

Support de cours

Cours:

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

Vidéo:

W12-02-constrdefault-CPP-pt4

Concepts (extraits des sous-titres générés automatiquement) :

Propos des constructeurs. Variantes de déclaration de constructeur. Constructeur explicite. Valeur d'initialisation explicite. Classe rectangle. Façons possibles. Objet de cette classe. Valeur d'initialisation. Défaut. Initialisation d'un objet de type. Variante c. Syntaxe de déclaration. Guise de paramètre. Variante d. Variante d..



[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/>

Constructeurs par défaut en C++ (Partie 4)

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

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

...

notes

résumé

0m 0s



Constructeur par défaut par défaut

Si aucun constructeur n'est spécifié, le compilateur *génère automatiquement* une **version minimale du constructeur par défaut** qui :

- ▶ appelle le constructeur par défaut des attributs objets.
- ▶ laisse non initialisés les attributs de type de base.

Dès qu'**au moins un constructeur a été spécifié**, ce constructeur par défaut par défaut **n'est plus fourni**.

Si donc on spécifie *un* constructeur sans spécifier de constructeur ~~par~~ défaut, on ne peut plus construire d'objet de cette classe sans les initialiser (ce qui est voulu !) puisqu'il n'y a plus de constructeur par défaut.

C++11...mais on peut le rajouter si on veut (voir plus loin).

Mais attention à propos des constructeurs par défaut par défaut Ils ont la particularité suivante : dès que dans la classe on spécifie au moins un constructeur explicite, qu'il soit par défaut ou non, alors le constructeur par défaut par défaut n'est plus fourni. Ceci signifie concrètement que si dans une classe je spécifie un constructeur explicite qui n'est pas un constructeur par défaut alors je ne peux plus construire d'objet de cette classe sans initialiser, sans fournir de valeur d'initialisation explicite pour les attributs. Ceci correspond typiquement à quelque chose de voulu. Lorsqu'on se donne la peine de programmer un constructeur explicite on ne souhaite pas que C++ nous en glisse un par défaut

notes

résumé

0m 1s





sans qu'on le lui ait demandé. Nous allons voir un peu plus loin que C++2011 permet si on le souhaite d'éventuellement réactiver le constructeur par défaut par défaut et nous allons voir selon quelle syntaxe il va être possible de le signifier au compilateur.

notes

résumé

0m 49s



A :

```
class Rectangle {  
private:  
    double h; double L;  
    // suite ...  
};
```

Nous allons maintenant, sur des exemples concrets, examiner différentes variantes de déclaration de constructeur dans une même classe, et voir ce que ces variantes permettent ou ne permettent pas de faire en terme de déclaration initialisation d'une instance.

notes

résumé

1m 6s



A :

```
class Rectangle {
private:
    double h; double L;
    // suite ...
};
```

B :

```
class Rectangle {
private:
    double h; double L;
public:
    Rectangle()
        : h(0.0), L(0.0)
    {}
    // suite ...
};
```

C :

```
class Rectangle {
private:
    double h; double L;
public:
    Rectangle(double h=0.0,
               double L=0.0)
        : h(h), L(L)
    {}
    // suite ...
};
```

Dans la variante A la classe Rectangle prévoit des attributs comme tout à l'heure pour la largeur et la hauteur. Prévoit éventuellement d'autres choses mais ne prévoit aucun constructeur explicite dans la classe. Dans la variante B, la classe Rectangle prévoit un constructeur par défaut explicitement programmé. Ce constructeur a pour vocation d'initialiser les différents attributs à 0.

notes

résumé

1m 19s



constructeur par défaut	Rectangle r1;	Rectangle r2(1.0, 2.0);
-------------------------	---------------	-------------------------

Troisième variante, la variante C, dans cette variante, la classe Rectangle prévoit un constructeur dont tous les paramètres ont une valeur par défaut. Ce constructeur peut éventuellement être invoqué sans qu'on lui associe de valeur d'initialisation et dans ce cas là, c'est 0 qui est mis dans chacun des attributs. Il s'agit là donc d'un constructeur par défaut explicitement programmé. Enfin, dans la dernière variante la variante D, la classe Rectangle fournit un constructeur explicite qui attend deux valeurs d'initialisation, il n'y a donc aucun constructeur par défaut programmé explicitement dans la classe Rectangle.

notes

résumé

1m 49s



Constructeur par défaut : exemples

	constructeur par défaut	Rectangle r1;	Rectangle r2(1.0, 2.0);				
(A)	constructeur par défaut par défaut	<table><tr><td>?</td><td>?</td></tr><tr><td>L</td><td>h</td></tr></table>	?	?	L	h	Illicite !
?	?						
L	h						

```
class Rectangle {
};
```

Pour rappel, dans la variante A, voici quel était le schéma de la classe en terme de déclaration des constructeurs. Ici il n'est pas reprécisé la déclaration des attributs, qui est entendue pour toutes les variantes de la même façon. Ici dans la classe Rectangle, dans la variante A, il n'y avait aucun constructeur explicitement déclaré. Ceci signifie qu'en terme de constructeur par défaut, dans la variante A puisqu'il n'existe aucun constructeur explicitement spécifié, alors nous disposons du constructeur par défaut par défaut qui est automatiquement généré. Lorsque l'on procède à une déclaration de rectangle de cette façon, on ne précise aucune valeur d'initialisation et comme il n'y a pas de constructeur explicite dans ma classe Rectangle c'est le constructeur par défaut par défaut qui est invoqué or nous avons vu que ce constructeur initialisait les choses de façon minimale. Donc ici pour nos attributs largeur et hauteur, comme il s'agit d'attributs de type de base, c'était des double, le constructeur par défaut par défaut

notes

résumé

2m 49s



Constructeur par défaut : exemples

	constructeur par défaut	Rectangle r1;	Rectangle r2(1.0, 2.0);		
(A)	constructeur par défaut par défaut	<table><tr><td>?</td><td>?</td></tr></table>	?	?	Illicite !
?	?				
(B)	constructeur par défaut explicitement déclaré	<table><tr><td>0</td><td>0</td></tr></table>	0	0	Illicite !
0	0				

```
class Rectangle {
    Rectangle()
        : h(0.0), L(0.0)
    {}
};
```

ne met aucune valeur d'initialisation pour ces attributs. Au final donc notre rectangle r1 est un rectangle dont les champs largeur et hauteur demeurent non initialisés. L'objet existe mais ses attributs ne sont pas initialisés. Comme dans une classe où il n'y a aucun constructeur explicite le seul constructeur utilisable est le constructeur par défaut par défaut. il ne peut être utilisé que selon cette syntaxe qui ne requiert aucune valeur d'initialisation. Par conséquent, toute tentative d'initialisation d'un objet en utilisant des valeurs d'initialisation est tout simplement illicite. Ceci donnera lieu concrètement à une erreur décelable à la compilation. Parlons maintenant du cas B. Dans le cas B, nous disposons dans la classe Rectangle d'un constructeur par défaut mais cette fois-ci explicitement déclaré. Il avait pour vocation d'initialiser la largeur et la hauteur à 0. Donc dans ce cas ci le constructeur par défaut est le constructeur par défaut explicitement déclaré. Il est donc possible d'avoir recours à cette syntaxe qui ne requiert

notes

résumé

3m 49s



Constructeur par défaut : exemples

	constructeur par défaut	Rectangle r1;	Rectangle r2(1.0, 2.0);				
(A)	constructeur par défaut par défaut	<table><tr><td>?</td><td>?</td></tr></table>	?	?	Illicite !		
?	?						
(B)	constructeur par défaut explicitement déclaré	<table><tr><td>0</td><td>0</td></tr></table>	0	0	Illicite !		
0	0						
(C)	un des trois construc- teurs est par défaut	<table><tr><td>0</td><td>0</td></tr></table>	0	0	<table><tr><td>1</td><td>2</td></tr></table>	1	2
0	0						
1	2						

```
class Rectangle {
    Rectangle(double h=0.0,
               double L=0.0)
        : h(h), L(L)
    {}
};
```

Rectangle r1;
 Rectangle r(2.5);
 Rectangle r(2.5, 3.4);

aucune valeur d'initialisation mais contrairement au cas précédent cette fois-ci le constructeur par défaut qui est invoqué ici va proprement initialiser la largeur et la hauteur en leur mettant les valeurs 0. Vu qu'il n'y a dans la classe rectangle aucun constructeur qui attend de valeurs en guise de paramètre cette façon de déclarer/initialiser un objet Rectangle demeure dans ce second cas illicite. Dans la variante C, un peu plus complexe que les précédentes nous avons un constructeur explicitement programmé dont tous les paramètres ont une valeur par défaut. Ce qui veut dire concrètement que l'on peut invoquer ce constructeur désormais de trois façons possibles cette façon-ci qui veut dire que l'on est en train d'invoquer le constructeur en acceptant les valeurs par défaut. Ce qui veut dire que notre rectangle aura 0 et 0 en guise de valeur d'attribut. Selon cette seconde variante, qui signifie que l'on prend la valeur 2.5 pour la valeur du premier paramètre et que par contre on accepte la valeur par défaut 0 pour le second paramètre. Enfin dernière variante, il est également possible de déclarer un rectangle en lui fournissant deux valeurs qui veut dire que désormais on prend en guise de valeur de ce paramètre la valeur 2.5 et en guise de valeur de ce paramètre la valeur 3.4 Il y a donc pour la variante C un constructeur par défaut qui est l'un des trois constructeurs. Celui qui est invocable selon cette syntaxe et qui donne les valeurs 0 aux différents attributs. Par conséquent cette ligne est clairement licite et va correspondre à l'initialisation des attributs à 0. Cette ligne qui correspond à ce type d'appel est bien évidemment aussi licite et va permettre ici dans ce cas, d'initialiser la largeur à 1 et la hauteur à 2. Pour la variante D, nous avons dans la classe un

notes

résumé

4m 59s



Constructeur par défaut : exemples

	constructeur par défaut	Rectangle r1;	Rectangle r2(1.0, 2.0);				
(A)	constructeur par défaut par défaut	<table><tr><td>?</td><td>?</td></tr></table>	?	?	Illicite !		
?	?						
(B)	constructeur par défaut explicitement déclaré	<table><tr><td>0</td><td>0</td></tr></table>	0	0	Illicite !		
0	0						
(C)	un des trois construc- teurs est par défaut	<table><tr><td>0</td><td>0</td></tr></table>	0	0	<table><tr><td>1</td><td>2</td></tr></table>	1	2
0	0						
1	2						

```
class Rectangle {
    Rectangle(double h=0.0,
               double L=0.0)
    : h(h), L(L)
    {}
};
```

Rectangle r1;
 Rectangle r(2.5);
 Rectangle r(2.5, 3.4);

constructeur explicite qui n'est pas un constructeur par défaut. Sa liste de paramètres est non vide et ses paramètres n'acceptent pas de valeur par défaut. Nous avons également vu que dès l'instant où on programme un constructeur explicite le constructeur par défaut par défaut disparaît. Ce qui veut dire concrètement qu'il n'y a plus de constructeur par défaut du tout dans la variante D. Par conséquent une déclaration de cette nature devient du coup illicite. Bien sûr il est possible avec cette variante de déclarer/initialiser un rectangle comme ceci puisque le seul constructeur existant attend deux valeur d'initialisation. Au final, nous allons construire ici un rectangle comme tout à l'heure qui a une hauteur de 1 et une largeur de 2.

notes

résumé