Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Tableaux dynamiques multidimensionnels

Concepts (extraits des sous-titres générés automatiquement) :

Tableaux dynamiques. Tableaux dynamiques de valeurs. Élément du tableau. Tableaux dynamiques d'entiers. Tableau dynamique. Deuxième tableau. Petite aparté. Troisième tableau. Tableau dynamique d'entiers. Élément de ce tableau dynamique. Séquence vidéo précédente. Tableau de notes. Tableau des notes d'un élève. Valeur initiale. Différentes notes.

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

vers la vidéo

EPFL

Tableaux : tableaux dynamiques multidimensionnels

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Dans une séquence vidéo précédente,

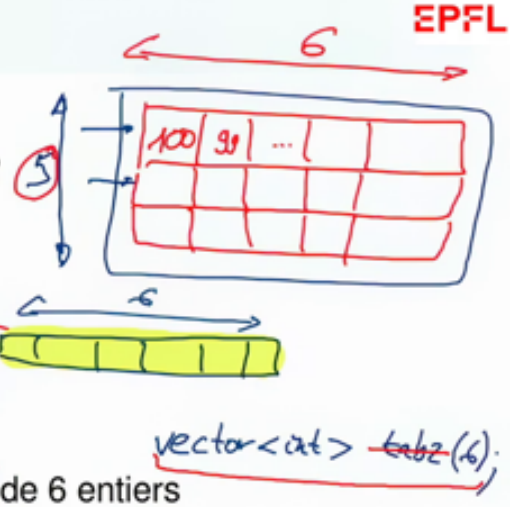
notes

résumé

0m 1s

Les tableaux multidimensionnels

EPFL



Comment déclarer un tableau à plusieurs dimensions ?

☞ On ajoute simplement un niveau de plus :

C'est en fait un tableau de tableaux...

Exemple :

```
vector<vector<int>> tab(5, vector<int>(6));
```

correspond à la déclaration d'un tableau de 5 tableaux de 6 entiers

`tab[i]` est donc un « `vector<int>` », c'est-à-dire un tableau dynamique d'entiers (qui, au départ, en contient 6)

`tab[i][j]` sera alors le $(j+1)$-ième élément de ce tableau.

nous avons vu les tableaux dynamiques à une dimension, c'est-à-dire des tableaux dynamiques de valeurs, par exemple, les différents âges des étudiants de ce cours. Comment faire si l'on veut représenter les tableaux à plusieurs dimensions ? Comme par exemple, le tableau de toutes les notes de tous les étudiants à tous les devoirs. Ce que je pourrais faire, c'est faire un tableau pour un étudiant qui contiendrait les différentes notes aux différents devoirs, puis faire un deuxième tableau pour un autre étudiant, qui contiendrait les notes aussi à ses devoirs, etc... faire un troisième tableau, et on aurait donc ainsi, globalement un tableau... de tableaux. Chaque élément du tableau étant lui-même un tableau de notes. Supposons donc par exemple que l'on veuille faire un tableau pour 5 étudiants et que par exemple, chacun des étudiants ait eu à remplir 6 épreuves et donc six notes au final. On aura donc un tableau dynamique de 5 tableaux de 6 notes. Donc, on va déclarer un tableau dynamique qu'on va appeler tab. Ce tableau dynamique contiendra lui-même des tableaux dynamiques d'entiers. Les entiers, ce sont les notes. Le tableau dynamique d'entiers, c'est le tableau des notes d'un élève, et le tableau dynamique des tableaux dynamiques d'entiers c'est le tableau des tableaux de tous les élèves. On va l'initialiser au nombre d'élèves donc ici on a 5 élèves, et chacun de ces 5 éléments est lui-même quoi ? Lui-même, un tableau dynamique d'entiers, pour représenter les notes, qui contient 6 notes. On a donc au final un tableau dynamique qui contient 5 tableaux dynamiques de 6 notes. La syntaxe utilisée ici pour donner une valeur initiale à chacun des tableaux de notes de chacun des élèves à chacun des sous-tableaux, est la même que nous avions déjà rencontrée lors de la copie d'un tableau précédemment. C'est un tableau anonyme. C'est comme si vous écriviez

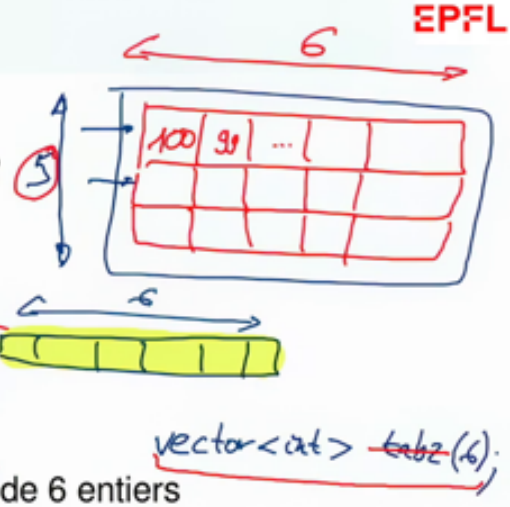
notes

résumé

0m 5s

Les tableaux multidimensionnels

EPFL



Comment déclarer un tableau à plusieurs dimensions ?

☞ On ajoute simplement un niveau de plus :

C'est en fait un **tableau de tableaux**...

Exemple :

```
vector<vector<int>> tab(5, vector<int>(6));
```


correspond à la déclaration d'un tableau de 5 tableaux de 6 entiers

`tab[i]` est donc un « `vector<int>` », c'est-à-dire un tableau dynamique d'entiers (qui, au départ, en contient 6)

`tab[i][j]` sera alors le $(j+1)$-ième élément de ce tableau.

vector tab2 initialisé avec 6 int au départ tous à la valeur nulle, et que vous ayez supprimé le nom. Vous vous retrouvez donc avec un tableau anonyme. Ici, on n'a pas donné de nom, tableau anonyme de 6 entiers tous nuls. Si je regarde maintenant le tableau dynamique de tableaux dynamiques d'entiers, que l'on a déclaré ici, qu'on a appelé tab, tab [i] (tab de i) est donc un élément de ce tableau dynamique qui correspond à un tableau de notes. tab [i] est donc lui-même un tableau dynamique d'entiers,

notes

résumé

Les tableaux multidimensionnels (2)

On a l'habitude de se représenter `tab[i][j]` comme l'élément de la `(i+1)ème` ligne et de la `(j+1)ème` colonne.

Mais attention ! Un `vector<vector<int>>` n'est pas une matrice, mais un tableau dynamique de tableaux dynamiques d'entiers (pas nécessairement tous de la même taille !).

0	1	2	3	42
4	5	6		
7	8			
9	0	1		

qui correspond bien au type que l'on a donné ici, tableau dynamique d'entiers, et à la valeur initiale qu'on a donnée ici. tab [i], tab [0] par exemple, c'est ce tableau ici de 6 entiers. tab [i] [j], c'est l'élément à la position j c'est-à-dire le (j+1)e élément du tableau tab[i]. Donc si je prends le tableau, ici, tableau dynamique de tableaux dynamiques d'entiers, que je prends tab [i] tab[i] est lui-même un tableau dynamique d'entiers, dont je cherche le (j+1)e élément, l'élément à la position j. On a donc l'habitude de représenter tab [i] [j] comme étant l'élément à la (i+1)e ligne et (j+1e) colonne. Mais il ne faut pas oublier qu'on a ici un tableau dynamique de tableaux dynamiques d'entiers. L'aspect tableau dynamique des lignes est toujours présent

notes

résumé

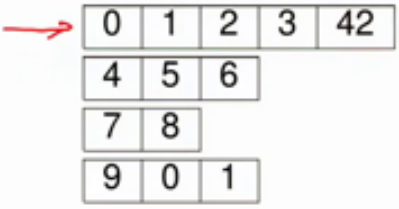
3m 13s

Code de l'exemple (version C++11)

```
vector<vector<int>> tableau(
  { { 0, 1, 2, 3, 42 },
    { 4, 5, 6        },
    { 7, 8           },
    { 9, 0, 1        } }
);

for(auto ligne : tableau) {
  for(auto element : ligne) {
    cout << element << " ";
  }
  cout << endl;
}

for(size_t i(0); i < tableau.size(); ++i) {
  cout << "tableau[" << i <<"].size()="
       << tableau[i].size() << endl;
}
```



ça veut dire que chacune des lignes peut elle-même avoir des éléments qui se rajoutent ou qui s'enlèvent, et donc au final, le tableau dynamique de tableaux dynamiques n'a pas nécessairement des lignes de la même taille. Par exemple, on peut très bien avoir une première, ici, taille pour tab[0] qui est bien un tableau dynamique d'entiers, une taille différente pour tab[1] qui est un deuxième tableau dynamique d'entiers et ainsi de suite. Regardons maintenant cet exemple en détails. Nous allons donc déclarer un tableau dynamique de tableaux dynamiques d'entiers que l'on appelle, par exemple, tableau. Si on veut l'initialiser à la valeur de l'exemple que j'ai redessiné ici, ce qu'on va faire, c'est qu'on va donc donner une valeur initiale sous forme de liste d'éléments, chacun des éléments étant lui-même un tableau dynamique d'entiers. On retrouve ici la première ligne.

notes

résumé

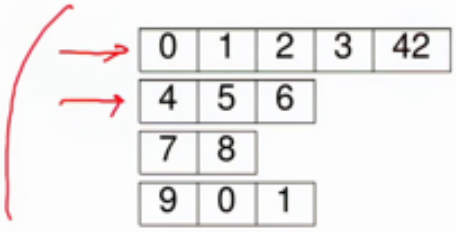
4m 25s

Code de l'exemple (version C++11)

```
vector<vector<int>> tableau(
  { { 0, 1, 2, 3, 42 },
    { 4, 5, 6        },
    { 7, 8           },
    { 9, 0, 1        } }
);

for(auto ligne : tableau) {
  for(auto element : ligne) {
    cout << element << " ";
  }
  cout << endl;
}

for(size_t i(0); i < tableau.size(); ++i) {
  cout << "tableau[" << i <<"].size()="
       << tableau[i].size() << endl;
}
```



Ensuite, on a la deuxième ligne et ainsi de suite pour la troisième et la quatrième ligne de notre tableau dynamique de tableaux dynamiques d'entiers. Cette initialisation correspond exactement au dessin qui est donné ici à droite. Une petite aparté concernant la syntaxe : Quand on déclare un tableau dynamique de tableaux dynamiques d'entiers, on a ici besoin de coller la fermeture des deux types sous forme de deux signes supérieurs qui sont collés comme ceci. Ceci n'est toléré qu'en C++ 2011. Si vous avez à écrire dans un compilateur qui n'est pas en C++ 2011 alors il faudra mettre un espace entre ces deux signes-là. La syntaxe avec les deux signes supérieurs collés comme ceci, ici, n'étant pas supportée par les anciens compilateurs. Regardons maintenant comment on peut parcourir ce tableau. Pour cela, on va utiliser une boucle à la C++ 2011 avec donc les deux points, le tableau que l'on veut parcourir, ici qu'est tableau, et la déclaration de la variable qui va parcourir les lignes. On va l'appeler ici naturellement ligne, ce qui va nous permettre de parcourir ligne à ligne. Puis ensuite dans le parcours d'une ligne,

notes

résumé

5m 25s

Code de l'exemple (version C++11)

```
vector<vector<int>> tableau(
  { { 0, 1, 2, 3, 42 },
    { 4, 5, 6        },
    { 7, 8           },
    { 9, 0, 1        } }
);

for(auto ligne : tableau) {
  for(auto element : ligne) {
    cout << element << " ";
  }
  cout << endl;
}

for(size_t i(0); i < tableau.size(); ++i) {
  cout << "tableau[" << i <<"].size()="
       << tableau[i].size() << endl;
}
```

0	1	2	3	42
4	5	6		
7	8			
9	0	1		

on va parcourir tous les éléments d'une ligne donnée. Par exemple, si je suis sur la première ligne, je vais ensuite parcourir les différents éléments de cette ligne-là dans une deuxième boucle for qui me parcourt cette fois-ci le tableau ligne que j'ai déclaré ici, qui est une ligne donnée, et je vais le parcourir en donnant comme nom le nom element donc element va parcourir les différents éléments d'une ligne. C'est ce que nous avons écrit ici et on peut donc utiliser cette variable element pour parcourir les différents éléments d'une ligne. Pour finir, illustrons le fait que les éléments du tableau sont bien des tableaux dynamiques en donnant la taille des différentes lignes, des différents tableaux dynamiques. Pour cela, on va utiliser une boucle for classique. On va démarrer i comme d'habitude

notes

résumé

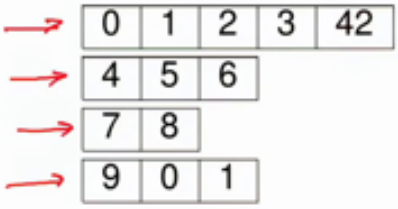
6m 37s

Code de l'exemple (version C++11)

```
vector<vector<int>> tableau(
  { { 0, 1, 2, 3, 42 },
    { 4, 5, 6        },
    { 7, 8           },
    { 9, 0, 1        } }
);

for(auto ligne : tableau) {
  for(auto element : ligne) {
    cout << element << " ";
  }
  cout << endl;
}

for(size_t i(0); i < tableau.size(); ++i) {
  cout << "tableau[" << i <<"].size()="
       << tableau[i].size() << endl;
}
```



déclaré comme size_t à la valeur 0.

notes

résumé

7m 37s

Tant que i est plus petit que la taille du tableau, le tableau dont on parle, le tableau, ici, de départ, et toujours pareil, i qui passe de 1 en 1. Ce qu'on va faire, c'est qu'on va afficher "tableau[i]" donc ça va afficher tableau de 0, tableau de 1, etc...]. size. Et puis, ensuite on peut utiliser tel quel tableau[i] qui est un tableau, .size pour renvoyer la taille du tableau. Ce qui va donc nous afficher 5 pour tab [0]. size(), 3 pour tab [1] . size(), 2 pour tab [2] . size(), et enfin, à nouveau 3 pour tab [3] . size() pour tab [3] . size()

notes

résumé

7m 43s