MEHDIDOU99

Zeste de savoir

Causerie++ - Episode 3

4 mai 2019

Table des matières

1. Introduction

Salut les agrumes ! Bienvenue dans le troisième épisode de Causerie++ ! Pour ceux qui ont raté l'épisode précédent, c'est par ici .

Aujourd'hui, on va commencer à parler de C++20 ! Je parlerai aussi d'une sympathique petite conférence sur les algorithmes de la STL.

Allons-y !

2. C++20

Le dernier meeting du Comité de standardisation du C++ a eu lieu il y a plus d'un mois, et a donné lieu à une version finale - à peu de choses près[1] - du standard C++20. Voici un billet de Herb Sutter résumant un peu où en sont les choses.

Du coup, je me suis dit qu'il serait peut-être temps d'en parler un peu (d'autant que C++20, c'est du *gros, gros* changement). J'ai donc décidé que dans les semaines à venir, je dirai un mot sur une nouvelle fonctionnalité à chaque épisode. Je compte aborder, dans l'ordre :

— Les contrats
— Les concepts
— Les Ranges
— Les coroutines (peut-être)
— Les modules (peut-être)

Peut-être que j'en ajouterai d'autres si j'en ai envie, mais pour l'instant je pars sur ça. Evidemment, vous pouvez m'en proposer si vous avez des idées.

Je précise que je ne ferai pas un cours sur la notion, je dirai juste deux-trois mots sur ce que je trouve cool dans ces nouvelles *features*.

Bon, trêve de bavardages, aujourd'hui on parle Contrats !

2.1. Les Contrats en C++ 20

Les contrats sont une fonctionnalité qui m'enthousiasme beaucoup, car ils permettent - en toute logique - une approche de la Programmation par Contrat en C++ d'un tout autre niveau que ce que l'on pouvait faire avant.

En effet, il y a plusieurs aspects très intéressants à cette nouvelle approche :

- On n'a plus besoin de la macro `assert` héritée du C (remplacée par `[[assert /* ... */]]`), ce qui fait un usage de moins du préprocesseur. Mais ce point est anecdotique, à la limite ; c'est le reste qui est vraiment intéressant.
- On n'aura plus besoin d'exceptions pour vérifier les post-conditions, donc on pourra concilier vérification des post-conditions et *no throw safety guarantee*.
- On peut déclarer des préconditions et des postconditions **directement dans la déclaration de la fonction** (avec les constructions `[[expect /* ... */]]` et [[ensures / ... /]]'. Les contrats les plus simples sont donc directement présents à la déclaration, ce qui rend, je trouve, les contrats bien plus explicites car cela évite les recours à l'implémentation. De plus, un contrat faisant partie des informations importantes à connaître pour pouvoir utiliser une fonction, il est beaucoup plus logique qu'il soit exprimé au niveau de la déclaration que dans l'implémentation.
- On a une souplesse beaucoup plus importante en termes de manières de vérification des contrats et de gestion des violations de contrats :
  - On peut spécifier un **niveau de contrat** pour savoir quand le contrat sera vérifié : `axiom` (contrat jamais vérifié, dynamiquement, c'est un contrat à destination du lecteur du code et des outils d'analyse statique), `default` et `audit`. On peut ensuite choisir un *build level*, qui détermine si on vérifie uniquement les contrats `default`, ou les contrats `default` et `audit`, ou aucun.
  - La gestion des violations de contrat peut être personnalisée à l'aide d'un *violation handler*. La manière de spécifier le *violation hander* est définie par les implémentations, donc je crois qu'on ne sait pas encore comment ça se fera - peut-être à l'aide d'une option de compilation en ligne de commande ?

Ainsi, on obtient des contrats plus précis, plus explicites, plus souples ; en somme plus puissants et plus expressifs. Et en bonus on dégage une macro. C'est pas beau ça ?

3. [CppCon2018] 105 STL Algorithms in Less Than an Hour

Il y a quelques mois, j'ai vu une conférence de Jonathan Boccara sur les algorithmes de la STL : 105 STL Algorithms in Less Than an Hour .

1. Si j'ai bien compris, toutes les fonctionnalités qui feront partie du standard ont été décidées, mais il reste quelques fonctionnalités qui ont été approuvées et dont il reste à discuter les termes exacts de leur spécification.

Élément externe (Video) —
Consultez cet élément à l'adresse https://www.youtube.com/embed/2olsGf6JIkU.

C'est d'ailleurs par cette conférence que j'ai découvert Fluent C++, le blog dont j'ai parlé dans l'épisode 1 .

Cette conférence aborde tous les algorithmes de la bibliothèque standard, et résume tout ça en moins d'une heure, ce que je ne pensais pas possible. A mon sens, c'est très important de les connaître tous, car cela permet gratuitement d'augmenter le niveau d'expressivité du code, en « respectant mieux les niveaux d'abstraction », pour citer Jonathan Boccara. C'est une très bonne conférence pour ceux qui débutent avec les algorithmes de la STL ; ça en fait un excellent support pédagogique. Elle est d'ailleurs parfaitement complétée par la Carte des Algorithmes de la STL (*The World Map of C++ STL Algorithms*) disponible ici . Merci Jonathan !

En parlant de support pédagogique, cette conférence est citée dans ce document qui répertorie des conférences utiles pour apprendre/enseigner le C++. Je vais essayer de regarder toutes celles que je n'ai pas encore vues, on se revoit dans six mois ! Plus sérieusement, ça a l'air intéressant, je parlerai peut-être de certaines de ces conférences à l'avenir.

4. Conclusion

C'est tout pour cet épisode, n'hésitez pas à me suggérer de nouvelles idées !

A lundi prochain ! <3