The C++ Programming Language

Level: foundation

Length: 35 - 40 hours

Course Objective: a solid introduction to the C++ programming language, learn the main principles and mechanisms of object oriented programming, how they are supported and used in C++

What You Will Learn

- How the programs are organized, development cycle of writing software, using an integrated development environment (IDE)
- Code structure: data structures, control structures, fundamental, primitives types
- C++ support for object oriented programming (OOP): data abstraction, relations between types, polymorphism, generic programming
- Exercise OOP paradigm
- Enhance soft skills: communication, team work, presentation

Who Can Attend

- Programmers who come from procedural programming (for example C) and want to enter the object oriented programming world by using C++
- Programmers who already use other object oriented programming languages (like Java, C#, Python, etc.) and want to learn C++

Prerequisites

- Comfortable to using the host operating system
- Knowledge of the C programming language at least at medium level, or knowledge of another object oriented programming language like Java, C#, etc.

Required Facilities: VGA projector, white board, workstation, C++ development tools. It's highly recommended using an IDE, good (free) examples are Microsoft Visual C++ Express Edition or Eclipse.

Minimal Bibliography: The C++ Programming Language, Fourth Edition, Bjarne Stroustrup, Addison-Wesley, ISBN 0-321-56384-0

Related Courses: The C programming language, Advanced C++ Programming, Object Oriented Analysis and Design, Design Patterns

Description

It is shown the place of object oriented programming among other programming paradigms, introduce the notion of class (user defined type) and object, types of relationship between classes, polymorphism, generic programming, error handling, operator overloading, and standard C++ library - Standard Template Library (STL).

This training teaches and exercises the object oriented programming principles: abstraction, information hiding, inheritance, aggregation, encapsulation, polymorphism, generic programming.

The theoretical notions are practically demonstrated and exercised by working examples and projects. At least 50% of the time budget is dedicated to hands on exercises and projects.

Note: the subjects, the practical part is adapted to the atendees' profile, their background, experience and goals.

Contents

1. Introduction of OOP and C++
2. Programming paradigms: procedural, modular, object oriented programming, functional
3. Outline of C, common issues; new elements extra classes & objects like references and name mangling
4. Support for data abstraction: object initialization and destruction, object assignation, templates, exception handling, type conversions
5. Support for OOP: abstract and concrete classes, multiple implementations, virtual functions and how are they implemented, multiple inheritance
6. Objects and classes: class definition, class usage, access levels to the class members, class scope, defining a class inside another class, incomplete declaration of a class, data members, static data members, objects as data members, pointers to data members, member functions, static member functions, inline methods, methods with constant this, constructors, constructors for classes with object data members, private constructors, default constructors, constructors with arguments, copy constructors, destructors, private destructors, friend to a class
7. Inheritance, simple inheritance, polymorphism, functional closure, multiple inheritance, virtual base class, dominance rule, scope resolution operator
8. Overloading, type conversions by using overloading, operator overloading, operators as function calls, operator overloading as member functions, operator overloading as friend functions, function call operator, index operator, assignment operator, limitation of operator overloading, prefix and postfix unary operators, member access by using ->, overloading of new and delete operators
9. Polymorphism, early and late binding, virtual functions, null virtual functions, abstract classes, vptr and vtab
10. Streams, files, formatting, filters
11. Standard Template Library (STL), containers, iterators, traversals and predicates, algorithms