C++ and Object-Oriented Programming

Overview

This 5-day course provides a thorough coverage of the C++ programming language. A complete sequence of working samples are used to demonstrate concepts presented in the course guide. Lab exercises are provided with detailed instructions and working solutions. If you are leveraging C++ to create applications on the job or on your own, this course will help you understand how C++ works, and immediately be more productive.

Key Learning Areas

- Learn the basic structural elements of a C++ program
- Learn a disciplined approach to program design
- Learn to compose types and implement encapsulation
- Learn the role of copy constructors
- Learn techniques for handling memory allocation errors
- Learn how to model your problem domain
- Learn the features of virtual functions and dynamic binding
- Learn the C++ exception mechanism
- Learn the RTTI mechanism
- Learn the principles behind generic programming
- Learn how to write simple template functions and classes.
- Use the iostream library for input and output
- Learn to distinguish between lvalues and rvalues
- Use move semantics to avoid copying and improve performance
- And much more…

Prerequisites

To gain the most benefit from this course, students should have some experience programming in C. Experience programming in a modern object-oriented language such as Java or C# is also sufficient

Course Outline

- **Language Primer & OO Concepts**
  - Examine the basic syntax and language constructs of a C++ program.
  - Learn how the object model provides the framework for abstraction, encapsulation and instantiation.
- **Classes in C++**
  - Use member data to represent data encapsulated in a class.
  - Use member functions to implement class' operations and provide access to its data.
  - Use the 'this' pointer to refer to the invoking object.
  - Implement an abstract data type using C++ classes.
  - Organize code for C++ classes into code files and header files.
  - Write simple test programs to exercise each member function of a class.

- **Functions in C++**
  - Use function prototypes in your code.
  - Take advantage of C++ support for strong type checking.
  - Make use of automatic conversion of parameters in function calls when there is a prototype.
  - Use inline functions.
  - Use default arguments.
  - Learn the benefits of overloading.
  - Learn the standard C/C++ call by value mechanism for passing parameters in functions calls.

- **Constructors and Destructors**
  - Learn the use and benefit of constructors.
  - Use multiple constructors in a class, including the default constructor.
  - Learn the use and benefit of destructors.
  - Simplify a class by using default arguments in a constructor.

- **Memory Management**
  - Learn the use of static, automatic (stack) and heap memory.
  - Use new and delete to manage memory.
  - Provide constructors and destructors to support dynamic objects.
  - Discuss techniques for handling memory allocation errors.

- Hide details of memory management in a class.

- **Argument Passing**
  - Use reference declarations to alias variables.
  - Use references in argument passing.
  - Learn the role of copy constructors.
  - Use constant types in your programs.

- **Operator Overloading**
  - Use overloaded operators in your code.
  - Learn the semantics of assignment.
  - Distinguish between initialization and assignment.
  - Overload the assignment operator.
  - Implement type conversions by overloading cast operators and by constructors.

- **Access Control**
  - Use C++ scoping facilities.
  - Use constants through enumeration types and through the const keyword.
  - Define "static members" and use them in your code.
  - Control access to member data and functions through public, private, and protected access specifiers.
  - Define "friend" function and explain how a friend function differs from a member function.

- **Inheritance**
  - Use inheritance to model your problem domain and achieve greater code reuse.
  - Use C++ class derivation to implement inheritance.
  - Use public, protected and private to control access to class members.
  - Use an initialization list for proper base class initialization and embedded member initialization.
  - Determine order of invocation of constructors and destructors.
  - Distinguish between use of inheritance and composition.

- **Polymorphism and Virtual Functions**
  - Learn the features of virtual functions and dynamic binding.
  - Learn pointer conversion in C++ inheritance and use pointers in connection with virtual functions.
  - Use polymorphism in C++ to write better structured, more maintainable code.
  - Provide virtual destructors for classes using virtual functions.

- Specify abstract classes using pure virtual functions.

- **Exception Handling**
  - Learn the C++ exception mechanism and contrast it with handling errors by function return codes as in C.
  - Learn the concepts of context and stack unwinding.
  - Review the automatic cleanup process that occurs with C++ exception handling.
  - Describe how matching of a thrown exception is done in the case of multiple catch handlers.

- **Runtime Type Information**
  - Learn the C++ runtime type information (RTTI) mechanism.
  - Use RTTI for special purposes in programs where the standard virtual function mechanism is not adequate.
  - Use dynamic cast to achieve type safety in working with pointer conversions.
  - Describe the C++ cast notation and discuss its benefits.

- **Templates**
  - Review the C++ template mechanism and implement programs using templates.
  - Learn how to write simple template functions and classes.
  - Understand the principles behind generic programming.
  - Implement a general array class in C++ using templates.
  - Review the basic elements of the Standard Template Library.

- **File I/O**
  - Use the iostream library for input and output
  - Use formatted I/O with iostreams.
  - Outline the inheritance hierarchy of the principal streams classes.
  - Overload operators >> and << to do I/O in your own classes.
- **Multiple Inheritance**
  - Examine variations on inheritance
  - Learn how to disambiguate
  - The importance of virtual base classes
- **R-Values and Move Semantics**
  - Compare and contrast r-values and l-values
  - Use r-value references as arguments via function overloading
  - Discuss the “rule of 5”
  - Use move semantics to allow an object to take ownership of another objects resourses