

C++, based on the C programming language, is an Object-Oriented Programming (OOP) language. Unlike C, C++ is built on the concept of "objects" instead of using data and actions on data as the basis for the program's logic. Using OOP, related data and routines are grouped into an object that then relates to other objects in the program. These objects can represent all of the parts and functions of a real entity or an abstract idea. C++ is a powerful language that is inherently useful for large-scale projects.

In this hands-on course, students will design and write programs in the C++ language. Emphasis is placed on object-oriented techniques, such as inheritance and function overloading to implement polymorphism. The class ends with an introduction to the Standard Template Library (STL).

Course Objectives:

- Explain what Object-Oriented Programming is.
- Identify abstractions to use as class names.
- Know when and how to use inheritance.
- Use polymorphism in C++ design.
- Create templates.
- Integrate exception handling into your programs.
- Create overloaded functions.
- Create and use overloaded operators.
- Pass objects by reference.
- Write constructors and destructors.
- Put objects on the heap.
- Define and use data and functions related to a class.
- Organize OOP code classes so classes properly support each other.

Audience: C programmers responsible for the development of advanced applications or systems programs in C++.

Prerequisites: *C Programming* or *Java Programming*. Programming experience in a C-style language, such as C, Java, or C# is required.

Number of Days: 5 days

<p>1. Course Introduction Course Objectives Overview Suggested References</p>	<p>Data Abstraction Enforcing Data Encapsulation File Organization Classes in C++</p>
<p>2. Classes Creating a Data Structure Methods Object Scope C++ Input and Output Namespaces</p>	<p>Objects this Pointer</p>
	<p>3. Constructors and Destructors Debug Output The Default Constructor When are Constructors Called?</p>

- The Destructor
- The Copy Constructor
- Other Constructors
- Composition
- The Report Class
- Code Reuse
- Initialization Lists
- 4. Inheritance**
 - Inheritance
 - Bugreport
 - Protected Access Modifier
 - Access and Inheritance
 - Constructors and Inheritance
 - Initialization Lists Revisited
 - Multiple Inheritance
- 5. Virtual Functions**
 - Inheritance and Assignment
 - Inside Report's Assignment Operator
 - Using Pointers – a Quick Look at Basics
 - Class Assignment and Pointers
 - Static Binding
 - Dynamic Binding
 - Polymorphism
 - The show_rep() Function
 - Using the show_rep() Function
 - Designing Member Function Inheritance
- 6. Pure Virtual Functions**
 - Bugfix and Its Relationship with Bugreport
 - Bugfix: Association with Bugreport
 - Using Bugfix with show_rep()
 - Adding Bugfix to the Hierarchy
 - Coding for the Document Class
 - Reexamining the Document Class
 - Pure Virtual Functions
 - Updated: Designing Member Function Inheritance
- 7. References and Constants**
 - References
 - Displaying References
 - Changing References
 - Pass by Reference
 - Returning by Reference
 - Constant Variables, References, and Methods
- 8. new and delete**
 - new and delete
 - Array Allocation
 - The Report Class
 - Compiler Version of the Copy Constructor
 - Guidelines for Copy Constructors
 - The Report Constructors and new
 - The Report Destructor and delete
 - Virtual Destructors
- 9. Casting in C++**
 - Casting: A Review
 - New Casting Syntax
 - Creating a String Class
 - The String Class
 - The Conversion Constructor
 - Expanding Our Casting Options
 - Casting Operator
 - Using the Casting Operator
- 10. Class Methods and Data**
 - Class Data
 - Class Methods
 - Using the New Data
 - More on Class Methods
- 11. Overloaded Functions**
 - Function Overloading
 - Using Overloaded Functions
 - Rules for Overloading
 - Overloading Based on Constness
 - Default Arguments
 - Invoking Functions with Default Arguments
- 12. Overloaded Operators**
 - The Basics of Overloading
 - Overloading operator+
 - Coping with Commutativity
 - Non-Commutative Operators
 - friends and Their Problems
 - The Assignment Operator
 - Overloading the << Operator
 - Using Date with cout
- 13. Exception Handling**
 - Why Exception Handling?
 - try / catch / throw
 - Exception Classes
 - Standard Exception Hierarchy

- Multiple catch Blocks
- Catching Everything
- Unhandled Exceptions
- Exception in Constructors and Destructors
- Designing for Exceptions
- 14. Standard Template Library**
- Class Template Concepts
- Standard Template Library (STL)
 - Overview
 - Containers
 - Iterators
 - Iterator Syntax
 - Non-Mutating Sequential Algorithms
 - Mutating Sequential Algorithms
 - Sorting Algorithms
 - Numeric Algorithms
 - auto_ptr Class
 - string Class
- 15. STL Containers**
- Container Classes
- Container Class Algorithms
- vector Class
- Additional vector Class Methods
- deque Class
- list Class
- set and multiset Classes
- map and multimap Classes
- 16. Appendix A – Reference Sheets**
- Constants, References, and Pointers
- Input/Output
- this Pointer
- The Complete Report/Document Hierarchy
- 17. Appendix B – Templates**
- Scenario
- Designing an Array Class
- Code for FloatArray and IntArray
- Templates
- Template Syntax
- Using Templates
- Using Classes with Templates
- Additional Template Features
- Standard Template Library
- 18. Appendix C – Sample Problems**
- Banking System
- Library Card Catalog
- Diagrams for Banking and Library Problems
- Object Diagram - Banking
- Event Trace Diagram - Banking
- Object Diagram – Library
- Event Trace Diagram - Library
- 19. Appendix D – Other C++ Features**
- Namespaces
- The static_cast and reinterpret_cast operator
- The dynamic_cast operator
- The const_cast operator
- mutable Data Members
- The bool Datatype
- new Operator Failure