Amusons-nous… la surcharge d'opérateur

420-C21-IN PROGRAMMATION II

GODEFROY BORDUAS – AUTOMNE 2021

MODULE BONUS

Comment définir des opérateurs

- Les opérateurs (<<, +, -=, etc.) sont, en C++, des fonctions définies pour les types qui existent de base dans le langage
- Comme les structures sont des types définis hors du langage, il n'existe pas de fonction définie pour elles
- Il est nécessaire de définir nos fonctions d'opérateur. Nous parlons alors de **surcharge d'opérateur**
- Prenons les opérateurs << et >>, ces derniers agissent les flux d'entrée et de sortie. Or, il existe deux types de flux `istream` et `ostream`. Il s'agit des flux de base
  - Sortie (lié à >>) : son type est `istream` (`cout` spécialise `istream` pour la console
  - Sortie (lié à <<) : son type est `ostream` (`cin` spécialise `ostream` pour la console

Une syntaxe simple à une idée complexe

- Ici, nous présentons l'idée pour l'opérateur <<. Toutefois, le principe reste valide pour tous les opérateurs.
  - Néanmoins, il faut faire attention aux types de paramètres impliqués
- Le prototype (générale) de la fonction d'opérateur << sera :

```
void operator<<(std::ostream&, const Structure_s);
```

- Pour tous les opérateurs, le nom de la fonction est toujours `operator` suivi du symbole de l'opérateur
- Le symbole **&** désigne une référence. Il s'agit d'un pointeur autogéré par le compilateur.
  - Ainsi le formalisme ne s'applique pas
- Nous appliquons la condition const sur le paramètre du flux pour éviter de le modifier accidentellement

Qu'est-ce que les deux paramètres signifient ?


```
void operator<<(std::ostream&, const Structure_s);
```

- Le premier reçoit le flux à manipuler
- Le second reçoit l'entité à utiliser dans la manipulation
- En gros, prenez la ligne suivante :

```
cout << 5;
```

- La fonction d'opérateur appeler a donc la signature suivante :

```
void operator<<(std::ostream&, const Int);
```

- La première ligne peut être remplacée par :

```
operator<<(cout, 5);
```

Imaginons la structure Vecteur2

- Regardez le code **vecteur2.cpp**

```
#include <iostream>
struct Vecteur2_s { float x; float y; };

void operator<<(std::ostream&, const Vecteur2_s&);

void operator<<(std::ostream& stream, const Vecteur2_s& vec) {
    stream << "(" << vec.x << ", " << vec.y << ")";
}
```

Petit problème, essayer d'ajouter une fin de ligne après l'appel

- Avec les choses actuelles, vous ne pouvez pas appliquer `endl` après avoir appelé `cout << vec`
- Ceci vient du fait que votre fonction d'opérateur ne retourne pas de flux.
- Regardons l'exemple suivant :

```
cout << vecteur << endl;
```

- Ceci revient à dire que (on applique la même logique qu'avant) :

```
operator<<(operator<<(cout, vecteur), endl);
```

- Pour avoir ce résultat, il faut utiliser le prototype suivant :

```
std::ostream& operator<<(std::ostream&, const Vecteur2_s&);
```

Notre exemple précédent devient

- Regardez le code **vecteur2_plus.cpp**

```
#include <iostream>
struct Vecteur2_s { float x; float y; };

std::ostream& operator<<(std::ostream&, const Vecteur2_s&);

std::ostream& operator<<(std::ostream& stream, const Vecteur2_s& vec) {
    stream << "(" << vec.x << ", " << vec.y << ")";
    return stream;
}
```

Tous les opérateurs peuvent être surchargés

- Regardez le code **vecteur2_plusplus.cpp**
  - Codez les fonctions suivantes :

```
void operator+=(Vecteur2_s&, const Vecteur2_s&);
void operator-=(Vecteur2_s&, const Vecteur2_s&);
void operator*=(Vecteur2_s&, const float&);

Vecteur2_s operator+(const Vecteur2_s&, const Vecteur2_s&);
Vecteur2_s operator-(const Vecteur2_s&, const Vecteur2_s&);
Vecteur2_s operator*(const Vecteur2_s&, const float&);
```

Allons-y aussi avec les binaires

- Regardez le code **vecteur2_plusplus_bin.cpp**
  - Codez les fonctions suivantes :

```
bool operator==(const Vecteur2_s&, const Vecteur2_s&);
bool operator!=(const Vecteur2_s&, const Vecteur2_s&);
```

Exercice : Les nombres rationnels

- Créer une structure Rationnel (deux valeurs entières : numérateur et dénominateur)
- Définissez les opérateurs : +, -, *, /, <<
- Petit rappel mathématique :

$$\frac{p}{q} + \frac{r}{s} = \frac{s * p + r * q}{q * s}$$

$$\frac{p}{q} * \frac{r}{s} = \frac{p * r}{q * s}$$

$$\frac{p}{q} \div \frac{r}{s} = \frac{p}{q} * \frac{s}{r}$$