# INTRODUCTION TO PROBLEM SOLVING USING C

## Module I

Introduction to Computers– Introduction to Programming - How to develop a program, Algorithms, Flow-charts, Types of Programming Languages - Debugging, Types of errors - Techniques of Problem Solving – Problem solving aspects – Top- Down aspects –Structured programming concepts.

### Basic Concepts of Computer

Computer is an electronic device which is used to store the data, as per given instructions it gives results quickly and accurately.

- ❖ Data : Data is a raw material of information.
- ❖ Information : Proper collection of the data is called information.

### Characteristics of Computer

1. Speed: - As you know computer can work very fast. It takes only few seconds for calculations that we take hours to complete. You will be surprised to know that computer can perform millions (1,000,000) of instructions and even more per second. Therefore, we determine the speed of computer in terms of microsecond.

2. Accuracy: - The degree of accuracy of computer is very high and every calculation is performed with the same accuracy. The accuracy level is 7. determined on the basis of design of computer. The errors in computer are due to human and inaccurate data.

3. Diligence: - A computer is free from tiredness, lack of concentration, fatigue, etc. It can work for hours without creating any error. If millions of calculations are to be performed, a computer will perform every calculation with the same accuracy. Due to this capability it overpowers human being in routine type of work.

4. Versatility: - It means the capacity to perform completely different type of work. You may use your computer to prepare payroll slips. Next moment you may use it for inventory management or to prepare electric bills.

5. Power of Remembering: - Computer has the power of storing any amount of information or data. Any information can be stored and recalled as long as you require it, for any numbers of years. It depends entirely upon you how much data you want to store in a computer and when to lose or retrieve these data.

6. No IQ: - Computer is a dumb machine and it cannot do any work without instruction from the user. It performs the instructions at tremendous speed and with accuracy. It is you to decide what you want to do and in what sequence. So a computer cannot take its own decision as you can.

7. Storage: - The Computer has an in-built memory where it can store a large amount of data. You can also store data in secondary storage devices such as floppies, which can be kept outside your computer and can be carried to other computers.

## Basic Computer Organization

Basic Elements of Computer System Basic elements of a computer system are Mouse, Keyboard, monitor, memory, CPU, motherboard, Hard Disk, Speakers, Modem, power supply and processor. Computer Organization The block diagram of computer is shown in Fig. 1.1.

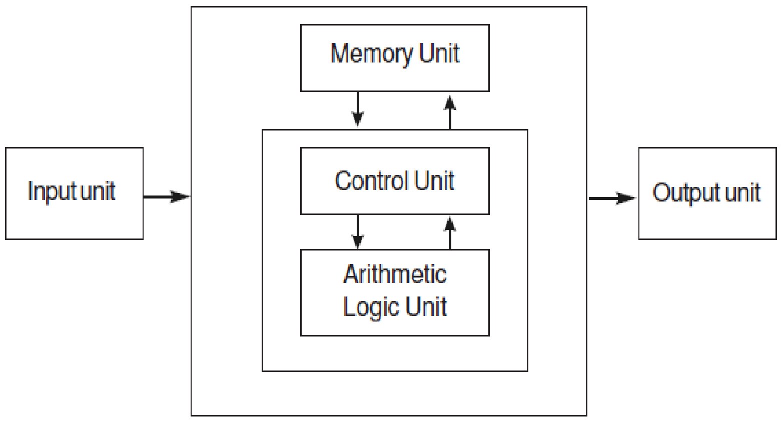


**Fig. 1.1** Block diagram of Computer Organisation

The computer performs basically five major operations of functions irrespective of their size and make. These are 1) it accepts data or instruction by way of input, 2) it stores data, 3) it can process data as required by the user, 4) it gives results in the form of output, and 5) it controls all operations inside a computer. We discuss below each of these operations.

### INPUT UNIT

Input unit accepts coded information from human operators through electromechanical devices such as the keyboard or from other computers over digital communication lines. The keyboard is wired so that whenever a key is pressed, the corresponding letter or digit is automatically translated into its corresponding code and sent directly to either the memory or the processor.

**RAM** is the primary memory of the computer. It can be used every time of computer process the instruction. It is an active memory of the system. It holds the data temporarily. It works high speed than other memory. However we are using RAM.

- Contains a large number of semiconductor cells each capable of storing one bit of information
- These cells are processed in group of fixed size called words containing „n" bits. The main memory is organized such that the contents of one word can be stored or retrieved in one basic operation.
- For accessing data, a distinct address is associated with each word location.
- Data and programs must be in the primary memory for execution.

## PROCESSOR UNIT

- The heart of the computer system is the Processor unit.
- It consists of Arithmetic and Logic Unit and Control Unit.

### Arithmetic and Logic Unit (ALU)

- Most computer operations (Arithmetical and logical) are executed in ALU of the processor.
- For example: Suppose two numbers (operands) located in the main memory are to be added. These operands are brought into arithmetic unit – actual addition is carried. The result is then stored in the memory or retained in the processor itself for immediate use.
- Note that all operands may not reside in the main memory. Processor contains a number of high speed storage elements called Registers, which may be used for temporary storage of frequently used operands. Each register can store one word of data.
- Access times to registers are 5 to 10 times faster than access time to memory.

### Control Unit

- The operations of all the units are coordinated by the control unity (act as the nerve centre that sends control signal to other units)
- Timing signal that governs the I/O transfers are generated by the Control Unit.
- Synchronization signals are also generated by the Control Unit

By selecting, interpreting and executing the program instructions the program instructions the control unit is able to maintain order and direct the operation of the entire system

### Storage Devices

Hard Disk, DVD, Pen Drive are the storage devices. It holds the data even switch off the computer. So it is called as permanent memory.

Hard Disk: Hard disk is used to store data permanently on computer.

**Optical Disks**: The Optical Disks are also called as the CD-ROM"s means Compact Disk Read Only Memory DVD means Digital Versatile Disk which is also used for Storing the data into the Disk and this is called as the Optical Disk.

CD – 700 MB DVD – 4.7 GB / 8.5 GB storage capacity.

## Planning the Computer Program

Two write a correct program, a programmer must write each and every instruct in the correct sequence

- ❖ Logic (instruction sequence) of a program can be very complex
- ❖ Hence, programs must be planned before they are written to ensure program instruction are:
  - ➢ Appropriate for the problem
  - ➢ In the correct sequence

**Algorithm**

Algorithm is a set of well defined instructions in sequence to solve the problem.

**Qualities of a good algorithm**

- Input and output should be defined precisely.
- Each steps in algorithm should be clear and unambiguous.
- Algorithm should be most effective among many different ways to solve a problem.
- An algorithm shouldn't have computer code. Instead, the algorithm should be written in such a way that, it can be used in similar programming languages.

Examples Of Algorithms In Programming

Write an algorithm to add two numbers entered by user.

```
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
```

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

**Advantages of algorithm**

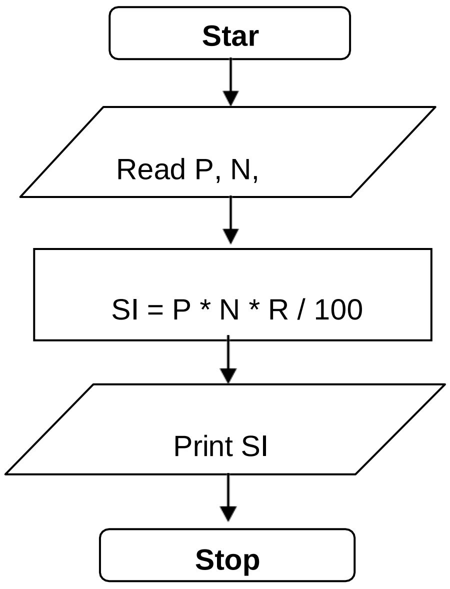
It is a step-wise representation of a solution to a given problem, which makes it easy to understand. An algorithm uses a definite procedure. It is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge. Every step in an algorithm has its own logical sequence so it is easy to debug. By using algorithm, the problem is broken down into smaller pieces or steps hence, it is easier for programmer to convert it into an actual program

## Flowchart

**Flowchart** is a pictorial representation of an algorithm. The following table shows flowchart symbol and it‟s usage.

| Symbol | Name | Function |
|---|---|---|
| → | Arrow | Control Flow |
| ▢ | Oval | Start / Stop |
| ▱ | Parallelogram | Input / Output |
| ▭ | Rectangle | Process / Calculation |
| ◇ | Diamond | Decision Making |
| ○ | Circle | Connector |

### Example Flowchart to calculate Simple Interest



**Types of Programming Languages**

There are two types of programming languages, which can be categorized into the following ways:

1. **Low level language**
    a) Machine language (1GL)
    b) Assembly language (2GL)
2. **High level language**
    a) Procedural-Oriented language (3GL)
    b) Problem-Oriented language (4GL)
    c) Natural language (5GL)

**1. Low level language**

This language is the most understandable language used by computer to perform its operations. It can be further categorized into:

**a) Machine Language (1GL)**

Machine language consists of strings of binary numbers (i.e. 0s and 1s) and it is the only one language, the processor directly understands. Machine language has an Merits of very fast execution speed and efficient use of primary memory.

Merits:

- ➢ It is directly understood by the processor so has faster execution time since the programs written in this language need not to be translated.
- ➢ It doesn’t need larger memory.

Demerits:

- ➢ It is very difficult to program using 1GL since all the instructions are to be represented by 0s and 1s.
- ➢ Use of this language makes programming time consuming.
- ➢ It is difficult to find error and to debug.
- ➢ It can be used by experts only.

**b) Assembly Language**

Assembly language is also known as low-level language because to design a program programmer requires detailed knowledge of hardware specification. This language uses mnemonics code (symbolic operation code like ‘ADD’ for addition) in place of 0s and 1s. The program is converted into machine code by assembler. The resulting program is reffered to as an object code.

Merits:

- ➢ It is makes programming easier than 1GL since it uses mnemonics code for programming. Eg: ADD for addition, SUB for subtraction, DIV for division, etc.
- ➢ It makes programming process faster.
- ➢ Error can be identified much easily compared to 1GL.
- ➢ It is easier to debug than machine language.

Demerits:

- ➢ Programs written in this language is not directly understandable by computer so translators should be used.
- ➢ It is hardware dependent language so programmers are forced to think in terms of computer’s architecture rather than to the problem being solved.
- ➢ Being machine dependent language, programs written in this language are very less or not portable.
- ➢ Programmers must know its mnemonics codes to perform any task.

**2. High level language**

Instructions of this language closely resembles to human language or English like words. It uses mathematical notations to perform the task. The high level language is easier to learn. It requires less time to write and is easier to maintain the errors. The high level language is converted into machine language by one of the two different languages translator programs; interpreter or compiler.

High level language can be further categorized as:

**a) Procedural-Oriented language (3GL)**

Procedural Programming is a methodology for modeling the problem being solved, by determining the steps and the order of those steps that must be followed in order to reach a desired outcome or specific program state. These languages are designed to express the logic and the procedure of a problem to be solved. It includes languages such as Pascal, COBOL, C, FORTAN, etc.

Merits:

- Because of their flexibility, procedural languages are able to solve a variety of problems.
- Programmer does not need to think in term of computer architecture which makes them focused on the problem.
- Programs written in this language are portable.

Demerits:

- It is easier but needs higher processor and larger memory.
- It needs to be translated therefore its execution time is more.

**b) Problem Procedural- Oriented language (4GL)**

It allows the users to specify what the output should be, without describing all the details of how the data should be manipulated to produce the result. This is one step ahead from 3GL. These are result oriented and include database query language.

Eg: Visual Basic, C#, PHP, etc.

The objectives of 4GL are to:

- Increase the speed of developing programs.
- Minimize user's effort to botain information from computer.
- Reduce errors while writing programs.

Merits:

Programmer need not to think about the procedure of the program. So, programming is much easier.

Demerits:

It is easier but needs higher processor and larger memory.
It needs to be translated therefore its execution time is more.

**c) Natural language (5GL)**

Natural language are stil in developing stage where we could write statrments that would look like normal sentences.

Merits:

- Easy to program.
- Since, the program uses normal sentences, they are easy to understand.
- The programs designed using 5GL will have artificial intelligence (AI).
- The programs would be much more interactive and interesting.

Demerits:

- It is slower than previous generation language as it should be completely translated into binary code which is a tedious task.
- Highly advanced and expensive electronic devices are required to run programs developed in 5GL.

Therefore, it is an expensive approach.

## What is Debugging?

Debugging is a methodical process of finding and reducing the number of bugs (or defects) in a computer program, thus making it behave as originally expected.

There are two main types of errors that need debugging:

Compile-time: These occur due to misuse of language constructs, such as syntax errors. Normally fairly easy to find by using compiler tools and warnings to fix reported problems.

Run-time: These are much harder to figure out, as they cause the program to generate incorrect output (or "crash") during execution. This lecture will examine how to methodically debug a run-time error in your C code.

## Types of program errors

We distinguish between the following types of errors:

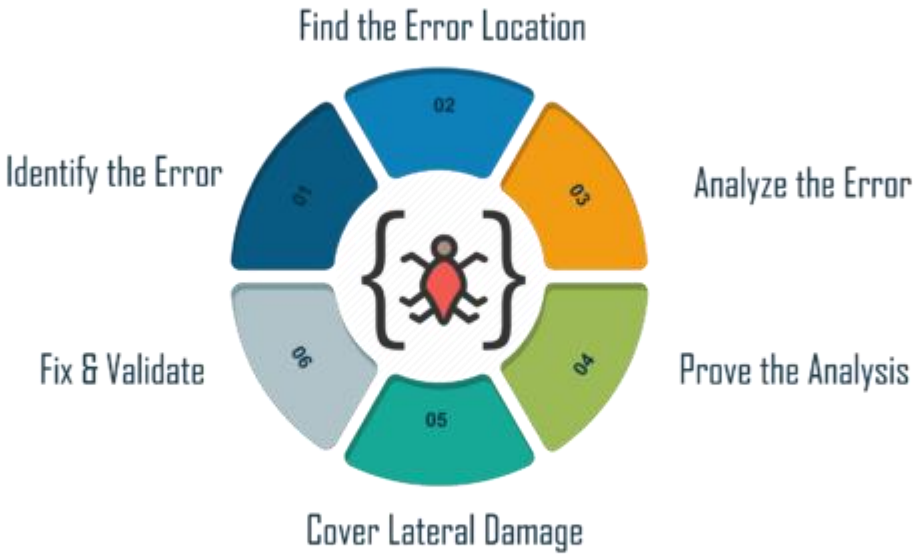
1. **Syntax errors**: errors due to the fact that the syntax of the language is not respected.
2. **Semantic errors**: errors due to an improper use of program statements.
3. **Logical errors**: errors due to the fact that the specification is not respected.

From the point of view of when errors are detected, we distinguish:

1. **Compile time errors**: syntax errors and static semantic errors indicated by the compiler.
2. **Runtime errors**: dynamic semantic errors, and logical errors, that cannot be detected by the compiler (debugging).

**Steps involved in Debugging**

The different steps involved in the process of debugging are:



1. Identify the Error: A bad identification of an error can lead to wasted developing time. It is usual that production errors reported by users are hard to interpret and sometimes the information we receive is misleading. It is import to identify the actual error.

2. Find the Error Location: After identifying the error correctly, you need to go through the code to find the exact spot where the error is located. In this stage, you need to focus on finding the error instead of understanding it.

3. Analyze the Error: In the third step, you need to use a bottom-up approach from the error location and analyze the code. This helps you in understanding the error. Analyzing a bug has two main goals, such as checking around the error for other errors to be found, and to make sure about the risks of entering any collateral damage in the fix.

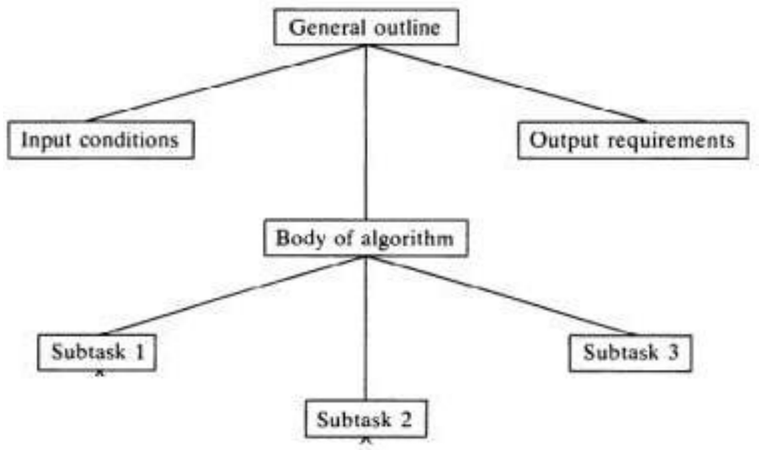
4. Prove the Analysis: Once you are done analyzing the original bug, you need to find a few more errors that may appear on the application. This step is about writing automated tests for these areas with the help of a test framework.

5. Cover Lateral Damage: In this stage, you need to create or gather all the unit tests for the code where you are going to make changes. Now, if you run these unit tests, they all should pass.

6. Fix & Validate: The final stage is the fix all the errors and run all the test scripts to check if they all pass.

## Top-down Design

A top-down approach (also known as stepwise design) is essentially the breaking down of a system to gain insight into the sub-systems that make it up. In a top-down approach an overview of the system is formulated, specifying but not detailing any first-level subsystems. Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements.



**Structured programming** is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program by making extensive use of the structured control flow constructs of selection (if/then/else) and repetition (while and for), block structures, and subroutines. Following the structured program theorem, all programs are seen as composed of control structures:

- "Sequence"; ordered statements or subroutines executed in sequence.
- "Selection"; one or a number of statements is executed depending on the state of the program. This is usually expressed with keywords such as if..then..else..endif.
- "Iteration"; a statement or block is executed until the program reaches a certain state, or operations have been applied to every element of a collection. This is usually expressed with keywords such as while, repeat, for or do..until. Often it is recommended that each loop should only have one entry point (and in the original structural programming, also only one exit point, and a few languages enforce this).