Bright Star Institute

302.335.8868

admissions@brightstarinstitute.com

C++ Programming for Non-C Programmers

Duration: 90 hours

Prerequisites: Prior programming experience in any programming language.

Description: This hands on C++ programming course provides an accelerated introduction to the most essential syntactical components of the C and C++ languages for the first briefer portion of class, and the majority remainder of the course focuses on object-oriented programming with C++. The C++ programming training begins by introducing the built in data types, fundamental control constructs, and rich expression operator repertoire common to both C and C++.

Later, the central concepts of C++ syntax and style are taught in the context of using object-oriented methods to achieve reusability, reliability, and adaptability. Emphasis is placed on the features of C++ that support abstract data types, inheritance, and polymorphism. Students will learn to apply the process of data abstraction and class design in their C++ code development. Extensive code examples and programming exercises are provided, with approximately half of course time spent performing hands on programming labs. Practical aspects of C++ programming including efficiency, performance, testing, and reliability considerations are emphasized throughout the course.

Course Overview

ANSI C++ Fundamentals

- Block Structure of C and C++ Programs
- Fundamentals of Syntax
- Built in Data Types
- The Preprocessor and Macros
- Standard Runtime Libraries and Header Files

Data Types, Storage, Classes, and Scope

- Data Types and Qualifiers
- Constants and String Literals
- Static versus Automatic Storage
- Scope and Variables
- Initialization Rules

Operators and Expressions

- Arithmetic, Logical, and Bit Operators
- Precedence and Associativity

Flow Control Constructs

- Conditional Constructs: `if`, `switch`
- Looping Constructs: `while`, `do`, `for`

- Assignment
- Type Conversion Rules
- Type Casting

- Programming Style

Pointers

- Advantages of Pointers
- Uses of Pointers
- Declaring Pointers
- Pointer and Address Arithmetic
- Initializing and Dereferencing Pointers
- Pointers vs. Arrays

Functions

- Purpose of Functions
- Functions versus Inlining
- The Argument Stack
- Passing by Value
- Passing by Reference
- Declaring External Functions
- Function Prototyping

Moving from C to C++

- New Compiler Directives
- Stream Console I/O
- Explicit Operators
- Standard Libraries
- Data Control Capabilities

Handling Data

- New Declaration Features
- Initialization and Assignment
- Enumerated Types
- The `bool` Type
- Constant Storage
- Pointers to Constant Storage
- Constant Pointers
- References
- Constant Reference Arguments
- Volatile Data
- Global Data

Functions

- Function Prototypes and Type Checking
- Default Function Data Types
- Function Overloading
- Problems with Function Overloading
- Name Resolution
- Promotions and Conversions
- Call by Value
- Reference Declarations
- Call-by-Reference and Reference Types
- References in Function Return
- Constant Argument Types
- Conversion of Parameters Using Default Initializers

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
- Block and Function Scope
- File and Global Scope
- Class Scope
- Scope Resolution Operator ::
- Using Objects as Arguments

Providing Default Arguments

- Inline Functions

Dynamic Memory Management

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

Inheritance

- Inheritance and Reuse
- Composition vs. Inheritance
- Inheritance: Centralized Code
- Inheritance: Maintenance and Revision
    - Public, Private and Protected Members
    - Redefining Behavior in Derived Classes
    - Designing Extensible Software Systems
- Syntax for Public Inheritance
- Use of Common Pointers
- Constructors and Initialization
- Inherited Copy Constructors
- Destructors and Inheritance
- Public, Protected, Private Inheritance

Introduction to Object Concepts

- The Object Programming Paradigm
- Object-Orientated Programming Definitions
- Information Hiding and Encapsulation
- Separating Interface and Implementation
- Classes and Instances of Objects
- Overloaded Objects and Polymorphism

Strings in C++

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

C++ Program Structure

Objects as Function Return Values

- Constant Methods
- Containment Relationships

Controlling Object Creation

- Object Copying and Copy Constructor
- Automatic Copy Constructor
- Conversion Constructor

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

Exceptions

- Types of Exceptions
- Trapping and Handling Exceptions
- Triggering Exceptions
- Handling Memory Allocation Errors

Reliability Considerations in C++ Projects

Organizing C++ Source Files
- Integrating C and C++ Projects
- Using C in C++

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

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

The Standard Template Library

- STL Containers
- Parameters Used in Container Classes
- The Vector Class
- STL Algorithms
- Use of Libraries

Function Prototypes
- Strong Type Checking
- Constant Types
- C++ Access Control Techniques

Multiple Inheritance

- Derivation from Multiple Base Classes
- Base Class Ambiguities
- Virtual Inheritance
  - Virtual Base Classes
  - Virtual Base Class Information

Operator Overloading

- Advantages and Pitfalls of Overloading
- Member Operator Syntax and Examples
- Class Assignment Operators
- Class Equality Operators
- Non-Member Operator Overloading
- Member and Non-Member Operator Functions
- Operator Precedence
- The this Pointer
- Overloading the Assignment Operator
- Overloading Caveats