TP 1(Rappel)

Exercice 1

Ecrire un programme qui lit la dimension N d'un tableau T du type **int** (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Calculer et afficher ensuite la somme des éléments du tableau.

Exercice 2

Ecrire un programme qui lit la dimension N d'un tableau T du type **int** (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Effacer ensuite toutes les occurrences de la valeur 0 dans le tableau T et tasser les éléments restants. Afficher le tableau résultant.

Exercice 3

Ecrire un programme qui lit la dimension N d'un tableau T du type **int** (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Ranger ensuite les éléments du tableau T dans l'ordre inverse sans utiliser de tableau d'aide. Afficher le tableau résultant.

Idée: Echanger les éléments du tableau à l'aide de deux indices qui parcourent le tableau en commençant respectivement au début et à la fin du tableau et qui se rencontrent en son milieu.

Exercice 4

Ecrire un programme qui lit la dimension N d'un tableau T du type **int** (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Copiez ensuite toutes les composantes strictement positives dans un deuxième tableau TPOS et toutes les valeurs strictement négatives dans un troisième tableau TNEG. Afficher les tableaux TPOS et TNEG.

Exercice 5

Ecrire un programme qui effectue la transposition t_A d'une matrice A de dimensions N et M en une matrice de dimensions M et N.

a) La matrice transposée sera mémorisée dans une deuxième matrice B qui sera ensuite affichée.

b) La matrice A sera transposée par permutation des éléments.

Rappel:

```
          /         \      /       \
tA =   t  | a b c d |  =   | a e i |
          | e f g h |      | b f j |
          | i j k l |      | c g k |
          \         /      | d h l |
                           \       /
```

Exercice 6

Lesquelles des chaînes suivantes sont initialisées correctement ? Corrigez les déclarations fausses et indiquez pour chaque chaîne de caractères le nombre d'octets qui sera réservé en mémoire.

```
a)   char a[]   = "un\ndeux\ntrois\n";
b)   char b[12] = "un deux trois";
c)   char c[]   = 'abcdefg';
d)   char d[10] = 'x';
e)   char e[5]  = "cinq";
f)   char f[]   = "Cette "   "phrase"   "est coupée";
g)   char g[2]  = {'a', '\0'};
h)   char h[4]  = {'a', 'b', 'c'};
i)   char i[4]  = "'o'";
```

Exercice 7

Ecrire un programme qui lit deux chaînes de caractères CH1 et CH2 et qui copie la première moitié de CH1 et la première moitié de CH2 dans une troisième chaîne CH3. Afficher le résultat.

a) Utiliser les fonctions spéciales de *<string>.*

b) Utiliser uniquement les fonctions **gets** et **puts**.

Exercice 8

Ecrire un programme qui lit un verbe régulier en "er" au clavier et qui en affiche la conjugaison au présent de l'indicatif de ce verbe. Contrôlez s'il s'agit bien d'un verbe en "er" avant de conjuguer. Utiliser les fonctions **gets**, **puts**, **strcat** et **strlen**.

Exemple:

```
Verbe : fêter
je fête
tu fêtes
il fête
nous fêtons
vous fêtez
ils fêtent
```

TP 1
CORRIGÉS DES EXERCICES

Exercice 1

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int T[50]; /* tableau donné  */
 int N;     /* dimension      */
 int I;     /* indice courant */
 long SOM;  /* somme des éléments - type long à cause */
            /* de la grandeur prévisible du résultat. */

 /* 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");
  /* Calcul de la somme */
 for (SOM=0, I=0; I<N; I++)
     SOM += T[I];
  /* Edition du résultat */
 printf("Somme de éléments : %ld\n", SOM);
 return 0;
}
```

Exercice 2

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int T[50]; /* tableau donné    */
 int N;     /* dimension        */
 int I,J;   /* indices courants */

 /* 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");
 /* Effacer les zéros et comprimer :        */
 /* Copier tous les éléments de I vers J et */
 /* augmenter J pour les éléments non nuls. */
 for (I=0, J=0 ; I<N ; I++)
     {
      T[J] = T[I];
      if (T[I]) J++;
     }
 /* Nouvelle dimension du tableau ! */
 N = J;
  /* Edition des résultats */
 printf("Tableau résultat :\n");
 for (I=0; I<N; I++)
     printf("%d ", T[I]);
 printf("\n");
 return 0;
}
```

Exercice 3

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int T[50]; /* tableau donné */
 int N;     /* dimension     */
 int I,J;   /* indices courants */
 int AIDE;  /* pour l'échange   */

 /* 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");
 /* Inverser le tableau */
 for (I=0, J=N-1 ; I<J ; I++,J--)
      /* Echange de T[I] et T[J] */
        {
         AIDE = T[I];
         T[I] = T[J];
         T[J] = AIDE;
        }
```

```
  /* Edition des résultats */
 printf("Tableau résultat :\n");
 for (I=0; I<N; I++)
     printf("%d ", T[I]);
 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

a) La matrice transposée sera mémorisée dans une deuxième matrice B qui sera ensuite affichée.

```
#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;
}
```

b) La matrice A sera transposée par permutation des éléments.

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

```
 int AIDE;       /* pour la permutation      */
 int DMAX;       /* la plus grande des deux dimensions */

 /* 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");
    }
 /* Transposition de la matrice A par permutation des   */
 /* éléments [I][J] à gauche de la diagonale principale */
 /* avec les éléments [J][I] à droite de la diagonale.  */
 DMAX = (N>M) ? N : M;
 for (I=0; I<DMAX; I++)
     for (J=0; J<I; J++)
         {
          AIDE = A[I][J];
          A[I][J] = A[J][I];
          A[J][I] = AIDE;
         }
  /* 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", A[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

Exercice 7

a) Utiliser les fonctions spéciales de *<string>.*

```
#include <stdio.h>
#include <string.h>
 main()
{
 /* Déclarations */
 char CH1[100], CH2[100]; /* chaînes données */
 char CH3[100]="";        /* chaîne résultat */

 /* Saisie des données */
 printf("Introduisez la première chaîne de caractères : ");
 gets(CH1);
 printf("Introduisez la deuxième chaîne de caractères : ");
 gets(CH2);


 /* Traitements */
 strncpy(CH3, CH1, strlen(CH1)/2);
 strncat(CH3, CH2, strlen(CH2)/2);
  /* Affichage du résultat */
 printf("Un demi \"%s\" plus un demi \"%s\" donne \"%s\"\n",
                                              CH1, CH2, CH3);
 return 0;
}
```

b) Utiliser uniquement les fonctions **gets** et **puts**.

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char CH1[100], CH2[100]; /* chaînes données  */
 char CH3[100]="";        /* chaîne résultat  */
 int L1,L2; /* longueurs de CH1 et CH2        */
 int I;     /* indice courant dans CH1 et CH2 */
 int J;     /* indice courant dans CH3        */

 /* Saisie des données */
```

```
 puts("Introduisez la première chaîne de caractères : ");
 gets(CH1);
 puts("Introduisez la deuxième chaîne de caractères : ");
 gets(CH2);

 /* Détermination les longueurs de CH1 et CH2 */
 for (L1=0; CH1[L1]; L1++) ;
 for (L2=0; CH2[L2]; L2++) ;
 /* Copier la première moitié de CH1 vers CH3 */
 for (I=0 ; I<(L1/2) ; I++)
     CH3[I]=CH1[I];
 /* Copier la première moitié de CH2 vers CH3 */
 J=I;
 for (I=0 ; I<(L2/2) ; I++)
    {
     CH3[J]=CH2[I];
     J++;
    }
 /* Terminer la chaîne CH3 */
 CH3[J]='\0';

 /* Affichage du résultat */
 puts("Chaîne résultat : ");
 puts(CH3);
 return 0;
}
```

Exercice 8

```
#include <stdio.h>
#include <string.h>
 main()
{
 /* Déclarations */
 char VERB[20]; /* chaîne contenant le verbe */
 char AFFI[30]; /* chaîne pour l'affichage   */
 int L;         /* longueur de la chaîne     */

 /* Saisie des données */
 printf("Verbe : ");
 gets(VERB);

 /* Contrôler s'il s'agit d'un verbe en 'er' */
 L=strlen(VERB);
 if ((VERB[L-2]!='e') || (VERB[L-1]!='r'))
      puts("\aCe n'est pas un verbe du premier groupe.!");
 else
     {
      /* Couper la terminaison 'er'. */
      VERB[L-2]='\0';
      /* Conjuguer ... */
      AFFI[0]='\0';
      strcat(AFFI, "je ");
      strcat(AFFI, VERB);
      strcat(AFFI, "e");
      puts(AFFI);
```

```
        . . .

       AFFI[0]='\0';
       strcat(AFFI, "ils ");
       strcat(AFFI, VERB);
       strcat(AFFI, "ent");
       puts(AFFI);
      }
  return 0;
}
```