

Support de cours

Cours:

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

Vidéo:

W13-01-static-CPP-pt2

Concepts (extraits des sous-titres générés automatiquement) :

Attribut de classe. Attributs d'instances. Information commune. Syntaxe particulière. Dehors de la classe. Ligne d'initialisation. Employés d'une entreprise. Fois de notre exemple du rectangle. Dehors de la classe rectangle. Telle ligne de code. Âge officiel de départ. Variable de classe. Dehors de la déclaration de la classe. Instance de la classe. Ensemble des instances d'une classe.



[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/>

Variables et méthodes de classe

(Partie 2)

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

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

...

notes

résumé

0m 0s





Les attributs d'instances sont typiquement initialisés

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

0m 1s



.....

.....

.....

.....

.....

Initialisation des attributs de classe

Un attribut de classe doit être initialisé explicitement à l'extérieur de la classe

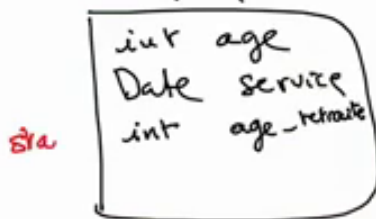
```
/* Initialisation de l'attribut de classe dans le fichier de
   définition de la classe, mais HORS de la classe.
*/
```

```
int Rectangle::compteur(0);
```

```
/* Rectangle::compteur existe même si l'on n'a déclaré
   aucune instance de la classe Rectangle */
```

67

Employe



au moment de la construction des objets. Or nous avons vu qu'un attribut de classe peut exister indépendamment de la construction de tout objet. Comment faire donc pour initialiser un attribut de classe ? Il faut pour cela avoir recours à une syntaxe particulière. En dehors de la classe, placez une ligne d'initialisation qui aurait cette syntaxe et qui consiste à dire : la variable compteur de la classe Rectangle, qui est de type entier, doit être initialisée à zéro. Une telle ligne de code apparaîtrait en dehors de la classe Rectangle, par exemple comme ceci, on reprendrait cette ligne en dehors de la déclaration de la classe Rectangle. On a recours à cette syntaxe particulière car un attribut de classe existe même si on a déclaré aucune instance de la classe. Donc on ne peut pas compter, par exemple, sur un constructeur pour réaliser l'initialisation nécessaire. Le recours à une variable de classe pour comptabiliser l'ensemble des instances d'une classe est évidemment un exemple relativement atypique. Il existe bien sûr de nombreuses autres situations où partager une information commune entre plusieurs instances est utile. Mais souvent il s'agira d'une information qui est de nature constante. Voyons un exemple. Sortons pour une fois de notre exemple du rectangle et imaginons une situation où l'on aurait besoin de représenter les employés d'une entreprise, un exemple plus administratif. On imagine par exemple qu'un employé est caractérisé par son âge, sa date d'entrée en service. Et puis, imaginons qu'il existe une information commune à tous les employés, qui serait l'âge officiel de départ à la retraite. Où placer cette information sur l'âge officiel de départ à la retraite, qui par exemple serait de 65 ans ? Cet âge officiel de départ à la retraite, de 65 ans, serait le même pour tous les employés. Imaginons que dans un premier temps, je le déclare comme un attribut d'instance,

notes

résumé

0m 5s



Initialisation des attributs de classe

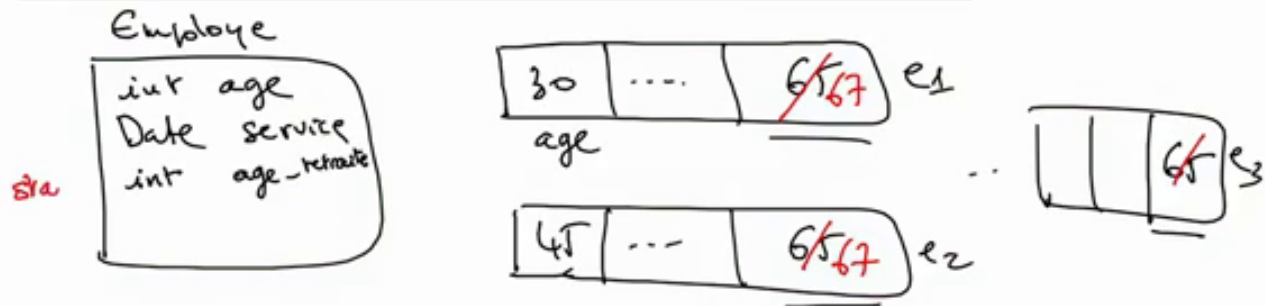
Un attribut de classe doit être initialisé explicitement à l'extérieur de la classe

```
/* Initialisation de l'attribut de classe dans le fichier de
   définition de la classe, mais HORS de la classe.
*/
```

```
int Rectangle::compteur(0);
```

```
/* Rectangle::compteur existe même si l'on n'a déclaré
   aucune instance de la classe Rectangle */
```

67



comme ceci. Ceci voudrait dire qu'à chaque fois qu'on crée un employé, cet employé aurait donc un âge, 30 ans, une date d'entrée en service, et un âge officiel de la retraite, qui serait de 65 ans. Idem pour le second employé, toujours avec 65. Ceci autant de fois que l'on aurait d'employés, avec toujours cette valeur 65 qui est dupliquée à tout les endroits. Cette information est donc répétée inutilement pour chacun des employés. Il y a donc une duplication et surtout elle induit un problème de maintenance. Imaginez par exemple que suite à un changement de loi, l'âge de la retraite passe à 67 ans. Il faudrait donc aller dans chacune des instances, et modifier partout cette valeur. Ce qui évidemment peut poser de gros soucis de maintenance. La solution consiste ici à définir cette information comme étant une variable de classe

notes

résumé

Initialisation des attributs de classe

Un attribut de classe doit être initialisé explicitement à l'extérieur de la classe

```
/* Initialisation de l'attribut de classe dans le fichier de
   définition de la classe, mais HORS de la classe.
*/

int Rectangle::compteur(0);

/* Rectangle::compteur existe même si l'on n'a déclaré
   aucune instance de la classe Rectangle */
```

Les attributs de classe sont très pratiques lorsque **différents** objets d'une classe doivent accéder à une **même** information.

Ils permettent notamment d'**éviter** que cette information soit **dupliquée** au niveau de chaque objet.

- ☛ Concrètement : réserver cet usage à des **constantes** utiles pour toutes les instances de la classe.

au moyen du mot réservé static. Ici typiquement, age_retraite serait une zone mémoire communément accessible par toutes les instances mais qui ne serait pas dupliquée dans chacune des instances. Typiquement, ceci disparaîtrait. Idem pour chacune des instances. Pour résumer, nous avons vu sur les exemples précédents que les attributs de classe sont typiquement très pratiques et très utiles lorsque différents objets, par exemple nos employés de tout à l'heure ont besoin d'accéder à une information, l'âge officiel de départ à la retraite. Nous avons vu que cela permet d'éviter de la duplication d'information, et permet une meilleure maintenance de cette information. Vous noterez enfin que le cas que nous avons étudié tout à l'heure, à savoir celui du comptage d'instance où la variable de classe était définie en non-constante, parce que nous avons besoin de décrémenter/incrémenter cette valeur à chaque création ou destruction d'objet. Eh bien, cette situation est relativement atypique, dans la pratique, vous allez le plus couramment rencontrer des variables de classe pour représenter des constantes utiles à toutes les instances d'une classe donnée. d'une classe donnée.

notes

résumé

3m 13s

