



oL10 - Programmation C++ moderne embarquée

Objectifs

- Découvrir les fonctionnalités modernes du C++
- Apprendre les changements de langage dans C++11, C++14, C++17 and C++20
- Découvrir les nouvelles fonctionnalités ajoutées à la bibliothèque standard
- Apprendre les fonctionnalités avancées du C++ moderne comme "perfect forwarding"
- Passer du C++ traditionnel au C++ moderne
- Mettre en évidence les fonctionnalités essentielles du C++ moderne utilisées dans les applications embarquées

Les travaux pratiques sont réalisés sur une carte ARM QEMU

Pré-requis

- Connaissances en programmation C (voir notre cours [oL2 - Langage C pour les MCUs embarqués](#))
- Connaissances de base du langage C++ (voir notre cours [oL3 - Programmation C++ embarqué](#))

Environnement du cours

- Cours théorique
 - Support de cours au format PDF (en anglais)
 - Cours dispensé via le système de visioconférence Teams
 - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique
- Activités pratiques
 - Les activités pratiques représentent de 40% à 50% de la durée du cours
 - Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.
 - Un PC Linux en ligne par stagiaire pour les activités pratiques
 - Exemples de code, exercices et solutions
 - Le formateur a accès aux PC en ligne des stagiaires pour l'assistance technique et pédagogique
 - Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante
- Une machine virtuelle préconfigurée téléchargeable pour les activités pratiques après le cours
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus

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 Outline

Première session

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

Exercise : Using auto to declare variables

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

Exercise : Using modern C++ keywords

Exercise : 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

Exercise : Using the new for loop syntax

Exercise : Using std::tuple

Exercise : Move semantics performance advantages on embedded systems

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

Exercise : Using the new elements added to the standard library

Exercise : Using std::optional

Deuxième session

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

Exercise : Using String class and String literals

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

Exercise : 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

Exercise : Understanding lambda

Exercise : 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

Exercise : Override new and delete

Exercise : Understanding unique and shared pointers