Bienvenue à ProSkills IT – Formations professionnelles au Togo

Fiche du cours

70 h

Titre :

CPP300 - Programmation C++ 3

Description :

Perfectionnement C++ moderne pour la production : concurrence & parallélisme (std::thread, mutex, atomiques, future/async, algorithmes parallèles), coroutines C++20, réseau (Asio), métaprogrammation (variadiques, fold expressions, constexpr), modules C++20, performance (profilage, cache, allocations, Google Benchmark), robustesse & sécurité (ASan/UBSan/TSan, fuzzing, politiques d'erreurs), packaging & CI (CMake avancé, vcpkg/Conan, GitHub Actions).
Capstone : un service natif C++ (réseau ou calcul) concurrent, testé, benchmarkté et livrable (CI/CD).

Objectifs :

- Écrire du code concurrent sûr (verrous RAII, ordonnancement, contention) et utiliser futures/async.*
- Exploiter algorithmes parallèles (std::execution) et coroutines C++20 pour I/O ou compute.*
- Concevoir des APIs génériques avancées (variadiques, constexpr, patterns de métaprogrammation).*
- Profiler/optimiser : coûts de copies/moves, cache locality, allocations personnalisées.*
- Développer un service réseau (Asio) : protocole, sérialisation, gestion d'erreurs/temps.*
- Livrer en qualité prod : tests (unitaires/intégration), fuzzing, Sanitizers, CI + packaging.

Chapitres :

1. Architecture & outillage avancés : CMake targets, options, presets, packaging (vcpkg/Conan), strat. de logs (spdlog/fmt)*
2. Concurrence I : std::thread, mutex, lock_guard/unique_lock, condition_variable, modèles de synchronisation*
3. Concurrence II : atomiques (std::atomic), mémoire/ordering, wait-free/lock-free (aperçu), queues thread-safe*
4. Futures & tâches : future/promise, async, thread pools (schémas), cancellation/cooperative stop token (C++20)*
5. Parallélisme STL : std::execution, reduce/transform parallèles, pièges (contended data, false sharing)*

6. Coroutines C++20 (I/O & compute) : co_await, awaiters/awaitables, générateurs ; intégration avec Asio*
7. Métaprogrammation avancée : templates variadiques, fold expressions, constexpr, type_traits avancés*
8. Modules C++20 : partitionnement, import/export, compat avec headers, build CMake & limites actuelles*
9. Performance & mémoire : profils CPU, flamegraphs, Google Benchmark, allocateurs, SBO, cache locality*
10. Réseau Asio (TCP/UDP) : modèle async, timers, timeouts, sérialisation (JSON/binaire), backpressure*
11. Résilience & erreurs : politiques d'erreurs (exceptions vs expected), retries/backoff, timeouts, invariants, contrats*
12. Qualité & sécurité : ASan/UBSan/TSan, fuzzing (libFuzzer), hardening (flags), revue statique (clang-tidy)*
13. Intégration & CI/CD : tests unitaires/intégration, artefacts, matrices (Linux/macOS/Windows), packaging & versioning*
14. Atelier Capstone : cadrage, milestones, critères de perf, budgets temps/mémoire, doc & scripts de run

À la fin :

Vous serez capables de bâtir des applications C++ performantes et sûres, concurrentes/async (threads, algorithmes parallèles, coroutines), de développer un service réseau robuste (Asio) et de mesurer/optimiser vos performances (profils, benchmarks). Vous saurez industrialiser : erreurs & résilience maîtrisées, tests (unitaires/intégration) + fuzzing, Sanitizers, CI/CD et packaging — avec un capstone prêt à montrer en portfolio ou à intégrer en production.