Les nombres rationnels

Rappel mathématique

https://fr.wikipedia.org/wiki/Nombre_rationnel

Nombre qui s'écrit sous forme de fraction. Certains nombres ne sont pas représentable par une série finie de chiffre. Par exemple $1/3$ s'écrit : 0.3333333... avec une infinie de 3. Quand on écrit `1.0/3.0` en C++, on réalise une division et le résultat est conservé en mémoire sous forme de nombre réel (donc avec un arrondi).

En C++ : `std::ratio`

Note : compile time et runtime. Rappel, 2 phases importantes dans un programme : compilation, transforme le code source C++ en langage machine, puis exécution par l'ordinateur. En C++, il est possible de faire des opérations dans une phase spécifique.

Remarque : en règle générale (donc pas que pour `std::ratio`), quand possible, le compilateur fera les calculs à compile time, en fonction des options de compilation. Par exemple :

```
cout << (2+3) << endl;
```

Le compilateur peut calculer directement l'opération (2+3), il va donc remplacer le code par le résultat et compilera en fait :

```
cout << 5 << endl;
```

C'est très intéressant, puisque tout ce qui est fait au compile time sera fait qu'une seule fois = meilleures performance.

`std::ratio` uniquement en compile time, donc le résultat est évalué à la compilation. (On verra en exo dans la partie POO pour créer une classe rationnel au runtime)

<http://en.cppreference.com/w/cpp/numeric/ratio>

main.cpp

```
std::ratio<2, 3>

std::ratio_add<std::ratio<2, 3>, std::ratio<3, 4>>

std::cout << std::ratio_add<std::ratio<2, 3>, std::ratio<3, 4>>::num << std::endl;
std::cout << std::ratio_add<std::ratio<2, 3>, std::ratio<3, 4>>::den << std::endl;
```

Utilisation de using. Note : ancienne écriture : typedef

Opération arithmétiques : ratio_add, ratio_subtract, ratio_multiply, ratio_divide

Opérateur comparaison : ratio_equal, ratio_not_equal, ratio_less, ratio_less_equal, ratio_greater, ratio_greater_equal

Ratios communs : micro, milli, kilo, mega, etc.

[Chapitre précédent](#) [Sommaire principal](#) [Chapitre suivant](#)