[Concepts] Paper 2

CHAPTER 9: ALGORITHM DESIGN AND PROBLEM-SOLVING

9.1 COMPUTATIONAL THINKING SKILLS

9.1.1 Abstraction

Definition: The process of removing unnecessary details to focus on essential features.

Need:

- Manage complexity
- Create models of real-world systems

Benefits:

- Focus on what matters
- Reusable solutions

9.1.2 Decomposition

Definition: Breaking down complex problems into smaller, manageable sub-problems.

Benefits:

- Easier to solve smaller problems
- Can work on different parts independently
- Leads to modular programming (procedures/functions)

9.2 ALGORITHMS

9.2.1 Algorithm Representation

Methods:

- Structured English
- Flowchart
- Pseudocode

Components:

- Input
- Process
- Output

Basic Constructs:

- Sequence (order of execution)
- Selection (IF, CASE)
- Iteration (loops)

9.2.2 Pseudocode Guidelines

Input/Output:

```
<TEXT>
INPUT variable
OUTPUT variable/value
```

Assignment:

```
<TEXT>
variable ← value
```

Selection:

```
<TEXT>
IF condition THEN
 statements
ELSE
 statements
END IF

CASE OF variable
 value1 : statement
```

Iteration:

```
<TEXT>

FOR counter ← start TO end
 statements
NEXT counter

WHILE condition
 statements
END WHILE
```

9.2.3 Stepwise Refinement

```
 statements
UNTIL condition
```

Process:

- Start with general solution
- Gradually add detail
- Continue until can be programmed

CHAPTER 10: DATA TYPES AND STRUCTURES

10.1 DATA TYPES

10.1.1 Primitive Data Types

Type	Description
INTEGER	Whole numbers
REAL	Decimal numbers
CHAR	Single character
STRING	Sequence of characters
BOOLEAN	True/False
DATE	Date values

10.1.2 Records

Definition: A structure that holds a set of data of different types under one identifier.

Purpose:

- Group related data of different types
- Like a record in a database

Example:

<TEXT>

```
RECORD Employee
 DECLARE EmpID : INTEGER
 DECLARE Name : STRING
 DECLARE Salary : REAL
END RECORD
```

10.2 ARRAYS

10.2.1 One-Dimensional Arrays

Definition: A collection of elements of the same type accessed by index.

Terms:

- Index: Position of element
- Lower bound: First index (usually 0 or 1)
- Upper bound: Last index

Declaration:

<TEXT>

```
DECLARE arrayname[upperbound] : TYPE
```

Access:

<TEXT>

```
arrayname[index]
```

10.2.2 Two-Dimensional Arrays

Definition: Array of arrays; elements accessed by row and column.

Declaration:

<TEXT>

```
DECLARE arrayname[row, column] : TYPE
```

10.2.3 Array Operations

Linear Search:

- Check each element sequentially
- Works on unsorted data
- O(n) time complexity

Bubble Sort:

- Compare adjacent elements
- Swap if in wrong order
- Repeat until sorted
- O(n²) time complexity

10.3 FILES

10.3.1 File Need

- Store data permanently
- Large amounts of data
- Data persistence between program runs

10.3.2 File Operations

Opening:

```
<TEXT>
```

OPENFILE filename FOR READ/WRITE/APPEND

Reading:

```
<TEXT>
```

READFILE filename, variable

Writing:

```
<TEXT>
```

WRITEFILE filename, variable

Closing:

```
<TEXT>
```

CLOSEFILE filename

10.4 ABSTRACT DATA TYPES (ADT)

10.4.1 Stack

Characteristics:

- LIFO (Last In, First Out)
- Add to top (push)
- Remove from top (pop)

Operations:

- Push: Add item to top
- Pop: Remove item from top
- Peek: View top item without removing

10.4.2 Queue

Characteristics:

- FIFO (First In, First Out)
- Add to rear (enqueue)
- Remove from front (dequeue)

10.4.3 Linked List

Characteristics:

- Dynamic structure
- Nodes contain data and pointer to next node
- Can grow/shrink easily

Operations:

- Add node
- Delete node
- Search

CHAPTER 11: PROGRAMMING FUNDAMENTALS

11.1 PROGRAMMING BASICS

11.1.1 Variables and Constants

Variable Declaration:

```
<TEXT>
DECLARE variable : TYPE
```

Constant Declaration:

```
<TEXT>
CONSTANT name ← value
```

Assignment:

```
<TEXT>
variable ← value
```

11.1.2 Input/Output

Input:

```
<TEXT>
INPUT variable
```

Output:

```
<TEXT>
OUTPUT "message", variable
```

11.1.3 Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
DIV	Integer division
MOD	Remainder

11.1.4 Logical Operators

Operator	Description
AND	Both true
OR	Either true
NOT	Negation

11.2 CONSTRUCTS

11.2.1 Selection Statements

IF Statement:

<TEXT>

```
IF condition THEN
 statements
ELSE
 statements
END IF
```

Nested IF:

```
<TEXT>
IF condition1 THEN
 IF condition2 THEN
 statements
 ELSE
 statements
 END IF
ELSE
 statements
END IF
```

CASE Statement:

```
<TEXT>
CASE OF variable
 value1 : statement
 value2 : statement
 OTHERWISE : statement
END CASE
```

11.2.2 Loop Structures

Count-Controlled (FOR):

```
<TEXT>
FOR counter ← start TO end
 statements
NEXT counter
```

Pre-Condition (WHILE):

```
<TEXT>
WHILE condition
 statements
END WHILE
```

Post-Condition (REPEAT):

```
<TEXT>
REPEAT
 statements
```

Choice of Loop:

- FOR: When number of iterations known
- WHILE: When condition checked before first iteration
- REPEAT: When condition checked after at least one iteration

11.3 STRUCTURED PROGRAMMING

11.3.1 Procedures

Definition: A reusable block of code that performs a specific task.

Declaration:

```
<TEXT>
PROCEDURE name(parameter1, parameter2)
 statements
END PROCEDURE
```

Call:

```
<TEXT>
CALL name(argument1, argument2)
```

Parameters:

- Passed by value: Copy of data passed
- Passed by reference: Actual data passed (can be modified)

11.3.2 Functions

Definition: Returns a value; used in expressions.

Declaration:

```
<TEXT>
FUNCTION name(parameter) RETURN type
 statements
 RETURN value
```

Use:

```
<TEXT>
```

```
result ← name(arguments)
```

CHAPTER 12: SOFTWARE DEVELOPMENT

12.1 PROGRAM DEVELOPMENT LIFECYCLE

12.1.1 Stages

1. **Analysis:** Understand problem, identify requirements
2. **Design:** Create solution structure, algorithms
3. **Coding:** Write actual program code
4. **Testing:** Find and fix errors
5. **Maintenance:** Update and improve after delivery

12.1.2 Development Models

Waterfall:

- Sequential phases
- Each phase completes before next begins
- Good for well-defined requirements

Iterative:

- Develop in cycles
- Each iteration produces working version
- Allows feedback and improvement

RAD (Rapid Application Development):

- Quick development using pre-built components
- Emphasis on prototyping
- Fast user feedback

12.2 PROGRAM DESIGN

12.2.1 Structure Chart

Purpose: Decompose problem into sub-tasks

Shows:

- Modules/procedures/functions
- Parameters passed between modules

12.2.2 State-Transition Diagrams

Purpose: Document algorithm behavior

Shows:

- Different states
- Transitions between states

12.3 PROGRAM TESTING AND MAINTENANCE

12.3.1 Error Types

Type	Description
Syntax	Violation of language rules
Logic	Incorrect algorithm implementation
Runtime	Error during execution

12.3.2 Testing Methods

Method	Description
Dry Run	Manual execution using test data
Walkthrough	Team reviews code
White Box	Tests internal structure

Method	Description
Black Box	Tests functionality only
Integration	Test combined components
Alpha	Testing by developers
Beta	Testing by users
Acceptance	Final testing by client

12.3.3 Test Data

Type	Description
Normal	Typical valid data
Extreme/Boundary	At limits of valid range
Abnormal/Invalid	Invalid data

12.3.4 Maintenance Types

Type	Description
Corrective	Fixing errors
Adaptive	Adapting to environment changes
Perfective	Improving performance/functionality