C++ PROGRAMMING FOR C PROGRAMMERS

General:

This course provides experienced C Language programmers with the skill to create object oriented programs using the C++ programming language. The student will learn the syntax of the language, as well as how to code and execute object oriented C++ programs.

Objectives:

Upon successful completion of this course, the student will be able to:

- Distinguish new C++ Language features not available in C Language.
- Compile and execute C++ programs.
- Define encapsulation, inheritance, and polymorphism.
- Use inline functions and function overloading.
- Create data abstractions through the use of classes.
- Share and restrict object members.
- Define and use constructors and destructors.
- Dynamically create and destroy space.
- Use the I/O Stream related classes.
- Use operator overloading.
- Declare and use named constants.
- Describe scope, storage class, and linkage.
- Increase software reusability through inheritance.
- Use dynamic binding and virtual functions.
- Process errors using exception handling.
- Declare and use class and function templates (Optional)

Audience:

Technical Users, Applications Programmers, and Systems Programmers.

Prerequisites:

Completion of *the Programming in C Language* course or a working knowledge of the C Programming Language. Knowledge of one of the program editors for the development environment used for the exercise sessions. An understanding of object oriented programming concepts would also prove helpful.

Duration:

Five (5) days including classroom lecture and lab sessions. Class and Function Templates unit is optional and is only presented if time permits.

C++ PROGRAMMING FOR C PROGRAMMERS
COURSE OUTLINE

I. **INTRODUCTION**

A. Features of C++
B. History and Origin of C++
C. Advantages of C++
D. C++ Compilation Process

II. **C++ vs. C**

A. Compatibility between C and C++
B. Comments and Readability
C. C++ Keywords and Modifiers
D. Variable Declarations in C++
E. Derived Data Types
F. Operator Precedence and Associativity
G. Scope/Global Operator (::)
H. Namespaces
I. Input/Output Streams

III. **FUNCTIONS**

A. Function Definition
B. Function Declaration
C. Reference Parameters
D. Default Argument Values
E. Function Overloading
F. Inline Functions
G. Type Safe Linkage
H. Name Mangling
I. Combining C and C++ Functions

IV. **OBJECT ORIENTED CONCEPTS**

A. Features of Object Oriented Languages
B. Procedural vs. Object Oriented
C. Data Abstraction
D. Encapsulation
E. Inheritance
F. Polymorphism
G. Effects of OO Approach
H. Basic OO Design (CRC Cards)

V. **CLASSES AND ENCAPSULATION**

A. Definition of Class
B. Class Syntax
C. Class Data Members
D. Class Member Functions
    1. Internal
    2. External
E. Private vs. Public Members

VI. **CONSTRUCTORS & DESTRUCTORS**

A. Class Constructors
    1. Constructors with Parameters
    2. Overloaded Constructors
    3. Internal and External Constructors
    4. Explicit and Implicit Constructor Invocation
B. Class Destructors
C. Dynamic Memory Allocation
    1. new Operator
    2. delete Operator
D. Pointers to Classes
E. this Pointer
F. Function and Class Friends
G. Copy Constructors
H. Avoiding Memory Leaks

VII. **I/O STREAMS**

A. Standard I/O Streams
B. Reading Input with cin
C. Writing Output with cout
D. Writing Errors with cerr
E. Other I/O Class Member Functions
    1. get
    2. put
    3. write
F. Manipulators
G. Simple File I/O
    1. Opening File Streams
    2. State Checking Member Functions
    3. Closing File Streams

PanTech
908-281-6759

VIII. **OVERLOADING OPERATORS**

A. Overloading Operators
B. Operator Overloading Rules and Restrictions
C. Valid Overloaded Operators
D. Overloading Binary Operators
E. Overloading Unary Operators
F. Overloading Operators Having Side Effects
G. Overloading Assignment Operator
H. Overloading ++ and -- Operators
I. Overloading Type Cast (conversion) Operator
J. Friend Operator Functions
K. Non-Member Operator Functions

IX. **CONSTANTS, SCOPE, & LINKAGE**

A. Named Constants
B. Pointers and Constants
C. References and Constants
D. Functions and Constants
E. Constant Member Functions
   1. Logically Constant Functions
   2. mutable Keyword
F. Scope and Storage Class
G. Linkage
H. static Class Members

X. **INHERITANCE**

A. Software Reusability
B. Inheritance - Concept and Terminology
C. Inheritance Syntax
D. Uses and Advantages
E. Base and Derived Classes
F. Class Access Specifiers
   1. public
   2. private
   3. protected
G. Single Inheritance
H. Multiple Inheritance
I. Constructors and Destructors
J. Containment

XI. **POLYMORPHISM AND DYNAMIC BINDING**

A. Polymorphism Definition
B. Types of Binding
   1. Static
   2. Dynamic
C. Virtual Functions
D. Rules for Dynamic Binding
E. Pure Virtual Functions and Abstract Classes

XII. **ERROR AND EXCEPTION HANDLING**

A. Exception Handling Model
B. Exception Handling Keywords
C. Generating Exceptions
D. Handling Exceptions
E. Creating Exception Classes
F. Catching or Passing Exceptions

XIII. **CLASS AND FUNCTION TEMPLATES (OPTIONAL)**

A. Template Definition
B. Template Container Class
   1. Syntax
   2. Template Arguments
   3. Overriding Template Definition
C. Global Function Templates
   1. Syntax
   2. Use
D. Advanced Template Topics
   1. Special Template Functions
   2. Inherited Template Classes