College of Engineering & Applied Sciences

CSPB 2270

Data Structures

Class Notes

University Of Colorado

2024

Sections

Week 1

C++ Review, Debugging, Unit Testing

1.0.1 Activities

The following are the activities that are planned for Week 1 of this course.

- Take the C++ assessment
- Read the C++ refresher or access other resources to improve your skills (book activities are graded but the grades are not included in your final grade for this course)
- Read the zyBook chapter(s) assigned and complete the reading quiz(s) by next Monday
- Access the GitHub Classroom and get your Assignment-0 repository created, cloned, edited, and graded by next Tuesday
- Watch the videos for Cloning GitHub Classroom Assignments, Setting up an IDE in Jupytherhub, and Unit Testing

1.0.2 Lectures

Here are the lectures that can be found for this week:

- Course Concepts
- GitHub Classroom
- GitHub Security
- Accepting an Assignment
- Accessing Git Files
- Cloning Into JupyterHub
- VSCode in JupyterHub
- Multi File Programming
- Unit Testing Basics

1.0.3 Programming Assignment

The programming assignment for Week 1 - **Using GitHub and GitHub Classroom**.

1.0.4 Chapter Summary

The first chapter of this week was **Chapter 1 - Introduction to Data Structures**.

Section 1.1 - Data Structures

We define Data Structures to be the following:

- A data structure is a method of organizing, storing, and performing operations on data.
- Operations performed on data structures include accessing or updating stored data, searching for specific

data, inserting new data, and removing data.

- Understanding data structures is crucial for effectively managing and manipulating data.

To summarize, data structures are methods of organizing, storing, and manipulating data, including arrays, linked lists, stacks, queues, trees, graphs, hash tables, and heaps.

Basic Data Structures

Below are some examples of basic data structures:
Arrays - Sequential collections of elements with efficient access and modification.

- Sequential collection of elements with unique indices. Indexes from zero.
- Efficient access and modification of elements at specific locations.
- Less efficient for inserting or removing elements in the middle.

Linked Lists - Chain of nodes allowing efficient insertion and removal.

- Chain of nodes where each node contains data and a reference to the next node.
- Efficient insertion and removal of elements.
- Sequential traversal required for access specific elements.

Stacks - Follows Last-In-First-Out (LIFO) principle for efficient insertion and removal from the top.

- Follows Last-In-First-Out (LIFO) principle.
- Insert and remove elements from the top of the stack.
- Useful for tasks like function class and undo operations.

Queues - Follows First-In-First-Out (FIFO) principle for efficient insertion, and removal from the front and rear.

- Follows First-In-First-Out (FIFO) principle.
- Insert elements at the rear and remove elements from the front.
- Useful for tasks like process scheduling.

Trees - Hierarchical structure for enabling efficient searching, insertion, and deletion.

- Hierarchical structure consisting of nodes connected by edges.
- Efficient searching, insertion, and deletion operations.
- Suitable for organizing file systems or representing hierarchical relationships.

Graphs - Collection of nodes connected by edges, useful for representing complex relationships.

- Collection of nodes connected by edges.
- Each node can have multiple connections.
- Used to represent complex relationships like social networks or computer networks.

Hash Tables - Data structure that uses hashing for efficient insertion, retrieval, and deletion of key-value pairs.

- Efficient data structure using hashing for fast key-value pair operations.
- Uses a hash function to convert keys into indices.
- Provides quick access to elements and handes collisions for proper storage.

Heaps - Binary-tree based structure that ensures efficient retrieval of the minimum or maximum element.

- Binary tree-based structure for efficient retrieval of minimum or maximum element.
- Maintains a partial order property, such as the min-heap or max-heap property.
- Supports fast insertion and deletion of elements while preserving the heap property.

In the study of data structures, we explore various methods of organizing, storing, and manipulating data.

Arrays are sequential collections of elements, allowing efficient access and modification. Linked Lists from a chain of nodes, facilitating efficient insertion and removal. Stacks follow the Last-In-First-Out principle and are useful tasks like function calls and undo opertions. Queues follow the First-In-First-Out principle and are suitable for process scheduling.

Trees, consisting of nodes connected by edges, provide a hierarchical organization, enabling efficient searching, insertion, and deletion. Graphs are collections of nodes connected by edges and represent complex relationships like social networks or computer networks.Hash Tables emply hashing for efficient insertion, retrieval, and deletion of key-value pairs. They use a hash function to convert keys into indices, providing fast access to elements while handling collisions. Heaps, based on binary trees, allow efficient retrieval of the minimum or maximum element. They maintain a partial order property and support fast insertion and deletion while preserving the heap property.

Understanding these data structures and their characteristics is essential for problem-solving and designing efficient algorithms in data-oriented scenarios.
