

# Problem Solving

**Problem solving** in **programming** is the process of analyzing a problem, designing appropriate algorithms to solve it, and then implementing these algorithms using a programming language to create a program that achieves the desired result.

## Steps Problem Solving

- 1- determine your problem.
- 2- Decomposition your problem.
  - Break a complex problem down into smaller, more easily solved problems.

### **How do I use it?**

- Define the main tasks in your project.
- Divide each task into smaller subtasks.
- Continue breaking it down until each task is directly executable.

**Example:** For a store management program:

- Divide it into: Product Management, Customer Management, Sales Management.
- Divide product management into: Add Product, Delete Product, Edit Product.

### 3- Algorithmic Thinking

- After you divided tasks to subtasks you Design a series of logical steps to solve it.

#### How do I use it?

- Start by defining the required inputs and outputs.
- Think about the steps necessary to transform the inputs into outputs.

## 4- Abstraction




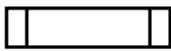
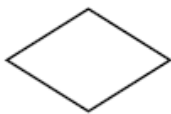





Focus on only the necessary information and ignore the unimportant details.

## 5- Debugging

- The process of identifying and fixing errors in code.

## 6- Tools

### Flowchart symbols

Symbol	Description
	Start/End (Terminator): represents the starting or ending point of the system.
	Input and Output data (Parallelogram): represents information entering or leaving the system. An input might be an order from a customer. Output can be a product to be delivered.
	Process (Rectangle): A box indicates some particular operation
	Pre-defined Process (Double-side Rectangle): A box indicates some particular predefined operation
	Decision/Condition (Diamond): represents a decision or branching point. Lines coming out from the diamond indicates different possible situations, leading to different sub-processes.
	Case (data): represents complement condition with switch.
	Flow (Line/Arrow): represents the flow of the sequence and direction of a process.
	Join/connector (Circle): join different flowline.
	Off-page Join/Connector: connect flowchart portion on different page
	Comment (Note): represents a printout, such as a document or a report.

## Types of boxes used to make a flowchart

1. Terminal



2. Data



3. Process



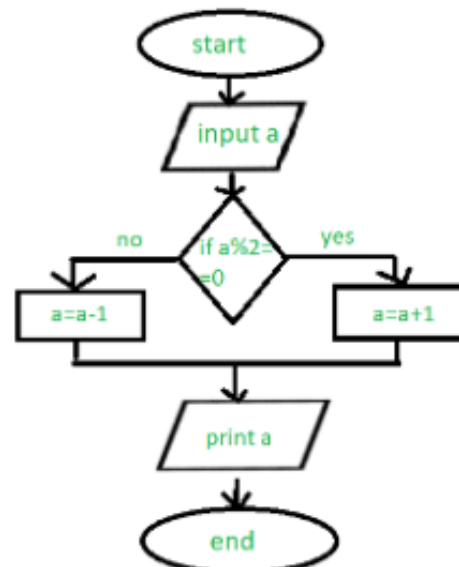
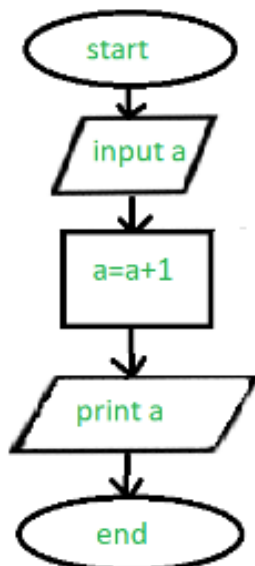
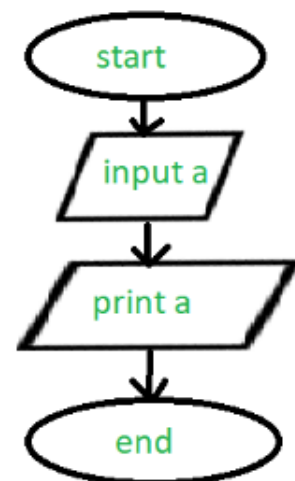
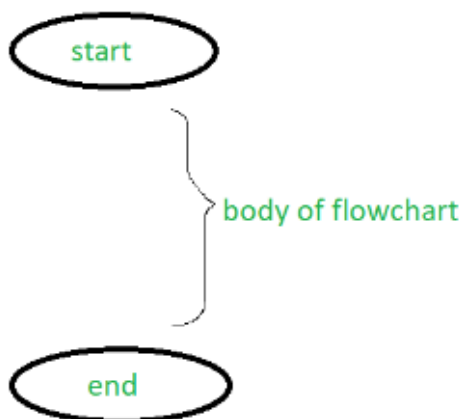
4. Decision



5. Flow



6. On-Page Reference



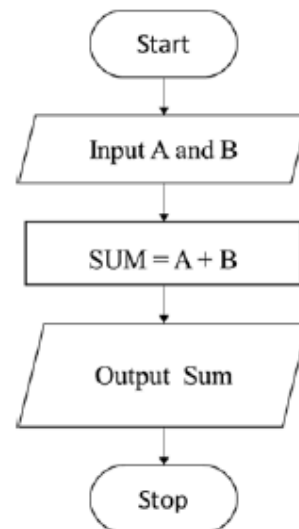
# Examples of flowchart

Write an algorithm and draw a flowchart to:

- **Example: Add two numbers**

**Algorithm**

1. Start
2. Input first number say A
3. Input second number say B
4.  $SUM = A + B$
5. Output or Display SUM
6. Stop

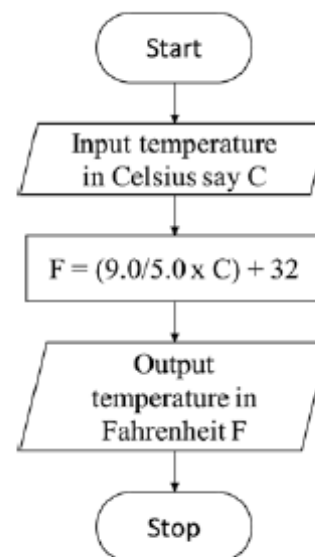


**Note:** sequential read/input could be merged into one step, i.e., steps 2 and 3 could be written as → “input values A and B”

- **Example: Convert temperature from Celsius to Fahrenheit**

**Algorithm**

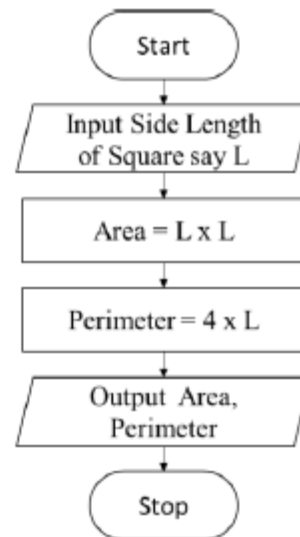
1. Start
2. Input temperature in Celsius say C
3.  $F = (9.0/5.0 \times C) + 32$
4. Output or Display Temperature in Fahrenheit F
5. Stop



- **Example: Find Area and Perimeter of Square**

**Algorithm**

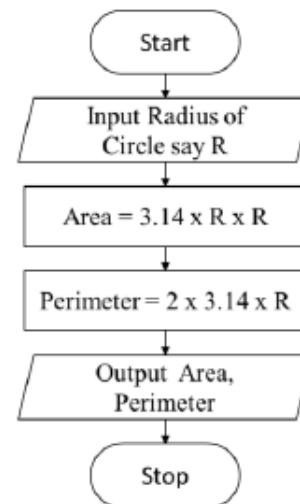
1. Start
2. Input Side Length of Square say L
3.  $\text{Area} = L \times L$
4.  $\text{Perimeter} = 4 \times L$
5. Output Area, Perimeter
6. Stop



- **Example: Find Area and Perimeter of a Circle**

**Algorithm**

1. Start
2. Input Radius of Circle say R
3.  $\text{Area} = 3.14 \times R \times R$
4.  $\text{Perimeter} = 2 \times 3.14 \times R$
5. Output Area, Perimeter
6. Stop



- **Tests:**

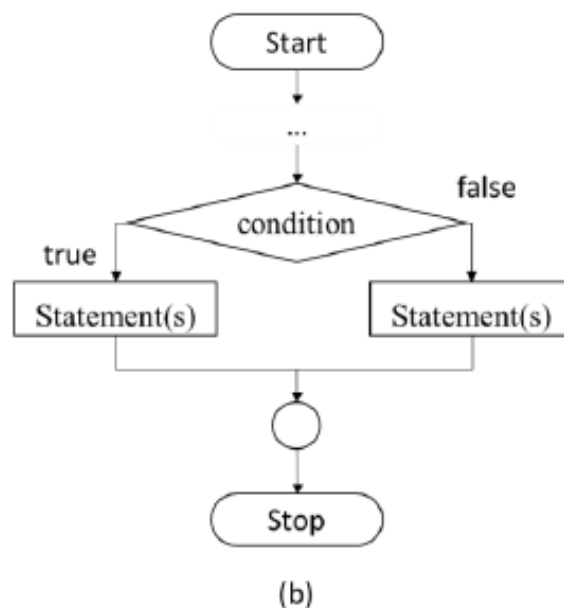
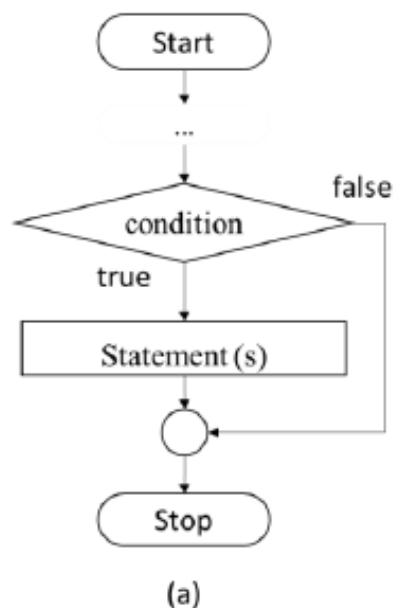
- a. **Find Area and Perimeter of a Triangle**
- b. **Swap Two Numbers using Temporary Variable**

## Problem-Solving: Conditional Statements (if, switch)

### 2.2.1 If Statement

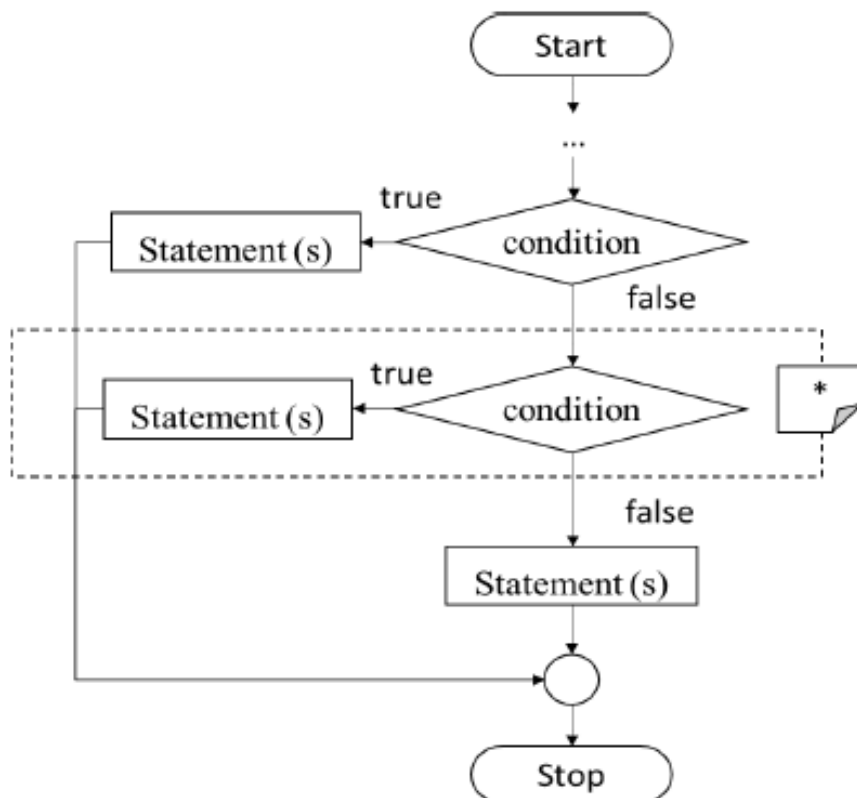
#### Types of if-statement

- Single condition (see figure (a))
  - a. If (condition)
    - i. Execution statement (s)
- Two conditions (see figure (b))
  - a. If (condition)
    - i. Execution statement (s)
  - b. Else
    - i. Execution statement (s)



- Multiple conditions (i.e., handle n cases) (see the following figure)
  - a. If (condition)
    - i. Execution statement (s)
  - b. [Else-if (condition)]
    - i. Execution statement (s)]\*  $\rightarrow$  n-2 cases
  - c. Else
    - i. Execution statement (s)

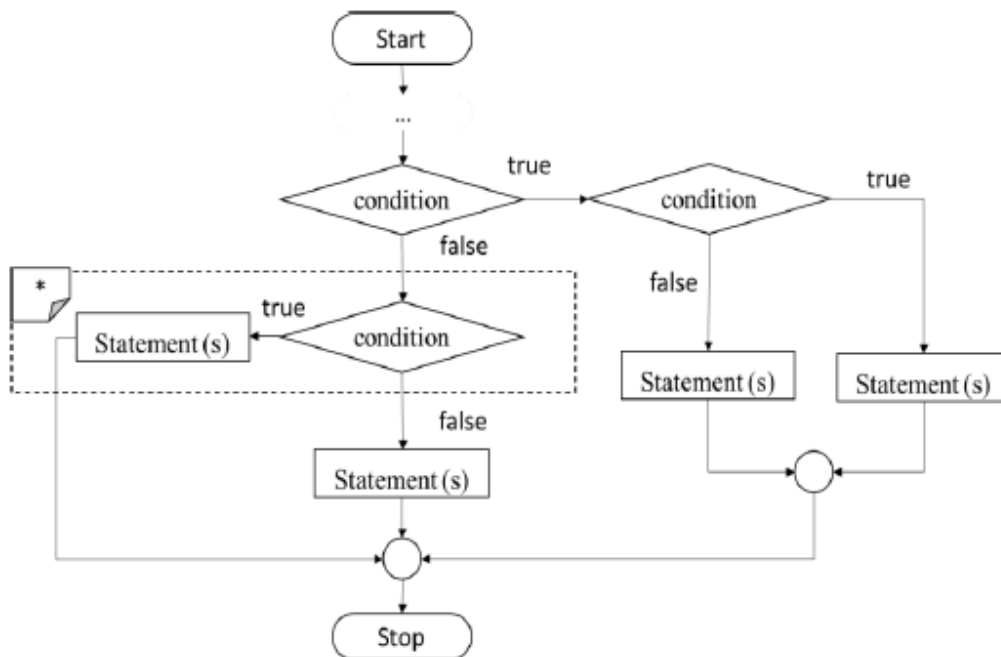
**Note:** Step (b) in the last type (compound if) could be repeated to handle n-2 cases





- Nested-if

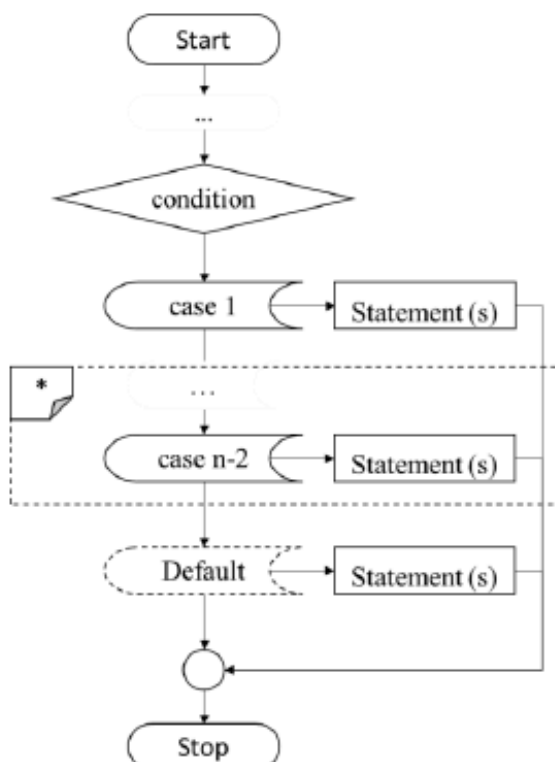
The nested if statement is used when a program requires more than one test expression. It is also called a multi-way selection statement. When a series of the decision are involved in a statement, we use if else statement in nested form (see the following figure).



### 2.2.2 Switch

Switch statement acts as a substitute for a long if-else-if that is used to test a list of cases.

A switch statement contains one or more case labels which are tested against the switch expression. When the expression match to a case then the associated statements with that case would be executed.

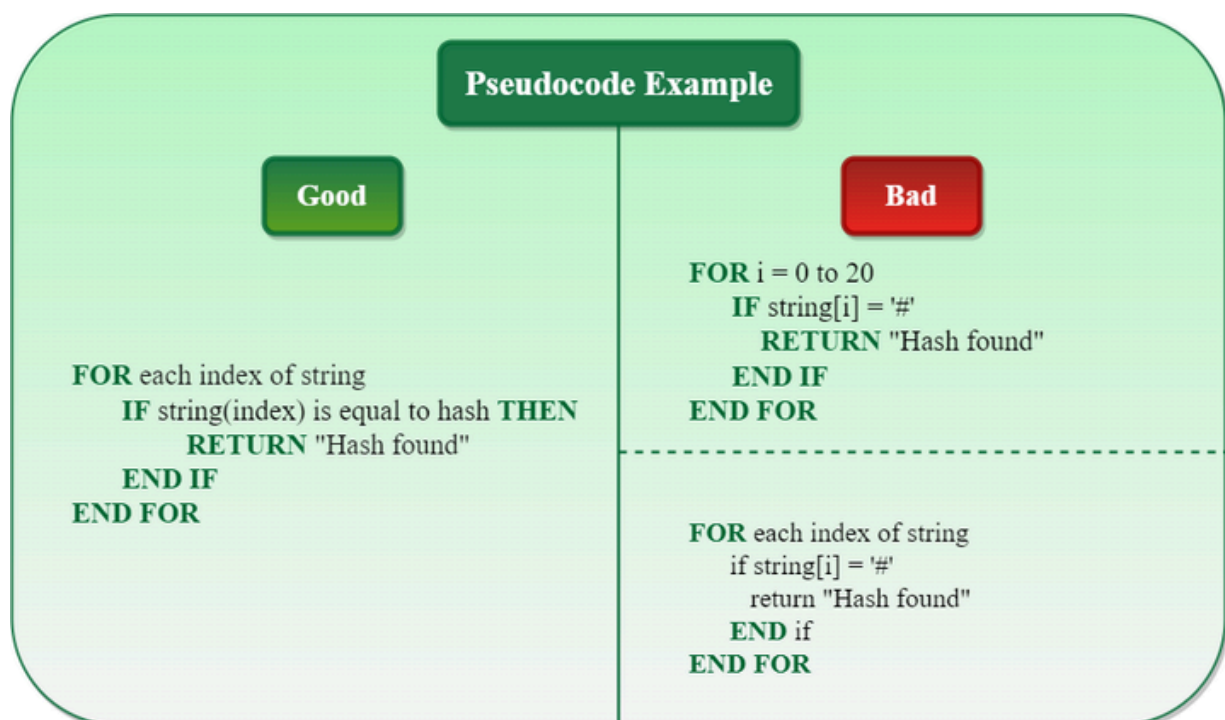


# What is PseudoCode

A **Pseudocode** is defined as a step-by-step description of an algorithm. Pseudocode does not use any programming language in its representation instead it uses the simple English language text as it is intended for human understanding rather than machine reading.

Pseudocode is the **intermediate state between an idea and its implementation(code)** in a high-level language.

## How to write a Pseudo-code?



THANK

YOU