C++ TEMPLATE #1

(1/?) Introduction, origin story, possibility & pitfall

(2/?) Lambda template, template of template, constexpr & design pattern

(3/?) Variadic template? Advance MetaProg? More?

Origin Story

- Avant le C++, le C avec des classes
- Vous avez fait du code dans les années 90’s en écoutant les Spice Girls.
- Copies de code
- fmaxf(float x, float y), fmax(double x, double y), fmaxl(…
- Macro: https://gcc.godbolt.org/z/JMCpNP

Template C++

- Créer un outils pour le compilateur, et c'est lui qui fera le boulot pour nous.
- Rendre la chose plus lisible

- (on ne rigole pas dans le fond de la salle, svp ;))

- Eviter les problèmes des macro & étendre les possibilités: Template de classe, de fonction, de variable et d'alias.
- La terminologie habituelle: on parle de classe et algorithme générique.
- Exemple: https://gcc.godbolt.org/z/2TBWfB

Definition de Template

```
template <typename T>
void func(T val)
```

- La possibilité de d'avoir des paramètre de Template qui soit des types ou des valeurs.

```
template <typename T, std::size_t SIZE>
struct array
```

- Les paramètre de Template doivent être connus a la compilation.
- Une fois créé a partir d'un Template, structures/fonctions se comporte comme le reste.
- Le processus de création d'un objet a partir d'un Template est communément appelé « synthétisation par substitution »
- https://gcc.godbolt.org/z/qU-qWv

Template de class

- Un Template n’est pas une classe, mais une classe créé a partir d’un Template se comporte de la même façon que les autres.

- Les variable static sont dupliquer par le nombre de classes généré: https://gcc.godbolt.org/z/cS7BIg

- Depuis le C++17, il est possible de laisser le constructeur de la classe en déduire les paramètre de Template: https://gcc.godbolt.org/z/qZfGsy

Template de variable

- Disponible depuis le C++14
- Peux être représenté comme du sucre syntaxique autour d'une variable statique dans une classe: https://gcc.godbolt.org/z/RYyj7e
- Utile pour simplifier la syntaxe dans les expressions compliqué

Template de alias

- Disponible depuis le C++11
- Alias un type existant: https://gcc.godbolt.org/z/RrkjTY
- Utile pour simplifier la syntaxe dans les expressions compliqué

Template de fonction

- Un Template n'est pas une fonction, mais une fonction créé a partir d'un Template se comporte de la même façon que les autres : https://gcc.godbolt.org/z/qJtuE9
- Les variable statique vont aussi être dupliqué a chaque différentes générations de la fonction.
- Sauf si l'utilisateurs explicite les paramètres de Template, le compilateur est libre d'essayer de les déduire.
- Il y est possible de surcharger une fonction avec les paramètres de Template.

Déduction de type (1/2)

- La déduction est faites avec les paramètres Template et les paramètre de la fonction: https://gcc.godbolt.org/z/hn-Zj4
- Le type déduit pour T, va presque systématiquement essayé d'être le plus simple, le compilateur tend a retirer les 'const' et '&': https://gcc.godbolt.org/z/UK5luE
- C'est le même algo de déduction quand vous utilisez 'auto' pour déclarer une variable dans une fonction.
- Les paramètres de la fonction qui servent a la déduction des paramètres de Template doivent exactement avoir les types qui correspondent (pas de conversion implicite) : https://gcc.godbolt.org/z/sMfAek

Déduction de type (2/2)

- Les déductions se font « en parallèle » ; elles n'essayent pas nécessairement de s'accorder entre elles et ressortent une liste de valeurs possibles : https://gcc.godbolt.org/z/eJw3Tn

- A la fin de ce processus, le compilateurs vérifie que chaque paramètres de Template ont au moins une solution après le retrait d'éventuel conflits.

Reference & const collapsing

- Reference collapsing: https://gcc.godbolt.org/z/5kII8J

  & + & = &

  & + && = &

  && + & = &

  && + && = &&

- Les règles sont conçu pour tout fonctionne correctement et facilement, mais vous risquer de voir des comportement curieux si vous instancier explicitement avec un type.
- Const collapsing: https://gcc.godbolt.org/z/bbi09C
- Support intéressant a retenir: https://www.ibm.com/support/knowledgecenter/en/SSGH3R_16.1.0/com.ibm.xlcpp161.aix.doc/language_ref/reference_collapsing.html

Ressource pour aller plus loin

- **CppCon 2018: Walter E. Brown “C++ Function Templates: How Do They Really Work?”** https://www.youtube.com/watch?v=NIDEjY5ywqU
- **CppCon 2016: Arthur O'Dwyer “Template Normal Programming (part 1 of 2)”** **https://www.youtube.com/watch?v=vwrXHznaYLA**
- **CppCon 2016: Arthur O'Dwyer “Template Normal Programming (part 2 of 2)"** **https://www.youtube.com/watch?v=VIz6xBvwYd8**

Plus de question ?