Formation Embedded Modern C++ Programming: Embedded Modern C++ Programming - Cours en ligne: Langages

oL10 - Embedded Modern C++ Programming

Objectives

- Discover the modern C++ features
- Learn the language changes in C++11, C++14 and C++17
- Discover the new functionalities added to the standard library
- Learn advanced modern C++ features like perfect forwarding
- Moving to from traditional C++ to modern C++
- Putting in evidence the essential modern C++ features used in embedded application

Labs are conducted QEMU ARM-based board

Prerequisite

- C programming skills (see our L2 course)
- Basic C++ Language knowledge (see our L3 course)

Course environment

- Dematerialized course material (in English)
- One Virtual Linux PC for two trainees.
- Eclipse environment and GCC compiler
- Downloadable preconfigured virtual machine

Duration

- Total: 20 hours
- 4 sessions, 5 hours each (excluding break time)
- Some Labs must be completed after the session

Plan

First Session

C++ programming overview

- History and definition
- Overview on C++98/C++03/C++11/C++14/ C++17 and C++2x
- Modern C++ objectives
- Object Oriented Programming in C++ Overview
  - Object construction and destruction
  - Construction parameters
  - Copy constructor

- Operator Overloading
- Inheritance
  - Inheritance polymorphism
  - Virtual methods
  - Virtual inheritance
- Generic objects and templates
- Generic library: STL
- Build automation tools
  - Makefile
  - CMake

Exercice : *Understand function mangling*
Exercice : *Overloading operators*
Exercice : *Understand inheritance*
Exercice : *Understand multiple inheritance and virtual bases*

Introduction to modern C++

- Overview
- Storage class specifiers
- Uniform initialization
- C++ Named Requirements
- Automatic type deduction
  - The auto keyword
  - The auto keyword as a return type from a function
  - Using auto for declaring function signatures
  - Automatic constant references
  - Forwarding references
  - Advantages of using auto in embedded systems

Exercice : *Using auto to declare variables*

Second Session

Keywords

- Enum class
- override and final
- Inline variables
- nullptr
- static_assert
- noexcept
- constexpr and if constexpr
- decltype
- Defaulted and deleted functions
  - Implementing a thread-safe singleton

Exercice : *Using modern C++ keywords*
Exercice : *Create a singleton using modern C++*

New functionalities

- Structured binding
- Range-based for loops
- Nested namespaces and namespace aliases
- Alignment
  - Alignas
  - Alignof
- Move semantics and r-value references

- Copy-constructing and Move-constructing
- r-value references
- Perfect forwarding

Exercice : *Using the new for loop syntax*
Exercice : *Using std::tuple*
Exercice : *Move semantics performance advantages on embedded systems*

Third Session

Modern C++ Standard Library

- Standard Library
  - std::optional
  - std::variant
  - std::any
  - std::byte
  - std::hash
  - Filesystem library
- Literals
  - Cooked literals
  - Standard literal operators
  - Raw literals
  - Raw string literals
- Random number generation
  - Random number generation engines
  - Random number generation distributors
- Containers
  - std::array
  - std::forward_list
  - Unordered associative containers

Exercice : *Using the new elements added to the standard library*
Exercice : *Using std::optional*

String Manipulation

- New string Types
  - std::u16string
  - std::u32string
- Basic string view
- Converting between numeric and string types
- Elementary string conversions
- Input/output manipulators
  - std::get_money, std::put_money
  - std::get_time, std::put_time
  - std::quoted
- Regular expressions
  - Format of a string
  - Parsing the content of a string
  - Replacing the content of a string

Exercice : *Using String class and String literals*

Fourth Session

Concurrency and Multithreading

- Introduction
- Thread
- Atomic operations
  - Atomic features
  - Non-class functions
  - Atomic flag
  - Memory order
- Mutex
  - Avoiding using recursive mutexes
- Sending notifications between threads
- Condition variables
- Future and Promise
- Task and Async
- Modern C++ and RTOS

Exercice : *Blink synchronously 4 Leds*

Lambda functions

- Syntax of lambdas
- Defining lambdas
- Using lambdas
  - Using lambdas with standard algorithms
  - Assigning lambdas to function pointers
  - Lambdas and std::function
  - Writing a function that accepts a lambda as parameter
- Polymorphic lambdas
- Recursive lambdas

Exercice : *Understanding lambda*
Exercice : *Using lambda to modify and display a vector*

Dynamic memory management

- Memory Management
- Memory Errors
- Smart Pointers
  - Raw Pointers
  - Automatic pointers
  - Unique Pointers
  - Shared Pointers
  - Weak Pointers

Exercice : *Override new and delete*
Exercice : *Understanding unique and shared pointers*

Renseignements pratiques

Durée : 24 heures
Prix : 2000 € HT