Algorithmique et structures de données

Tableaux à deux dimensions

Julien BERNARD

Université de Franche-Comté – UFR Sciences et Technique

Licence Informatique – 2e année

Plan

Tableaux à deux dimensions

Tableau à deux dimensions

Un **tableau à deux dimensions** (appelé aussi matrice) est un tableau qui contient des tableaux. L'accès aux éléments se fait à l'aide de deux indices : un indice pour le premier tableau qui renvoie vers un second tableau, et un indice pour le second tableau qui renvoie vers l'élément.

Indices

Les deux indices des tableaux sont appelés **ligne** et **colonne**. On notera R le nombre de lignes *(rows)* et C le nombre de colonnes *(columns)*.

Implémentation

Il existe trois manières en C de représenter un tableau à deux dimensions :

- Tableau statique
- Tableau de tableaux
- Tableau linéaire

Plan

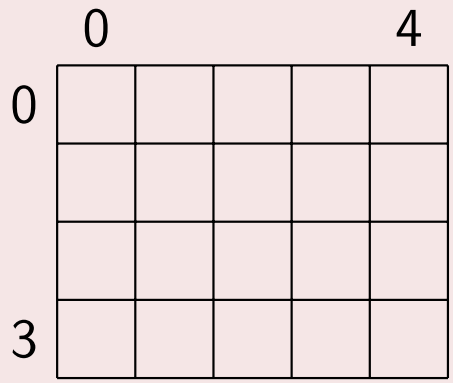
Tableaux à deux dimensions

Tableau statique

Tableau statique

- On déclare un tableau statique de tableau statique
- Uniquement si R et C sont des *constantes*
- Avantages :
  - accès direct avec les indices r et c
  - mémoire contiguë
- Inconvénient : aucun ?

Représentation



Tableaux à deux dimensions

Tableau statique

Déclaration

```
#define R 4
#define C 5
struct array2d {
  int data[R][C];
};
```

Accès

```
int array2d_access(struct array2d *self, size_t r, size_t c) {
  assert(r < R);
  assert(c < C);
  return self->data[r][c];
}
```

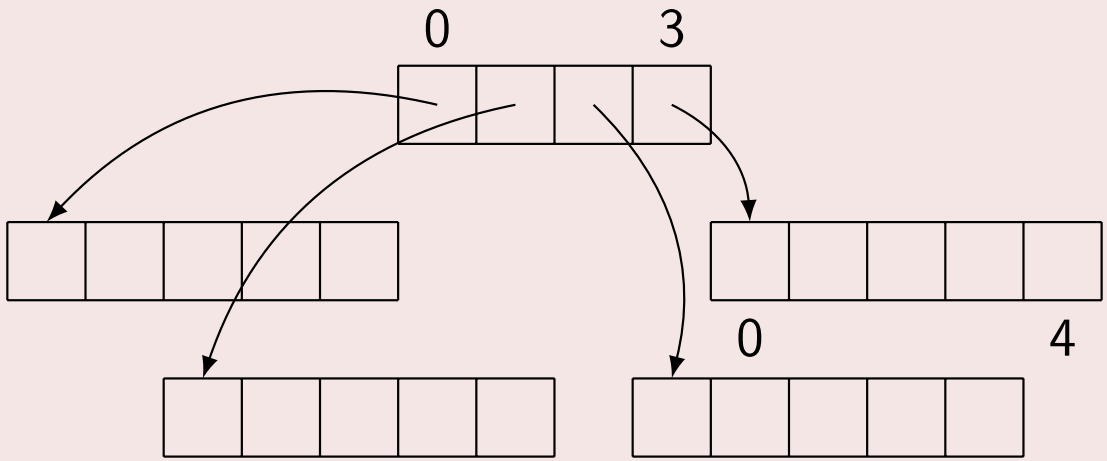
Tableaux à deux dimensions

Tableau de tableaux

Tableau de tableaux

- On déclare un pointeur de pointeur `int**`
- Avantage : accès direct avec les indices r et c
- Inconvénients :
  - beaucoup d'allocations et de libérations de mémoire
  - mémoire non-contiguë

Représentation



Tableaux à deux dimensions

Tableau de tableaux

Déclaration

```
struct array2d {
  size_t rows;
  size_t cols;
  int **data;
};
```

Accès

```
int array2d_access(struct array2d *self, size_t r, size_t c) {
  assert(r < self->rows);
  assert(c < self->cols);
  return self->data[r][c];
}
```

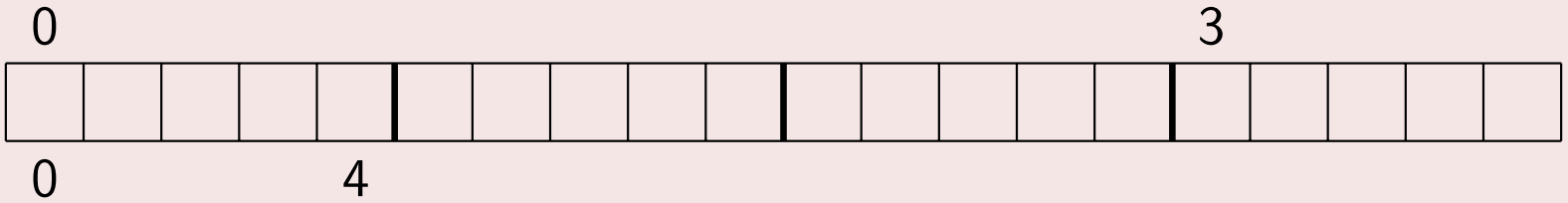
Tableaux à deux dimensions

Tableau linéaire

Tableau linéaire

- On déclare un pointeur `int*`, comme si on avait mis les lignes bout à bout
- Avantage : mémoire contiguë
- Inconvénients : pas d'accès direct avec les indices r et c

Représentation



Tableaux à deux dimensions

Tableau linéaire

Déclaration

```
struct array2d {
  size_t rows;
  size_t cols;
  int *data; // size = rows * cols
}
```

Accès

```
int array2d_access(struct array2d *self, size_t r, size_t c) {
  assert(r < self->rows);
  assert(c < self->cols);
  return self->data[r * self->cols + c];
}
```

Comment choisir ?

Comment choisir ?

- Si les dimensions sont constantes $\rightarrow$ tableau statique
- Sinon $\rightarrow$ tableau linéaire
- On ne choisit jamais le tableau de tableaux

Plan

Ordre de stockage

Ordre de stockage

L'**ordre de stockage** est la méthode pour ordonner les éléments dans un tableau à deux dimensions. Il existe deux ordres de stockage :

- *row-major* : on stocke le tableau ligne par ligne
- *column-major* : on stocke le tableau colonne par colonne

Remarque

Dans tous les exemples précédents, on a utilisé l'ordre *row-major*.

Comment choisir ?

- Parfois, le choix est imposé. Exemple : OpenGL $\rightarrow$ *column-major*
- Sinon, on choisit ce qu'on veut mais on fait attention au parcours du tableau

C'est tout pour le moment...

Des questions ?