Covered Topics Under UNIT-1 of "PPS-PROGRAMMING FOR PROBLEM SOLVING (BCS101 / BCS201)"

shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com

Class Notes

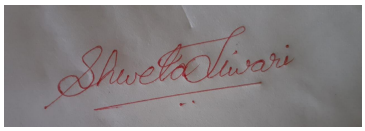
Published Date: November, 2022

PPS: UNIT-1

Introduction to Components of a Computer System

FALL SEMESTER, YEAR (I/II sem, 1st yr)

FALL SESSION (2022-23)
(PPS)
MS. SHWETA TIWARI
Published Date: November, 2022



shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com

TOPIC On : UNIT-1: Algorithm

—

By SHWETA TIWARI

Under On: Introduction to Components of a Computer System

PREPARED FOR
Engineering Students
All Engineering College

PREPARED BY
SHWETA TIWARI

TOPIC On : UNIT-1: Algorithm

Computer Languages:

To write a program for a computer, we must use a **computer language.** Over the years computer languages have evolved from machine languages to natural languages.

1940's Machine level Languages

1950's Symbolic Languages

1960's High-Level Languages

Machine Languages

In the earliest days of computers, the only programming languages available were machine languages. Each computer has its own machine language, which is made of streams of 0's and 1's.

Instructions in machine language must be in streams of 0's and 1's because the internal circuits of a computer are made of switches, transistors and other electronic devices that can be in one of two states: off or on. The off state is represented by 0 , the on state is represented by 1.

The only language understood by computer hardware is machine language.

Symbolic Languages:

In the early 1950's Admiral Grace Hopper, A mathematician and naval officer developed the concept of a special computer program that would convert programs into machine language.

The early programming languages simply mirror to the machine languages using symbols of mnemonics to represent the various machine language instructions because they used symbols, these languages were known as symbolic languages.

Computer does not understand symbolic language; it must be translated to the machine language. A special program called assembler translates symbolic code into machine language. Because symbolic languages had to be assembled into machine language they soon became known as assembly languages.

Symbolic language uses symbols or mnemonics to represent the various ,machine language instructions.

High Level Languages:

Symbolic languages greatly improved programming efficiency; they still required programmers to concentrate on the hardware that they were using. Working with symbolic languages was also very tedious because each machine instruction has to be individually coded. The desire to improve programmer efficiency and to change the focus from the computer to the problem being solved led to the development of high-level language.

High level languages are portable to many different computers, allowing the programmer to concentrate on the application problem at hand rather than the intricacies of the computer. High-level languages are designed to relieve the programmer from the details of the assembly language. High level languages share one thing with symbolic languages, They must be converted into machine language. The process of converting them is known as compilation.

The first widely used high-level language, FORTRAN (FORmula TRANslation)was created by John Backus and an IBM team in 1957;it is still widely used today in scientific and engineering applications. After FORTRAN was COBOL(Common Business-Oriented Language). Admiral Hopper played a key role in the development of the COBOL Business language.

C is a high-level language used for system software and new application code.

What is an Algorithm?

*The word **Algorithm** means " A set of finite rules or instructions to be followed in calculations or other problem-solving operations " Or " A procedure for solving a mathematical problem in a finite number of steps that frequently involves recursive operations".*

What is an algorithm?

Algorithm is a *set of steps to complete a task.*

For example,

Task: to make a cup of tea.

Algorithm:

- add water and milk to the kettle,
- boil it, add tea leaves,
- Add sugar, and then serve it in a cup.

What is a Computer *algorithm*?

"a set of steps to accomplish or complete a task that is described precisely enough that a computer can run it".

Described precisely: very difficult for a machine to know how much water, milk to be added etc. in the above tea making algorithm.

These algorithms run on computers or computational devices.For Example, GPS in our smartphones, Google hangouts.

GPS uses the shortest *path algorithm.* Online shopping uses cryptography which uses the RSA algorithm.

ALGORITHM:

Algorithms were developed by an Arab mathematician. It is a chalked out step-by-step approach to solve a given problem. It is represented in an English-like language and has some mathematical symbols like ->, >, <, = etc. To solve a given problem or to write a program you approach towards the solution of the problem in a systematic, disciplined, non-ad hoc, step-by-step way is called Algorithmic approach. Algorithm is a penned strategy(to write) to find a solution.

Algorithms:

Structure and Properties of Algorithm:

An algorithm has the following structure

1. Input Step
2. Assignment Step
3. Decision Step
4. Repetitive Step
5. Output Step

An algorithm is endowed with the following properties:

1. **Finiteness:** An algorithm must terminate after a finite number of steps.
2. **Definiteness:** The steps of the algorithm must be precisely defined or unambiguously specified.
3. **Generality:** An algorithm must be generic enough to solve all problems of a particular class.
4. **Effectiveness:** the operations of the algorithm must be basic enough to be put down on pencil and paper. They should not be too complex to warrant writing another algorithm for the operation.
5. **Input-Output:** The algorithm must have certain initial and precise inputs, and outputs that may be generated both at its intermediate and final steps.

Example: Algorithm/pseudo code to add two numbers

Step 1: Start

Step 2:Read the two numbers in to a,b

Step 3: c=a+b

Step 4: write/print c

Step 5: Stop.

Different Approaches to Design an Algorithm:

An algorithm does not enforce a language or mode for its expression but only demands adherence to its properties.

Practical Algorithm Design Issues:

1. **To save time (Time Complexity):** A program that runs faster is a better program.

2. **To save space (Space Complexity):** A program that saves space over a competing program is considerably desirable.

Efficiency of Algorithms:

The performances of algorithms can be measured on the scales of **time** and **space**. The performance of a program is the amount of computer memory and time needed to run a program. We use two approaches to determine the performance of a program. One is analytical and the other is experimental. In performance analysis we use analytical methods, while in performance measurement we conduct experiments.

Time Complexity: The time complexity of an algorithm or a program is a function of the running time of the algorithm or a program. In other words, it is the amount of computer time it needs to run to completion.

Space Complexity: The space complexity of an algorithm or program is a function of the space needed by the algorithm or program to run to completion.

The time complexity of an algorithm can be computed either by an **empirical** or **theoretical** approach. The **empirical** or **posteriori testing** approach calls for implementing the complete algorithms and executing them on a computer for various instances of the problem. The time taken by the execution of the programs for various instances of the problem are noted and compared. The algorithm whose implementation yields the least time is considered as the best among the candidate algorithmic solutions.

Analyzing Algorithms

Suppose M is an algorithm, and suppose n is the size of the input data. Clearly the complexity f(n) of M increases as n increases. It is usually the rate of increase of f(n) with some standard functions. The most common computing times are

$O(1), O(\log_2 n), O(n), O(n \log_2 n), O(n^2), O(n^3), O(2^n)$

Example: