

Module 1: Introduction to Problem Solving

Premanand S

Assistant Professor,
School of Electronics and Engineering,
Vellore Institute of Technology, Chennai

premanand.s@vit.ac.in

August 2, 2024

Efficiency in solving problems depends on how correctly & precisely we define the problem, design the solution (algorithm), and implement the solution (code) using the programming language

[Unknown]

How our syllabus are framed?

- Pre-Programming Phase (Module 1)
- Coding Phase (Module 2 - Module 7) - Basics of Python

Topics to be covered in Pre-Programming Phase,

- Problem Solving: Definition and Steps
- Problem Analysis Chart (PAC), Interactivity Chart (IC), Input Process Output (IPO) chart, Coupling Diagram, and Data Dictionary
- Developing an Algorithm
- Flowchart
- Pseudo-code

Terminologies to brush up,

- Problem
- Programmers
- User
- Software
- Fact1: Computer only knows, how to follow codes in software
- Fact2: The programmer needs to know how to solve the problem

What is PROBLEM?

- A problem is a situation, condition, or issue that requires a solution or resolution.
- It often involves obstacles that hinder progress toward a desired goal.
- Problems can vary in complexity, from simple everyday tasks to intricate challenges in professional or technical fields.

What is PROBLEM? (Programming)

- A problem is a specific challenge or task that needs to be solved through code. It could be anything from automating a task, fixing a bug, or developing a new feature.
- Example: Imagine you're building an app and need to alphabetically sort a list of names. The need to sort is the problem.

Why do we need to solve the PROBLEM?

- Create the solution
- Improve the efficiency
- Enhance user experience
- Learning growth
- Drive business success
- Personal satisfaction

Problem solving in some domains,

- Engineering - Designing, testing, refining
- Medicine - Diagnosis illness, treatment, patient care
- Business - Strategic planning, managing operations, improving customer satisfaction, and driving innovation
- Information Technology - software, hardware, and networks
- Education - Curriculum, classroom and student management
- Everyday life - decision making, society

Why machines in problem-solving?

- Computer technology, completing any task quickly and accurately with skills
- Technology – not advanced enough to solve problems on own, hence we need to give step-by-step instruction to proceed
- Eg: Zomato, Swiggy – Ordering our favorite, Bookmyshow – Booking cinema in no time, Amazon, Netflix, Hotstar – Entertainment all in one place.
- Eg: Groceries in Netflix

Program Development Cycle

- Analyze – Problem definition
- Design – Problem solution
- Choose recipes – Flowchart, Pseudocode, Charts, and Algorithms
- Code – Converting algorithm to programming language
- Test & Debug – Real-time errors
- Documentation – Problem-solving to product development

Steps in Pre-programming phase

- Analyzing the Problem – Problem Analyzing Chart (PAC) - beginning analysis of the problem
- Developing Interactivity Chart (IC) - overall layout or structure of the solution.
- Developing Input Process Output (IPO) chart - shows the input, the processing, and the output
- Algorithms - show the sequence of instructions comprising the solution
- Program Flowchart - which are graphic representations of the algorithms
- Pseudocode - represents a language-like solution

Problem Analyzing Chart (PAC)

- A Problem Analysis Chart (PAC) is a tool used to break down and analyze a problem systematically.
- It helps in organizing thoughts and data related to the problem in a structured format.
- The primary purpose of a PAC is to gain a clear understanding of the problem and its components, which is crucial for developing effective solutions.

Components of PAC

- **Given data** - All the information provided in the problem statement, including constants and variables
- **Required Results** - Desired output or solution
- **Processing Required** - Necessary calculations, equations, and expressions needed to transform the given data into the required results
- **Solution Alternatives** - Potential approaches or algorithms

Example 1

- Simple Problem Analysis Chart (PAC) for the problem of being hungry
 - **Given Data:** Experiencing the physical and mental sensations of hunger.
 - **Required Results:** The goal is to alleviate the feeling of hunger and feel satisfied.
 - **Processing Required:** This outlines the steps needed - assessing food options, selecting something to eat, preparing and consuming it, and waiting for the hunger to go away.
 - **Solution Alternatives:** This lists different options to address the hunger, from eating a full meal to having a snack to drinking water or chewing gum

Example 2

- Calculate the amount for call driver for our own car, who works on hourly basis
 - **Given Data:**
 - **Required Results:**
 - **Processing Required:**
 - **Solution Alternatives:**

Example 2

- Calculate the amount for call driver for our own car, who works on hourly basis
 - **Given Data:** - Hourly wage rate for the driver (e.g., 150/hour) - Total hours worked (e.g., 8 hours) - Any additional costs (e.g., fuel, maintenance)
 - **Required Results:** Total amount to pay the driver for the hours worked, including any additional costs.
 - **Processing Required:** 1. Calculate the total payment based on hours worked and hourly rate.
2. Add any additional costs to the total payment.
 - **Solution Alternatives:** 1. Pay the driver based on a fixed hourly rate.
2. Offer a bonus for overtime hours worked.
3. Include a fuel reimbursement policy.

Example 3

- Calculate the average of five numbers by using PAC
 - **Given Data:**
 - **Required Results:**
 - **Processing Required:**
 - **Solution Alternatives:**

Example 3

- Calculate the average of five numbers by using PAC
 - **Given Data:** Five numbers to average (e.g., 12, 15, 20, 25, 30)
 - **Required Results:** Average of the five numbers (20.4)
 - **Processing Required:** 1. Sum the five numbers.
2. Divide the total sum by the number of values (5).
 - **Solution Alternatives:** 1. Use a calculator for quick computation.
2. Check for any outliers that might affect the average.

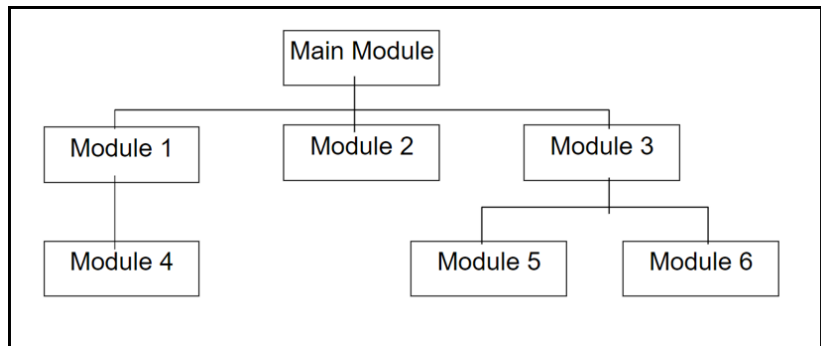
Assignments - PAC

- Calculate the gross pay of an employee by using PAC
- Convert the distance in kilometres to miles where, $1\text{mile} = 1.609\text{km}$ by using PAC
- Calculate the gross pay of an employee given the hours worked and rate of pay. The gross pay is calculated by multiplying the hours worked by the rate of pay.
- Analyse the problem for area of circle, $\text{area} = \pi * \text{radius} * \text{radius}$
- Write a PAC to compute and display the temperature inside the earth in Celsius & Fahrenheit, where $\text{Celsius} = 10 * (\text{depth}) + 20$ & $\text{Fahrenheit} = 1.8 * (\text{Celsius}) + 32$

Developing Interactivity Chart (IC)

- Serves to organize and represent the interactions between various modules or components of a system.
- It helps in breaking down complex problems into manageable parts, illustrating how different modules work together to achieve a goal.
- The IC can be particularly useful in structured programming, allowing developers to see the flow of data and control between different parts of a program.

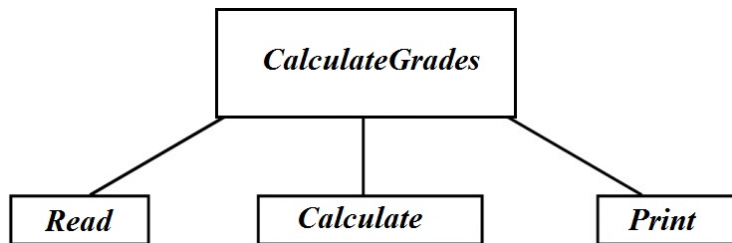
Interactivity Chart (IC)



Main components of Interactivity Chart (IC)

- **Main Module:** The control module that manages the overall process and coordinates the interaction between various sub-modules
- **Sub-Modules:** These are individual components that perform specific functions. Each module is connected to others, showing how they interact and depend on one another.
- **Hierarchy:** The IC is structured hierarchically, with the main module at the top and sub-modules branching out below. This layout helps in understanding the flow of control and data.
- **Processing Steps:** Each module may include specific processing steps that outline the actions taken to achieve the desired outcome.

Interactivity Chart (IC)



Developing Input Process Output (IPO) chart

- The concept of IPO (Input-Process-Output) being equal to the combination of PAC (Problem Analysis Chart) and IC (Interactivity Chart) suggests that these frameworks work together to provide a comprehensive understanding of a system or process
- **Inputs:** Resources, data, or materials required to start the process.
- **Processes:** Actions or transformations that convert inputs into outputs.
- **Outputs:** Final products, results, or services generated by the process.

Example - IPO

- Let's create an IPO chart for a simple example: Making a Fruit Smoothie:
 - **Inputs:** - Fresh fruits (e.g., bananas, strawberries) - Yogurt or milk - Ice - Blender
 - **Processes:** 1. Gather all ingredients. 2. Chop the fruits. 3. Add fruits, yogurt/milk, and ice to the blender. 4. Blend until smooth. 5. Pour the smoothie into a glass.
 - **Outputs:** - A fruit smoothie ready to drink - Optional: leftover smoothie for later consumption
 - **Module** - Blending

Assignments - IPO

- Calculating a Tip
- Withdrawing Cash from ATM
- Ordering Sandwich
- Registering for a Class

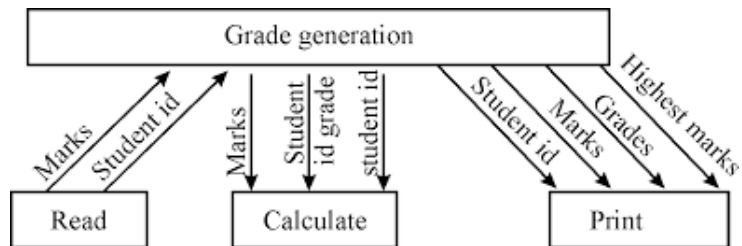
Coupling diagram

- A Coupling Diagram is a visual representation used to illustrate the relationships and dependencies between different components or modules within a system.
- It helps in understanding how changes in one part of the system may affect other parts, which is crucial for system design, analysis, and maintenance

Components of Coupling diagram

- **Components/Modules:** Each module represents a distinct part of the system, such as a class, function, or service.
- **Connections/Dependencies:** Arrows or lines indicate the relationships between modules, showing how they interact or depend on each other.
- **Data Flow:** The diagram may also illustrate the direction of data flow between components, indicating which module sends or receives data.

Coupling Diagram



Data Dictionary

- A Data Dictionary is a centralized repository that contains definitions, descriptions, and details about the data elements within a database or information system.
- It serves as a reference for users and developers, providing essential information about the structure, relationships, and constraints of the data.
- A data dictionary provides a clear overview of the tables, fields, and relationships within a database. This understanding of the data structure is crucial when trying to locate relevant data for analysis or troubleshooting.

Components of Data Dictionary

- **Object Names and Definitions:** Names and descriptions of tables, fields, and columns.
- **Data Types and Sizes:** Information about the types and sizes of data elements.
- **Classification and Relationships:** Classification of data and relationships between data elements.
- **Business Rules and Validation:** Rules and constraints for data validation.
- **Reference Data:** Information about missing or incomplete data.

Benefits of Data Dictionary

- **Improved Data Quality:** Ensures data consistency and accuracy.
- **Enhanced Data Accessibility:** Facilitates easier data discovery and usage.
- **Better Data Governance:** Provides transparency and control over data usage.
- **Reduced Data Anomalies:** Helps detect and correct data inconsistencies.

Data Dictionary

DATA					DATA DICTIONARY (METADATA)		
employee_id	first_name	last_name	nin	dept_id	Column	Data Type	Description
44	Simon	Martinez	HH 45 09 73 D	1	employee_id	int	Primary key of a table
45	Thomas	Goldstein	SA 75 35 42 B	2	first_name	nvarchar(50)	Employee first name
46	Eugene	Comelsen	NE 22 63 82	2	last_name	nvarchar(50)	Employee last name
47	Andrew	Petculescu	XY 29 87 61 A	1	nin	nvarchar(15)	National Identification Number
48	Ruth	Stadick	MA 12 89 36 A	15	position	nvarchar(50)	Current position title, e.g. Secretary
49	Bany	Scardelis	AT 20 73 18	2	dept_id	int	Employee department. Ref: Department
50	Sidney	Hunter	HW 12 94 21 C	6	gender	char(1)	M = Male, F = Female, Null = unknown
51	Jeffrey	Evans	LX 13 26 39 B	6	employment_start_date	date	Start date of employment in organization.
52	Doris	Bemdt	YA 49 88 11 A	3	employment_end_date	date	Employment end date.
53	Diane	Eaton	BE 08 74 68 A	1			

- An algorithm is a step-by-step procedure or formula for solving a problem.
- It is a sequence of instructions that can be followed to achieve a specific goal or perform a task.
- Algorithms are fundamental to computer science and programming, as they provide a clear method for processing data and making decisions.

- **Finite Sequence:** Algorithms are a finite sequence of instructions that must be followed to complete a particular task.
- **Well-Defined:** They must be clearly defined and produce at least one output.
- **Input and Output:** Algorithms take zero or more inputs and produce at least one output.
- **Execution:** They must be executed and finished in a finite number of steps.

Types of Algorithms

- **Exact Algorithms:** These solve problems exactly and are suitable for problems where exact solutions are required.
- **Approximation Algorithms:** These provide approximate solutions and are used for problems where exact solutions are impractical.

Steps in Algorithmic Problem Solving

- **Problem Analysis:** Understand the problem and identify the data requirements, processing requirements, and output requirements.
- **Developing a High-Level Algorithm:** Create a high-level plan for solving the problem, which is often represented using pseudocode or flowcharts.
- **Refining the Algorithm:** Add more detail to the high-level algorithm to make it more specific and executable.
- **Reviewing the Algorithm:** Ensure the algorithm solves the problem correctly and identify opportunities for simplification or generalization.

Example 1: Cooking Maggi with Egg

- Start: Begin by preparing the ingredients.
- Boil Water: Pour 1.5 cups of water into a small pot and bring it to a boil.
- Add Noodles: Open the Maggi noodles packet and add the noodles to the boiling water.
- Add Flavor Packet: Open the flavor packet and add it to the pot.
- Simmer: Lower the heat and simmer the noodles for 2 minutes.
- Add Egg: Crack an egg into a bowl, whisk it, and pour it into the pot.
- Stir: Stir the egg into the noodles until it is scrambled.
- Add Cheese: Stir in 0.5 cup of grated cheese.
- Serve: Serve the Maggi noodles hot.

Example 2: Adding Two Numbers

- Start: Begin the addition process.
- Read the First Number: 5.
- Read the Second Number: 3.
- Add the Numbers: $5 + 3 = 8$.
- Write the Sum: 8.
- Stop: End the addition process.

Example 3: Swap Two Numbers

- Step 1: Start
- Step 2: Take two numbers as input.
- Step 3: Declare a temporary variable.
- Step 4: Store the first number in the temporary variable.
- Step 5: Store the second number in the first number.
- Step 6: Store the temporary variable in the second number.
- Step 7: Print the first and second numbers.
- Step 8: End.

Example 4: Greater among three Numbers

- Start: Begin the comparison process.
- Read the First Number: 5.
- Read the Second Number: 8.
- Read the Third Number: 12.
- Compare the Numbers: 5 is less than 8.
- Compare the Numbers: 8 is less than 12.
- Write the Greater Number: 12.
- Stop: End the comparison process.

Assignment for Algorithms

- How to make biryani? (veg/non-veg)
- Find Roots of a Quadratic Equation $ax^2 + bx + c = 0$
- Find the factorial of a number
- Check whether a number is prime or not
- Find the Fibonacci series till the term less than 1000

Importance of Algorithms

- Algorithms can be represented in many forms, and among that there are few methods which will be used globally.
- There are few methods to represent algorithms in other forms,
 - Flowchart
 - Pseudocode
- It shows the logic behind algorithms without implementation.
- Non programmer also understand the flow.

- A flowchart is a visual representation of the sequence of steps and decisions in a process.
- It is used to illustrate the flow of data and control in a system.






Components of a Flowchart

- **Start:** The starting point of the flowchart.
- **Process:** The steps or actions performed in the process.
- **Decision:** A point where the flowchart branches based on a condition.
- **Input/Output:** The data that enters or exits the process.
- **Connector:** Lines that connect the different components of the flowchart.
- **Terminal:** The end point of the flowchart.

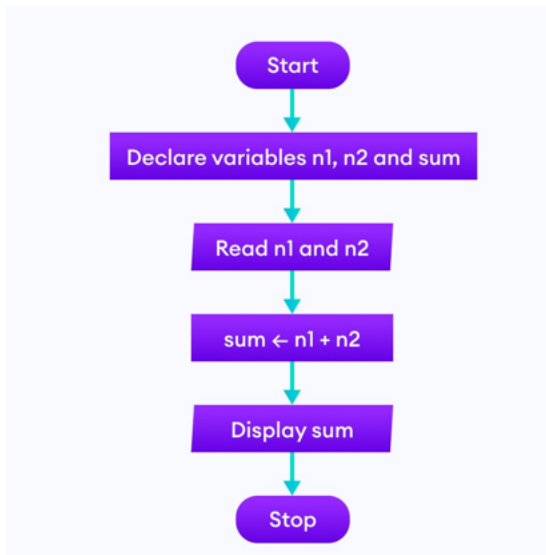
Symbols Used in Flowcharts

- **Rectangles:** Represent processes.
- **Oval:** Start and End operation
- **Diamonds:** Represent decisions.
- **Parallelogram** Represent inputs and outputs.
- **Arrows:** Connect the different components of the flowchart.

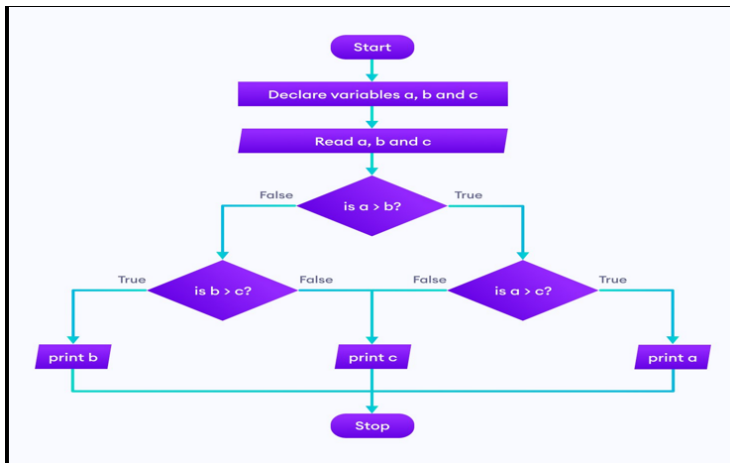
Flowchart Symbols

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

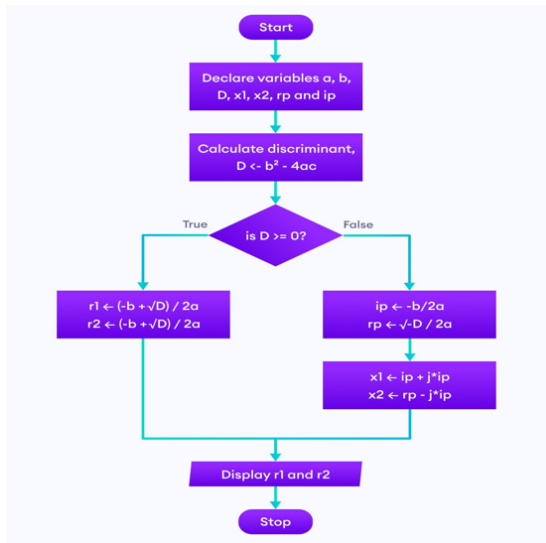
Example 1: Adding two numbers



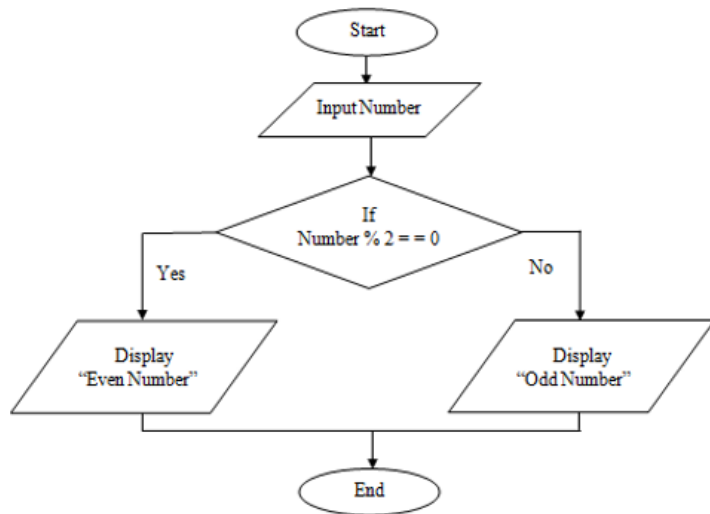
Example 2: Greatest among three numbers



Example 3: Finding all the roots of a quadratic equation



Example 4: Odd or Even



Assignment - Flowchart

- Read the sequence of numbers, find the average of the number and print the average.
- Find the Fibonacci series till term 1000.
- Hiring process in any company starting from advertisement till offer letter
- Draw a flowchart for - Should I Break Up With Him or Her

Pseudocode

- Pseudocode is a method used to describe the steps of an algorithm in a way that is easy to understand for anyone with basic programming knowledge.
- It combines elements of programming languages with informal language to outline the logic of a program without getting bogged down by syntax rules.
- This makes it a valuable tool for planning, designing, and communicating algorithms.

Common constructs used in Pseudocode

- SEQUENCE: Represents linear tasks performed one after the other.
- IF-THEN-ELSE: Conditional statements that dictate different actions based on conditions.
- WHILE: Loops that continue as long as a condition is true.
- FOR: Loops that iterate a specific number of times.
- REPEAT-UNTIL: Loops that continue until a condition is met.
- CASE: A generalized form of IF-THEN-ELSE for multiple conditions.
- CALL: Used for invoking classes or calling functions.
- EXCEPTION: Used for handling exceptions, along with the WHEN keyword.

Do's and Dont's

// program to find the largest of 2 numbers

```
IF n1>n2
    Print "n1"
ELSE
    Print "n2"
```



```
IF number 1 is greater than number 2
    Print "number 1 is greater"
ELSE
    Print "number 2 is greater"
```



Pseudocode

```
INPUT orderAmount
IF orderAmount >= 50 THEN
    APPLY discount
    DISPLAY "Discount has been applied!"
ELSE
    DISPLAY "Add more items to cart for discount!"
ENDIF
END
```

Advantage of using Pseudocode

- **Simplifies Logic:** It allows programmers to focus on the logic and flow of the algorithm without being distracted by specific programming syntax.
- **Facilitates Collaboration:** Pseudocode can be easily understood by team members who may not be familiar with a specific programming language, enhancing communication.
- **Error Reduction:** Writing pseudocode helps identify logical errors before actual coding begins, making the coding process smoother and more efficient.

Example 1: Basic Even/Odd Check

```
INPUT number
IF number MOD 2 = 0 THEN
    DISPLAY number + " is even."
ELSE
    DISPLAY number + " is odd."
ENDIF
END
```

Example 2: Sum of Two Numbers

START

```
// This program calculates the sum of two numbers
```

```
DECLARE number1, number2, sum AS INTEGER
```

```
// Get user input for the first number
```

```
DISPLAY "Enter the first number:"
```

```
INPUT number1
```

```
// Get user input for the second number
```

```
DISPLAY "Enter the second number:"
```

```
INPUT number2
```

```
// Calculate the sum of the two numbers
```

```
sum = number1 + number2
```

```
// Display the result
```

```
DISPLAY "The sum of " + number1 + " and " + number2 + " is " + sum
```

END

Example 3: Comparing two numbers

```
START
// This program compares two numbers
DECLARE number1, number2 AS INTEGER

// Get user input for the first number
DISPLAY "Enter the first number:"
INPUT number1

// Get user input for the second number
DISPLAY "Enter the second number:"
INPUT number2

// Compare the numbers using IF-THEN-ELSE
IF number1 is greater than number2 THEN
    DISPLAY number1 + " is greater than " + number2
ELSE IF number1 is less than number2 THEN
    DISPLAY number2 + " is greater than " + number1
ELSE
    DISPLAY number1 + " and " + number2 + " are equal"
ENDIF
END
```

Example 4: For comparsion

- If student's grade is greater than or equal to 60
- Print passed
- Else Print failed

Example 5: Area of rectangle

- READ height of Rectangle
- READ width of Rectangle
- COMPUTE area as height times width
- PRINT area of rectangle

Assignment for Pseudocode

- Write pseudocode for calculating Area ($l \times h$) and Perimeter ($2 \times (l + h)$) of Rectangle
- Write pseudocode for issue of driver license (if else)
- Write pseudocode for a given number is positive or negative
- Write pseudocode to read 50 numbers and find their sum and average. (for loop)

Revision 1: You can apply all the concepts

- Mr. Jones always gives True/False tests to his class. His tests always have 20 questions. The maximum class size is 35. He needs a program to calculate the students' grades based on the best score. Grade A will range from the best score, to the best score minus 2. B will range from the best score minus 3, to the best score minus 4. C will range from the best score minus 5, to the best score minus 6. D will range from the best score minus 7, to the best score minus 8. F will be anything below the best score minus 8. Each student's ID and test answers will be entered. The output will be each student's ID, number correct, and grade, along with the single highest score for the class. Develop a solution for Mr. Jones's problem. Use four one-dimensional arrays—one for the correct scores and the other three for the needed output.

Revision 2: You can apply all the concepts

- A restaurant manager wants to know how many employees are needed at the restaurant each hour of the day. The minimum number of employees needed at any hour is 3. After that, one additional employee is required for each 20 customers. The restaurant is open 24 hours a day. The manager has counted the number of customers each hour for 14 days. The manager will use the average number of customers for each hour over the 14 days to calculate the needed number of employees for each hour. Develop a solution to output the needed number of employees per hour. (There is no such thing as a partial employee.)

Revision 3: You can apply all the concepts

- An instructor has 30 students in her class. Each student is identified by a number from 1 to 30. Grades are stored in a one-dimensional array. The instructor would like to enter a student number and have the student's test score printed on the monitor. Develop a solution to output the needed information.

Verdict from Module 1



Verdict from Module 1

- Problem Solving,
 - Identifying the problem
 - Finding solution
 - Implementing algorithms
 - **Developing program through any programming language (Module 2 onwards)**

Importance of Computers or coding methods,

Computer Science is a science of abstraction creating the right model for a problem and devising the appropriate mechanizable technique to solve it.

[Alfred Aho & Jeffery Ullman]

mail me: er.anandprem@gmail.com / premanand.s@vit.ac.in

ring me: +91 73586 79961

Follow me: LinkedIn

Medium Blogs

Analytics Vidhya: Blogs

**Learning gives Creativity,
Creativity leads to Thinking,
Thinking provides Knowledge,
and
Knowledge makes you Great
- Dr APJ Abdul Kalam**