# Algorithm, Pseudocode and Flowchart

Md. Aminul Islam Shazid

# Outline

1. Introduction
2. Algorithms
3. Flowcharts
4. Pseudocode
5. Control Structures
6. Exercises

# Introduction

# Introduction

- Algorithms, flowcharts, and pseudocode are essential tools for problem-solving
- They provide a bridge between problem analysis and actual programming
- This lecture introduces their concepts, notations, and best practices

# Algorithms

# What is an Algorithm?

- A step-by-step procedure to solve a problem
- Unambiguous and finite sequence of instructions
- Example: A recipe for cooking is an algorithm in real life

# Characteristics of a Good Algorithm

- Finiteness: must terminate after finite steps
- Definiteness: each step is clearly defined
- Input: specified set of inputs
- Output: specified set of outputs
- Effectiveness: steps can be performed with available resources

# Examples of Simple Algorithms

- Finding the maximum of three numbers
- Calculating factorial of a number
- Linear search in an array

# Example Algorithm: Factorial of a Number

- Input an integer, n
- Set result = 1
- While n is larger than 1:
  - result = result × i
  - n = n - 1
- Output the result

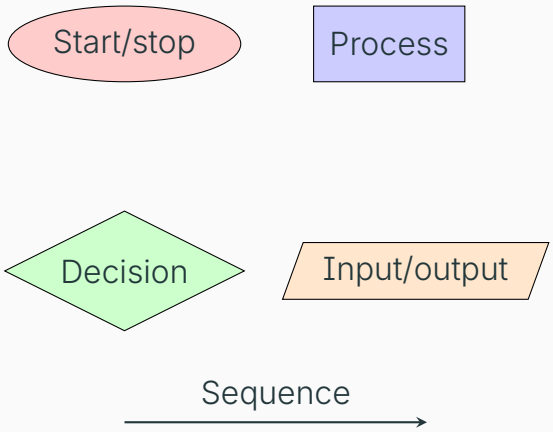
# Flowcharts

# Definition and Purpose

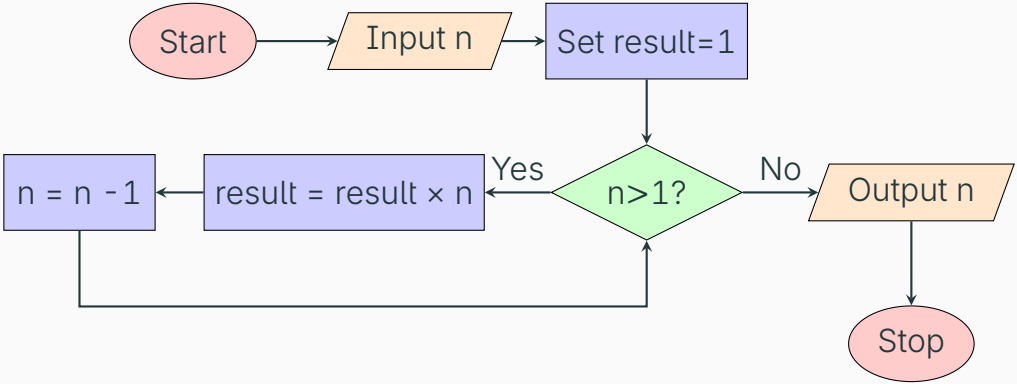
- Flowchart: graphical representation of an algorithm
- Uses standard symbols to show the flow of control
- Helps visualize program logic before coding

# Flowchart Shapes

- **Start/Stop**: ellipse
- **Process**: rectangle
- **Decision**: diamond
- **Input/Output**: parallelogram
- **Sequence**: arrow



# Example Flowchart: Factorial of a Number



# Pseudocode

# Purpose of Pseudocode

- Represents algorithms in structured, human-readable code
- Independent of programming language, but may include programming key-words
- Easier to understand and refine before coding

# Conventions

- Indentation to show structure
- Keywords like IF, WHILE, FOR
- Use natural language mixed with structured logic

# Example pseudocode: Factorial of a Number

```
Input n
Set result = 1
While n>1:
    result = result * n
    n = n - 1
EndWhile
Output n
```

# Control Structures

# Sequence

- Default mode of execution: step by step
- Example: Read number, calculate square, print result

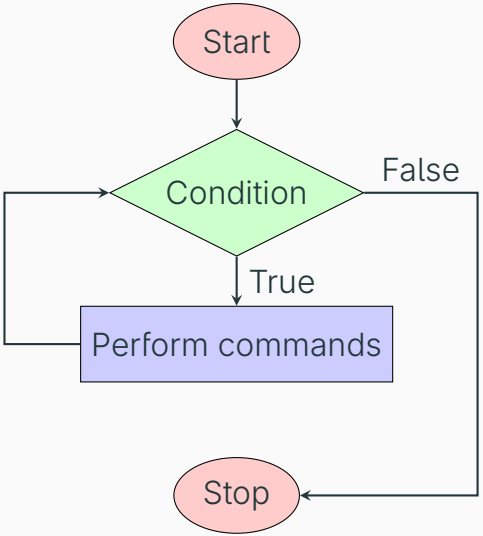
# Selection

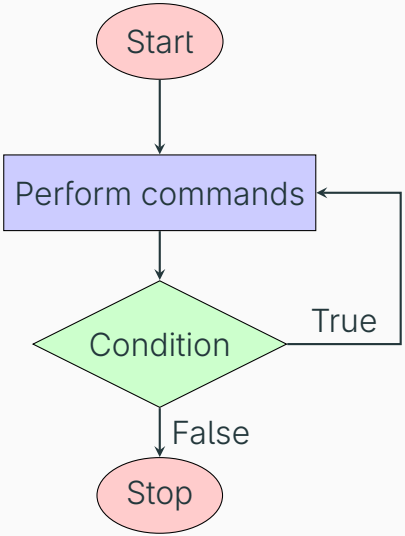
- **IF**: execute a block if condition is true
- **IF–ELSE**: choose between two alternatives
- **ELSE IF ladder**: multiple conditions

# Iteration

- **FOR loop** – fixed number of iterations
- **WHILE loop** – repeat while condition is true
- **DO–WHILE loop** – run at least once, then repeat if condition holds

# While vs Do-While



# Break and Continue

- **BREAK**: exit the nearest loop immediately (`exit` in Fortran)
- **CONTINUE**: skip rest of current iteration, proceed to next (`cycle` in Fortran)

# Recursion

- Function calling itself to solve smaller subproblems
- Example: factorial, Fibonacci
- Must have a base case to terminate

# Recursion Example: Factorial of a Number

```
Function Factorial(n):
    If (n==0):
        Return 1
    Else:
        Return n * Factorial(n-1)
    EndIf
EndFunction
```

# Best Practices

- Keep flowcharts clean and uncluttered
- Use consistent symbols and indentation
- Pseudocode should be language-independent
- Algorithms should be logically ordered and unambiguous

# Common Pitfalls

- Overcomplicating flowcharts with too many details
- Ambiguous pseudocode (mixing multiple languages)
- Ignoring edge cases in algorithms
- Writing unstructured logic

# Putting It All Together

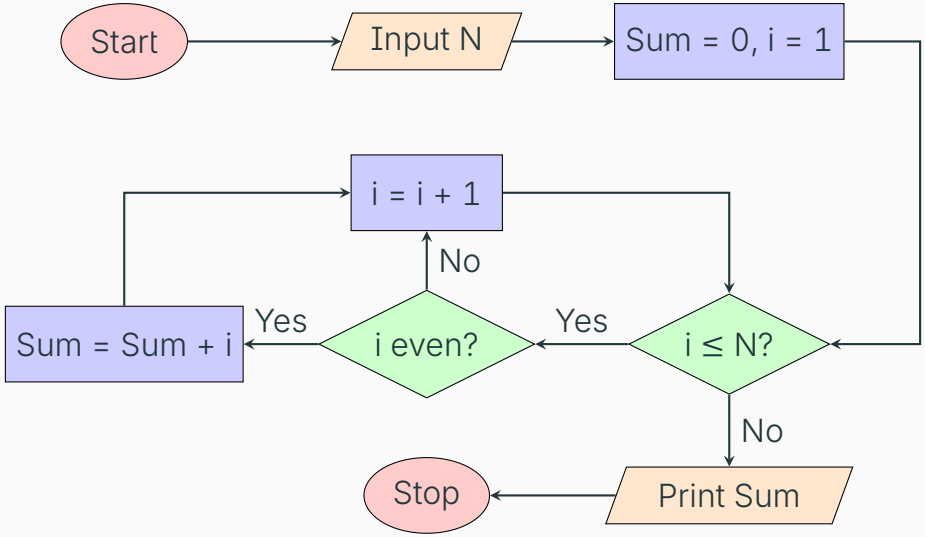
**Example task**

Compute the sum of all even numbers from 1 to N

# Algorithm

1. Read n
2. Set sum = 0, i = 1
3. While i <= n:
    - If i is even, add i to Sum, else do nothing
    - Add 1 to i
4. Print sum.

# Flowchart


Start
Input N
Sum = 0, i = 1
i = i + 1
No
Sum = Sum + i
Yes
i even?
Yes
i ≤ N?
No
Stop
Print Sum

# Pseudocode

```
Input n
sum = 0
i = 1
While (i <= n):
    If (i is even):
        sum = sum + i
    EndIf
    i = i + 1
EndWhile
Output sum
```

# Exercises

# Exercises

1. Design a algorithm and flow chart to find the largest of the three numbers
2. Develop pseudocode for computing the sum of the digits of a given integer
3. Write an algorithm and pseudocode to check whether a number is prime

# Questions?