Introduction to Programming in C++

Course Text

There is no text for this course. All materials are included in the course fee.

Course Description

This course introduces programming concepts to students using the language of their choice: C++, Python, or Java, while following a consistent course sequence, structure, and schedule of assessments.

The course teaches the core computer science concepts of variables, branching, loops, arrays/lists, and functions/methods. It also introduces object-oriented programming with classes and inheritance. The course covers use pointers and streams and teaches a variety of good coding practices, including iterative development, code formatting, and variable naming schemes.

Course Objectives

After completing this course, students will be able to:

- Use standard input and output, and understand common syntax errors
- Declare and initialize variables with valid identifiers
- Develop programs that branch based on user input
- Combine loops and arrays/lists, and develop programs with multiple arrays/lists
- Write a function/method, then return from a function/method and parameterize a function
- Initialize class variables with class constructor
- Create derived and abstract classes
- Write a recursive function
- Use binary search, O notation, and algorithm analysis

Course Prerequisites

It is suggested, though not required, that students take Pre-Calculus or its equivalent before enrolling in this course.

Important Terms

In this course, different terms are used to designate tasks:

- **Participation Activities**: The Participation Activities in the textbook, which are interactive activities and form the core learning material. Student graded on completion of Participation Activities.
- **Challenge Activities**: The Challenge Activities in the textbook, which are small coding problems designed to bridge the gap between reading and labs. Each problem is auto-graded.
- **Labs**: A set of graded programming tasks. Students write a program to perform a defined task. Each lab targets a particular concept. All labs are graded automatically.
- **Midterm**: A graded online test. Fully-automated grading.
- **Final**: A graded and proctored online test. Fully-automated grading.

Academic Integrity Statement

Academic integrity is the pursuit of scholarly activity in an honest, truthful and responsible manner. Violations of academic integrity include, but are not limited to, plagiarism, cheating, fabrication and academic misconduct. Failure to comply with the Academic Integrity Policy can result in a failure and/or zero on the attempted assignment/examination, a removal from the course, disqualification to enroll in future courses, and/or revocation of an academic transcript.

Course Completion Policy

In order for a course to be considered complete, all required coursework must be attempted, submitted, and graded. Required coursework consists of graded assignments. Any Academic Integrity Policy violations may prevent a course from being considered complete.

Course Evaluation Criteria

Your score provides a percentage score and letter grade for each course. A passing percentage is **70%** or higher.

There are a total of 1000 points in the course:

Topic	Assessment	Points Available
1	Chapter 1: Introduction to C++	33
2	Chapter 2: Variables / Assignments	33
3	Chapter 3: Branches	33
4	Chapter 4: Loops	33
5	Chapter 5: Arrays / Vectors	33
6	Chapter 6: User-Defined Functions	33
7	Chapter 7: Objects and Classes	33

Topic	Assessment	Points Available
8	Chapter 8: Pointers	33
8	Midterm Exam	205
9	Chapter 9: Streams	33
10	Chapter 10: Inheritance	33
11	Chapter 11: Recursion	33
12	Chapter 12: Exceptions	33
13	Chapter 13: Templates	33
14	Chapter 14: Containers	33
15	Chapter 15: Searching and Sorting Algorithms	33
16	Final Exam	300
Total		1000

Course Topics and Objectives

Topic Number	Topic Title	Subtopics
1	Introduction to C++	• Programming (general) • Programming basics • Comments and whitespace • Errors and warnings • Computers and programs (general) • Computer tour • Language history • Problem solving • Why programming • Why whitespace matters • C++ example: Salary Calculation • C++ example: Married-couple names
2	Variables / Assignments	• Variables and assignments (general) • Variables (int) • Identifiers

Topic Number	Topic Title	Subtopics
		• Arithmetic expressions (general) • Arithmetic expressions (int) • Example: Health data • Floating-point numbers (double) • Scientific notation for floating-point literals • Constant variables • Using math functions • Integer division and modulo • Type conversions • Binary • Characters • Strings • Integer overflow • Numeric data types • Unsigned • Random numbers • Debugging • Auto (since C++11) • Style guidelines • C++ example: Salary calculation with variables • C++ example: Married-couple names with variables
3	Branches	• If-else branches (general) • If-else • More if-else • Equality and relational operators • Detecting ranges (general) • Detecting ranges with if-else statements • Logical operators • Order of evaluation • Example: Toll calculation • Switch statements • Boolean data type • String comparisons • String access operations • Character operations • More string operations • Conditional expressions • Floating-point comparison • Short circuit evaluation • C++ example: Salary calculation with branches

Topic Number	Topic Title	Subtopics
		• C++ example: Search for name using branches
4	Loops	• Loops (general) • While loops • More while examples • For loops • More for loop examples • Loops and strings • Nested loops • Developing programs incrementally • Break and continue • Variable name scope • Enumerations • C++ example: Salary calculation with loops • C++ example: Domain name validation with loops
5	Arrays / Vectors	• Array concept (general) • Vectors • Array iteration drill • Iterating through vectors • Multiple vectors • Vector resize • Vector push_back • Loop-modifying or copying/comparing vectors • Swapping two variables (General) • Debugging example: Reversing a vector • Arrays vs. vectors • Two-dimensional arrays • Char arrays / C strings • String library functions • Char library functions: ctype • C++ example: Annual salary tax rate calculation with vectors • C++ example: Domain name validation with vectors
6	User-Defined Functions	• User-defined function basics • Return • Reasons for defining functions • Functions with branches/loops

Topic Number	Topic Title	Subtopics
		• Unit testing (functions) • How functions work • Functions: Common errors • Pass by reference • Functions with string/vector parameters • Functions with C string parameters • Scope of variable/function definitions • Default parameter values • Function name overloading • Parameter error checking • Preprocessor and include • Separate files • C++ example: Salary calculation with functions • C++ example: Domain name validation with functions
7	Objects and Classes	• Objects: Introduction • Using a class • Defining a class • Inline member functions • Mutators, accessors, and private helpers • Initialization and constructors • Classes and vectors/classes • Separate files for classes • Choosing classes to create • Unit testing (classes) • Constructor overloading • Constructor initializer lists • The ‘this’ implicit parameter • Operator overloading • Overloading comparison operators • Vector ADT • Namespaces • Static data members and functions • C++ example: Salary calculation with classes • C++ example: Domain name availability with classes
8	Pointers	• Why pointers?

Topic Number	Topic Title	Subtopics
		• Pointer basics • Operators: new, delete, and ->
• String functions with pointers • A first linked list • Memory regions: Heap/Stack • Destructors • Memory leaks • Copy constructors • Copy assignment operator • Rule of three • C++ example: Employee list using vectors
9	Streams	• Output and input streams • Output formatting • Input string stream • Output string stream • File input • File output • C++ example: Parsing and validating input files • C++ example: Saving and retrieving program data • Overloading stream operators
10	Inheritance	• Derived classes • Access by members of derived classes • Overriding member functions • Polymorphism and virtual member functions • Abstract classes: Introduction (generic) • Abstract classes • Is-a versus has-a relationships • UML • C++ example: Employees and overriding class functions • C++ example: Employees using an abstract class
11	Recursion	• Recursion: Introduction • Recursive functions • Recursive algorithm: Search • Adding output statements for debugging

Topic Number	Topic Title	Subtopics
		• Creating a recursive function • Recursive math functions • Recursive exploration of all possibilities • Stack overflow • C++ example: Recursively output permutations
12	Exceptions	• Exception basics • Exceptions with functions • Multiple handlers • C++ example: Generate number format exception
13	Templates	• Function templates • Class templates • C++ example: Map values using a function template
14	Containers	• Range-based for loop • List • Pair • Map • Set • Queue • Deque • find() function • sort() function
15	Searching and Sorting Algorithms	• Searching and algorithms • Binary search • O notation • Algorithm analysis • Sorting: Introduction • Selection sort • Insertion sort • Quicksort • Merge sort
16	Final Exam	• Final Exam

Back to Top