

Support de cours

Cours:

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

Vidéo:

W12-02-constrdefaut-CPP-pt3

Concepts (extraits des sous-titres générés automatiquement) :

Valeur d'initialisation. Cas des constructeurs. Liste de paramètres vide. Initialisation d'un objet. Petite parenthèse de rappel. Déclaration d'un rectangle. Champ hauteur. Constructeur. Méthodes quelconques. Champ largeur. Position du rectangle. Défaut. Deuxième point de la séquence. Seul constructeur. Version minimale d'initialisation.



[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 3)

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

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

...

notes

résumé

0m 0s





Nous venons de voir qu'un constructeur par défaut est un constructeur

notes

résumé

0m 1s



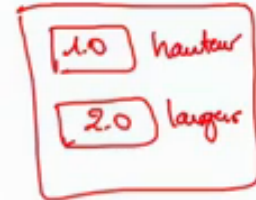
Constructeur par défaut : autre exemple

Autre façon de faire : regrouper les 2 premiers constructeurs en utilisant les valeurs par défaut des paramètres :

```
// DEUX constructeurs dont le constructeur par défaut
Rectangle(double c = 1.0) : hauteur(c), largeur(2.0*c)
{}

// 3ème constructeur
Rectangle(double h, double L) : hauteur(h), largeur(L)
{}
```

Rectangle r:



que l'on peut invoquer sans fournir aucune valeur d'initialisation. C'est le cas des constructeurs qui ont une liste de paramètres vide. C'est aussi le cas des constructeurs dont tous les paramètres auraient une valeur par défaut. En voici un exemple. Comme pour des fonctions ou des méthodes quelconques, un constructeur peut parfaitement avoir des valeurs par défaut pour certains de ces paramètres, voir tous. Ce constructeur est un constructeur par défaut car tous ces paramètres ont une valeur par défaut. Petite parenthèse de rappel, une valeur par défaut est une valeur que l'on peut associer à un paramètre selon cette syntaxe dans le prototype de la fonction ou de la méthode. Lorsqu'une méthode ou une fonction a une valeur par défaut, cela veut dire que l'on peut invoquer la méthode ou la fonction sans fournir la valeur. Dans ce cas là, c'est la valeur par défaut qui est prise. Donc ici, très concrètement, si je fais une déclaration d'un rectangle comme ceci, comme je n'ai fourni aucune valeur d'initialisation, on va chercher dans la classe un constructeur par défaut. C'est celui-ci car il est possible d'invoquer ce constructeur sans lui fournir de valeur, dans ce cas là il va prendre 1. Ce qui veut dire concrètement, que l'on va construire grâce à cet appel, un rectangle dont la valeur pour le champ hauteur serait la valeur ici donnée par défaut, c'est à dire 1. Et dont la valeur pour le champ largeur vaudrait deux fois cette valeur par défaut à savoir 2. Vous noterez au passage que le fait d'utiliser des valeurs par défaut pour les paramètres nous a permis d'écrire en un seul constructeur les deux constructeurs que nous avons ici à l'étape précédente. On écrit en un seul ce constructeur, invocable sans aucune valeur,

notes

résumé

0m 5s



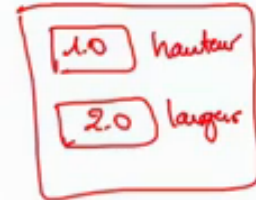
Constructeur par défaut : autre exemple

Autre façon de faire : regrouper les 2 premiers constructeurs en utilisant les valeurs par défaut des paramètres :

```
// DEUX constructeurs dont le constructeur par défaut
Rectangle(double c = 1.0) : hauteur(c), largeur(2.0*c)
{}

// 3ème constructeur
Rectangle(double h, double L) : hauteur(h), largeur(L)
{}
```

Rectangle ri



et celui-ci qui est invocable avec une valeur.

notes

résumé

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.

```
class Rectangle{
    double largeur;
    double hauteur;
    Position p;
```

Ce constructeur peut en effet parfaitement être invoqué de la façon suivante. On déclare une variable, mais cette fois-ci on précise une valeur autre que la valeur par défaut. Ce qui veut dire qu'on ne veut plus garder la valeur par défaut. On utilise le constructeur à un paramètre mais au lieu d'avoir la valeur par défaut pour C, on veut la valeur 5. On peut invoquer ce constructeur aussi bien selon cette syntaxe que celle-ci. Vous savez donc maintenant ce qu'est un constructeur par défaut. Intéressons-nous au deuxième point de la séquence à savoir : Que se passe-t-il si une classe ne contient aucun constructeur explicitement programmé ? En fait, l'initialisation d'un objet est quelque chose de tellement important que l'on ne peut pas s'en passer du coup C++ même si vous ne précisez aucun constructeur dans une classe va en fournir un de base automatiquement généré. Ce constructeur par défaut, qui ne requiert aucune valeur d'initialisation, qui est automatiquement généré, est ce que l'on peut appeler un constructeur par défaut par défaut. Il est fourni par défaut si on ne fournit aucun constructeur dans la classe et c'est évidemment un constructeur par défaut puisqu'il n'attend aucune valeur d'initialisation. Que va faire concrètement le constructeur par défaut par défaut ? Il va initialiser les attributs. Si ces attributs sont des objets alors pour les initialiser, il va utiliser les constructeurs par défaut associés à ces objets. Par contre si les attributs sont de type de base, ils resteront non initialisés. Prenons un exemple concret. Imaginons que j'écrive une classe Rectangle qui aurait pour attribut largeur et hauteur, de type de base comme tout à l'heure. On peut imaginer que ce rectangle, on souhaite lui associer un autre attribut qui serait cette fois un objet qui permettrait

notes

résumé

2m 1s



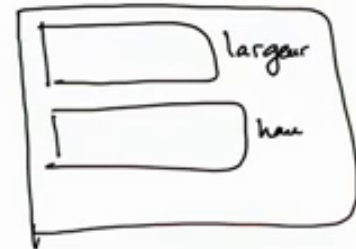
Constructeur par défaut par défaut

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.

```
class Rectangle{
    double largeur;
    double hauteur;
    Position p;
};
```

Rectangle r;



de modéliser par exemple la position du rectangle sur un écran par exemple. Si je procède à la déclaration d'un rectangle qui serait codé de cette façon. C'est possible, même s'il n'y a aucun constructeur explicite dans la classe. Ce qui veut dire qu'à ce moment là je vais invoquer le constructeur par défaut par défaut que fait ce constructeur ? Il utilise une version minimale d'initialisation qui va construire un objet où les champs largeur et les champs hauteur demeurent non initialisés car ils sont de type de base. Et où le champ position aurait une valeur donnée par le constructeur par défaut de la classe position s'il y en a un. On peut imaginer par exemple que la position soit une coordonnée en deux dimensions que le constructeur par défaut de position ait initialisé cette position à 0 ; 0. Dans ce cas-ci on aurait ces valeurs-là dans l'attribut position. Pour rappel on entend par type de base les types int, double, char ou bool les types int, double, char ou bool

notes

résumé

4m 1s

