Unit 1

Introduction to C Programming

Terminology

- Problem
- Program
- Language

- Program:
  - A program is a set of sequenced instruction to cause a computer to perform particular operation or to solve the given problem
- Programming:
  - Process of developing such program is called programming
- Programmer:
  - Person who does programming is called programmer

- Programming Language:
  - The language that is used to develop the program is called programming language
  - “Programming languages are set of instruction used to write a program”
  - The language used to communicate with computer is known as programming language
  - program can be categorized based on how close to normal speech they are, and thus how far from the computer’s internal languages

Types of Programming Languages

- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate translation steps.
- Hundreds of such languages are in use today. These may be divided into three general types:
  - Machine languages
  - Assembly languages
  - High-level languages

Machine Languages

- The machine-level language is a language that consists of a set of instructions that are in the binary form 0 or 1.
- As we know that computers can understand only machine instructions, which are in binary digits, i.e., 0 and 1, so the instructions given to the computer can be only in binary codes.
- Creating a program in a machine-level language is a very difficult task as it is not easy for the programmers to write the program in machine instructions.
- It is error-prone as it is not easy to understand, and its maintenance is also very high.
- A machine-level language is not portable as each computer has its machine instructions, so if we write a program in one computer will no longer be valid in another computer.

Assembly Language

- The assembly language contains some human-readable commands such as mov, add, sub, etc.
- The problems which we were facing in machine-level language are reduced to some extent by using an extended form of machine-level language known as assembly language.
- Since assembly language instructions are written in English words like mov, add, sub, so it is easier to write and understand.
- As we know that computers can only understand the machine-level instructions, so we require a translator that converts the assembly code into machine code. The translator used for translating the code is known as an assembler.
- The assembly language code is not portable because the data is stored in computer registers, and the computer has to know the different sets of registers.

High-Level Language

- The high-level language is a programming language that allows a programmer to write the programs which are independent of a particular type of computer.
- The high-level languages are considered as high-level because they are closer to human languages than machine-level languages.
- When writing a program in a high-level language, then the whole attention needs to be paid to the logic of the problem.
- A compiler is required to translate a high-level language into a low-level language.

Low-level language	High-level language
It is a machine-friendly language, i.e., the computer understands the machine language, which is represented in 0 or 1.	It is a user-friendly language as this language is written in simple English words, which can be easily understood by humans.
The low-level language takes more time to execute.	It executes at a faster pace.
It requires the assembler to convert the assembly code into machine code.	It requires the compiler to convert the high-level language instructions into machine code.
The machine code cannot run on all machines, so it is not a portable language.	The high-level code can run all the platforms, so it is a portable language.
It is memory efficient.	It is less memory efficient.
Debugging and maintenance are not easier in a low-level language.	Debugging and maintenance are easier in a high-level language.

Programming Approach

- **Top-Down Model**
- It is a system design approach where design starts from the system as a whole.
- Complete System is then divided into smaller sub-applications with more details.
- Each part again goes through the top-down approach till the complete system is designed with all minute details.
- The basic idea in top-down approach is to break a complex algorithm or a problem into smaller segments called modules
- Top Down approach is also termed as breaking the bigger problem into smaller problems and solving them individually in recursive manner.

- **Bottom-Up Model**
- It is a system design approach where parts of the system are defined in details.
- Once these parts are designed and developed, then these parts or components are linked together to prepare a bigger component.
- This approach is repeated until the complete system is built.
- Advantage of Bottom-Up Model is in making decisions at very low level and to decide the re-usability of components.

key	Bottom-Up Model	Top-Down Model
Focus	In Bottom-Up Model, the focus is on identifying and resolving smallest problems and then integrating them together to solve the bigger problem.	In Top-down Model, the focus is on breaking the bigger problem into smaller one and then repeat the process with each problem.
Language	Bottom-Up Model is mainly used by object oriented programming languages like Java, C++ etc.	Top-Down Model is followed by structural programming languages like C, Fortran etc.
Redundancy	Bottom-Up model is better suited as it ensures minimum data redundancy and focus is on re-usability.	Top-down model has high ratio of redundancy as the size of project increases.
Interaction	Bottom-Up model have high interactivity between various modules.	Top-down model has low interactivity between various modules.
Issues	In Bottom-Up, some time it is difficult to identify overall functionality of system in initial stages.	In Top-Down, it may not be possible to break the problem into set of smaller problems.

ALGORITHM

- An algorithm is a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer
- Formally defined procedure for performing some calculation
- An effective procedure for solving a problem in a finite number of steps
- An algorithm gives the step by step description of how to arrive at a solution

Characteristics of algorithm

- Accuracy
- Clarity
- Not even a single instruction must be repeated infinitely
- After the algorithm gets terminated, the desired result must be obtained

Control Structures used in Algorithms

- **Sequence** means that each step of the algorithm is executed in the specified order.
- **Decision** statements are used when the execution of a process depends on the outcome of some condition.
- **Repetition** involves executing one or more steps for a number of times. Can be implemented using constructs such as *while*, *do-while*, and *for* loops.

Algorithm

- **Algorithm for finding sum of any two numbers**
- Step 1: START
- Step 2: Display “Enter two numbers”
- Step 3: Read first number as A and second number as B
- Step 4: set C=A+B
- Step 5: Display “C as sum of two numbers”
- Step 6: END

- **Algorithm for calculating the simple interest using formula SI=(P*T*R)/100**
- Step 1: START
- Step 2: Display “Enter principal, rate and time”
- Step 3: Read Principal as P, Time as T and rate as R
- Step 4: Calculate simple interest SI=(P*T*R)/100
- Step 5: Display SI as simple interest
- Step 6: END

- **Algorithm to determine whether a number is odd or even**
- Step 1: START
- Step 2: Display “Enter any number”
- Step 3: Read a number as Num from user
- Step 4: Calculate remainder using division of Num by 2 **(REM=Num%2)**
- Step 5: IF REM=0, THEN

  Display “The number is even”

  ELSE

  Display “The number is odd”
- Step 6: END

- **Algorithm to find the sum of first N naturals numbers**
- Step 1: START
- Step 2: Display “Enter a number “
- Step 3: Read number as N
- Step 4: SET i=1, sum=0
- Step 5: Repeat Step 6 while i<=N
- Step 6: SET sum=sum+I

  SET i=i+1
- Step 5: Display sum
- Step 6: END

- Write an algorithm to find largest of two numbers
- Write an algorithm to swap two numbers
- Write an algorithm to find the largest number among three numbers
- Write an algorithm to read N number from user and display the sum of all numbers between 1 and N.
- Write an algorithm to print the grade obtained by a student using the following rules:
  - Marks Grade
  - Above 80 Distinction
  - 60-79 first division
  - 50-59 second division
  - 40-49 third division
  - Less than 40 fail

FLOWCHART

- A graphical or symbolic representation of a process
- Used to design and document complex processes to help the viewers to visualize the logic of the process
- Provides a better understanding of the process and find flaws, bottlenecks, and other less obvious features within it
- Usually drawn in the early stages of formulating computer solutions
- When designing a flowchart, each step in the process is depicted by a different symbol and is associated with a short description

Symbols Used In Flowchart

Symbol	Purpose	Description
	Flow line	Indicates the flow of logic by connecting symbols.
	Terminal(Stop/Start)	Represents the start and the end of a flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for arithmetic operations and data-manipulations.

	Decision	Used for decision making between two or more alternatives.
	On-page Connector	Used to join different flowline
	Off-page Connector	Used to connect the flowchart portion on a different page.
	Predefined Process/Function	Represents a group of statements performing one processing task.

C Programming

- The C Language is developed for creating system applications that directly interact with the hardware devices such as drivers, kernels, etc.
- C programming is considered as the base for other programming languages, that is why it is known as mother language.
- It can be defined by the following ways:
  - Mother language
  - System programming language
  - Procedure-oriented programming language
  - Structured programming language
  - Mid-level programming language

- **C as a mother language**
  - Most of the compilers, JVMs, Kernels, etc. are written in C language, and most of the programming languages follow C syntax, for example, C++, Java, C#, etc.
- **C as a system programming language**
  - A system programming language is used to create system software.
  - C language is a system programming language because it can be used to do low-level programming (for example driver and kernel

- **C as a procedural language**
  - A procedural language **specifies a series of steps for the program to solve the problem**.
  - A procedural language breaks the program into functions, data structures, etc.
- **C as a structured programming language**
  - **Structure means to break a program into parts or blocks** so that it may be easy to understand.
  - In the C language, we break the program into parts using functions. It makes the program easier to understand and modify.

- **C as a mid-level programming language**
  - C is considered as a middle-level language because it **supports the feature of both low-level and high-level languages**.
  - A **Low-level language** is specific to one machine, i.e., machine dependent. It is machine dependent, fast to run. But it is not easy to understand.
  - A **High-Level language** is not specific to one machine, i.e., machine independent. It is easy to understand.

History of C

- **C programming language** was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.
- **Dennis Ritchie** is known as the **founder of the c language**.
- It was developed to overcome the problems of previous languages such as B, BCPL, etc.
- Initially, C language was developed to be used in **UNIX operating system**. It inherits many features of previous languages such as B and BCPL.

Features of C Language

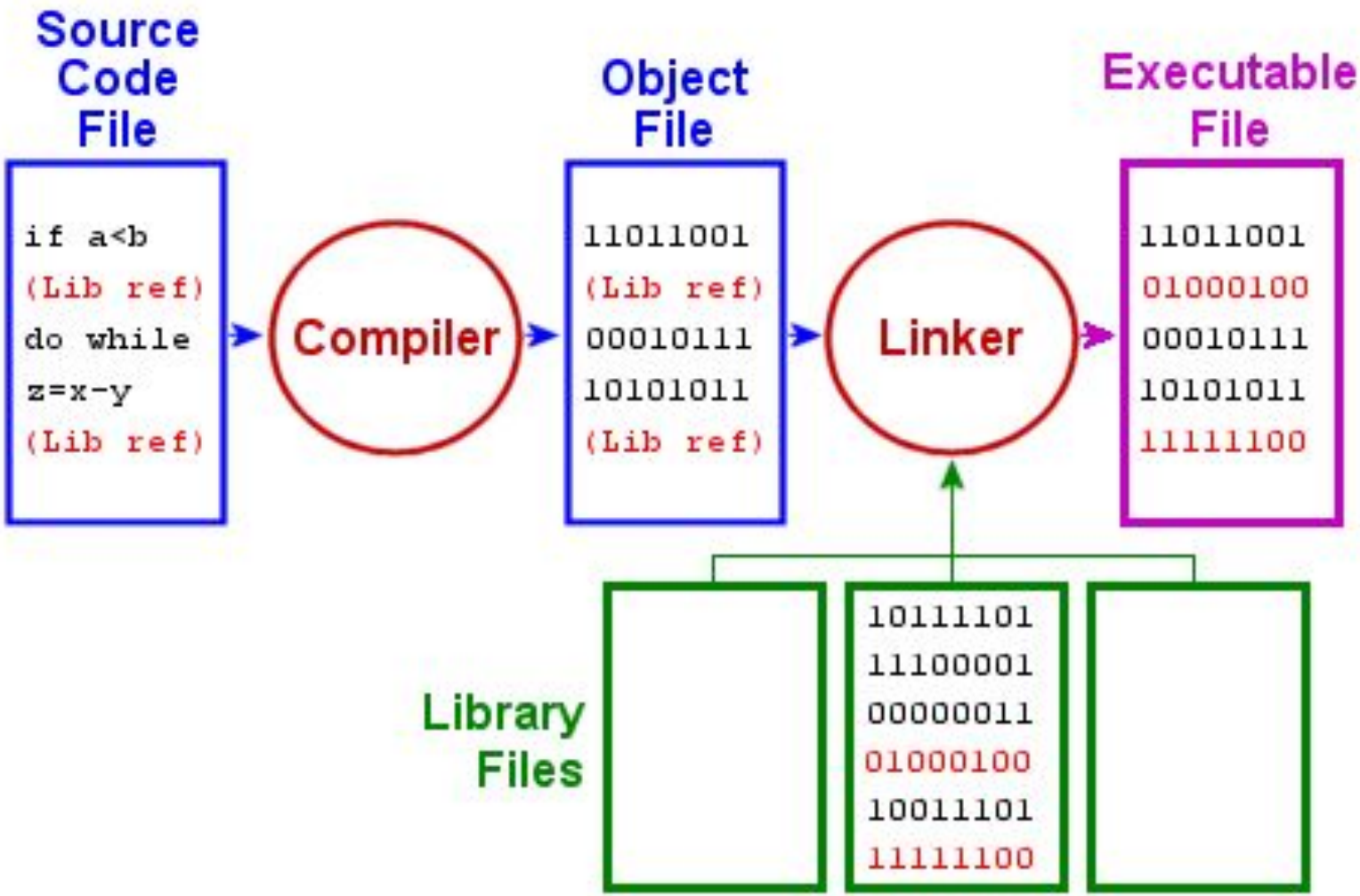
1. Small
   - C is a language of few words, containing only a handful of terms, called keywords, which serve as the base on which language's functionality is built
2. Simple
   - C is a simple language in the sense that it provides a structured approach (to break the problem into parts), the rich set of library functions, data types, etc.
3. Machine Independent or Portable
   - Unlike assembly language, c programs can be executed on different machines with some machine specific changes. Therefore, C is a machine independent language.

4. Mid-level programming language
   - supports the feature of both low-level and high-level languages.
5. Structured programming language
   - C is a structured programming language in the sense that we can break the program into parts using functions. So, it is easy to understand and modify.

First C Program

```
#include<stdio.h>
void main()
{
 printf("Hello World");
}
```

- **#include <stdio.h>** includes the standard input output library functions. The **printf()** function is defined in stdio.h .
- main() The main() function is the entry point of every program in c language.
- **printf()** function is used to print data on the console.


Source Code File
if a<b
(Lib ref)
do while
z=x-y
(Lib ref)
Compiler
Object File
11011001
(Lib ref)
00010111
10101011
(Lib ref)
Linker
Executable File
11011001
01000100
00010111
10101011
11111100
Library Files
10111101
11100001
00000011
01000100
10011101
11111100


Your C program
Compiling C program
Running program
OUTPUT