

**Price:** R8,600.00 excl. VAT  
**Duration:** 5 days  
**Code:** SCPLS

# Standard C++ Programming

## Description

The Standard C++ Programming course provides a compiler-independent introduction to the ANSI C++ language for C programmers. This course covers all aspects of the C++ language, with the emphasis on portability. It includes an introduction to object oriented design techniques.

## Objectives

Delegates who complete the Standard C++ Programming course will be able to:

- Develop portable, standards compliant code using C++ language constructs.
- Read, debug and maintain C++ code.
- Understand object-oriented concepts as they pertain to C++.
- Implement error-handling techniques using exception handling.
- Understand generic programming and the use of templates in C++.

## Intended Audience

The Standard C++ Programming course is intended for C programmers who want to move to C++ and object-oriented development.

## Prerequisites

Delegates must have a good C foundation and previous experience programming with C. Our Standard C Programming course and some experience.

## Course Contents

*The lecturer reserves the right to modify the contents of the course to suit the needs of the delegates.*

**A Different and Enhanced C** • Const types. • Character literals and char arrays. • Structured and enumerated types. • Empty parameter lists. • Void pointers. • C++ comments. • Type checking. • Operators as functions. • Default parameters. • Inline functions. • References. • Dynamic memory allocation. • Exception handling. • Scope resolution. • Namespaces.

**Object Oriented Programming Basics** • Data encapsulation. • Structured programming and OOP. • Classes and objects. • Inheritance and abstract data types. • Polymorphism.

**C++ Standard Library Overview** • Streams and overloaded stream operators. • I/O Manipulators. • Strings and the STL. • Collections and iterators. • Generic algorithms.

**Classes and Objects** • Member functions, constructors and destructors. • Constructor overloading. • Copy constructor. • Canonical form for classes. • Overloaded assignment operator. • Shallow versus deep copies. • Dynamic memory allocation in constructors.

**Data Members** • Structured type members. • Access specifiers. • Static members.

**Expressions and Functions** • Expression evaluation. • C++ operators and precedence. • Implicit type conversion. • Function signatures. • Overloading functions. • Variable argument lists. • Pass and return by references.

**Templates and the STL** • Templates and generic functions. • Deriving from template classes.

**Friends, Operators and Member Functions** • Friend functions. • Overloading standard operators. • The this keyword. • Inline member functions. • Scope resolution. • Static functions.

**Inheritance** • Base and derived classes. • Initializer lists. • Public, private and protected inheritance. • Polymorphism in detail. • Multiple inheritance. • Virtual base classes.