

Support de cours

Cours:

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

Vidéo:

INIT-CPP-7_5-PointeursAllocation-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Moment de la programmation. Exemple typique. Ligne de code. Allocation dynamique. Exécution du programme. Séquence d'introduction. Nouvelle case. Vector d'entier v. Cas particulier des pointeurs. Moment de l'exécution du programme. Allocation de mémoire nécessaire. Besoin d'une zone mémoire. Cas d'allocation dynamique. Adresse d'un objet. Première façon.



[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/>

Pointeurs : allocation dynamique

(Partie 1)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes


résumé

0m 0s



Les « pointeurs », à quoi ça sert ?

En programmation, les « pointeurs » servent essentiellement à trois choses :

- ① à permettre à plusieurs portions de code de *partager* des objets (données, fonctions,..)
sans les dupliquer
☞ **référence**
- ② à pouvoir *choisir des éléments* non connus *a priori* (au moment de la programmation)
☞ **généricité**
- ③ à pouvoir manipuler des objets dont la *durée de vie* dépasse la portée
☞ **allocation dynamique**
(moins important depuis  en raison de la *move semantic*)



Dans la séquence d'introduction sur les pointeurs, vous avez vu

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

0m 1s



.....

.....

.....

.....

.....

Il y a **deux façons** d'*allouer de la mémoire* en C++.

que ces derniers servent essentiellement trois objectifs : ils permettent le référencement, c'est-à-dire pointer indirectement sur une donnée, la généricité, choisir, sélectionner une donnée non-connue au moment de la programmation, et enfin, ils servent également à ce qu'on appelle l'allocation dynamique. C'est ce volet qui va nous occuper maintenant. Lorsqu'un programme a besoin de manipuler une donnée, il va

notes

résumé

0m 5s



Il y a deux façons d'allouer de la mémoire en C++.

`int i;` → allocation statique

être nécessaire d'allouer de la mémoire pour justement y stocker cette donnée. Et en C++, vous disposez de deux façons possibles d'allouer de la mémoire. La première façon que vous connaissez déjà est ce que vous faites lorsque vous écrivez une ligne de code comme celle-ci, une déclaration de variable. À ce moment-là, il y a ce qu'on appelle une allocation statique. Pourquoi parle-t-on de statique ? Eh bien, c'est par le fait que l'on n'a pas besoin

notes

résumé

0m 25s



Il y a deux façons d'allouer de la mémoire en C++.

Vecto



d'attendre l'exécution du programme pour connaître le besoin en mémoire. Ici, au moment où l'on compile cette ligne de code, eh bien on sait déjà que le programme va avoir besoin d'une zone mémoire dédiée à contenir un entier, et qui s'appelle v. Il existe par contre des situations où le besoin en mémoire s'avère seulement au moment de l'exécution du programme. Et un exemple typique est lié à l'utilisation des tableaux dynamiques

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

.....

.....

.....

.....

.....

0m 49s



Il y a deux façons d'allouer de la mémoire en C++.



Vector
V. plus

de type vector que vous avez eu l'occasion de croiser. Supposons que dans un programme jaillit un vector d'entier v,

notes

résumé

1m 13s



Il y a deux façons d'allouer de la mémoire en C++.



Vector
→ `V.push_back(3);`

et je peux parfaitement, donc, ajouter, pendant que le programme s'exécute, une nouvelle case, une nouvelle cellule à mon tableau, par le biais d'une ligne comme celle-ci. C'est au moment où j'exécute cette ligne de code que cette fonctionnalité

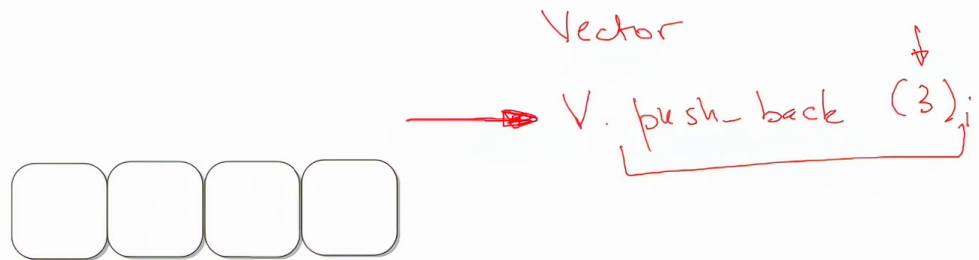
notes

résumé

1m 21s



Il y a deux façons d'allouer de la mémoire en C++.



va faire l'allocation de mémoire nécessaire à contenir ce nouvel entier. Donc, c'est seulement au moment où cette ligne-là s'exécute que l'allocation a lieu, que le nouvel emplacement mémoire est réservé pour les besoins

notes

résumé

1m 37s



Il y a deux façons d'allouer de la mémoire en C++.



Vector
→ `V.push_back(3);`

allocation
dynamique

de votre programme. Donc bien sûr, nous parlons dans ce cas d'allocation dynamique, parce que nous avons besoin d'attendre l'exécution du programme, non pas simplement la compilation pour déterminer le besoin en mémoire.

notes

résumé

1m 45s



Il y a deux façons d'allouer de la mémoire en C++.



Vector
→ `V.push_back(3);`

allocation
dynamique

Pourquoi ? Parce qu'il n'est pas sûr que cette ligne soit forcément exécutée,

notes

résumé

2m 1s



Il y a **deux façons** d'allouer de la mémoire en C++.

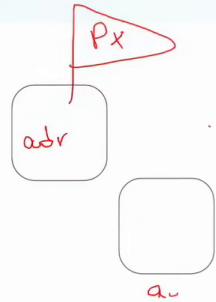
① *déclarer des variables*

La réservation de mémoire est déterminée à la compilation :
allocation statique

② **allouer dynamiquement** de la mémoire **pendant l'exécution** d'un programme.

Les structures dynamiques comme les tableaux de taille variable (**vector**) ou les chaînes de caractères de type **string** sont des exemples de ce type.

Dans le cas particulier des *pointeurs*, l'**allocation dynamique** permet également de réserver de la mémoire **indépendamment de toute variable** : on pointe directement sur une zone mémoire plutôt que sur une variable existante.



donc nous ne pouvons pas, au moment de la compilation, savoir quelle zone mémoire réserver. Nous devons attendre l'exécution. Ici, la zone mémoire que nous avons allouée dynamiquement via le `push_back` a été allouée pour contenir la valeur 3, et nous nous trouvons donc dans cette situation. Vous noterez que dans le cas particulier des pointeurs, l'allocation dynamique va permettre de manipuler des données sans pour autant qu'elles soient associées à des noms explicites, sans pour autant qu'elles correspondent à des variables. Par exemple, si dans un programme, on dispose, on a déclaré un pointeur `px`, ce dernier peut parfaitement contenir l'adresse d'un objet que l'on aurait alloué dynamiquement pendant que le programme s'exécute. Donc, on peut parfaitement mettre dans ce pointeur l'adresse de ce nouvel objet.

notes

résumé

2m 3s





Et ce que l'on peut constater, c'est qu'il n'est pas nécessaire que cet objet ait un nom, il suffit d'y accéder par le biais du pointeur qui y est associé. du pointeur qui y est associé.

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

2m 49s



.....

.....

.....

.....

.....