Versions de lecture :

- Version débutant
  - pas de connaissance en C++/programmation, pas de ressources externes, pas de code externe, pas destiné a etre utilisé en pro, mais comme base d'un autre apprentissage. Equivalent a "cours pour les zero"
- Version intermédiaire
  - commence a voir des ressources externes et lire du code non ecrit par soi meme. Peut etre utilisé en pro. Connaitre les pieges, savoir resoudre une problématique dans un contexte pro.
- Version avancée
  - orienter le lecteur va l'etude des ressources externes, pour devenir expert

Débuter en C++ moderne - Première partie

- machine model

img img

- qu'est-ce qu'une abstraction (modele) ? representation, simplifié, inexacte mais correcte sur les comportements qui nous interesse peut y avoir plusieurs abstraction sur la meme chose. zero-hoverhead principle (different de zero-cost) "don't paid for what you don't use". Aller plus loin : au dela des abstractions, implications d'utiliser du matos reels.
- paradigme imperatif, CPU abstraction, machine de turing
- variables. https://en.cppreference.com/book/intro/variables
  - namelookup, scope, adl, template deduction, etc.
  - https://en.cppreference.com/w/cpp/language/lookup
  - https://en.cppreference.com/w/cpp/language/adl
- La structure de données
- programmation generique : prendre un algo concret et l'utiliser sur differentes representation des donnees (differents types). Design : doit etre aussi simple a ecrire et a utiliser qu'un algo non generique.
- retour de fonction. Erreur. C'est quoi une erreur ? Quel est le probleme et qu'est-ce qu'on veut faire. Classification des types d'erreurs et des outils pour y repondre (retourn erreur, retour objet invalide, exception). Pas parfait, d'autres outils.

Débuter en C++ moderne - Seconde partie

La programmation orientée objet

- Introduction à la programmation orientée objet
- Discussions sur la notion de classe (Avancé)
- abstractions

Sémantique de valeur

- Les attributs de classe
  - Intermédiaire - Les attributs de classe
- Les fonctions membres
- Les classes génériques Intermédiaire ?
- RAII, constructeurs et destructeur
- Copie et mouvement
- Les conversions de types explicite et implicite. Intermédiaire ?
- La surcharge des opérateurs Intermédiaire ?
  - Les classes à sémantique de collection Intermédiaire ?
- La programmation par contrat const

Sémantique d'entité

- Les sémantiques de valeur et d'entité
- L'héritage
- Les fonctions virtuelles
- Les exceptions Intermédiaire ?

```
The writer of Vector doesn't know what the user would like to have done in this case (the wr

The user of Vector cannot consistently detect the problem (if the user could, the out-
of-range access wouldn't happen in the first place).

Bjarne Stroustrup - Tour of C++
```

- Allocation dynamique
- solid
- conception, modules, "Make interfaces easy to use correctly and hard to use incorrectly" - Scott Meyers

Intermédiaire ?

- duree de vie des objets.
- points de variation
- pointeurs nus.

- static, singletons. relation entre static et non static. Comment appeler un static dans un non static. Comment appeler un non static depuis un static. appeler fonction membre dans un static. registration d'un objet/fonction ou list de objets/fonctions

etude de quelques patterns :

- GoF
- event loop
- ecs
- Pointers, References and Optional References in C++: https://www.fluentcpp.com/2018/10/02/pointers-references-optional-references-cpp/

Projets

Ressources

Sur le C++ moderne

- Syllabus : https://github.com/cjdb/cpp-open-syllabus/blob/master/comp6770.md
- pleins de sujets aborde autour de hello : https://speakerdeck.com/olvemaudal/42-silly-ways-to-say-hello-in-modern-c-plus-plus-sep-2018