AMBITION GURU

Computer Concept and Programming

I semester, BSc. CSIT

Syllabus

AMBITION GURU

Unit	Contents	Hours
1.	**Computer Fundamental and Programming Methodology**	**3**
2.	Overview of C Language	4
3.	Control Structures	6
4.	Arrays and Strings	6
5.	Functions	8
6.	Pointers	5
7.	Structures and Unions	5
8.	File Handling in C	5
9.	Introduction to Graphics	3

Practical Works

Credit hours : 3

AMBITION GURU

Practical Works

Laboratory Work:

The Laboratory work must cover programming part of all the topics covered in the course. The instructor can conduct the programming as required. Some important contents that should be included in lab exercises are as follows:

1. Create algorithms and flowchart for solving the problem.
2. Create, compile, debug, run and test simple C programs
3. Create decision making programs using control statements like; if, if..else, if..else ladder, nested if, and switch cases.
4. Create programs using loops (for, while, do while, nested loops) and realize the differences between entry controlled and exit controlled loops.
5. Create, manipulate arrays and matrices (single and multi-dimensional), work with pointers, dynamically allocate/de-allocate storage space during runtime, manipulate strings (character arrays) using various string handling functions.
6. Create user-defined functions with/without parameters or return type, create recursive functions, use function call by value and call by address, work with automatic, global and static variables.
7. Create programs that addresses pointer arithmetic, pointers and arrays, pointer and character strings, pointers and functions, pointer and structure, and dynamic memory allocation.
8. Create and use simple structures, array of structures, nested structure. Passing structure and array of structure to function, concept of pointer to structure
9. Create files that address random access and input/output operations in file, create files to keep records and manipulation of records etc.
10. Create graphics program that address some basic functions of *graphics.h* header file, e.g. line(), arc(), circle(), rectangle() etc.

AMBITION GURU

Resources

1. Some chapters in pdf form.
2. Question Collection & Assignments (50-80 Questions)

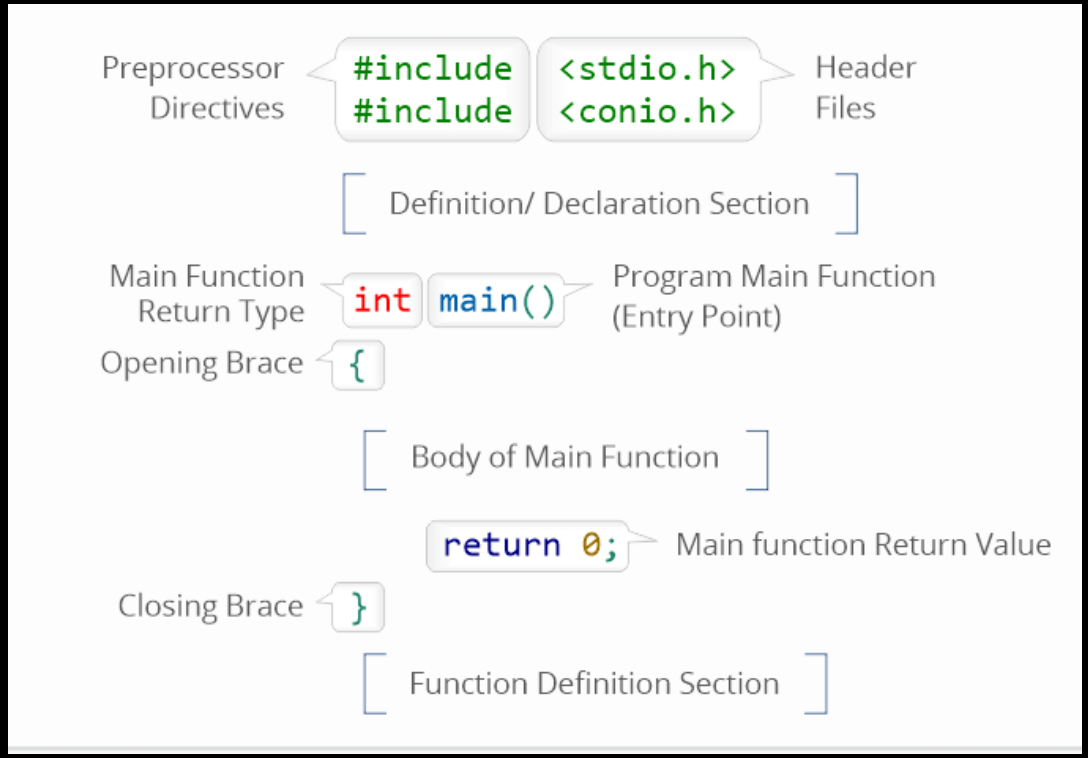
Resources

Textbooks/Reference Books

- Balagurusamy, E. (2019), **Programming in ANSI C (8th ed.),** New Delhi, India: Tata McGraw-Hill.
- Bryon S Gottfried (2018), **Programming with C (4th ed.),** Mc Graw Hill India.
- PK Sinha, **Computer Fundamentals (8th ed.)**, BPB Publications, India
- Dennis M. Ritchie, Brian W. Kernighan, **The C Programming Language (2nd ed.)**, Prentice Hall.
- Kanetkar, Y. P. (2022), **Let us C (15th Ed.)**, New Delhi, BPB Publication

AMBITION GURU

Format for C program



AMBITION GURU

Unit 1

(3 hrs.)

Introduction of Programming Concept

- Introduction, Components of PC, Computer Architecture, Memory Types, Memory Hierarchy, Computer Peripherals, Input and Output Devices, Basic of Computer Networking, Computer Program, Steps for Program Development
- Problem Solving Tools: Algorithmic Thinking and Flowchart, Pseudocode, Program Control Structures, Program Methodology, Program Models

AMBITION GURU

Unit 1: Part 1

Introduction, Components of PC, Computer Architecture, Memory Types, Memory Hierarchy, Computer Peripherals, Input and Output Devices, Basic of Computer Networking, Computer Program, Steps for Program Development

Introduction to computers

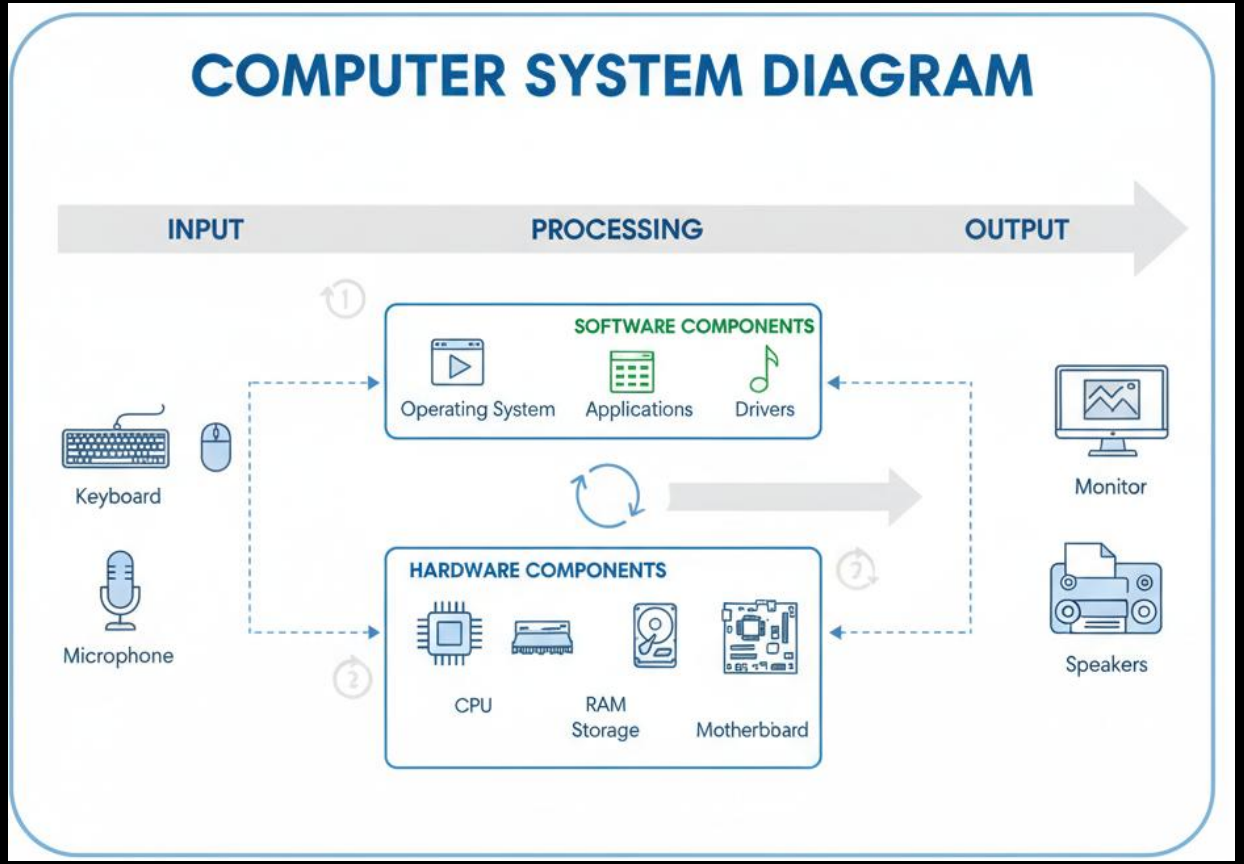
AMBITION GURU

Computer:

An electronic device that processes data to produce meaningful output.

Functions:

- Input
- Processing
- Storage
- Output
- Control


COMPUTER SYSTEM DIAGRAM
INPUT
PROCESSING
OUTPUT
SOFTWARE COMPONENTS
Operating System
Applications
Drivers
Keyboard
Microphone
HARDWARE COMPONENTS
CPU
RAM
Storage
Motherboard
Monitor
Speakers

Components of a PC

AMBITION GURU

Hardware

- CPU (Processor)
- RAM (Memory)
- Motherboard
- Hard Disk Drive (HDD/SSD)
- Monitor, Keyboard, Mouse

Software

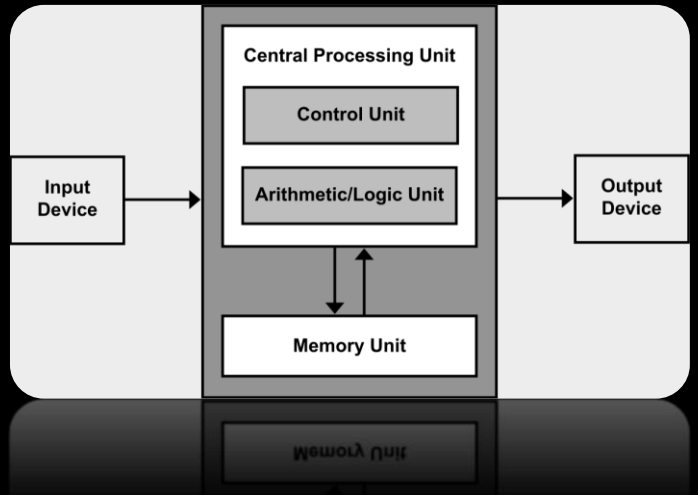
- Operating System (e.g., Windows, Linux)
- Applications (MS Office, Browser, etc.)

Computer Architecture

AMBITION GURU

Von Neumann Architecture:

- Von Neumann architecture is based on the stored-program computer concept, where instruction data and program data are stored in the same memory.
- Von Neumann architecture was first published by John von Neumann in 1945.
- The computer architecture design consists of a Control Unit, Arithmetic and Logic Unit (ALU), Memory Unit, Registers and Inputs/Outputs.



Memory Types

AMBITION GURU

Primary Memory:

Primary memory is volatile, fast, and directly accessible by the CPU for short-term storage. **E.g. RAM** (Volatile, fast), **ROM** (Non-volatile, stores firmware)

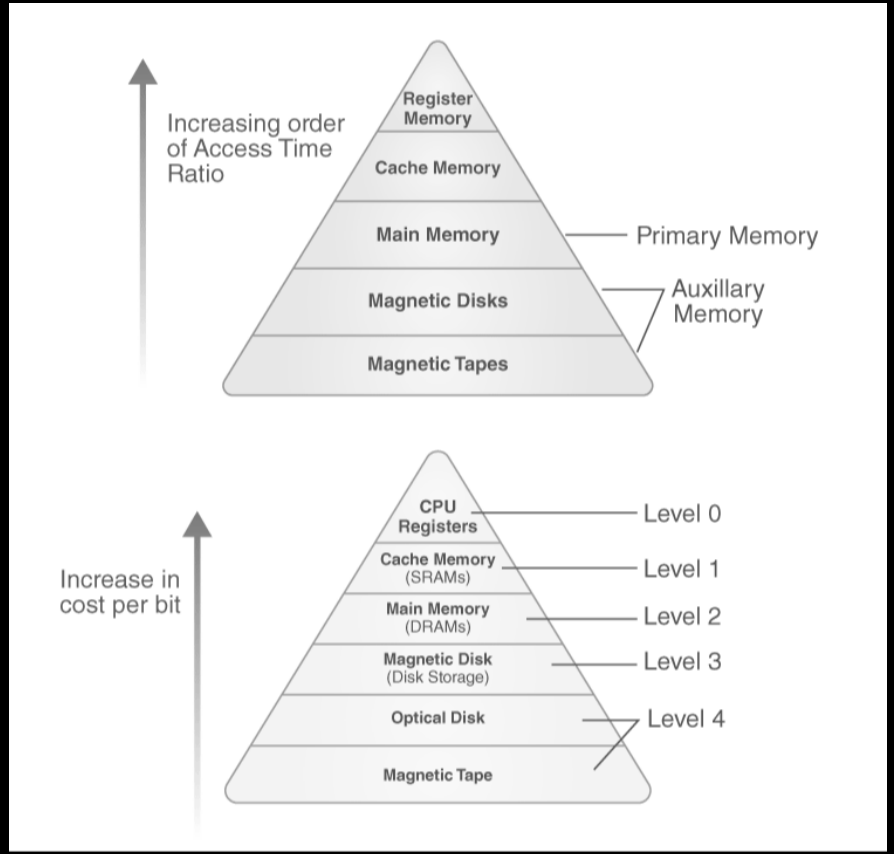
Secondary Memory:

Secondary memory is non-volatile, slower, and used for long-term data storage **E.g. Hard Disk**, **SSD**, **Pen Drive**, **CD/DVD**

Memory Hierarchy

AMBITION GURU

- A memory hierarchy is a structured arrangement of storage devices in a computer, organized by speed, cost, and capacity, to optimize data access.



Computer Peripherals

AMBITION GURU

Input Devices:

Keyboard, Mouse, Scanner, Microphone

Output Devices:

Monitor, Printer, Speaker

Input/Output Devices: Touchscreen, USB drives

Input and Output Devices

AMBITION GURU

Input:

Converts user actions to digital signals

E.g., Keyboard, Mouse, Joystick

Output:

Converts digital signals to human-readable form

E.g., Monitor (visual), Printer (paper), Speaker (audio)

Basics of Computer Networking

AMBITION GURU

Computer Network:

A computer network is a system of two or more interconnected computing devices that communicate and share data, resources, and services.

Types:

- LAN (Local Area Network)
- MAN (Metropolitan)
- WAN (Wide Area Network)

Components:

- Router, Switch, Hub, Modem, Network Cable

What is a Computer Program?

AMBITION GURU

A **set of instructions** written to perform a specific task. Written in programming languages like **C, C++, Python**.

Types of Programming Languages:

- Low-level (Machine, Assembly)
- High-level (C, Java, Python)

Steps for Program Development

AMBITION GURU

1. Define the problem
2. Plan the solution (Algorithm / Flowchart)
3. Write pseudocode
4. Convert to programming language (coding)
5. Compile and test the code
6. Debug and fix errors
7. Maintain and update the program

AMBITION GURU

Unit 1: Part 2

Problem Solving Tools: Algorithmic Thinking and Flowchart, Pseudocode

Introduction to Programming Language

AMBITION GURU

- A programming language is a standardized communication technique for describing instructions for a computer.
- Each programming language has a set of syntactic and semantic rules used to define computer programs.

Programming language are classified mainly in two categories:

- Low level programming language *(Types: Machine language & Assembly Language)*
- High level programming language

Differences between high level language and low-level language.

High Level Vs Low Level Language

AMBITION GURU

High Level Language	Low Level Language
It is programmer friendly language	It is machine friendly language
Compiler or interpreter is required for translation.	Assembler is required for translation.
They execute slower.	They execute faster.
For writing program hardware knowledge is not required.	For writing program hardware knowledge is required
They are easier to learn and understand by human.	It is difficult to learn and understand by human.
It can run on any platform	It is machine dependent
It is simple to debug and maintain.	It is difficult to debug and maintain as compared to high level language.
Example: C, C++, Java etc.	Example: Assembly language

Language Translator

- A programmer write a program in high level language that is to be translated into machine language equivalent code. This task is achieved by using language translator.

The common language translator are:

- Compiler
- Interpreter
- Assembler

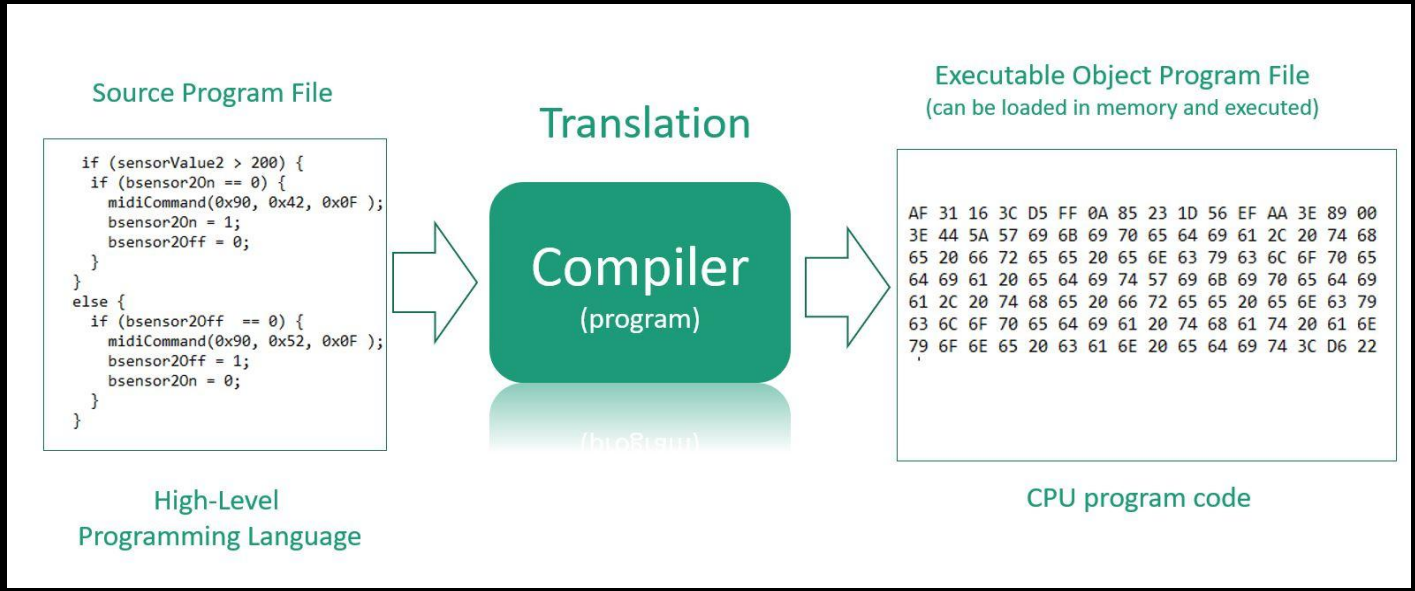
Language Translator: Compiler Vs Interpreter

AMBITION GURU

Compiler	Interpreter
A compiler translates the entire source code to object code and then only object code is executed.	An interpreter translates one statement at a time, executes it and continues for another statement
Compiler is faster than interpreter	Interpreter is slower than compiler
It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.	It continuously translates the program until the first error is met, in which case it stops. Hence debugging is easy.
A compiler is a complex program	Interpreter is less complex program than compiler.
As compared to an interpreter developing a compiler is difficult.	As compared to a compiler developing interpreter is easier.
Programming language like C,C++,FORTAN use compiler	Python use interpreter.

AMBITION GURU

Language Translator: Compiler



Language Translator: Assembler

AMBITION GURU

- An assembler is a program (software) which translates the program written in assembly language to machine language.
- It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.
- Assemblers are similar to compilers in that they produce executable code. However, assemblers are more simplistic since they only convert low-level code (assembly language) to machine code.
- Since each assembly language is designed for a specific processor, assembling a program is performed using a simple one-to-one mapping from assembly code to machine code in that they produce executable code.

Programming Design Tools:
Algorithm, Flow chart, Pseudo codes

Programming Design Tools: Algorithm

AMBITION GURU

An **Algorithm** is defined as a sequence of steps or procedures necessary to solve problem. To be an algorithm set of step must be unambiguous. Each step tells what task to be performed.

An *Excellent Algorithm* should have the following properties.

1. **Finiteness:** Each algorithm must have finite number of steps to solve a problem.
2. **Definiteness:** The action of each step should be defined clearly without any ambiguity.
3. **Inputs:** Inputs of the algorithm should be define precisely, which can be given initially or while the algorithm runs.
4. **Output:** An algorithm can give one or more result. After an algorithm terminates, algorithm must give desired result.
5. **Effectiveness:** An algorithm should be more effective among different ways of solving the problem.

Programming Design Tools: Algorithm

AMBITION GURU

Algorithm to find the sum of any two numbers

Step 1: Start

Step 2: Declare variables a ,b and sum

Step 3: Read value of a and b

Step 4: calculate sum= a + b

Step 5: Display sum

Step 6: Stop

Programming Design Tools: Algorithm

An algorithm to convert Fahrenheit when temp in Centigrade is given.

Step 1: Start

Step 2: Declare variables f and c

Step 3: Read value of c

Step 4: calculate f=1.8*c+32

Step 5: Display f

Step 6: Stop

Programming Design Tools: Algorithm

AMBITION GURU

Advantages of Algorithms:

- 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.

Programming Design Tools: Algorithm

AMBITION GURU

Disadvantages of Algorithms:

- Algorithm is time consuming.
- Difficult to show branching and looping in algorithms.
- Big tasks or problem are difficult to put in algorithms.

Flow chart

AMBITION GURU

- A flowchart is an pictorial representation of an algorithm.
- It uses boxes of different shapes to denote different types of instructions. The actual instructions are written within these boxes use clear and concise statements.
- These boxes are connected by solid lines having arrow marks to indicate the flow of operation, that is exact sequence in which instructions are to be executed.

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
	On-page Connectors	Used to connect remote flowchart portions of the same page.

Flow chart

Algorithm to find the sum of any two numbers

Step 1: Start

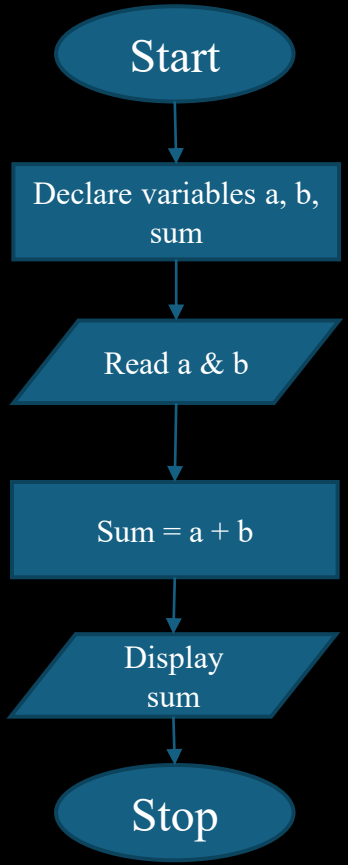
Step 2: Declare variables a ,b and sum

Step 3: Read value of a and b

Step 4: calculate sum= a + b

Step 5: Display sum

Step 6: Stop



Flow chart

Step 1: Start

Step 2: Enter the person’s age

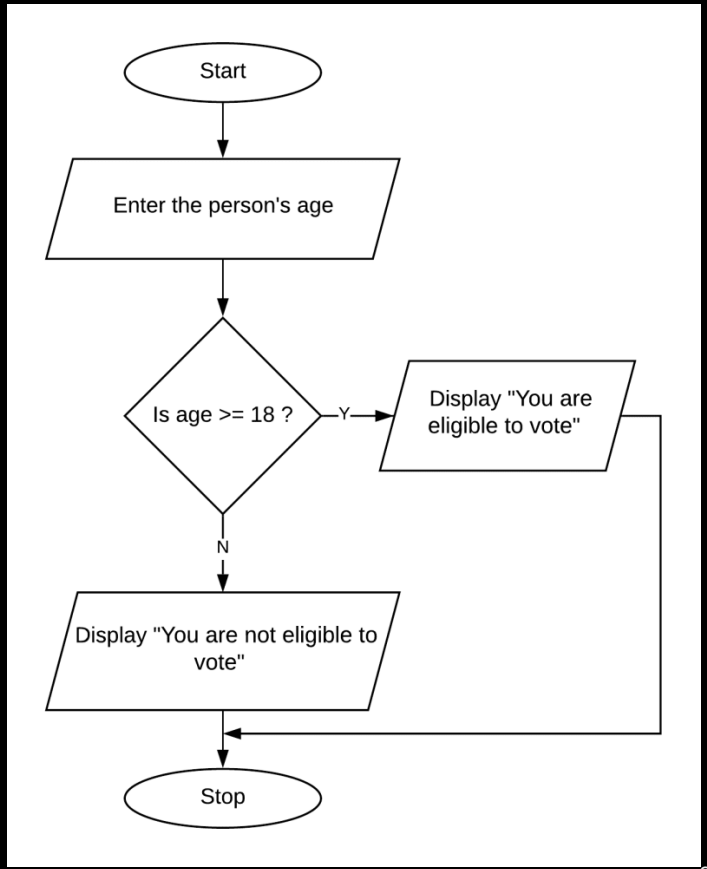
Step 3: If age is greater than or equal to 18:

Display "You are eligible to vote!".

Else:

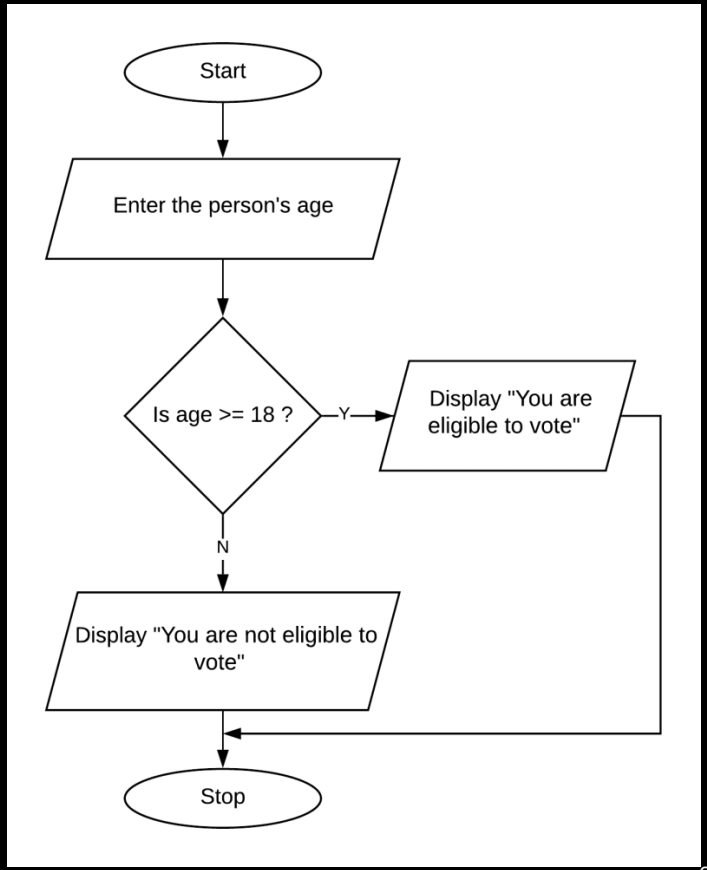
Display "You are not eligible to vote!".

Step 4: Stop



AMBITION GURU

Flow chart



Flow chart

AMBITION GURU

Advantages of using flowcharts

1. **Communication:** Flowcharts are better way of communicating the logic of the system to all concerned.
2. **Effective analysis:** With the help of flowchart, problem can be analyzed in more efficient way.
3. **Proper Documentation:** Program flowcharts serve as a good program documentation, which is needed for various purposes
4. **Efficient coding :** The flowcharts act as a guide or blueprint during the system analysis and program development phase
5. **Proper Debugging:** The flowchart helps in debugging process.
6. **Efficient program maintenance:** Maintenance of operating program becomes easy with the help of flowchart.

Flow chart

AMBITION GURU

Disadvantages of using flowcharts

1. **Complexity:** Flowcharts become difficult to manage and understand when dealing with large or complex systems.
2. **Time-consuming**: Creating and updating flowcharts requires significant time and effort.
3. **Inflexibility**: Small changes in the process may require redrawing the entire flowchart.
4. **Limited logic representation**: Flowcharts are not well-suited for illustrating complex decision-making or algorithms.
5. **Lack of standardization:** Misuse or inconsistent use of symbols can lead to confusion and misinterpretation.

Pseudocodes

- Pseudocode is a way of writing or describing programming code or algorithms in a natural language such as English.
- Pseudocode cannot be compiled nor executed as it is only meant to be read by humans. It is also known as 'false code' or 'representation of code.

Pseudocodes

```
// Pseudocode for adding two numbers

Start

1. Declare variables:
   - int num1, num2, sum;

2. Prompt the user to input the first number:
   - Print "Enter the first number:"
   - Read num1

3. Prompt the user to input the second number:
   - Print "Enter the second number:"
   - Read num2

4. Add the two numbers and store the result in sum:
   - sum = num1 + num2

5. Display the result:
   - Print "The sum is:" and the value of sum

End
```

AMBITION GURU

Pseudocodes

Pros of pseudocodes:

1. Easy to write
2. Easy to convert to code
3. Easy to communicate
4. Errors doesn’t matter
5. Simple to change or update.

Question: Write a pseudocode to find the smallest between two numbers.

AMBITION GURU

Unit 1: Part 3

Program Control Structures, Program Methodology, Program Models

Program

AMBITION GURU

- A program is a set of instructions that a computer can execute to perform a specific task.
- These instructions are written in a programming language, which is a formal language designed for expressing computations. Programs can range from simple scripts to complex applications.

Features of a Good Program:

A good program is not just about getting the job done; it's about doing it well. Here are some key features that distinguish a good program:

- **Correctness:** The program should produce the expected output for all valid inputs. It should handle errors gracefully and provide informative messages.
- **Efficiency:** The program should use resources (CPU time, memory) efficiently. It should avoid unnecessary computations or data storage.

Program

- **Readability:** The code should be easy to understand and maintain. Use meaningful variable names, comments, and proper indentation.
- **Maintainability:** The program should be easy to modify or update. Well-structured code with modular design makes it easier to change.
- **Usability:** The program should be easy to use for the intended audience. A good user interface can significantly improve the user experience.
- **Reliability:** The program should be robust and handle unexpected situations without crashing. Regular testing and debugging are essential.
- **Reusability:** The program or parts of it should be reusable in other projects. This can save time and effort in future development.

Program

- **Portability:** The program should be able to run on different platforms (operating systems, hardware) with minimal changes.
- **Security:** The program should be secure against vulnerabilities and attacks. This is especially important for programs handling sensitive data.
- **Documentation:** Good documentation is essential for both developers and users. It should include instructions, explanations, and examples.

Program Methodology

Program Methodology: Top-Down

AMBITION GURU

Definition:

In the Top-Down methodology:

- Start from the main problem
- Break it into smaller, manageable sub-problems
- Then design and implement each sub-module (function)

Steps:

1. Understand the complete problem.
2. Design the high-level solution (main function logic).
3. Identify sub-tasks and functions needed.
4. Implement each sub-task using functions.
5. Test each function and finally integrate.

Example:

C, Pascal, Fortran, etc.

Program Methodology: Bottom-Up

AMBITION GURU

Definition:

In the **Bottom-Up** methodology:

- Start by **designing and implementing individual components (functions)**
- Then integrate them to form the full application.

Steps:

- Identify reusable or independent modules first.
- Code and test each module.
- Gradually integrate modules to build the final program.

Example:

C++, Java, Python, etc.

Program Control Structures in C

Program Control Structures in C

Control structures define the flow or control of a program's execution. They help manage decision-making, looping, and branching, making programs flexible and dynamic.

C provides three main types of control structures:

1. Sequential Control Structure
2. Selection (Decision- Making) Control Structure
3. Repetition (Looping) Control Structure

AMBITION GURU

Program Control Structures in C

- Sequential Control Structure: Default flow of a program, statements are executed one after another, top to bottom.

- **Selection (Decision-Making) Control Structure:** Allows branching — the program makes decisions using conditions. E.g. if statement, if…else statement, etc.

- **Repetition (Looping) Control Structure:** Used when a block of code needs to run **multiple times**. E.g. for loop, while loop, etc.

Different Programming Techniques: Procedural Programming, Modular Programming, Object Oriented Programming

Procedural Programming

AMBITION GURU

Definition: A programming paradigm based on structured procedures or routines (functions) that operate on data.

Key Characteristics:

- Sequential execution of instructions.
- Uses procedures or functions to break the program into smaller parts.
- Data and functions are separate; data is passed to functions.

Procedural Programming

AMBITION GURU

Advantages:

- Simplicity: Easy to learn and implement for small-scale projects.
- Efficient: Direct control over hardware and resources.
- Good for linear and straightforward tasks.

Disadvantages:

- Poor scalability: Difficult to manage for large programs.
- Lack of reusability: Code often needs duplication.
- Difficult to maintain: Changes in data structures can require significant changes in functions.

Examples: Languages: C, Pascal, FORTRAN.

Modular Programming

Definition: A technique that divides a program into separate, self-contained modules that perform specific functions.

Key Characteristics:

- Emphasizes code modularity and reusability.
- Each module performs a distinct task and interacts with other modules.
- Facilitates team collaboration.

AMBITION GURU

Modular Programming

Advantages:

- Reusability: Modules can be used in different programs.
- Maintainability: Easier to debug and update individual modules.
- Scalability: Suitable for large and complex systems.

Disadvantages:

- Overhead: Slightly more complex than procedural programming.
- Dependency: Modules may need to be tightly coupled if not designed well.

Examples:

- Languages: Python, Java, Ada.

Object Oriented Programming

AMBITION GURU

Definition: A programming paradigm based on the concept of "objects," which contain data (attributes) and code (methods).

Key Characteristics:

- Encapsulation: Bundles data and methods that operate on the data.
- Inheritance: Enables a class to derive properties and behavior from another class.
- Polymorphism: Allows methods to perform different tasks based on the object that invokes them.
- Abstraction: Hides implementation details from the user.

Object Oriented Programming

AMBITION GURU

Advantages:

- Code reusability: Classes can be reused across programs.
- Scalability: Facilitates large-scale application development.
- Easier maintenance: Modular structure makes it easier to manage changes.

Disadvantages:

- Complexity: Can be harder to learn and implement for beginners.
- Overhead: May introduce performance and resource consumption issues.

Examples: Languages: Python, Java, C++, C#.

Differences between Procedural and Object-Oriented Programming

AMBITION GURU

Procedural Programming	Object Oriented Programming
In Procedural programming, a program is divided into small programs that are referred to as functions.	In OOP, a program is divided into small parts that are referred to as objects.
It is less secure than OOPs	Data hiding is possible in object-oriented programming due to abstraction. So, it is more secure than procedural programming.
It follows a top-down approach.	It follows a bottom-up approach.
There are no access modifiers in procedural programming.	The access modifiers in OOP are named as private, public, and protected.
Procedural programming does not have the concept of inheritance.	There is a feature of inheritance in object-oriented programming.
It is not appropriate for complex problems.	It is appropriate for complex problems.
Examples of Procedural programming include C, Fortran, Pascal, and VB.	The examples of object-oriented programming are - .NET, C#, Python, Java, VB.NET, and C++.

AMBITION GURU

Comparison

Feature	Procedural Programming	Modular Programming	Object-Oriented Programming
Focus	Sequence of instructions	Modularity of code	Objects and their interactions
Code Reusability	Limited	Moderate	High
Scalability	Low	Moderate	High
Data Management	Data and functions separate	Modular encapsulation	Data encapsulated with methods
Ease of Learning	Easy for small programs	Moderate	Complex

AMBITION GURU

Questions:

1. Define a language translator. Explain its role in programming languages. Name and briefly describe the types of language translators.
2. What is an interpreter? Explain how an interpreter works. Compare an interpreter with a compiler.
3. Define algorithm. List the characteristics of a good algorithm.
4. Define syntax and semantics in programming languages. Explain the difference between them with suitable examples.
5. Define flowchart. Write the symbols used in flowcharts and their purposes.
6. Draw a flowchart to find the largest of three numbers.
7. Define program. What are the features of a good program.
8. Write an algorithm to check whether a number is positive, negative, or zero. Draw a flowchart for it
9. Write an algorithm to calculate the factorial of a given number. Draw a flowchart for it.
10. Differences between Procedural and Object-Oriented Programming.
11. Write in brief about modular programming. List down its pros and cons.

THANK YOU
Any Queries ?