Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Introduction (partie 2)

Concepts (extraits des sous-titres générés automatiquement) :

Sortes de pointeurs. Nouvelle norme c. Cas numéro. Vrais pointeurs. Façon générique différents. Pointeur générique. Programmeur débutant. Situations d'utilisation des pointeurs. Pointeurs intelligents. Objets possibles. Cas d'utilisation. Exemple précédent. Smart pointers. Façon générique. Cas d'utilisation numéro.

vers la recherche de séquences vidéo
(dans Initiation à la programmation (en C++).)

vers la vidéo

EPFL

Pointeurs et références : introduction
(Partie 2)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Les différents « pointeurs »

EPFL

En C++, il existe plusieurs sortes de « pointeurs » :

- les **références**

  totalement gérées en interne par le compilateur. Très sûres, donc; mais sont *fondamentalement différentes des vrais pointeurs.*

- C++11 les « pointeurs intelligents » (***smart pointers***)

  gérés par le programmeur, mais avec des gardes-fous.
  Il en existe 3 : `unique_ptr`, `shared_ptr`, `weak_ptr`
  (avec `#include <memory>`)

- les « **pointeurs « à la C »** » (*build-in pointers*)

  les plus puissants (peuvent tout faire) mais les plus « dangereux »

En C++, et surtout depuis la nouvelle norme C++ 2011, il existe en fait trois sortes de pointeurs entre guillemets. Il existe les références, qui ne sont pas à strictement parler de vrais pointeurs et sont même en fait assez fondamentalement différentes des vrais pointeurs, en ce sens qu'elles sont gérées complètement par le compilateur. Et ce sont donc des objets très sûrs qui ne sont pas manipulés

notes

résumé

0m 1s

Les différents « pointeurs »

EPFL

☞ lesquels utiliser ?

par le programmeur lui-même, mais par le compilateur. Ensuite, nous avons depuis le nouvelle norme C++ 2011 des pointeurs intelligents, ce qu'on appelle des smart pointers, et puis enfin nous avons les anciens pointeurs hérités du langage C ancêtre du C++, qui sont vraiment les pointeurs très puissants, les pointeurs à tout faire. Ce sont donc ceux-ci qu'on appelle simplement pointeurs quand on ne spécifie pas plus particulièrement de quelle sorte de pointeur entre guillemets il s'agit. Avant de rentrer dans les détails de chacun de ces nouveaux types, la question que l'on peut se poser c'est lesquels devons-nous utiliser dans quelle situation.

notes

résumé

0m 25s

Les différents « pointeurs »

EPFL

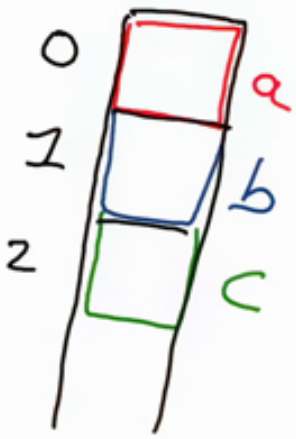
☞ lesquels utiliser ?

« Utilisez des références quand vous pouvez, des pointeurs quand vous devez. »

1) référence : références
(ou pointeurs « à la C »)

2) généricité : pointeurs « à la C »
ou index dans un tableau
(le tableau des choix/des possibles, s'il existe)

allocation dynamique : smart-pointers,
surtout `unique_ptr` dans un premier temps
(ou pointeurs « à la C »)



La règle générale est d'utiliser des références chaque fois qu'on le peut et d'utiliser des pointeurs quand on le doit. Reprenons tour à tour les trois situations d'utilisation des pointeurs. Dans le cas numéro un où l'on veut utiliser des pointeurs pour des références, bien sûr dans ce cas-là il vaut mieux utiliser des vraies références à ce moment-là plutôt que des pointeurs ou alors on peut utiliser aussi des pointeurs « à la C » puisque ces pointeurs sont universels et peuvent servir à tout. Mais en tant que programmeur débutant, si vous avez le choix et que vous êtes dans un cas d'utilisation donc de référence, alors utilisez des références que nous allons détailler juste dans la séquence qui suit. Si jamais vous êtes dans le cas d'utilisation numéro deux, c'est-à-dire que vous voulez utiliser un pointeur générique qui décrit de façon générique différents objets possibles que vous ne connaissez peut-être pas encore, alors à ce moment-là, ce qu'il faut utiliser ce sont des pointeurs « à la C ». Mais il existe aussi un autre cas particulier où les objets en question qu'on veut décrire de façon générique, je les avais appelés a, b et c dans l'exemple précédent, donc supposons que ces objets existent déjà. Et supposons justement qu'ils aient été du coup stockés puisqu'ils existent déjà, stockés dans un tableau. A ce moment-là, il n'y a même pas besoin du tout de compteur, on peut simplement utiliser les index, les positions dans ce tableau

notes

résumé

1m 1s

comme décrivant de façon générique, donc par exemple, par un size_t i comme décrivant de façon générique si i vaut zéro ça décrira a, si i vaut un ça décrira b, si i vaut deux ça décrira c. C'est donc qu'il est bien aussi d'avoir présent à l'esprit que cette possibilité-là peut aussi exister en rajoutant une structure de données supplémentaires, ici un tableau. On arrive aussi, si tant est que les objets existent déjà au moment où on crée le tableau, à, ou alors qu'on peut les rajouter au fur et à mesure dans un tableau dynamique par des push back et des popback, à aussi donc décrire de façon générique avec une abstraction de plus qui est l'index de l'élément dans le tableau

notes

résumé

2m 25s

Les différents « pointeurs »

EPFL

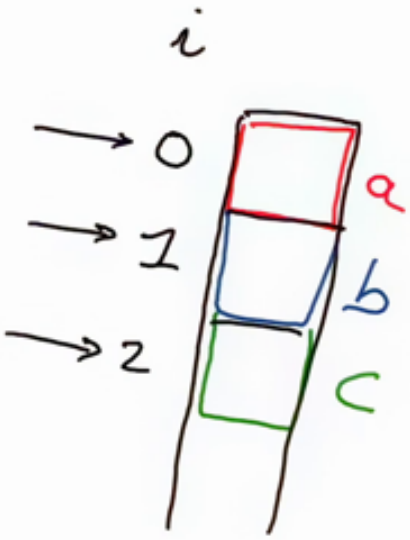
☞ lesquels utiliser ?

« Utilisez des références quand vous pouvez, des pointeurs quand vous devez. »

1) référence : références (ou pointeurs « à la C »)

2) généricité : pointeurs « à la C » ou index dans un tableau (le tableau des choix/des possibles, s'il existe)

3) allocation dynamique : smart-pointers, surtout `unique_ptr` dans un premier temps (ou pointeurs « à la C »)



plutôt que l'élément lui-même. Et si vous voulez vraiment des pointeurs, donc dans ce cas d'utilisation numéro deux, alors ce seront des pointeurs « à la C » qu'il faudra utiliser. Dans le cas d'utilisation numéro trois, pour faire de l'allocation dynamique, alors là il fait bien sûr utiliser vraiment que des pointeurs, ils sont faits pour ça, et dans ce cas-là, je vous conseille d'utiliser C++ 2011 et les pointeurs intelligents, ce qu'on appelle aussi les smart pointers,

notes

résumé

3m 1s

parmi lesquels je vous recommande surtout d'utiliser, commencer par unique_ptr qui sera détaillé donc dans une séquence vidéo suivante. Voilà donc pour les différents cas d'utilisation de toutes ces sortes de pointeurs entre guillemets. Les séquences vidéos suivantes vont maintenant entrer dans les détails tour à tour des références, des pointeurs « à la C », puis nous finirons par les pointeurs intelligents de C++ 2011. intelligents de C++ 2011.

notes

résumé

3m 25s