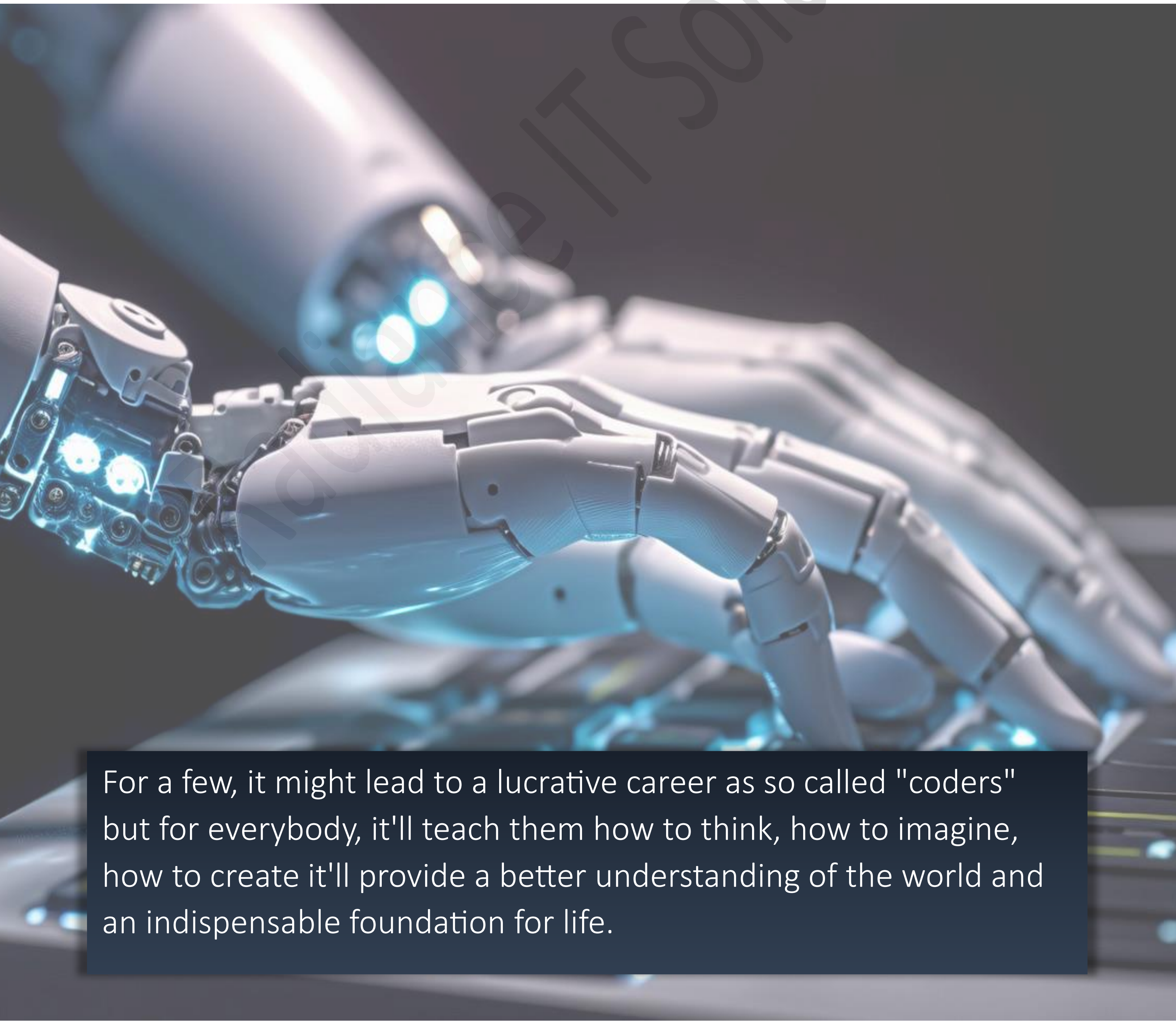
The Art of Computational Thinking

Algorithms, Flowcharts, and Programming



For a few, it might lead to a lucrative career as so called "coders" but for everybody, it'll teach them how to think, how to imagine, how to create it'll provide a better understanding of the world and an indispensable foundation for life.

The Art of Computational Thinking: Algorithms, Flowcharts, and Programming

Computational thinking is a problem-solving and cognitive process that involves approaching complex problems in a way that a computer, or a human acting as a computer, could solve them. It's a fundamental skill in computer science and is not just about learning how to program; rather, it's a broader set of skills that encompasses various strategies for problem-solving and systematic thinking.

Key components of computational thinking include:

1. **Decomposition:** Breaking down a complex problem or system into smaller, more manageable parts or steps. This helps in understanding and solving each part independently.

2. **Pattern Recognition:** Identifying patterns, trends, and regularities within data or problems. Recognizing similarities between different problems can lead to more efficient solutions.

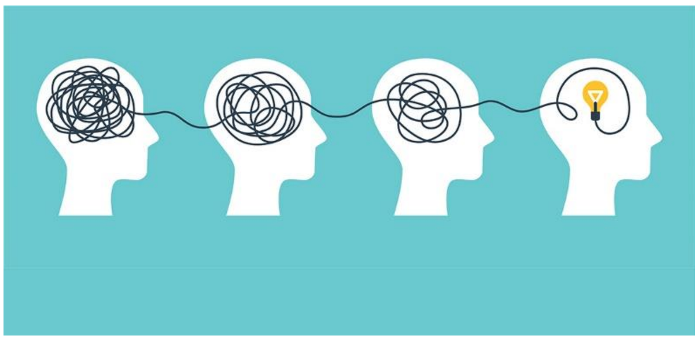
3. **Abstraction:** Removing unnecessary details and focusing on the essential aspects of a problem. This involves creating models or representations that capture the key elements without getting bogged down in specifics.

4. **Algorithmic Thinking:** Developing a step-by-step solution or set of rules to solve a problem. This is where algorithms come into play, providing a clear and systematic method for reaching a solution.

Computational thinking is not limited to computer scientists but is a **valuable skill for anyone dealing with complex problems in various domains.** It can be applied in fields like mathematics, science, engineering, and, of course, in programming and ICT.

Computational thinking is an art—a creative and strategic approach to problem-solving that involves understanding, breaking down, and solving problems using algorithms and programming.

It suggests that these skills are not just technical tools but part of a broader, more artistic process of thinking and problem-solving.



Algorithms, flowcharts, and programming are interconnected concepts in the field of computer science and information and communication technology (ICT).

Let's break down their connections:

1. Algorithms:

- An algorithm is a step-by-step procedure or set of rules for solving a specific problem or accomplishing a particular task.

- It's a high-level description of the logic and steps needed to perform a computation or solve a problem.

2. Flowcharts:

- A flowchart is a visual representation of the steps involved in a process or algorithm.

- It uses different shapes and arrows to represent different types of steps and the flow of control in a program or process.

- Flowcharts are a way to represent algorithms graphically, making it easier to understand the logical flow of a program or process.

3. Programming:

- Programming is the process of translating an algorithm into a programming language that a computer can execute.

- Programmers use programming languages like Python, Java, C++, etc., to write code that implements the steps outlined in the algorithm.

- The code is a set of instructions that tells the computer how to perform the tasks specified by the algorithm.

Connections:

- Algorithm to Flowchart: Before writing code, it's often helpful to create a flowchart to visualize the logical flow of the algorithm. This helps in understanding the structure and sequence of steps.

- Flowchart to Programming: The flowchart serves as a blueprint for writing code. Each step in the flowchart corresponds to a set of instructions in the program.

- Algorithm to Programming: The algorithm provides the logical framework for solving a problem. Programming involves turning that logical framework into executable code.

In summary, **algorithms provide the conceptual foundation for solving problems, flowcharts offer a visual representation of the algorithmic steps, and programming is the practical implementation of the algorithm in a specific programming language.** Together, they form a cohesive process for developing solutions in the realm of ICT and computer science.

Computers are, in a sense, "dumb" machines that rely on explicit instructions from humans. This notion is at the core of why programming is essential. Computers lack inherent understanding and intelligence; instead, they operate based on the precise commands and algorithms given to them.

Breakdown of the key points:

1. Computers as Machines:
   - Description: Referring to computers as machines underscores their mechanical nature. They lack inherent knowledge or decision-making abilities.
   - Implication: This highlights the need for external guidance and input to make computers perform desired tasks.

2. Dependence on Instructions:
   - Description: Stating that computers need instructions for every action emphasizes their dependence on human guidance.
   - Implication: It underscores the user's role in providing clear, detailed instructions for the computer to carry out tasks effectively.

3. Programming as Communication:
   - Description: Expressing that everything in a computer works based on instructions suggests that programming is a form of communication with the machine.
   - Implication: Programming becomes a language through which humans convey their intentions and logic to the computer.

4. The Importance of Learning Programming:
   - Description: Emphasizing the necessity of learning programming implies that it is the means by which individuals can communicate with and command computers.
   - Implication: Learning programming empowers individuals to interact with computers effectively, enabling them to perform a wide range of tasks.

Ultimately, it's not helpful to label computers as dumb or smart. They're different machines with unique strengths and weaknesses. Our partnership lies in leveraging their computational power while providing the instructions and understanding they lack. Together, we can achieve amazing things!

Think of it like this: Would you call a hammer dumb because it needs someone to swing it? It does its job flawlessly, but it needs human direction. Computers are similar – powerful tools waiting to be guided by our creativity and purpose.

"The Pencil and Computer are, if left to their own devices, equally dumb and only as good as the person driving them."

- Norman Foster-

Process of Solving Practical Problems and Understanding Algorithms

Imagine that a group of relatives have arrived when you were alone in the house. You need to serve them some tea. Here, you should prepare a cup of tea by following different steps.

On another occasion, you will have to make a fruit salad for a dessert or make a birthday cake. On all these occasions, you need to solve problems. Compare it with calculating the area of a rectangle during your mathematics lesson.

When we have a certain aim, we do certain activities to achieve it. In our day to day life, we often solve problems.

In the realm of ICT and problem-solving, understanding the process of solving practical problems is similar to following a well-defined algorithm. We'll explore the practical aspects of problem-solving and the importance of algorithms in everyday tasks.

Problem Solving in Everyday Life:

The problem needs to be analyzed well before solving it. Then you can get a good understanding of how to solve the problem. The process of problem solving has an input, an outcome, and a process.

In day to day life, problem-solving is a common occurrence, whether it's preparing tea for unexpected guests, making a fruit salad for dessert, or calculating the area of a rectangle in a math lesson.

Problem-solving involves careful analysis, understanding the goal, and following a systematic process to achieve the desired outcome.

The Problem-Solving Process:

1. Input: Identify the **elements required** to solve the problem. For making a fruit salad, it's a variety of fruits; for calculating the area of a rectangle, it's the length and width.

2. Process: Follow **a set of guidelines** or steps to solve the problem. This includes washing, cutting, and mixing for the fruit salad, and multiplying length by width for the area of a rectangle.

3. Output: Obtain the result or **outcome** of the problem-solving process. For the fruit salad, the output is the delicious fruit salad itself; for the rectangle, it's the area.

So, think of it like this: when you follow a recipe step by step, you're essentially solving a problem—creating a delightful dish!

Example: 1 The input, process, and output of preparing a fruit salad is as follows.

Input - a variety of fruits
Process - washing fruits, cutting fruit, mixing
Output - Fruit salad

Example: 2 The input, process, and output of finding the area of a rectangle are as follows.

Input - the length and the width of the rectangle
Process - length x width
Output - area of the rectangle

Algorithm: Understanding the Systematic Approach:

An algorithm is a method that includes all the steps needed to solve a problem in a specific order.

It is comparable to everyday activities like tying shoelaces, putting on a school uniform, or preparing a cup of tea—all of which follow a set sequence of steps.

Step 01		Finding various kinds of fruits
Step 02		Washing all the fruits well
Step 03		Cutting fruits into small pieces
Step 04		Putting the pieces of fruit into a bowl
Step 05		Add sugar and mix
Step 06		Serve the fruit salad in bowls

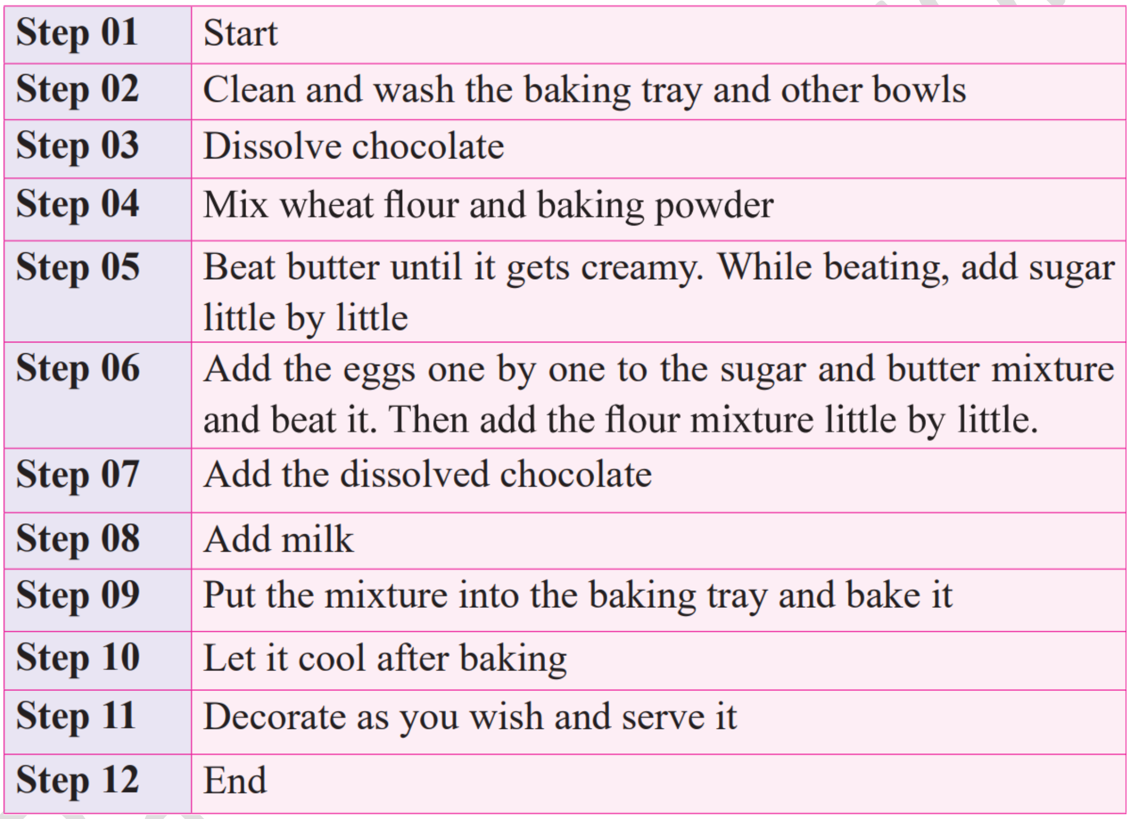
Writing an Algorithm: Best Practices:

- Sequential Steps: It's crucial to write the steps of an algorithm in sequential order. Any deviation can impact the outcome, as seen in the example of washing fruits after adding sugar to a fruit salad.

- Start and End: Every algorithm must have a clear start and end. Initial steps are vital to set the stage for problem-solving, and the end signifies the completion of the algorithm.

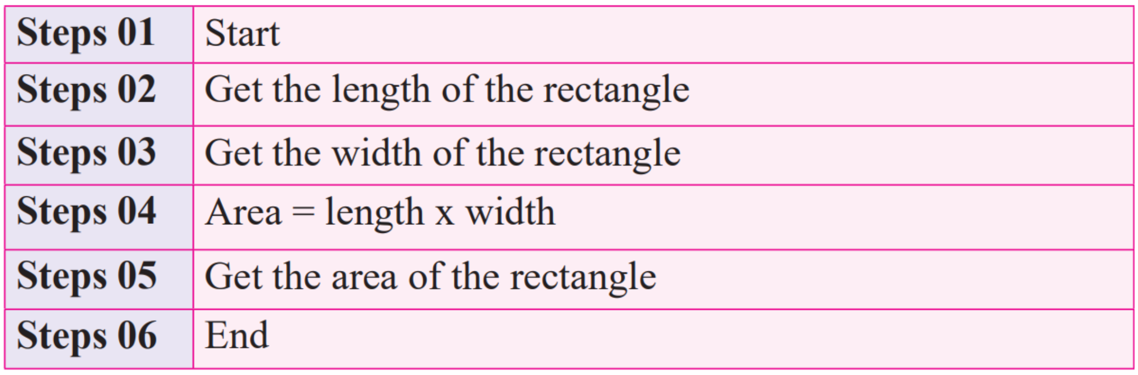
Examples:

The algorithm for making a chocolate cake:



Step 01	Start
Step 02	Clean and wash the baking tray and other bowls
Step 03	Dissolve chocolate
Step 04	Mix wheat flour and baking powder
Step 05	Beat butter until it gets creamy. While beating, add sugar little by little
Step 06	Add the eggs one by one to the sugar and butter mixture and beat it. Then add the flour mixture little by little.
Step 07	Add the dissolved chocolate
Step 08	Add milk
Step 09	Put the mixture into the baking tray and bake it
Step 10	Let it cool after baking
Step 11	Decorate as you wish and serve it
Step 12	End

Finding the Area of a Rectangle:

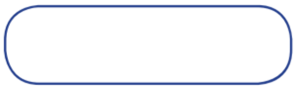
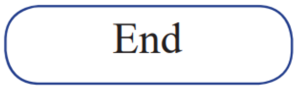
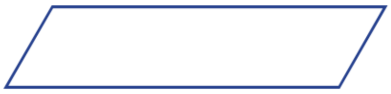
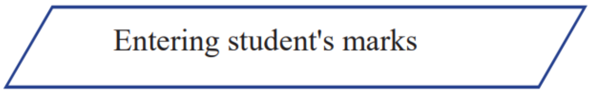
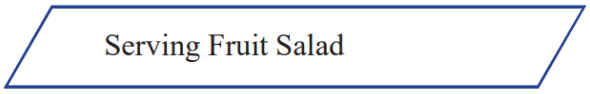
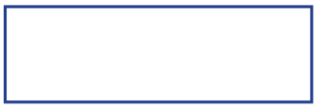
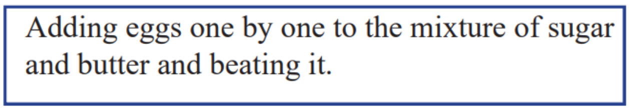
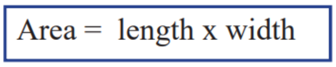
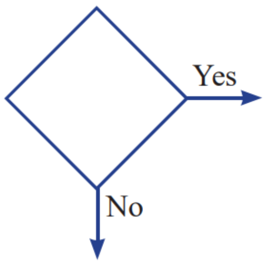
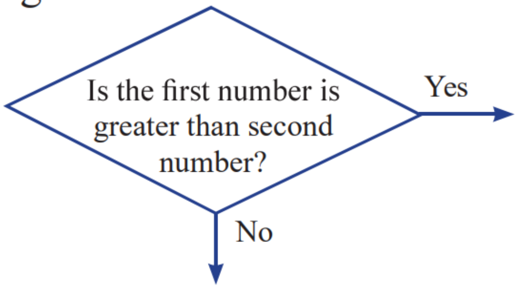
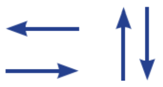


Steps 01	Start
Steps 02	Get the length of the rectangle
Steps 03	Get the width of the rectangle
Steps 04	Area = length x width
Steps 05	Get the area of the rectangle
Steps 06	End

Flow Chart

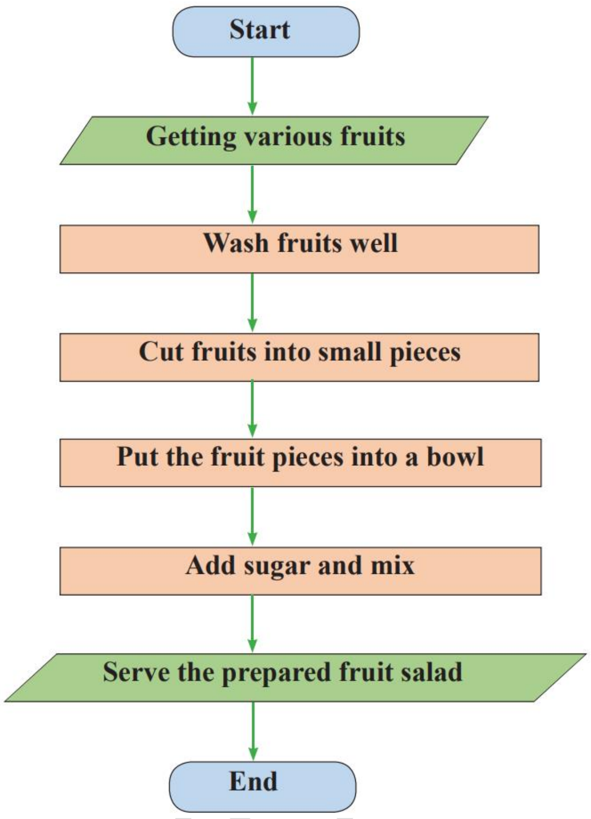
A flow chart is a graphical representation of the algorithmic steps.

Here, Basic symbols are used to show each action.

Symbol	Usage
	Used to indicate the start and the end. Eg:  End
	Used to indicate the input and the output. Eg:  Entering student's marks  Serving Fruit Salad
	Used to show an action/a process Eg:  Adding eggs one by one to the mixture of sugar and butter and beating it.  Area = length x width
	Used to indicate an instance of decision making.  Is the first number is greater than second number?
	It is used to indicate the direction of data flow.

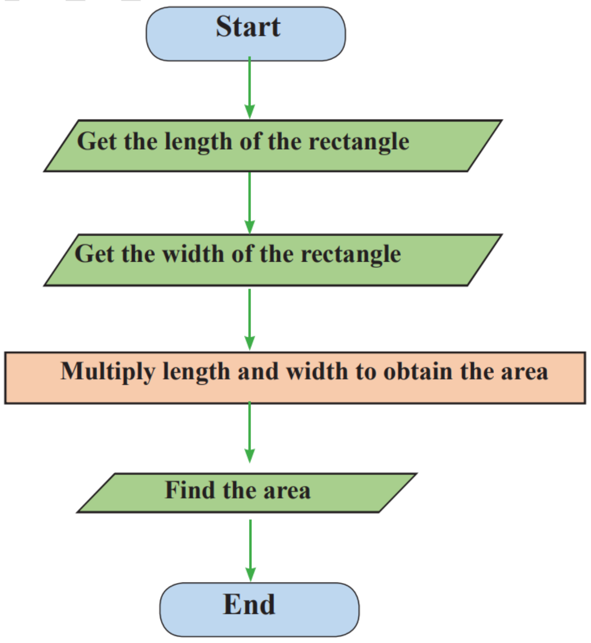
Example 01
Drawing the flow chart for making a fruit salad using the above symbols is given below.

Here the symbols related to start, end input output, and process are used.



Start
Getting various fruits
Wash fruits well
Cut fruits into small pieces
Put the fruit pieces into a bowl
Add sugar and mix
Serve the prepared fruit salad
End

Example 02
Drawing the flow chart for finding the area of a rectangle is given below.

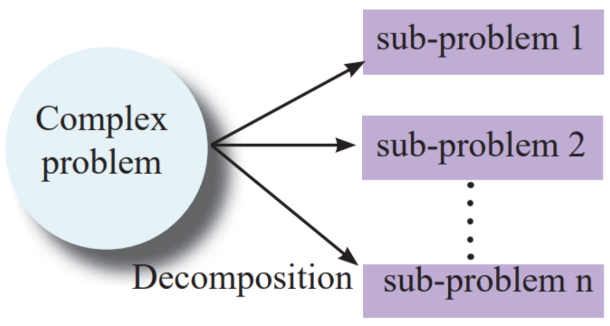


Start
Get the length of the rectangle
Get the width of the rectangle
Multiply length and width to obtain the area
Find the area
End

Decomposing the Problems

In the exciting journey of algorithms, flowcharts, and programming, we dive into a crucial concept: **Decomposing the Problems.** As we progress, we encounter problems of varying complexities, and It is essential to understand a problem thoroughly before going to solve it.

So it will be easier to find a solution to a complex problem after decomposing it into smaller sub-problems.



Advantages of decomposing a problems

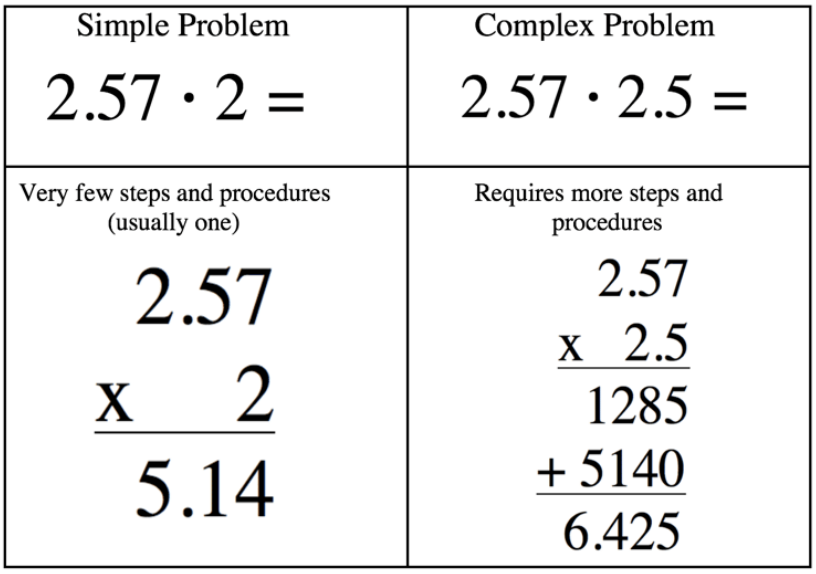
- Easy to solve
- Easy to understand
- Easy to describe
- Easy to analyze

Decomposing: Breaking down a problem into smaller, more manageable problem parts.

Navigating Complexity:

Understanding Problem Nature:
Problems vary in complexity—some are simple, while others are more intricate. Understanding a problem becomes more challenging when it's complex.

Before you start solving a problem, it's crucial to understand all the details of the problem.

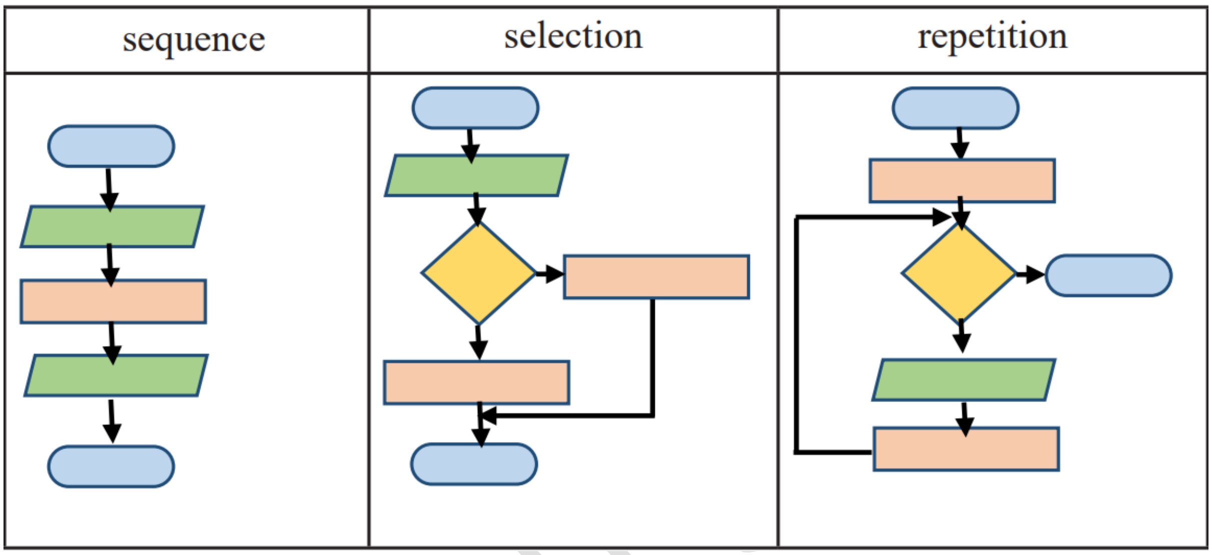


Simple Problem	Complex Problem
$2.57 \cdot 2 =$	$2.57 \cdot 2.5 =$
Very few steps and procedures (usually one)	Requires more steps and procedures
$\begin{array}{r} 2.57 \\ \underline{\text{x} \quad 2} \\ 5.14 \end{array}$	$\begin{array}{r} 2.57 \\ \underline{\text{x} \quad 2.5} \\ 1285 \\ \underline{+\,5140} \\ 6.425 \end{array}$

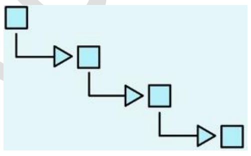
Using Flowcharts for Algorithms

A flow chart is a graphical representation of the algorithmic steps.

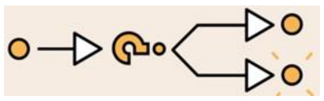
In Early, we explored how flowcharts serve as a visual tool to represent algorithms. Building on that foundation, here we introduce three fundamental control structures that play a pivotal role in shaping algorithms.



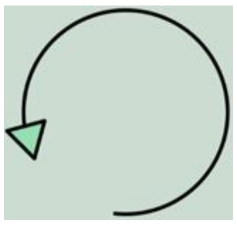
1. Sequence:
Execution of instructions in an algorithm sequentially from top to the bottom is called a sequence.



2. Selection:
Here it is expected to make a making decision on which step to follow depending on the condition given by the algorithm.
In a selection, the condition is checked first, and the flow direction is chosen based on whether the condition is true or false.



3. Repetition:
Execution of an instruction or several instructions in an algorithm repeatedly until a condition is satisfied is called repetition.

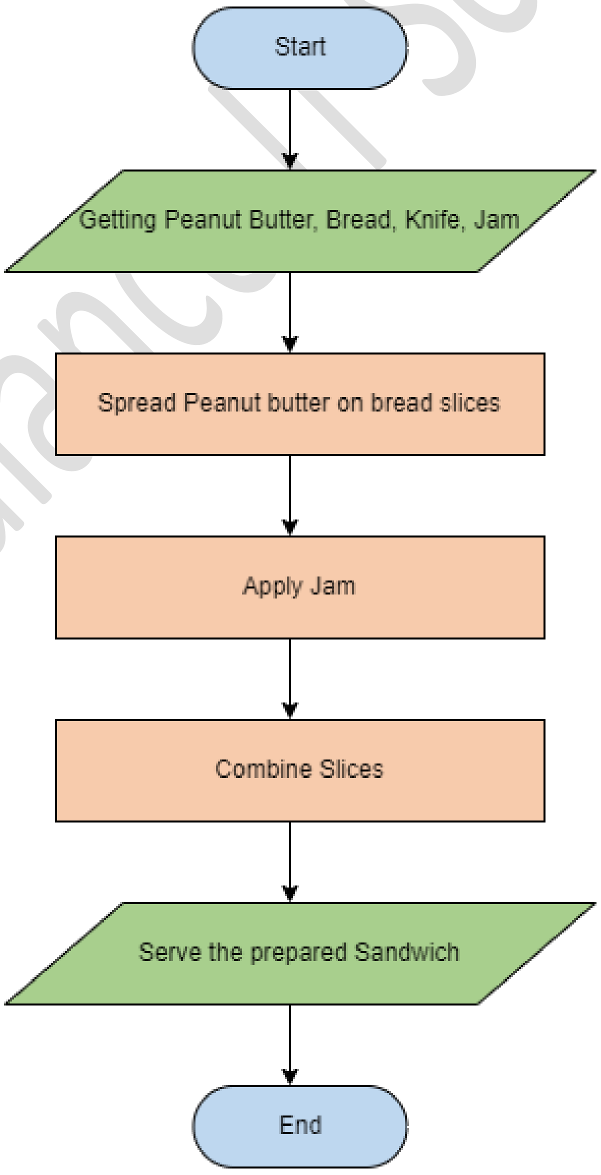


Essential Control Structures:

1. Sequence:
- Symbol: Represented by a straight arrow in a flowchart.
- Concept: If the steps from the beginning to the end of an algorithm are carried out in a strict order, it is called a sequence. where actions occur in a linear order.

User Case: Making a Peanut Butter Sandwich
- Steps:
  1. Spread Peanut Butter: Open the jar, take a knife, and spread peanut butter on one slice of bread.
  2. Apply Jam: Take another slice and spread jam.
  3. Combine Slices: Place the slices together to form a sandwich.

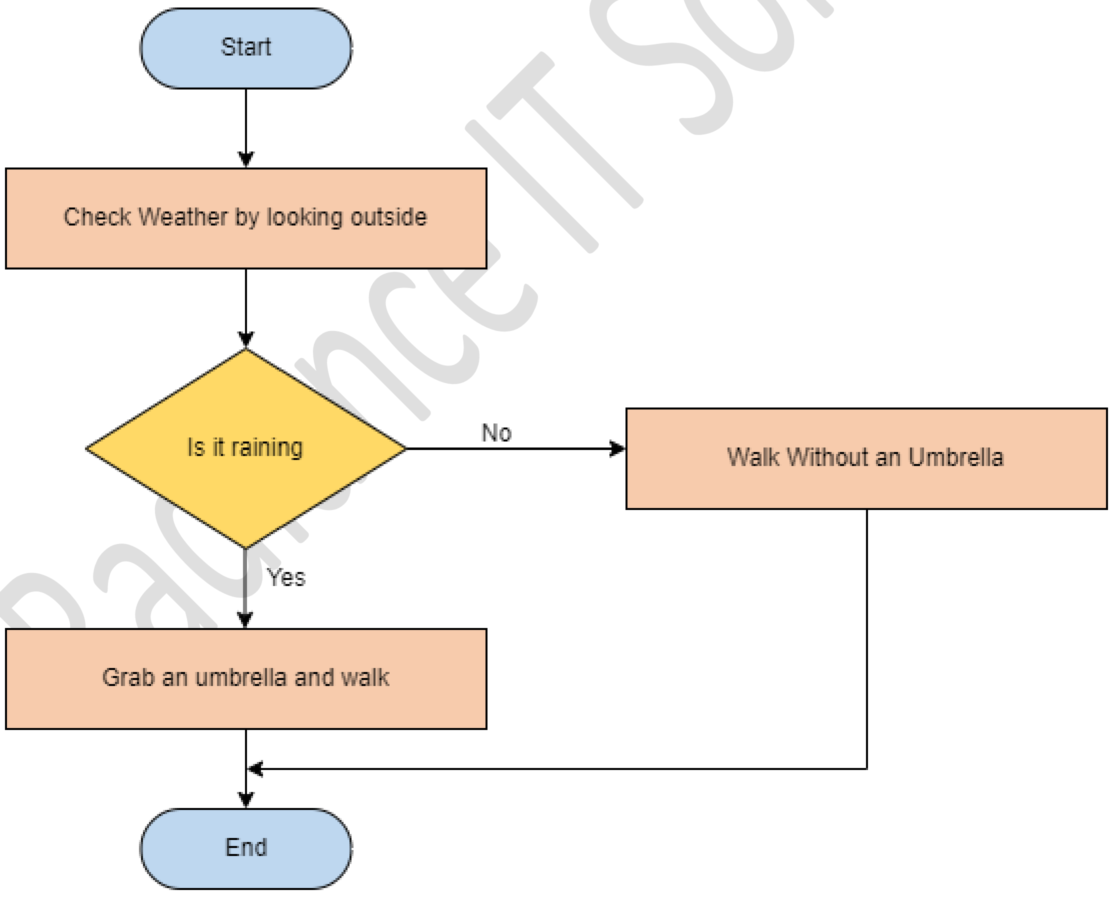
Concept Explanation: The steps are carried out in a strict order—spread peanut butter, apply jam, and then combine the slices. This linear sequence ensures the sandwich is made correctly.


Start
Getting Peanut Butter, Bread, Knife, Jam
Spread Peanut butter on bread slices
Apply Jam
Combine Slices
Serve the prepared Sandwich
End

2. Selection:
- Symbol: Represented by a diamond-shaped symbol.
- Concept: For decision-making, it’s like choosing a path depending on a 'yes' or 'no' scenario.

User Case: Weather-Dependent Decision
- Steps:
  1. Weather Check: Look outside to see if it's raining.
  2. Decision Point:
     - Yes: Grab an umbrella and walk.
     - No: Walk without an umbrella.

Concept Explanation: This user case involves a straightforward decision-making process based on the weather. If it's raining, the decision is to grab an umbrella and walk; otherwise, just walk without an umbrella.

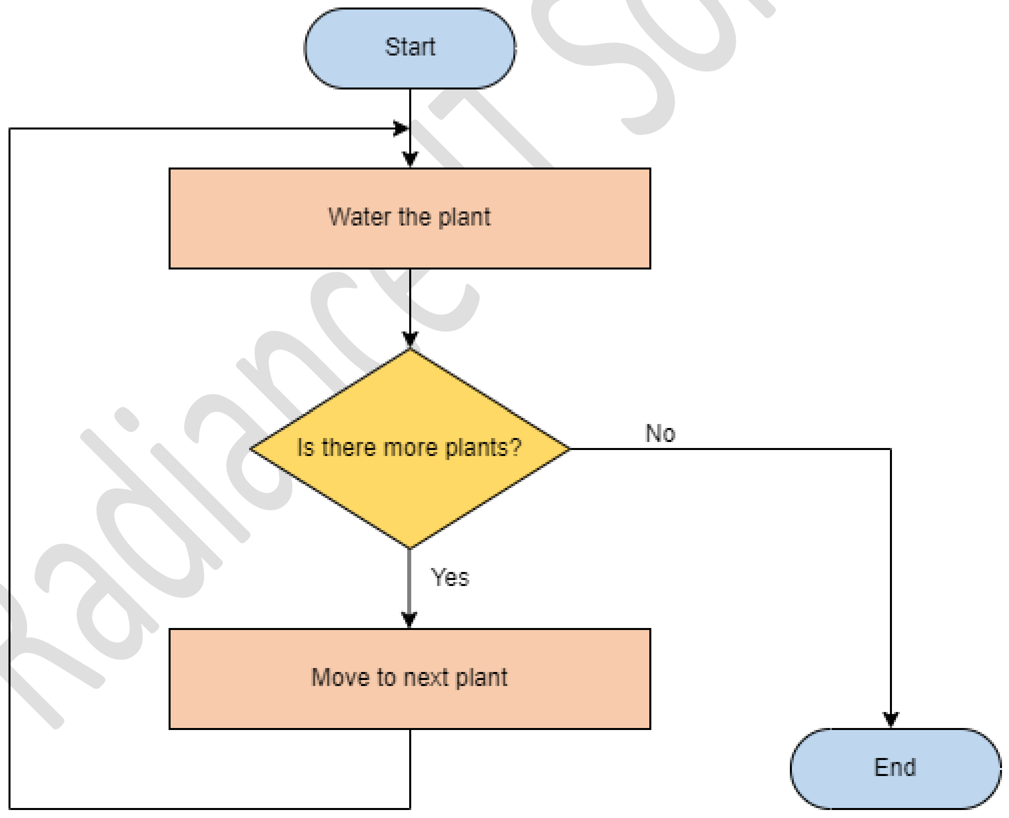


3. Repetition:
- Symbol: Represented by an oval or rounded rectangle.
- Concept: To showcase loops or repeated actions, emphasizing the cyclical nature.

User Case: Watering Plants

- Steps:
  1. Start at the First Plant: Water the first plant.
  2. Decision Point:
     - More Plants?
       - Yes: Move to the next plant and repeat.
       - No: Finish.

- Concept Explanation: The repetition occurs as you water each plant in a loop until there are no more plants to water. This cyclical process emphasizes the repeated action of watering each plant.


Start
Water the plant
Is there more plants?
No
Yes
Move to next plant
End