Document Generated: 04/01/2026

Learning Style: Virtual Classroom

Technology:

Difficulty: Beginner

Course Duration: 4 Days

Next Course Date: May 26, 2026

Introduction to C++ Programming Essentials (TTCP2100)

About This Course:

Introduction to C++ Programming Essentials is a four-day, hands-on course geared for developers who have a foundational grasp of object-oriented (OO)

programming. Throughout the course you'll explore how C++ can transform your programming skills, enabling you to tackle more complex and performance-intensive projects. Whether it's system software, game development, or optimizing existing code, the skills acquired in this course will serve as a strategic asset in your professional toolkit.

The course structure is a balanced mix of theoretical knowledge and practical application, with 50% of the time dedicated to hands-on labs. You'll begin by grasping the essentials of C++ file organization and toolsets, moving on to advanced topics like data handling with pointers and references, and function intricacies including overloading and inline functions. The curriculum also delves into class design, object lifecycle management, and dynamic memory management, equipping you with the skills to write efficient and maintainable code.

Working in a hands-on learning environment, guided by our expert instructor, you'll learn strategic problem-solving skills and build confidence in applying C++ effectively in your work environment. The labs simulate real-world challenges, preparing you to immediately implement your new skills. As you conclude this course, you'll leave with a comprehensive understanding of C++ applications, ready to handle complex programming tasks and contribute significantly to your project success.

Course Objectives:

After completing this course, students will be able to:

- Master data manipulation using pointers, references, and various data types in C++, essential for high-performance applications
- Gain proficiency in function overloading, inline functions, and call-by-reference, crucial for efficient and modular code
- Acquire skills in designing classes with constructors, destructors, and access modifiers, and managing object lifecycles for robust software development.
- Learn effective memory management techniques, including handling allocation errors, to write memory-efficient C++ programs.
- Understand and implement inheritance and polymorphism in C++ for creating flexible and reusable code.
- Utilize C++ Standard Library resources for efficient algorithm implementation and data handling.
- Master using private, public, and protected keywords for class member access control, and develop robust exception handling skills using try and catch blocks.
- Learn advanced class features like const and static members, operator overloading, and implement file I/O operations and string streams for comprehensive C++ programming.

Audience:

- This is a technical course that introducees C++ programming to experienced developers

Prerequisites:

- Practical hands-on prior programming experience and knowledge is required, preferably with some background in OO development. This course is not for non-developers, or new developers without practical experience.

Course Outline:

Getting Started

- Using the development environment
- C++ file organization and tools

Handling Data

- Primitive Types
- Initialization and Assignment
- Const
- Pointers
- Constant Pointers
- References
- Constant Reference Arguments
- Scope

Functions

- Function Prototypes and Type Checking
- Function Overloading
- Name Resolution
- Call by Value
- Call-by-Reference and Reference Types
- References in Function Return
- Constant Argument Types
- Providing Default Arguments
- Inline Functions

Declaring and Defining Classes

- Components of a Class
- Class Structure

- Class Declaration Syntax
- Member Data
- Built-in Operations
- Constructors and Initialization
- Initialization vs. Assignment
- Class Type Members
- Member Functions and Member Accessibility
- Inline Member Functions
- Friend Functions
- Static Members
- Modifying Access with a Friend Class

Creating and Using Objects

- Creating Automatic Objects
- Creating Dynamic Objects
- Calling Object Methods
- Constructors
- Initializing Member consts
- Initializer List Syntax
- Allocating Resources in Constructor
- Destructors
- Scope Resolution Operator ::
- Using Objects as Arguments
- Objects as Function Return Values
- Constant Methods
- Containment Relationships

Controlling Object Creation

- Object Copying and Copy Constructor
- Automatic Copy Constructor

Dynamic Memory Management

- Static, Automatic, and Heap Memory
- Free Store Allocation with new and delete
- Handling Memory Allocation Errors

Strings in C++

- Character Strings
- The String Class
- Operators on Strings
- Member Functions of the String Class

Streaming I/O

- Streams and the iostream Library
- Built-in Stream Objects

- Stream Manipulators
- Stream Methods
- Input/Output Operators
- Character Input
- String Streams
- Formatted I/O
- File Stream I/O
- Overloading Stream Operators
- Persistent Objects

Templates

- Purpose of Template Classes
- Constants in Templates
- Templates and Inheritance
- Container Classes
- Use of Libraries

Inheritance

- Inheritance and Reuse
- Composition vs. Inheritance
- Syntax for Public Inheritance
- Use of Common Pointers
- Constructors and Initialization
- Inherited Copy Constructors
- Destructors and Inheritance

Polymorphism in C++

- Definition of Polymorphism
- Calling Overridden Methods
- Upcasting
- Accessing Overridden Methods
- Virtual Methods and Dynamic Binding
- Virtual Destructors
- Abstract Base Classes and Pure Virtual Methods

The Standard Library

- Survey of the library
- Containers
- Algorithms
- Numerics
- Date & Time