NU BALIWAG

COURSE MATERIAL

Subject Code:	CCDATRCL
Course Title:	Data Structure and Algorithms
Course Description:	This course focuses on the use of data structures and algorithms for solving computing and real-life problems efficiently. This course covers the following data structures: dynamic arrays, lists, stacks, queues, trees, graphs and maps. Analysis of time and space complexity of algorithmic solutions will also be covered.
No. of Units:	3
Class Schedule:	

Course Learning Outcome:

At the end of the course, the student must be able to:

1. Compare and contrast the following data structures: graph, stack, queue, tree, sets, dynamic array, linked list, binary search tree, heap,hash table.
2. Create solutions to programming and real-life problems of varying complexities using data structures and algorithms discussed in class.
3. Apply time and space complexity analysis to programming problems.

About the Instructor :	**Samuel S. Espino Jr.** **Information Technology Department**
Contact Information :	ssespino@nu-baliwag@edu.ph / 09476772162

Topic(s) :	Overview of Data Structure and Algorithms
Table of contents :	Week 1

Overview of Data Structure and Algorithms

- Introduction to Data Structure and Algorithms
- Data Structures
- Algorithms
- Flowchart Visualization

- Why Are They Important?

I. Pre-test / Activity:

II. **Learning Outcomes:**

1. Define the role and importance of data structures in organizing and managing data efficiently.
2. Identify and classify different types of data structures: primitive, composite, and abstract.
3. Explain how algorithms provide systematic approaches to problem-solving and task execution.
4. Illustrate how algorithms are applied to process data and achieve desired outcomes in various applications.
5. Recognize the significance of efficient data structures and algorithms in software development.

III. Content:

Introduction to Data Structure and Algorithms

Data structures and algorithms are foundational concepts in computer science that are essential for efficient problem-solving and software development. Data structures provide ways to organize and store data for easy access and modification, while algorithms offer systematic approaches to process this data. Mastering these concepts allows developers to create efficient, scalable, and high-performing applications, making them crucial tools for tackling complex computational challenges.

Data refers to the collection of values or pieces of information that are stored and manipulated by a computer program. These values can represent a wide range of information types, such as numbers, characters, strings, and more complex entities like records, images, and graphs. The purpose of a data structure is to organize and manage these values in a way that enables efficient access, modification, and storage.

Types of Data

1. Primitive Data Types:
    - Integers: Whole numbers, such as 1, 2, 3.
    - Floats: Numbers with decimal points, such as 3.14, 2.718.
    - Characters: Single symbols, such as 'a', 'B', '3'.
    - Booleans: True or false values.

2. Composite Data Types:
    - Arrays: Collections of elements of the same type stored in contiguous memory locations.
    - Strings: Sequences of characters.

3. Abstract Data Types:
    - Lists: Ordered collections of elements, which can be of any type.
    - Trees: Hierarchical structures with nodes representing data elements.
    - Graphs: Collections of nodes (vertices) connected by edges.

Data Structures

Data structures are specialized formats for organizing, processing, retrieving, and storing data. They provide a means to manage large amounts of data efficiently for uses such as large databases and internet indexing services. Proper data structure selection is crucial for designing efficient algorithms and can greatly impact the performance and scalability of software systems. Examples include arrays, linked lists, stacks, queues, trees, graphs, hash tables, and more.

Key Characteristics of Data Structures

- Efficiency - They allow for efficient data retrieval, insertion, deletion, and modification.
- Flexibility - Some data structures are dynamic, allowing for flexible changes in size.
- Scalability - Efficient data structures support the scalability of applications as data grows.

Algorithms

Algorithms are step-by-step procedures or formulas for solving a problem or performing a task. They take input, process it through a series of computational steps, and produce an output. Algorithms are fundamental to computer science and software development, providing systematic approaches to problem-solving.

Key Characteristics of Algorithms:

- Definiteness - Each step of an algorithm is clearly defined.
- Input - Algorithms take zero or more inputs.

- Output - Algorithms produce one or more outputs.
- Finiteness - Algorithms terminate after a finite number of steps.
- Effectiveness - Each step of the algorithm must be basic enough to be carried out, in principle, by a person using only pencil and paper.

Let's create an algorithm to solve a classic problem: finding the factorial of a number. We'll introduce a flowchart to visualize the steps involved in the algorithm.

Finding the Factorial of a Number

The factorial of a non-negative integer nnn is the product of all positive integers less than or equal to n.

Example:

- 5!=5×4×3×2×1=120
- 5! = 5 \times 4 \times 3 \times 2 \times 1 = 120
- 5!=5×4×3×2×1=120

Algorithm to Find Factorial

Steps:

1. Initialize a variable to store the factorial result and loop.
2. Input the number n for which factorial is to be found.
3. Iterate from 1 to n.
    a. Multiply each iteration's value with the factorial variable.
4. Output the factorial value.

Flowchart Visualization

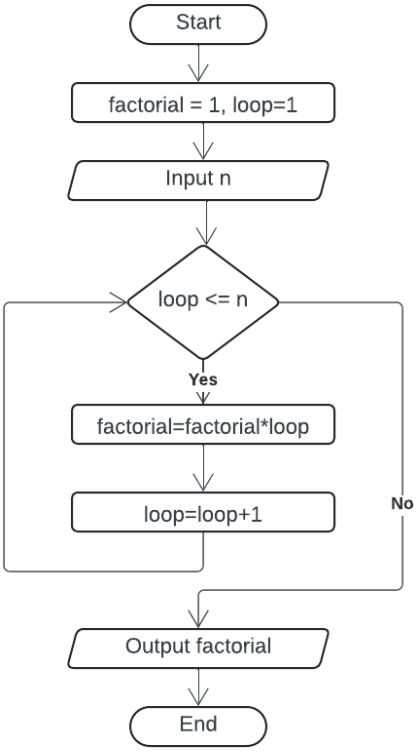
Flowcharts are diagrams that represent the step-by-step processes or algorithms. They use symbols connected by arrows to show the flow of control.

Explanation of Flowchart Symbols

- Start/End: Represents the beginning and end of the flowchart.

- Input/Output: Represents inputting or displaying data.
- Process: Represents a process or action, such as setting variables or performing calculations.
- Decision: Represents a conditional statement, directing the flow based on a condition (not used in this example).

Below is a flowchart illustrating the factorial algorithm.



Why Are They Important?

1. **Optimization of Performance:**
   - Efficient data structures and algorithms ensure that software applications run faster and use resources more effectively.
   - They help in reducing time complexity (how execution time increases with the size of the input) and space complexity (how memory usage increases with the size of the input).

2. **Fundamental for Solving Complex Computational Problems**
    - They enable the handling of large datasets and complex computations.
    - Used in a wide array of applications such as databases, artificial intelligence, networking, and operating systems.
    - Essential for developing scalable solutions that can grow with increasing data and user demands.

Examples of Data Structures and Algorithms

- **Data Structures**: Arrays, Linked Lists, Stacks, Queues, Trees (Binary Trees, Binary Search Trees, AVL Trees), Graphs, Hash Tables.
- **Algorithms**: Sorting algorithms (Quick Sort, Merge Sort), Searching algorithms (Binary Search), Graph algorithms (Dijkstra's Algorithm, Depth First Search, Breadth First Search), Dynamic Programming (Fibonacci sequence, Knapsack problem), Greedy algorithms (Coin Change problem).

Understanding and implementing the right data structures and algorithms is crucial for efficient problem-solving and optimal performance in software development.

IV. Post test:

- Activity: Create an algorithm to solve this problem and visualize it using a flowchart.

1. Checking if a given number is a prime number. A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself. 5 is a prime number because its only divisors are 1 and 5.
2. Finding the maximum number from 3 numbers inputted by the user. Display the maximum number.

V. References:

Wengrow, Jay 2017, A Common Sense Guide to Data Structures and Algorithms

https://www.techtarget.com/searchdatamanagement/definition/data-structure