Data Structures in C++ Syllabus

High School (130-145 Contact Hours)

Course Overview and Goals

Data Structures in computer science focuses on different ways to store data, beyond traditional variables and lists. In this course, students will learn about advanced data structures such as maps, queues, sets, etc. while applying them in larger, real-world assignments and projects.

The Data Structures course is designed for students that have previously completed a full year computer science course, such as AP CSA. While C++ is used as the language for the course, the focus of the course is on understanding and applying advanced data structures. Prior C++ knowledge is not a prerequisite, however students should have a working knowledge of basic computer science concepts such as variables, control structures, and functions/methods in at least one programming language.

Learning Environment

The course utilizes a blended classroom approach. The content is fully web-based, with students writing and running code in the browser. Teachers utilize tools and resources provided by CodeHS to leverage time in the classroom and give focused 1-on-1 attention to students. Each unit of the course is broken down into lessons. Lessons consist of tutorials, short quizzes, example programs to explore, and written programming exercises, adding up to over 100 hours of hands-on programming practice and projects in total. Each unit ends with a comprehensive unit test that assesses a student's mastery of the material from that unit.

Programming Environment

Students write and run C++ programs in the browser using the CodeHS editor.

Quizzes

Each lesson includes at least one formative short multiple-choice quiz. At the end of each module, students take a summative multiple choice quiz that assesses their knowledge of the concepts covered in the module.

Prerequisites

Prior knowledge of C++ is not required, however students should have taken at least one other full length course in another language.

Course Breakdown

Module 1: C++ Basics (2-3 weeks/12-15 hours)

In this module, students will learn basic C++ syntax while reviewing fundamental concepts that they may have seen in other languages, such as loops and conditionals.

Objectives / Topics Covered	• Basic C++ Syntax and Program Structure • Review User Input and Output • Variables in C++ • Conditional Statements • Loops • Functions
Example Assignments / Labs	• Input, Output, And Program Structure ○ How do we read and write to the console? ○ What is the structure of a C++ program? ○ Example activity: Create a Haiku - Students learn about basic output by writing a haiku over 3 lines. • Basic Data Types ○ What are the basic variables types in C++? ○ How do we declare and define a new variable? ○ Example activity: Loose Change - Students are given a value and need to find the least number of coins needed to make that change. • Conditional Statements ○ How do we control the flow of a program using if/Else If statements? ○ Example activity: Choose Your Own Adventure - Students write a story with 8 possible endings based on user input. • Loops ○ When and where should we use loops in a program? ○ Example activity: Hacker Speak - Students use a loop to turn a string into h4ck3r sp34k! • Functions in C++ ○ How do we define and call a function in C++? ○ How do we pass variables by reference versus value? ○ When do we use a function prototype? ○ Example activity: Encryption - Students will create a basic Caesar cipher encryption program. • Project: Code Verification ○ Students are tasked with comparing an employee key generator to a company key generator and determine how many times they match. ○ This project involves basic math, internal binary conversion, and string comparisons to solve the problem.

Module 2: Going Beyond the Basics (4 week/20 hours)

Expanding from module 1, students will explore a few basic C++ principles that they may not have seen in other courses, such as vectors, structs, file handling, error handling.

Objectives / Topics Covered	● Defining and Using Vectors ● How to Use Function Default Values ● Defining and Using Structs ● Reading and Writing to a File ● Basic Error Handling
Example Assignments / Labs	● Vector Basics ○ How do we create and use a vector? ○ Example activity: Final Grade Calculator - Students prompt the user for up to 4 grades and then calculate the average needed for the remainder of the year to get various grades. ● Function Default Values ○ How do we create default values for parameters? ○ Why do we use default values? ○ Example activity: Appointment Follow Up - Students write a program to select the follow up day based on the day of an appointment and the type of doctor they are seeing. ● Structs ○ How do we create and use a struct? ○ Example activity: Movies Part 1 - This activity is part of a 3 part activity in this module where students will create a searchable movie database by importing data from IMDB. ● File Input/Output ○ How do we open a file to read or write? ○ How do we process data imported from a file? ○ Example activity: Sum From a File - Student will read in values from a file and calculate the sum ● Error Handling ○ How do we throw and catch errors? ○ What types of errors does C++ generate? ○ Example Activity: Next Birthday - Students will put error processing around a basic task to ensure a quality input value. ● Project: Karel Internship ○ Students are tasked with helping to find a bug in a new program designed to teach dogs tricks. Students will use structs, file input, and problem solving skills to determine where the game instructions create an infinite loop.

Module 3: Libraries (2-3 weeks/12-15 hours)

Students will learn how to create header files and libraries, as well as get a glimpse of the libraries we will use in the remainder of the course. They will also have an opportunity to use these libraries in a project.

Objectives / Topics Covered	● Creating Header and Implementation Files ● Using Libraries
Example Assignments / Labs	● Header Files ○ What is a header file? ○ What is an implementation file? ○ Example activity: Basic Math Library - Students create a header and implementation file to create and use basic math operations. ● Using Libraries ○ What are some useful libraries to use? ○ Example activity: Patient Records - Students use basic libraries to determine which patients are eligible to visit the mall. ● Project: The Game of Pig ○ Students recreate a popular game of chance where they roll a die and decide whether to risk their current sum or bank and move on.

Module 4: 2D Vectors, Stacks, and Queues (4-5 weeks/20-25 hours)

In this module students will explore several different sequential data structures and apply these to real world projects. This module has 3 larger projects.

Objectives / Topics Covered	● 2D Vectors ● Stacks ● Queues
Example Assignments / Labs	● 2D Vectors ○ How do we define, populate, and access a 2D vector? ○ Example activity: Magic Square Tester - Students will practice accessing a 2D vector to test a square of numbers to determine if it is a magic square (rows, columns, and diagonals all add to the same number). ● Project: The Game of Life ○ Students recreate the classic John Conway simulator, the Game of Life. Students will create a grid and update each round based on basic rules. ● Stacks ○ What is a stack and where would we use it? ○ Example activity: Browser History - Using two stacks, students will

	create a basic backward and forward feature to view browser history. ● Project: RPN Calculator ○ Students will learn about the Reverse Polish Notation calculator and then create their own basic version of this calculator. ● Queues ○ What is a queue and where do we use them? ○ Example activity: Line Simulator - Using one and three queues, students will explore the best way to handle a grocery store queue. ● Project: Karaoke Night ○ Students create an application for a DJ to use on karaoke night. The students will create an interface that allows users to enter a new song and find out who is next up.

Module 5: Sets and Maps (4 weeks/20 hours)

Building on the previous module, students will learn about associative data structures and how they can be used to solve problems.

Objectives / Topics Covered	● Sets ● Maps ● Iterating Through a Data Structure
Example Assignments / Labs	● Pairs and Iterators ○ How do we create a pair and where do we use them? ○ How can we use an iterator to loop through a data structure? ● Sets ○ What is a set and when would we make use of them? ○ How do we iterate through a set? ● Project: Bingo ○ Students create a multiplayer bingo game using sets to create Bingo boards and record picked numbers. ● Maps ○ When do we use maps? ○ How do we iterate through a map ● Project: FRC Scoring System ○ Students create a basic scoring system for a First Robotics Competition. The scoring system will allow them to import matches, score them, and display the current rankings.

Module 6: Recursion (4-5 weeks/20-25 hours)

This module will focus on problem solving using recursive function calls. Students will learn several basic strategies and apply them to real-world problems.

Objectives / Topics Covered	• Functional recursion • Procedural recursion • Recursive Backtracking
Example Assignments / Labs	• Writing Basic Functional Recursive ○ How do we define the base case? ○ How do we define the recursive call? • Procedural Recursion ○ How can we write a procedural recursion to solve a problem? • Recursive Backtracking ○ How do we set up a basic backtracking problem? ○ What problems can be solved with backtracking? • Project: Air Freight ○ Students will learn about backtracking with a spin on the classic knapsack problem as they find the optimal solution to staying within a shipping limit. • Project: Sudoku Solver ○ Students use a recursive backtracking algorithm to find solutions to a classic numbers problem.

Module 7: Pointers, Linked Lists, and Graphs (4-5 weeks/20-25 hours)

In this module, students will examine pointers and how they can be applied to different data structures to solve problems.

Objectives / Topics Covered	• Pointer values • Singly linked lists • Doubly linked lists • Graphs
Example Assignments / Labs	• Pointer values ○ How do we create a pointer? ○ How do we dereference a pointer value? ○ How do we pass pointer values to functions? • Project: Team Roster ○ Students will explore how a pointer value can be used to help keep

	track of information across multiple records as they create a team roster for their robotics team. • Singly Linked list ○ How do we create a linked list? ○ Where can we use linked lists? • Doubly Linked list ○ What is the difference between a singly and doubly linked list? ○ Which problems are better solved with a doubly linked list? • Project: Mail Forwarding ○ Students will explore the use of a linked list to keep track of an email forwarding list for the high school graduating class. • Graphs ○ What is a graph? ○ Which problems are best solved with a graph?

Module 8: Final Project (1-2 weeks/5-10 hours)

The final project will involve students applying different data structures in an open ended problem format.

Objectives / Topics Covered	• Apply multiple data structures to a real world problem • Allow students to design and build a significant project over the course of several weeks
Example Assignments / Labs	• Project: Virtual Lines ○ Students will create a system for riders at an amusement park to add and remove their name from a queue.