Introduction to Computer Science in C++ Syllabus

High School (45 Contact Hours)

Course Overview and Goals

Introduction to Computer Science in C++ focuses on broad computer science concepts such as input/output, variables, control statements, and basic data structures. The course emphasizes problem-solving skills while focusing on real-world assignments and projects.

The Intro to Computer Science in C++ is an introductory level course, so students do not need any prior computer science experience. This course is a quarter-long course and does move quicker through concepts compared to other introductory courses at CodeHS. Once students complete the course, they will have a foundation of computer science skills and an understanding of basic C++ syntax.

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 module of the course is broken down into lessons. Lessons consist of videos, short quizzes, example programs to explore, and written programming exercises, adding up to 45 hours of hands-on programming practice and projects in total. After each content module, students will have a chance to demonstrate knowledge with a real-world project.

Programming Environment

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

Assessments

Each lesson includes one formative short multiple-choice quiz. At the end of each unit, students complete a summative project synthesizing skills learned across all units to that point.

Prerequisites

The Intro to Computer Science in C++ course is designed for complete beginners with no previous background in computer science. As a language, C++ may be more challenging for students and this course does move faster than other introductory courses at CodeHS.

Course Breakdown

Module 1: Programming Basics (1 - 2 weeks/7 - 9 hours)

In this module, students will learn basic C++ programming syntax, including input/output, variable types, and math functions.

Objectives / Topics Covered	• Basic C++ Syntax and Program Structure • User Input and Output • Variables Types • Math 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: ASCII Art - Students use basic output to create a pattern.
	• Basic Variable Types
	  ○ What are the basic variables types in C++?
	  ○ How do we declare and define a new variable?
	  ○ Example activity: Schedule Change - Students use variable swaps to rearrange their class schedule.
	• Math Functions
	  ○ Basic math calculations
	  ○ Casting values between integer and double
	  ○ Example Activity: Order Checkout - Students take a burger order and calculate the total including sales tax.

Module 2: Project: College Calculator (2 days/2 hours)

In this module, students will create a calculator that can compute the cost of attending college. After researching the college of their choice, students will be able to see a hypothetical cost of attending college based on their calculator.

This module is designed as a summative project covering concepts learned in module 1.

Module 3: Program Control (1 - 2 weeks/12-15 hours)

In this module, students will extend their skills as they learn about basic program control, including if statements, while loops, and for loops.

Objectives / Topics Covered	• Comparision and Logical Operators • If/Else Branching Statements • While/For Loops

Example Assignments / Labs	• Comparision and Logical Operators
	  ○ Simple and compound comparison operators
	  ○ Example activity: Driver's License - Students will ask the user for some basic information and then print whether they are eligible for a driver's license.
	• If/Else Statements
	  ○ Basic branching statements
	  ○ Compound branching statements
	  ○ Example activity: Decision Day - Based on several different inputs, the program will print out what to expect for the coming day.
	• While/For Loops
	  ○ Basic while and for loops
	  ○ Looping with different constraints and intervals.
	  ○ Example activity: A Random Walk - Students create a simulator to test how many steps are needed to move off a bridge, given a set of constraints.

Module 4: Project: Algebra Test (2 days/2 hours)

In this module, students will use random numbers, loops, and conditional statements to create an algebra quiz asking the user to solve simple two-step algebra equations. Students will keep track of the number of questions correct and print out the results at the end.

This module is designed as a summative project covering concepts learned in the previous modules.

Module 5: Functions (1 week /5 - 6 hours)

In this module, students will explore the use of functions, including the use of parameters and return values. They will then use these as they look at functional decomposition to help make their programming problems easier to solve and their code easier to read.

Objectives / Topics Covered	● Basic Functions ● Function Parameters and Return Values ● Functional Decomposition
Example Assignments / Labs	● Basic Functions ○ Function structure ○ Defining and calling functions ○ Example activity: Write Your Name in ASCII Art - Students create functions for the different letters in their name and then call them to print out their names. ● Function Parameters and Return Values ○ Using parameters to help customize functions ○ Adding return statements to return values to the calling line of code. ○ Example activity: HiLo Game - Students create a game where the user tries to predict if a number is going to be higher or lower than 7.

Module 6: Project: The Game of Nim (1 - 2 weeks/5 - 7 hours)

In this module, students will create two different versions of the game of Nim. Nim is a mathematical strategy game where two players remove items from a stack in turn. In their versions, students will create a game to play against the computer.

This module is designed as a summative project covering concepts learned in the previous modules..

Module 7: Vectors (1 week/4 - 5 hours)

In this module, students will learn how to use vectors, a basic data structure designed to hold a list of values of the same type.

Objectives / Topics Covered	• Creating and Accessing Vectors • Adding and Removing from Vectors • Traversing Vectors

Example Assignments / Labs	• Creating and Accessing Vectors
	  ○ Declaring and creating vectors with initial values
	  ○ Accessing and updating the values of a vector
	  ○ Example activity: Top Songs - Students create a vector containing their favorite songs and then update and print one song.
	• Adding and Removing from Vectors
	  ○ Creating empty vectors
	  ○ Adding and removing from vectors
	  ○ Example activity: The Playlist - Students get user input to create a list of songs and their length to create a playlist.
	• Traversing Vectors
	  ○ Accessing all elements of a vector in a systematic way
	  ○ Example activity: The Loop Problem - In this problem-solving challenge, students need to use a mathematical calculation to count the number of steps it takes to create a loop of numbers.

Module 8: Mastermind (1 - 2 weeks/5 - 7 hours)

In this module, students will recreate a version of the classic strategy game Mastermind. The project utilizes skills learned in all modules with an emphasis on problem-solving as students will be challenged to figure out the logic of which pegs need to be placed.

This module is designed as a summative project for the course and covers concepts learned throughout the entire course.