0301102 LOGIC DEVELOPMENT & PROGRAMMING

UNIT – 1

Introduction to Algorithms and Flowcharts

Basic Concept of Logic

What is Logic?

- Logic programming is a way of writing computer programs using languages that are based on formal logic.
- Logic is the study of how truth is defined, and how we prove that certain statements are true or false.
- Programming logic is used by programmers to model the programming language instructions carried out by the computer when the program is executed.

•

Basic Concept of Logic

- **Example:**
- **Consider the following procedure of washing clothes using a washing machine**
- Put the clothes in the washtub
- Pour water.
- Pour detergent powder.
- Switch on the washing machine.
- Set the timer and wait for a few minutes.
- Drain the water out.
- End.
- This procedure gets the work done

Basic Concept of Logic

- **Example:**
- **Now, suppose the same steps are performed in a slightly different order**
- Put the clothes in the washtub.
- Switch on the washing machine.
- Set the timer and wait for a few minutes.
- Pour water.
- Pour detergent powder.
- Drain the water out.
- End.
- ***In this case, your clothes may tear off, since you have switched on the washing machine before pouring water into it.***

Basic Concept of Logic

Simple Definition of logic:

- a proper or reasonable way of thinking about or understanding something
- a particular way of thinking about something
- the science that studies the formal processes used in thinking and reasoning

Features of Algorithm

What Is Algorithm?

The word “algorithm” relates to the name of the mathematician **Al-khowarizmi**, which means a procedure or a technique.

“An algorithm is a sequence of steps to solve a particular problem”

“An algorithm is a finite sequence of instructions, a logic and explicit step-by-step procedure for solving a problem starting from a known beginning”.

Features of Algorithm

- Algorithms have a definite beginning and a definite end, and a finite number of steps.
- An algorithm produces the same output information given the same input information, and several short algorithms can be combined to perform complex tasks such as writing a computer program.
- **A cookbook recipe, a diagnosis, a problem solving routine, are some common examples of simple algorithms.**

Advantages of Algorithm

- It is a step-wise representation of a solution to a given problem, which makes it easy to understand.
- An algorithm uses a definite procedure.
- It is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge.
- Every step in an algorithm has its own logical sequence so it is easy to debug.
- A sequential solution of any program that written in human language,called algorithm.
- Algorithm is first step of the solution process, after the analysis of problem, programmer write the algorithm of that problem.

Example of Algorithms:

How to Write Algorithm?

- **Step 1 Define your algorithms input:**

  e.g. to calculate the area of rectangle input may be the rectangle height and rectangle width.

- **Step 2 Define the variables:**

  e.g. We can define two variables for rectangle height and rectangle width as HEIGHT and WIDTH.

- **Step 3 Outline the algorithm's operations:**

  e.g. to find area of rectangle multiply the HEIGHT and WIDTH variable and store the value in new variable AREA.

- **Step 4 Output the results of your algorithm's operations**

  e.g. In case of area of rectangle output will be the value stored in variable AREA

How to Write Algorithm?

```
Step 1: Start
Step 2: Declare variables num1, num2 and sum.
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2 and assign the result to sum.
        sum ← num1+num2
Step 5: Display sum
Step 6: Stop
```

Flowchart

- **“Flowchart is diagrammatic /Graphical representation of sequence of steps to solve a problem. ”**
- **Graphical representation of any program is called flowchart.**
- Flowchart uses different symbols to design a solution to a problem
- The first design of flowchart goes back to **1945 which was designed by John Von Neumann**
- Flowchart is often considered as a blueprint of a design used for solving a specific problem.

Flowchart

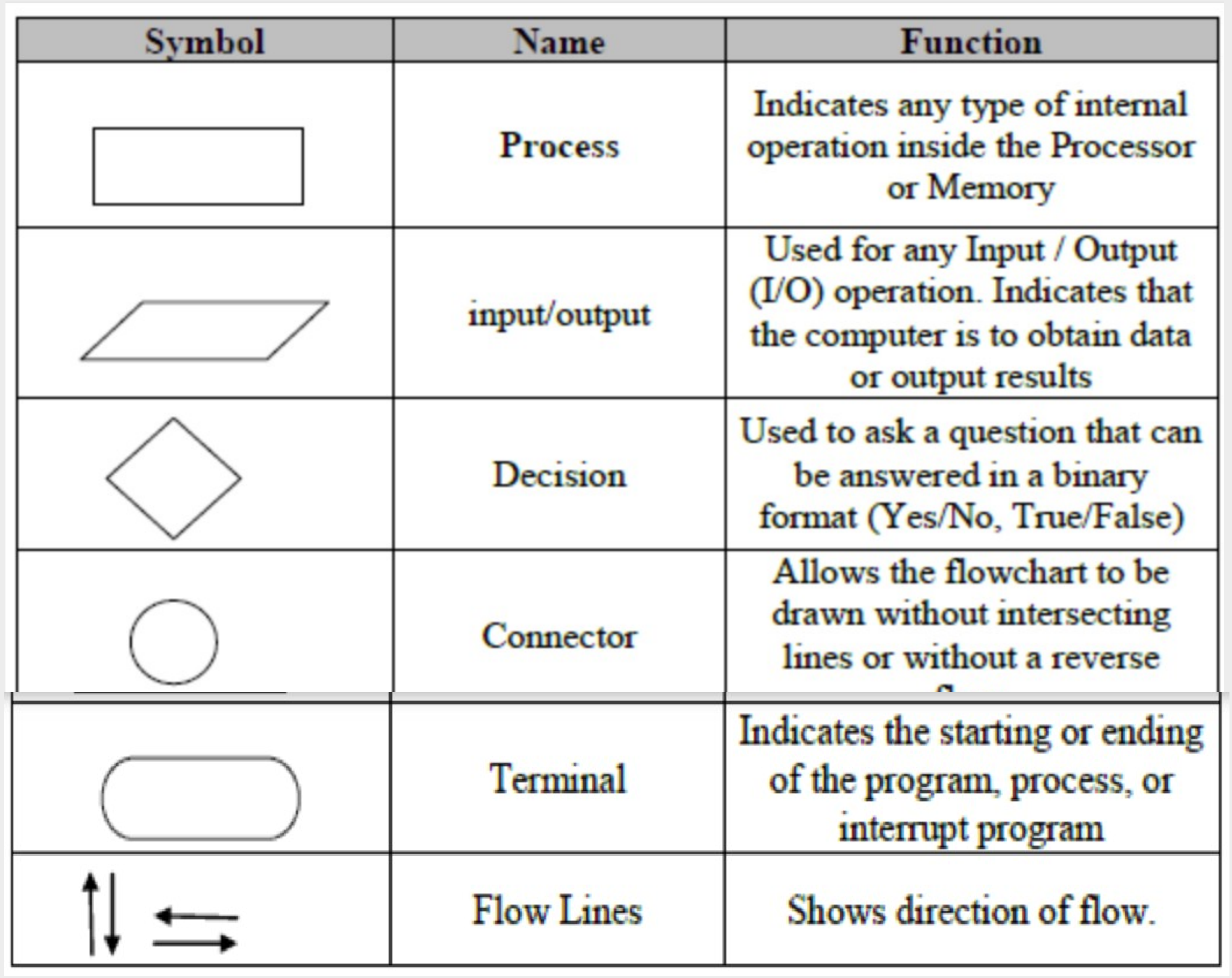
- **Steps for solving Problem:**
  - Define the problem
  - Idenify the input, output and constraints
  - Find various alternative solutions
  - Select the best possible alternatives
  - Prepare detailed stepwise result for the identified alternative
  - Compute the required result using the identified set of instructions
  - Check the correctness of the answer obtained

Advantages of flowchart

- Flowchart is an excellent way of communicating the logic of a program.
- Using flowchart Problem analysis is easy and efficient.
- Flowchart plays the role of a blueprint, which makes program development process easier.
- The flowchart makes program or system maintenance easier.
- It is easy to convert the flowchart into any programming language code.

- **There are some standard graphics that are used in flowchart as following:**

Symblos of Flowcharts



Symbol	Name	Function
▭	Process	Indicates any type of internal operation inside the Processor or Memory
▱	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
◇	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
○	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse
⬭	Terminal	Indicates the starting or ending of the program, process, or interrupt program
⇅ ⇄	Flow Lines	Shows direction of flow.

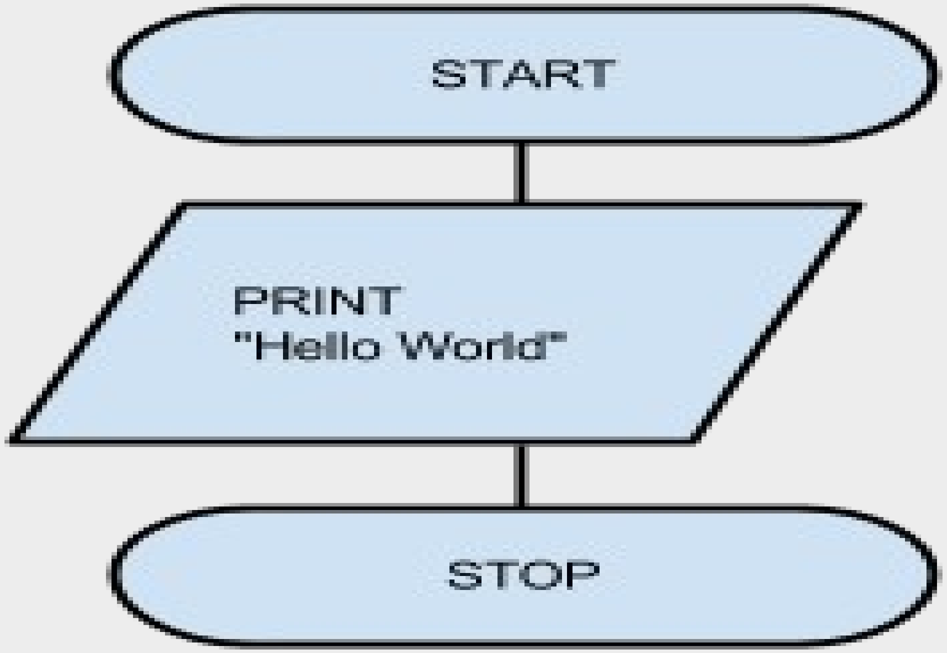
Simple flowcharts & Algorithm

Write an algorithm and drow flowchart to print “Hello World”

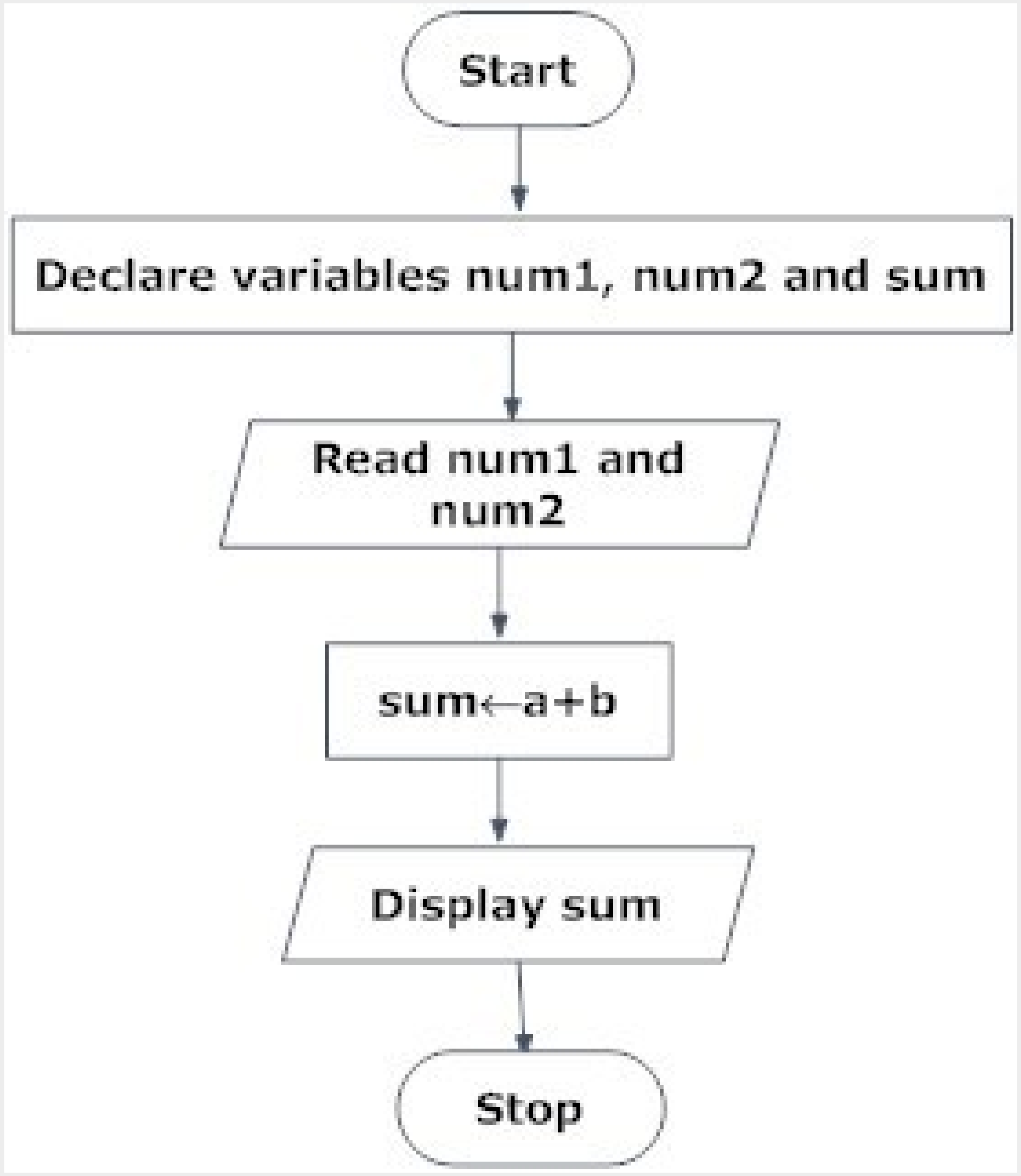
Step 1 – Start

Step 2 – Print “Hello World”

Step 3 – Stop



Flowchart


Start
Declare variables num1, num2 and sum
Read num1 and num2
sum←a+b
Display sum
Stop

Types of Structures

The algorithm and flowchart include following three types of control structures.

1.**Sequence**: In the sequence structure, statements are placed one after the other and the execution takes place starting from up to down.

2.**Branching (Selection):** In branch control, there is a condition and according to a condition, a decision of either TRUE or FALSE is achieved. In the case of TRUE, one of the two branches is explored; but in the case of FALSE condition, the other alternative is taken. Generally, the 'IF-THEN' is used to represent branch control.

3.**Loop (Repetition):** The Loop or Repetition allows a statement(s) to be executed repeatedly based on certain loop condition e.g. WHILE, FOR loops.

Sequential Flowcharts & Algorithm

- Step – 1 start
- Step – 2 Input First number A
- Step – 3 Input Second number B
- Step – 4 SUM = A + B
- Step – 5 Display SUM
- Step – 6 Stop

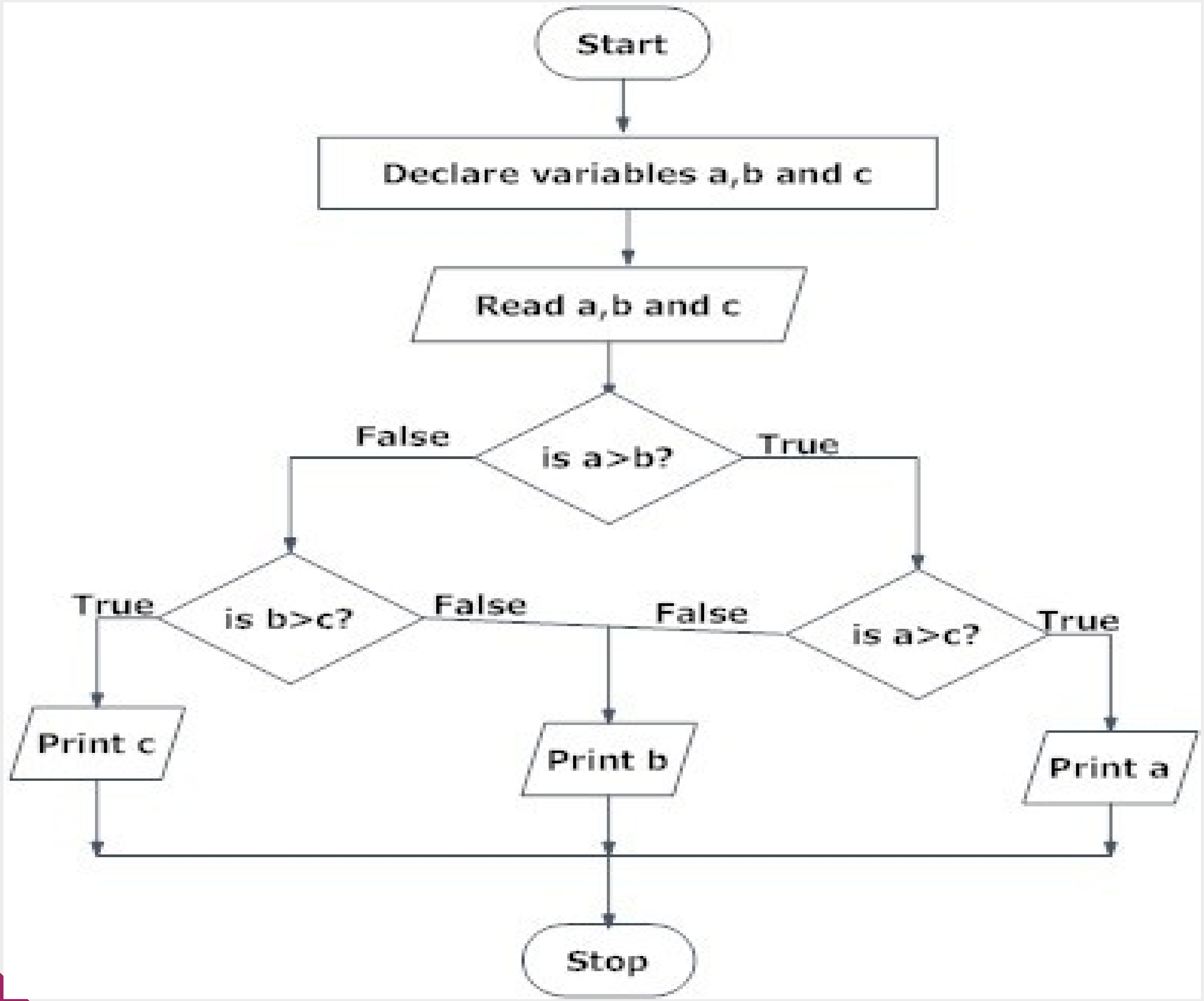
Decision Making Flowcharts & Algorithm

```
Step-1 Start
Step-2 Input two numbers say NUM1,NUM2
Step-3 IF NUM1 < NUM2 THEN
print smallest is NUM1
 ELSE
print smallest is NUM2
 ENDIF
Step-4 Stop
```

Write an algorithm to find the largest among three different numbers entered by the user.

```
Step 1:   Start
Step 2:   Declare variables a,b and c.
Step 3:   Read variables a,b and c.
Step 4:   If a > b
              If a > c
                  Display a is the largest number.
              Else
                  Display c is the largest number.
          Else
              If b > c
                  Display b is the largest number.
              Else
                  Display c is the greatest number.
Step 5:   Stop
```

Write an algorithm to find the largest among three different numbers entered by the user.



Introduction to Programming

- A computer is a device that can accept human instruction, processes it and responds to it or a computer is a computational device which is used to process the data under the control of a computer program.
- Program is a sequence of instruction along with data.
- A program is a set of instructions given to a computer to perform a specific operation or computer is a computational device which is used to process the data under the control of a computer program.
- While executing the program, raw data is processed into a desired output format.
- These computer programs are written in a programming language which are high level languages.

Introduction to Programming

- Like we have different languages to communicate with each other, likewise, we have different languages like C, C++, C#, Java, python, etc to communicate with the computers.
- The computer only understands binary language (the language of 0’s and 1’s) also called machine-understandable language or low-level language but the programs we are going to write are in a high-level language which is almost similar to human language.
- The main() is a standard function that you will always include in any program that you are going to create from now onwards.
- Note that the execution of the program starts from the main() function.

Introduction to Programming

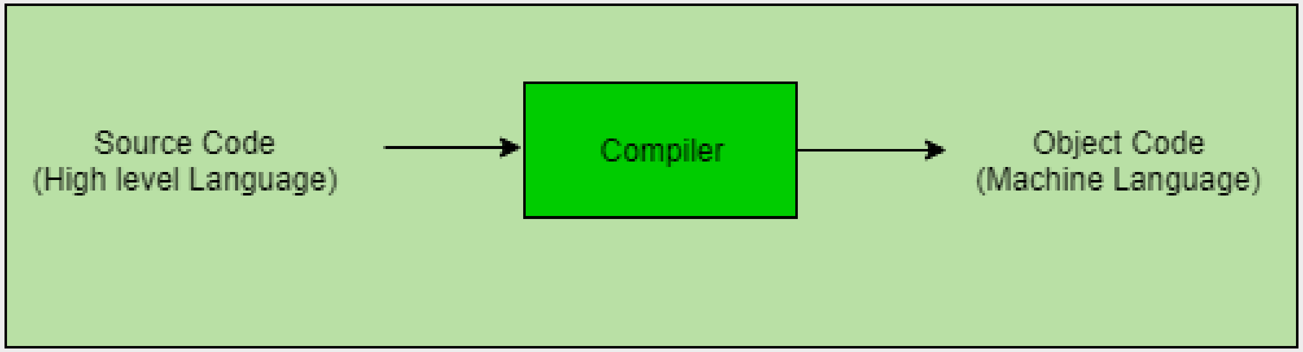
- **There are 3 types categories of language:**
- **Machine Language**
- **Assembly Language**
- **Higher level Language**
- **Machine Language:** It is a computer's natural language which can be directly understood by the system. It is directly executed by the CPU. A program written in 1's and 0's is called Machine language(binary code).
- It is directly understood by the processor so has faster execution time since the programs written in this language need not to be tanslated.
- It is very difficult to program since all the instructions are to be represented by 0s and 1s.
- It is time consuming and difficult to find error and to debug.

Introduction to Programming

- **Assembly Language:** Assembly language is a little easier than machine language. It uses more convenient numbers, symbols, and abbreviations to describe the huge strings of 1s and 0s, to make it both easier and more memorable to type in instructions.
- The programs are written in **alphanumeric** symbols, instead of 0's and 1's. This numeric symbols are called **mnemonics** like ADD, SUB, PRINTF, etc.
- The program is converted into machine code by **assembler**.
- It is not **portable** because every processor has its own assembly langauge.
- **Higher level Language:** This language uses **English-like statements and symbols,** and are **independent** of the type of computer you are using. This language is very easy to understand and speedy also. The programs can be written in **English words.** It is **portable**.
- The high level language is converted into machine language by one of the two different languages translator programs; **interpreter or compiler.**

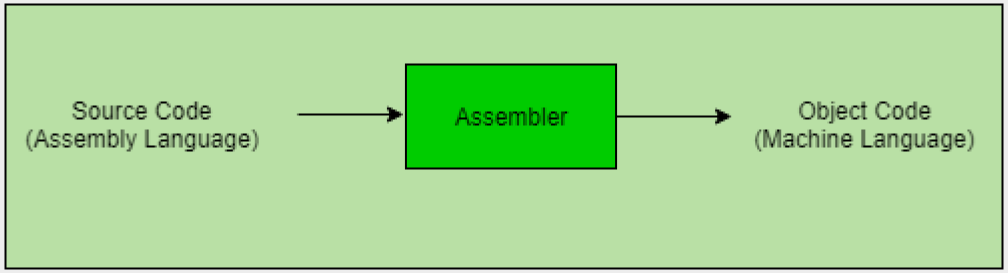
Introduction to Programming

- **Higher level Language:** The programs are written in high level languages like C, C++, Java, Python etc. and are called **source code.**
- These source code cannot be executed directly by the computer and must be converted into machine language to be executed.
- Hence, a special translator system software is used to translate the program written in high-level language into machine code is called **Language Processor** and the program after translated into machine code.
- **Compiler:** The language processor that reads the complete source program written in high level language as a whole in one go and translates it into an equivalent program in machine language is called as a **Compiler**.
- Example: C, C++, C#, Java
- The compiler specifies the errors at the end of compilation with line numbers when there are any errors in the source code.



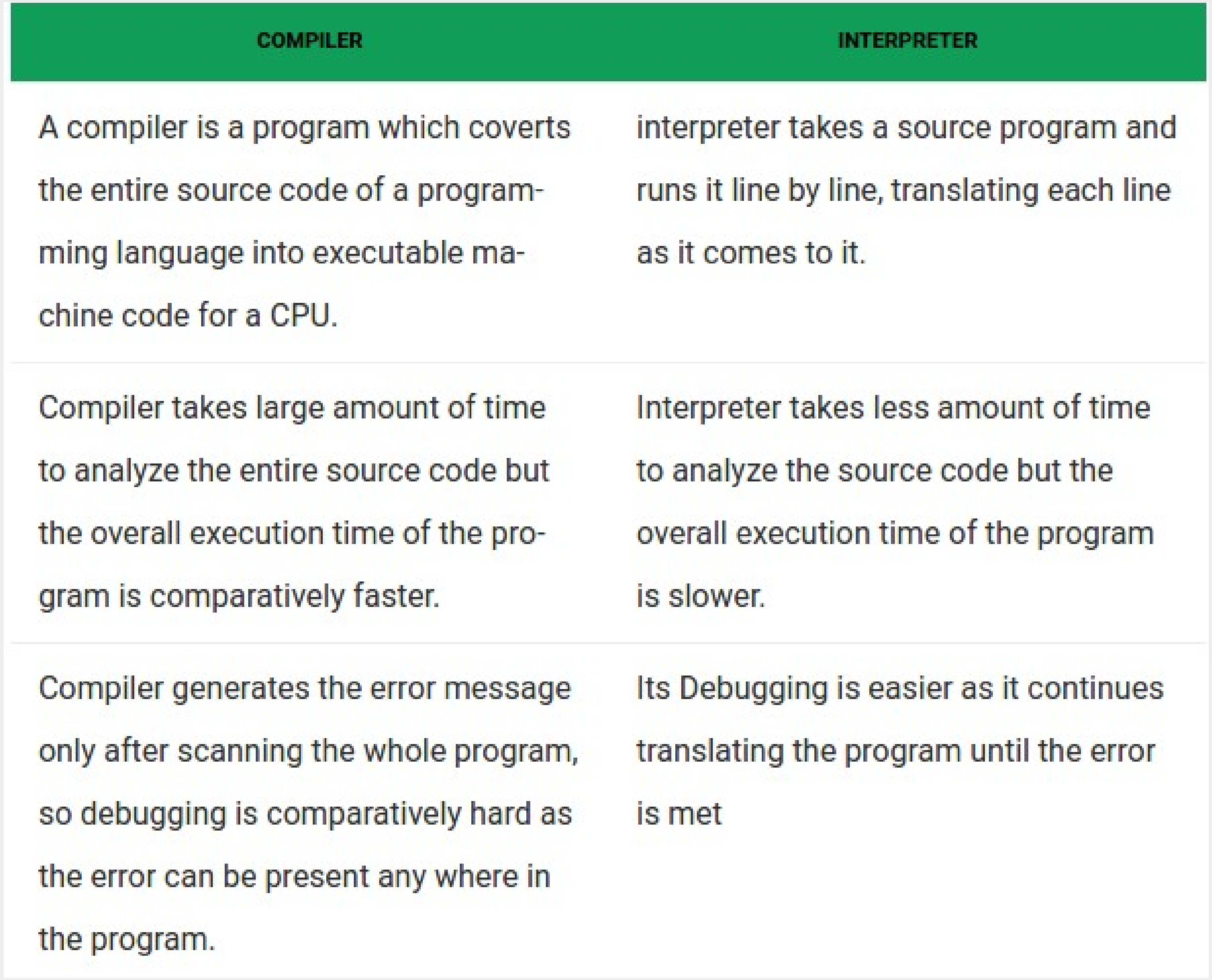
Introduction to Programming

- **Assembler:** The Assembler is used to translate the program written in Assembly language into machine code. The source program is a input of assembler that contains assembly language instructions. The output generated by assembler is the object code or machine code understandable by the computer.



- **Interpreter:** The translation of **single statement of source code** into machine code is done by language processor and executes it immediately before moving on to the next line is called an interpreter. If there is an error in the statement, the interpreter terminates its translating process at that statement and displays an error message. The interpreter moves on to the next line for execution only after removal of the error.
- Example: Perl, Python.

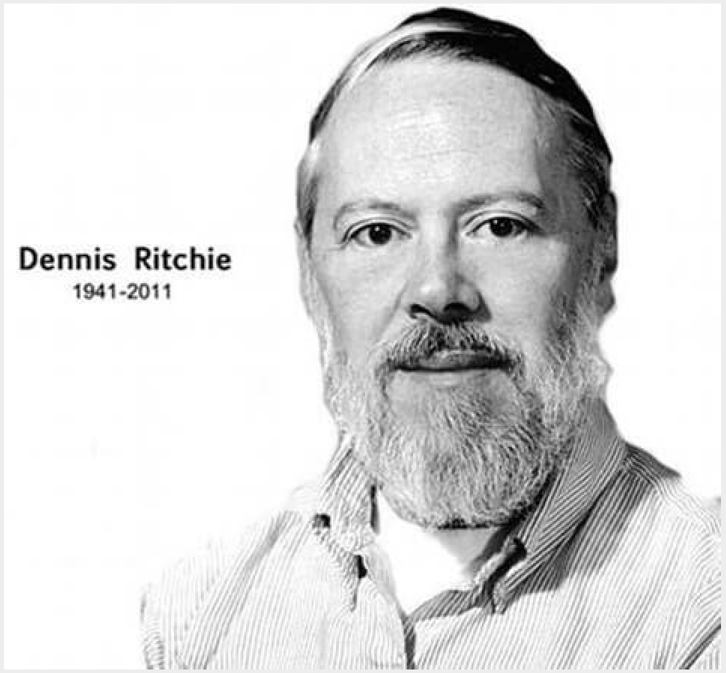
Introduction to Programming



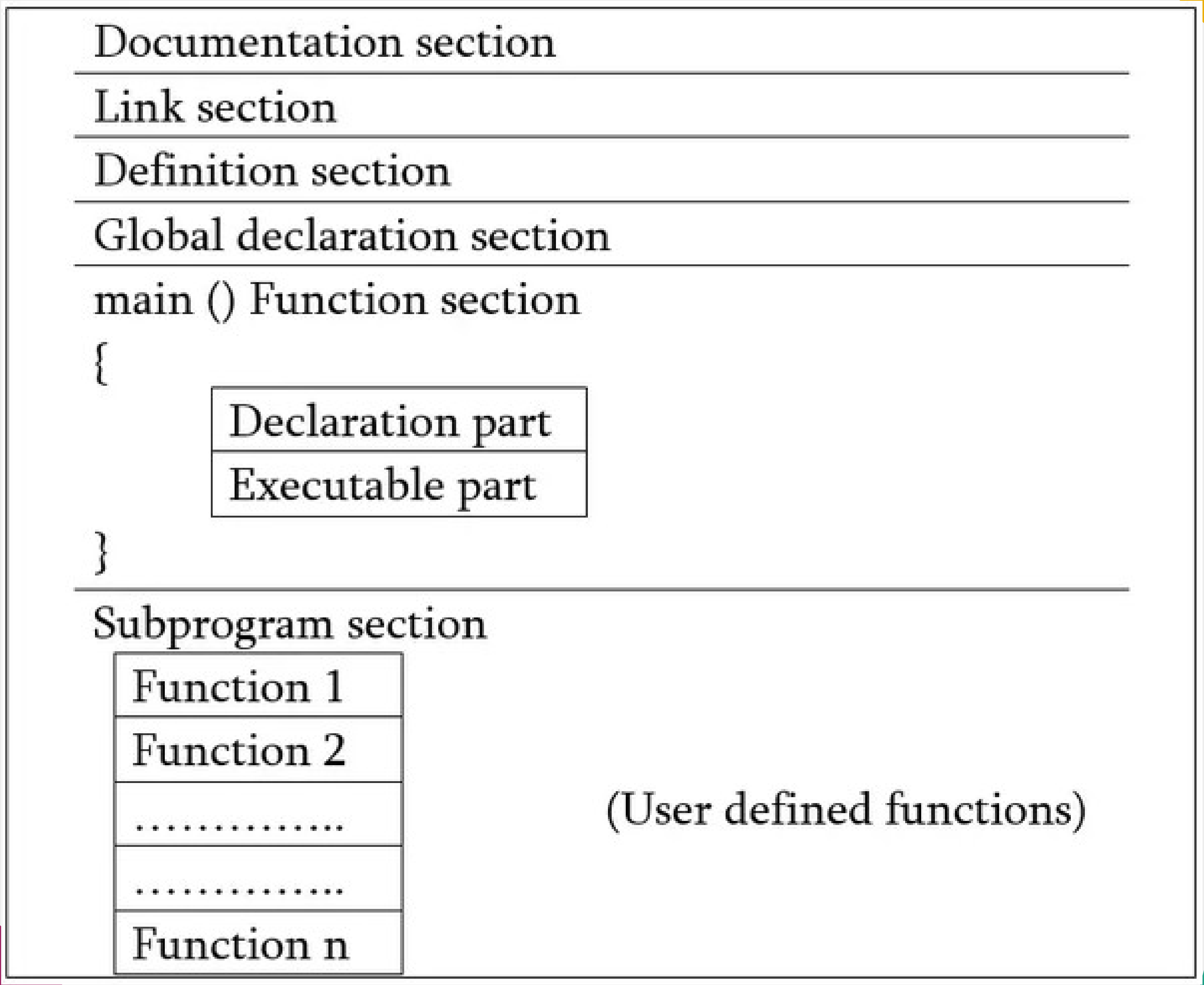
COMPILER	INTERPRETER
A compiler is a program which coverts the entire source code of a programming language into executable machine code for a CPU.	interpreter takes a source program and runs it line by line, translating each line as it comes to it.
Compiler takes large amount of time to analyze the entire source code but the overall execution time of the program is comparatively faster.	Interpreter takes less amount of time to analyze the source code but the overall execution time of the program is slower.
Compiler generates the error message only after scanning the whole program, so debugging is comparatively hard as the error can be present any where in the program.	Its Debugging is easier as it continues translating the program until the error is met

Introduction to C

- C is a general-purpose programming language that is extremely popular, simple and flexible.
- C is a procedural programming language.
- It was initially developed by **Dennis Ritchie** in the year 1972.
- It was mainly developed as a system programming language to write an **operating system.**
- C is a base for the programming.



Structure of 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

(User defined functions)

Structure of Program

- **Documentation section:** The Documentation section consists of a set of comment lines. Like 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 instruction to the compiler to link the header files or functions from the system library such as using the #include<stdio.h>.
- **Definition section:** The definition section defines all symbolic constants such by using the **#define** directive.
- **Global declaration section:** There are some variables that are used in more than one function, such variables are called global variables.
- In C there are two types of variable declaration,
  - **Local variable declaration:** Variables that are declared inside the main function.
  - **Global variable declaration:** Variables that are declared outside the main function.

Structure of Program

- **Main function section:** Every C-program should have **one main() function.**
- **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. These two parts must appear between the opening and closing braces. The program execution begins at the opening brace and ends at the closing brace. The closing brace of the main function is the logical end of the program. All statements in the declaration and executable part end with a **semicolon**.
- **Sub-program section:** If the program is a multi-function program, then the subprogram section contains all user-defined functions that are called in the main() function.

Structure of Program

```
//comments (if any)

#include <header files here>

int main()

{

  …

        logic

  …

}
```

Even though the comments are optional, it is **definitely recommend to have at least a single-line comment describing what the program does.**

The main() function is the first thing that runs. Everything that needs to be executed has to be somehow inside the main().

Structure of Program

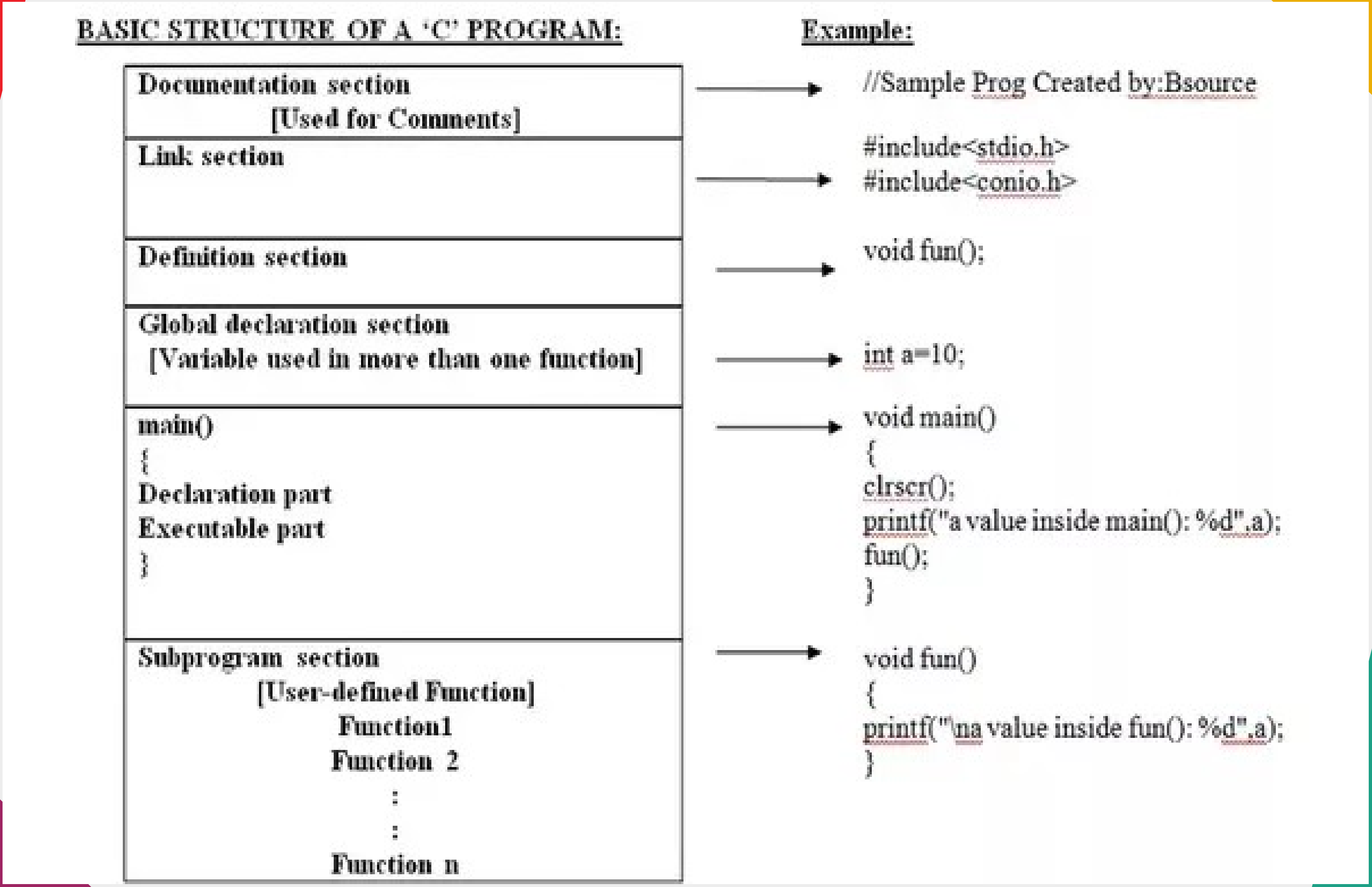
```
 #include <stdio.h>


int main() {
   /* my first program in C */
   printf("Hello, World! \n");
}
```

Structure of Program

- The first line of the program #include <stdio.h> is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation.
- The next line int main() is the main function where the program execution begins.
- The next line /*...*/ will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.
- The next line printf(...) is another function available in C which causes the message "Hello, World!" to be displayed on the screen.
- The next line return 0; terminates the main() function and returns the value 0.

Structure of 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 Created by:Bsource

#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);
}
```

Variables

- Variables – used in computer programming to store specific values within a program. In other words, it is used to store some form of data.
- A variable is a name of the memory location.
- Different types of variables require different amounts of memory.
- **Syntax to declare a variable:**
  - **data_type variable_name;**
- **The example of declaring the variable:**
  - int a,y,q,r,d;
  - float b;
  - char c;
- Here, a, b, c are variables. The int, float, char are the data types.

Variables

- We can also provide values while declaring the variables as given below:
  - int a=10,b=20;//declaring 2 variable of integer type
  - float f=20.8;
  - char c='A';
- **Rules for defining variables:**
  - A variable can have alphabets, digits, and underscore.
  - A variable name can start with the alphabet, and underscore only. It can't start with a digit.
  - Variables are case sensitive.
  - No special symbols are allowed other than underscore.
  - No whitespace is allowed within the variable name.
  - A variable name must not be any reserved word or keyword, e.g. int, float, etc.
- **Valid variable names:** int a; **OR** int _ab; **OR** int a30;
- **Invalid variable names:** int 2; **OR** int hello world; **OR** int long;