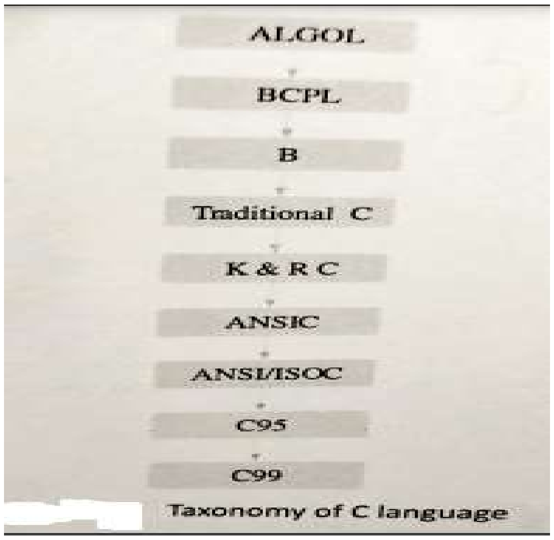
Chapter 4: Introduction to C

4.1 Introduction

The Programming language C was developed in the early 1970s by Dennis Ritchie at Bell Laboratories to be used by Unix Operating System. It was named 'C' because many of its features were derived from an earlier language called B. Although, C was designed for implementing system software, it was later on widely used for developing portable application software.

Background



Taxonomy of C language

1. ALGOL - Algorithmic Language uses block structure (1960)
2. BCPL - Basic Combined Programming Language (1967 – Martin Richards) – type-less, direct access of memory, so helped system programmers
3. B – (1970 Ken Thompson) Used to develop first version of UNIX.
4. Traditional C – (Dennis Ritchie 1972 at bell Laboratories) – ALGOL+BCPL+B+datatypes
5. K & R C – Traditional C was documented and popularized in the book 'The C Programming Language by Brian W. Kernighan and Dennis Ritchie in 1978.
6. Ansi C - The American National Standards Institute (ANSI) started working on defining the standard for C. This standard was approved in December 1989 and came to be known as ANSI C.
7. Ansi / ISO C - In 1990, the International Standards Organization (ISO) adopted the ANSI standard. This version of C came to be known as C89.
8. C95 - In 1995, some minor changes were made to C89; the new modified version was known as C95.
9. C99 - During 1990s, C++ and Java programming languages became popular among the users so the Standardization Committee of C felt that a few features of C++ / Java if added to C would enhance its usefulness. So, in 1999 when some significant changes were made to C95, the

modified version came to be known as C99.

Some of the changes made in the C99 version are as follows:

- Extension to character data types, so that they can support even non-English characters
- Boolean data type
- Extension to the integer data type
- Inclusion of type definitions in the for statement
- Inclusion of imaginary and complex types
- Addition of //, better known as C++ style line comment

Characteristics & Uses of C Language:

C is a robust language whose rich set of built-in functions and operators can be used to write complex programs.

The C compiler combines the features of assembly languages and high-level languages, which makes it best suited for writing system software as well as business packages.

- C is a high-level programming language, which enables the programmer to concentrate on the problem at hand and not worry about the machine code on which the program would be run.
- Small size - C has only 32 keywords. This makes it relatively easy to learn as compared to other languages.
- C makes extensive use of function calls.
- C is well suited for structured programming.
- Unlike PASCAL it supports loose typing (as a character can be treated as an integer and vice versa).
- Structured language as the code can be organized as a collection of one or more functions.
- Stable language. ANSI C was created in 1983 and since then it has not been revised.
- Quick language as a well written C program is likely to be as quick as or quicker than a program written in any other earlier language.
- Facilitates low level (bitwise) programming.
- Supports Pointers to refer computer memory, arrays, structures and functions
- C is a core language as many other programming languages (like C++, Java) are based on C
- If we know C, learning other computer languages becomes much easier.
- C is a portable language, i.e. a C program written for one computer can be run on another computer with little or no modification.
- C is an extensible language as it enables the user to add his own functions to the C library.
- C is often treated as the second best language for any given programming task.

4.2 Structure of a C Program

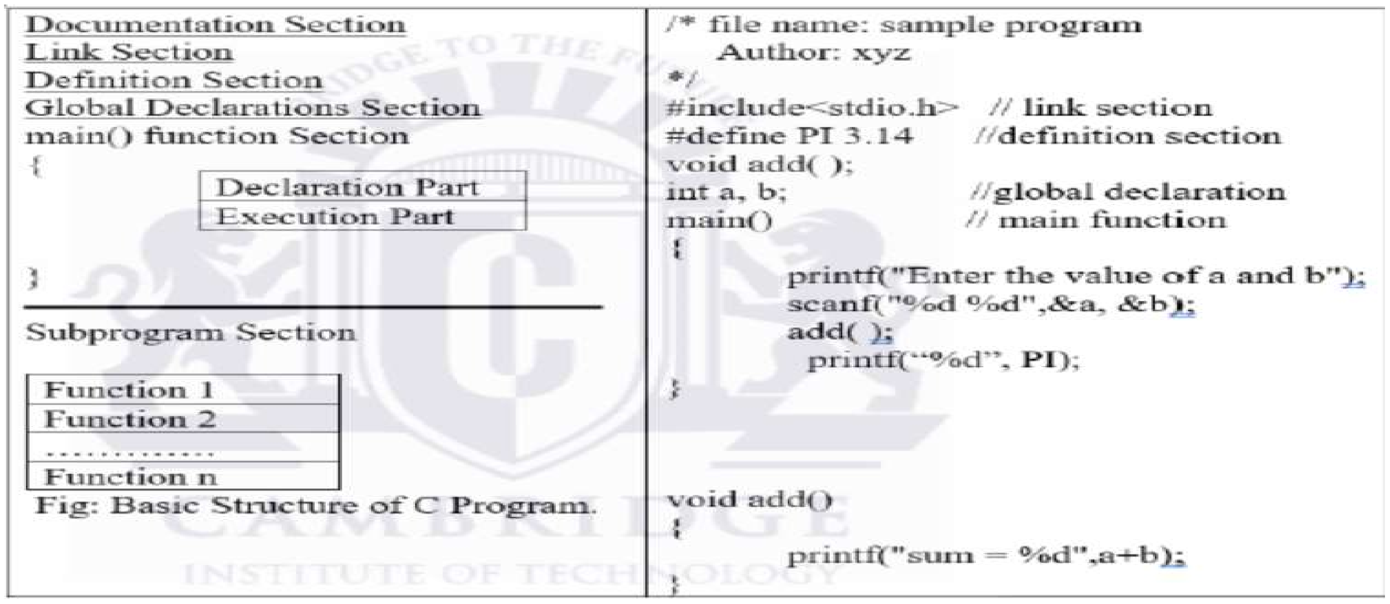
```
Documentation Section (optional)
Preprocessor Directives
Global Declarations
main()
{
    Local declarations
    Statements
}
function1()
{
    Local declarations
    Statements
}
……………….
functionn()
{
    Local declarations
    Statements
}
```

Structure of a C program

- A C program is composed of preprocessor commands, a global declaration section, and one or more functions.
- The preprocessor directives contain special instructions that indicate how to prepare the program for compilation.
- One of the most important and commonly used preprocessor commands is **include** which tells the compiler that to execute the program, some information is needed from the specified header file. Ex - **#include<stdio.h>, #define PI 3.14**
- Global declaration part will be revisited in the chapter on Functions.
- A program contains one or more functions, where a function is defined as a group of C statements that are executed together.
- The statements in a C function are written in a logical sequence to perform a specific task.
- The main() function is the most important function and is a part of every C program. The execution of a C program begins at this function.

- All functions (including main()) are divided into two parts the declaration section and the statement section.
- The declaration section precedes the statement section and is used to describe the data that will be used in the function.
- The data declared within a function are known as local declaration as that data will be visible only within that function, (i.e) the life-time of the data will be only till the function ends.
- The statement section in a function contains the code that manipulates the data to perform a specified task.
- From the structure we can conclude that a C program can have any number of functions depending on the tasks that have to be performed, and each function can have any number of statements arranged according to specific meaningful sequence.

Programmers can choose any name for functions. It is not mandatory to write Function1, Function2, etc., but with an exception that every program must contain one function that has its name as main().



```
Documentation Section
Link Section
Definition Section
Global Declarations Section
main() function Section
{
            Declaration Part
            Execution Part
}
________________________________
Subprogram Section

Function 1
Function 2
.............
Function n
```

Fig: Basic Structure of C Program.

```
/* file name: sample program
   Author: xyz
*/
#include<stdio.h>   // link section
#define PI 3.14     //definition section
void add( );
int a, b;           //global declaration
main()              // main function
{
        printf("Enter the value of a and b");
        scanf("%d %d",&a, &b);
        add( );
          printf("%d", PI);
}


void add()
{
        printf("sum = %d",a+b);
}
```

4.3 WRITING THE FIRST C PROGRAM

To write a C program, we first need to write the code. For this, open a text editor. If we are a Windows user we may use Notepad and if we prefer working on UNIX/Linux we can use emacs or vi editor. Once the text editor is opened on our screen, type the following statements

```
#include <stdio.h>
int main()
{
        printf("\n Welcome to the world of C");
        return 0;
}
```

Output

```
Welcome to the world of C
```

#include <stdio.h>

- This is a preprocessor command that comes as the first statement in our code.
- All preprocessor commands start with symbol hash (#).
- The include statement tells the compiler to include the standard input/output library or header file (stdio.h) in the program. This file has some in-built functions.
- By simply including this file in our code we can use these functions directly.
- The standard input output header file contains functions for input and output of data as reading values from the keyboard and printing the results on the screen.

int main()

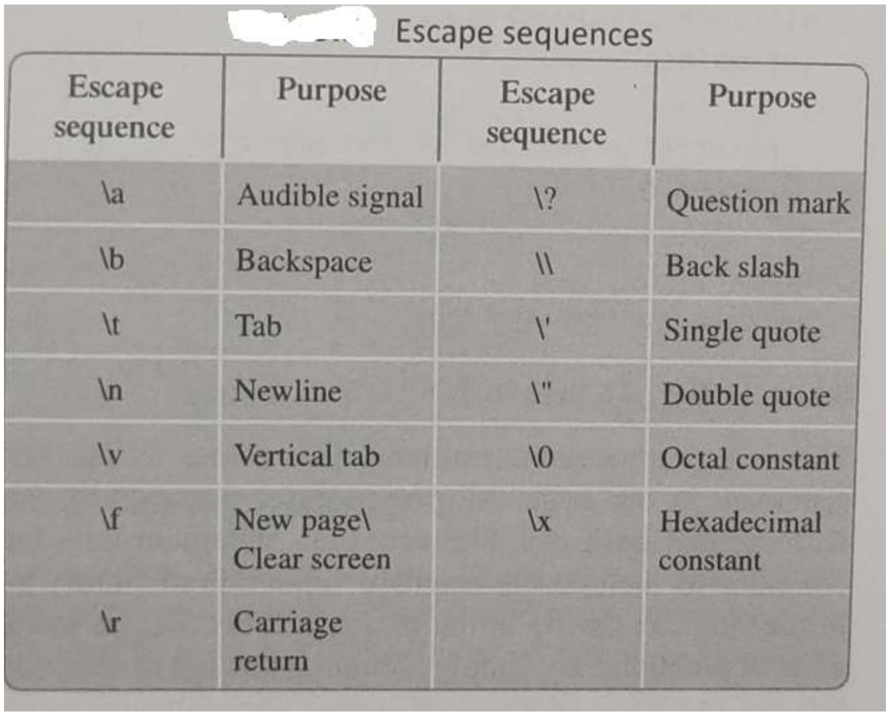
- Every C program contains a main() function which is the starting point of the program.
- int is the return value of the main() function.
- After all the statements in the program have been written, the last statement of the program will return an integer value to the operating system.

{} The two curly brackets

- These are used to group all the related statements of the main function. All the statements between the braces form the function body. The function body contains a set of instructions to perform the given task.

printf("\n Welcome to the world of C");

- The printf function is defined in the stdio.h file and is used to print text on the screen.
- The message that has to be displayed on the screen is enclosed within double quotes and put inside brackets.
- The message is quoted because in C a text (also known as a string between characters) is always put between inverted commas.
- \n is an escape sequence and represents a newline character.
- It is used to print the message on a new line on the screen.
- Like the newline character, the other escape sequences supported by C language :



[redacted] Escape sequences

Escape sequence	Purpose	Escape sequence	Purpose
\a	Audible signal	\?	Question mark
\b	Backspace	\\	Back slash
\t	Tab	\'	Single quote
\n	Newline	\"	Double quote
\v	Vertical tab	\0	Octal constant
\f	New page\ Clear screen	\x	Hexadecimal constant
\r	Carriage return		

If we do not place a parenthesis after main, a compiler error will be generated.

Placing a semicolon after the parenthesis of main function will generate a compiler error.

Escape sequences are actually non-printing control characters that begin with a backslash (\)

Every statement in the main function ends with a semi-colon (;)

return 0;

- This is a return command that is used to return the value 0 to the operating system to give an indication that there were no errors during the execution of the program.
- We have written all the statements using the text editor, save the text file as hello.c.
- If we are a Windows user then open the command prompt by clicking Start->Run and typing 'command' and clicking ok.
- Using the command prompt, change to the directory in which we had saved our file and then type: **tc hello.c**
- In case if we are working on UNIX/Linux operating system, then exit the text editor and type **cc hello.c –o hello**

  The -o is for the output file name. If we leave out the –o - then the file name a.out is used This command is used to compile our C Program.
- If there are mistakes in the program then the compiler will tell us the mistake we have made and on which line we made it.
- In case of errors we need to re-open our .C file and correct those mistakes.
- However, if everything is right then no error(s) will be reported and the compiler will

create an exe file for our program.

- This .exe file can be directly run by typing 'hello.exe' for Windows and './hello' for UNIX/ Linux operating system. When we run the .exe file, the output of the program will be displayed on screen. That is, Welcome to the world of C

> Note :
> The printf and return statements have been indented or moved away from the left side. This is done to make the code more readable.

- **Using Comments**

Comments are just a way of explaining what a program does. The compiler ignores comments when forming the object file. Comments are non-executable statements. Commented statements can be used anywhere in the program.

C supports 2 types of comments:

1) Line comment : // is used to comment a single statement. A line comment can be placed anywhere on the line and it does not require to be ended.
2) Block comment : /*is used to comment multiple statements. A /* is ended with */ and the sattements that lie within these characters are commented.

```
/* Author : Sonia
File name : hello.c
Description : to print 'welcome to the world of C' on the screen */
#include <stdio.h>
int main()
{
        // Prints message
        printf("\n Welcome to the world of C");
        return 0; // returns a value 0 to the OS
}
```

Output

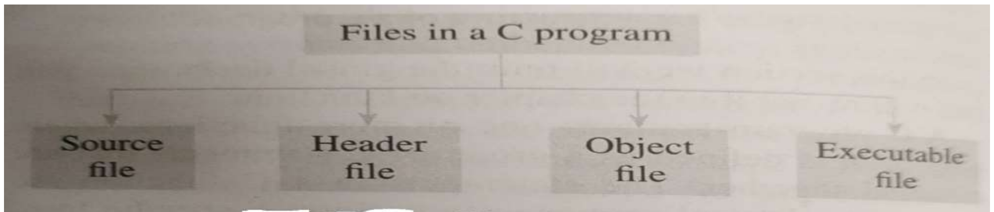
```
    Welcome to the world of C
```

4.4 FILES USED IN A C PROGRAM

Every C Program has four kinds of files associated with it. These include:

4.4.1 Source Code Files

The source code file contains the source code of the program. The file extension of any C source code file is '.c'. This file contains C source code that defines the main function and may be other functions. The main() function is the starting point of execution when we successfully compile and run the program. Ex – hello.c



Files in a C program

4.4.2 Header Files

When working with large projects, it is often desirable to separate out certain subroutines from the main() function of the program and store them in a different file known as header file.

The advantages of header files can be realized in the following cases:

- The programmer wants to use the same subroutines in different programs.
- The programmer wants to change or add subroutines and have those changes reflected in all the other programs.

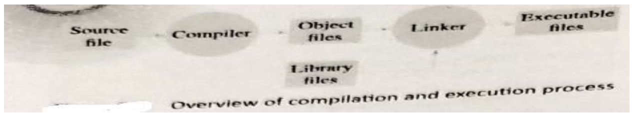
Header files names ends with a '.h' extension

Although some standard header files are automatically available to C programmers, in addition to those header files, the programmer may have his own **user-defined header files**.

- Standard Header Files - In the program, we used printf() function that has not been written by us. We do not know the details of how this function works. Such functions that are provided by all C compilers are included in standard header files.
- Examples of these standard header files include:
  string.h: for string handling functions
  stdlib.h: for some miscellaneous functions
  stdio.h: for standardized input and output functions
  math.h: for mathematical functions
  alloc.h: for dynamic memory allocation
  conio.h: for clearing the screen:
- All the header files are referenced at the start of the source code file that uses one or more functions from that file.

4.4.3 Object Files

Compiler - A compiler is a special program that translates a programming language's source code into object code.



Overview of compilation and execution process

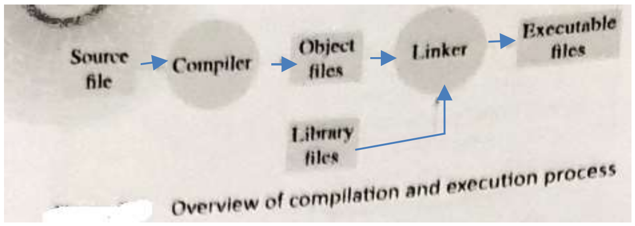
Compilation - The compilation is the process of transforming source code into object code. Object files are generated by the compiler as a result of processing the source code file. Object files have a '.o' extension, although some operating systems including Windows and MS-DOS have a '.obj' extension for the object file.

4.4.4 Binary Executable Files

The binary executable file is generated by the linker. The linker links the various object files to produce a binary file that can be directly executed. On Windows operating system, the executable files have a '.exe' extension.

4.5 COMPILING AND EXECUTING C PROGRAMS

- C is a compiled language. So once a C program is written, we must run it through a C compiler that can create an executable file to be run by the computer.
- While the C program is human-readable, the executable file, on the other hand, is a machine-readable file available in an executable form.
- **Every programming language has its own compiler.**
  The compiler translates the source code into an object code. The object code contains the machine instructions for the CPU, and calls to the operating system. However, even the object file is not an executable file.
- Therefore, in the next step, ***the object file is processed with another special program called a linker. The output of the linker is an executable or runnable file.***



- *In C language programs, there are two kinds of source files. In addition to the main (c) source file, which contains executable statements there are also header (.h) source files.*
- *Since all input and output in C programs is done through library functions, every C program therefore uses standard header files.*
- *So, preprocessing is done by the pre-processor program before the actual compilation.*
- *The preprocessor reads the source file as text, and produces another text file as output.*
- *Source code lines which begin with # symbol are written in preprocessor language.*
- *The output of a preprocessor is a text file which does not contain any preprocessor statements. This file is then processed by compiler and then by linker to produce the final executable file.*

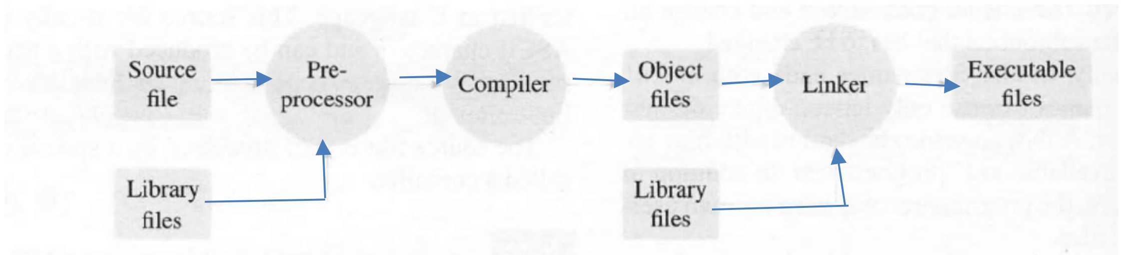


Figure 9.5 Preprocessing before compilation

- In modular programming, the source code is divided into two or more source files. All these source files are compiled separately thereby producing multiple object files. These object files are then combined by the linker to produce an executable file.

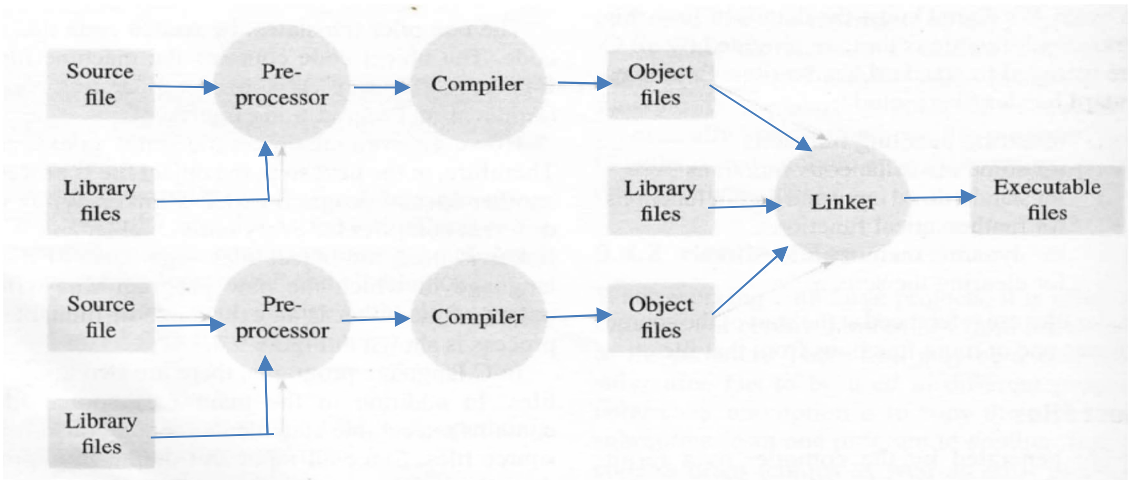
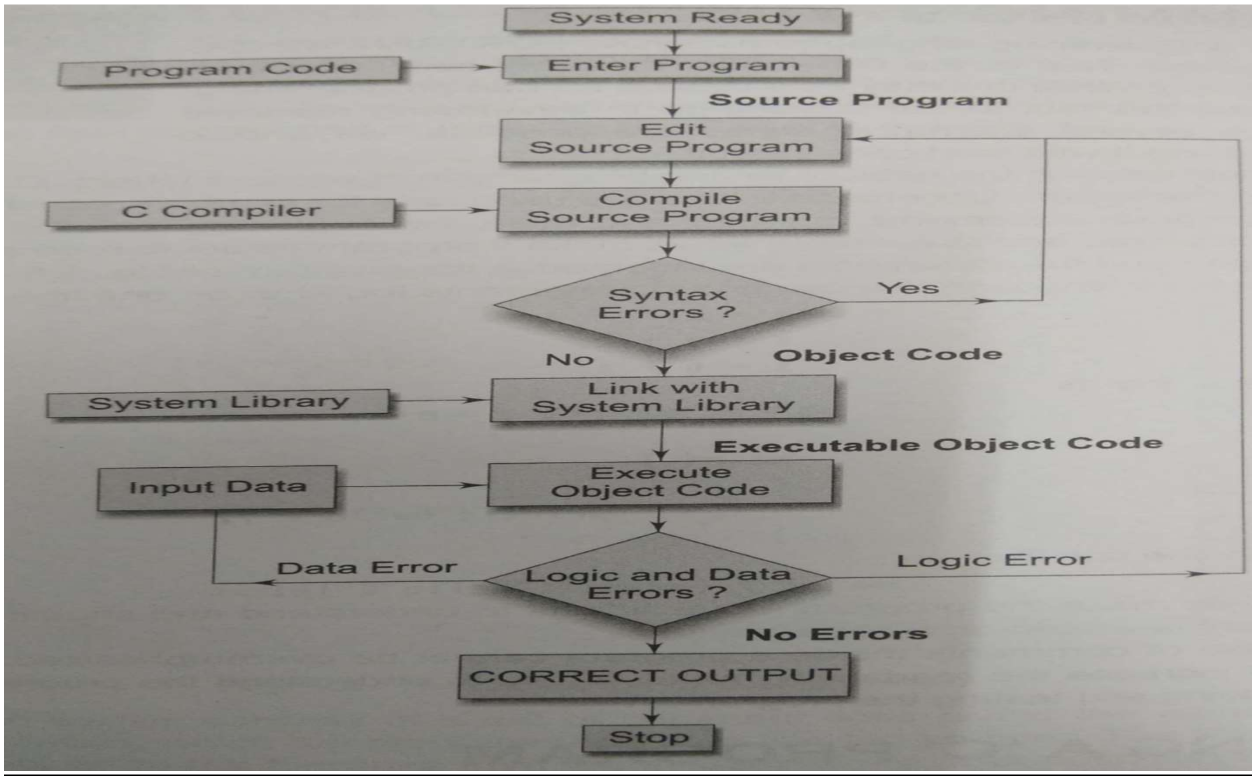


Figure 9.6 Modular programming—the complete compilation and execution process


System Ready
Program Code
Enter Program
Source Program
Edit Source Program
C Compiler
Compile Source Program
Syntax Errors ?
Yes
No
Object Code
System Library
Link with System Library
Executable Object Code
Input Data
Execute Object Code
Data Error
Logic and Data Errors ?
Logic Error
No Errors
CORRECT OUTPUT
Stop