L10 - Embedded Modern C++ Programming

The Modern C++ Language for Embedded Systems

Objectives

- Discover the modern C++ features
- Learn the language changes in C++11, C++14, C++17 and C++20
- Discover the new functionalities added to the standard library
- Learn advanced modern C++ features
- Moving to from traditional C++ to modern C++
- Putting in evidence the essential modern C++ features used in embedded application

Equipment

- Training manuals and software exercises
- One PC for two trainees
- One target board with ARM Cortex M4 microcontroller (STM32F)
- Eclipse environment and GCC compiler

Prerequisite

- Basic C++ Language knowledge (see for example our L3 - Embedded C++course)
- Preferably knowledge of embedded C Language (see for example our L2 - C language for Embedded MCUscourse)

Durée

- Totale : 12 heures
- 2 sessions de 6 heures chacune (hors temps de pause)
- De 40% à 50% du temps de formation est consacré aux activités pratiques
- Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante

Course Environment

- Theoretical course
  - Printed and PDF material in English.
  - Trainer assistance throughout.
- Each session starts with a trainee check-in.

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First Day

Introduction to modern C++

- Overview
- Storage class specifiers
- Uniform initialization
- 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

Keywords

- enum class
- override and final
- Inline variables
- nullptr
- static_assert
- noexcept
- constexpr and if constexpr
- decltype
- Defaulted and deleted functions

New functionalities

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

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

Second Day

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

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

Dynamic memory management

- Smart Pointers
    - Raw Pointers
    - Automatic pointers
    - Unique Pointers
    - Shared Pointers
    - Weak Pointers

Concurrency and Multithreading

- Introduction
- Thread
- Implementing a thread-safe singleton
- 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