

Support de cours

Cours:

Introduction à la programmation orientée objet (en C++)

Vidéo:

W13-04-surchopinterne-CPP-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Lancée de la surcharge des opérateurs. Premier opérande. Séquence vidéo. Points principaux. Moyen d'une fonction. Définition de cet opérateur. Travers d'un objet. Opérateur +. Surcharge interne. Nombres complexes. Premiers arguments de la fonction. Définition externe. Appel de fonction. Moyen de l'opérateur de résolutions. Moyen d'une surcharge.



[vers la recherche de séquences vidéo](#)

(dans Introduction à la programmation orientée objet (en C++).)



[vers la vidéo](#)

Center for Digital Education. Plus de matériel de soutien pédagogique ici :

<https://www.epfl.ch/education/educational-initiatives/cede/educational-technologies-gallery/boocs-en/>

Surcharge d'opérateurs : surcharge interne

(Partie 1)

Introduction à la programmation orientée objet (en C++)

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

...

notes

résumé

0m 0s



Surcharge interne des opérateurs

Pour surcharger un opérateur *Op* dans une classe *NomClasse*, il faut **ajouter la méthode** `operatorOp` dans la classe en question :

```
class NomClasse {
    ...
    // prototype de l'opérateur Op
    type_retour operatorOp(type_parametre);
    ...
};

// définition de l'opérateur Op
type_retour NomClasse::operatorOp(type_parametre)
{
    ...
}
```

Rappel : les méthodes ne doivent pas recevoir l'instance courante en paramètre

Dans la séquence vidéo qui suit nous continuons sur la lancée de la surcharge des opérateurs en nous intéressant plus particulièrement à la surcharge interne. Dans les séquences qui ont précédé nous avons en effet vu que un opérateur pouvait être surchargé soit à l'extérieur d'une « class », dans ce cas là qu'il s'agira d'une fonction soit à l'intérieur d'une « class » et dans ce cas-là il s'agira d'une méthode appartenant à une « class ». Si par exemple on a défini l'opérateur '+' pour les nombres complexes au moyen d'une fonction lorsque l'on écrit quelque chose comme « $z1 + z2$ », où $z1$ et $z2$ sont des « Complexes », ceci correspond donc à un appel de fonction à l'appel de la fonction « opérateur + » qui prend en argument « $z1$ » et « $z2$ ». Le premier opérande correspond aux premiers arguments de la fonction et le second au second arguments de la fonction. Une alternative qui s'offre à nous ici serait de définir l'« opérateur + » au moyen d'une surcharge interne c'est-à-dire en plaçant la définition de cet opérateur à l'intérieur de la « class » « Complexe » ; puisqu'il s'agit alors d'une méthode il faudra nécessairement l'invoquer au travers d'un objet ce qui veut dire que lorsque nous écrivons ceci, cela va se traduire par quelque chose de différent du cas de la surcharge externe cela va se traduire par un appel de méthode qui a l'allure suivante. Le premier opérande est l'objet courant l'objet auquel s'applique l'opérateur '+' et le second doit correspondre à l'argument de la méthode nous voyons donc que dans le cas d'une surcharge interne nous n'avons plus besoin que d'un seul argument qui va correspondre au deuxième opérande ; pour résumer si l'on veut réaliser la surcharge interne d'un

notes

résumé

0m 1s



Surcharge interne des opérateurs

Pour surcharger un opérateur *Op* dans une classe *NomClasse*, il faut **ajouter la méthode** `operatorOp` dans la classe en question :

```
class NomClasse {
    ...
    // prototype de l'opérateur Op
    type_retour operatorOp(type_parametre);
    ...
};

// définition de l'opérateur Op
type_retour NomClasse::operatorOp(type_parametre)
{
    ...
}
```

Rappel : les méthodes ne doivent pas recevoir l'instance courante en paramètre

opérateur donné il faudra placer la méthode « `operatorOp` » à l'intérieur d'une « class ». Comme pour toutes les autres méthodes, on se contentera usuellement de placer à l'intérieur de la « classe » le prototype de l'opérateur et l'on donnera la définition de l'opérateur à l'extérieur de la « classe », l'opérateur devra être rattaché à la classe à laquelle il appartient dans sa définition externe au moyen de l'opérateur de résolutions portées comme pour toutes les méthodes que nous avons l'habitude de définir. Un des points principaux à retenir est que le premier opérande d'un tel opérateur est l'instance courante et dans ce cas-là cette instance ne doit pas être passée en paramètre. Les méthodes ne reçoivent en paramètre qu'un éventuel second opérande. qu'un éventuel second opérande.

notes

résumé