C++ Training

Trainer's name: Mahamed Ali

Program objective:

A Comprehensive Training on CPP to make audience understand C++ from internal and design perspective. A Hands-on driven approach to make them confortable with complete C++ thoughts.

Pre-requisite:

- Medium degree of proficiency in "C".
- Medium degree of proficiency in OO concepts
- low degree of proficiency in "C++" concepts

Duration of program

5-Days

H/w – S/w required for Lab Setup

Type of hardware	List of software
	Any modern CPP compiler. VS 2015 preferred.

Target Audience

People who intend to learn C++ in depth and have 0-4yrs experience with basic oops and basic C++ knowledge.

Day 1 outline

Duration

Module 1: Basic structure of C++ programme

C++ Training

- Source file (.cpp)
- Header files (.h)
- Object files (.obj)
- Complie 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

Namespace

- What is a namespace?
- Using Namespace with Scope resolution
- Using Namespace with The using directive
- Using Namespace with The using declaration
- Namespace Aliases
- Unnamed Namespace
- Namespace Composition
- Selection
- Resolving Potential Clash
- Name spaces are open

Module 2: Functions

Function Internals

- Function Stack Frame
- Calling Conventions
- Naming Conventions
- Inline function
- Function prototype
- Recursive Function
- 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
- Standard Generic Algorithms

. The standard C++ library (STL)

- Library design alternatives.
- Standard containers and sequences.
- Container adapters.
- Associative containers.
- Container-like classes.
- Strings.

Iterators, algorithms and function objects

- Iterators and compile-time polymorphism.
- Polymorpic algorithms.
- Iterator categories.
- Iterator adapters.
- Function objects and predicates.
- Standard algorithms.
- Binders and negators.

Module 3: Object Model

Object Model

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

- Class Template Full specialization
- Class Template Partial specialization
- Template Bloating

Data Members

- static data members
- const data members & const_cast
- static const data member
- mutable data member
- Instance data member
- Memento Pattern

Member Function

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

Function Pointers

- Callbacks using Function Pointer
- Synchronize v/s Asyncronize calls
- Observer Pattern

Container class

- Creating Container Class
- Iterator Pattern
- Standard containers(vector, stack, list, map, deque, set, multimap, multiset)
- Composite Pattern
- Command Pattern

Day 2

Module 4: Initialization & Clean up

Object Initialization

- Compiler Synthesized Constructor
- Deep copy v/s Shallow copy
- Overloaded constructor
- 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
- Prototype Pattern
- Builder Pattern

Module 5: 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

Techniques for Memory Management

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

Patterns for Memory Management

- Reference Counting
- Singleton pattern
- Creating Fly Weight objects

Day 3

Module 6: Inheritance & Containment

Techniques using containment

- Containment
- Composite Pattern
- Friend class
- Nested Class

Techniques using Inheritance

- Template Pattern
- Containment v/s Inheritance
- Private v/s protected inheritance
- Changing scope of base member in derived class
- Derived class

Patterns using Inheritance & containment

- Runtime Inheritance using Decorator pattern
- Changing Parent class using State Pattern
- Adopting to changes using Adopter Pattern
- Change implementation at Runtime using Strategy Pattern
- Reducing Complexity in Relationship using Mediator Pattern

Module 7: Virtual

Virtual functions

- Virtual member function
- 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

Runtime Type Identification

- typeid function
- type_info object
- dynamic_cast vs static_cast
- RTTI Internals
- RTTI on non polymorphic types

Techniques using Virtual functions

- Proxy pattern using reinterpret_cast of interface pointers
- Virtual destructor
- Virtual constructor using Prototype pattern
- Non member virtual function
- Dual dispatching
- Multi dispatching

Patterns using Virtual functions

- Template Method
- Bridge Pattern
- Abstract Factory Pattern
- Adding Functionality horizontally using Visitor Pattern

Day 4

Module 8: Exception Handling

Exception Handling

- Resumption v/s Termination
- Throwing exception
- try block
- catch block
- multiple catch blocks
- catch any block

- set_terminate functions

Exception Handling Issues

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

Advanced Exception Handling

- standard exceptions
- creating custom exception class
- Exception handling internals
- Exception Handling Performance Issues
- Chain of Responsibility Pattern
- Interpreter Pattern

Extensions to the C++ core language

Core language runtime performance enhancements

- Rvalue references and move constructors
- constexpr – Generalized constant expressions
- Modification to the definition of plain old data

Core language usability enhancements

- Initializer lists
- Uniform initialization
- Type inference
- Range-based for loop
- Lambda functions and expressions
- Alternative function syntax
- Object construction improvement

Day 5

- Explicit overrides and final
- Null pointer constant
- Strongly typed enumerations
- Right angle bracket
- Explicit conversion operators

C++ training

- Alias templates
- Unrestricted unions

Core language functionality improvements

- Variadic templates
- New string literals
- User-defined literals
- Explicitly defaulted and deleted special member functions
- Type long long int
- Static assertions
- Allow sizeof to work on members of classes without an explicit object

C++ standard library changes

- Upgrades to standard library components
- Tuple types
- Hash tables
- Regular expressions
- General-purpose smart pointers
- Extensible random number facility

concurrency in C++!

- What is Concurrency?
- Why Use Concurrency?
- Concurrency and Multithreading in C++
- A Simple Threading

❖ Managing threads

- Basic thread managment
- Passing arguments to a thread function
- Transferring ownership of a thread
- Choosing the number of threads at runtime
- Identifying threads
- Sharing data between threads
- Protecting shared data with mutexes

Training Reading

C++ training

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

- Test will be conducted for all participants at the end of 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.