TD-6
itérations imbriquées et matrices

Exercice 1 :

Ecrire un programme qui demande deux entiers l et L, et qui dessine avec des $*$ un rectangle de largeur l et de longueur L

Exercice 2 :

Ecrire un programme qui defini et rempli une matrice 5x5 avec des chiffres pris au hasard.

Exercice 3 :

Écrire un programme qui calcule les sommes des éléments de chaques lignes d'une matrice avec un affichage correspondant à l'exemple :

```
4 3 1 2 2        12
4 6 6 5 5        26
2 1 1 2 8        14
9 1 6 6 6        28
4 1 9 9 9        32
```

Faire la même chose pour les colonnes. Exemple :

```
 4  3  1  2  2
 4  6  6  5  5
 2  1  1  2  8
 9  1  6  6  6
 4  1  9  9  9

23 12 23 24 30
```

Exercice 4 :

Écrire un programme qui défini une matrice g obtenue en tournant une matrice a de 90 degrés à gauche. Exemple :

```
7 5 5 9 8                    8 4 2 6 2
9 5 4 4 4                    9 4 4 2 1
9 9 8 4 2      devient       5 4 8 2 3
1 2 2 2 6                    5 5 9 2 1
1 1 3 1 2                    7 9 9 1 1
```

Donner un algorithme qui tourne la matrice de 90 degrés à droite.

Exercice 5 :

On cherche à calculer les 5 premieres lignes du triangle de Pascal.
Une première solution consiste à utiliser une matrice 5x5 qui contient tout le triangle. La décrire.

Une seconde solution consiste à modéliser la ligne pour laquelle on calcule les coéfficients par un vecteur. Donner une solution qui n'utilise qu'un seul vecteur.

Notes et corrigé pour le TD-6

Notes Notions à introduire dans ce TD : Les tableaux de tableaux.

Corrigé Exercice 1 :

```
"programme" rectangle
  "variables"
     i,j,l,L : entiers positifs
  "debut" affiche1
     afficher ‘‘donner la largeur’’
     lire l

     afficher ‘‘donner la longueur’’
     lire L

     "Pour" i <- 1 "à" L "faire"
        "Pour" j <- 1 "à" l "faire"
            "afficher" '*'
        "fin pour"
        "afficher" '\n'
     "fin pour"
"fin" rectangle
```

Corrigé Exercice 2 :

en pseudo-langage

```
"programme" matriceRandom
"constantes"
    dim = 5
"types"
    vecteur = "tableau de" Dim entiers
    matr    = "tableau de" Dim vecteurs
"variables"
    a : matrice
    i,j : entier positifs
"debut" matrice
   // remplissage
   "pour" i <-- 1 "a" dim "faire"
      "pour" j <-- 1 "a" dim "faire"
         a[i][j] <-- random(0,9)
      "fin pour"
   "fin pour"

   // affichage
   "pour" i <-- 1 "a" dim "faire"
      "pour" j <-- 1 "a" dim "faire"
         "afficher" a[i][j]
      "fin pour"
   "fin pour"
"fin" matriceRand
```

en C++

```
#include <iostream.h>
#include <stdlib.h>
#include <time.h>

const int dim = 5;

typedef int vecteur[dim];
typedef vecteur matr[dim];

void main () {
  int i,j;
  matr a;

  srand(time(NULL));

  // initialisation
  for (i = 0; i < dim; i++) {
    for (j = 0; j < dim; j++) {
      a[i][j] = rand() % 10;
    }
  }

  // affichage
  for (i = 0; i < dim; i++) {
    for (j = 0; j < dim; j++) {
      cout << a[i][j] << " ";
    }
    cout << "\n";
  }
}
```

Corrigé Exercice 3 (en C++) : La partie essentielle (somme des lignes) :

```
for (i = 0; i < dim; i++) {
  somme = 0;
  for (j = 0; j < dim; j++) {
    cout << a[i][j] << " "; somme = somme + a[i][j];
  }
  cout << "        " << somme << "\n";
}
```

La partie essentielle (somme des colonnes) :

```
for (j = 0; j < dim; j++) {
  somme = 0;
```

```
  for (i = 0; i < dim; i++) {
    somme = somme + a[i][j];
  }
  cout << somme << " ";
}
```

Corrigé Exercice 4 (en C++) : La partie essentielle (tourner à gauche) :

```
for (i = 0; i < dim; i++) {
    for (j = 0; j < dim; j++) {
      b[i][j] = a[j][dim-i-1];
    }
  }
```

La partie essentielle (tourner à droite) :

```
for (i = 0; i < dim; i++) {
    for (j = 0; j < dim; j++) {
      b[i][j] = a[dim-j-1][i];
    }
  }
```

Corrigé Exercice 5 : *en pseudo-langage*

```
"programme" PascalMatrice
"constantes"
    dim = 5
"types"
    vecteur = "tableau de" Dim entiers
    matr    = "tableau de" Dim vecteurs
"variables"
    a   : matrice
    l,c : entier positifs
"debut" PascalMatrice
   // initialisation a 0
   initialiseNul a

   // le bord gauche du triangle
   "pour" l <-- 1 "a" dim "faire"
      a[l][1] <-- 1
   "fin pour"

   // remplissage
   "pour" l <-- 2 "a" dim "faire"
      "pour" c <-- 2 "a" l "faire"
         a[l][c] <-- a[l-1][c-1] + a[l-1][c]
      "fin pour"
   "fin pour"

   // affichage
   affiche a

"fin" PascalMatrice
```

Pour la seconde solution, on calcule de droite à gauche.

en pseudo-langage

```
"programme" PascalVecteur
"constantes"
    dim = 5
"types"
    vecteur = "tableau de" Dim entiers
"variables"
    a     : vecteur
    l,c   : entier positifs
"debut" PascalMatrice
   // initialisation a 0
   initialiseNul a

   // initialisation et invariant du triangle
   a[1] <-- 1

   // remplissage et affichage de chaque ligne
   "pour" l <-- 2 "a" dim "faire"
      "pour" c <-- l "a" 2 "faire"
         a[c] <-- a[c-1] + a[c]
      "fin pour"

       // affichage de la ligne
       affiche a
   "fin pour"

"fin" PascalVecteur
```