1.1 Problem Solving

Problem solving is a process of identifying a problem and finding the best solution for it. Problem solving is a skill that can be developed by following a well organized approach. We solve different problems every day. Every problem is different in its nature. Some problems are very difficult and require more attention to identify the solution. A problem may be solved in different ways. One solution may be faster, less expensive and more reliable than others. It is important to select the best suitable solution.

Different strategies, techniques and tools are used to solve a problem. Computers are used as a tool to solve complex problems by developing computer programs. Computer programs contain different instruction for computer. A programmer writes instructions and computer executes these instructions to solve a problem. A person can be good programmer if he has the skill of solving problems. Different problem-solving techniques are as follows:

- Program
- Algorithm
- Flowchart etc.

1.2 Program

A set of instructions that tells a computer what to do is called **program**. A computer works according to the given instructions in the program. Computer programs are written in programming languages. A person who develops a program is called **programmer**. The programmer develops programs to instruct the computer how to process data to convert into information. Programmer uses programming languages or tools to write programs.

1.2.1 Advantages of Computer Program

Different advantages of computer program are as follows:

- A computer program can solve many problems by giving instructions to computer.
- A computer program can be used to perform a task repeatedly and quickly.
- A program can process a large amount of data easily.
- It can display the results in different styles.
- The processing of a program is more efficient and less time consuming.
- Different types of programs are used in different fields to perform certain tasks.

1.3 Algorithms & Pseudo Code

An algorithm is a step-by-step procedure to solve a problem. The process of solving a problem becomes simpler and easier with help of algorithm. It is better to write algorithm before writing the actual computer program.

1.3.1 Properties of Algorithm

Following are some properties of an algorithm:

- The given problem should be broken down into simple and meaningful steps.
- The steps should be numbered sequentially.
- The steps should be descriptive and written in simple English.

Algorithms are written in a language that is similar to simple English called **pseudo code**. There is no standard to write pseudo code. It is used to specify program logic in an English like manner that is independent of any particular programming language.

Pseudo code simplifies program development by separating it into two main parts.

1. Logic Design

In this part, the logic of the program is designed. We specify different steps required to solve the problem and the sequence of these steps.

2. Coding

In this part, the algorithm is converted into a program. The steps of algorithm are translated into instructions of any programming language.

The use of pseudo code allows the programmer to focus on the planning of the program. After the planning is final, it can be written in any programming language.

Example 1

The following algorithm inputs two numbers, calculates sum and then displays the result on screen.

```
1. Start
2. Input A
3. Input B
4. Total = A + B
5. Display Total
6. Exit
```

Example 2

The following algorithm inputs radius from the user and calculates the area of circle. (Hint: Area = 3.14 * radius * radius)

```
1. Start
2. Input radius in r
3. area = 3.14 * r * r
4. Print area
5. End
```

Example 3

The following algorithm calculates the distance covered by a car moving at an average speed of V m/s in time T. It should input average speed v and time t. (Hint: s = vt where s is the distance traveled).

```
1. Start
2. Input average speed in v
3. Input time in t
4. s = v * t
5. Print s
6. End
```

Example 4

The following algorithm finds the sum of first fifty natural numbers.

```
1. Start
2. sum = 0
3. N = 0
4. Repeat Step 5 and 6 While (N <= 50)
5. sum = sum + N
```

ng things: ... because it helps a programmer to decide about the

- The facts and figures which are necessary for developing the program.
- The way in which the program will be designed.
- The language in which the program will be most suitable.
- What is the desired output and in which form it is needed, etc.

2. Designing the Algorithm

An algorithm is a sequence of steps that must be carried out before a programmer starts preparing his program. The programmer designs an algorithm to help visualize possible alternatives in a program.

3. Coding or Writing the Program

The process of writing a program is a very important step in program development. In this step, an algorithm is converted into program. The program consists of different steps given in the algorithm. A large number of programming languages are available to write programs. A programmer selects programming language according to the nature of problem.

4. Testing Program

A program must be tested in the process of program development. This process verifies the accuracy of a program. The program is tested by executing it again and again. Different values are given as input and output is checked. The program may not give required results if it contains any error. The errors must be detected and corrected if the output is not correct. All bugs in the program are detected and removed during program testing. It ensures that the program gives desired results and the problem is solved correctly.

5. Final Documentation

When the program is finalized, its documentation is prepared. Final documentation is provided to the user. It guides the user how to use the program in the most efficient way. Another purpose of documentation is to allow some other programmer to modify the code if necessary. Documentation should be done in each step during development of a program.

1.6 Programming Languages

A set of words, symbols and codes used to write programs is called **program language**. Different programming languages are available for writing different types of programs. Some languages are specially used for writing business programs, other are used for writing scientific programs etc.

There are two types of computer programming languages:

- Low-level languages
- High-level languages

1.6.1 Low Level Languages

These languages are near to computer hardware and far from human languages. Computer can understand these languages easily. Writing a program in low-level languages requires a deep knowledge of the internal structure of computer hardware. Two low-level languages are machine language and assembly language.

1. Machine Language

A type of language in which instructions are written in binary form is called **machine language**. It is the only language that is directly understood by the computer. It is the fundamental language of the computer.

Program written in machine language can be executed very fast by the computer. Programs written in machine language are machine-dependent. Every computer has its own machine language. Machine language is difficult to understand. Writing and modifying program in machine language takes a lot of time. Machine language is also known as **first generation language.**

2. Assembly Language

Assembly language is a low-level language. It is one step higher than machine language. In assembly language, symbols are used instead of binary code. These symbols are called **mnemonics**. For example **Sub** instruction is used to subtract two numbers.

Assembly language is also called **symbolic language.** Programs written in assembly language are easier to write and modify than machine language. Assembly language is mostly used for writing system software. Assembly language is also known as **second generation language.**

1.6.2 High Level Languages

A type of language that is close to human languages is called high level language. High-level languages are easy to understand. Instructions of these languages are written in English-like words such as **input** and **print** etc.

A program written in high-level language is easier to write and modify. High-level languages are further divided into following categories:

- Procedural Languages
- Object-Oriented Languages
- Non-Procedural Languages

1.6.2.1 Procedural Languages

Procedural languages are also known as **third-generation languages** or **3GL**. In these languages, program is a predefined set of instructions. Computer executes these instructions in the same sequence in which the instructions are written. Each instruction in this language tells the computer what to do and how to do. Some most popular procedural languages are:

1. FORTRAN

FORTRAN stands for **FORmula TRANslation.** It is mainly used for engineering application and scientific use.

2. BASIC

BASIC stands for **Beginner All Purpose Symbolic Instruction Code**. It was created in the late 1960. It was used mainly by students to use the computer for solving simple problems. It is easy to understand. It is widely used for education purpose.

3. COBOL

COBOL stands for **Common Business Oriented Language**. It is specially designed for business application. It was developed in early 1960s. The programs written in COBOL are lengthy but easy to read, write and maintain.

1.6.4 Natural Programming Languages

Natural programming languages are also known as **fifth generation languages (5GL)** or **intelligent languages.** Translator programs for these languages are very complex and require a large amount of computer resources. That is why most of these languages are still in experimental phase.

1.7 Types of Codes

There are two types of codes that are as follows:

1.7.1 Source Code

A program written in a high-level language is called **source code**. Source code is also called **source program**. Computer cannot understand the statements of high-level language. The source code cannot be executed by computer directly. It is converted into object code and then executed.

1.7.2 Object Code

A program in machine language is called **object code**. It is also called **object program** or **machine code**. Computer understands object code directly.

1.7.3 Difference between Source Code and Object Code

The main difference between source code and object code is as follows:

Source Code	Object Code
1. Source code is written in high-level or assembly language.	1. Object code is written in machine language through compilers.
2. Source code is easy to understand.	2. Object code is difficult to understand.
3. Source code is easy to modify.	3. Object code is difficult to modify.
4. Source code contains fewer statements than object code.	4. Object code contains more statements than source code.

1.8 Language Processor

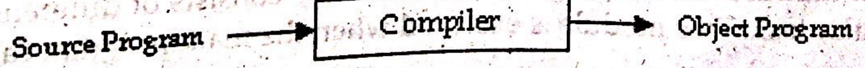
Computer understands only machine language. A program written in high-level or assembly language cannot be run on a computer directly. It must be converted into machine language before execution. **Language processor** or translator is a software that converts these programs into machine language. Every computer language has its own translators.

Different types of language processors are as follows:

1.8.1 Compiler

A **compiler** is a program that converts the instruction of a high level language into machine language as a whole. A program written in high-level language is called **source program**. Compiler converts source program into machine code known as **object program**.

The compiler checks each statement in the source program and generates machine instructions. Compiler also checks **syntax errors** in the program. A source program containing an error cannot be compiled.



A compiler can translate the programs of only that language for which it is written. For example C compiler can translate only those programs that are written in C language.

1.8.2 Interpreter

An **interpreter** is a program that converts one statement of a program at one time. It executes this statement before translating the next statement of the source program. If there is an error in the statements, the interpreter stops working and displays an errors message.

The advantage of interpreters over compilers is that an error is found immediately. So the programmer can correct errors during program development.

The disadvantage of interpreter is that it is not very efficient. The interpreter does not produce an object program. It must convert the program each time it is executed. Visual Basic uses interpreter.

1.8.3 Assembler

An **assembler** is translating program that translates the instruction of a assembly language into machine language.

$$\begin{tikzcd}\text{Assembly Language Program} \arrow[r] & \boxed{\text{Assembler}} \arrow[r] & \text{Object Program}\end{tikzcd}$$

1.8.4 Difference between Compiler and Interpreter

Following is a brief difference between compiler and interpreter:

Compiler	Interpreter
1. Compiler converts a program into machine code as a whole.	1. Interpreter converts a program into machine code statement by statement.
2. Compiler creates object code file.	2. Interpreter does not create object code file.
3. Compiler converts high-level program that can be executed many times.	3. Interpreter converts high-level program each time it is executed.
4. Program execution is fast.	4. Program execution is slow.
5. Compiler displays syntax errors after compiling the whole program.	5. Interpreter displays the syntax error on each statement of program.

Relationship of Object Program, Source Program and Compiler

A source program is written by a programmer in a programming language such as C. This program is not in a form that a computer can understand. A compiler translates that source program into the object program that the computer can understand and execute.

1.9 Programming Techniques

Computer programs are developed by using different programming techniques. Some important programming techniques are as follows:

1.9.1 Structured Programming

Structured programming is a programming technique in which a program is divided in small units called **modules** or **subprogram.** Each module consists of different instructions to perform a particular task. This module is executed when the main program calls it.

When main program calls a module, the control moves to the called module temporarily. The control moves back to the main program after executing the instructions of the module.

Structured programming is an easy and simple technique for writing programs. It makes programs easier to write, check, read and modify. This technique uses only three types of instructions. So the programs are not very complex. Following types of instructions are used in this technique:

- Sequential Structure
- Conditional / Selective Structure
- Iterative / Repetitive Structure

1. Sequential Structure

In **sequential structure,** the statements are executed in the same order in which they are specified in program. The control flows from one statement to other in a logical sequence. All statements are executed exactly once. It means that no statement is skipped and no statement is executed more than once

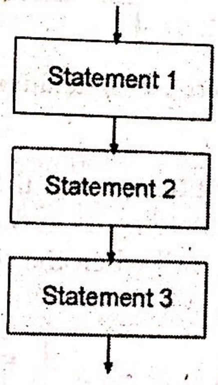


Figure 1.1: Execution of Sequential Statements

2. Conditional Structure

A **selection structure** selects a statement or set of statements to execute on the basis of a condition. In this structure, statement or set of statements is executed when a particular condition is **true** and ignored when the condition is **false** It is also known as **branching** or **conditional structure.**

Suppose a program displays **Pass** if the student gets 40 or more than 40 marks. It displays **Fail** when the marks are below 40. The program checks the marks before displaying the message.

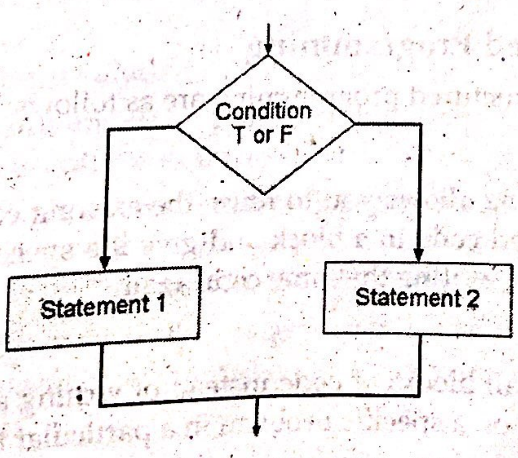


Figure 1.2: Executic Conditional Statements

3. Iterative Structure

A **repetition structure** executes a statement or set of statements repeatedly. It is also known as **iteration structure** or **loop**. Loops are basically used for two purposes:

- To execute a statement or number of statements for a specified number of times. For example, a user may display his name on screen for 10 times.
- To use a sequence of values. For example, a user may display a set of natural numbers from 1 to 10.

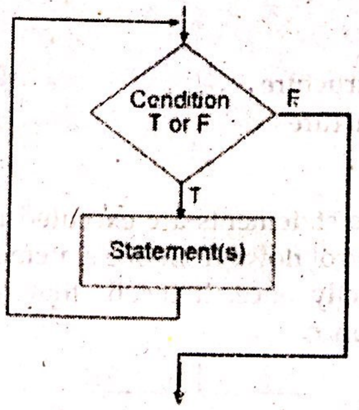


Figure 1.3: Execution of Iterative Statements

4. Function call

Function call is a type of statement that moves the control to another block of code. The control returns back after executing all statements in the block. The remaining statements are executed immediately after the function call when the control is returned.

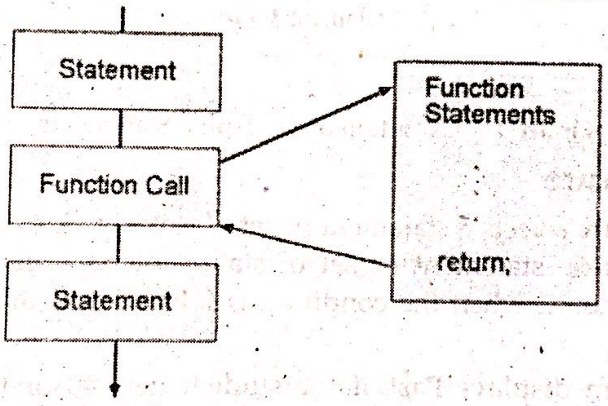


Figure 1.4: Execution of Function Call Statements

Advantages of Structured Programming

Some advantages of structured programming are as follows:

1. Reusability

Structured programming allows you to reuse the existing code as and when required. You can write frequently used code in a block and give it a specific name. This code can be used again and again without writing the same code again.

2. Easier to Code

It is easier to write small blocks of code instead of writing a long program as a whole. You can focus your attention on a specific problem in a particular block.