

Support de cours

Cours:

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

Vidéo:

### W11-02-progclasses-CPP-pt6

Concepts (extraits des sous-titres générés automatiquement) :

**Propres attributs. Variable différente. Méthode surface de la classe. Méthode surface générale de la classe. Instance. Façon. Première séquence vidéo d'écriture concrète de vrais codes. Rect1.surface. Surface. Axe. Objet de la prochaine séquence vidéo. Rect1. Instances. Rectangles. Objet.**



[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/>

# Classes, objets, attributs et méthodes en C++ (Partie 6)

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

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

...

notes

résumé

0m 0s



```
// ...
class Rectangle {
    double hauteur;
    double largeur;
    double surface() const
    { return hauteur * largeur; }
};

int main()
{
    Rectangle rect1;

    rect1.hauteur = 3.0;
    rect1.largeur = 4.0;

    cout << "surface : " << rect1.surface()
        << endl;
    // ...
}
```

Il faut bien comprendre que quand j'appelle ici « rect1.surface »

notes

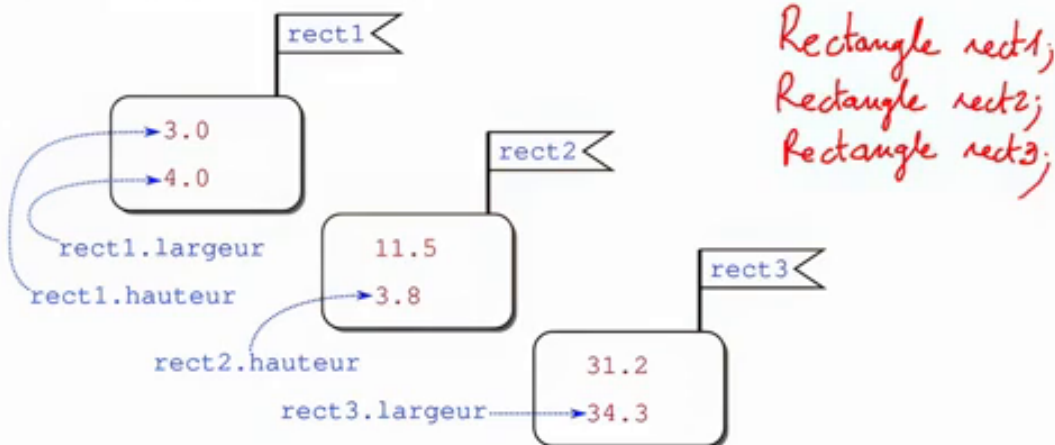
résumé

0m 1s



## Résumé : Accès aux attributs et méthodes

Chaque instance a ses propres attributs : aucun risque de confusion d'une instance à une autre.



`rect1.surface()` : méthode `surface` de la classe `Rectangle` s'appliquant à `rect1`  
`rect1.surface()`  $\simeq$  `Rectangle::surface(&rect1)`

`surface` s'applique bien sûr à « `rect1` » et prend la hauteur et la largeur de « `rect1` ». Chaque instance a ses propres attributs. Si j'avais donc déclaré comme cela, trois rectangles, que je les avais affectés respectivement la largeur de « `rect1` » à 4.0, sa hauteur à 3 « `rect2` », la largeur à 11.5, sa hauteur à 3.8 et d'autres valeurs encore pour « `rect3` ». Chacune de ces trois instances est une variable différente en mémoire qui a ses propres attributs et lorsqu'on va appeler « `rect1.surface` », c'est la méthode `surface` générale de la classe « `Rectangle` » qui va s'appliquer à l'objet « `rect1` » et qui dans sa définition, dans son corps, aura accès à la hauteur et à la largeur de « `rect1` ». Si j'appelle « `rect2.surface` », à ce moment-là, cela sera la hauteur et la largeur de « `rect2` ». D'une façon résumée, on peut dire que l'appel « `rect1.surface` » c'est un peu comme si on appelait la méthode `surface` de la classe « `Rectangle` » et on lui passait comme paramètres, l'adresse de « `rect1` ». Voilà ce qui conclut notre première séquence vidéo d'écriture concrète de vrais codes C++ où on a vu comment déclarer les attributs et des méthodes, on a vu essentiellement comment écrire cet axe ci, reste maintenant à voir comment écrire cet axe là qui est l'objet de la prochaine séquence vidéo. de la prochaine séquence vidéo.

notes

résumé

0m 13s

