SE002CPP: *C++ Advanced*

DURATION: 5 Days

WHAT YOU WILL LEARN
This five-day instructor-led course provides to teach the details of C++ language

This course designed for programmers to explore the advance features of C++.

The approach used is theory followed by powerful seriously designed exercises to reinforce the understanding of the theory.

Besides the language syntax, many aspect of the course are intended to make the participants as better software developer, especially using C++ as programming tool. Many of the aspects covered in this course will benefit the participants in many other project developments

Upon completion of this course, participants should be able to:

- Apply different advance programming constructs in C++ appropriately.
- Appreciate some fundamental OOP concepts.
- Use C++ syntax in dealing with OOP.
- Realise the pitfall and precaution when using C++.

PREREQUISITES
Participants should have completed the basic introductory to C++ programming. Especially those non-object oriented features of C++.

COURSE OUTLINES

Module 1: *Basic Object-oriented concepts*
This module explains the fundamental Object-oriented concepts that are related to in C++ programming.
Lessons

- From modular concept to OOT: *The paradigm shift*
- Object
- Identity
- Attributes
  - Attributes Value
  - State
- Behavior
- Relationship
- Operation
- Method
- Abstraction
- Classification
- Encapsulation
- Class
  - Superclass
  - Subclass
- Abstract Class
- Taxonomy
- Generalization
- Specialization
  - Restriction
  - Extension
  - Overriding (*Dynamic Polymorphism*)
- Multiplicity
- Inheritance
- Multiple Inheritance
- Instantiation
  - Instance
  - Direct and Indirect Instance
- Information Hiding
- Polymorphism
- Part-Of relationship
  - Aggregation
  - Composition
- Foundation Classes

Lab: *Basic OOP concepts*

- Exercise 1: *Defining Taxonomy*
- Exercise 2: *Identify relationships*

After completing this module, students will be able to:

- Know the important of new OO development paradigm.
- Understand the common terminologies used in OOP.
- Apply C++ syntax in implementing the OO taxonomy
- Learn some techniques in identify various relationships in OO technology
- Appreciate the value of “Information Hiding” as software development strategy to confront with the complexity.

Module 2: OOP Supports in C++
This module explains the fundamental Object-oriented concepts that are related to in C++ programming.

Lessons

- From **struct** to **class**
- Access Modifiers: **private, protected**, and **public**
- Data members
- Member Functions
- Object Instantiation
- **this** keyword
- **static** members
  - *“I call see you but you can’t call me”*

Lab : *Basic OOP features in C++*

- Exercise 1: *Circle example*
- Exercise 2: *Who can see what?*
- Exercise 3: *Bank Account*

After completing this module, students will be able to:

- Understand the relationship between **struct** and **class** in C++.
- Use **this** keyword appropriately
- Define members of the class instances.
- Appreciate the use of members of the class.

- Avoid the reference to instance members from the class member functions.

Module 3: *Constructors and Destructor*
This module will cover the instance constructors and destructor.
Lessons
- Constructors
- Default constructor
- Parameterless Constructor
- Use of member initialization list
- Destructor
- Default Destructor

Lab: *Object Instantiation*
- Exercise 1: *Person class*
- Exercise 2: *Student class*
- Exercise 3: *MyString class*
- Exercise 4: *Object Left Cycle*

After completing this module, students will be able to:
- Understand various object instantiation forms in C++.
- Apply constructor(s) and destructor to the objects.
- Determine the objects life cycle.

Module 4: *Inheritance*
This module will cover the support of Inheritance in C++.
Lessons
- Inheritance: The support of reusability
- How to and why to involve superclass constructor
- Multiple inheritance
- Conflict resolution
- **private**, **protected**, **public**, and **virtual** inheritance
- Generic reference

Lab: *Reusability via inheritance in C++*
- Exercise 1: *Staff class*
- Exercise 2: *Batman class*
- Exercise 3: *Manager related classes*
- Exercise 4: *Inheritance types*

After completing this module, students will be able to:
- Understand how inheritance promotes reusability in C++.
- Know how and why to involve superclass constructor
- Resolve conflict in multiple inheritance
- Apply the various type of inheritance appropriately by knowing the basic rules.

Module 5: *Polymorphism*
This module will cover the concept of polymorphism in C++ and its related issues.
Lessons
- Function and operator overloading
- Rules for Operator Overloading
- Rationale for Operator Overloading
- Overloading Member Functions
- Overloading Non-Member Functions
- Overloading Type Cast (conversion) Operator
- Overloading operator **new** and **delete**
- Overloading **[ ]**
- Overloading Increment and Decrement Operators
- Polymorphism and virtual functions
  - Static Vs dynamic instantiation
- The late binding issues
  - Code efficiency
  - Object size

Lab: *Polymorphism in C++*
- Exercise 1: *Matrix class*
- Exercise 2: *Handle out of bound*
- Exercise 3: *Sales Manager class*

After completing this module, students will be able to:
- Understand the two forms of polymorphism supported by C++, namely static and dynamic polymorphism.
- Apply operator overloading appropriately to simplify the code
- Use advance operator overloading
- Appreciate what are the early and the late binding about.
- Avoid the late binding issues

Module 6: *Friend functions*
This module will explain the use of the friend functions in C++.
Lessons
- Why use friend functions
  - Scenarios
- Syntax
- Access privilege
- Avoid violation of Information Hiding principal

Lab: *Why giving friend status to external function?*
- Exercise 1: *The improve version of Matrix class*

After completing this module, students will be able to:
- Appreciate the reason of using friend function
- Apply friend function to avoid violation of Information Hiding principal

Module 7: *Pure Virtual function and Abstract Class*
This module will explain the concept of abstract class.
Lessons
- Abstract Class and how it happened?
  - The story of ***What*** and ***How***
- The Pure Virtual Function
- Prohibited Instantiation
- Methods enforcement

Lab: ***Use Abstract method instead of polymorphism***

- Exercise 1: *Revisit the Staff class*

After completing this module, students will be able to:

- Appreciate the reason why abstract class occurs
- Understand the concept of pure virtual function and its method enforcement

Module 8: ***Object copying Issues***
This module will explain the issues of object copying.
Lessons

- Deep Vs. shallow copying
- Default objects initialization from existing objects.
    - Copy Constructor
- Object assignment
    - overload operator=
    - use assignment operator in the condition

Lab: ***Deep copy versus shallow copy***

- Exercise 1: *The issues of String class*

After completing this module, students will be able to:

- Understand the default copying behavior of object initialization and assignment from existing objects.
- Create Copy constructor
- Overload the assignment operator with complete syntax

Module 9: ***Inline functions***
This module will cover the Inline function in C++.
Lessons

- Inline functions
- Inline functions Vs. macro functions
- Inline member functions
- Method implementation out of the class
    - Reason
    - Syntax for explicit inline declaration

Lab: ***Understand inline functions***

- Exercise 1: *inline Vs macro functions*
- Exercise 2: *Multiple modules project*

After completing this module, students will be able to:

- Understand the different and advantages of using inline
- Appreciate the default inline support of the method declaration in the class
- Use explicit declaration to preserve the inline for member functions

Module 10: ***Template Library***
This module will cover the concept of template library in C++.
Lessons

- Template - Parameterize Types
- Template functions
- Specializing a template function
- Template classes
- An array template class
- Instantiating a template class object
- Rules for templates
- Non member function with a template argument
- Friends of template classes
- Templates with multiple type parameters
- Comments regarding templates
- Using export
- Member Templates
- STL (Standard Template Library)

Lab: ***Understand inline functions***

- Exercise 1: *Template functions*
- Exercise 2: *Stack class*

After completing this module, students will be able to:

- Understand the advantages of using template function and class
- Apply template concept in reducing coding efforts
- Use standard template library in C++

Module 11: ***Exception Handling***
This module will explain the exception concept and behavior, followed by showing how to implement in C++.
Lessons

- Exception Handling
    - Modern runtime environment
    - When to apply
- Traditional approaches to error handling
- try, catch, and throw
- A simple exception handler
- Multiple catch blocks
- The exception specification list
- Rethrowing
- Cleanup
- Exception matching
- Inheritance and exceptions
- Catch by reference
- Standard exceptions
- Asynchronous Exceptions

Lab: ***Exception handling***

- Exercise 1: *illegal user input*
- Exercise 2: *Improved Person class*

After completing this module, students will be able to:

- Understand the behavior of code when exception occur
- Apply handling appropriately
- Throw exception when appropriate
- Create custom exception object

- Handle multiple exceptions

Module 12: *Namespace*
This module will explain the concept of namespace in C++.
Lessons
- Namespace Definitions
- Unnamed Namespaces
- Using Directives
- Using Declarations
- Namespace Aliases
- The namespace std
- The namespace anonymous

Lab: *Exception handling*
- Exercise 1: *Using namespace*

After completing this module, students will be able to:
- Understand the namespace is about
- Apply it appropriately

Module 13: *I/O Streams*
This module will cover the I/O stream in C++.
Lessons
- 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

Lab: Using I/O Stream
- Exercise 1: *Streaming*
- Exercise 2: *overload << and >>*
- Exercise 3: *File access*

After completing this module, students will be able to:
- Use standard I/O stream classes
- Overload <, and >> operators
- Read and write disk file