Faculté d'ingénierie et de technologie – Génie informatique

Algorithmique et programmation

Damien Berthet & Vincent Labatut

TP 07 | tableaux multidimensionnels

Présentation

Le but de ce TP est de manipuler des tableaux multidimensionnels d'entiers. On étudie d'abord leur stockage en mémoire, puis on effectue des calculs simples sur des tableaux à 2 dimensions.

1 Occupation mémoire

Exercice 1

Soit un tableau `short m[N][P][Q]` pour $N = 4$, $P = 3$ et $Q = 2$. Dessinez la représentation graphique de son occupation de la mémoire.

Exercice 2

Donnez les formules permettant de calculer les adresses de `m[i]`, `m[i][j]`, et `m[i][j][k]` en fonction de :

- Adresse du tableau : `m` ;
- Type des données qu'il contient : `short` ;
- Index : `i`, `j` et `k`.

Écrivez un programme permettant de vérifier votre calcul : il demande à l'utilisateur de saisir les valeurs `i`, `j` et `k` et affiche chaque adresse de deux façons différentes :

- Adresses directement obtenues en appliquant l'opérateur d'adressage `&` aux 3 expressions mentionnées au début de l'exercice ;
- Adresses calculées avec vos 3 formules.

Bien sûr, si vos formules sont correctes, vous devez obtenir les mêmes adresses avec les deux méthodes.

exemple : (les `??????` représentent des adresses connues seulement à l'exécution)

```
Entrez i,j,k (avec les virgules) : 1,2,0
Adresse du tableau m : 0x??????
Adresse réelle de   m[1] : 0x??????
Adresse calculée de m[1] : 0x??????
Adresse réelle de   m[1][2] : 0x??????
Adresse calculée de m[1][2] : 0x??????
Adresse réelle de   m[1][2][0] : 0x??????
Adresse calculée de m[1][2][0] : 0x??????
```

Remarques :

- `N`, `P` et `Q` doivent être déclarées comme des constantes.
- La fonction `sizeof(t)` permet de déterminer combien d'octets un type `t` occupe en mémoire. Par exemple, `sizeof(char)` renvoie la valeur 1.

Exercice 3

Écrivez un programme qui initialise ce tableau de manière à ce que l'élément situé à l'adresse `m+2x` contienne la valeur `x`. Effectuez une vérification en utilisant vos formules de l'exercice précédent, qui vous permettent de calculer `2x` en fonction de `i`, `j`, et `k`.

exemple :

```
Entrez i,j,k (avec les virgules) : 1,2,0
     Valeur theorique : xxxxx
     Valeur effective  : xxxxx
```

2 Opérations matricielles

Pour des raisons pratiques, on se concentre ici sur des matrices carrées de dimension $N \times N$.

Exercice 4

Écrivez un programme permettant d'afficher une matrice `int m[N][N]` à l'écran, comme ci-dessous.

exemple : affichage de la matrice identité

```
  1  0  0  0
  0  1  0  0
  0  0  1  0
  0  0  0  1
```

Remarque : on suppose la matrice contient uniquement des valeurs comprises entre 0 et 99.

Exercice 5

Écrivez un programme qui place dans une matrice `res[N][N]` le résultat de la multiplication d'une matrice `m[N][N]` par un scalaire `int s`. Vous devez initialiser `m` en utilisant la méthode que vous voulez, puis calculer le produit et afficher son résultat `res`. Utilisez (en l'adaptant) le code source de l'exercice précédent pour effectuer l'affichage du résultat.

Exercice 6

Écrivez un programme qui calcule la transposée d'une `m[N][N]` et place le résultat dans la matrice `res[N][N]`. Initialisez et affichez le résultat comme dans l'exercice précédent.

Exercice 7

Écrivez un programme qui calcule la somme de deux matrices `m1[N][N]` et `m2[N][N]` et place le résultat dans la matrice `res[N][N]`. Initialisez et affichez le résultat comme dans les exercices précédents.

Exercice 8

Même chose avec le produit de deux matrices `m1[N][N]` et `m2[N][N]`, dont le résultat est à placer dans une troisième matrice `res[N][N]`.