# C Programming Beginner to Advance

-Hariom Raj Chauhan

"It is based on the new syllabus."

# Day 1

# Welcome to Day 1 !

- **Topic:** Introduction to Computer Programming, Overview of C Programming, Operators and Expressions.
- **Objective:** To understand and master the basic concepts of C programming and program execution flow.
- **Agenda:**
  - Introduction to programming concepts
  - Setup C Environment + Configuration + Execution
  - Structure of a C program
  - Variables and data types
  - Operators and expressions
  - Hands-on coding and debugging with errors.

# 1. Introduction to Programming

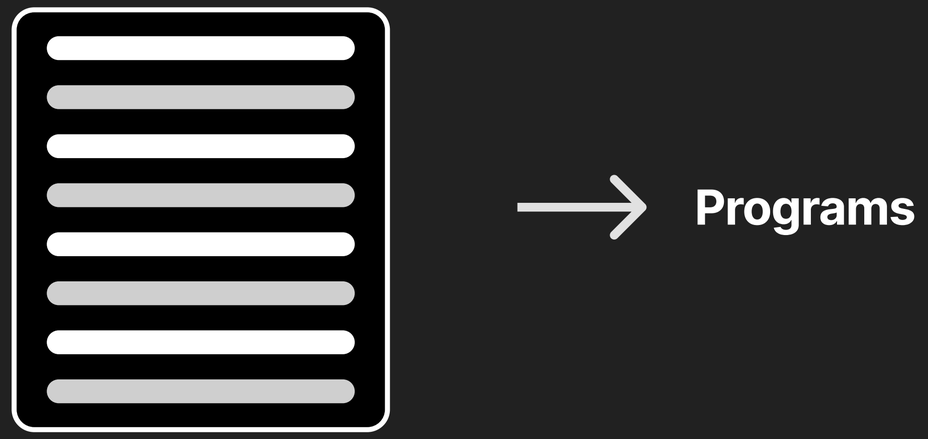
- What is a program and programming language
- Steps of problem solving
  - Problem analysis
  - Algorithm
  - Flowchart (basic symbols only)
- Program development steps
  - Coding
  - Compilation
  - Execution
  - Debugging (syntax vs logical errors)

## 1.1 Definitions

- **Computer programming** or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks.

- A **programming language** is a system of notation for writing computer programs. A programming language is usually described in terms of its syntax and semantics.

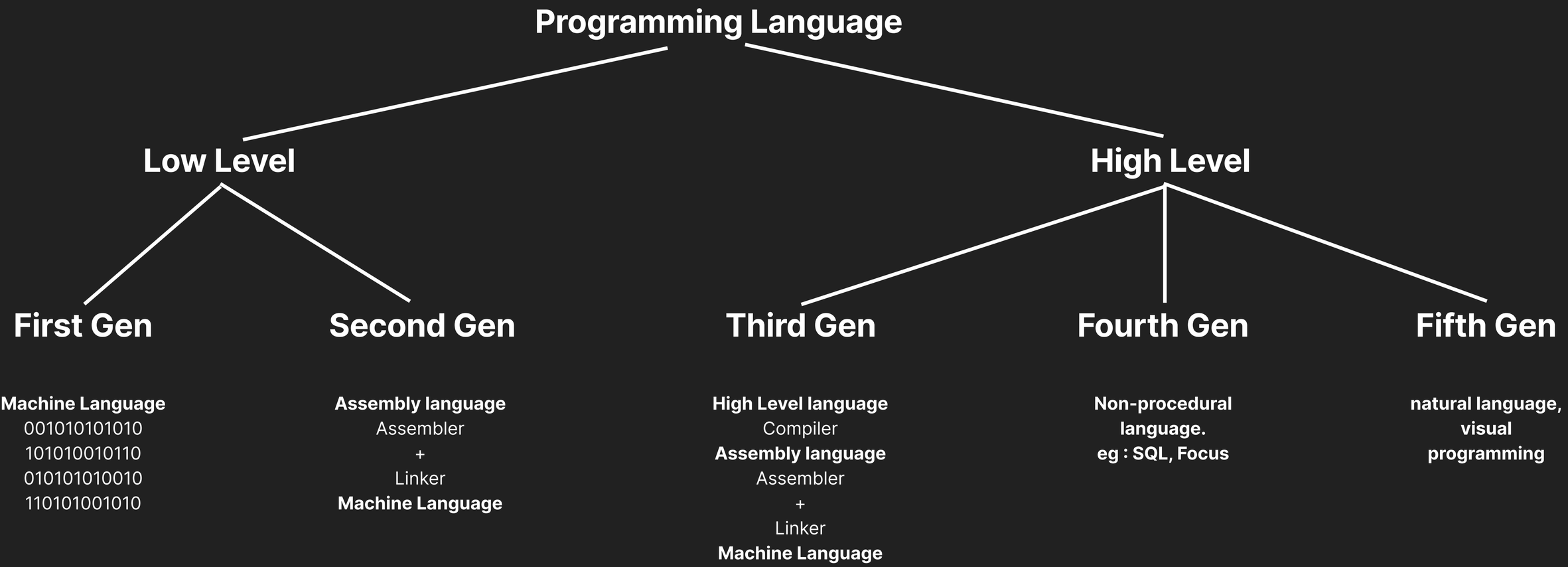
sequence of instructions



**Programs**

# 1.2 Types and Generations of Programming Languages

- There are 5 Generations of Programming Language.



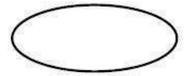
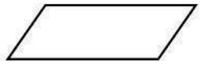
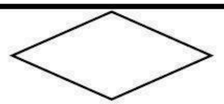
# 1.3 Problem-Solving using a Computer

Problem Analysis
↓
Algorithm Development
↓
Flowcharting
↓
Program Coding
↓
Compilation & Execution
↓
Debugging & Testing
↓
Documentation

## Algorithm

- Start
- Define constants here....
- Define Variables here....
- Read variables
- Calculate here...
- Display variables
- Stop

## Flowchart

| | | |
|---|---|---|
|  | Arrow | Used to connect flowchart symbols and direction indicate flow of logic. |
|  | Oval<br>Start/Stop/End | Used to represent beginning and end of task. |
|  | Rectangle | Used for arithmetic and data manipulation operations. |
|  | Input/output | Used for input and input. |
|  | Connectors | Used to different flow lines and remote parts of flow charts. |
|  | Decision | Used for decision making and branching operations that have two options. |
|  | Function Call | Used whenever u called the function. |
|  | Loops | Used to indicate for loops |

# Activity

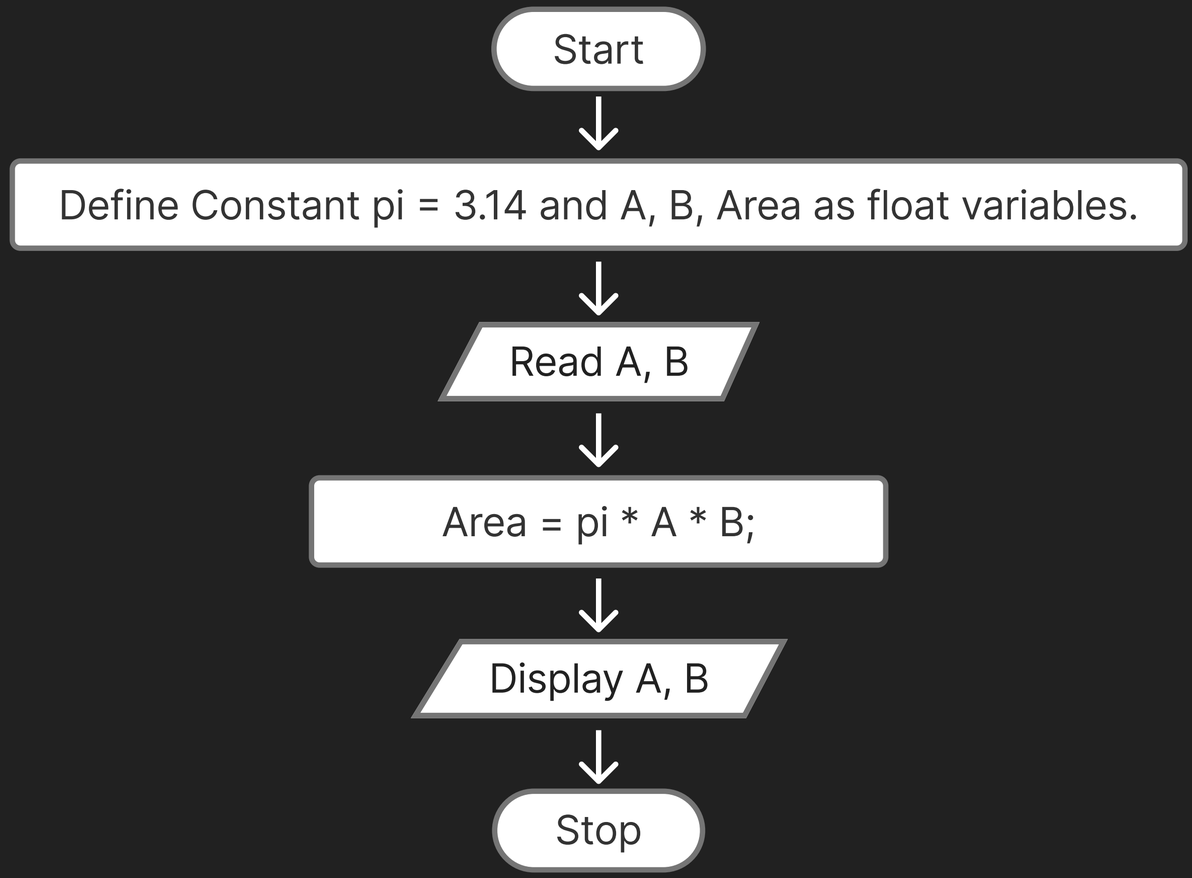
**Problem 1:** Given the two dimensions major and minor axis of an ellipse, what is the area?

let,
major axis = A = 2a
major axis = B = 2b

Algorithm solution :

- Start
- Define constant: pi = 3.14
- Define A, B and Area as float variables
- Read A, B
- Calculate the area of an ellipse: pi * A * B;
- Display area of an ellipse
- Stop

Flowchart solution:



# Setup & Configurations

## Setup:

- Download MinGW installer
  - Go to the SourceForge Website.
- Run the installation:
  - mingw32-gcc-g++ (for C and C++)
  - mingw32-base
- Add to your system PATH
  - C:\MinGW\bin to the PATH variable.
  - Install VS Code
  - Set Up VS Code for GCC

## Configurations:

- Install C/C++ Extension Pack and Code Runner from Vscode Extensions Marketplace
- Verify Your GCC Installation
  - Open Terminal
  - gcc --version
    - Copyright (C) 2021 Free Software Foundation, Inc.
- Configure Code Runner
  - Open the Command Palette (Ctrl+Shift+P or Cmd+Shift+P).
  - Search for Preferences: Open Settings (JSON) and select it.
  - { "code-runner.runInTerminal": true }
  - Run Code : Ctrl+Alt+N

## Coding :

```c
#include<stdio.h>

int main()
{
    printf(“Hello World!\n”);
    return 0;
}
```

## Coding :

```c
#include<stdio.h>
#define pi 3.14

int main(){
    float A,B,Area;
    printf("Enter A and B : \n");
    scanf("%f %f",&A, &B);
    Area = pi * A * B;
    printf("Area = %.3f",Area);
    return 0;
}
```

# Tasks

**Problem 2:** Draw a flowchart to find the factorial of a positive integer.

let,
positive number = Num
Factorial = Fact = 1
iterator = i = 1

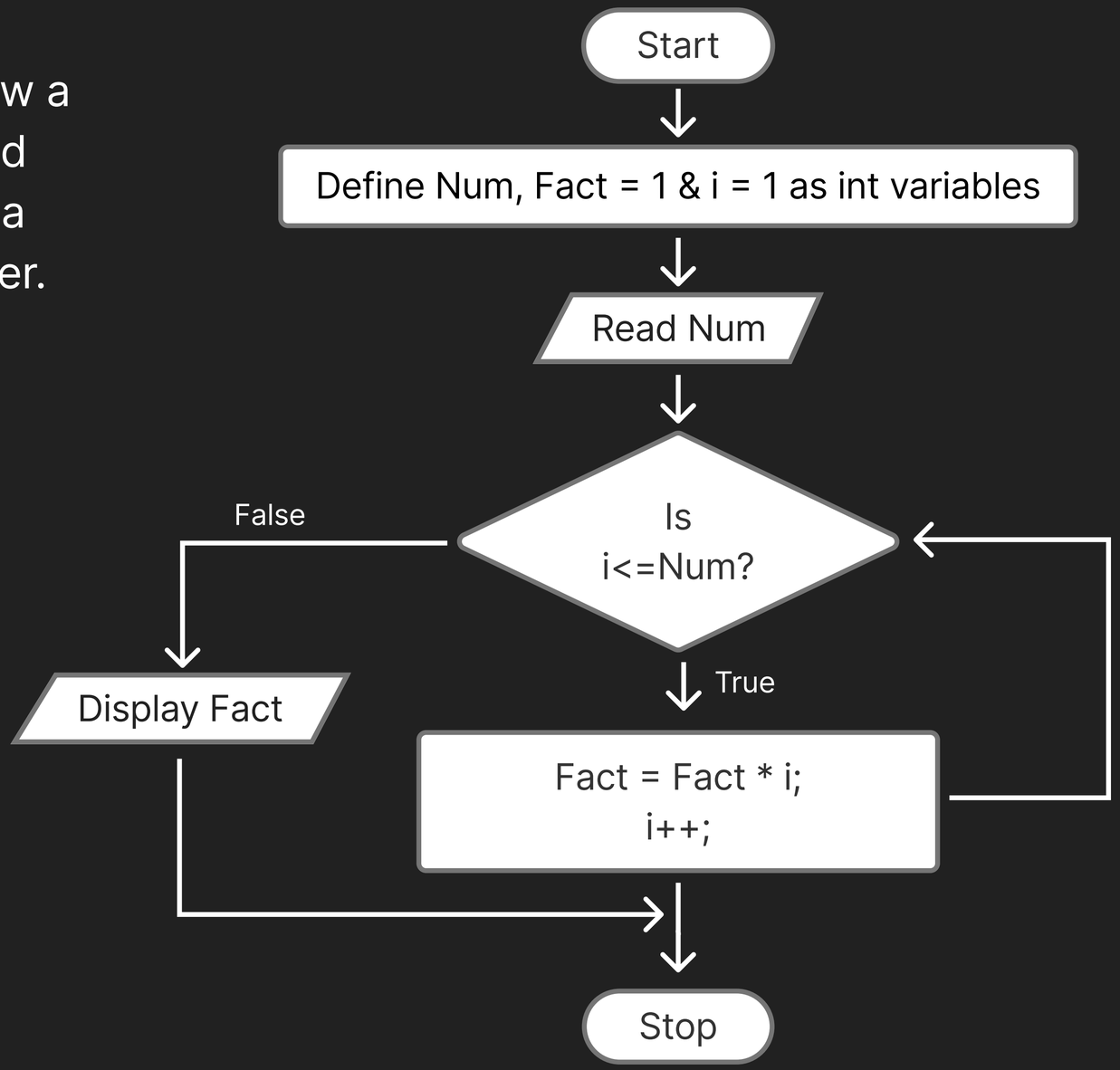
Algorithm solution :

- Start
- Define Num, Fact = 1 & i = 1 as int variables
- Read Num
- While(i<=Num)
  - Calculate the Factorial : fact = fact * i;
  - i++
- Display Fact
- Stop

# Tasks

**Problem 2:** Draw a flowchart to find the factorial of a positive integer.

Flowchart solution:



# Tasks

**Problem 3:** Draw a flowchart that calculates the sum of odd digits present in a number given by user.

**Problem 4:** Write an algorithm and draw a flowchart to generate the Fibonacci Series.

**Problem 5:** Write an algorithm and draw a flowchart to find greatest number among 3 numbers enterd by the users.

**Problem 6:** Write an algorithm and draw a flowchart to read as 5 digit number and check whether the number is palindrome or not.

**Problem 7:** Draw a flowchart to check whether a number is Armstrong or not.

## 2. Overview of C Programming

- Why C language (features & importance)
- Basic structure of a C program
- Header files (stdio.h, math.h)
- Tokens in C
  - Keywords
  - Identifiers
  - Constants
  - Variables
- Data types
- Type casting
  - Implicit
  - Explicit (basic idea)

# Headers Files (`stdio.h`)

## 1. printf Function

- Syntax:

```
int printf(const char *format, ...);
```

**Format Specifiers:**

Format specifiers begin with % and define the type of data to be printed:

- %d or %i: Integer.
- %f: Floating-point number.
- %c: Character.
- %s: String.
- %x or %X: Hexadecimal integer.
- %o: Octal integer.
- %%: Prints a literal % character.

# Headers Files (`stdio.h`)

### 2. scanf Function

- Syntax:

```
int scanf(const char *format, ...);
```

**Format Specifiers:**

Format specifiers are similar to those in printf, but the data is written into variables instead of being printed:

- %d: Reads an integer.
- %f: Reads a floating-point number.
- %c: Reads a single character.
- %s: Reads a string (stops at whitespace).

# Headers Files (`math.h`)

**Mathematical Functions:**

- sqrt(), pow(): Square root and power functions.
- sin(), cos(), tan(): Trigonometric functions.
- fabs(): Absolute value of a floating-point number.

**// Power (exponential) example :**

```
float base = 2.0;
float exponent = 3.0;
float powerResult = pow(base, exponent);
printf("%.2f raised to the power %.2f is: %.2f\n", base, exponent,
        powerResult);
```

# Some Other Functions of Headers Files

**stdio.h (Standard Input/Output):**

- printf(): Prints formatted output.
- scanf(): Reads formatted input.
- fopen(), fclose(): Opens and closes files.
- fgets(), fputs(): Reads and writes strings to files.

**stdlib.h (Standard Library):**

- malloc(), calloc(), free(): Dynamic memory allocation and deallocation.
- exit(): Exits the program.
- rand(), srand(): Generates random numbers.

# Some Other Functions of Headers Files

**string.h (String Manipulation):**

- strlen(): Calculates the length of a string.
- strcpy(), strcat(), strcmp(): String manipulation functions.

**math.h (Mathematical Functions):**

- sqrt(), pow(): Square root and power functions.
- sin(), cos(), tan(): Trigonometric functions.
- fabs(): Absolute value of a floating-point number.

**ctype.h (Character Functions):**

- isalpha(), isdigit(): Checks if a character is alphabetic or numeric.
- toupper(), tolower(): Converts characters to uppercase or lowercase.

# Some Other Functions of Headers Files

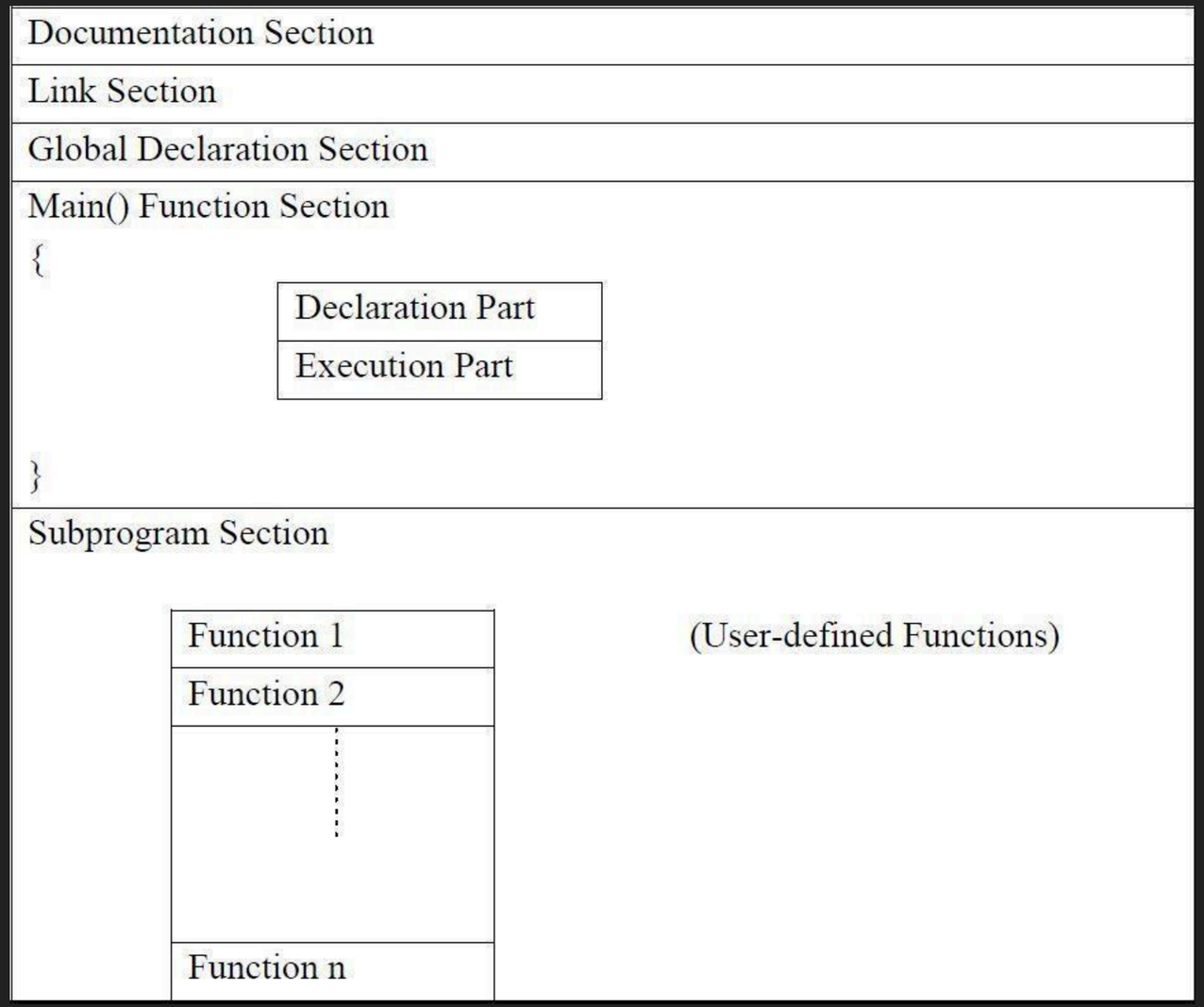
**time.h (Time and Date Functions):**

- time(): Returns the current time.
- asctime(), localtime(): Converts time to string representation.

**stdbool.h (Boolean Type):**

- bool: Boolean type (introduced in C99).
- true, false: Boolean values.

# Structure of C Programming



# Tokens in C

**Definition:**
Tokens are the smallest individual units of a C program that the compiler understands.

**Types of tokens in C include:**

- Character Set
- Keywords
- Identifiers
- Constants
- Variables

# Types of Tokens in C

## Character Set

**Definition:**

The character set is the set of valid characters that can be used in a C program.

**Syntax / Examples:**

```
A-Z, a-z          // letters
0-9               // digits
+ - * / % =       // special symbols
```

## Keywords

**Definition:**

Keywords are reserved words in C with predefined meanings and cannot be used as identifiers. There are 32 keywords.

**Syntax / Examples:**

```
int, float, if, else, while, return
```

# Types of Tokens in C

### Data Types and Type Modifiers

1. char
2. double
3. float
4. int
5. long
6. short
7. signed
8. unsigned
9. void
10. const
11. volatile
12. extern
13. register
14. static
15. auto

### Control Flow Statements

1. break
2. case
3. continue
4. default
5. do
6. else
7. for
8. goto
9. if
10. switch
11. while
12. return

### User-Defined Types and Special Keywords

1. enum
2. struct
3. typedef
4. union
5. sizeof

# Preprocessor directives :

**definition:**

Preprocessor directives give instructions to the compiler before the program is compiled.

**Types:**

1. #include
2. #define
    a. Used to define symbolic constants or macros.
3. #undef
    a. Used to remove a defined macro.
4. Conditional Compilation
    a. Used to compile code only if a condition is true.

```
#ifdef PI
printf("PI is defined");
#endif
```

**What to remember ?**

- No semicolon (;) at the end
- Handled before compilation
- Mainly used for headers and constants

# Comments:

## definition:

Comments are completely ignored by the C compilers.

**Types of Comments:**

1. Single line comment
2. Multi-line comment

1. **Single-line**

```
// declaring and initializing two integer variables
```

1. Multi-line comment

```
/*
Write Your Documentation here.....
*/
```

# Types of Tokens in C

## Identifiers

**Definition:**

Identifiers are names given to variables, functions, arrays, etc.

**Syntax:**

```
datatype identifier_name;
```

## Rules of Identifiers in C

1. First character must be an alphabet (or Underscore).
2. Must consist of only letters, digits or underscore.
3. Only first 31 characters are significant.
4. Cannot use a keyword.
5. Must not contain white space.

# Types of Tokens in C

### Valid Identifiers

```
total
_marks
sum1
studentName
```

### Constants

**Definition:**

Constants are fixed values that do not change during program execution.

**Syntax / Examples:**

```
10        // integer
3.14      // float
'A'       // character
"Hello"   // string
```

# Types of Tokens in C

## Variables

**Definition:**

Variables are named memory locations used to store data that can change.

**Syntax:**

```
datatype variable_name;
```

or

```
datatype variable_name = value;
```

Example:

```
int age = 20;
```

# Type Casting(Implicit and explicit)

**Definition:** Type casting is the process of converting a value from one data type to another.

## 1. Implicit Type Casting (Automatic)

- Done automatically by the compiler
- Occurs when different data types are used in an expression
- Smaller data type → larger data type (safe conversion)

Example:

```
int a = 10;
float b;

b = a;   // int converted to float automatically
```

# 2.7 Type Casting(Implicit and explicit)

## 2. Explicit Type Casting (Manual)

- Done by the programmer
- Used when you want controlled conversion
- Syntax:

Example:

```
int a = 5, b = 2;
float result;

result = (float)a / b;   // converts a to float
```

# 2.8 Data Types

**Definition:** Data types specify the type of data a variable can store and the amount of memory allocated to it.

**Primary Data Types:**