Langage C++ - Exercices
cppl1

R. Absil (abs)

14 novembre 2017

Ce document détaille des énoncés d'exercices de C++, laissés à la discrétion des étudiants. Ces exercices simples permettent aux étudiants d'exercer leur maîtrise des concepts vus en cours. Les concepts utilisés par une question sont explicitement mentionnés dans l'énoncé.

Exercice 1 (Classe, surcharge d'opérateurs, conversions)**.** Écrivez une classe fraction.

- Il doit être possible de convertir implicitement des `int` en fraction, et inversement.
- Cette classe doit posséder les opérateurs `+`, `+=`, `-`, `-=`, `*`, `*=`, `/`, `/=`, `<`, `<=`, `>`, `>=`, `==`, `!=`. Évitez la redondance de code au maximum pour cette classe.
- Limitez au maximum les erreurs d'overflow lors des opérations arithmétiques, entre autres via l'utilisation de fonctions calculant le P.G.C.D., P.P.C.M, les types des attributs de la classe, etc.

En particulier, le code suivant doit compiler (et fonctionner) :

```
1 frac c1; frac c2 = 1;
2 frac c3(1,2); // =1/2
3 frac c4(1,-2);// =-1/2
4 c1 + c2; 1 + c1; c1 + 1;
5 c1 < c2; 1 < c1; c1 < 1;
```

Exercice 2 (Fonctions, templates)**.** Écrivez les fonctions templates `constexpr` suivantes

1. `gcd`, `lcm`, permettant de calculer le pgcd et le ppcm entre deux « entiers »
2. `factorial`, permettant de calculer la factorielle d'un « entier »

Définissez une variable `constexpr` template dénotant la valeur de π, et une fonction `binom` permettant de calculer $\binom{n}{k}$.

Vous pouvez tester si vos fonctions sont *véritablement* `constexpr` par exemple via la structure suivante :

```
1 template<int n>
2 struct constN
3 {
4     constN() { std::cout << n << endl; }
5 };
6
7 constN<isPrime(97)> out;
```

Votre code doit être compatible en C++14.

Exercice 3 (Classe, héritage, polymorphisme)**.** Écrivez une classe modélisant un polynôme.

- Il doit être possible de construire un polynome par liste d'initialisation, où `pol p = {2,3,4,5};` dénote le polynôme $2x^3 + 3x^2 + 4x + 5$.

— Il doit être possible de convertir implicitement `double` en polynômes, mais pas l'inverse.
— Cette classe doit posséder les opérateurs `+`, `+=`, `-`, `-=`, `*`, `*=`. Assurez-vous que le degré du polynôme soit bien calculé après chacune de ces opérations.
— Il doit être possible d'évaluer ce polynôme en un réel quelconque via surcharge de l'opérateur `()`. Par exemple, sur le polynôme f précédent, `f(1)` doit valoir 14.
— Surcharger l'opérateur d'injection de flux pour donner une forme élégante à un polynôme imprimé en console.

En particulier, le code suivant doit compiler (et fonctionner) :

```
1 pol p1 = {2,3,4,5}; pol p2(3); //pol p3 = 3; //ko
2 p1 + p2; 1 + p1; p1 + 1;
```

Écrivez une classe modélisant un polynôme de second degré, qui a toutes les caractéristiques d'un polynôme et permet d'en calculer les racines. Suivez la même démarche pour les polynômes de premier degré.

Exercice 4 (Classe, template, allocation)**.** Écrivez une classe template modélisant un tableau dont la taille est constante et fixée à la compilation.

— Vous *devez* utiliser[1] l'opérateur `new` pour l'implémentation de cette classe. Assurez-vous que la copie et l'affectation se passent correctement.
— Surchargez l'opérateur `[]` permettant à la fois d'accéder à un élément et de l'affecter à une valeur.
— Lancez des exceptions en cas de dépassement de borne ou de taille erronée.

Exercice 5 (Classe, template, allocation)**.** Écrivez deux versions d'une classe template modélisant une liste doublement chaînée[2], l'une en utilisant[3] l'opérateur `new`, l'autre en utilisant les pointeurs intelligents.

— Il doit être possible de parcourir les éléments de la liste via une boucle « for each ».
— Il doit être possible d'ajouter et de supprimer des éléments en début et en fin de la liste.
— Il doit être possible de trouver et supprimer la première occurrence d'un élément dans la liste.
— Il doit être possible d'insérer un élément à une position arbitraire de la liste, par le biais d'un itérateur.

De plus, assurez-vous que la copie et l'affectation d'une liste se passe sans encombres.

1. Pour des raisons académiques
2. À l'évidence, vous ne pouvez pas utiliser les conteneurs standards pour cet exercice.
3. Pour des raisons académiques