

Support de cours

Cours:

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

Vidéo:

### INIT-CPP-7\_4-PointeursDeclarationetOperateurs-pt2

Concepts (extraits des sous-titres générés automatiquement) :

**Nom d'une variable de type. Contexte d'utilisation. Premier exemple. Nom d'un type. Contextes d'utilisation du symbole. Cas de l'étoile. Premier contexte. Premier cas. Première ambiguïté possible. Second contexte. Premier contexte d'utilisation. Adresse de cette variable. Variable de type pointeur. Fait différent. Nom d'une variable.**



[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/>

# Pointeurs : déclaration et opérateurs de base

## (Partie 2)

### Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s





## GARE AUX CONFUSIONS !



C++ utilise (malheureusement) deux *notations identiques* (& et \*) pour *des choses différentes* !



C++ a la réputation d'être un langage à la syntaxe un peu compliquée.

notes

---

---

---

---

---

---

---

---

---

---

résumé

0m 1s



---

---

---

---

---



## GARE AUX CONFUSIONS !

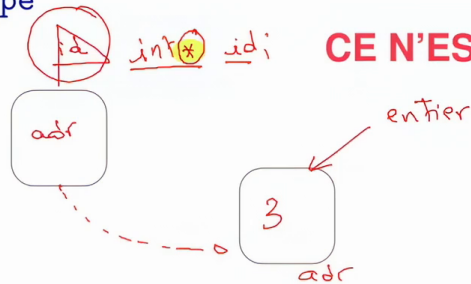


C++ utilise (malheureusement) deux *notations identiques* (& et \*) pour *des choses différentes* !

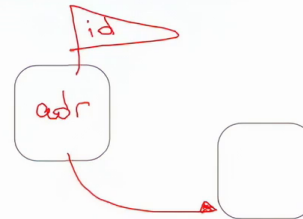
`type* id;` déclare une variable `id` comme un pointeur sur un type de base

`*id` (où `id` est un pointeur) représente le contenu de l'endroit pointé par `id`

type



**CE N'EST PAS LA MÊME CHOSE !**



Avec les opérateurs « \* » et « & » que nous venons de voir, nous en avons un premier exemple. En fait, selon le contexte d'utilisation, ces deux opérateurs ne veulent pas toujours dire la même chose. Regardons tout ceci en détails. Il existe désormais deux contextes d'utilisation du symbole « & ». Le premier contexte où le « & » suit le nom d'un type et le second contexte où le « & » précède le nom d'une variable de type pointeur. Dans le premier cas, nous allons utiliser le « & » pour déclarer une référence, soit lors d'un passage par référence de fonction, soit pour déclarer une référence en elle-même. Par exemple, je peux déclarer une variable « i » de type entier. Et je peux déclarer une référence à cette variable qui s'appelle « id » qui veut dire simplement que « id » est désormais un autre nom pour la variable « i ». Donc premier contexte d'utilisation, utiliser le « & » pour signifier la déclaration d'une référence. Second contexte d'utilisation, celui où le symbole « & » précède le nom d'une variable, et nous venons de voir que dans ce cas, nous allons simplement obtenir l'adresse de cette variable. Et cette adresse peut évidemment être affectée à une variable de type pointeur pour établir le lien entre un pointeur et une variable pointée. Donc ici nous avons typiquement un contexte d'utilisation qui est tout à fait différent du premier et qui nous permet d'aboutir à une situation tout à fait différente. Donc première ambiguïté possible, deux contextes possibles d'utilisation du « & » qui ne veulent strictement pas dire la même chose en C++. Deuxième exemple où C++ utilise le même symbole pour représenter deux choses différentes, le cas de l'étoile. Nous avons en effet vu dans ce qui a précédé que lorsque l'étoile suit le nom d'un type

notes

résumé

0m 5s





## GARE AUX CONFUSIONS !

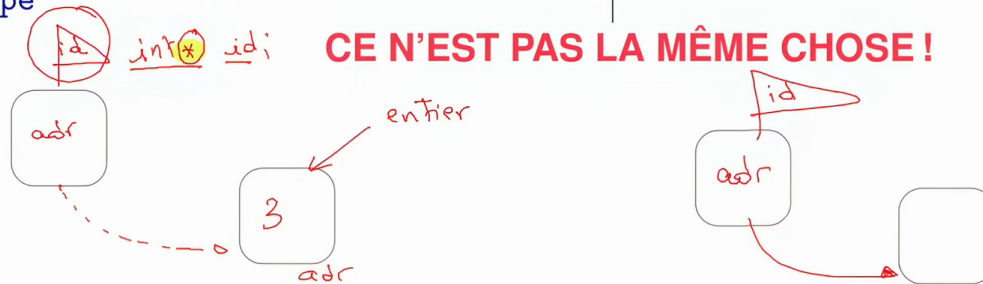


C++ utilise (malheureusement) deux *notations identiques* (& et \*) pour *des choses différentes* !

**type\*** **id**; déclare une variable **id** comme un pointeur sur un type de base

**\*id** (où **id** est un pointeur) représente le contenu de l'endroit pointé par **id**

type



comme c'est le cas ici, il sert tout simplement à déclarer une variable de type pointeur. Par exemple, si j'écris dans un programme quelque chose comme ceci, je suis en train de déclarer une variable appelée « id » et cette variable est déclarée de type pointeur sur un entier grâce à cette étoile. Ce qui veut dire qu'elle peut contenir, potentiellement, l'adresse d'une zone mémoire contenant un entier. Comme ceci Donc « id » est une variable de type pointeur déclarée grâce à l'étoile Si par contre, l'étoile est employée en précédant le nom d'une variable, cela présuppose que la variable en question est une variable de type pointeur. Et l'étoile va servir à retrouver la valeur pointée par le pointeur en question. Donc ici nous sommes plutôt dans cette situation. Nous avons une variable de type pointeur qui contient l'adresse d'une certaine zone mémoire, laquelle contient une valeur.

### notes

### résumé

Imaginons que cette valeur soit un entier. Donc « \*id » va tout simplement nous retourner la valeur 3 la valeur pointée par le pointeur « id ». Donc ici, l'étoile nous sert à retrouver la valeur pointée par un pointeur. Il s'agit d'un contexte d'utilisation tout à fait différent de celui qui a précédé ici. tout à fait différent de celui qui a précédé ici.

3m 1s

