GBDIVERS

Zeste de savoir

C++ TL ;DR news 1

8 juin 2021

Table des matières

Voici une petite revue d'articles récents, choisis et classés aléatoirement.

Lisez les autres C++ TL ;DR news: https://zestedesavoir.com/billets/3947/c-tl-dr-news/

— [Tous] Un top 7 de pièges que tous les développeurs devraient connaître
— [C++20] Un courte introduction aux concepts
— [Débutant] Une courte introduction aux templates

---

1. Top-7 Performance Traps for Every Developer

Lire l'article: https://www.cppstories.com/2021/perf-traps/

Beaucoup de développeuses et développeurs utilisent le C++ parce que "c'est plus performant". C'est une erreur assez classique. Le C++ permet d'optimiser, encore faut-il savoir le faire correctement. Cet article résume les 7 principales erreurs que font les développeuses et développeurs quand ils veulent optimiser un programme:

1. faire des prédictions sur les performances
2. faire des changements qui n'ont pas d'impact
3. ne pas connaître ses données
4. ne pas connaître son environnement technique
5. suivre aveuglément la notation big-O
6. trop optimiser son code
7. écrire de mauvais benchmarks

Comme on le dit souvent: testez votre code, faites du profiling!

L'auteur a également écrit un livre de 175 pages sur l'optimisation du code: "Performance Analysis and Tuning on Modern CPUs", Denis Bakhvalov. Ce livre est gratuit et le lien vers le PDF est donné dans l'article.

2. C++20 Concepts - a Quick Introduction

Lire l'article: https://www.cppstories.com/2021/concepts-intro/

Les concepts sont une nouveauté en C++ introduite dans le C++20. Cet outil va permet de repenser complètement la façon d'écrire des templates, ce qui va avoir un impact majeur sur les codes C++. Cet article est une courte introduction aux concepts, pour bien débuter avec cet outil.

Les concepts sont un moyens de mettre des contraintes sur des paramètres template. Par exemple pouvoir exprimer qu'une paramètre template doit être un nombre entier ou doit contenir une fonction spécifique. Il est déjà possible de faire de telles choses en C++, par exemple avec le SFINAE ou `enable_if`, mais les concepts permettent de simplifier tout cela.

Il existe plusieurs syntaxe pour définir un concept. Par exemple pour contraindre un type d'être un entier, il est possible d'écrire le concept suivant:

```
1 template <class T>
2 concept integral = std::is_integral_v<T>;
```

Pour utiliser ce concept, il suffit alors d'utiliser cette contrainte dans un template:

```
1 template <typename T>
2 requires integral<T>
3 void DoSomething(T param) { }
```

Ou la version simplifiée:

```
1 void DoSomething(integral auto param) { }
```

Il existe également plusieurs concepts prédéfinis dans l'en-tête `<concepts>`: `same_as`, `derived_from`, `convertible_to`, etc.

L'article détaille d'autres syntaxes alternatives. Globalement, ce ne sont pas des syntaxes difficiles à retenir, il suffira de se faire un pense-bête au début, le temps de s'en souvenir. Et de copier-coller-adapter les codes de cet article!

3. Templates - First Steps

Lire l'article: http://www.modernescpp.com/index.php/template-get-insight

Cet article est une introduction très courte sur les templates. L'article est basé sur plusieurs questions simples qu'une personne débutante peut se poser.

1. Quand utiliser les templates?

Pour implémenter une idée générale, qui n'est pas spécifique d'un type particulier. Par exemple "max".

2. Comment écrire un template?

Pour transformer une fonction non template (par exemple qui utilise le type `int`) en template, ajoutez `template <typename T>` et remplacez `int` par `T`.

```
template <typename T>
T max(T lhs, T rhs) {
    return (lhs > rhs)? lhs : rhs;
}
```

3. Que se passe-t-il quand un template est instancié?

Le compilateur va générer une instance de la fonction template pour chaque type utilisé.

```
max(10, 5);      // génère la fonction max<int>
max(10.5, 5.5);  // génère la fonction max<double>
```

4. Que se passe-t-il lorsqu'un template est instancié plusieurs fois avec le même type?

Une seule instance est créée pour chaque type. L'instanciation est paresseuse: le code est généré uniquement s'il est utilisé (il est donc possible d'écrire un code template invalide, tant que celui-ci n'est pas instancié).

L'idée de cette revue de presse est de proposer du contenu hebdomadaire. Pour faciliter cela, le contenu doit être simple à rédiger et relativement courts. Et idéalement, que d'autres autrices et auteurs puissent ajouter facilement leurs revues. Pour cela, je vous propose de poster vos revues dans l'issue suivante: https://github.com/NotANameServer/discord/issues/26 . Celle-ci sera publiée le dimanche 23 mai.