Sela.

C2

Object Oriented Programming in C++

college@sela.co.il 03-6176666

Object Oriented Programming in C++

C2 - Version: 4.1

7 days Course

Description:

This course teaches the Object-Oriented and Multi Paradigm approaches through deep learning and practicing of the C++ Language. Many examples are provided to emphasize the possibilities and advantages of using the new paradigms for characterizing, designing, and programming medium to large projects. The course deals with different advanced aspects including Generic Programming (Templates), STL, Inheritance, Polymorphism, Exception Handling and much more. The course presents many complete examples in C++, including various exercises.

Intended audience:

This course is intended for project managers, project designers, programmers, and engineers wishing to master the C++ programming language.

Prerequisites:

Experience in software development
Working knowledge of the C programming language

Objectives:

Design solutions for real life problems
Familiarity with the main mechanisms of the C++ language
Understanding of the Object Based and Object Oriented programming ideas
The participant will learn recommended methods for using the new tools that the C++ language offers to programmers

Topics:

Object-Oriented Development Introduction

- What are the benefits of OOD? How does it meet the challenges of software programming today?
- The theoretical background of OOD.
- Basic OOD concepts – Information Hiding; Message vs. Method; etc.
- What is an object? What is an OO application structure and how does it act?
- Classes - as a factory of Objects.

C++ First Steps

- C++ language – Formal details.
- Main characteristics.
- C compatible.
- Strongly typed.
- Multi paradigm language.
- Classes Definition:
  - Data members
  - Methods
  - Public Vs. Private
- Methods definition.
- Defining and activating objects.
- Using the C++ I/O library.

C++ Goodies

- General Features added to the language

- Reference
- Const
- Function Overloading
- Default arguments
- C++ Casts
- Namespace
- bool Type

Object based programming

- Constructors and Destructors; Composed Classes; Copy Constructor
- Operators Overloading - Relational; Assignment; Arithmetic.
- Static Members
- User defined conversions
- Member vs. Non-member operators - I/O operators; Symmetric operators
- Friend
- efficiency Issues – Possible efficiency problems; Optimization

Exception Handling

- Throwing exceptions
- Handling exceptions
- Resource Management in EH environment
- Exceptions Specification
- Exception Handling and Efficiency
- Exception Handling and General Project Design Consideration
- The Standard Exceptions Hierarchy

Generic Programming Templates

- Generic Programming motivation – the importance of code reusability
- How to create Generic code in C++ - Definition and use of Functions and Classes Templates.

Getting Ready to STL

- Containers – some basic concepts; Nested Classes; The concept of Iterators

STL The Standard Templates Library

- The benefits of using STL.
- General Overview.
- Containers.
- Iterators.
- Algorithms.
- The ways to customize STL functionality.

Object Oriented Programming

- Inheritance – The Basics
- The Inheritance relation definition
- Defining Derived Classes
- Extending and Overriding Default behaviour - Virtual Methods
- The Protected accessibility level
- Polymorphism
- The Polymorphism Idea
- Base – Derived compatibility
- Creating Polymorphic Code – Examples
- Utilizing Inheritance in Development
- Creating an inheritance tree – distributing data members declarations; methods declarations and methods definitions

- Pure virtual methods and abstract classes
- Construction and Destruction of derived objects – Virtual Destructors
- Interface Classes
- Dynamic Binding
- Static vs. Dynamic Binding; How is dynamic binding implemented? What are the “prices”? – Storage and run-time
- Inheritance and Polymorphism – Advanced Topics
- Inheritance and static data members; Redefining non-virtual Methods; Object slicing; Private and Protected Inheritance;Overloading between base and derived; etc.
- Multiple Inheritance – When is it required? Possible ambiguity problems; The problematic “diamond structure” - Virtual Inheritance
- RTTI – Run Time Type Information – When is it required? The type_id operator; The dynamic_cast operator