

Effective C++ Programming

Duration: 5days

The C++ Language Basics

A quick contrast on features of C++ & C

A simple C++ program

A C++ build Phase

Exploring GNU Tool chains for debugging skills

Cascading of I/O operators

Type compatibility

Scope resolution operator

Pointers and arrays

Const qualifiers with pointers

Storage classes

Memory Allocation

new

delete

Functions in C++

Strict prototyping

Inline functions

Default arguments

Return by reference

Function overloading

Pass by value, address and reference

Classes and Objects

Class declaration & object mechanism

Access specifiers

Separating interface from implementation

Inline member functions
Nested member functions
Memory allocation for objects
Objects & references

Constructors and Destructors

Definition & declaration
Characteristics of constructors
Overloaded constructors
Copy constructor
Destructors
Dynamic constructors
Array of objects
Use of constructors
Constructors and Destructors
Managing the heap
Constructors & Destructors – internal behaviour
Passing objects
Objects & references
Copy constructor
Overloading copy constructor
Objects & references
Copy constructor
Overloading copy constructor
Returning objects
Returning object references
Passing & returning pointers to objects

Static members and objects on the heap

Static members of a class

Static data members

Static member functions

A discussion on static members

Objects on the heap

The this pointer

Friend and classes

Object communication

Friend functions

Friend classes

Const objects and const member functions

Object composition & destruction mechanism

Boundary classes & object interfaces

Operator Overloading

The operator function

Operators that cannot be overloaded

Overloading unary operators

Overloading Binary operators

Using member functions & friend functions – when to use what

Interpreting the operator function call

Function overloading the operator function

The assignment operator

Implicit overloading mechanism

Explicit overloading mechanism

Returning objects & assignment operator

Overloading >> & << operators

Cascading operators & returning reference

Overloading new & delete operators

Special operator overloads

operator == , operator [], operator (), operator ->, operator comma (,)

Inheritance

The inheritance mechanism

Types of inheritance

Single level, multi level, multiple, hierarchical, hybrid

Inheritance – is it allocation or accessibility

The protected access specifier

Inheritance in private, protected & public mode

Constructors & destructors in a derived class

Multiple inheritance

Virtual Base class

Invoking base class constructors

Why a constructor is not inherited

Is-a & Has-a relationship

Nested classes & Containership

Runtime Polymorphism, Virtual Functions & Dynamic Binding

Pointers & Classes

Pointers to Class Members

Method overriding

Base class pointers & derived class objects

Virtual functions

Virtual functions & base class pointers

Virtual functions & base class reference

Dynamic Binding v/s Early Binding

Pure virtual functions
Virtual destructors
Abstract classes & applications

Runtime Type Identification & Casting Operators

RTTI

Runtime Polymorphism & RTTI

typeid & type_info

Application illustration using

Base class pointer

Passing pointers to functions

Function receiving references

Factory methods & RTTI

Template classes & RTTI

The Casting operators

dynamic_cast

const_cast

static_cast

reinterpret_cast

Exception Handling

Exception handling fundamentals

try, catch & throw

Using multiple catch

The 'catch all' exception handler

Nested try blocks

Rethrowing an exception & overruling a function's return value

Restricting Exceptions – the throw list

Handling derived class exceptions

Setting the terminate and unexpected handlers

Application of Exception handling

Templates

Generic functions & Data abstraction

Function templates

Explicitly specializing a function template

Overloading Function Template

Using standard parameters

A Generic Sort algorithm

Generic Classes (Class template)

Using more than one generic type

Using non-type arguments & default arguments

Explicit Specializations

Applications of templates - a Stack template

Template template parameter

Design of STL

Techniques of Usage

Note: This is a generic content & can be customized for individual needs!