

MOOC Init. Prog. C++

Exercices semaine 5

Exercice 15 : Échauffement avec les tableaux dynamiques (niveau 1)

Cet exercice correspond à l'exercice n°17 (page 55 et 218) de l'ouvrage [C++ par la pratique \(3^e édition, PPUR\)](#).

Rappel : Pour utiliser le type **vector**, il faut inclure la librairie définissant ce type, au moyen de la directive :

```
#include <vector>
```

En vous aidant si nécessaire d'un programme, répondez aux questions suivantes :

A : Quelles valeurs contient le tableau `tab` après l'exécution des lignes suivantes ? Expliquez.

```
int const taille(10);
vector<int> tab;
for (size_t i(0); i < taille; ++i) {
    tab.push_back(tab.size());
}
```

B : Que fait la fonction `f` suivante ?

```
void f(vector<int>& tab, vector<int>& tab2)
{
    for (int i(0); i < tab.size(); ++i) {
        tab2.push_back(tab[i]);
    }
}
```

Exercice 16 : produit scalaire (tableaux dynamiques, niveau 1)

Cet exercice correspond à l'exercice n°18 (pages 55 et 218) de l'ouvrage [C++ par la pratique \(3^e édition, PPUR\)](#).

Écrivez un programme `scalaire.cc` qui calcule le produit scalaire de deux vecteurs, **implémenté au moyen de tableaux dynamiques**.

Votre programme devra utiliser (entre autres) les éléments suivants :

- deux variables de type tableau dynamique de réels ;
- une fonction qui calcule le produit scalaire : `double scalaire(vector<double> u, vector<double> v) ;`

Méthode :

- demander à l'utilisateur d'entrer n , la taille effective des vecteurs.
- demander à l'utilisateur les composantes ($v1_0 \dots v1_{n-1}$, $v2_0 \dots v2_{n-1}$) des vecteurs $v1$ et $v2$.
- appeler la fonction `scalaire(...)` pour calculer le produit scalaire de $v1$ et $v2$.
- afficher le résultat.

Rappel :

Le produit scalaire de a par b est: $a.b = a_1*b_1 + a_2*b_2 + \dots + a_n*b_n$

Exemple: $a = (5, 3, -1)$ $b = (2, 1, 2)$ $a.b = 11$

Exercice 17 : Multiplication de matrices (tableaux dynamiques, niveau 2)

Cet exercice correspond à l'exercice n°20 (pages 57 et 222) de l'ouvrage [C++ par la pratique \(3^e édition, PPUR\)](#).

On cherche ici à écrire un programme `mulmat.cc` qui calcule la multiplication de deux matrices (rappel ci-dessous).

Vous utiliserez pour représenter la matrice un *tableau dynamique de tableaux dynamiques de doubles*.

Déclarations :

- dans `main()`, déclarez deux matrices `M1` et `M2`.

Fonctions :

- la fonction de prototype

```
vector<vector<double>> lire_matrice();
```

qui lit depuis le clavier les éléments d'une matrice (après avoir demandé sa taille) et retourne la matrice résultante.

- la fonction de prototype

```
vector<vector<double>> multiplication(const vector<vector<double>>& Mat1,  
                                   const vector<vector<double>>& Mat2);
```

qui multiplie deux matrices de tailles et renvoie le résultat.

- la fonction de prototype

```
void affiche_matrice(const vector<vector<double>>& M);
```

qui affiche le contenu d'une matrice ligne par ligne.

Méthode :

- lire depuis le clavier les dimensions **l1** (nombre de lignes) et **c1** (nombre de colonnes) de la première matrice `M1`
- lire le contenu de `M1`.
- De même, lire les dimensions puis le contenu de la seconde matrice `M2`.
- Vérifier que le nombre de lignes de `M2` est identique au nombre de colonnes de `M1`.
Dans le cas contraire, afficher un message d'erreur « Multiplication de matrices impossible ! ».
- Effectuer la multiplication des matrices : $M = M1 * M2$:
 - Les dimensions de `M` sont : `l1` (nombre de lignes) et `c2` (nombre de colonnes).
 - l'élément M_{ij} est défini par
$$M_{ij} = \sum_k M1_{ik} \times M2_{kj}.$$
- afficher le résultat ligne par ligne.

Exemple d'utilisation :

```
Saisie d'une matrice :  
Nombre de lignes : 2  
Nombre de colonnes : 3  
M[1,1]=1  
M[1,2]=2  
M[1,3]=3  
M[2,1]=4  
M[2,2]=5  
M[2,3]=6  
Saisie d'une matrice :  
Nombre de lignes : 3  
Nombre de colonnes : 4  
M[1,1]=1  
M[1,2]=2  
M[1,3]=3
```

```
M[1,4]=4
M[2,1]=5
M[2,2]=6
M[2,3]=7
M[2,4]=8
M[3,1]=9
M[3,2]=0
M[3,3]=1
M[3,4]=2
Résultat :
38 14 20 26
83 38 53 68
```
