uCertify

Course Outline

C++ All-in One

04 Aug 2025

1. Course Objective
2. Exercises, Quizzes, Flashcards & Glossary

   Number of Questions
3. Expert Instructor-Led Training
4. ADA Compliant & JAWS Compatible Platform
5. State of the Art Educator Tools
6. Award Winning Learning Platform (LMS)
7. Chapter & Lessons

   Syllabus

   Chapter 1: Introduction

   Chapter 2: Configuring Your Desktop System

   Chapter 3: Configuring Your Mobile System

   Chapter 4: Creating Your First C++ Application

   Chapter 5: Storing Data in C++

   Chapter 6: Directing the Application Flow

   Chapter 7: Dividing Your Work with Functions

   Chapter 8: Splitting Up Source Code Files

   Chapter 9: Referring to Your Data Through Pointers

   Chapter 10: Working with Classes

   Chapter 11: Using Advanced C++ Features

   Chapter 12: Planning and Building Objects

   Chapter 13: Building with Design Patterns

   Chapter 14: Considering Functional Programming

   Chapter 15: Working with Lambda Expressions

   Chapter 16: Advanced Lambda Expressions

   Chapter 17: Dealing with Bugs

   Chapter 18: Debugging an Application

   Chapter 19: Stopping and Inspecting Your Code

1. Course Objective

The C++ All-in-One course is designed to equip you with everything you need to become a proficient C++ developer, whether you're a beginner or looking to enhance your existing skills. The course is structured to provide you with a thorough understanding of C++ programming concepts and techniques. The course covers the key areas such as Introduction to C++, Fundamental Concepts, Object-Oriented Programming, Advanced Topics, Standard Template Library, Project-Based Learning, Debugging and Troubleshooting.

2. Exercises

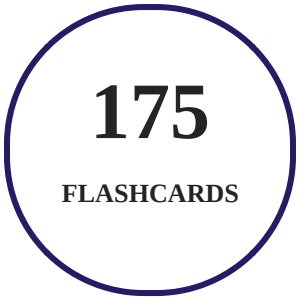
There is no limit to the number of times learners can attempt these. Exercises come with detailed remediation, which ensures that learners are confident on the topic before proceeding.

3. Quiz

Quizzes test your knowledge on the topics of the exam when you go through the course material. There is no limit to the number of times you can attempt it.

4. flashcards

Flashcards are effective memory-aiding tools that help you learn complex topics easily. The flashcard will help you in memorizing definitions, terminologies, key concepts, and more. There is no limit to the number of times learners can attempt these. Flashcards help master the key concepts.



5. Glossary of terms

uCertify provides detailed explanations of concepts relevant to the course through Glossary. It contains a list of frequently used terminologies along with its detailed explanation. Glossary defines the key terms.

6. Expert Instructor-Led Training

uCertify uses the content from the finest publishers and only the IT industry's finest instructors. They have a minimum of 15 years real-world experience and are subject matter experts in their fields. Unlike a live class, you can study at your own pace. This creates a personal learning experience and gives you all the benefit of hands-on training with the flexibility of doing it around your schedule 24/7.

7.  ADA Compliant & JAWS Compatible Platform

uCertify course and labs are ADA (Americans with Disability Act) compliant. It is now more accessible to students with features such as:

- Change the font, size, and color of the content of the course
- Text-to-speech, reads the text into spoken words
- Interactive videos, how-tos videos come with transcripts and voice-over
- Interactive transcripts, each word is clickable. Students can clip a specific part of the video by clicking on a word or a portion of the text.

JAWS (Job Access with Speech) is a computer screen reader program for Microsoft Windows that reads the screen either with a text-to-speech output or by a Refreshable Braille display. Student can easily navigate uCertify course using JAWS shortcut keys.

8.  State of the Art Educator Tools

uCertify knows the importance of instructors and provide tools to help them do their job effectively. Instructors are able to clone and customize course. Do ability grouping. Create sections. Design grade scale and grade formula. Create and schedule assessments. Educators can also move a student from self-paced to mentor-guided to instructor-led mode in three clicks.

9.  Award Winning Learning Platform (LMS)

uCertify has developed an award winning, highly interactive yet simple to use platform. The SIIA CODiE Awards is the only peer-reviewed program to showcase business and education technology's finest products and services. Since 1986, thousands of products, services and solutions have been recognized for achieving excellence. uCertify has won CODiE awards consecutively for last 7 years:

- **2014**
    1. Best Postsecondary Learning Solution

- **2015**
    1. Best Education Solution
    2. Best Virtual Learning Solution
    3. Best Student Assessment Solution
    4. Best Postsecondary Learning Solution
    5. Best Career and Workforce Readiness Solution
    6. Best Instructional Solution in Other Curriculum Areas
    7. Best Corporate Learning/Workforce Development Solution

- **2016**
    1. Best Virtual Learning Solution
    2. Best Education Cloud-based Solution
    3. Best College and Career Readiness Solution
    4. Best Corporate / Workforce Learning Solution
    5. Best Postsecondary Learning Content Solution
    6. Best Postsecondary LMS or Learning Platform
    7. Best Learning Relationship Management Solution

- **2017**
    1. Best Overall Education Solution
    2. Best Student Assessment Solution
    3. Best Corporate/Workforce Learning Solution
    4. Best Higher Education LMS or Learning Platform

- **2018**
    1. Best Higher Education LMS or Learning Platform
    2. Best Instructional Solution in Other Curriculum Areas
    3. Best Learning Relationship Management Solution

- **2019**
    1. Best Virtual Learning Solution
    2. Best Content Authoring Development or Curation Solution
    3. Best Higher Education Learning Management Solution (LMS)

- **2020**
    1. Best College and Career Readiness Solution
    2. Best Cross-Curricular Solution
    3. Best Virtual Learning Solution

10. Chapter & Lessons

uCertify brings these textbooks to life. It is full of interactive activities that keeps the learner engaged. uCertify brings all available learning resources for a topic in one place so that the learner can efficiently learn without going to multiple places. Challenge questions are also embedded in the chapters so learners can attempt those while they are learning about that particular topic. This helps them grasp the concepts better because they can go over it again right away which improves learning.

Learners can do Flashcards, Exercises, Quizzes and Labs related to each chapter. At the end of every lesson, uCertify courses guide the learners on the path they should follow.

Syllabus

Chapter 1: Introduction

- About This Course
- Icons Used in This Course
- Where to Go from Here

Chapter 2: Configuring Your Desktop System

- Obtaining a Copy of C++ 20
- Obtaining Code::Blocks

- Installing Code::Blocks
- Touring the Essential Code::Blocks Features

Chapter 3: Configuring Your Mobile System

- Obtaining CppDroid
- Considering Other Alternatives
- Touring the Essential CppDroid Features
- Obtaining CppDroid Help

Chapter 4: Creating Your First C++ Application

- Code::Blocks Creating a Project
- Typing the Code
- Starting with Main
- Showing Information
- Let Your Application Run Away

Chapter 5: Storing Data in C++

- Putting Your Data Places: Variables
- Manipulating Integer Variables

- Characters
- Strings
- Making Decisions Using Conditional Operators
- Telling the Truth with Boolean Variables
- Reading from the Console

Chapter 6: Directing the Application Flow

- Doing This or Doing That
- Evaluating Conditions in C++
- Including Evaluations in C++ Conditional Statements
- Repeating Actions with Statements That Loop
- Looping for
- Looping while
- Doing while
- Breaking and continuing
- Nesting loops

Chapter 7: Dividing Your Work with Functions

- Dividing Your Work

- Calling a Function
- Writing Your Own Functions
- Improving On the Basic Function
- Calling All String Functions
- Understanding main()

Chapter 8: Splitting Up Source Code Files

- Creating Multiple Source Files
- Sharing with Header Files
- Sharing Variables among Source Files
- Using the Mysterious Header Wrappers

Chapter 9: Referring to Your Data Through Pointers

- Understanding the Changes in Pointers for C++ 20
- Heaping and Stacking the Variables
- Creating New Raw Pointers
- Freeing Raw Pointers
- Working with Smart Pointers

- Passing Pointer Variables to Functions
- Returning Pointer Variables from Functions

Chapter 10: Working with Classes

- Understanding Objects and Classes
- Working with a Class
- Starting and Ending with Constructors and Destructors
- Building Hierarchies of Classes
- Creating and Using Object Aliases

Chapter 11: Using Advanced C++ Features

- Filling Your Code with Comments
- Converting Types
- Reading from the Console
- Understanding Preprocessor Directives
- Using Constants
- Using Switch Statements
- Supercharging enums with Classes
- Working with Random Numbers

- Storing Data in Arrays

Chapter 12: Planning and Building Objects

- Recognizing Objects
- Encapsulating Objects
- Building Hierarchies

Chapter 13: Building with Design Patterns

- Delving Into Pattern History
- Introducing a Simple Pattern: the Singleton
- Watching an Instance with an Observer
- Mediating with a Pattern

Chapter 14: Considering Functional Programming

- Understanding How Functional Programming Differs
- Defining an Impure Language
- Seeing Data as Immutable
- Considering the Effects of State
- Eliminating Side Effects

- Understanding the Role of auto
- Passing Functions to Functions
- Using Lambda Expressions for Implementation

Chapter 15: Working with Lambda Expressions

- Creating More Readable and Concise C++ Code
- Defining the Essential Lambda Expression
- Developing with Lambda Expressions

Chapter 16: Advanced Lambda Expressions

- Considering the C++ 20 Lambda Extensions
- Working in Unevaluated Contexts
- Using Assignable Stateless Lambda Expressions
- Dealing with Pack Expansions

Chapter 17: Dealing with Bugs

- It’s Not a Bug. It’s a Feature!
- Make Your Application Features Look Like Features
- Anticipating (Almost) Everything

- Avoiding Mistakes, Plain and Simple

Chapter 18: Debugging an Application

- Programming with Debuggers
- Debugging with Different Tools
- Debugging a Code::Blocks Application with Command-Line Arguments

Chapter 19: Stopping and Inspecting Your Code

- Setting and Disabling Breakpoints
- Watching, Inspecting, and Changing Variables

Chapter 20: Traveling About the Stack

- Stacking Your Data
- Debugging with Advanced Features

Chapter 21: Working with Arrays, Pointers, and References

- Building Up Arrays
- Pointing with Pointers
- Referring to References

Chapter 22: Creating Data Structures

- Working with Data
- Structuring Your Data
- Naming Your Space

Chapter 23: Constructors, Destructors, and Exceptions

- Constructing and Destructing Objects
- Programming the Exceptions to the Rule

Chapter 24: Advanced Class Usage

- Inherently Inheriting Correctly
- Using Classes and Types within Classes

Chapter 25: Creating Classes with Templates

- Templatizing a Class
- Going Beyond the Basics
- Parameterizing a Template
- Typedefing a Template
- Deriving Templates

- Templatizing a Function

Chapter 26: Programming with the Standard Library

- Architecting the Standard Library
- Containing Your Classes
- The Great Container Showdown
- Copying Containers
- Creating and Using Dynamic Arrays
- Working with Unordered Data
- Working with Ranges

Chapter 27: Filing Information with the Streams Library

- Seeing a Need for Streams
- Programming with the Streams Library
- Handling Errors When Opening a File
- Flagging the ios Flags

Chapter 28: Writing with Output Streams

- Inserting with the << Operator

- Formatting Your Output

Chapter 29: Reading with Input Streams

- Extracting with Operators
- Encountering the End of File
- Reading Various Types

Chapter 30: Building Directories and Contents

- Manipulating Directories
- Getting the Contents of a Directory
- Copying Files
- Moving and Renaming Files and Directories

Chapter 31: Streaming Your Own Classes

- Streaming a Class for Text Formatting
- Manipulating a Stream

Chapter 32: Exploring the Standard Library Further

- Considering the Standard Library Categories

- Parsing Strings Using a Hash
- Obtaining Information Using a Random Access Iterator
- Locating Values Using the Find Algorithm
- Using the Random Number Generator
- Working with Temporary Buffers

Chapter 33: Working with User-Defined Literals (UDLs)

- Understanding the Need for UDLs
- Working with the UDLs in the Standard Library
- Creating Your Own UDLs

Chapter 34: Building Original Templates

- Deciding When to Create a Template
- Defining the Elements of a Good Template
- Creating a Basic Math Template
- Building a Structure Template
- Developing a Class Template
- Considering Template Specialization
- Creating a Template Library

- Using Your Template Library

Chapter 35: Investigating Boost

- Considering the Standard Library Alternative
- Understanding Boost
- Obtaining and Installing Boost for Code::Blocks
- Creating the Boost Tools
- Using Boost.Build
- Using Inspect
- Understanding BoostBook
- Using QuickBook
- Using bcp
- Using Wave
- Building Your First Boost Application Using Date Time

Chapter 36: Boosting up a Step

- Parsing Strings Using RegEx
- Breaking Strings into Tokens Using Tokenizer

- Performing Numeric Conversion
- Creating Improved Loops Using Foreach
- Accessing the Operating System Using Filesystem

11.  Live Labs

The benefits of live-labs are:

- Exam based practical tasks
- Real equipment, absolutely no simulations
- Access to the latest industry technologies
- Available anytime, anywhere on any device
- Break and Reset functionality
- No hardware costs

Lab Tasks

Creating Your First C++ Application

- Building and Executing the First C++ Application
- Adding Two Numbers

Storing Data in C++

- Making a Calculator
- Adding Two Strings and Changing a Part of the String
- Comparing Two Integers Using the Conditional Operator

Directing the Application Flow

- Using Logical Operators
- Using the for Loop
- Using the do-while Loop
- Using the Nested for Loop

Dividing Your Work with Functions

- Using the pow() Function
- Using the PrintName() function

Splitting Up Source Code Files

- Defining and Calling a Function
- Adding a New Source Code File

Referring to Your Data Through Pointers

- Using Pointers to Point to a String
- Displaying the Allocated Memory
- Replacing Characters within a String
- Using Pointers to Modify a Variable Passed into a Function

Working with Classes

- Using the Public and Private Functions
- Using the this Variable
- Using Constructors and Destructors
- Deriving One Class from Another

Using Advanced C++ Features

- Performing TypeConversion
- Using the Switch Statement
- Creating a Class for enums
- Adding and Subtracting Pointers

Considering Functional Programming

- Finding the Factorial of a Number
- Using the auto Keyword
- Using Declarative Programming Techniques
- Using a Transform

Working with Lambda Expressions

- Demangling a Type Name
- Using the auto Keyword with Lambda Expressions
- Sorting Values Using a Lambda Expression

Advanced Lambda Expressions

- Defining a Priority Queue Comparator
- Using a Variadic Template

Stopping and Inspecting Your Code

- Setting a Breakpoint

Traveling About the Stack

- Calling the Nested Function

Working with Arrays, Pointers, and References

- Using the sizeof() Function
- Referencing a Method

Creating Data Structures

- Calculating the Maximum Value and Size of Data Types
- Printing the ASCII Values
- Copying Structures
- Pulling Names into Other Namespaces

Constructors, Destructors, and Exceptions

- Using the Copy Constructor
- Creating a Virtual Destructor
- Creating a Basic try-catch Block

Advanced Class Usage

- Using the Friend Function
- Using Multiple Inheritance

Creating Classes with Templates

- Creating and Using a Template
- Using Static Members in a Template
- Overloading a Function Template

Programming with the Standard Library

- Iterating Through a map
- Using Vectors as Container Classes
- Associating Objects with map
- Creating a Stack and a Queue

Filing Information with the Streams Library

- Reading and Writing to a File

Writing with Output Streams

- Displaying Flag-Formatted Numbers
- Using the Precision Function
- Setting the Width and Creating Fields

Reading with Input Streams

- Using the Extraction Operator

Streaming Your Own Classes

- Using Manipulators

Exploring the Standard Library Further

- Generating a Hash
- Generating Key and Value Pairs

Working with User-Defined Literals (UDLs)

- Using Prefixes and Suffixes
- Defining a UDL Operator

Building Original Templates

- Defining a Series of Function Templates
- Building a Structure Template

Boosting up a Step

- Generating Tokens from Strings
- Using the BOOST_FOREACH Loop

Here's what you get

68	7	14
LIVE LABS	VIDEO TUTORIALS	MINUTES