Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Array (partie 1)

Concepts (extraits des sous-titres générés automatiquement) :

Tableaux dynamiques. Nouvelle norme c. Langage c. Tableaux de taille fixe. Types de tableaux de taille. Tableaux statiques. Nouvelle norme. Séquences vidéos précédentes. Fonction des compilateurs. Arguments de la fonction. Norme c. Syntaxe d'initialisation particulière. Programme de jeu d'échec. Heure actuelle. Nombre de cantons.

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

vers la vidéo

EPFL

Tableaux : array
(Partie 1)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

		taille initiale connue *a priori* ?	
		non	oui
taille pouvant varier lors de l'utilisation du tableau ?	oui	`vector`	(`vector`)
	non	(`vector`)	`array` (C++11) tableaux « à la C »

Dans les séquences vidéos précédentes, nous nous sommes surtout intéressés aux tableaux dynamiques. C'est-à-dire aux tableaux dont la taille peut changer lorsque le programme se déroule.

notes

résumé

0m 1s

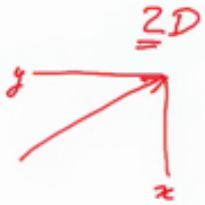
Intéressons-nous ici aux tableaux dont la taille est connue à priori. C'est-à-dire est connue au moment où on écrit le programme, et dont la taille ne peut pas varier lors de l'utilisation. Par exemple si l'on veut faire un programme de géométrie dans le plan avec des vecteurs en dimensions 2, on aura toujours un tableau qui contient les deux coordonnées. Ce chiffre est connu à l'avance : c'est 2. Il ne varie pas au fur et à mesure que le programme évolue. Un autre exemple, c'est si vous voulez faire un programme de jeu d'échec, Vous savez au préalable, avant d'écrire le programme, qu'il y a par exemple deux joueurs. Et vous savez que ce nombre ne va pas changer.

notes

résumé

0m 13s

		taille initiale connue *a priori* ?	
		non	oui
taille pouvant varier lors de l'utilisation du tableau ?	oui	`vector`	(`vector`)
	non	(`vector`)	`array` (C++11) tableaux « à la C »



Vous savez aussi que le jeu d'échec, ça va toujours être 8 fois 8 cases. Vous connaissez ce nombre au préalable et ce nombre ne va pas changer.

notes

résumé

0m 49s

Inconvénients des tableaux de taille fixe à la C

EPFL

Les tableaux de taille fixe à la C :

- sont toujours passés par référence
- n'ont pas connaissance de leur propre taille
- ne peuvent pas être manipulés globalement (pas de « = »)
- ne peuvent pas être retournés par une fonction
- ont une syntaxe d'initialisation particulière

☞ **AUCUN** avantage !

Mais ils resteront malgré tout certainement assez répandus (inertie)... :-(

Pour ceux que cela intéresse : voir annexe à ce cours.

On est donc dans ce cas où la taille est connue à priori et où elle n'évolue pas. Donc, dans cette partie là du tableau. Ce que l'on appelle des tableaux statiques ou des tableaux de taille fixe. Il existe en C++, depuis la nouvelle norme C++-2011 deux types de tableaux de taille fixe. Les tableaux « array » introduits avec cette nouvelle norme en C++-2011 et depuis le début, les tableaux hérités de l'ancêtre du C++, le langage C, qu'on appellera les « tableaux à la C ». Dans le cadre de cette vidéo, nous allons maintenant nous focaliser sur les « arrays ». Les « arrays » ont été introduits dans la nouvelle norme C++-2011 parce que les « tableaux à la C », ou « building arrays », présentent un certain nombre d'inconvénients. Tout d'abord, ils sont systématiquement passés par référence

notes

résumé

0m 59s

même si on ne met pas le petit signe de passage par référence aux arguments de la fonction. Les « tableaux à la C » sont passés par référence. Ensuite, ils n'ont jamais connaissance de leur propre taille. Alors qu'avec les « arrays » on a comme avec les « vectors » un moyen simple de connaître la taille. Ensuite, les « tableaux à la C » ne peuvent pas être manipulés globalement. Si j'ai un tableau « tab1 » et un tableau « tab2 » je ne peux pas faire avec les « tableaux à la C » « tab1 » égal « tab2 » Ensuite, on ne peut pas retourner un « tableau à la C » d'une fonction. On ne peut pas avoir un type de retour de fonction qui est un « tableau à la C ». Et enfin, les « tableaux à la C » ont une syntaxe d'initialisation particulière pas aussi générale que, par exemple, les « vectors ». Bref, je pense que depuis la norme C++-2011, depuis les « arrays », les « tableaux à la C » n'ont aucun avantage. Ceci dit, ils étaient très répandus et je pense que par inertie,

notes

résumé

1m 49s

C++11 Déclaration d'un tableau de taille fixe

EPFL

En C++11, le type « tableau de taille fixe » est défini dans la bibliothèque `array`.

Pour les utiliser, il faut ajouter en début de programme :

```
#include <array>
```

Une variable correspondant à un tableau de taille fixe se déclare de la façon suivante :

```
array<type, taille> identificateur;
```

où *identificateur* est le nom du tableau, *type* correspond au type des éléments du tableau et *taille* est le nombre d'éléments que contient le tableau.

Ce nombre doit être **connu** au moment où on écrit le programme (→ sinon `vector`)

par habitude, ils vont être largement utilisés, Donc, pour ceux qui voudraient étudier ce sujet, nous mettrons une annexe sur les « tableaux à la C ». Passons maintenant aux « arrays » de C++-2011. Pour utiliser un tableau de taille fixe, un tableau dont on connaît la taille au moment où on écrit le programme, et c'est vraiment ça qui va déterminer si vous utilisez un « array » ou un « vector »

notes

résumé

2m 37s

C++11 Déclaration d'un tableau de taille fixe

EPFL

Exemples :

```
#include <array>
...
array<double, 3> vecteur_3d;
```

correspond à la déclaration d'un *tableau de 3* `double`.

```
const size_t nb_cantons(26);
array<unsigned int, nb_cantons> habitants;
```

correspond à la déclaration d'un tableau de 26 entiers.

pour déclarer un tableau de taille fixe, on va utiliser le type « array » il faut bien sûr faire commencer le programme avec la directive « #include ». Ensuite, un tableau va se déclarer, donc avec le mot-clef « array » Puis on mettra derrière le type et la taille, qui justement est connue. C'est cette taille qui est nouvelle par rapport aux vecteurs. Elle est justement connue au moment où on écrit le programme. Puis on terminera, comme d'habitude, avec le nom que l'on veut donner au tableau. Donc par exemple, si je veux déclarer un tableau qui contient des « double », un tableau de taille fixe 3, que je connais, qui contient des « double », je vais donc écrire à ce moment là : array de double, virgule trois, puis ensuite mettre derrière le nom par exemple ici, « vecteur_3d ». J'aurai au préalable inclus au début de mon programme, la bibliothèque « array ». Avec cette déclaration, on a donc « array » qui est un tableau de taille fixe. Ca ne pourra plus bouger. C'est exactement la taille qui est donnée au départ, qui va être la taille tout au long de l'exécution du programme. Et qui contient de la place pour stocker trois « double ». Un autre exemple, si on voulait déclarer les habitants d'un certain nombre de cantons que l'on connaît au préalable.

notes

résumé

3m 1s

On peut tout à fait déclarer comme ça, une expression constante, « const » ; le nombre de cantons de type « size_t » puisque l'on sait que c'est une taille de tableau que l'on veut indiquer ici. Donc « nb_cantons », le nom de la variable. On peut l'initialiser ici, par exemple à 26. Ce qui nous permet donc ensuite, d'utiliser cette variable, ici, comme le nombre de cantons, dans le tableau dynamique ici, « array » de unsigned_int. Ce sont des entiers positifs. Le nombre d'habitants est un entier positif. Donc on indique le nom ici. Donc « habitant » sera un tableau qui contient, exactement, 26 entiers non signés, 26 « unsigned int ».

notes

résumé

4m 25s

C++11 Initialisation d'un tableau de taille fixe

EPFL

Comme pour les tableaux dynamiques, un tableau de taille fixe peut être initialisé directement lors de sa déclaration :

```
array<type, taille> identificateur({val1, ... , valtaille});
```

ou

```
array<type, taille> identificateur = {val1, ... , valtaille};
```

Exemple :

```
const size_t taille(5);

array<int, taille> ages (
  { 20, 35, 26, 38, 22 });
// ou :
array<int, taille> ages =
  { 20, 35, 26, 38, 22 };
```

Âge
20
35
26
38
22

Un `array` non initialisé contient « n'importe quoi ».

Nous venons de voir comment déclarer un tableau de taille fixe. On peut aussi bien sûr l'initialiser, c'est-à-dire lui donner des valeurs au moment où on le déclare. Avec une syntaxe usuelle, très similaire à celle rencontrée pour les vecteurs. Entre parenthèses rondes ici, la valeur pour un tableau c'est-à-dire un ensemble de valeurs, donc on n'oubliera pas de mettre les accolades pour un ensemble de valeurs qui indiquent les valeurs initiales du tableau. Chacune étant séparée par une virgule. Par exemple, si je veux déclarer un tableau de cinq entiers un tableau statique, un tableau de taille fixe de cinq entiers, je vais déclarer ici « array ». Chacun des éléments sera de type entier. Puis je passe donc la taille qui est connue au moment où j'écris le programme. Donc ici, je l'aurai au préalable déclarée comme une expression constante de type size_t je l'ai donc appelée « taille » et qui contient 5. Je vais donc déclarer un tableau qui contient cinq entiers. Ce tableau je l'appelle « ages ». Puis je l'initialise entre parenthèses rondes avec les valeurs données

notes

résumé

5m 11s

comme un ensemble de valeurs avec les accolades. L'ensemble des valeurs est séparé par des virgules. Donc 20, 35, 26, 38, 22 ; qui va donc me donner le tableau représenté ici. Il existe aussi une seconde façon d'initialiser les tableaux. On l'avait vu avec les tableaux dynamiques, c'est avec le signe égal. Et puis donc de nouveau, de l'autre côté du signe égal, l'ensemble des valeurs entre accolades, séparées par des virgules. Donc si je reprends l'exemple précédent, j'aurais pu aussi, on ne peut pas, évidemment, mettre les deux en même temps, C'est une alternative. Mais j'aurais pu aussi déclarer mon tableau « ages » et l'initialiser ici avec le signe égal puis des accolades. En fonction des compilateurs, d'ailleurs, à l'heure actuelle, certains compilateurs ne tolèrent que cette seconde façon d'initialiser. D'autres tolèrent les deux façons d'initialiser. A noter pour finir qu'un tableau de taille fixe non initialisé contient n'importe quoi, en ce sens qu'on ne sait pas dire ce qu'il contient. qu'on ne sait pas dire ce qu'il contient.

notes

résumé

6m 13s