TD-5
itérations, tableaux et constantes

Exercice 1 :
Écrire un programme qui remplit un tableau `a` avec dix entiers pris au hasard entre 0 à 99, et qui l'affiche.

Exercice 2 :
Écrire un programme qui calcule le produit scalaire de deux tableaux `a,b`.

Exercice 3 :
Ecrire un programme qui remplit un tableau `b` avec les éléments de `a` pris en ordre inverse.

Exercice 4 :
Écrire un programme qui modifie le tableau `a` en mettant ses éléments en ordre inverse.

Exercice 5 :
Écrire un programme qui affiche la valeur et la position d'un des plus grand élément d'un tableau `a`.

Exercice 6 :
Écrire un programme qui échange le plus grand élément d'un tableau `a` avec son dernier élément de `a`.

Exercice 7 :
Écrire un programme qui étant donné un tableau `a` de notes entre 0 et 20, remplit un tableau `note` de sorte que `note[i]` contienne le nombre de notes valant `i`. Afficher ensuite le tableau `note` sous la forme :

$$\begin{array}{c|l} 0 & *** \\ 1 & **** \\ \vdots & \vdots \\ 20 & ****** \end{array}$$

avec une $*$ par occurence de notes

Exercice 8 :
Écrire un programme qui s'inspire de l'exercice 6 pour trier le tableau `a` par ordre croissant.

Exercice 9 :
Ecrire un programme qui recherche dans une matrice les éléments qui sont des maximaux stricts à la fois sur une ligne donnée, et sur une colonne.

Notes et corrigé pour le TD-5

Notes Notions à introduire dans ce TD: Les tableaux et les constantes. On peut rappeler que la numerotation en pseudo va de 1 a dim alors qu'en C elle va de 0 a dim-1.

Rq : l'affichage d'un tableau et son initialisation aleatoire sera vu une fois. Puis dans la suite, en pseudo on dira simplement `affiche <tableau>` ou `initialise <tableau>`.

L'exercice 1 servira de base pour les tests en TP. Pas de procedures il faudra faire des copier-coller.

Corrigé Exercice 1 :

En pseudo

```
"programme" remplis

"constantes"
    // on fixe la dimension du tableau a 10
    dim = 10

"types"
    vecteur = "tableau de" Dim entiers

"Variables"
    a : vecteur
    i : entier positif

"debut" remplis

  // un remplissage aleatoire entre 0 et 99
  "pour" i <-- 1 "a" dim "faire"
      a[i] <-- random(0,99)
  "fin pour"

  // l'affichage de gauche a droite
  "pour" i <-- 1 "a" dim "faire"
      "afficher" a[i]
  "fin pour"

"fin" remplis
```

en C++

```
#include <iostream.h>

#include <stdlib.h> // pour random
#include <time.h>

const int dim = 10;     // dimension du tableau

typedef int vecteur[dim];

main () {

  int i;
  vecteur a;

  srand(time(NULL));    // initialisation du generateur aleatoire
                              // remplir tableau a
  for (i = 0; i < dim; i++) { // avec des entiers
    a[i] = rand() % 100;      // aleatoires
  }                           // entre 0 et 99

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

}
```

Corrigé Exercice 2 :

```
"programme" scalaire

"constantes"
    // on fixe la dimension du tableau a 10
    dim = 10

"types"
    vecteur = "tableau de" Dim entiers

"Variables"
    a,b  : vecteur
    i,sc : entier positif
```

```
"debut" scalaire

  // initialisation aleatoire de a et b
  initialise a,b

  // au depart sc est nul
  sc <-- 0

  // calcul iteratif du produit
  "pour" i <-- 1 "a" dim "faire"
      sc <-- sc + a[i]*b[i]
```

```
  "fin pour"

  // affichage du resultat
  afficher ''le produit scalaire de'' a
          ''et de'' b ''vaut'' sc
"fin" scalaire
```

Corrigé Exercice 3 :

En pseudo

```
"programme" copieInverse

"constantes"
    // on fixe la dimension du tableau a 10
    dim = 10

"types"
    vecteur = "tableau de" Dim entiers

"Variables"
    a,b  : vecteur
    i    : entier positif

"debut" copieInverse

  // initialisation aleatoire de a
  initialise a

  "pour" i <-- 1 "a" dim "faire"
      b[i] <-- a[dim-i+1]
  "fin pour"

  // affichage du resultat
  afficher ''le tableau'' a
          ''a pour miroir'' b
"fin" copieInverse
```

La partie significative en C :

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

Corrigé Exercice 4 : On peut donner deux solutions ici, l'une déduite de l'exercice 3, l'autre sans tableau auxilliaire :

En pseudo

```
"programme" Inverse

"constantes"
    // on fixe la dimension du tableau a 10
    dim = 10

"types"
    vecteur = "tableau de" Dim entiers

"Variables"
    a     : vecteur
    aux,i : entier positif

"debut" Inverse

  // initialisation aleatoire de a
  initialise a
  afficher ''initialement on a'' a

  "pour" i <-- 1 "a" dim/2 "faire"
      aux  <-- a[i]       // sauvegarde
      a[i] <-- a[dim-i+1] // puis
      a[dim-i+1] <-- aux  // echange
  "fin pour"

  // affichage du resultat
  afficher ''et pour miroir'' a
"fin" Inverse
```

La partie significative en C :
On peut ici souligner l'importance de l'inégalité stricte de $i < dim/2$ pour le passage du pseudo en C, dû à la difference de numerotation. Si on deroule l'exemple buggé ou $<$ est remplace par $<=$ et $dim = 4$ il y aura deux flips entre $a[1]$ et $a[2]$ (numérotés en C).

```
  for (i = 0; i < dim / 2; i++) {
    aide = a[dim - i - 1];
    a[dim - i - 1] = a[i];
    a[i] = aide;
  }
```

Corrigé Exercice 5 : En pseudo

```
"programme" MaxPos

"constantes"
    // on fixe la dimension du tableau a 10
    dim = 10
```

```
"types"
   vecteur = "tableau de" Dim entiers

"Variables"
   a         : vecteur
   pos,max,i : entier positif

"debut" MaxPos

  // initialisation aleatoire de a
  initialise a
  afficher ``initialement on a'' a

  // on commence par un maximum provisoire
  pos <-- 0
  max <-- a[1]

  "pour" i <-- 1 "a" dim "faire"
      // inv : a[pos] >= a[j] pour j<= i
      "si" (a[i] > max) "faire"
          max <-  a[i]
          pos <-- i
      "fin si"
  "fin pour"

  // affichage du resultat
  afficher ``le max est'' max ``par exemple en position'' pos
"fin" maxPos
```

La partie significative en C :

```
max = a[0];
pos = 0;

// Rq que l'on peut commencer a 1
for (i = 1; i < dim; i++) {
  if (a[i] > max) {
    max = a[i];
    pos = i;
  }
}
```

Corrigé Exercice 6 : Il se déduit immédiatement de l'exercice 5

```
max = a[0];
pos = 0;

for (i = 1; i < dim; i++) {
  if (a[i] > max) {
    max = a[i];
    pos = i;
  }
}

aide = a[dim-1];
a[dim-1] = a[pos];
a[pos] = aide;
```

Corrigé Exercice 7 : Rq que les notes sont de 0 a 20, soit 21 possibilités. l'initialisation aleatoire, ou a zero peuvent etre mise dans le pseudo, pourvu que l'on explique a cote que c'est un Pb plus simple.

Ici la convention d'indicage des tableaux pose des problemes. Il est peut etre interessant de donner aussi la solution en C.

```
"programme" Statistiques

"constantes"
   // on fixe la dimension du tableau a 10
   dim = 10
   // on note de 0 a 20
   intervalle = 21
"types"
   vecteur     = "tableau de" Dim entiers
   echantillon = "tableau de" intervalle entiers

"Variables"
   a         : vecteur
   notes     : echantillon
   i,j       : entier positif

"debut" Statistiques

  // initialisation aleatoire de a entre 0 et 20 !
  initialise a
  afficher ``le pannel des notes'' a
```

```
  // initialisation a 0 de l'echantillon
  mise a zero notes

  "pour" i <-- 1 "a" dim "faire"
      j <-- a[i]
      note[j+1] <-- note[j+1] +1
  "fin pour"

  // affichage graphique du resultat
  "pour" i <-- 1 "a" intervalle "faire"
      // indication de la note
      afficher (i-1) `` | ''
      // une etoile par occurence
      "pour" j <-- 1 "a" note[i] "faire"
      afficher ``*''
      "fin pour"

      // retour a la ligne
      afficher ``/n''

  "fin pour" // passage a la note suivante

"fin" statistiques
```

Corrigé Exercice 8 (complet):

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

const dim = 10;

typedef int vecteur[dim];

main () {

  unsigned germe = time(NULL);
  int i, j, max, pos, aide;
  vecteur a;

  srand(germe);                      // remplir le tableaux a
  for (i = 0; i < dim; i++) {        // avec des entiers
    a[i] = rand() % 100;             // aleatoires entre 0 et 99
  }

  for (i = 0; i < dim; i++) {        // afficher le tableau a
    cout << a[i] << " ";
  }
  cout << "\n";

  for (j = dim - 1; j > 0; j--) {    // tri par insertion

    max = a[0];
    pos = 0;
    for (i = 1; i <= j; i++) {       // Trouver le maximum du tableau
      if (a[i] > max) {              // entre 0 et j
        max = a[i];
        pos = i;
      }
    }

    aide = a[j];                     // echanger a[j] avec a[pos]
    a[j] = a[pos];                   // a[pos] est le maximum
    a[pos] = aide;
  }

  for (i = 0; i < dim; i++) {        // afficher le tableau a
```

```
    cout << a[i] << " ";
  }
  cout << "\n";

}
```