Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Arguments par défaut et surcharge (partie 2)

Concepts (extraits des sous-titres générés automatiquement) :

Surcharge de fonctions. Type des paramètres. Paramètres différents. Paramètre de type int. Type de leurs paramètres. Type de retour. Première fonction. Deuxième fonction. Fonctions de même nom. Paramètre de type double. Surcharge des fonctions. Nom des fonctions. Appel d'affiche. Partie de ces distinctions. Exemple int.

vers la recherche de séquences vidéo
(dans Initiation à la programmation (en C++).)

vers la vidéo

EPFL

Fonctions : arguments par défaut et surcharge
(Partie 2)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

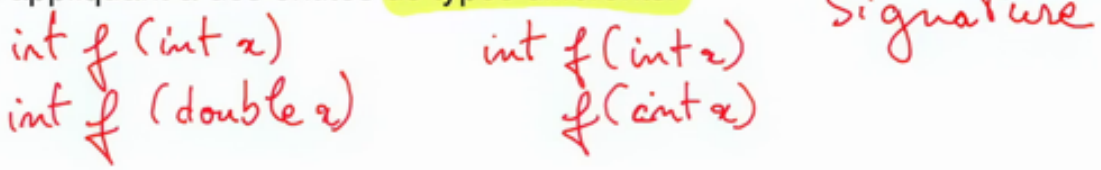
0m 0s

La surcharge de fonctions

EPFL

En C++, il est de ce fait possible de définir **plusieurs fonctions de même nom** si ces fonctions n'ont pas les mêmes listes de paramètres : nombre ou types de paramètres différents.

Ce mécanisme, appelé **surcharge des fonctions**, est très utile pour écrire des fonctions « *sensibles* » au type de leurs arguments
c'est-à-dire des fonctions correspondant à des traitements de même nature mais s'appliquant à des entités de types différents.



Nous allons terminer en abordant la surcharge de fonctions. En C++ il est possible de définir plusieurs fonctions qui ont le même nom si ces fonctions ont des paramètres différents. C'est à dire, si le nombre ou le type des paramètres sont différents. Ça va permettre au compilateur de distinguer entre les fonctions. C'est ce qu'on appelle la surcharge des fonctions et c'est très utile quand on va voir des fonctions qui font des traitements similaires, mais à partir de données différentes. Donc, en c++ ce qui différencie deux fonctions ce n'est pas seulement le nom des fonctions, mais aussi le type de leurs paramètres, ce qu'on appelle techniquement la 'signature' de la fonction. À noter que le type de retour lui ne fait pas partie de ces distinctions, on n'a pas le droit d'avoir deux fonctions de même nom et de même paramètre qui ne différeraient que par le type de retour. C'est à dire que je peux avoir deux fonctions " f ", la première fonction a un paramètre de type int, la deuxième fonction a un paramètre de type double, et le type de retour pour ces deux fonctions est le même, par exemple, int. En revanche, je ne peux pas avoir deux fonctions " f " qui ont toutes les deux un paramètre de type int par exemple, et qui ne différerait que par le type de retour. Par exemple int pour la première,

notes

résumé

0m 1s

La surcharge de fonctions : exemple

EPFL

```
void affiche(int x) {
  cout << "entier : " << x << endl;
}
void affiche(double x) {
  cout << "reel : " << x << endl;
}
void affiche(int x1, int x2) {
  cout << "couple : " << x1 << x2 << endl;
}
```

`affiche(1)`, `affiche(1.0)` et `affiche(1,1)` produisent alors des affichages différents.

Remarque :

```
void affiche(int x);
void affiche(int x1, int x2 = 1);
```

est interdit !

☞ ambiguïté

affiche(2);

et double pour la deuxième. En résumé, en c++, on peut avoir des fonctions ayant le même nom, du moment qu'elles aient des paramètres de types différents. Par exemple, je peux avoir trois fonctions qui s'appellent toutes affiche. La première fonction a un paramètre de type int, la deuxième, un paramètre de type double, et la troisième, deux paramètres tout les deux de type int. Ce sont donc bien 3 fonctions différentes. Maintenant, si j'appelle affiche(1), comme l'argument 1 est de type int, c'est la première fonction, qui a un paramètre de type int qui va être appelé. Et dans ce cas là, j'obtiendrai l'affichage "entier : 1". Si maintenant j'appelle affiche(1.0) comme l'argument (1.0) est de type double, c'est la deuxième fonction qui a un paramètre de type double qui sera appelée, et j'obtiendrai comme affichage "reel : 1". À l'appel d'affiche(1,1) comme les deux arguments sont tous les deux de type int, c'est la troisième fonction qui sera appelée, et j'obtiendrai comme affichage "couple : 11". Attention, je ne peux pas définir ces 2 fonctions ensemble dans un même programme. La première fonction a un paramètre de type int, la deuxième a deux paramètres, tous les deux de types int, mais le deuxième paramètre a une valeur par défaut. Si maintenant j'appelle affiche(2)

notes

résumé

1m 37s

on ne peut pas savoir si je veux appeler la première fonction "affiche" en passant 2 comme argument au paramètre "x", ou la deuxième fonction, en passant 2 au paramètre "x1" et utiliser la valeur par défaut 1 le paramètre "x2". Il y a donc une ambiguïté entre ces deux fonctions à cet appel. Donc, c'est pour ça que définir ces deux fonctions ensemble dans le même programme est interdit. est interdit.

notes

résumé

3m 49s