01 Flowchart and Algorithm

Duration: 6 hours

I. INTRODUCTION

Overview

Before writing code, programmers must first plan how the program will work. Two tools commonly used for planning are flowcharts and algorithms. A flowchart is a visual representation of the logical steps in a program, while an algorithm is a written sequence of steps to solve a problem.

Relevance to C#

Learning how to break down problems into logical steps using algorithms and flowcharts helps students write structured, logical C# programs.

Learning Objectives:

1. Understand what flowcharts and algorithms are.
2. Identify the basic flowchart symbols.
3. Apply problem-solving skills before coding.

II. SOFTWARE AND TOOLS REQUIRED

Tool	Purpose
Pen and Paper / Diagram Software (e.g., draw.io, Lucidchart)	For drawing flowcharts
.Word Processor / Text Editor	For writing algorithms

III. LABORATORY EXERCISES / PROCEDURES

Activity 1: Identify Flowchart Symbols

Draw and label the following flowchart symbols:

- Start/End

- Process

- Input/Output

- Decision

- Arrows (Flowlines)

Activity 2: Create an Algorithm

Create an algorithm to check if a number is positive, negative, or zero.

Activity 3: Create a Flowchart

Use the algorithm from Activity 2 and draw its flowchart.

Activity 4: Write an Algorithm for a Real-World Scenario

Write an algorithm that describes how to make a cup of coffee.

Activity 5: Basic Input Validation

Draw a flowchart for a program that asks the user to enter their age. If the entered age is less than 0 or greater than 120, display an error message 'Invalid Age' and ask them to re-enter until a valid age is provided.

Activity 6: Division by Zero Protection

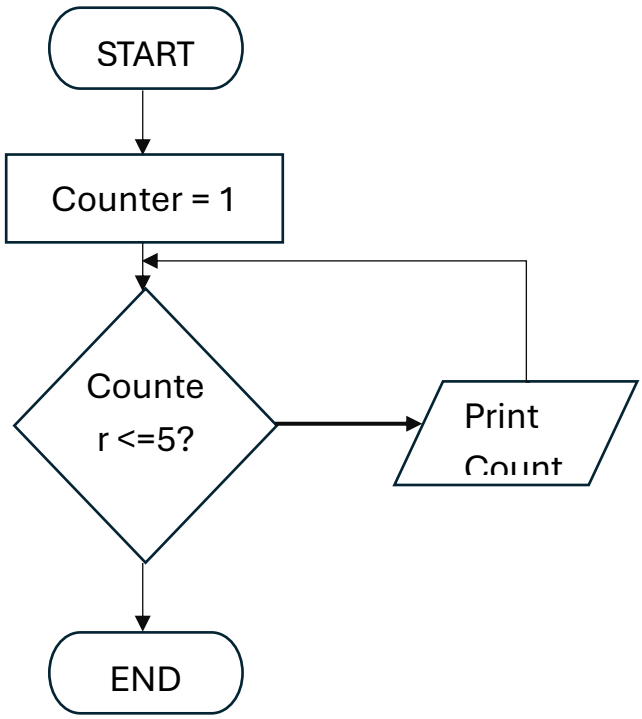
Draw a flowchart for a program that asks the user for two numbers, A and B. It then calculates and displays A divided by B. However, if B is 0, it should display an error message 'Cannot divide by zero' and prevent the calculation.

Activity 7: Debugging a Flawed Flowchart

Examine the flowchart .

1. Trace its execution mentally or on paper with an initial Counter = 1.
2. Identify any logical errors or missing components.
3. Describe the exact problem(s) you find.

4. Redraw the corrected flowchart.



IV. OUTPUT AND OBSERVATIONS

Algorithm: Average of Three Grades

Step	Description
1	Start
2	Input three test scores
3	Compute the average
4	If average ≥ 75, then print "Passed"
5	Else, print "Failed"
6	End

Test Cases Table

Input	Expected Output	Observations
80, 85, 90	Passed	
0, 60, 65	Failed	

V. ANALYSIS AND INTERPRETATION

1. Why is it important to plan a program before writing code?

   _____
   _____
   _____
   _____

2. What are the advantages do flowcharts offer over plain text?

   _____
   _____
   _____
   _____

3. Can you use these planning techniques for complex programs in C#

   _____
   _____
   _____
   _____

4. What do you think will be the most challenging part of converting a flowchart into code?

   _____
   _____
   _____
   _____

VI. DISCUSSION

1. Common Mistakes:

   _____
   _____
   _____
   _____

2. Real-World Applications:

   _____
   _____

_____

_____

VII. CONCLUSION

1. Summary of Skills Learned:

_____

_____

_____

_____

_____

_____

_____

_____

_____

2. Personal Insights:

_____

_____

_____

_____

_____

_____

_____

_____

_____

VIII. REFERENCES

- GeeksforGeeks. (n.d.). *An introduction to flowcharts*. Retrieved [Insert Date You Accessed] from https://www.geeksforgeeks.org/an-introduction-to-flowcharts/
- Minia University. (n.d.). *ALGORITHM & FLOWCHART MANUAL for STUDENTS*. Retrieved [Insert Date You Accessed] from https://courses.minia.edu.eg/Attach/16036flowchart-algorithm-manual.pdf
- Rebus Press. (n.d.). *Flowcharts – Programming Fundamentals*. In *Programming Fundamentals*. Retrieved [Insert Date You Accessed] from https://press.rebus.community/programmingfundamentals/chapter/flowcharts/