

# Class Note: Introduction to Algorithms, Pseudocode, and Flowcharts (Section 1)

## Overview

This section introduces the foundational concepts of programming: **algorithms**, **pseudocode**, and **flowcharts**. These tools help students design solutions to problems before coding in C. As of May 15, 2025, this note emphasizes practical understanding through detailed examples and hands-on exercises, preparing students for real-world problem-solving using C programming.

## Learning Objectives

- Grasp the concept of an algorithm as a step-by-step problem-solving method.
- Learn to write pseudocode to plan program logic in a language-independent way.
- Understand and create flowcharts to visualize algorithmic processes.
- Gain experience through examples and exercises to apply these concepts effectively.

## Key Concepts

### 1. Algorithms

- **Definition:** A sequence of well-defined steps to solve a problem or complete a task.
- **Characteristics:**
  - Finite (ends after a specific number of steps).
  - Clear and unambiguous.
  - Effective (produces a correct result).
  - Generalizable (works for similar problems).
- **Importance:** Acts as a blueprint for programming, ensuring logical flow before coding.

## 2. Pseudocode

- **Definition:** A high-level, readable description of an algorithm, using plain English and simple programming constructs (e.g., IF , FOR , WHILE ).
- **Purpose:** Bridges the gap between human logic and computer code, making it easier to plan and debug.
- **Rules:**
  - Use indentation for readability.
  - Avoid specific syntax (e.g., no semicolons or braces).
  - Focus on logic over language details.

## 3. Flowcharts

- **Definition:** A graphical representation of an algorithm using standardized shapes.
- **Common Shapes:**
  - **Oval:** Start/End.
  - **Rectangle:** Process (e.g., calculation).
  - **Diamond:** Decision (e.g., yes/no condition).
  - **Arrow:** Flow direction.
- **Purpose:** Visualizes the flow of control, aiding in understanding and communication.

## Examples with Explanations

### Example 1: Algorithm, Pseudocode, and Flowchart - Sum of Two Numbers

#### Algorithm:

1. Prompt the user to enter two numbers.
2. Read the two numbers.
3. Calculate their sum.
4. Display the sum.
5. End.

#### Pseudocode:

```
BEGIN
    INPUT number1, number2
    SET sum = number1 + number2
    OUTPUT sum
END
```

### Flowchart:

[Start] --> [Input number1, number2] --> [sum = number1 + number2] --> [Output sum] --> [End]

### C Code (for Reference):

```
#include <stdio.h>
int main() {
    int num1, num2, sum;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    sum = num1 + num2;
    printf("Sum: %d\n", sum);
    return 0;
}
```

### Output (for inputs 5 and 3):

```
Enter two numbers: 5 3
Sum: 8
```

**Explanation:** This simple example shows how an algorithm is translated into pseudocode and a flowchart, then implemented in C. The flowchart uses a linear flow, ideal for straightforward tasks.

## Example 2: Algorithm, Pseudocode, and Flowchart - Check Even or Odd

### Algorithm:

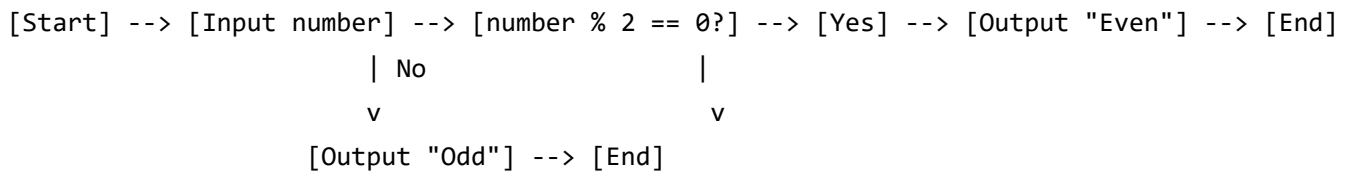
1. Prompt the user to enter a number.
2. Read the number.
3. If the number is divisible by 2 (no remainder), it's even; otherwise, it's odd.
4. Display the result.

5. End.

### Pseudocode:

```
BEGIN
    INPUT number
    IF number % 2 == 0 THEN
        OUTPUT "Even"
    ELSE
        OUTPUT "Odd"
    ENDIF
END
```

### Flowchart:



### C Code (for Reference):

```
#include <stdio.h>
int main() {
    int number;
    printf("Enter a number: ");
    scanf("%d", &number);
    if (number % 2 == 0) {
        printf("Even\n");
    } else {
        printf("Odd\n");
    }
    return 0;
}
```

### Output (for input 6):

```
Enter a number: 6
Even
```

**Explanation:** Introduces a decision point in the flowchart (diamond shape), reflecting the `if-else` logic. The pseudocode uses a conditional structure, making it easy to visualize the flow.

## Example 3: Algorithm, Pseudocode, and Flowchart - Sum of First N Numbers

### Algorithm:

1. Prompt the user to enter a positive number N.
2. Read N.
3. Initialize a sum to 0.
4. Add numbers from 1 to N to the sum.
5. Display the sum.
6. End.

### Pseudocode:

```
BEGIN
    INPUT n
    SET sum = 0
    FOR i = 1 TO n DO
        SET sum = sum + i
    END FOR
    OUTPUT sum
END
```

### Flowchart:

```
[Start] --> [Input n] --> [sum = 0] --> [i = 1] --> [i <= n?] --> [No] --> [Output sum] --> [End]
                | Yes                                |
                v                                    v
            [sum = sum + i] --> [i = i + 1]
```

### C Code (for Reference):

```
#include <stdio.h>
int main() {
    int n, sum = 0;
    printf("Enter a positive number: ");
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        sum += i;
    }
    printf("Sum of 1 to %d: %d\n", n, sum);
    return 0;
}
```

**Output** (for input 5):

Enter a positive number: 5

Sum of 1 to 5: 15

**Explanation:** This example introduces a loop in the algorithm, pseudocode, and flowchart. The diamond checks the loop condition, and the process repeats until  $i > n$ .

## Example 4: Algorithm, Pseudocode, and Flowchart - Find Largest of Three Numbers

**Algorithm:**

1. Prompt the user to enter three numbers.
2. Read the three numbers.
3. Compare the numbers to find the largest.
4. Display the largest number.
5. End.

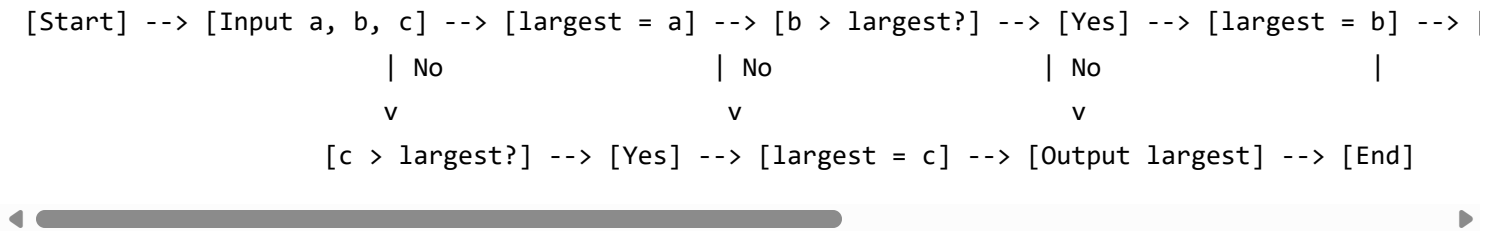
**Pseudocode:**

```

BEGIN
    INPUT a, b, c
    SET largest = a
    IF b > largest THEN
        SET largest = b
    ENDIF
    IF c > largest THEN
        SET largest = c
    ENDIF
    OUTPUT largest
END

```

### Flowchart:



### C Code (for Reference):

```

#include <stdio.h>
int main() {
    int a, b, c, largest;
    printf("Enter three numbers: ");
    scanf("%d %d %d", &a, &b, &c);
    largest = a;
    if (b > largest) largest = b;
    if (c > largest) largest = c;
    printf("Largest number: %d\n", largest);
    return 0;
}

```

### Output (for inputs 4, 9, 2):

```

Enter three numbers: 4 9 2
Largest number: 9

```

**Explanation:** Demonstrates multiple decision points to compare values, with the flowchart showing each comparison step clearly.

## Classwork

### Task 1: Write Pseudocode and Flowchart - Multiplication Table

- Write pseudocode and draw a flowchart to generate a multiplication table for a number input by the user (e.g.,  $5 \times 1$  to  $5 \times 10$ ).
- **Hint:** Use a loop to iterate from 1 to 10.

### Task 2: Convert Pseudocode to C - Average of Numbers

- Write pseudocode to calculate the average of 4 numbers input by the user.
- Convert the pseudocode into a C program and test it with inputs like 10, 20, 30, 40.

### Task 3: Create Flowchart and Pseudocode - Password Check

- Create a flowchart and pseudocode for a program that checks if a user-entered password matches a predefined password (e.g., "pass123"). Allow up to 3 attempts.
- **Hint:** Use a loop and a decision to track attempts.

### Task 4: Implement C Program - Grade Calculator

- Write pseudocode and a flowchart for a program that takes a score (0-100) and outputs a grade (A: 90+, B: 80-89, C: 70-79, D: 60-69, F: below 60).
- Implement the program in C and test with scores 95, 85, 75, 65, and 55.

### Task 5: Enhance Understanding with Complex Logic

- Write pseudocode and a flowchart for a program that finds the second largest number among five numbers input by the user.
- Convert it into a C program and test with inputs like 10, 45, 23, 67, 12.



# Real-World Applications

## Application 1: Daily Task Scheduler

**Context:** Plan daily tasks as of May 15, 2025.

**Pseudocode:**

```

BEGIN
    INPUT task
    IF task = "Study" THEN
        OUTPUT "Schedule: 2 hours at 11 AM WAT"
    ELSE IF task = "Exercise" THEN
        OUTPUT "Schedule: 1 hour at 6 PM WAT"
    ELSE
        OUTPUT "Task not scheduled"
    ENDIF
END
  
```

**Flowchart:**

```

graph TD
    Start([Start]) --> InputTask[Input task]
    InputTask --> TaskStudy{task = "Study"?}
    TaskStudy -- Yes --> OutputStudy[Output "Schedule: 2 hours at 11 AM WAT"]
    TaskStudy -- No --> TaskExercise{task = "Exercise"?}
    TaskExercise -- Yes --> OutputExercise[Output "Schedule: 1 hour at 6 PM WAT"]
    TaskExercise -- No --> OutputNotScheduled[Output "Task not scheduled"]
    OutputStudy --> End1([End])
    OutputExercise --> End2([End])
    OutputNotScheduled --> End3([End])
  
```

**C Code:**

```
#include <stdio.h>
int main() {
    char task[20];
    printf("Enter task (Study/Exercise): ");
    scanf("%s", task);
    if (strcmp(task, "Study") == 0) {
        printf("Schedule: 2 hours at 11 AM WAT\n");
    } else if (strcmp(task, "Exercise") == 0) {
        printf("Schedule: 1 hour at 6 PM WAT\n");
    } else {
        printf("Task not scheduled\n");
    }
    return 0;
}
```

**Output** (for input "Study"):

```
Enter task (Study/Exercise): Study
Schedule: 2 hours at 11 AM WAT
```

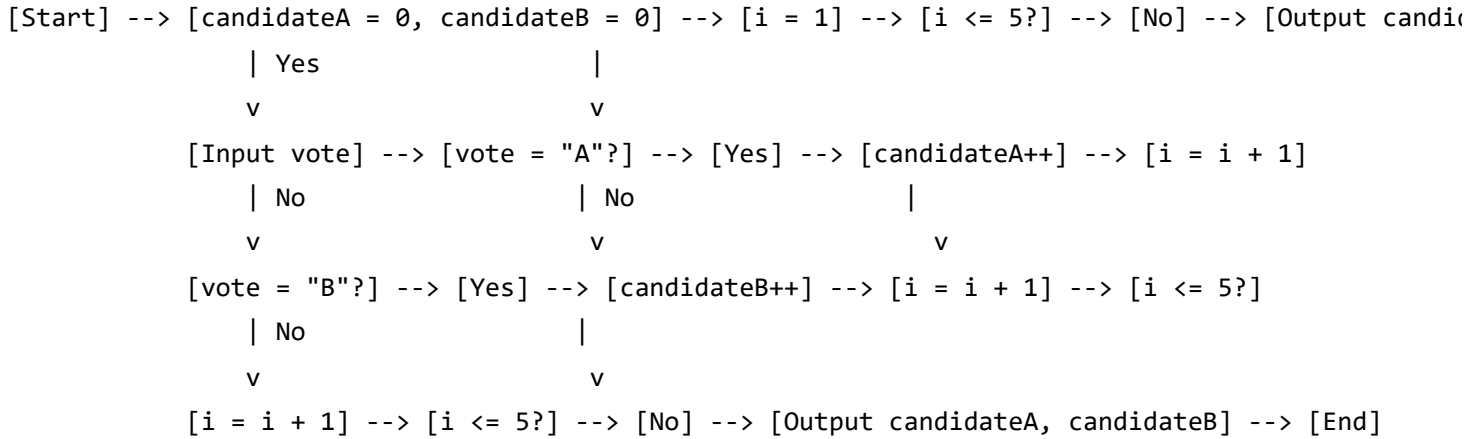
**Explanation:** Reflects current time (10:57 AM WAT) with a scheduling context.

## Application 2: Simple Voting System

**Context:** Count votes for two candidates.

**Pseudocode:**

```
BEGIN
    SET candidateA = 0, candidateB = 0
    FOR 5 times DO
        INPUT vote
        IF vote = "A" THEN
            INCREMENT candidateA
        ELSE IF vote = "B" THEN
            INCREMENT candidateB
        ENDIF
    END FOR
    OUTPUT "Candidate A: ", candidateA
    OUTPUT "Candidate B: ", candidateB
END
```

**Flowchart:****C Code:**

```

#include <stdio.h>
int main() {
    int candidateA = 0, candidateB = 0;
    char vote;
    for (int i = 1; i <= 5; i++) {
        printf("Enter vote (A/B) for vote %d: ", i);
        scanf(" %c", &vote);
        if (vote == 'A') {
            candidateA++;
        } else if (vote == 'B') {
            candidateB++;
        }
    }
    printf("Candidate A: %d\nCandidate B: %d\n", candidateA, candidateB);
    return 0;
}

```

**Output** (for votes A, B, A, A, B):

Enter vote (A/B) for vote 1: A  
Enter vote (A/B) for vote 2: B  
Enter vote (A/B) for vote 3: A  
Enter vote (A/B) for vote 4: A  
Enter vote (A/B) for vote 5: B  
Candidate A: 3  
Candidate B: 2

**Explanation:** Demonstrates loops and decisions, simulating a real voting process.

## Objective Questions

1. **What is an algorithm?**

- a) A programming language
- b) A step-by-step procedure to solve a problem
- c) A type of flowchart
- d) A compiled program

**Answer:** b) A step-by-step procedure to solve a problem

2. **Which shape in a flowchart represents a decision point?**

- a) Oval
- b) Rectangle
- c) Diamond
- d) Arrow

**Answer:** c) Diamond

3. **What is the purpose of pseudocode?**

- a) To run on a computer
- b) To plan and describe an algorithm in plain language
- c) To draw flowcharts
- d) To replace C code

**Answer:** b) To plan and describe an algorithm in plain language

4. **In the pseudocode IF  $x > y$  THEN OUTPUT "Greater" , what happens if  $x = 5$  and  $y = 3$ ?**

- a) Nothing
- b) Outputs "Greater"
- c) Error
- d) Outputs "Less"

**Answer:** b) Outputs "Greater"

5. Which of the following is a valid step in an algorithm?

- a) Compile the code
- b) Read two numbers
- c) Debug the program
- d) Execute the function

**Answer:** b) Read two numbers

## Additional Notes

- **Practice:** Draw flowcharts by hand or use tools like Lucidchart to visualize logic.
- **Tools:** Use paper or digital tools to sketch flowcharts alongside pseudocode.
- **Extension:** Explore more complex algorithms (e.g., sorting) in future sections.