
- The programmer can write the statement any where between the two braces following the declaration part. The user can also write more than one statement in one line separating them with a semicolon.

```
a = b+c;
d = e * f;
   or
a = b+c; d = e * f;
```

- The opening and closing braces should be balanced. i.e, if opening braces are three; then closing braces should also be three;
- Use comments whenever necessary. Comments increases the program readability and also help to understand the program logic.

What is Modular Programming?

Modular programming is a program design technique in which a large program is divided into sub programs/ functions that are called modules, it improve maintainability of a program. It makes software development, debug, modify, update faster and easy.

QUICK FACTS

- Why C Language is named as C ? There is no such logic behind the naming of C Language. It was developed to cover all the inabilities of B language (simplified version of BCPL). So, it was just named C as it is next to B in the English alphabet.
- The C language is not called C at the beginning. It evolves from ALGO. Evolution of C language: ALGO -> BCPL -> B -> Tradition C -> K&R C -> ANSI C -> ANSI/ISO C -> C99
- C is the only programming language that exists for such a long period and still it is widely used.
- C is the basis of many other programming languages like C++, Java, JavaScript, Go, C#, PHP, Python, Perl, C-shell and many more.
- Unix was one of the first operating system kernels implemented in a language other than assembly and that was C.
- C18 is the latest version of C programming Language published in June 2018.
- ANSI published standards for C language.
- C is often referred to as the mother of all programming language because it is one of the most popular programming languages. Right from the time, it was developed, C has become the most widely used and preferred programming languages. Most of the compilers and kernels are written in C today.
- C language had provided many new concepts for the programming language like variables, data types, loops, array, function, file handling, dynamic memory allocation, pointers and most of these concepts are used in many modern languages like C++, Java, C#.
- Structured programming is a subset of procedural programming language. It is also known as modular programming. Structure programming makes use of the top-down model.