

Learning Objectives

- Explain the concept of pattern recognition and identify repeated designs, sequences, and computational processes in everyday life and software.
- Explain why pattern recognition enhances algorithmic efficiency (reducing redundant code, saving development time, single-point testing, optimizing memory).
- Define a sub-routine as a named, self-contained algorithm that can be called multiple times from the main program or across multiple programs.
- Identify, draw, and interpret the standard Predefined Process flowchart symbol (rectangle with double vertical side bars).
- Trace algorithms with sub-routines step-by-step using input data, tracking calls, jumps, execution flow, and returns.
- Analyze and modify nested sub-routines and apply a structured 4-step debugging protocol to fix common errors.

Question and Answers

Q1: What is pattern recognition in computer science?

Answer: Pattern recognition is the computational thinking is a process of identifying recurring sequences, designs, or repeated processes in a problem to simplify and optimize solutions.

Q2: What is a sub-routine and why is it useful?

Answer: A sub-routine is a self-contained, named algorithm that performs a specific task. It is useful because it can be written once, tested thoroughly, and reused multiple times across one or more algorithms.

Q3: What happens when a program 'calls' a sub-routine?

Answer: The main algorithm pauses its current execution, jumps to the designated sub-routine, runs all instructions inside it, and upon reaching STOP/RETURN, jumps back to resume the main algorithm.

Q4: Why are brackets () used after a sub-routine identifier (e.g., addNumbers())?

Answer: The brackets indicate that the identifier refers to a callable sub-routine/function rather than a standard variable, and they represent parameter containers in programming.

Q5: What is the flowchart symbol for a sub-routine?

Answer: A rectangle with vertical lines on the left and right inner sides (the Predefined Process symbol).

Q6: How do song lyrics illustrate the real-world concept of a sub-routine?

Answer: Song lyrics frequently print '[Repeat Chorus]' rather than reprinting the entire chorus lines. The chorus acts as a reusable sub-routine called multiple times between unique verses.

Q7: What is a nested sub-routine?

Answer: A nested sub-routine occurs when one sub-routine calls another sub-routine during its execution (e.g., findLargest() calling firstSmall()).

Q8: What are the 4 common errors found in sub-routine flowcharts?

Answer: Some of the common errors found in sub-routine flowcharts are:

- (1) Using wrong comparison operators,
- (2) Using wrong identifier names,
- (3) Misplacing the call symbol, and
- (4) Trying to access data/variables that do not exist in that sub-routine's scope.

Q9: List the steps to debugging a flawed flowchart, follow this systematic iterative cycle:

Answer: The following is a 4 step error correction cycle to be followed.

Step 1: Test the flowchart with sample data to observe what it actually outputs vs. what it should output.

Step 2: Modify the specific flawed part of the flowchart.

Step 3: Retest the flowchart with the same data to verify if the issue is resolved.

Step 4: Repeat the cycle until zero defects remain.

4. Second-Order Thinking & Analytical Scenarios

SO-Q1 [Software Maintenance & Refactoring]: Scenario Analysis: Imagine an online banking portal that calculates currency exchange fees.

In System A, the 20-line exchange calculation is copy-pasted into 40 different transaction scripts.

In System B, the calculation is placed into a single sub-routine `calculateExchangeFee()` called by all 40 scripts.

When government changes the tax exchange changes from 0.5 to 0.7, evaluate the maintenance impact and bug risk on both systems.

System A Evaluation: Developers must locate and modify 40 separate scripts. If they miss one or make a typo in three, the banking system will calculate inconsistent fees, causing accounting errors, security vulnerabilities, and massive testing overhead.

System B Evaluation: Developers update the formula in exactly one place (inside `calculateExchangeFee()`). All 40 transaction processes instantly inherit the correct formula with zero chance of inconsistent divergence, proving the power of modular software engineering.

SO-Q2 [Algorithmic Tracing & Edge Cases]: Scenario Analysis: An algorithm calls sub-routine `calculate()` which inputs three numbers and computes `result = first * second * third`.

If the user inputs `1, 2, 3`, what is the exact string output?

If the user later enters `0, 5, 10`, what will the output be?

Input 1, 2, 3: The sub-routine performs `1 * 2 * 3 = 6` and outputs: `1 * 2 * 3 = 6`.

Input 0, 5, 10: The sub-routine performs `0 * 5 * 10 = 0` and outputs: `0 * 5 * 10 = 0`.

5. Key Vocabulary Glossary

Key Term	Textbook Definition & Classroom Context
Pattern	A repeated design, sequence, or recurring process.
Pattern Recognition	The act of identifying recurring sequences or structures to simplify problem solving.
Sub-routine	A self-contained algorithm with an identifier that can be used multiple times in a program or across different algorithms.
Identifier	The unique, meaningful name given to a sub-routine (e.g., addNumbers()).
Call	The action of triggering a sub-routine from another flowchart or script, temporarily pausing main execution.
Return	The action of jumping back to the calling flowchart after the sub-routine reaches STOP/RETURN.
Efficient	Achieving maximum productivity with minimum wasted time, code duplication, or computer memory.
Predefined Process	The official flowchart symbol for a sub-routine (a rectangle with double vertical side borders).