

ISIE4 – C++ & C++ embarqué LOO, C++ et Embarqué

LOO, utile pour l'embarqué?

- La méthode objet (rappel minimaliste)
 - Fonctionne par analogie avec le modèle du problème
 - Mettre en correspondance l'espace des problèmes et l'espace des solutions
 - Idée clé
 - Analogie entre le modèle logiciel et le modèle physique → « avatar numérique »
- Dans l'embarqué on manipule souvent des objets « physiques »
 - Capteurs, actionneurs, ressources MCU...
 - L'intérêt de la méthode objet est évident

C++, le LOO pour l'embarqué?

- Le langage « officiel » de l'embarqué est et reste le langage C
 - Malheureusement...
- Le C++ est pensé pour :
 - Avoir une compatibilité presque totale avec le langage C
 - Rares exceptions, bien connues, faciles à identifier
 - Garantir des performances comparables à celles du langage C
 - Obsession de conception
 - On « colle » à la machine

Le C++ n'a alors que des avantages ?

- Côté analyse, conception, performances, oui
- Sur d'autres plans... moins évident...
 - Très portable, mais moins que d'autres (JAVA, Python...)
 - Peu consommateur en ressources, mais plus que le C
 - Objectivement, le C++ est un langage difficile à maîtriser
 - Probablement le langage le plus complexe qui soit largement utilisé (c'est pas du Brainfuck, qui est plus complexe, mais confidentiel...)

Pourquoi le C++ est-il « si » complexe ?

- Il a le défaut de ses avantages...
- Les « concepteurs » du C++ font évoluer le langage
 - Intégration de toutes les évolutions/fonctionnalités (jugées pertinentes) apportées pour/par d'autres langages
 - Espaces de nommage, généricité, variables et expressions constantes, références...
 - Tout en maintenant les exigences de départ
 - Performances, compatibilité C, accès au matériel...
- C'est une forme de grand écart...

Embarqué et embarqué...

- Si le C++ est parfait pour l'Embarqué, il ne l'est pas pour l'embarqué...
- Distinguer deux grandes familles de systèmes embarqués
 - Les Systèmes Embarqués construits autour d'un CPU ou d'un MCU (type SoC) minimum 32bits
 - OK pour du C++
 - Bare Metal, MicroKernel ou OS
 - Les systèmes embarqués mettant en œuvre des MCU 8 ou 16 bits
 - C++ pas adapté (gourmand en ressources mémoire, peu de compilateurs disponibles...)
 - Même en Bare Metal

La question du C++ se pose

- Si la cible est de type SBC (Single Board Computer)
 - rPi, BeagleBone, Udoo...
 - ARM Cortex A ou équivalent
 - Développement d'application en C++ pour un OS embarqué (GNU/Linux en général) avec le compilateur GCC
- Si la cible est ou embarque un MCU 32bits
 - STM32, LPC, ...
 - ARM Cortex M/R ou équivalent
 - Développement en Bare Metal ou avec RTOS, compilateur croisé spécifique ou GCC

Deux visions du C++

- Bottom-Up
 - Le C++ est une extension objet du C
- Top-Down
 - Le C++ est un LOO à la syntaxe inspirée du C
- Ces deux visions sont compatibles!
 - Surtout dans une approche pédagogique
- Elles sont cependant toutes deux fausses
 - Et alors ?...

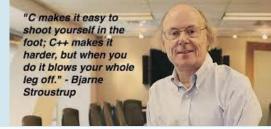
C++ et modèle objet

- Encapsulation
 - Droits d'accès
 - Philosophie de la « boite noire »
- Instanciation de classe
 - Construction / Destruction
 - Statique / dynamique
- Messages
 - Invocation (appel) de méthodes
- Généralisation
 - Héritage, redéfinition, extension
 - Types abstraits, factorisation, méthodes virtuelles

Mais aussi...

- Trois polymorphismes
 - ad-hoc
 - Surcharge
 - Par sous-typage
 - Héritage
 - Généralisé/Universel
 - Modèles
- Généricité
 - Fonctions/Méthodes génériques
 - Types génériques
 - Templates

Timeline du C++



- C++ = Bjarne STROUSTRUP (Bell)
- 1979-1982 : C avec classe
- 1983-1985 : C++, cfront, manuel de référence
- 1986-1988 : Diffusion importante du langage
- **1989-1998**
 - cfront 2.0 et 3.0
 - STL Alexander STEPANOV (HP)
 - ISO 14882:1998 aka C++98
- 2003 : ISO 14882:2003
- 2011 : ISO 14882:2011 aka C++11
- 2014 : ISO 14882:2014 aka C++14
- 2017 : ISO 14882:2017 aka C++17
- 2020 : ISO 14882:2020 aka C++20
- 2023 : ISO 14882:2023 aka C++23 🍑



Le C++, un langage inspiré...

- Le C (1972, Ken Thompson, Dennis Ritchie et Brian Kernighan - Bell)
 - Of course...
- Simula (1960, Dahl & Nygaard Oslo)
- Ada 83 (1980, Ichbiah CII-Honeywell Bull)
- Algol 68 (inspiré d'Algol 60, Backus & Naur)
- CLU (1975, Liskov MIT)
- ML (1980, Milner Edimbourg)
- _____

Quelques conseils...

- Ne pas hésiter à s'appuyer sur d'autres cours (passés, présents ou à venir) :
 - Syntaxe : Le C Avec méfiance
 - POO: Le JAVA
 - Analyse & Conception : UML / Génie Logiciel
 - RTOS: Systèmes Temps Réel
 - Embarqué : MCU (et un peu AFEN)
- Faire les efforts nécessaires pour appréhender le principal, ne pas se laisser désarçonner, accepter de ne pas toujours tout comprendre
 - il n'est pas nécessaire d'être un génie pour écrire du bon code C++