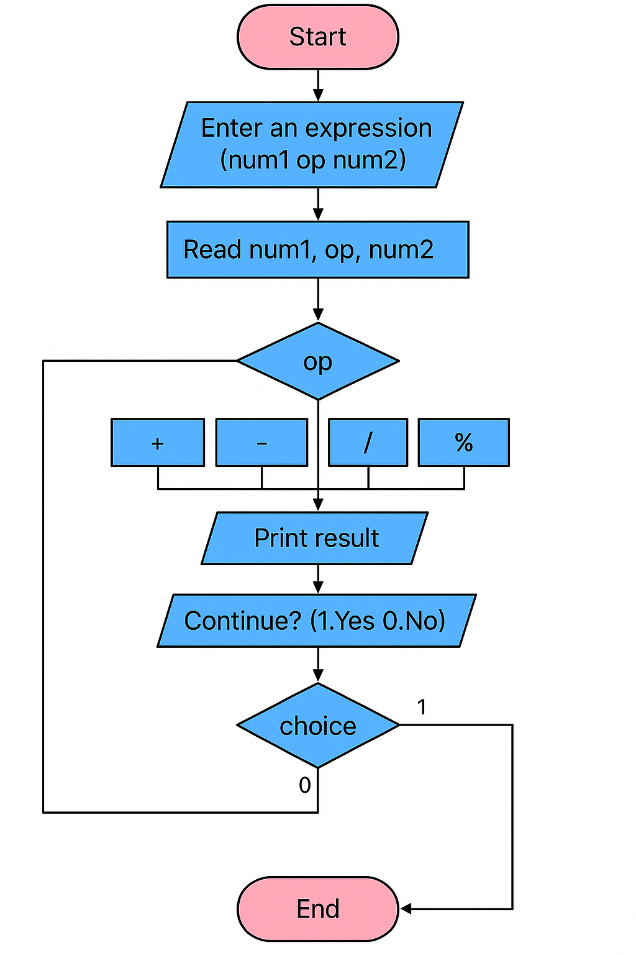
**1. Simple Calculator Program**

**Algorithm:**

1. Start
2. Declare num1, num2, choice, and op
3. Loop until the user chooses to quit:
   * Prompt user to enter an expression (num1 op num2)
   * Read the input values
   * Perform operation based on op:
     + + → Addition
     + - → Subtraction
     + \* → Multiplication
     + / → Division
     + % → Modulus
   * Print the result
   * Ask the user if they want to continue
4. If the user enters 0, exit the loop.
5. End

**Flowchart:**

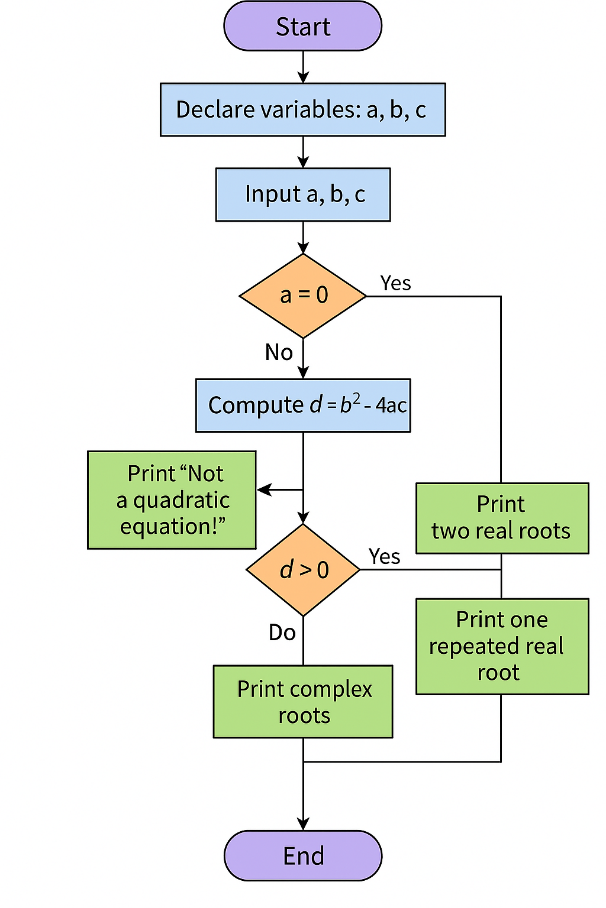
****

**2. Quadratic Equation Solver**

**Algorithm:**

1. Start
2. Declare variables: a, b, c, d
3. Prompt user to enter coefficients a, b, c
4. Read input values
5. If a == 0, print "Not a quadratic equation!" and exit
6. Compute the discriminant: d = b² - 4ac
7. If d > 0, compute and print two real roots
8. If d == 0, compute and print one repeated real root
9. If d < 0, compute and print complex roots
10. End

**Flowchart:**



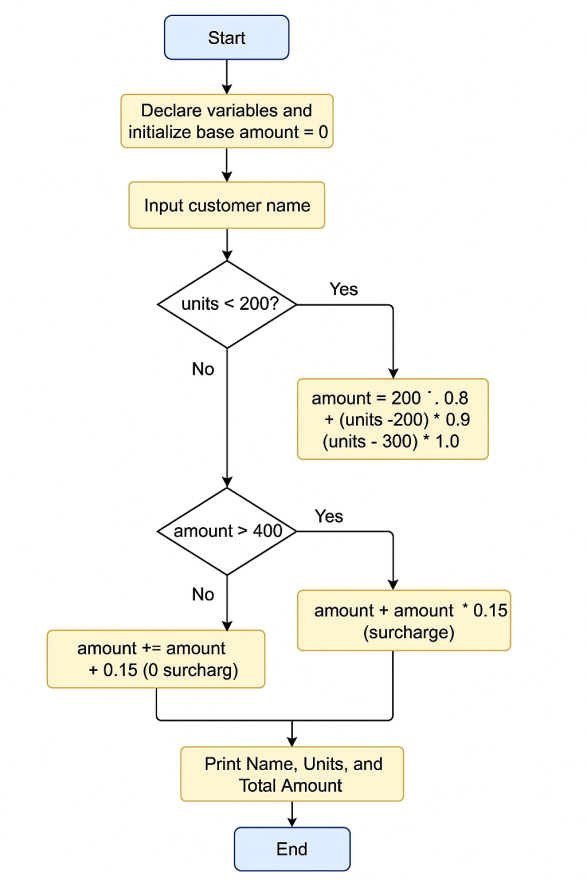
**3 .Electricity Bill**

An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit: for the next 100 units 90 paise per unit: beyond 300 units Rs 1 per unit. All users are charged a minimum of Rs. 100 as meter charge. If the total amount is more than Rs 400, then an additional surcharge of 15% of total amount is charged. Write a program to read the name of the user, number of units consumed and print out the charges

**Algorithm: Electricity Bill Calculation**

1. **Start**
2. Declare variables: name, units, bill
3. Set bill = 100 (minimum meter charge)
4. Prompt the user to enter name
5. Read the name
6. Prompt the user to enter units
7. Read the units
8. **If** units <= 200  
   → bill += units \* 0.80
9. **Else if** units <= 300  
   → bill += 200 \* 0.80 + (units - 200) \* 0.90
10. **Else**  
    → bill += 200 \* 0.80 + 100 \* 0.90 + (units - 300) \* 1.00
11. **If** bill > 400  
    → bill += bill \* 0.15 (add 15% surcharge)
12. Display the customer name, units, and final bill
13. **End**

**Flowchart:**



**4. Write a C Program to display the following by reading the number of rows as input,**

**1**

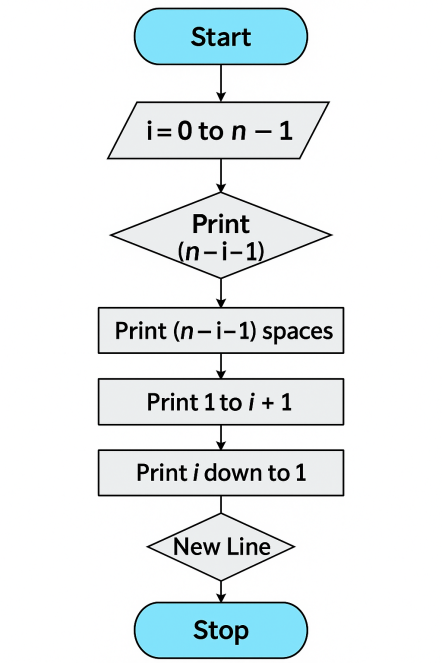
**1 2 1**

**1 2 3 2 1**

**1 2 3 4 3 2 1 --------------------------- nth row**

**Algorithm:**

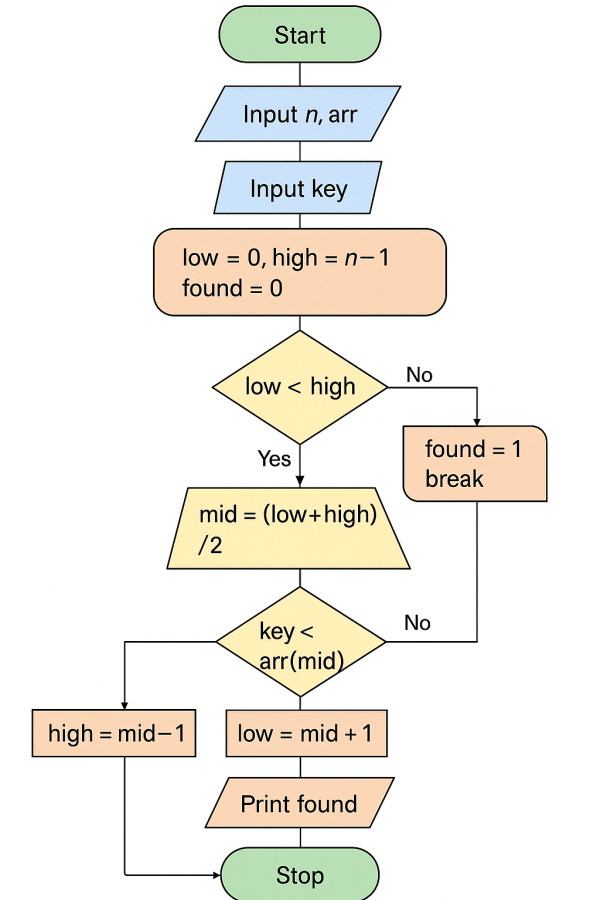
1. **Start**
2. **Input** number of rows n
3. **Repeat for** i = 0 to n - 1 (row index):
   * Print (n - i - 1) spaces
   * Print numbers from 1 to i + 1
   * Print numbers from i down to 1
   * Move to the next line
4. **Stop**



**5. Implement Binary Search on Integers.**

**Algorithm:**

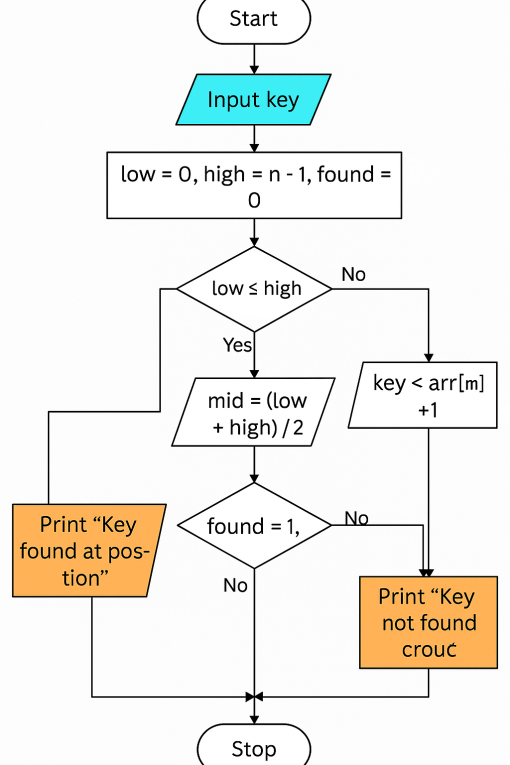
1. Start
2. Input the number of elements (n)
3. Input array elements in ascending order
4. Input key to be searched
5. Initialize low = 0, high = n - 1, found = 0
6. Repeat while low <= high
   * Calculate mid = (low + high)/2
   * If arr[mid] == key, set found = 1, break loop
   * Else if key < arr[mid], set high = mid - 1
   * Else set low = mid + 1
7. If found == 1, print position
8. Else, print "Key not found"
9. Stop



1. **Implement Matrix multiplication and validate the rules of multiplication.**

**Algorithm:**

1. Start
2. Input dimensions of Matrix 1 (m × n)
3. Input dimensions of Matrix 2 (p × q)
4. If n != p, print "Multiplication not possible", exit
5. Input elements of Matrix 1
6. Input elements of Matrix 2
7. Initialize result matrix to 0
8. For each row i in Matrix 1
   * For each column j in Matrix 2
     + For each element k, compute result[i][j] += mat1[i][k] \* mat2[k][j]
9. Print result matrix
10. Stop

****

1. **Compute Sine and Cosine of an Angle**

**Compute sin(x)/cos(x) using Taylor series approximation. Compare your result with the built-in library function. Print both the results with appropriate inferences**

**Algorithm**

Step 1: Start

Step 2: Input angle x in radians

Step 3: Initialize variables:

* sin = 0, cos = 0, term = 1
* n\_terms = number of terms for accuracy (e.g., 10)

Step 4: Compute sin(x) using Taylor series:

sin⁡(x)=x−x33!+x55!−x77!+…\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dotssin(x)=x−3!x3​+5!x5​−7!x7​+…

Step 5: Compute cos(x) using Taylor series:

cos⁡(x)=1−x22!+x44!−x66!+…\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dotscos(x)=1−2!x2​+4!x4​−6!x6​+…

Step 6: Compute sin(x)/cos(x) (i.e., tan(x)) using the approximated values

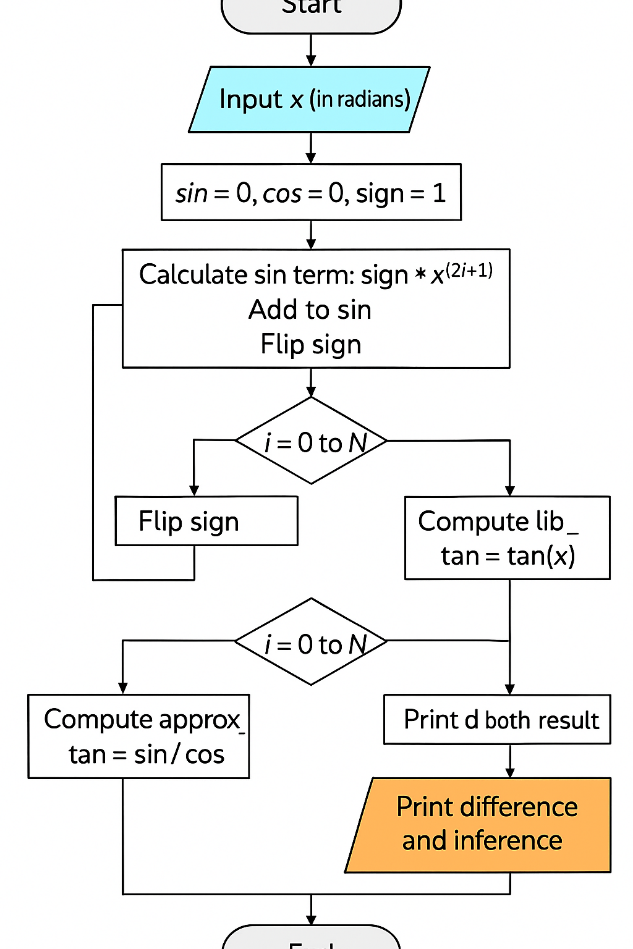
Step 7: Compute tan(x) using built-in function tan(x) from math library

Step 8: Display both results and compute the difference

Step 9: Print inference:

* If the difference is small, the approximation is accurate

Step 10: End

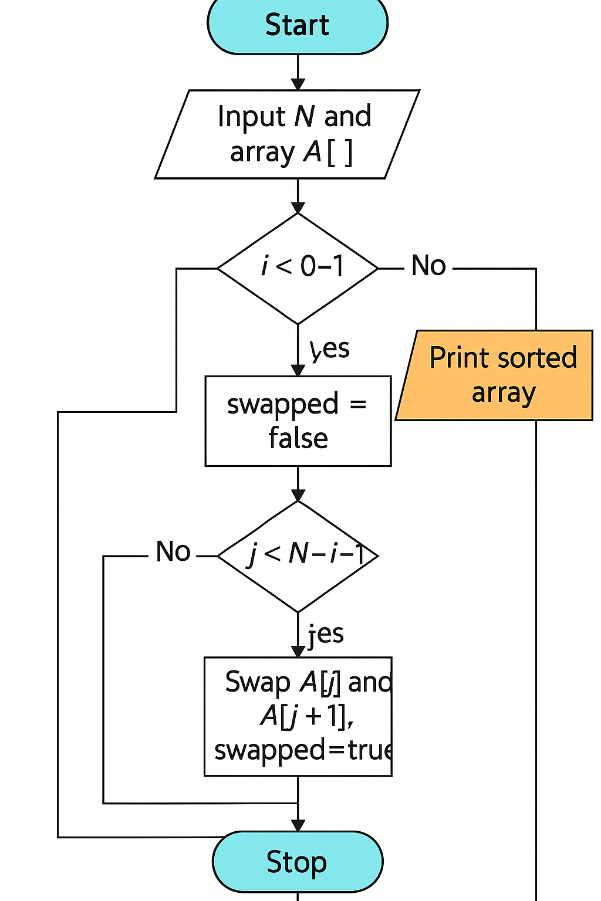
****

**8 Bubble Sort**

**Sort the given set of N numbers using Bubble sort**

### Algorithm: Bubble Sort

1. **Start**
2. Input the number of elements N and the array A[].
3. Repeat steps 4 to 7 for i = 0 to N - 2
4. Set swapped = false
5. Repeat steps 6 to 7 for j = 0 to N - i - 2
6. If A[j] > A[j + 1], then swap them and set swapped = true
7. If swapped == false, break the loop (array is sorted)
8. Print the sorted array
9. **Stop**

****

**9 String Operations**

**Write functions to implement string operations such as compare, concatenate, string length. Convince the parameter passing techniques.**

## Algorithm

### 1. **Start**

### 2. Input two strings: str1 and str2

### 3. Call function stringLength(str1)

→ Returns length of str1

### 4. Call function stringCompare(str1, str2)

→ Returns 0 if equal, positive if str1 > str2, negative if str1 < str2

### 5. Call function stringConcat(str1, str2)

→ Concatenates str2 to str1

### 6. Display all results

### 7. **Stop**

### C:\Users\admin\Downloads\ChatGPT Image May 27, 2025, 12_56_51 PM.png

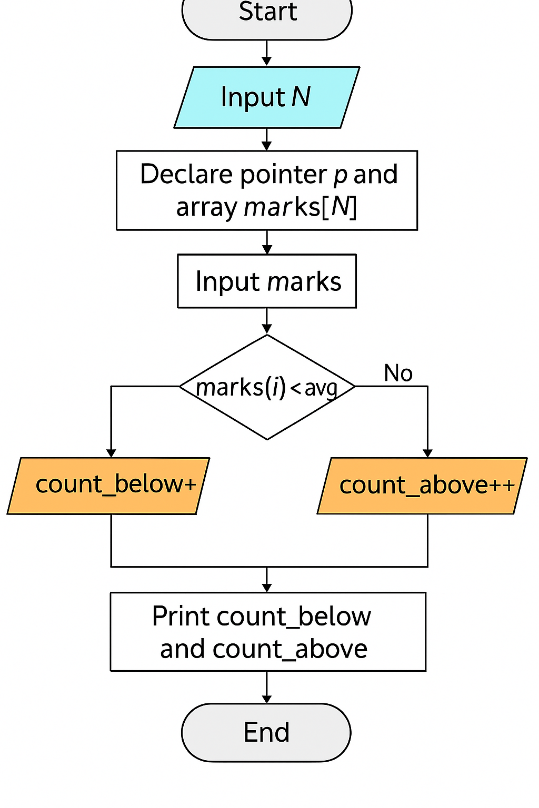
**10 C Structures**

**Implement structures to read, write and compute average- marks and the students scoring above and below the average marks for a class of N students.**

**Pointers and Arrays**

**Algorithm**

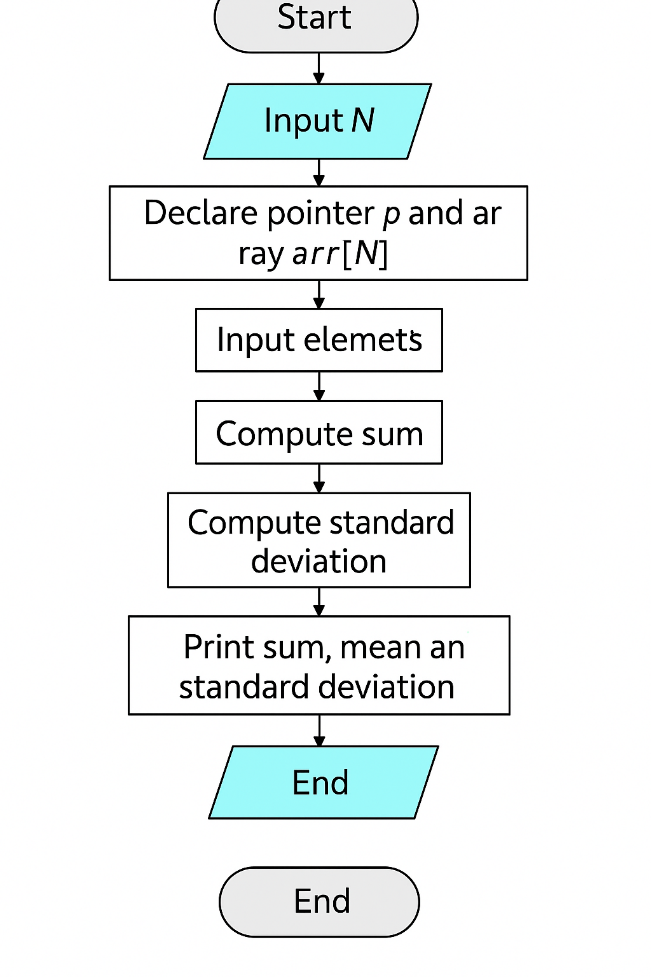
1. **Start**
2. Declare a structure Student with:
   * Name
   * Marks
3. Input total number of students N
4. Use an array of structures to store data for N students
5. Use a loop to:
   * Input name and marks for each student
   * Store in structure
6. Compute **total marks** using a pointer to the structure
7. Calculate **average = total / N**
8. Use another loop to:
   * Compare each student’s marks with average
   * Count and print students scoring **above** and **below** average
9. Print average and list of students above/below average
10. **Stop**

****

**11 Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers.**

**Algorithm**

1. **Start**
2. Input number of elements N
3. Declare array of size N and a pointer to float
4. Input N elements using pointer
5. Initialize sum = 0, loop through array using pointer to calculate sum
6. Calculate **mean = sum / N**
7. Use pointer to calculate:
   * sum of squares of differences from the mean
8. Compute **standard deviation** using:
   * sqrt(sum\_of\_squares / N)
9. Print sum, mean, and standard deviation
10. **Stop**

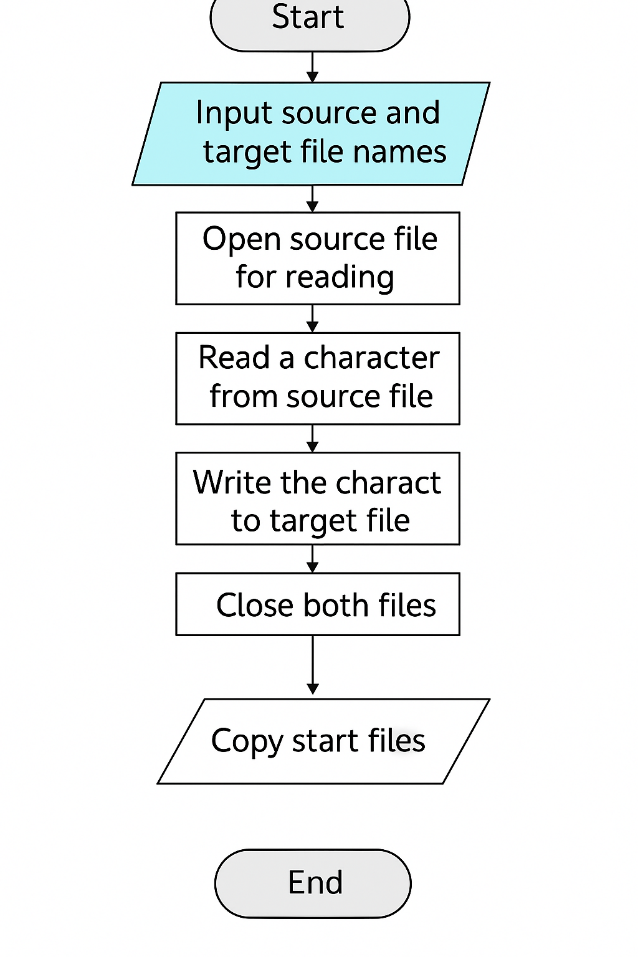
****

**12. File Copy**

**Write a C program to copy a text file to another, read both the input file name and target file name.**

**Algorithm**

1. **Start**
2. Input source (input) file name and target (output) file name
3. Open source file in **read** mode
4. Open target file in **write** mode
5. Check if source file opened successfully:
   * If not, display error and **stop**
6. Read contents of source file character by character
7. Write each character to the target file
8. Repeat until **end of file** is reached
9. Close both files
10. Display success message
11. **Stop**

****