Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Pointeurs - allocation dynamique (partie 4)

Concepts (extraits des sous-titres générés automatiquement) :

Emplacement mémoire capable. Zone mémoire. Déclaration d'une variable px. Première variante du programme. Pointeur capable. Emplacement mémoire. Petit programme. Pointeur px. Bonne valeur. Deuxième ligne de code. Segmentation fault. Zone mémoire du pointeur. Premiers programmes. Moment de l'exécution. Erreur redoutable.

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

vers la vidéo

EPFL

Pointeurs : allocation dynamique
(Partie 4)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

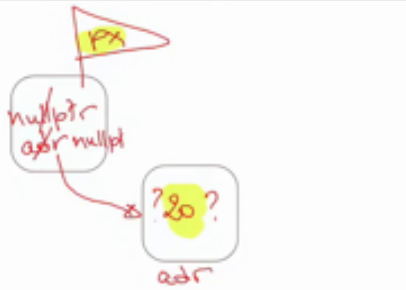
notes

résumé

0m 0s

Exemple

```
int* px(nullptr);
px = new int;
*px = 20;
cout << *px << endl;
delete px;
px = nullptr;
```



Voilà, vous connaissez maintenant beaucoup de choses essentielles sur les pointeurs et on va résumer tout ceci sur un petit exemple. Alors ici, un petit programme que l'on va dérouler pas à pas. Ici, il y a déclaration d'une variable px, qui est capable de pointer, qui est un pointeur capable de pointer sur un entier, et je prends la précaution de l'initialiser à nullptr, ce qui veut dire qu'il ne pointe sur rien. Deuxième ligne de code, il y a une allocation dynamique. Cette allocation dynamique réserve un emplacement mémoire capable de contenir un entier. Et cet emplacement mémoire a une adresse, c'est exactement cette adresse qui résulte de l'exécution de "new int", et c'est cette adresse que je vais affecter à px. Donc désormais, je fais pointer le pointeur px sur cette zone mémoire nouvellement allouée. Ensuite, troisième instruction. J'utilise la fameuse petite étoile pour accéder au contenu pointé par px. Le contenu pointé par px est ceci. A l'intérieur de ce contenu pointé, je vais mettre la valeur 20, ce qui correspond à ceci. Ensuite, je veux m'assurer du fait que j'ai bien mis la bonne valeur dans cette zone mémoire, et donc, je veux en faire afficher le contenu. A nouveau, j'accède au contenu pointé par px au travers de la petite étoile, et ceci va donc simplement m'afficher la valeur 20, suivi d'un retour de ligne, d'un saut de ligne. Ensuite je présume que je n'ai plus besoin de ma zone mémoire, et donc je fais un delete, ce qui veut dire que cette zone mémoire n'a plus un contenu qui est utilisable. Et je veux m'assurer du fait que px n'a plus un contenu qui est obsolète, qui ne pointe vers aucune adresse utilisable de façon sûre, et donc,

notes

résumé

0m 1s

Exemple

```
int* px(nullptr);
px = new int;
*px = 20;
cout << *px << endl;
delete px;
px = nullptr;
```

```
int* px(nullptr);

px = new int(20);
...
```

à nouveau, je mets nullptr dans ma zone mémoire du pointeur.

notes

résumé

1m 49s

La première variante du programme que nous avons examiné peut s'écrire de façon plus concise, sachant que ces deux instructions, qui consistent à allouer l'emplacement mémoire et y mettre une valeur à l'intérieur, peut en fait être rédigée en une seule et même ligne par le biais de cette syntaxe. De façon encore plus concise encore, je peux remplacer ces deux lignes de code par une seule ligne, qui est celle-ci. Dans cette ligne, je commence par faire la déclaration du pointeur, seulement au lieu de commencer par l'initialiser à nullptr pour ensuite mettre à l'intérieur le résultat de "new int(20)", eh bien je mets directement la valeur "new int(20)" dans px. Ce qui fait que je fais l'économie d'une instruction. Lorsque vous écrirez vos premiers programmes avec des pointeurs,

notes

résumé

1m 54s

et même les suivants d'ailleurs, il est vraisemblable que, de temps en temps, vous tombiez sur une erreur redoutable, le "segmentation fault", qui va causer un arrêt abrupt de votre programme. De quoi s'agit-il ?

notes

résumé

2m 37s

Toujours allouer avant d'utiliser !

Conseil – Bonnes pratiques

Initialisez toujours vos pointeurs. Utilisez `nullptr` si vous ne connaissez pas encore la mémoire pointée au moment de l'initialisation :

```
int* px(nullptr);
```

Typiquement, les erreurs de type "segmentation fault" vont se produire lorsqu'on essaie, via un pointeur, d'accéder à une zone mémoire qui n'a pas été réservée. Donc, si on regarde ici cet exemple, cette instruction, donc, déclare un pointeur px mais ne réserve aucun emplacement mémoire alloué. Il n'y a pas d'initialisation, pas de new, rien du tout. Donc ici, le contenu de px est inconnu et ici, ce qu'on essaie de faire c'est d'accéder au contenu pointé, donc à une éventuelle case pointée par px, mais qui n'existe pas, puisqu'on ne l'a jamais allouée, et pour y mettre la valeur 20 à l'intérieur. Evidemment, ce n'est pas possible, et cela va résulter en une erreur de type segmentation fault, qui va causer justement l'arrêt de votre programme. La bonne solution consiste évidemment, ici, à allouer un emplacement mémoire à associer au pointeur px, ce qui se traduit par ceci, au moment de l'exécution, donc on alloue la zone mémoire, et à ce moment-là, on aura notre pointeur px qui va contenir l'adresse de cet emplacement mémoire, nouvellement alloué. Et donc, évidemment, une fois que cette zone mémoire existe, on peut parfaitement y accéder, accéder à son contenu pour y mettre la valeur 20. Donc, à ce moment-là, cette instruction va parfaitement se dérouler. Nous venons de voir qu'il était impératif d'allouer avant d'utiliser, il en suit un conseil, une bonne pratique courante concernant l'initialisation des pointeurs. Il est en fait conseillé de toujours initialiser les pointeurs, ne serait-ce qu'à la valeur nullptr. Donc, si au moment où on déclare un pointeur, on ne connaît pas encore la mémoire que ce pointeur pointe, alors il est conseillé de l'initialiser au moyen de nullptr, qui veut dire que le pointeur ne pointe vers rien,

notes

résumé

2m 46s

mais qui le dit explicitement. Et grâce à cette initialisation, il devient possible de mettre en place un certain nombre de garde-fous, par exemple avant d'accéder à la zone mémoire pointée par un pointeur, je peux désormais prendre la précaution de contrôler si le pointeur en question pointe effectivement vers quelque chose ou non. Donc, il devient possible, grâce à cette initialisation, de tester si le pointeur pointe vers quelque chose avant d'accéder à la zone pointée, comme je pourrais le faire ici. Donc, on voit que, grâce à cette initialisation, un certain nombre de garde-fous peuvent être mis en place pour l'accès aux zones mémoires pointées par des pointeurs. pointées par des pointeurs.

notes

résumé

4m 25s