SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

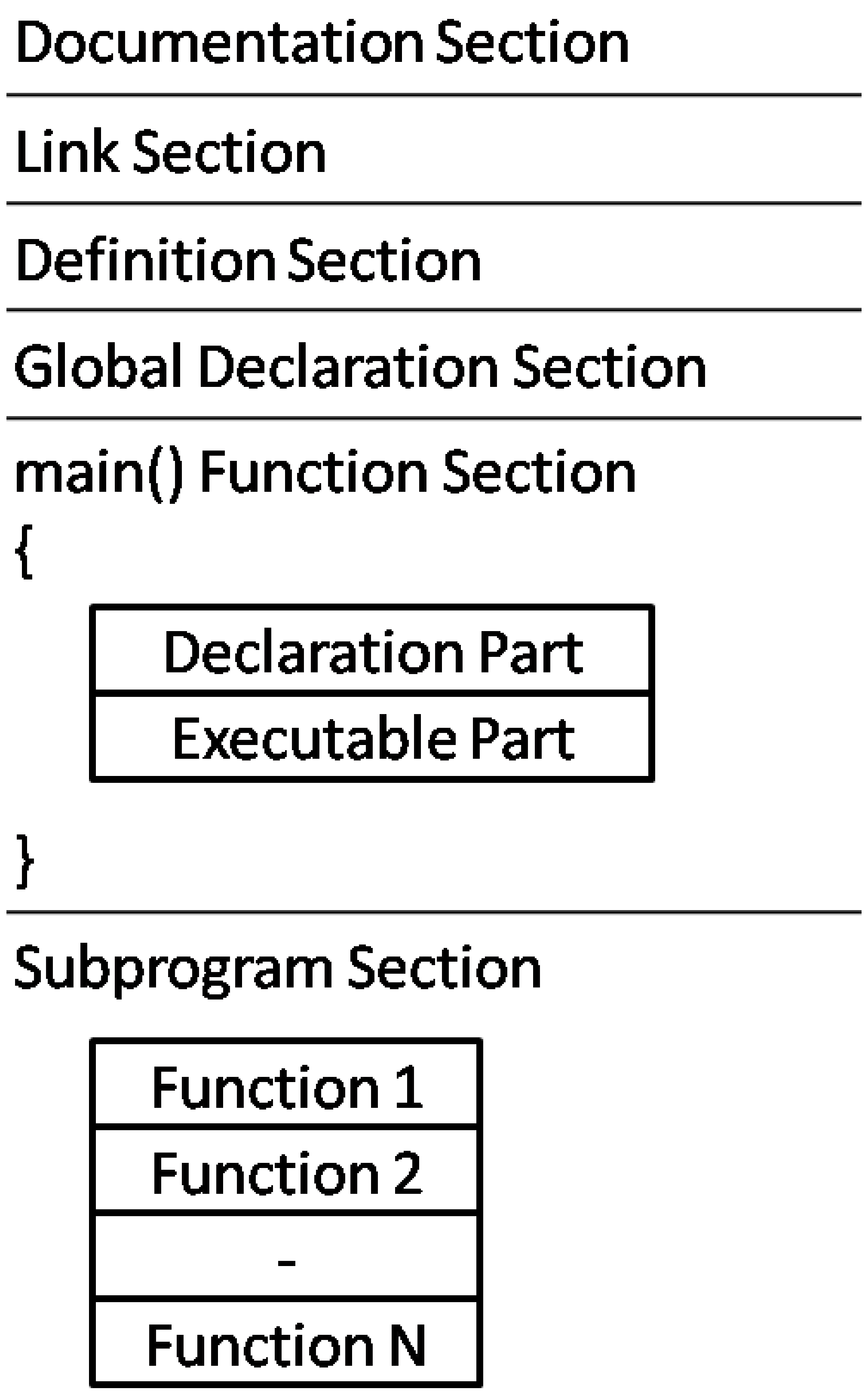
19ITT101-PROGRAMMING IN C AND DATA STRUCTURES

I YEAR - II SEM

UNIT 1 – INTRODUCTION TO C

TOPIC 5 – Structure of a 'C' program

STRUCTURE OF C PROGRAM



Documentation Section

Link Section

Definition Section

Global Declaration Section

main() Function Section

{

Declaration Part

Executable Part

}

Subprogram Section

Function 1

Function 2

-

Function N

STRUCTURE OF C PROGRAM

- C program can be viewed as a group of building blocks called functions.
  - A function is a subroutine that may include one or more statements designed to perform a specific task.

- 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.

- Link Section
  - The link section provides instructions to the **compiler to link functions** from the system **library**.

- Definition Section
  - The definition section defines all symbolic constants.

STRUCTURE OF C PROGRAM

- Global Declaration Section
  - There are some variables 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
- Main() Function Section
  - Every C program must have one main() function section.
  - This section contains two parts:
    - Declaration part
      - »declares all **the variables** used in the executable part
    - Executable part.
      - »There should be at least **one statement** in the executable part.
  - These 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.

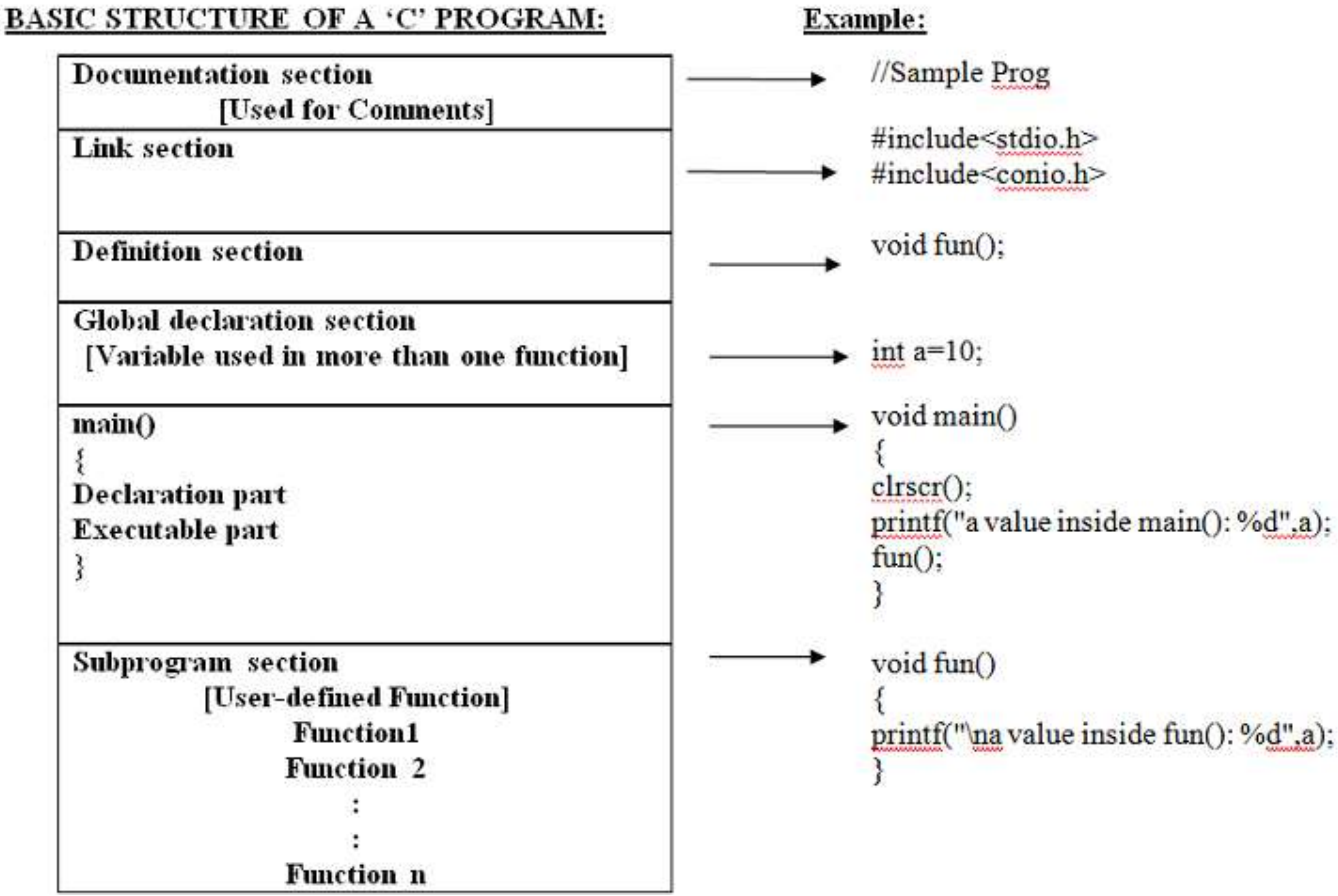
STRUCTURE OF C PROGRAM

- The closing brace of the main function section is the logical end of the program.
- All statements in the declaration and executable parts end with a semicolon(;).

- 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 the main function section may be absent when they are not required

STRUCTURE OF C PROGRAM



BASIC STRUCTURE OF A ‘C’ PROGRAM:

Documentation section **[Used for Comments]**
Link section
Definition section
Global declaration section **[Variable used in more than one function]**

main()
{
Declaration part
Executable part
}

Subprogram section **[User-defined Function]** **Function1** **Function 2** **:** **:** **Function n**

Example:

```
//Sample Prog

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

void fun();

int a=10;

void main()
{
clrscr();
printf("a value inside main(): %d",a);
fun();
}

void fun()
{
printf("\na value inside fun(): %d",a);
}
```

EXECUTING A 'C' PROGRAM

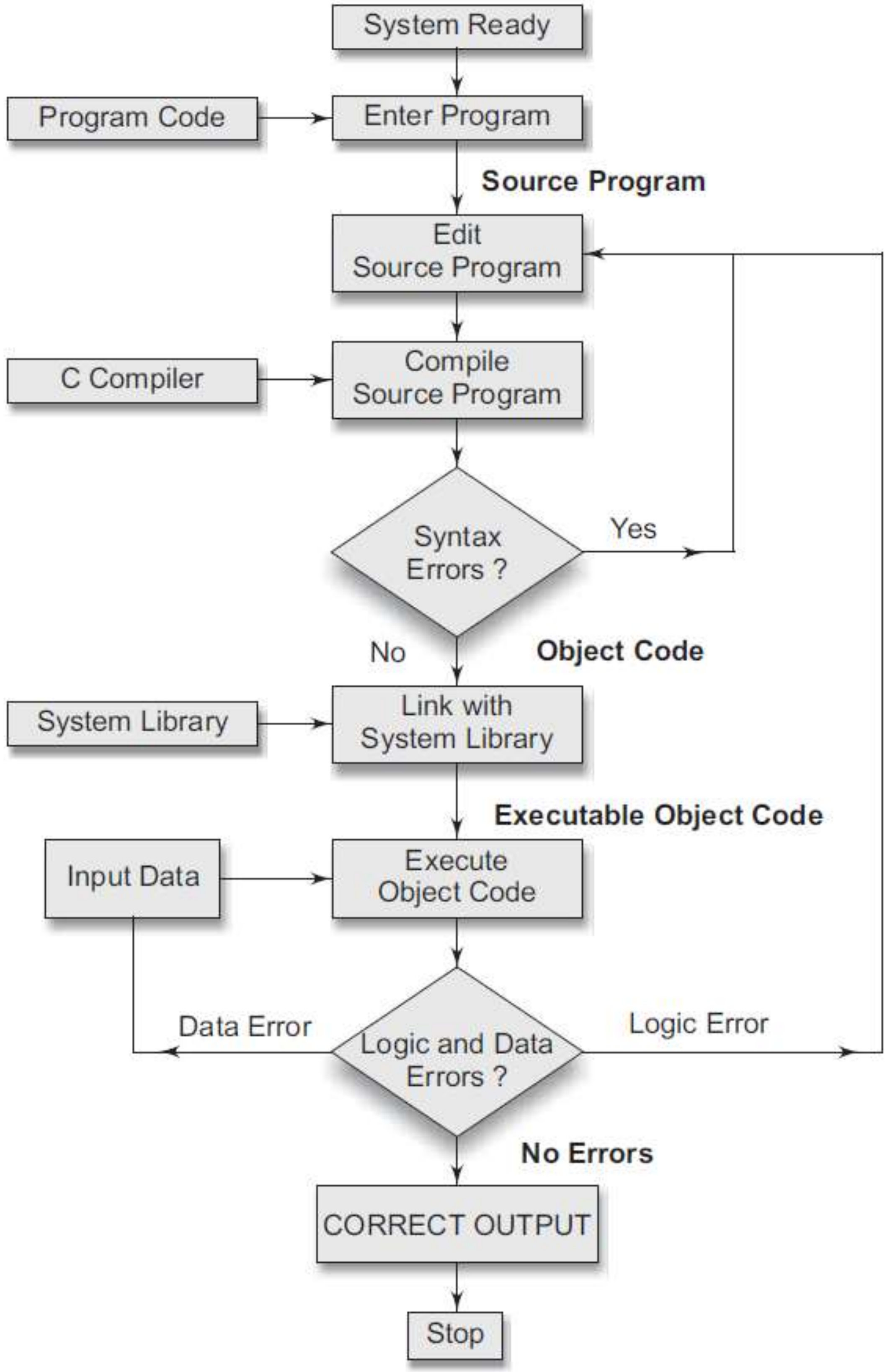
➢Executing a program written in C involves a series of steps. These are:

1. **Creating** the program;

2. **Compiling** the program;

3. **Linking** the program with functions that are needed from the C library; and

4. **Executing** the program.



EXECUTING A ‘C’ PROGRAM

➢ 1. Creating the Program

- The program must be **entered into a file**.
- The file name can consist of letters, digits and special characters, followed by a dot and a letter c.
- Examples of valid file names are:
  - ❖hello.c
  - ❖program.c
  - ❖ebg1.c
- The file is created with the help of a text editor (ex. notepad) and some **standard C editors**.
- The **program that is entered into the file is known as the source program**, since it represents the original form of the program

EXECUTING A ‘C’ PROGRAM

- **2. Compiling**
  - Let us assume that the source program has been created in a file named ebg1.c.
  - Now the program is ready for compilation.
  - The source program instructions are **now translated into a form that is suitable** for execution by the computer.
  - The translation is done after examining each instruction for its correctness.
  - If everything is alright, the compilation proceeds silently and the translated program is stored on another file with the name **ebg1.o.**
  - This program is known as **object code.**

EXECUTING A ‘C’ PROGRAM

- **<u>3. Linking</u>**
  - Linking is the process of putting together other program files and functions that are required by the program.
  - For example, if the program is using exp() function, then the object code of this function should be brought from the **math library** of the system and linked to the main program.
  - The linking is **automatically done** (if no errors are detected) in most of the Standard C Editors.

- If any mistakes in the **syntax and semantics** of the language are discovered, they are listed out and the compilation process ends right there.
- The errors should be corrected in the source program with the help of the editor and the compilation is done again.
- The compiled and linked program is called the <u>executable object code</u> and is stored automatically in another file named **<u>a.out</u>.**
- Note: Different systems use different compilation commands for linking various functions.

EXECUTING A ‘C’ PROGRAM

- **4.Executing the Program**
- Execution is a simple task
- **load the executable object code** into the computer memory and execute the instructions
- During execution, the program may **request for some data to be entered** through the keyboard.
- Sometimes the program does not produce the desired results.
- Perhaps, something is wrong with the program logic or data.
- Then it would be necessary to correct the source program or the data.
- In case the source program is modified, the **entire process** of compiling, linking and executing the program should be repeated.
- Note that the linker always assigns the same name a.out.
- When we compile another program, this file will be overwritten by the executable object code of the new program.