Université Cadi Ayad- Marrakech

Faculté des Sciences – Semlalia

Département Informatique

TP2

Programmation en Langage C SMI3

Exercice 1 :

```
#include <stdio.h>
main()
{
 /* Déclarations */
 /* Les tableaux et leurs dimensions */
 int A[50] ,N, i;


 /* Saisie des données */
 printf("Dimension du tableau A (max.50) : ");
 scanf("%d", &N );
 for (i=0; i<N; i++)
   {
    printf("Elément A[%d] : ", i);
    scanf("%d", &A[i]);
   }


   for (i=0; i<N; i++)

    printf("%d ", A[i]);

}
```

Exercice 2 :

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50]; /* tableau donné */
 int VAL;   /* valeur à rechercher   */
 int POS;   /* position de la valeur */
 int N;     /* dimension      */
```

```
 int I;    /* indice courant */

 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );
 for (I=0; I<N; I++)
   {
    printf("Elément %d : ", I);
    scanf("%d", &A[I]);
   }
 printf("Elément à rechercher : ");
 scanf("%d", &VAL );
 /* Affichage du tableau */
 printf("Tableau donné : \n");
 for (I=0; I<N; I++)
    printf("%d ", A[I]);
 printf("\n");
 /* Recherche de la position de la valeur */
 POS = -1;
 for (I=0 ; (I<N)&&(POS==-1) ; I++)
      if (A[I]==VAL)
         POS=I;
  /* Edition du résultat */
 if (POS==-1)
    printf("La valeur recherchée ne se trouve pas "
          "dans le tableau.\n");
  else
    printf("La valeur %d se trouve à la position %d. \n",
 VAL, POS);
 return 0;
}
```

Exercice 3:

```
#include <stdio.h>
main()
{
 /* Déclarations */
 /* Les tableaux et leurs dimensions */
 int A[50], B[50], FUS[100];
 int N, M;
 int IA, IB, IFUS; /* indices courants */

 /* Saisie des données */
 printf("Dimension du tableau A (max.50) : ");
 scanf("%d", &N );
 printf("Entrer les éléments de A dans l'ordre croissant :\n");
```

```
for (IA=0; IA<N; IA++)
  {
   printf("Elément A[%d] : ", IA);
   scanf("%d", &A[IA]);
  }
printf("Dimension du tableau B (max.50) : ");
scanf("%d", &M );
printf("Entrer les éléments de B dans l'ordre croissant :\n");
for (IB=0; IB<M; IB++)
  {
   printf("Elément B[%d] : ", IB);
   scanf("%d", &B[IB]);
  }
/* Affichage des tableaux A et B */
printf("Tableau A :\n");
for (IA=0; IA<N; IA++)
   printf("%d ", A[IA]);
printf("\n");
printf("Tableau B :\n");
for (IB=0; IB<M; IB++)
   printf("%d ", B[IB]);
printf("\n");

/* Fusion des éléments de A et B dans FUS */
 /* de façon à ce que FUS soit aussi trié. */
IA=0; IB=0; IFUS=0;
while ((IA<N) && (IB<M))
      if(A[IA]<B[IB])
         {
          FUS[IFUS]=A[IA];
          IFUS++;
          IA++;
         }
      else
         {
          FUS[IFUS]=B[IB];
          IFUS++;
          IB++;
         }
/* Si IA ou IB sont arrivés à la fin de leur tableau, */
/* alors copier le reste de l'autre tableau.         */
while (IA<N)
      {
       FUS[IFUS]=A[IA];
       IFUS++;
       IA++;
      }
```

```
while (IB<M)
      {
       FUS[IFUS]=B[IB];
       IFUS++;
       IB++;
      }

 /* Edition du résultat */
 printf("Tableau FUS :\n");
 for (IFUS=0; IFUS<N+M; IFUS++)
    printf("%d ", FUS[IFUS]);
 printf("\n");
 return 0;
}
```

Exercice 4:

```
#include <stdio.h>

main()

{

 /* Déclarations */

 /* Les tableaux et leurs dimensions */

 int T[50], TPOS[50], TNEG[50];

 int N,     NPOS,     NNEG;

 int I; /* indice courant */


 /* Saisie des données */

 printf("Dimension du tableau (max.50) : ");

 scanf("%d", &N );

 for (I=0; I<N; I++)

   {

    printf("Elément %d : ", I);

    scanf("%d", &T[I]);

   }

 /* Affichage du tableau */
```

```
printf("Tableau donné :\n");
for (I=0; I<N; I++)
   printf("%d ", T[I]);
printf("\n");
/* Initialisation des dimensions de TPOS et TNEG */
NPOS=0;
NNEG=0;
/* Transfer des données */
for (I=0; I<N; I++)
    { if (T[I]>0) {
                TPOS[NPOS]=T[I];
                NPOS++;
               }
      if (T[I]<0) {
                TNEG[NNEG]=T[I];
                NNEG++;
               }
    }
 /* Edition du résultat */
printf("Tableau TPOS :\n");
for (I=0; I<NPOS; I++)
   printf("%d ", TPOS[I]);
printf("\n");
printf("Tableau TNEG :\n");
for (I=0; I<NNEG; I++)
   printf("%d ", TNEG[I]);
printf("\n");
return 0;
}
```

Exercice 5:

1. et 2.

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50][50]; /* matrice donnée   */
 int N, M;     /* dimensions des matrices */
 int I, J;     /* indices courants */

 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &M );
 printf("*** Matrice A ***\n");
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &A[I][J]);
      }

printf("Matrice donnée A :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
```

3. Somme de deux Matrice A et B

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50][50]; /* matrice donnée   */
 int B[50][50]; /* matrice donnée   */
 int C[50][50]; /* matrice résultat */
 int N, M;     /* dimensions des matrices */
 int I, J;     /* indices courants */

 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes (max.50) : ");
```

```
 scanf("%d", &M );
 printf("*** Matrice A ***\n");
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &A[I][J]);
      }
 printf("*** Matrice B ***\n");
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &B[I][J]);
      }
 /* Affichage des matrices */
 printf("Matrice donnée A :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
 printf("Matrice donnée B :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", B[I][J]);
    printf("\n");
   }

 /* Affectation du résultat de l'addition à C */
 for (I=0; I<N; I++)
    for (J=0; J<M; J++)
         C[I][J] = A[I][J]+B[I][J];
  /* Edition du résultat */
 printf("Matrice résultat C :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", C[I][J]);
    printf("\n");
   }
 return 0;
}
```

4. Le produit des deux matrices

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50][50]; /* matrice donnée   */
 int B[50][50]; /* matrice donnée   */
 int C[50][50]; /* matrice résultat */
 int N, M, P;   /* dimensions des matrices */
 int I, J, K;   /* indices courants */

 /* Saisie des données */
 printf("*** Matrice A ***\n");
 printf("Nombre de lignes de   A (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes de A (max.50) : ");
 scanf("%d", &M );
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &A[I][J]);
      }
 printf("*** Matrice B ***\n");
 printf("Nombre de lignes de   B : %d\n", M);
 printf("Nombre de colonnes de B (max.50) : ");
 scanf("%d", &P );
 for (I=0; I<M; I++)
   for (J=0; J<P; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &B[I][J]);
      }
 /* Affichage des matrices */
 printf("Matrice donnée A :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", A[I][J]);
    printf("\n");
   }
 printf("Matrice donnée B :\n");
 for (I=0; I<M; I++)
   {
    for (J=0; J<P; J++)
        printf("%7d", B[I][J]);
    printf("\n");
   }
```

```
/* Affectation du résultat de la multiplication à C */
for (I=0; I<N; I++)
   for (J=0; J<P; J++)
      {
       C[I][J]=0;
       for (K=0; K<M; K++)
           C[I][J] += A[I][K]*B[K][J];
      }
 /* Edition du résultat */
printf("Matrice résultat C :\n");
for (I=0; I<N; I++)
  {
   for (J=0; J<P; J++)
       printf("%7d", C[I][J]);
   printf("\n");
  }
return 0;
}
```

5. La transposition d'une matrice.

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int A[50][50];  /* matrice initiale */
 int B[50][50];  /* matrice résultat */
 int N, M;      /* dimensions des matrices */
 int I, J;      /* indices courants */

 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &N );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &M );
 for (I=0; I<N; I++)
   for (J=0; J<M; J++)
      {
       printf("Elément[%d][%d] : ",I,J);
       scanf("%d", &A[I][J]);
      }
 /* Affichage de la matrice */
 printf("Matrice donnée :\n");
 for (I=0; I<N; I++)
   {
    for (J=0; J<M; J++)
        printf("%7d", A[I][J]);
```

```
    printf("\n");
   }
 /* Affectation de la matrice transposée à B */
 for (I=0; I<N; I++)
    for (J=0; J<M; J++)
        B[J][I]=A[I][J];
  /* Edition du résultat */
 /* Attention: maintenant le rôle de N et M est inversé. */
  printf("Matrice résultat :\n");
 for (I=0; I<M; I++)
   {
    for (J=0; J<N; J++)
        printf("%7d", B[I][J]);
    printf("\n");
   }
 return 0;
}
```

Exercice 6:

a) `char a[] = "un\ndeux\ntrois\n";`

Déclaration correcte

Espace: 15 octets

b) `char b[12] = "un deux trois";`

Déclaration incorrecte: la chaîne d'initialisation dépasse le bloc de mémoire réservé.

Correction: `char b[14] = "un deux trois";`

ou mieux: `char b[] = "un deux trois";`

Espace: 14 octets

c) `char c[] = 'abcdefg';`

Déclaration incorrecte: Les symboles `''` encadrent des caractères; pour initialiser avec une chaîne de caractères, il faut utiliser les guillemets (ou indiquer une liste de caractères).

Correction: `char c[] = "abcdefg";`

Espace: 8 octets

d) `char d[10] = 'x';`

Déclaration incorrecte: Il faut utiliser une liste de caractères ou une chaîne pour l'initialisation

Correction: `char d[10] = {'x', '\0'}`

ou mieux: `char d[10] = "x";`

Espace: 2 octets

e) `char e[5] = "cinq";`

Déclaration correcte

Espace: 5 octets

f) `char f[] = "Cette " "phrase" "est coupée";`

Déclaration correcte

Espace: 23 octets

g) `char g[2] = {'a', '\0'};`

Déclaration correcte

Espace: 2 octets

h) `char h[4] = {'a', 'b', 'c'};`

Déclaration incorrecte: Dans une liste de caractères, il faut aussi indiquer le symbole de fin de chaîne.

Correction: `char h[4] = {'a', 'b', 'c', '\0'};`

Espace: 4 octets

i) `char i[4] = "'o'";`

Déclaration correcte, mais d'une chaîne contenant les caractères `'\''`, `'o'`, `'\''` et `'\0'`.

Espace: 4 octets