Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Array (partie 2)

Concepts (extraits des sous-titres générés automatiquement) :

Tableau de taille fixe. Tableaux dynamiques. Tableau dynamique. Tableau statique. Taille du tableau. Tableau statique de taille. Position i d'un tableau statique. Petite spécificité. Seule accolade. Nombre de cantons. Auto element. Pouvoir. Première dimension. Heure actuelle. Tableaux statiques.

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

vers la vidéo

EPFL

Tableaux : array
(Partie 2)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Utilisations d'un tableau de taille fixe

EPFL

L'accès aux éléments d'un tableau de taille fixe se fait de la même façon que pour un tableau dynamique :

- directement : `tab[i]`
- par itération C++11 : `for(auto element : tableau)`
- itération `for` « classique »

Les tableaux de taille fixe `array` ont aussi une « fonction spécifique » `size()` qui renvoie leur taille.

tab.size()

On peut également faire des affectations globales de tableaux de taille fixe `array` :

```
array<int, 3> tab1 = { 1, 2, 3 };
array<int, 3> tab2 ;
...
tab2 = tab1 ; // copie de tab1 dans tab2
```

Maintenant que l'on a déclaré, initialisé un tableau de taille fixe, on peut aussi bien sûr l'utiliser. Il n'y a ici rien de nouveau par rapport aux tableaux dynamiques. Les syntaxes sont vraiment identiques. Pour utiliser un élément à la position i d'un tableau statique qui s'appelle « tab », on utilisera « tab[i] » exactement comme on l'aurait fait pour un tableau dynamique. On peut aussi itérer sur un tableau statique avec exactement la même syntaxe que pour un tableau dynamique, soit le « for » à la C++-2011 : « for(auto element : tableau) » soit avec un « for » classique. Les « array » ont aussi une fonction spécifique « size() » qui indique leur taille. On peut tout à fait, de nouveau, écrire comme pour les tableaux dynamiques tab.size pour obtenir la taille du tableau, tableau statique « tab ». On peut aussi faire des affectations globales. Affecter globalement un tableau tab1 à tab2.

notes

résumé

0m 1s

Si j'ai déclaré un tableau « tab1 » qui est un tableau statique de taille 3 ça ne bougera pas d'entiers que j'initialise aux valeurs 1, 2, 3. Je peux donc tout à fait le recopier directement dans un autre tableau statique de même taille, de taille 3, en écrivant comme ça, « tab2 = tab1 » ce qui va recopier l'intégrité de « tab1 » dans « tab2 ». Mais ça reste bien sûr deux tableaux complètement séparés ça recopie « tab1 » dans « tab2 », Mais « tab1 » et « tab2 » vivent leurs vies séparément. Si je modifie « tab1 », « tab2 » ne sera pas modifié bien sûr. Vous voyez donc que l'utilisation des « array »

notes

résumé

1m 1s

Tableaux à plusieurs dimensions

EPFL

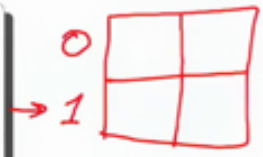
Comme pour les tableaux dynamiques, on peut déclarer des tableaux de taille fixe multidimentionnels.

(on peut même faire des tableaux dynamiques de tableaux de taille fixe, des tableaux de taille fixe de tableaux dynamiques, etc.)

Exemples :

```
array<array<double, 2>, 2> rotation;
array<array<int, nb_statistiques>, nb_cantons> statistiques;
array<array<array<double, 4>, 2, 3> tenseur;

rotation[1][0] = 0.231;
```



`statistiques[i]` est un « `array<int, nb_statistiques>` », c'est-à-dire un tableau de `nb_statistiques` entiers

☞ `statistiques` est bien un tableau de tableaux.

est très similaire à celle des « vector » La différence, c'est simplement que la taille d'un « array » est fixée au départ et ne peux pas bouger. De même, comme on pouvait faire des « vector » de « vector » on va pouvoir faire des « array » de « array » : des tableaux statiques de tableaux statiques. Et puis, si l'on a besoin, on peut bien sûr combiner, et faire des tableaux statiques de tableaux dynamiques des tableaux dynamiques de tableaux statiques, etc. Regardons ici les tableaux statiques de tableaux statiques, array de array, donc comme on faisait des vector de vector, on va pouvoir faire des « array » de « array ». La différence c'est qu'on va devoir indiquer les tailles. Donc par exemple je vais faire un « array » de 2. De deux quoi ? De deux « array » de deux « double ». Donc cette matrice, que j'appelle ici « rotation », c'est bien une matrice qu'on a l'habitude de représenter comme un tableau 2 * 2, un tableau à quatre éléments. Qui est bien un array de deux array de deux double. Voilà donc la déclaration d'un tableau statique à 2 * 2 éléments. On peut donc bien sûr faire d'autres exemples. On peut faire un tableau qui collectionnerait les statistiques sur les cantons. On aura donc un tableau à nombre de cantons, de première dimension qui est le nombre de cantons et puis chacun des cantons aura lui même un certain nombre de statistiques, par exemple deux statistiques comme la surface et le nombre d'habitants. On peut même faire des objets encore plus compliqués comme des tableaux à trois indices : des tableaux de tableaux de tableaux. Ce qui mathématiquement correspondrait à un tenseur. Ce

notes

résumé

1m 44s

Tableaux à plusieurs dimensions

EPFL

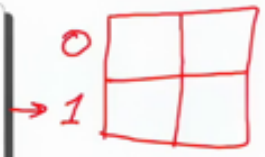
Comme pour les tableaux dynamiques, on peut déclarer des tableaux de taille fixe multidimentionnels.

(on peut même faire des tableaux dynamiques de tableaux de taille fixe, des tableaux de taille fixe de tableaux dynamiques, etc.)

Exemples :

```
array<array<double, 2>, 2> rotation;
array<array<int, nb_statistiques>, nb_cantons> statistiques;
array<array<array<double, 4>, 2, 3> tenseur;

rotation[1][0] = 0.231;
```



`statistiques[i]` est un « `array<int, nb_statistiques>` », c'est-à-dire un tableau de `nb_statistiques` entiers

☞ `statistiques` est bien un tableau de tableaux.

qui compte ici c'est de bien comprendre les différentes tailles. On aura un tableau de 3, de 3 tableaux de 2, tableaux de 2, tableaux de quatre éléments. Cette couche de 2 * 4 éléments étant reproduite trois fois dans le tableau initial. Ceci est certainement un petit peu compliqué et peu courant, mais le but était ici de montrer que l'on peut généraliser à n'importe quel nombre de tableau et surtout d'indiquer l'ordre des différentes tailles. Voyons maintenant l'utilisation de ces tableaux à plusieurs dimensions. Revenons à ce que l'on avait appelé la matrice « rotation » qui est un tableau de 2 fois 2 « double », que l'on représente comme ceci. L'utilisation est exactement comme pour les « vector ». On va utiliser les crochets avec deux dimensions. On va regarder « rotation » de 1, la numérotation, je vous le rappelle, commence à 0 va jusqu'à taille -1. Donc ici, rotation de 1.

notes

résumé

Dans rotation de 1, on va chercher l'élément à la position 0. Cet élément ici, auquel on donnera la valeur 0.231. C'est bien l'élément rotation de 1, crochet 0 qui nous intéresse. De même, si je prends le tableau statistique, ici, statistique de i est donc un « array » de « nb_statistiques » « int » C'est donc un « array » de « nb_statistiques » « int ». C'est-à-dire, lui-même un tableau de « nb_statistiques ».

notes

résumé

4m 37s

Tableaux à plusieurs dimensions (2)

Les tableaux multidimensionnels peuvent également être initialisés lors de leur déclaration.
Il faut bien sûr spécifier autant de valeurs que le produit des dimensions.

Exemple :

```
array<array<int, 3>, 4>
matrice = {
        0, 1, 2 ,
        3, 4, 5 ,
        6, 7, 8 ,
        9, 0, 1
};
```

0	1	2
3	4	5
6	7	8
9	0	1

Tout ceci est exactement comme on procédait avec les tableaux dynamiques de tableaux dynamiques. Rien de nouveau ici, si ce n'est que toutes les tailles sont fixées et donc toutes les lignes ont la même longueur. Une petite spécificité, cependant, par rapport aux « vector » qui concerne l'initialisation des « array » de « array ». La syntaxe actuellement supportée par les compilateurs ne tolèrent qu'une seule accolade dans les initialisations.

notes

résumé

5m 12s

Pour résumer

EPFL

Tableaux dynamiques	Tableaux statiques
`#include <vector>`	`#include <array>`
`vector<double> tab;`	`array<double, 5> tab;`
`vector<double> tab2(5);`	
`tab[i][j]`	
`tab.size()`	
`for(auto element : tab)`	
`for(auto& element : tab)`	
`tab.push_back(x);`	—
`tab.pop_back();`	—
`vector<vector<int>> tableau( { { 0, 1, 2, 3, 42 }, { 4, 5, 6 }, { 7, 8 }, { 9, 0, 1 } } );`	`array<array<int, 3>, 4> matrice = { 0, 1, 2 , 3, 4, 5 , 6, 7, 8 , 9, 0, 1 };`

Donc si je fais un tableau statique de tableau statique, par exemple, à 4 fois 3 entiers. Donc ici je mets des entiers, que je vais initialiser. Contrairement aux tableaux dynamiques, il va falloir initialiser avec simplement une seule accolade et mettre à l'intérieur l'ensemble des valeurs que l'on veut déclarer, dans l'ordre de lecture. Ici, par exemple, 0, 1, 2 et puis ensuite 3, 4 , 5, séparés simplement par des virgules. Voilà à l'heure actuelle, la syntaxe utilisée pour initialiser les tableaux statiques de tableaux statiques. Pour conclure, résumons les différentes syntaxes entre les tableaux statiques et les tableaux dynamiques, entre les « vector » et les « array ». Les tableaux dynamiques dont la taille peut varier au fur et à mesure que le programme s'exécute. Par opposition aux tableaux statiques dont la taille doit être connue au moment où l'on écrit le programme. Les premiers tableaux dynamiques, sont des « vector » et les tableaux statiques sont des « array ». On déclarera un tableau dynamique soit vide, on peut tout à fait déclarer un tableau dynamique vide.

notes

résumé

5m 37s

Pour résumer

EPFL

Tableaux dynamiques	Tableaux statiques
`#include <vector>`	`#include <array>`
`vector<double> tab;`	`array<double, 5> tab;`
`vector<double> tab2(5);`	
`tab[i][j]`	
`tab.size()`	
`for(auto element : tab)`	
`for(auto& element : tab)`	
`tab.push_back(x);`	—
`tab.pop_back();`	—
`vector<vector<int>> tableau( { { 0, 1, 2, 3, 42 }, { 4, 5, 6 }, { 7, 8 }, { 9, 0, 1 } } );`	`array<array<int, 3>, 4> matrice = { 0, 1, 2 , 3, 4, 5 , 6, 7, 8 , 9, 0, 1 };`

Soit lui donner une valeur initiale, une taille initiale : 5. Mais cette taille va pouvoir varier au fur et à mesure du programme. On pourra rajouter ou enlever des éléments. Alors que du côté statique, la taille doit absolument être connue. On peut soit l'écrire comme ceci, soit utiliser une variable. Mais cette taille doit absolument être connue. Donc ici, on n'a qu'une seule syntaxe pour les tableaux statiques. Ensuite pour utiliser les éléments d'un tableau à une ou deux dimensions, c'est la même syntaxe. A une dimension : tab[i] Si le tableau a deux dimensions : tab[i][j]. Pour connaître la taille d'un tableau statique ou d'un tableau dynamique, on utilise la fonction « size() ». Pour parcourir un tableau statique ou dynamique, ça se fait de la même façon, on peut soit utiliser une boucle for classique, soit utiliser une boucle for C++-2011 donc avec les deux points. Si on veut parcourir sans modifier les éléments, on écrira « for(auto » et puis le nom que l'on veut utiliser pour les éléments. Si on veut modifier les éléments on n'oubliera pas de rajouter le signe &. Et ça de façon identique dans un tableau statique ou dans un tableau dynamique. Ensuite dans un tableau dynamique, on va pouvoir éventuellement rajouter des éléments. C'est spécifique au tableau dynamique,

notes

résumé

6m 49s

ça n'existe pas pour les tableaux statiques. On ne peut pas modifier un tableau statique. Donc pour un tableau dynamique uniquement : « push_back » on rajoute la valeur que l'on a donnée « x » à la fin du tableau. Donc ça va recopier « x » à la fin du tableau. Si l'on veut enlever le dernier élément d'un tableau dynamique, alors on utilisera « pop_back » et ceci, bien sûr, n'existe pas pour les tableaux statiques qu'on ne peut pas modifier. Enfin dernière différence, pour l'initialisation des tableaux dynamiques de tableaux dynamiques, on pourra utiliser la syntaxe, supportée par tous les compilateurs, d'initialisation avec un tableau de tableaux avec les parenthèses rondes. Alors que pour les tableaux statiques de tableaux statiques il faudra utiliser la syntaxe « = » à l'heure actuelle et utiliser uniquement un seul niveau d'accolade avec la liste des valeurs désirées, comme ça, les unes derrière les autres. les unes derrière les autres.

notes

résumé

8m 1s