straighterline

Syllabus

CS102 | **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 provides a comprehensive foundation in C++ programming fundamentals. Students begin with basic syntax, variables, and control structures including branches and loops. The course progresses through essential data structures like arrays and vectors, then advances to object-oriented programming concepts including classes, inheritance, and polymorphism. Key topics include memory management with pointers, file I/O through streams, and advanced features like templates and containers. Students explore recursive programming, exception handling, and implement common algorithms for searching and sorting. Hands-on coding practice problems and laboratory exercises reinforce theoretical concepts, preparing students for advanced programming coursework and real-world software development challenges.

Learning Outcomes

After completing this course, students will be able to:

1. Analyze the fundamental syntax and structure of C++ programs to identify correct programming practices.
2. Apply control flow structures (branches and loops) to solve computational problems.
3. Implement data storage and manipulation using arrays, vectors, and other container classes.
4. Design modular programs using user-defined functions with appropriate parameter passing and return values.
5. Create object-oriented solutions by defining classes with appropriate data members, member functions, and inheritance relationships.
6. Evaluate memory management techniques using pointers and dynamic memory allocation.
7. Construct programs that handle file input/output and data streams effectively.
8. Develop recursive algorithms to solve problems that can be broken down into smaller subproblems.
9. Apply exception handling mechanisms to create robust programs that gracefully handle runtime errors.
10. Synthesize generic programming concepts using templates and implement common searching and sorting algorithms.

Course Prerequisites

There are no prerequisites for this course.

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.

Assessment Types

StraighterLine courses may include any combination of the assessment types described below. Review the descriptions to learn about each type, then review the Course Evaluation Criteria to understand how your learning will be measured in this course.

Benchmarks

Benchmarks test your mastery of course concepts. You have 3 attempts, and your highest score counts. **Note:** Cumulative Benchmarks (final exams) only allow 1 attempt.

Capstones

Capstones are project-based assessments that help you apply concepts to real-world scenarios. You have 2 attempts, and your highest score counts.

Checkpoints

Checkpoints are quick knowledge checks on important course concepts. All are open-book, and most have 1-3 attempts.

AI Use-Case Policies

StraighterLine Capstone assessments operate under one of three AI Use-Case Policies. These designations are selected intentionally to support learners in developing digital literacy, ethical reasoning, and authentic communication skills. Each model requires students to engage meaningfully with the course outcomes while adhering to academic standards.

Independent Work Requirement: Capstones with this designation must be completed independently without using AI tools. The goal is for learners to showcase their own understanding and skills without AI assistance. Students are expected to generate and submit original work developed solely through their own reasoning and effort.

AI-Assisted Planning Option: Capstones with this designation may allow AI tools to support brainstorming and assessment planning. If allowed, students will be asked to document any AI assistance by noting how it informed their work. Documentation must be included within the assignment or in a designated reflection field. Examples include describing how an AI tool helped organize an outline, generate ideas, or surface sources for further exploration.

AI-Integration Requirement: Capstones with this designation require AI tools as part of the learning process. Students will be asked to reflect upon their AI interactions and AI contributions to the assessment. Reflections must include which tools were used, how they were used, and what insights students gained from the process. This promotes transparency, ethical use, and metacognitive skill-building.

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:

Assessment	Points	Learning Outcomes
Checkpoint 1: Introduction to C++	20	1
Benchmark 1: Introduction to C++	42	1
Checkpoint 2: Variables/Assignments	25	1
Benchmark 2: Variables/Assignments	42	1
Checkpoint 3: Branches	25	2
Benchmark 3: Branches	42	2
Checkpoint 4: Loops	25	2
Benchmark 4: Loops	42	2
Checkpoint 5: Arrays/Vectors	25	3
Benchmark 5: Arrays/Vectors	42	3
Checkpoint 6: User-Defined Functions	25	4
Benchmark 6: User-Defined Functions	42	4
Checkpoint 7: Objects and Classes	25	5
Benchmark 7: Objects and Classes	42	5
Checkpoint 8: Pointers	25	6
Benchmark 8: Pointers	42	6
Checkpoint 9: Streams	25	7
Benchmark 9: Streams	42	7
Checkpoint 10: Inheritance	25	5
Benchmark 10: Inheritance	42	5
Checkpoint 11: Recursion	25	8
Benchmark 11: Recursion	42	8
Checkpoint 12: Exceptions	25	9
Benchmark 12: Exceptions	42	9
Checkpoint 13: Templates	25	10

Assessment	Points	Learning Outcomes
Benchmark 13: Templates	42	10
Checkpoint 14: Containers	25	3
Benchmark 14: Containers	42	3
Checkpoint 15: Searching and Sorting Algorithms	25	10
Benchmark 15: Searching and Sorting Algorithms	42	10
Total	1000	

Course Roadmap

This roadmap provides an overview of the checkpoints and lessons covered in this course.

Checkpoint 1: Introduction to C++

- Programming
- Programming basics
- Console input
- Comments and whitespace
- Errors and warnings
- Computers and programs
- Integrated development environment
- Computer tour
- Language history
- Problem solving
- Why programming
- Why whitespace and precision matter
- C++ example: Salary Calculation
- C++ example: Married-couple names

Checkpoint 2: Variables/Assignments

- Variables and assignments (general)
- Variables (int)
- Identifiers
- 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

Checkpoint 3: Branches

- If-else branches (general)
- Detecting equal values with branches
- Detecting ranges with branches (general)
- Detecting ranges with branches
- Detecting ranges using logical operators
- Detecting ranges with gaps
- Detecting multiple features with branches
- Common branching errors
- Example: Toll calculation
- Order of evaluation
- Switch statements
- Boolean data type
- String comparisons
- String access operations
- Character operations
- Finding, inserting, and replacing text in a string
- Conditional expressions
- Floating-point comparison
- Short circuit evaluation
- C++ example: Salary calculation with branches
- C++ example: Search for name using branches

Checkpoint 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

Checkpoint 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
- C-String library functions
- Char library functions: cctype
- C++ example: Annual salary tax rate calculation with vectors
- C++ example: Domain name validation with vectors

Checkpoint 6: User-Defined Functions

- User-defined function basics
- Print functions
- Reasons for defining functions
- Writing mathematical functions
- Functions with branches
- Functions with loops
- Unit testing (functions)
- How functions work
- Functions: Common errors
- Pass by reference
- Using pass by reference to modify 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

Checkpoint 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

Checkpoint 8: Pointers

- Pointers (General)
- Pointers and dynamically allocated arrays
- Changing the size of a dynamically allocated arrays
- Dynamically allocating objects
- Allocating arrays of objects
- Classes with dynamically allocated data
- String functions with pointers
- Memory regions: Heap/Stack
- Destructors
- Memory leaks
- Copy constructors
- Copy assignment operator
- Rule of three
- Smart pointers
- C++ example: Employee list using vectors

Checkpoint 9: Streams

- Output and input streams
- Output formatting
- Input string stream
- Output string stream
- File input
- C++ example: Parsing and validating input files
- File output
- C++ example: Saving and retrieving program data
- Overloading stream operators

Checkpoint 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

Checkpoint 11: Recursion

- Recursion: Introduction
- Recursive functions
- Recursive algorithm: Search

- Adding output statements for debugging
- Creating a recursive function
- Recursive math functions
- Recursive exploration of all possibilities
- Stack overflow
- C++ example: Recursively output permutations

Checkpoint 12: Exceptions

- Handling exceptions
- Throwing exceptions
- Exceptions with files
- User-defined exceptions
- C++ example: Generate number format exception

Checkpoint 13: Templates

- Function templates
- Class templates
- C++ example: Map values using a function template

Checkpoint 14: Containers

- Range-based for loop
- List
- Pair
- Map
- Set
- Queue
- Deque
- find() function
- sort() function

Checkpoint 15: Searching and Sorting Algorithms

- Searching and algorithms
- Binary search
- O notation
- Algorithm analysis
- Sorting: Introduction
- Selection sort
- Insertion sort
- Quicksort
- Merge sort

Related Courses

IT101

IT Fundamentals

View Course →

MAT102

Quantitative Reasoning

View Course →

MAT202

Introduction to Statistics

View Course →