INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

Introduction to Computer Science and Engineering

CSN-101

Problem-solving using computers, Flowing charting technique and writing Algorithm

Prof. Partha Pratim Roy
Associate professor, CSE Department
Indian Institute of Technology Roorkee

Introduction

- Today, computers are all around us. We use them for doing various tasks in a faster and more accurate manner. For example, using a computer or smartphone, we can book train tickets online.
- Number of problems in our daily life.
- Suppose we have to calculate simple interest.
- Suppose we have to prepare a mark sheet .
- A computer is DUMB machine.
- A computer cannot do anything alone without software i.e. Program

Software

- **Software**, instructions that tell a computer what to do. The software comprises the entire set of programs, procedures, and routines associated with the operation of a computer system.
- A set of instructions that directs a computer's hardware to perform a task is called a program, or software program.
- The two main types of software are system software and application software.
- System software controls a computer's internal functioning, chiefly through an operating system, and also controls such peripherals as monitors, printers, and storage devices.

Stages while solving a problem using computer

- Problem Analysis
- Algorithm Development
- Flowcharting
- Coding
- Compilation and Execution
- Debugging and Testing
- Documentation

Problem Analysis

- Process of becoming familiar with the problem.
- We need to analyze and understand it well before solving it.
- The user's requirements cannot be fulfilled without a clear understanding of his / her problem in depth.
- Inadequate identification of problem may cause program less useful and insufficient.
- Example Banking Solution, Hospital medical study

Algorithm Development

- Algorithm development is **the act of designing the steps that solve a particular problem for a computer or any other device to follow not excluding human beings** but in this case computers only and computer-like devices.
- Algorithm development steps. Steps in the development of Algorithms.
- Step by step description of the method to solve the problem.
- Effective procedure for solving the problem with an infinite number of steps.
- Developing an algorithm in a step of program design.

The algorithm to add two numbers.

- Declare variable (Two variables to store the number input by the user and one variable is used to store the output).
- Take the input of two numbers.
- Apply the formula for addition.
- Add two numbers.
- Store the result in a variable.
- Print the result.

Algorithm to find greatest number of three given numbers

- Ask the user to enter three integer values.
- Read the three integer values in num1, num2, and num3 (integer variables).
- Check if num1 is greater than num2.
- If true, then check if num1 is greater than num3
- If false, then check if num2 is greater than num3.

Algorithm to find area of rectangle

1. Define the width of the rectangle.
2. Define the Height of the rectangle.
3. Define Area of the rectangle.
4. Calculate the area of the rectangle by multiplying the width and height of the rectangle.
5. Assign the area of the rectangle to the area variable.
6. print the area of the rectangle.

Features of Algorithm

- **Unambiguous** − The algorithm should be unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.
- **Input** − An algorithm should have 0 or more well-defined inputs.
- **Output** − An algorithm should have 1 or more well-defined outputs and should match the desired output.
- **Finiteness** − Algorithms must terminate after a finite number of steps.
- **Feasibility** − Should be feasible with the available resources.
- **Independent** − An algorithm should have step-by-step directions, which should be independent of any programming code.

Flowcharting

- Graphical representation of an algorithm using standard symbols.
- Includes a set of various standard shaped boxes that are interconnected by flow lines.
- Flow lines have arrows directed of flow.
- Activities are written within boxes in English.
- Communicates between programmers and business persons.

Advantages of Flowcharts

- Communication
- Quickly provide logic , ideas and descriptions of algorithms.
- Effective analysis
- Clear overview entire problem.
- Proper documentation
- Helps us to understand its logic in future.
- Efficient coding
- More ease with comprehension flowchart as a guide
- Easy in debugging and program maintenance
- Debugging and maintenance of operating program

Flowcharting

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

Difference

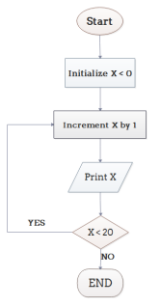
Algorithm	Flowchart
It is a procedure for solving problems.	It is a graphic representation of a process.
The process is shown in step-by-step instruction.	The process is shown in block-by-block information diagram.
It is complex and difficult to understand.	It is intuitive and easy to understand.
It is convenient to debug errors.	It is hard to debug errors.
The solution is showcased in natural language.	The solution is showcased in pictorial format.

Example 1: Print 1 to 20:

Algorithm:

- Step 1: Initialize X as 0,
- Step 2: Increment X by 1,
- Step 3: Print X,
- Step 4: If X is less than 20 then go back to step 2.

Flowchart

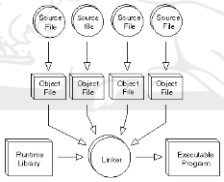


Coding

- The process of transforming the program logic design into computer language format.
- An act of transforming operations in each box of the flowchart in terms of the statement of a program.
- The code written using programming language is also known as source code.
- Coding isn't the only task to be done to solve a problem using computer.

Compilation

- Process of changing high-level language into machine-level language.
  It is by special software, **COMPILER**
- The compilation process tests the program whether it contains syntax errors or not.
- If syntax errors are present, the compiler can not compile the code.



Execution

- Execution in computer and software engineering is the **process by which a computer or virtual machine reads and acts on the instructions of a computer program**. ...
- Programs for a computer may be executed in a batch process without human interaction or a user may type commands in an interactive session of an interpreter.
- Once the compilation is completed then the program is linked with other object programs needed for execution, there by resulting in a binary program
- And then the program is located in the memory for the purpose of execution and finally it is executed.

Debugging and Testing

- Testing means verifying correct behavior. Testing can be done at all stages of module development: requirements analysis, interface design, algorithm design, implementation, and integration with other modules
- Debugging is a cyclic activity involving execution testing and code correction. The testing that is done during debugging has a different aim than final module testing. Final module testing aims to demonstrate correctness, whereas testing during debugging is primarily aimed at locating errors. This difference has a significant effect on the choice of testing strategies.

Debugging and Testing

- Testing ensures that program performs correctly the required task.
- Verification ensures that program does what the programmer intends to do.
- Validation ensures that program produces the correct results for a set of test data
- Test data are supplied to the program and the output is observed
- Expected output= Error free

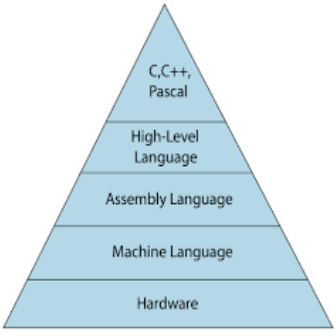
Program Documentation

- Any written text, illustrations, or video that describes software or program to its users is called a **program or software document**. Users can be anyone from a programmer, system analyst, and administrator to end-user.
- At various stages of development, multiple documents may be created for different users. Software **documentation** is a critical process in the overall software development process.
- In modular programming documentation becomes even more important because different modules of the software are developed by different teams.

Two types of documentations

- Programmers' Documentation (Technical documentation)
  - Maintain, redesign and upgrade
  - Logic, DFD, E-R, Algorithmand flowchart
- User documentation (user manual)
  - Support to the user of the program
  - Instructions for installation

Types or levels of programming language



Low level language

- A low-level language, often known as a computer's native language, is a sort of programming language.
- It is very close to writing actual machine instructions, and it deals with a computer's hardware components and constraints. It works to control a computer's operational semantics and provides little or no abstraction of programming ideas
- In contrast to high-level language that used for developing software, low-level code is not human-readable, and it is often cryptic.
- Assembly language and machine language are two examples of low-level programming languages.

Machine level language

- Machine language is the lowest and most elementary level of programming language and was the first type of programming language to be developed.
- Machine language is basically the only language that a computer can understand and it is usually written in hex.
- In fact, a manufacturer designs a computer to obey just one language, its machine code, which is represented inside the computer by a string of binary digits (bits) 0 and 1.
- The symbol 0 stands for the absence of an electric pulse and the 1 stands for the presence of an electric pulse. Since a computer is capable of recognizing electric signals, it understands machine language.

Machine Level Language

Advantage	Disadvantage
• Computer directly understands machine instructions	• Difficult to use
• Takes less execution time	• Machine dependent
• Directly starts executing	• Difficult to Debug and modify
• Machine language makes fast and efficient use of the computer.	• All operation codes have to be remembered

Assembly Language

- An assembly language is a type of low-level programming language that is intended to communicate directly with a computer's hardware.
- Unlike machine language, which consists of binary and hexadecimal characters, assembly languages are designed to be readable by humans.
- Low-level programming languages such as assembly language are a necessary bridge between the underlying hardware of a computer and the higher-level programming languages—such as Python or JavaScript—in which modern software programs are written.

High Level Language

- User-friendly, Similar to natural languages
- Platform independent
- Easy to write or remember
- Easy to learn and work
- While execution: translated into assembly language than to machine language.
- Slow in execution but is efficient for developing programs.
- Ex: C, C++, Python, Java, etc.

Difference between

High level language	Low level language
It is programmer friendly language.	It is a machine friendly language.
High level language is less memory efficient.	Low level language is high memory efficient.
It is easy to understand.	It is tough to understand.
It is simple to debug.	It is complex to debug comparatively.
It is simple to maintain.	It is complex to maintain comparatively.
It is portable.	It is non-portable.
It can run on any platform.	It is machine-dependent.
It needs compiler or interpreter for translation.	It needs assembler for translation.

Compiler

- A high-level source program must be translated into a forming machine that can understand. This is done by a software called the compiler.
- Source code => Machine language code(Object code)
- During the process of translation, the compiler reads the source programs statement-wise and checks for syntax errors.
- In case of any error, the computer generates a message about the error. • Ex: C, C++, Java, FORTRAN, Pascal, etc.

Interpreter

- Like a compiler, it is also a translator which translates high-level to machine-level language.
- Translates and executes the program line by line.
- Each line is checked for syntax error and then converted to the equivalent machine code.
- Ex. QBASIC, PERL, PHP, ASP, PYTHON, RUBY

Difference between compiler and interpreter

Interpreter	Compiler
Interpreter translates just one statement of the program at a time into machine code.	Compiler scans the entire program and translates the whole of it into machine code at once.
An interpreter takes very less time to analyze the source code. However, the overall time to execute the process is much slower.	A compiler takes a lot of time to analyze the source code. However, the overall time taken to execute the process is much faster.
An interpreter does not generate an intermediary code. Hence, an interpreter is highly efficient in terms of its memory.	A compiler always generates an intermediary object code. It will need further linking. Hence more memory is needed.
Keeps translating the program continuously till the first error is confronted. If any error is spotted, it stops working and hence debugging becomes easy.	A compiler generates the error message only after it scans the complete program and hence debugging is relatively harder while working with a compiler.

Python – programming language

- Python is an easy-to-learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming.
- Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.
- Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary.

Python –programming language

1. Python is currently the most widely used multi-purpose, high-level programming language.
2. Python allows programming in Object-Oriented and Procedural paradigms.
3. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and the indentation requirement of the language makes them readable all the time.
4. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

Reason for Increasing popularity

1. Emphasis on **code readability, shorter codes**, ease of writing
2. Programmers can express logical concepts in **fewer lines** of code in comparison to languages such as C++ or Java.
3. Python supports **multiple** programming paradigms, like object-oriented, imperative and functional programming or procedural.
4. There exists inbuilt functions for almost all of the frequently used concepts.
5. Philosophy is "Simplicity is the best".

Thanks…