

## The C++ Programming Language

**Length:** 20 – 40 hours

**Required facilities:** VGA projector, white board, one workstation (per two programmers), C++ development tools (text editor, compiler, linker, debugger, standard C++ libraries including STL). It's highly recommended using an IDE, a good (free) example is Microsoft Visual C++ Express Edition.

**Who can participate:**

- 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 or C#) and want to learn C++

**Prerequisites:** knowledge of the C programming language saves the time usually spent with the common issues of C and C++

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

**Related courses:** Object Oriented Analysis and Design, Design Patterns

**Attendees' evaluation:** optional, during the training and/or a final test

**Bibliography:** The C++ Programming Language, Third Edition, Bjarne Stroustrup, Addison-Wesley, ISBN 0-201-88954-4

**Description:** this course is mainly addressed to C programmers who want to enter the object oriented programming (OOP) world by using C++.

It's shown the place of OOP 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, standard C++ library - Standard Template Library (STL).

This training will exercise the OOP principles: 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.

## Contents:

1. Introduction of OOP and C++
2. Programming paradigms: procedural, modular, data abstraction, object oriented programming, functional
3. Comparison with C, common issues
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 static 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. Polymorfism, 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