This course supports the assessment for Data Structures. The course covers 8 competencies and represents 4 competency units.

Introduction

Overview

Welcome to Data Structures! This course is part of the soft applications degree program covering the fundamentals of dynamic data structures and their associated algorithms using object-oriented design and abstract data types as a design paradigm. Problem-solving techniques applied to the design of efficient and maintainable software applications is emphasized in this course. You will have the chance to implement simple applications using the techniques learned.

Data Structures is a foundational course in any Information Technology program. It presents you with a number of advanced conceptions to help you create maintainable and efficient software applications. Data Structures leverages the knowledge you have learned from introductory courses from the IT core and programming. The exercises within this course will challenge your skills making you a better and stronger programmer.

Select Coursework from the menu to begin.

Competencies
This course provides guidance to help you demonstrate the following 8 competencies:

- **Competency 4008.1.1: Introduction to Abstract Data Types, Algorithms, and Data Structures Using Bags (Unordered Lists)**
  The graduate explains how to design abstract data types (ADTs), data structures to represent an ADT in storage, algorithms to manipulate ADTs (using the Bag ADT as an example). The graduate also describes bag data types and the use of both sequential and linked allocation to implement them.
- **Competency 4008.1.2: Introduction to Analysis of Algorithms**
  The graduate analyzes the time and space complexity of basic algorithms.
- **Competency 4008.1.3: Stacks, Queues, and Deques**
  The graduate describes design, specification, implementation of stacks, queues, and

deques. The graduate also implements a simple application using sequentially allocated queues as well as stacks that employ a linked allocation.

- **Competency 4008.1.4: Lists**
  The graduate describes design, specification, and implementation of list structures.
- **Competency 4008.1.5: Sorting and Sorted Lists**
  The graduate identifies basic selection and insertion sorting algorithms, as well as faster sorting algorithms, and describes the design and implementation of sorted lists.
- **Competency 4008.1.6: Basic Searching Algorithms and Associative ADTs**
  The graduate describes how to use searching algorithms for lists, and explains the concept of a dictionary as an associative ADT.
- **Competency 4008.1.7: Hashing Algorithms and Structures**
  The graduate describes hash tables, bucket hashing, their use as an efficient way to implement an associative ADT, and implements a simple application that uses bucket hashing.
- **Competency 4008.1.8: Tree Structures**
  The graduate describes tree structures and binary trees, and implements a simple application involving building and searching a binary tree.

Course Instructor Assistance
As you prepare to successfully demonstrate competency in this subject, remember that course instructors stand ready to help you reach your educational goals. As subject matter experts, mentors enjoy and take pride in helping students become reflective learners, problem solvers, and critical thinkers. Course instructors are excited to hear from you and eager to work with you.

Successful students report that working with a course instructor is the key to their success. Course instructors are able to share tips on approaches, tools, and skills that can help you apply the content you're studying. They also provide guidance in assessment preparation strategies and troubleshoot areas of deficiency. Even if things don’t work out on your first try, course instructors act as a support system to guide you through the revision process. You should expect to work with course instructors for the duration of your coursework, and you are encouraged to contact them as soon as you begin. Course instructors are fully committed to your success!

Coursework

Getting Started

There are two assessments associated with this course. Read the following items to understand the BEST way to be successful:

- The preassessment should be taken first and can be accessed by selecting the Assessment tab above.
  (Data Structures (PABO - TAKE NOW)
- Each lesson of this course corresponds with a topic from your coaching report.
- Based on the results of the preassessment, you will know exactly which areas need attention.

- If you have taken your preassessment and have demonstrated sufficient competency for specific topics, then you have the choice to skip over the lessons representing those topics.
- Going through the course from beginning to end is beneficial as it builds on itself to more and more complex material. In other words, if this material is generally new to you from the start, your best bet is to start at Lesson 1, complete all the practice included, and work your way through to the end.

Learning Resources

All of the course content will be found within the uCertify course:

- You can directly access the learning resources by clicking on the Launch uCerify button below.
- You can access each lesson from the course menu.
- You may be prompted to log in to the WGU student portal to access the resources. Access to each Lesson and Self-Assessment is available via the Coursework menu of the course.

Launch Course

Suggested Course Pace

Completing three lessons per week will keep you in the suggested course completion timeframe.

Good luck, ask your course instructor any questions you have, and enjoy!

Lock In Your Progress

Once you are ready to start or are actively working in the uCertify learning resource, lock in your progress. You only need to complete this step once; any future activity in uCertify will be saved.
Mark this Activity Complete to Lock In Your Progress

Click the check mark above or below if you are actively engaged in this course.

Lesson 1: Abstract Data Types and Sequentially Allocated Bags

In this lesson, you will review the concept of an abstract data type using the bag data type as an example. You will also learn about sequential allocation of bags, that is, allocating storage to

bags using arrays.

Learn

Lesson 1: Abstract Data Types and Sequentially Allocated Bags

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the concept of an ADT.
- Describe the bag ADT.
- Identify appropriate uses of a bag ADT.
- Describe the implementation of the bag ADT using sequential allocation.
- Identify a bag object that uses arrays for storage allocation.
- Describe the strengths and weaknesses of sequential allocation for the implementation of the bag ADT.

Lesson 2: Implementing Bags With Linked Allocation

You studied the concept of bag data types and the implementation of bags using sequential allocation in the previous lesson. Now you will use linked allocation for implementing bags. Linked allocation involves using linked chains to allocate storage to bags.

Learn

Lesson 2: Implementing Bags With Linked Allocation

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the linked organization of data.
- Describe the implementation of the bag ADT using linked allocation.
- Implement a bag object that uses linked allocation of storage.
- Compare the strengths and weaknesses of linked allocation with sequential allocation for the implementation of a bag.

Lesson 3: Introduction to Analysis of Algorithms

Throughout this course you will analyze the performance of the algorithms used to manipulate data structures and compare the performance of operations as different designs are used for the implementation of various data structures. In this lesson, you first will study concepts of time complexity and space complexity as performance measures of an algorithm. This includes the worst-case, best-case, and average-case complexities. You will also learn about the big-O notation for specifying the performance complexity of algorithms and the basic method of counting operations to calculate the time complexity of an algorithm.

Learn

Lesson 3: Introduction to Analysis of Algorithms

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Define the concepts of best, average, and worst-case time complexity and space complexity.
- Explain how to calculate time complexity by counting basic operations.
- Define big-O notation.
- Identify the time complexity of some simple algorithms involving loops and nested loops.
- Compare the worst-case and average-case time complexity of bag methods for linked and sequential allocation structures.

Lesson 4: Stacks

In this lesson, you will investigate a basic but very important data structure, described by the Stack ADT. Just as with the Bag ADT, the Stack ADT may be implemented using either sequential or linked allocation.

Learn

Lesson 4: Stacks

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the stack ADT.
- Identify code to specify stack methods.
- Describe the use of the stack ADT to verify matching parentheses in an input.
- Implement a stack ADT with linked allocation.
- Implement a stack with sequential allocation.

Lesson 5: Queues

In this lesson, you will investigate the Queue ADT. Analogous to stacks, queues may be implemented either with linked or sequential allocation.

Learn

Lesson 5: Queues

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the queue ADT.
- Describe the use of a queue to simulate waiting in line for service.
- Describe the use of the queue ADT to calculate the capital gain of a stock.
- Implement code for a queue ADT with linked allocation.
- Implement code for a queue ADT with sequential allocation.

Lesson 6: Deques

This lesson covers doubly-ended queues, otherwise known as the Deque ADT; this is a generalization of both stacks and queues.

Learn

Lesson 6: Deques

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the deque ADT.
- Describe the use of the deque ADT to calculate the capital gain of a stock.
- Describe a doubly-linked implementation for the deque ADT.

- Implement the deque ADT with doubly-linked allocation.

Lesson 7: Lists

In this lesson, you will investigate the List ADT. Analogous to the Stack and Queue ADTs, the List ADT may be implemented either with sequential or linked allocation.

Learn

Lesson 7: Lists

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the list ADT.
- Describe the implementation of a list using sequential allocation.
- Describe the implementation of the list ADT using linked allocation.
- Compare the efficiency of the sequential and linked implementations of the list ADT.
- Differentiate between queue, deques, stacks, lists, and bags.

Lesson 8: Basic Sorting Algorithms

This lesson covers some very important algorithms for sorting data. First, you should understand the sorting model, sorting by comparisons. In this lesson you will learn about basic sorting algorithms that require O(n^2) comparisons.

Learn

Lesson 8: Basic Sorting Algorithms

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the concept of sorting by comparisons.
- Describe the selection sort algorithm.
- Explain why the worst-case time complexity for the selection sort algorithm is O(n^2).
- Describe the insertion sort algorithm.
- Explain why the worst-case time complexity for the insertion sort algorithm is O(n^2).

Lesson 9: Faster Sorting Algorithms

In this lesson, you will examine some faster sorting algorithms, Merge Sort and Quicksort. It is possible to have even faster sorting algorithms if knowledge of the data is used instead of having to sort by comparisons only; this is exploited by the Radix sort algorithm.

Learn

Lesson 9: Faster Sorting Algorithms

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the Merge Sort algorithm.
- Identify the worst-case, average-case, and best-case time complexity for Merge Sort as O(nlog(n)).
- Describe the Quicksort algorithm.
- Describe the Radix Sort algorithm, and contrast with sorting by comparison.
- Explain why the time complexity of Radix Sort is O(n).

Lesson 10: Sorted Lists

In this lesson, you will learn about the Sorted List ADT and will investigate the definition and description of this ADT as well as using linked allocation for its implementation. You will also examine an algorithm to insert new elements into a sorted list.

Learn

Lesson 10: Sorted Lists

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the sorted list ADT.
- Describe linked allocation for the sorted list ADT.
- Describe an algorithm to insert a new element into a sorted list.

Lesson 11: Searching a List

This lesson covers algorithms for searching for data in a list. These algorithms involve searching for data in a list that can either be sorted or unsorted. They also involve lists that are allocated

sequentially or using linked chains, giving rise to a variety of search algorithms.

Learn

Lesson 11: Searching a List

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the problem of searching a list.
- Describe algorithm to search an unsorted array.
- Describe algorithm for sequential search of a sorted array.
- Describe algorithm for binary search of a sorted array.
- Describe algorithm for sequential search of an unsorted chain.
- Describe algorithm for sequential search of a sorted chain.
- Contrast the worst-case and average-case time complexity of the searching algorithms.

Lesson 12: Dictionary as an Associative ADT

In this lesson you will learn about the dictionary ADT as an associate ADT that provides an abstraction for search algorithms and their underlying data structure.

Learn

Lesson 12: Dictionary as an Associative ADT

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the dictionary ADT.
- Identify code that uses a dictionary ADT.

Lesson 13: Sequential Hash Tables

In this and the next lesson you will learn some hashing techniques that are more efficient for searching than the basic searching of lists. You will learn about the concept of hashing, hash functions, and sequential hash tables; the problem of collisions in a sequential hash table and the use of linear probing to resolve collisions; the use of algorithms to insert new elements into; and items in a sequential hash table for which linear probing is used to resolve collisions.

Learn

Lesson 13: Sequential Hash Tables

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Explain the idea of hashing.
- Explain the problem of collisions and how they are resolved by linear probing.
- Describe the algorithm to insert a new entry into a hash table using linear probing.
- Describe the algorithm to search for an entry from a sequential hash table.

Lesson 14: Bucket Hashing

In this lesson, you will investigate an alternate hashing structure, bucket hashing, which uses the searching for elements in a linked chain to resolve collisions.

Learn

Lesson 14: Bucket Hashing

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the bucket hashing structure.
- Describe the algorithm to add a new entry into a bucket hash structure.
- Describe the algorithm to search for an entry in a bucket hash structure.
- Compare sequential hash tables and bucket hash structures.
- Implement a dictionary ADT, using a bucket hash structure, and a small application to exercise the ADT.

Lesson 15: Introduction to Trees

In this lesson, you will learn about tree data structures: the concept of trees in general, the important category of trees called binary trees, tree traversals, and some of the applications of general trees and binary trees.

Learn

Lesson 15: Introduction to Trees

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the concept of a general tree structure.
- Describe the concept of a binary tree.
- Describe the various types of tree traversals.
- Give examples of applications of binary trees and general trees.

Lesson 16: Implementation of Binary Trees

This lesson explores the concept and implementation of binary trees in greater detail.

Learn

Lesson 16: Implementation of Binary Trees

This topic highlights the following objectives:

- Describe the binary tree ADT.
- Define the methods for a binary tree.

Lesson 17: Binary Search Trees

In this final lesson, you will learn about another data structure to support efficient searching: the binary search tree. You will investigate the methods to create a binary search tree and search a binary search tree for a value. Other important algorithms on a binary search tree are for the insertion and deletion of elements in a binary search tree. You will learn about insertion into a binary search tree and how to investigate deletion from a binary search tree. Make sure to study the analysis of the time complexity of algorithms on binary search trees carefully. Finally, complete your investigation of implementing a dictionary ADT using binary search trees.

Learn

Lesson 17: Binary Search Trees

Assess

Self-assessment:

- Quiz

This topic highlights the following objectives:

- Describe the concept of a binary search tree, including the ADT specification.
- Define method to create a binary search tree.
- Define an algorithm to search for an element from a binary search tree.
- Define an algorithm to add an element to a binary search tree.
- Define an algorithm to delete an element from a binary search tree.
- Describe worst-case scenarios and time complexity for add, delete, and lookup methods.
- Describe the strengths and weaknesses of lists, hashing, and binary search trees for implementing a dictionary ADT.
- Implement a dictionary ADT using a binary search tree and modify the ADT application built using a bucket hash structure above to use the binary search tree implementation of a dictionary ADT.

Final Steps

One of the many things that makes WGU unique is its competency-based education model. If you know the material, all you have to do is prove it by passing the exam. If you can do this, you can accelerate the receipt of your degree.

Performance Assessment

Follow the instructions on the Assessment tab to complete and submit your performance assessment.

Objective Assessment

One of the many things that makes WGU unique is its competency-based education model. If you know the material, all you have to do is prove it by passing the exam. If you can do this, you can accelerate the receipt of your degree.

To make sure you have the best chance possible to pass the exam on your first attempt, the following steps should be completed successfully before you take it:

- Complete all uCertify lessons (including activities, quizzes, and labs).
- Make sure you get 100% on each quiz for each lesson. If you do not pass a quiz on your first attempt, talk to your course instructor and go back to the lesson content; it is not likely you are ready to take the objective assessment if passing each quiz does not come easily to you.

- Complete the three lab assignments and submit them to the course instructor for feedback.
- Re-take the preassessment and make sure you pass with at least 80%. (Note: It is suggested that you take and pass the performance assessment before you attempt to take your objective assessment.)
- If you have completed the steps above and you feel comfortable with all of the concepts presented, you are most likely ready to attempt the exam.

If you fail your first attempt, you will be required to contact the course instructor to see what went wrong and how you can prepare to ensure a successful second attempt. After determining you are ready, your course instructor will approve your request once to make another exam attempt.