

Support de cours

Cours:

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

Vidéo:

### W16-03-heritmultvirtuelles-CPP-pt1

Concepts (extraits des sous-titres générés automatiquement) :

**Classes virtuelles. Fameux exemple des ovovivipares. Classe ovipare. Classe animal. Héritage multiple. Hiérarchie de véhicules. Classe vivipare. Petite remarque générale. Défaut de ces super-classes. Nouveau concept. Diagrammes d'héritage. Genre de problèmes. Attributs de la classe animal. Classe iostream. Travers d'ovipare.**



[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/>  
page 1/13

# Classes virtuelles

## (Partie 1)

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

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

...

notes

résumé

0m 0s





Dans cette séquence vidéo,

notes

---

---

---

---

---

---

---

---

---

---

résumé

0m 1s



---

---

---

---

---

---

---

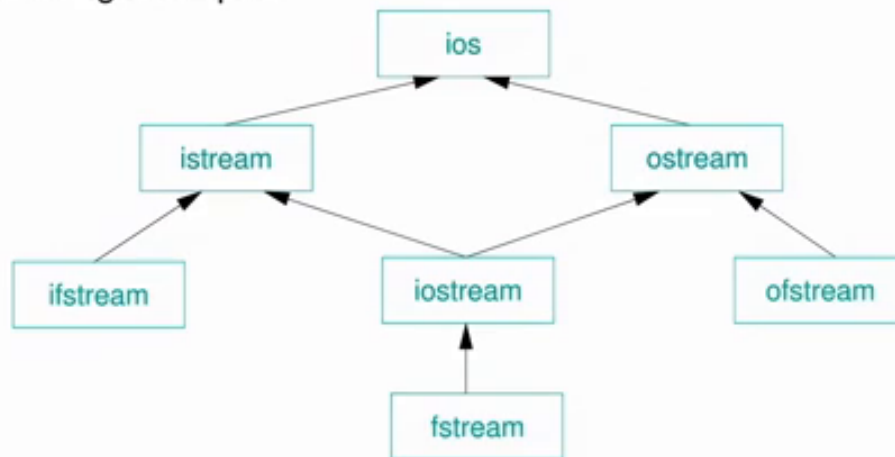
---

---

---

## Classes virtuelles : problème

Il peut se produire qu'une super-classe soit incluse *plusieurs fois* dans une hiérarchie à héritage multiple :



nous allons aborder le point le plus subtil, le plus délicat de l'héritage multiple, qui d'ailleurs fait que certains langages ne veulent pas avoir d'héritage multiple, ce que l'on appelle donc les «classes virtuelles». Mais commençons par illustrer le problème auquel répond ce nouveau concept. Reprenons pour cela notre fameux exemple des ovovivipares qui sont des ovipares et des vivipares, donc qui ont un héritage multiple par rapport à une classe Ovipare et une classe Vivipare et imaginons, ce n'est pas complètement loufoque, que la classe Ovipare hérite de la classe Animal (on dit qu'un Ovipare est un Animal). De même, un Vivipare est un Animal et donc Ovovivipare, d'un certain point de vue, est deux fois un Animal.

### notes

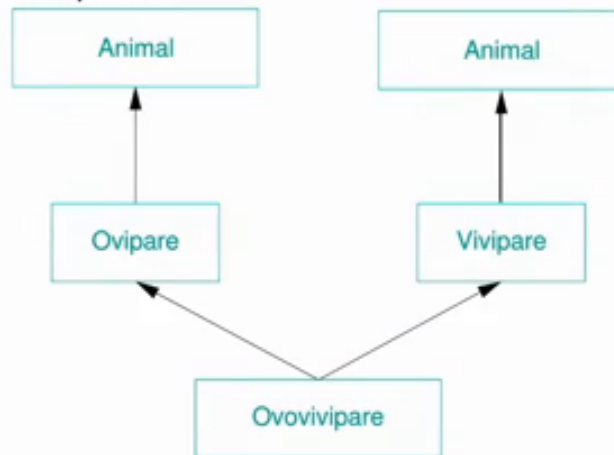
### résumé

0m 5s



## Classes virtuelles : problème

Il peut se produire qu'une super-classe soit incluse *plusieurs fois* dans une hiérarchie à héritage multiple :



Les attributs/méthodes de la super-classe seront inclus *plusieurs fois* !

- ☛ Chaque objet de la classe **Ovovivipare** possédera **deux** copies des attributs de la classe **Animal**.

Un autre exemple, un peu technique, tiré de la bibliothèque système de C++, dont le but n'est bien sûr pas de comprendre tous les détails mais d'illustrer que le genre de problèmes dont nous parlons ici existe bien dans la vraie vie. Donc avec la classe `iostream`, (dont vous avez déjà entendu parler avec `cout`) il se trouve que cette classe `iostream` est un `ostream` et est aussi un `istream`. Et puis les concepteurs de ces classes là ont décidé que chacun de `istream` et `ostream` étaient aussi des `ios`, peu importe ce que c'est ici. Donc on a ici aussi `iostream` qui est apparemment deux fois un `ios`. Une petite remarque générale : d'ailleurs, c'est comme ceci, en réalité, que l'on représente les diagrammes d'héritage, on ne met jamais plusieurs fois la classe sur un diagramme d'héritage mais on l'écrit qu'une seule fois et donc avec ici deux héritages qui reviennent sur la même classe. La raison pour laquelle je l'ai dessiné comme ceci sur ce diagramme là, c'est bien pour vous faire comprendre quel est le problème. Que signifie qu'un `ovovivipare` est un `ovipare` qui est un `animal` et un `ovovivipare` est un `vivipare` qui est un `animal` ? Est-ce que ça signifie qu'un `ovovivipare` est deux fois un `animal` ? Par exemple, si un `animal` a une tête, est-ce qu'un `ovovivipare` a deux têtes ? Est-ce qu'il aurait une tête d'`ovipare` et une tête de `vivipare` ? Ou est-ce qu'il n'a qu'une seule tête, Est-ce que c'est le même `animal` ? C'est bien là le cœur du problème et c'est bien à cause de ce genre de problème que certains langages ne veulent pas avoir d'héritage multiple. En C++ si l'on ne fait rien de particulier (et c'est pour cela que je l'ai dessiné comme ceci de façon peu habituelle), on va effectivement hériter

### notes

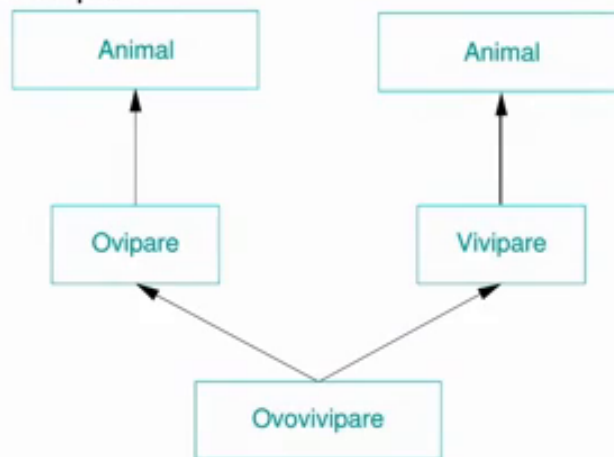
### résumé

0m 49s



## Classes virtuelles : problème

Il peut se produire qu'une super-classe soit incluse *plusieurs fois* dans une hiérarchie à héritage multiple :



Les attributs/méthodes de la super-classe seront inclus *plusieurs fois* !

- ☛ Chaque objet de la classe **Ovovivipare** possédera **deux** copies des attributs de la classe **Animal**.

deux fois de la classe **Animal** dans la classe **Ovovivipare**. Chaque objet de la classe **Ovovivipare** possédera deux fois les attributs de la classe **Animal**. Voyons ceci sur un exemple très concret :

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

.....

.....

.....

.....

## Classes virtuelles : exemple du problème

```
class Animal {
public:    Animal(string const& description) : tete(description) {}
protected: string tete;
};

class Ovipare : public Animal { public:    Ovipare() : Animal("à cornes") {} };
class Vivipare : public Animal { public:    Vivipare() : Animal("de poisson") {} };

class Ovovivipare : public Ovipare, public Vivipare {
public:
    void affiche() const { cout << "j'ai une tête " << Ovipare::tete
                                << " et une tête " << Vivipare::tete << " !" << endl;
                                }
};
// ...
Ovovivipare x;
x.affiche();
```

🖨 j'ai une tête à cornes et une tête de poisson !

Imaginons donc l'exemple suivant, écrit ici de façon un petit peu compacte pour tenir sur une seule image. On aurait donc la classe Animal avec ici un constructeur qui permet d'initialiser l'attribut qu'elle aurait qui serait ici une chaîne de caractères pour indiquer la tête qu'a cet animal. Donc on reçoit une chaîne de caractères et on initialise l'attribut tête avec le paramètre que l'on a reçu et puis on a une classe Ovipare qui hérite de la classe Animal et cette classe Ovipare redéfinit un constructeur par défaut, ici, j'ai pas voulu faire plus compliqué qui appelle donc le constructeur d'Animal en passant comme chaîne de caractères «à cornes». On va dire "il a une tête à cornes", par exemple. Et puis on a la classe Vivipare, ici, qui hérite aussi d'Animal et qui a un constructeur par défaut qui dit que l'animal a une tête de poisson par exemple. Et enfin on a une classe Ovovivipare qui hérite d'Ovipare et qui hérite de Vivipare, donc on a vraiment le diagramme que je vous ai présenté précédemment et qui aurait donc par exemple une méthode publique «affiche», ici, qui dit "j'ai une tête" en affichant la tête héritée d'Animal au travers d'Ovipare, ici, on retrouve l'opérateur de résolution de portée. On a bien une ambiguïté puisque Ovovivipare hérite effectivement de deux têtes, au travers, une, d'Ovipare et l'autre de Vivipare. Donc ici, pour résoudre l'ambiguïté, on va utiliser comme d'habitude, l'opérateur de résolution de portée. Donc il dit "j'ai une tête" et affiche la tête d'Ovipare, "et une tête", et affiche la tête de Vivipare et si par exemple, quelque part dans la fonction main ou ailleurs je déclare ici un ovovivipare x, c'est tout à fait possible puisque Ovovivipare a de toute façon un constructeur par défaut qui va être

notes

résumé

2m 37s



## Classes virtuelles : exemple du problème

```
class Animal {
public:    Animal(string const& description) : tete(description) {}
protected: string tete;
};

class Ovipare : public Animal { public: Ovipare() : Animal("à cornes") {} };
class Vivipare : public Animal { public: Vivipare() : Animal("de poisson") {} };

class Ovovivipare : public Ovipare, public Vivipare {
public:
    void affiche() const { cout << "j'ai une tête " << Ovipare::tete
                           << " et une tête " << Vivipare::tete << " !" << endl;
                           }
};
// ...
Ovovivipare x;
x.affiche();
```

🖨 j'ai une tête à cornes et une tête de poisson !

construit par défaut, lequel va appeler les deux constructeurs par défaut de ces super-classes

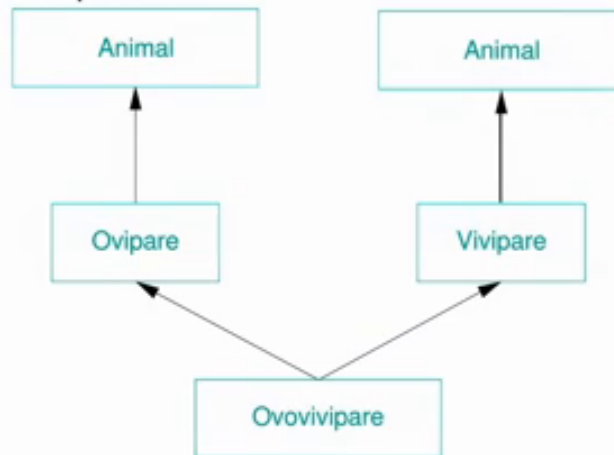
notes

résumé



## Classes virtuelles : problème

Il peut se produire qu'une super-classe soit incluse *plusieurs fois* dans une hiérarchie à héritage multiple :



Les attributs/méthodes de la super-classe seront inclus *plusieurs fois* !

- Chaque objet de la classe `Ovovivipare` possédera **deux** copies des attributs de la classe `Animal`.

dans l'ordre de déclaration de l'héritage, ici au niveau de la classe. Donc ici, on va appeler les constructeurs d'Ovipare et de Vivipare. On peut tout à fait faire cette construction par défaut ici. Et puis ensuite, si on affiche, on va bien avoir l'affichage pour cet Ovovivipare : "j'ai une tête à corne", qui est le résultat de l'initialisation qui a eu lieu au travers du constructeur par défaut d'Ovipare "et une tête de poisson" qui est le résultat de l'affichage de «Vivipare: :tete» qui a été initialisé au travers du constructeur par défaut de la classe Vivipare. Donc la classe Ovovivipare a effectivement bien hérité de deux têtes il y a bien deux animaux dans la classe Ovovivipare. Donc ici, bien sûr,

notes

résumé

4m 37s





c'est un petit peu indésirable qu'un ovovivipare récupère deux têtes au travers de cet héritage multiple et qu'il ait deux copies d'Animal mais à noter qu'il y a certains cas

notes

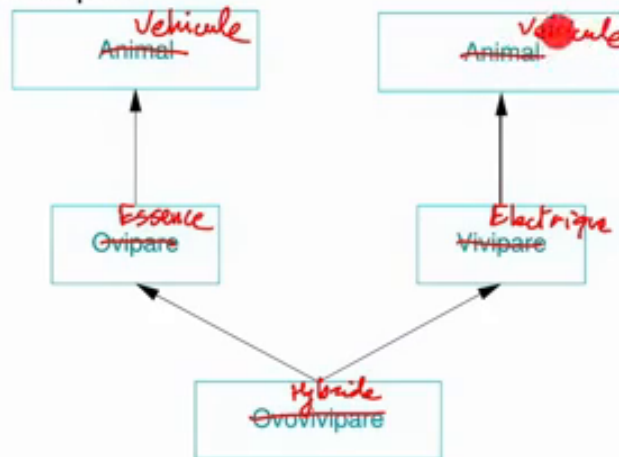
résumé

5m 25s



## Classes virtuelles : problème

Il peut se produire qu'une super-classe soit incluse *plusieurs fois* dans une hiérarchie à héritage multiple :



Les attributs/méthodes de la super-classe seront inclus *plusieurs fois* !

- Chaque objet de la classe **Ovovivipare** possédera **deux** copies des attributs de la classe **Animal**.

ou l'on peut très bien vouloir hériter effectivement deux fois les attributs. Admettons que par exemple, on ait une hiérarchie de véhicules et que l'on ait, par exemple, des véhicules «à essence» et des véhicules «électriques» et que l'on imagine avoir des véhicules «hybrides», qui sont à la fois un véhicule «à essence» et un véhicule «électrique». La question, c'est donc : est-ce que «hybride» est un seul véhicule, comme les animaux avec les ovovivipares ? Ou effectivement deux véhicules ? et tout ceci va dépendre de ce que l'on entend par «véhicule». Si l'on entend qu'un véhicule, ça a quatre roues, un volant, bien sûr, on voudrait qu'un véhicule hybride n'ait qu'une seule fois quatre roues et un volant et certainement pas huit roues et deux volants. Si, par contre, on entend qu'un véhicule a un moteur, et donc là, on a un moteur «électrique»

notes

résumé

5m 34s





et là on a un moteur «à essence» alors dans ce cas là, on souhaitera effectivement avoir deux moteurs dans «hybride».

notes

---

---

---

---

---

---

---

---

---

---

résumé

6m 25s



---

---

---

---

---

## Classes virtuelles solution

Pour éviter la duplication des attributs d'une super-classe plusieurs fois incluse lors d'héritages multiples, il faut déclarer son **lien d'héritage** avec toutes ses sous-classes comme **virtuel**.

Cette super-classe sera alors dite « **virtuelle** »  
(à ne pas confondre avec classe abstraite !!)

Syntaxe :

```
class NomSousClasse : public virtual NomSuperClasseVirtuelle
```

Exemple :

```
class Ovipare : public virtual Animal { // ...  
// ...  
class Vivipare: public virtual Animal { // ...
```

A noter que c'est la classe pouvant être héritée plusieurs fois qui est virtuelle (i.e. ici la super-super-classe) et non pas directement les classes utilisées dans l'héritage multiple (i.e. les super-classes).

Donc vous voyez comme ce problème est quand même assez subtil et relève vraiment de la conception et de bien savoir ce que l'on veut dire par la relation d'héritage, en particulier lors d'un héritage multiple. Si l'on ne souhaite donc pas recevoir plusieurs fois les attributs d'une super-super-classe, alors il va falloir faire quelque chose de particulier pour l'empêcher et l'on va devoir déclarer son lien d'héritage comme un lien virtuel. A nouveau le mot clé "virtuel", ici, mais pour autre chose, ici, ça va être le lien d'héritage qui va être virtuel. Et donc on dira qu'une super-classe, comme ceci, qui a des liens d'héritage vers ses sous classes comme virtuels va être appelé une super-classe «virtuelle» Attention, ça n'a rien a voir avec une classe «abstraite» ! avec une classe «abstraite» !

notes

résumé

6m 34s

