Modern Expressive C++

Learn via: **Classroom / Virtual Classroom / Online**

Duration: **5 Gün**

https://bilginc.com/tr/egitim/modern-expressive-cplusplus-758-egitimi/

Overview

C++ is undoubtedly one of the most popular programming languages for software development. It brings language enhancements and object-oriented programming support to the extremely popular language C . However, C ++ is a large and sometimes difficult language, and even with a C or object oriented background, a programmer needs to understand C ++ programming style as well as C ++ constructs to get the best out of it.

The course is written from a developers rather than an academics perspective, providing a thorough practical coverage of the language. It aims to eliminate common misconceptions and poor programming practice by teaching the features of the language and standard library that enforce good practice.

In particular the course teaches the Modern C++ approach, to deliver clear expressive and efficient code. Although C++11 and more recent additions to the language are taught throughout, most of the material is useful and relevant to pre C++11 users.

This is a hands-on course with a mix of tuition and practical sessions for each technical chapter which reinforce the Modern Expressive C ++ programming techniques covered in the course. Delegates will write unit tests to verify their work as they develop a GUI based tool to support their learning.

Prerequisites

- Delegates must have solid experience of programming, with a clear understanding of functions and multiple source file projects. A basic understanding of Object Oriented principles is assumed. Those coming from C, C# or Java will have an advantage.
- Delegates with no recent programming experience should attend the Programming Foundations course first.
- Experienced C++ programmers should also consider the QA course Modern Robust C++ Development.

What You Will Learn

- Define and use data types.
- Declare, define and call functions.
- Use pointers, smart pointers, dynamic memory and object lifetime.
- Understand the importance and application of const consistency.
- Understand the key concepts and vocabulary of object orientation.
- Implement classes.
- Provide inward and outward conversions to UDT's.
- Build new classes from other classes using composition and aggregation.
- Build new classes from other classes using inheritance.
- Design and write code with polymorphic behaviour.
- Use container classes and templates.
- Make extensive use of algorithms.
- Write code that is efficient and robust

Outline

Chapter 1 – Introduction

- Style and Approach

Chapter 2 – Language Overview

- Why use C++?
- Language Distinctives
- Classic v Modern C++
- File Structure
- Online Compilers

- Hello World
- Identifiers
- Keywords
- Declarations
- Definitions
- Expressions and Statements
- Member Access
- Operators
- Layout

Chapter 3 –Variables and Functions

- Mutable and Immutable Variable types
- Auto Variables
- Brace or List Initialization, Uniform Initialization
- Scope Types
- Lifetime
- Namespaces
- Name Hiding
- Scope Resolution
- Function Prototype
- Parameter Types
- Reference
- Function Return Types
- Trailing Return
- Header Files
- Function Parameter Defaults
- Function implementation
- Inline function
- Source-code Implementation
- Deduced Return Type
- Anonymous Return
- Un-named arguments

Chapter 4 – Collections

- Arrays
- Array Initialisation
- Array Behaviour
- Arrays as Arguments
- std::array
- Vector Basics
- Enumeration – Range For-loop
- File System Basics
- File-Streaming

Chapter 5 – Types and Const Qualifiers

- Primitive Types
- Uninitialised Values
- Type Aliases
- This-pointer
- Const Objects
- Const Function Parameters
- Queries and Modifiers
- Free Functions

Chapter 6 – Foundational Design Principles

- Encapsulation
- Private Access
- Things that Break Encapsulation
- Single Responsibility Principle
- Expressiveness
- Expressive Names
- Resource Acquisition Is Initialization – RAII

Chapter 7 – Literals and Strings

- Literals and Magic-Numbers
- Numeric Limits
- Strings
- Tokenization
- stringstream

- Formatting Streams

Chapter 8 – Flow Control

- Boolean
- Conditions & Boolean Operators
- If - Else
- Range-For
- Counted For-Loop
- While-loop
- Enum
- Switch
- Cost of testing and branching

Chapter 9 – Header Files

- One Definition Rule
- #define
- constexpr literals
- Precompiled Header Files

Chapter 10 – Unit Testing

- What should be tested?
- Project Arrangement
- Unit Tests
- Compiling Source Code to Use Libraries

Chapter 11 – Iterators

- Operators
- Prefix / Postfix Increment / Decrement
- Iterators
- Member Access Operators
- Moving Iterators
- Query Iterators
- Iterator Position

Chapter 12 – Pointers

- Naked Reference
- Problems with Null-Pointers
- Query Pointers
- Array Pointers
- Aggregate Pointers
- Function Pointers
- void, void pointers
- Pointers as Iterators

Chapter 13 – Zero-Cost Abstractions

- Encapsulating Concepts
- Enumerated Type, enum
- Bitwise (Flag) Enums

Chapter 14 – Lambdas

- How Expressiveness Breaks Encapsulation
- Locally defined function
- Lambda Syntax
- Capture List
- Argument List
- Return Type
- Lambda Body
- Lambda Closure as Function Pointer
- Inlined Lambda
- Immediately Invoked Lambda
- Lambdas in Header Files

Chapter 15 – Algorithms

- Non-Range Algorithms
- Range Algorithms
- Algorithm Examples
- back_inserter

- Loops and Folds
- Accumulating Strings

Chapter 16 – Inline and Extern

- Symbol Tables
- Header Files Defining Objects or Functions
- Inlining
- Extern

Chapter 17 – Container Types

- Compile-time sized containers
- Dynamically sized Sequential Containers
- Dynamically Sized Associative Containers

Chapter 18 – Type Conversions

- Implicit Type Conversions
- Keyword Casts
- Conversion Constructor – Inward Conversion
- Class Operators
- Conversion Assignment
- Conversion Operator - Outward Conversion
- Allowable Conversions

Chapter 19 – Function Overloading

- Function Overloading
- Overloading on Const
- Function Delegation
- Inlined Function Definitions
- Template Functions
- Template Classes

Chapter 20 – Classes

- Programming Paradigms
- Object Oriented Programming (OOP)
- Class Definition
- Public Interface
- Class Header File
- Construction
- In-Class Defaults
- Default Constructor
- Destruction
- Synthesised Functions
- Member Function Implementation
- Member Initializer List (MIL)
- std::initializer_list Constructor
- Static Storage
- Class Static Data
- Class Static Functions
- Struct

Chapter 21 – Inheritance

- Inheritance Hierarchy
- Liskov substitution
- Public Inheritance
- Function Specialisation
- Protected Access
- Private Inheritance
- Accessing Base Class Members
- Multiple Inheritance
- Base Class Construction
- Constructor Delegation
- Generalisation
- Abstract Classes

Chapter 22 – Polymorphism

- Virtual Functions
- Override
- Virtual Destructor

- Pure Functions
- Pure Abstract Class (Interface Class)
- Null-Objects

Chapter 23 – Association

- Association
- Composition
- Inheritance
- Aggregation
- Object Creation for Aggregation
- new and delete
- Array New & Delete
- New and Delete Problems
- Expressive Lifetime Management
- unique_ptr
- shared_ptr
- Safe Aggregation
- Multiple Association
- Friendship