

C++ Programming, Patterns & Practices

1. OBJECTIVES

- ❖ To enable the participants to develop good object oriented design and use C++ effectively.

2. Duration

- ❖ This is a **Five Day** Program

3. Entry Profile

- ❖ Basic Programming Concepts
- ❖ Medium degree of proficiency in OO concepts
- ❖ low degree of proficiency in "C++" concepts

4. Course Contents

Module 1: **Basic structure of C++ program**

Runtime memory layout of a application

Module 2: **Functions**

Function Internals

Parameters, Arguments & return value

Template Function

Module 3: **Object Model**

SOM, TOM & BJARNE object Model

Member Function

Data Members

Module 4: **Initialization & Clean up**

Object Initialization

Initialization List

Object Cleanup

Module 5: Advanced Uses Of Constructors

Creating a C++ Library

Module 6: Memory management.

Module 7: Advanced Polymorphism And Inheritance

Module 8: Class templates in C++

Module 9: Other Topics

Detailed Course Contents

Module 1: Basic structure of C++ programme

Layout of a Executable file.

- ❖ Source file (.cpp)
- ❖ Header files (.h)
- ❖ Object files (.obj)
- ❖ Compile time v/s link time v/s Runtime
- ❖ Preprocessor Definitions/ Compile time switches
- ❖ Executable File Format

Runtime memory layout of a application

- ❖ Code Segment
- ❖ Data Segment
- ❖ Heap
- ❖ Stack

Library

- ❖ Static link library v/s Dynamic link library
- ❖ Creating a static link library
- ❖ Creating a dynamic link library
- ❖ C++ library Design

Module 2: Namespace

- ❖ The STD Namespace
- ❖ Custom Namespaces
- ❖ Namespaces and Their Limitations

Module 3: Functions

Function Internals

- ❖ Function Stack Frame
- ❖ Calling Conventions
- ❖ Naming Conventions
- ❖ Inline function
- ❖ Function prototype
- ❖ Callbacks using Function Pointer
- ❖ Overloading Functions

Parameters, Arguments & return value

- ❖ Pass by value v/s Pass by ref
- ❖ Unnamed Function Parameters
- ❖ Default function Arguments
- ❖ Variable Parameter List
- ❖ Temporary objects

Template Function

- ❖ Template Function
- ❖ Overloading vs Generic Function
- ❖ Function Template Internals
- ❖ Full Specialization
- ❖ Partial Specialization
- ❖ Library Design Issue

Module 4: Object Model

Object Model

- ❖ Simple Object Model
- ❖ Table driven object model
- ❖ C++ object model
- ❖ Class internals

Member Function

- ❖ Instance function
- ❖ Static function
- ❖ Const function
- ❖ Friend functions

Data Members

- ❖ Instance data member
- ❖ static data members
- ❖ const data members & const_cast
- ❖ static const data member
- ❖ mutable data member

Operator overloading

- ❖ Overloading unary & binary operators
- ❖ Overloading the input/output stream operators
- ❖ Operators that cannot be overloaded
- ❖ Conversion functions

Module 5: Initialization & Clean up

Object Initialization

- ❖ Compiler Synthesized Constructor
- ❖ Deep copy v/s Shallow copy
- ❖ Overloaded constructor
- ❖ Copy constructor & Generic Copy constructor
- ❖ Explicit constructor
- ❖ Copy Constructor v/s Assignment operator

Initialization List

- ❖ Initialization List
- ❖ Order of Initialization
- ❖ Initialization v/s Assignment
- ❖ Default Arguments
- ❖ Calling base class constructor

Object Cleanup

- ❖ Destructor
- ❖ Compiler Synthesized Destructor
- ❖ Preventing destroying object instance

Patterns & Techniques for construction

- ❖ Destroying instance in the constructor
- ❖ Preventing object Instance
- ❖ Preventing Stack based objects
- ❖ Preventing Heap based objects
- ❖ Identifying object location

Module 6: Dynamic Memory Management

New operator

- ❖ New vs malloc vs calloc
- ❖ Handling bad_alloc exception
- ❖ Using new with nothrow
- ❖ Placement new
- ❖ Overloading new operator

delete operator

- ❖ Delete vs Free
- ❖ Destroying objects on heap
- ❖ Destroying array of objects

Patterns & Techniques for Memory Management

- ❖ Preventing Heap based objects
- ❖ Identifying object is on Heap or Stack
- ❖ Smart pointer

Module 7: Inheritance & Containment

Patterns & Techniques using Inheritance & containment

- ❖ Containment vs Inheritance
- ❖ Private vs protected inheritance
- ❖ Hybrid Inheritance / virtual base class
- ❖ Changing scope of base member in derived class – The publicizer technique
- ❖ Friend class v/s Derived class

Virtual functions

- ❖ Virtual member function
- ❖ Layering technique, issues related to tight-coupling and loose-coupling, avoiding transitive dependencies
- ❖ Pure virtual function
- ❖ Abstract class v/s Interface v/s Concrete class

Virtual function Issues

- ❖ Calling virtual function from constructor
- ❖ Calling virtual function from destructor
- ❖ Calling virtual function from non virtual member function
- ❖ Object Slicing

Virtual Internals

- ❖ Virtual functions in Single Inheritance
- ❖ Virtual functions in Multiple Inheritance
- ❖ Virtual Inheritance & Vtables

Runtime Type Identification

- ❖ typeid function
- ❖ type_info class object
- ❖ dynamic_cast , static_cast, const_cast
- ❖ RTTI Internals
- ❖ RTTI on non polymorphic types

Techniques using Virtual functions

- ❖ Virtual destructor
- ❖ Non member virtual function
- ❖ Dual dispatching

Patterns using Virtual functions

- ❖ Bridge Pattern
- ❖ Abstract Factory Pattern
- ❖ Factory method
- ❖ Template Method
- ❖ Singleton pattern

Module 8: Templates and Generic programming

Generic Classes & Metaprogramming

- ❖ Class templates
- ❖ Class Template Full specialization
- ❖ Class Template Partial specialization

- ❖ Code Blow and Class templates

STL - Implementing Generic classes

- ❖ Standard containers (vector, stack, list, map,etc.,)
- ❖ Algorithms & Iterators
- ❖ Functors

Module 9: Exception Handling

Exception Handling

- ❖ Resumption v/s Termination
- ❖ Throwing exception
- ❖ try block
- ❖ catch block
- ❖ multiple catch blocks
- ❖ catch any block
- ❖ set_terminate functions
- ❖ custom exception class

Exception Handling Issues

- ❖ Order of catch blocks
- ❖ Catching exception by value
- ❖ Throwing exception in constructor
- ❖ Throwing exception in destructor

Post Training Reading

- 1) C++ Primer by Lippman.
- 2) Inside Objects by Lippman
- 3) C++ By Bjarne Stroustrup
- 4) C++ Bible by AL Stevens
- 5) Complete Reference by Herbert Schildt.
- 6) Advanced C++ James coplin
- 7) Effective C++ Scott mayor
- 8) More Effective C++ Scott mayor
- 9) Effective STL Scott mayor
- 10) Design Patterns (GOF)

Evaluation Criteria for the Training

- ❖ Monitoring of responses while the session is in progress, degree of interaction and Quality of doubts / questions raised is related to degree of training imbibed

Assessment Criteria for effectiveness

- ❖ Clarity of concepts, that enable the participants to develop good object oriented design and use concepts in the project
- ❖ Confidence in Engineers to tackle Design problems, by using design patterns
- ❖ Confidence in Engineers to attempt Good Design issues.