

**Course Description:** C++ is the object oriented superset of ANSI C. This course provides students with a comprehensive study of the C++ Programming Language. The course stresses the object paradigm including classes, inheritance, virtual functions, and templates in the development of C++ programs. Lab exercises reinforce the lectures.

**Who Should Attend:** Anybody who has the need to write programs in the C++ language including programmers, engineers, scientists, or other technical support personnel will benefit from this course.

**Prerequisites:** Students should have completed the Introduction to C course or have equivalent knowledge.

**Benefits of Attendance:** Upon completion of this course, students will be able to:

- Explain how object-oriented software engineering enhances the software development process.
- Identify the major elements in an object-oriented programming language.
- Implement the concepts of data abstraction and encapsulation in the creation of abstract data types.
- Implement operator overloading.
- Use inheritance in C++.
- Select the proper class protection mechanism.
- Demonstrate the use of virtual functions to implement polymorphism.
- Write programs utilizing the I/O classes in C++.
- Understand some advanced features of C++ including templates, exceptions, and multiple inheritance.
- Compare the object vs the procedural approach to writing software.
- Use correct object oriented terminology.
- Define and use classes in a C++ program.
- Create and use abstract data types.
- Derive classes using inheritance in C++.
- Implement polymorphism by using virtual functions in a program.

### Course Outline:

#### Perspective

The Software Crisis  
Building Software Has Been Difficult  
Design Techniques  
Large Software Systems  
Roots Of Object Technology  
What Is Object-Oriented Programming?  
C++ and Object-Oriented Programming  
Why C++ ?  
Features of C++  
Pros and Cons of C++?

#### The Language of Object-Oriented

What Is an Object?  
What Is A Class?  
Encapsulation  
Data Hiding  
The Public Interface  
Relationships Among Classes  
Inheritance  
Polymorphism  
Object-Oriented Design  
Exercises

#### C vs. C++

Comments  
Namespaces  
Performing Simple Output  
Performing Simple Input  
Definitions Near To First Use  
Function Prototypes  
The inline Specifier  
const  
Structure Members  
The Reference Type  
Overloading Function Names  
Default Parameters  
The Scope Resolution Operator  
Aggregates  
Operators new and delete  
The bool Data Type  
The string Data Type  
Exercises

#### Fundamentals of Classes

Data Types  
User Created Data Types  
Using The Class Concept  
Defining a class  
public and private Access  
Levels  
The Scope Resolution Operator ::  
public and private Access  
Levels (again)  
Using class Objects Like Built-in Types  
Scope  
Constructors  
Member Initialization Lists  
Destructors  
Array of Objects  
Pointers  
The this Pointer  
Passing Objects To Functions  
Returning Objects From Functions  
static Class Members  
Exercises

#### Operator Overloading

Introduction  
Rules for Operator Overloading  
Rationale for Operator Overloading  
Overloading Member Functions  
Overloading Non-Member Functions  
friend Functions  
The copy Constructor  
The Assignment Operator  
Overloading [ ]  
Overloading Increment and Decrement Operators  
const Objects and References  
Exercises

#### Composition of Classes

Relationships  
Composition Of Classes  
The Point class  
The Line class  
Member Initialization Lists  
An Application w/ Composition  
The Copy Constructor under Composition  
operator= under Composition  
Exercises

#### Inheritance

Introduction  
Inheritance - public base classes  
The protected Access Level  
Member Initialization Lists  
What Isn't Inherited?  
Assignments Between Base And Derived Objects  
Compile-Time vs. Run-Time Binding  
virtual Functions  
Polymorphism  
virtual Destructors  
Pure virtual Functions  
Abstract Base Classes  
An Extended Inheritance Example  
Exercises

#### I/O in C++

The iostream Library  
Predefined Streams  
operator<<  
Overloading << for User-Defined Classes  
Overloading >> for User-Defined Classes  
Manipulators  
Stream States  
Formatted I/O  
Disk Files  
Internal Transmission of Data  
Reading & Writing Objects  
Exercises

#### Advanced Topics

Template Functions  
Template Classes  
Multiple Inheritance  
User-Defined Conversions  
Data Structures  
An Iterator Class  
Exceptions  
Exercises

#### Introduction to the C++ Standard Template Library

Introduction  
The Standard Template Library

Design Goals  
STL Components  
Iterators  
Example: vector  
Example: list  
Example: set  
Example: map  
Example: find  
Example: merge  
Example: accumulate  
Function Objects  
Adaptors  
Exercises

#### Appendix A: Introduction

Background  
Environmental Considerations  
A Sample C Program  
Variables and Data Types  
Arrays  
Example of a Program Using an int Array  
Components of a C Program  
C Operators  
Examples of the Operators  
Control Structures  
Functions  
Function Prototypes  
Simple I/O  
Exercises

#### Appendix B: More I/O in C

The printf function  
The scanf Function  
The Preprocessor  
Conditional Compilation  
Avoiding Multiple Inclusion For The Same File  
Exercises

#### Appendix C: Aggregates in C

Data Types Revisited  
Aggregate Types  
Arrays  
Structures  
Structures and Functions  
Bit Fields

Enumeration Types  
Exercises

#### Appendix D: Pointers in C

Fundamental Concepts  
Pointer Operations  
Using Pointers to Alter a Function Argument  
Using Pointers for Array Traversal  
Pointer Arithmetic  
Sending an Array to a Function  
Command Line Arguments  
Pointers vs. Arrays  
Sending an Aggregate to a Function  
Summary of the Uses of Pointers  
Exercises

#### Appendix E: Bibliography

Bibliography