Université Mohammed Premier
Ecole Nationale des Sciences Appliquées - Oujda

TP avec corrections - Langage C
Filière STPI
Pr. Rachid MALEK

TP 1 – Généralités sur le langage C

Les objectifs du TP1 : Se familiariser avec DevC++, compiler et exécuter les deux exemples et faire sortir les aspects fondamentaux du langage C (structure générale, la fonction main, déclarations, …)

Exemple 1

```
// Directives au pré-processeur - Ceci est un commentaire
#include <stdio.h>
#include <math.h>

main ( ) // Fonction principale

{
// Déclaration des variables - Ceci est un commentaire
int i ;
float x ;
float racx ;

printf ("Bonjour\n") ; // Affichage sur l'écran - Ceci est un commentaire

printf ("\n") ; // Retour à la ligne - Ceci est un commentaire

printf ("Calcul de 5 racines carrées\n") ; // Affichage sur l'écran - Ceci est un commentaire
printf ("\n") ; // Retour à la ligne - Ceci est un commentaire

// Boucle - Ceci est un commentaire
for (i=0 ; i<5 ; i++) {
   printf ("Donnez un nombre réel x :\n ") ;

   scanf ("%f", &x) ; // Saisie au clavier de x - Ceci est un commentaire

   // Test - Ceci est un commentaire
   if (x < 0.0) printf ("Le nombre %f ne possède pas de racine carrée\n", x) ;
   else {
       racx = sqrt (x) ;
       printf ("Le nombre %f a pour racine carrée : %f\n", x, racx) ;
   } // Fin du test - Ceci est un commentaire
} // Fin de la boucle - Ceci est un commentaire

printf ("Travail terminé\n") ;
getch ( );
} // Fin du programme - Ceci est un commentaire
```

Exemple 2

```
#include <stdio.h>

main ( )
{
char op ;
int n1, n2 ;

printf ("opération souhaitée (+ ou *) ?\n ") ;
scanf ("%c", &op) ;

printf ("donnez 2 nombres entiers : \n") ;
scanf ("%d %d", &n1, &n2) ;

if (op == '+') printf ("leur somme est : %d \n", n1+n2) ;
else printf ("leur produit est : %d \n", n1*n2) ;
getch ( );  } // Fin du programme
```

TP 2 – Les types de base en langage C

L'objectif du TP2 : Manipuler les types de base (int, short int, long int, float, double, long double, char)

Exercice 1
Ecrire un programme qui affiche le nombre d'octets réservés sur votre machine pour les types :

- int, short int et long int
- float, double et long double
- char

Utiliser la fonction sizeof ( ) dans un printf ( ). Exemple : *printf ("La taille en octets d'un entier (int) est : %d \n", **sizeof (int)**) ;*

Exercice 2
Soient les déclarations suivantes :
int i; short int j ; long int k ; float x ; double y ; long double z; char c ;

Ecrire un programme qui lit au clavier toutes ces variables et affiche leur adresse ainsi que leur valeur respective. Quel est l'ordre de placement de ces variables en mémoire ?
Pour afficher l'adresse d'une variable ***var***, utilisez &*var* dans un printf ( ). Exemple pour ***float x*** : *printf ("La valeur de x est : %f et son adresse est : %d \n", x, &x);*

Exercice 3
Ecrire un programme qui lit un caractère au clavier et affiche le caractère ainsi que son code numérique (code ASCII).

Exercice 4
Ecrire un programme qui lit au clavier les valeurs de trois résistances et de trois capacités et calcule leur résistance et leur capacité équivalente, respectivement, dans les deux cas :

- Les trois résistances et les trois capacités sont placées en série
- Les trois résistances et les trois capacités sont placées en parallèle

Les résultats doivent être affichés dans chaque cas. Toutes les variables utilisées dans ce programme doivent être déclarées avec le type ***double***.

Exercice 5
Soient les déclarations suivantes : int i ; float f ; char c ;

```
c = 98 ;  // commenter cette instruction
c = (char) 98;  // commenter cette instruction

i =  'a' ; // commenter cette instruction
i = (int) 'a' ; // commenter cette instruction

f = 3 ; // commenter cette instruction
f = (float) 3 ; // commenter cette instruction

i = 3.4 ; // commenter cette instruction
i = -3.3 ; // commenter cette instruction
i = (int) 3.4 ; // commenter cette instruction
```

Eléments de réponses - TP2

Exercice 1

printf ("Le nombre d'octets réservés pour un entier est : %d\n", sizeof (int));
Même chose pour les autres types.

Exercice 2

printf("L'adresse de la variable i est : %d\n", &i);
Même chose pour les autres variables.

Exercice 3

```
#include <stdio.h>
main()
{
  double R1, R2, R3, Rs, Rp;
  printf("Introduisez les valeurs pour R1, R2 et R3 : ");
  scanf("%lf %lf %lf", &R1, &R2, &R3);
  Rs = R1+R2+R3;
  printf("Résistance résultante en série   : %f\n", Rs);
  Rp = (R1*R2*R3)/(R1*R2+R1*R3+R2*R3);
  printf("Résistance résultante en parallèle  : %f\n", Rp);


return 0;
}
```

Exercice 4

```
#include <stdio.h>
#include <conio.h>
 main()
{
 int c;
 printf("Introduire un caractère \n");
 c = getch();
  printf("Le caractère %c a le code ASCII %d\n", c, c);

getch() ; // juste pour garder la fenêtre de l'exécution
 return 0;
}
```

Exercice 5

```
// conversion entier vers char.
c=98; // implicite : c prend le code ASCII 98 c-à-d 'b'
c = (char) 98;  // explicite plus propre

// char vers entier
i= 'a' ; // i prend la valeur 97
i= (int) 'a' ; //plus propre

// entier vers réel
f=3; // f prend la valeur 3.0;
f=(float) 3; //+ propre

//réel vers entier, attention : troncature
i = 3.4; // i prend la valeur 3
i= -3.3; // i prend la valeur -3
i = (int) 3.4; // + propre
```

TP 3 – Les opérateurs et les expressions en langage C

Les objectifs du TP3 : Manipuler les différents opérateurs et les expressions en langage C.

Exercice 1
Evaluer les expressions suivantes en supposant : A=20 B=5 C=-10 D=2 X=12 Y=15

(1)	(5*X)+2*((3*B)+4)
(2)	(5*(X+2)*3)*(B+4)
(3)	A == (B=5)
(4)	A += (X+5)
(5)	A != (C *= (-D))
(6)	A *= C+(X-D)
(7)	A %= D++
(8)	A %= ++D
(9)	(X++)*(A+C)
(10)	A = X*(B<C)+Y*!(B<C)

Exercice 2
Ecrire un programme qui permute les valeurs de trois variables entières A, B et C.
A prendra la valeur de C, B celle de A et C celle de B. Afficher le résultat final.

Exercice 3
Ecrire un programme qui calcule la distance entre deux points A et B du plan Oxy.
Le programme doit lire les coordonnées de A et B. Afficher le résultat final.

Exercice 4
Ecrire un programme qui lit trois nombres entiers A, B et C et affiche leur valeur maximale.
Utiliser les trois méthodes suivantes :

a) **if - else** et une variable d'aide MAX
b) **if - else if - ... - else** sans variable d'aide
c) **opérateur conditionnel ?** et une variable d'aide MAX

Exercice 5
Ecrire un programme qui lit trois nombres entiers A, B et C et effectue un tri par ordre décroissant de ces derniers en échangeant leur valeur. Introduire une variable AIDE.

Exercice 6
Ecrire un programme qui calcule les solutions réelles d'une équation du second degré.
On supposera que les coefficients a, b et c sont des nombres entiers.

Exercice 7
Ecrire un programme qui affiche le signe du produit de deux entiers A et B sans faire la multiplication.

Eléments de réponses – TP3

Exercice 1

(1)	(5*X)+2*((3*B)+4)	-> 98	/
(2)	(5*(X+2)*3)*(B+4)	-> 1890	/
(3)	A == (B=5)	-> 0	B=5
(4)	A += (X+5)	-> 37	A=37
(5)	A != (C *= (-D))	-> 0	C=20
(6)	A *= C+(X-D)	-> 0	A=0
(7)	A %= D++	-> 0	D=3 A=0
(8)	A %= ++D	-> 2	D=3 A=2
(9)	(X++)*(A+C)	-> 120	X=13
(10)	A = X*(B<C)+Y*!(B<C)	-> 0+15 = 15	A=15

Exercice 2

```
#include <stdio.h>
main()
{
        int A, B, C, AIDE;
        printf("Introduisez trois nombres (A, B, C) : ");
        scanf("%d %d %d", &A, &B, &C);
        /* Affichage à l'aide de tabulations */
        printf("A = %d\tB = %d\tC = %d\n", A, B, C);
        AIDE=A;
        A=C;
        C=B;
        B=AIDE;
        printf("A = %d\tB = %d\tC = %d\n", A, B, C);
        return 0;
}
```

Exercice 3

```
#include <stdio.h>
#include <math.h>
 main()
{
 int XA, YA, XB, YB;
 double DIST;
 /* Attention: La chaîne de format que nous utilisons */
 /* s'attend à ce que les données soient séparées par */
 /* une virgule lors de l'entrée. */
 printf("Entrez les coordonnées du point A :  XA,YA  ");
 scanf("%d,%d", &XA, &YA);
 printf("Entrez les coordonnées du point B :  XB,YB  ");
 scanf("%d,%d", &XB, &YB);
 DIST=sqrt(pow(XA-XB,2)+pow(YA-YB,2));
 printf("La distance entre A(%d,%d) et B(%d, %d) est %.2f\n",
XA, YA, XB, YB, DIST);
 return 0;
}
```

Exercice 4

a) if - else et une variable d'aide MAX

```
#include <stdio.h>
main()
```

```
{
 int A, B, C;
 int MAX;
 printf("Introduisez trois nombres entiers :");
 scanf("%d %d %d", &A, &B, &C);
 if (A>B)
    MAX=A;
 else
    MAX=B;
 if (C>MAX)
    MAX=C;
 printf("La valeur maximale est %d\n", MAX);
  return 0;
}
```

b) if - else if - ... - else sans variable d'aide

```
 int A, B, C;
 printf("Introduisez trois nombres entiers :");
 scanf("%d %d %d", &A, &B, &C);
 printf("La valeur maximale est ");
 if (A>B && A>C)
    printf("%d\n",A);
 else if (B>C)
    printf("%d\n",B);
 else
    printf("%d\n",C);
```

c) opérateur conditionnel ? et une variable d'aide MAX

```
 int A, B, C;
 int MAX;
 printf("Introduisez trois nombres entiers :");
 scanf("%d %d %d", &A, &B, &C);
 MAX = (A>B) ? A : B;
 MAX = (MAX>C) ? MAX : C;
 printf("La valeur maximale est %d\n", MAX);
```

Exercice 5

```
#include <stdio.h>
main()
{
 /* Tri par ordre décroissant de trois entiers
    en échangeant les valeurs
 */
 int A, B, C, AIDE;
 printf("Introduisez trois nombres entiers :");
 scanf("%d %d %d", &A, &B, &C);
 printf("Avant le tri : \tA = %d\tB = %d\tC = %d\n", A, B, C);
 /* Valeur maximale -> A */
 if (A<B)
    {
     AIDE = A;
     A = B;
     B = AIDE;
    }
 if (A<C)
    {
     AIDE = A;
     A = C;
     C = AIDE;
    }
 /* trier B et C */
 if (B<C)
```

```
    {
     AIDE = B;
     B = C;
     C = AIDE;
    }
 printf("Après le tri : \tA = %d\tB = %d\tC = %d\n", A, B, C);
  return 0;
}
```

Exercice 6

```
#include <stdio.h>
#include <math.h>
 main()
{
 /* Calcul des solutions réelles d'une équation du second degré */
 int A, B, C;
 double D; /* Discriminant */
 printf("Calcul des solutions réelles d'une équation du second \n");
 printf("degré de la forme   ax^2 + bx + c = 0 \n\n");
 printf("Introduisez les valeurs pour a, b, et c : ");
 scanf("%d %d %d", &A, &B, &C);

 /* Calcul du discriminant b^2-4ac */
 D = pow(B,2) - 4.0*A*C;

 /* Distinction des différents cas */
 if (A==0 && B==0 && C==0) /* 0x = 0 */
                    printf("Tout réel est une solution de cette équation.\n");
 else if (A==0 && B==0)  /* Contradiction: c # 0 et c = 0 */
                    printf("Cette équation ne possède pas de solutions.\n");
 else if (A==0) /* bx + c = 0 */
    {
     printf("La solution de cette équation du premier degré est :\n");
     printf(" x = %.4f\n", (double)C/B);
    }
 else if (D<0)  /* b^2-4ac < 0 */
     printf("Cette équation n'a pas de solutions réelles.\n");
 else if (D==0) /* b^2-4ac = 0 */
    {
     printf("Cette équation a une seule solution réelle :\n");
     printf(" x =  %.4f\n", (double)-B/(2*A));
    }
 else /* b^2-4ac > 0 */
    {
     printf("Les solutions réelles de cette équation sont :\n");
     printf(" x1 = %.4f\n", (-B+sqrt(D))/(2*A));
     printf(" x2 = %.4f\n", (-B-sqrt(D))/(2*A));
    }
  return 0;
}
```

Exercice 7

```
#include <stdio.h>
main()
{
 /* Afficher le signe du produit de deux entiers sans
    faire la multiplication
 */
 int A, B;
 printf("Introduisez deux nombres entiers :");
```

```
 scanf("%d %d", &A, &B);
 if ((A>0 && B>0) || (A<0 && B<0))
   printf("Le signe du produit %d * %d est positif\n", A, B);
 else if ((A<0 && B>0) || (A>0 && B<0))
   printf("Le signe du produit %d * %d est négatif\n", A, B);
 else
   printf("Le produit %d * %d est zéro\n", A, B);
  return 0;
}
```

TP 4 – Boucles & Tableaux numériques à une dimension

Exercice 1

Soit le programme suivant :

```
#include <stdio.h>
main ( )
{
int i, n, som ;
som = 0 ;
for (i=0 ; i<4 ; i++) {
        printf ("donnez un entier : ") ;
        scanf ("%d", &n) ;
        som += n ;
}
printf ("Somme : %d\n", som) ;
}
```

Écrire un programme réalisant exactement la même chose, en employant, à la place de l'instruction for une instruction **while** et une instruction **do... while**.

Exercice 2

Calculer la moyenne de notes fournies au clavier avec un dialogue de ce type :
note 1 : 12
note 2 : 15.25
note 3 : 13.5
note 4 : 8.75
note 5 : -1
La moyenne de ces 4 notes est : 12.37

Le nombre de notes n'est pas connu a priori et l'utilisateur peut en fournir autant qu'il le désire. Pour signaler qu'il a terminé, on convient qu'il fournira une note fictive négative. Celle-ci ne devra naturellement pas être prise en compte dans le calcul de la moyenne.

Exercice 3

Écrire un programme qui détermine la *n*-ième valeur u_n (*n* étant fourni en donnée) de la suite de Fibonacci définie comme suit :
$u_1 = 1$
$u_2 = 1$
$u_n = u_{n-1} + u_{n-2}$ pour $n>2$

Exercice 4

Ecrire un programme qui détermine la plus grande et la plus petite valeur dans un tableau d'entiers A. Afficher ensuite la valeur et la position du maximum et du minimum. Si le tableau contient plusieurs maxima ou minima, le programme retiendra la position du premier maximum ou minimum rencontré.

Exercice 5

Rechercher dans un tableau d'entiers A une valeur VAL entrée au clavier. Afficher la position de VAL si elle se trouve dans le tableau, sinon afficher un message correspondant. La valeur POS qui est utilisée pour mémoriser la position de la valeur dans le tableau, aura la valeur -1 aussi longtemps que VAL n'a pas été trouvée.

a) La recherche séquentielle
Comparer successivement les valeurs du tableau avec la valeur donnée.

b) La recherche dichotomique ('recherche binaire', 'binary search')
Condition: *Le tableau A doit être trié*

Ecrire le programme dans le cas où le tableau A est trié par ordre croissant.

Eléments de réponses – TP4

Exercice 1

a)

```
#include <stdio.h>

main()
{ int i, n, som ;
som = 0 ;
i = 0 ; /* ne pas oublier cette "initialisation" */

while (i<4)
{ printf ("donnez un entier ") ;
scanf ("%d", &n) ;
som += n ;
i++ ; /* ni cette "incrémentation" */
}

printf ("Somme : %d\n", som) ;
}
```

b)

```
#include <stdio.h>

main()
{ int i, n, som ;
som = 0 ;
i = 0 ; /* ne pas oublier cette "initialisation" */

do
{ printf ("donnez un entier ") ;
scanf ("%d", &n) ;
som += n ;
i++ ; /* ni cette "incrémentation" */
} while (i<4) ; /* attention, ici, toujours <4 */

printf ("Somme : %d\n", som) ;
}
```

Exercice 2

```
#include <stdio.h>

main()
```

```
{ float note, /* note courante */
som, /* somme des notes */
moy ; /* moyenne des notes */
int num ; /* numéro note courante */
som=0 ; num=0 ;

while ( printf("note %d : ",num+1),
scanf ("%f", &note), note>=0 )
{ num++ ;
som += note ;
}

if (num>0)
{ moy = som/num ;
printf ("moyenne de ces %d notes : %f", num, moy) ;
} else printf ("--- aucune note fournie ---") ;
}
```

Exercice 3

```
main()
{
int u1, u2, u3 ; /* pour "parcourir" la suite */
int n ; /* rang du terme demandé */
int i ; /* compteur */

do{ printf ("rang du terme demandé (au moins 3) ? ") ;
scanf ("%d", &n) ;
} while (n<3) ;

u2 = u1 = 1 ; /* les deux premiers termes */
i = 2 ;

while (i++ < n) /* attention, l'algorithme ne fonctionne */
{ u3 = u1 + u2 ; /* que pour n > 2 */
u1 = u2 ;
u2 = u3 ;
}
printf ("Valeur du terme de rang %d : %d", n, u3) ;
}
```

Exercice 4

```
#include <stdio.h>
main()
{
```

```
 /* Déclarations */
 int A[50]; /* tableau donné  */
 int N;     /* dimension      */
 int I;     /* indice courant */
 int MIN;   /* position du minimum */
 int MAX;   /* position du maximum */

/* 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]);
    }

 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<N; I++)
     printf("%d ", A[I]);
 printf("\n");

 /* Recherche du maximum et du minimum */
 MIN=0;
 MAX=0;
 for (I=0; I<N; I++)
     {
      if(A[I]>A[MAX]) MAX=I;
      if(A[I]<A[MIN]) MIN=I;
     }
  /* Edition du résultat */
 printf("Position du minimum : %d\n", MIN);
 printf("Position du maximum : %d\n", MAX);
 printf("Valeur   du minimum : %d\n", A[MIN]);
 printf("Valeur   du maximum : %d\n", A[MAX]);
 return 0;
}
```

Exercice 5

a) La recherche séquentielle

Comparer successivement les valeurs du tableau avec la valeur donnée.

```
#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;
}
```

b) La recherche dichotomique ('recherche binaire', 'binary search')

```
#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 */
 int INF, MIL, SUP; /* limites du champ de recherche */

 /* 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");

 /* Initialisation des limites du domaine de recherche */
 INF=0;
 SUP=N-1;

 /* Recherche de la position de la valeur */
 POS=-1;
 while ((INF<=SUP) && (POS==-1))
        {
         MIL=(SUP+INF)/2;
         if (VAL < A[MIL])
              SUP=MIL-1;
         else if (VAL > A[MIL])
              INF=MIL+1;
         else
              POS=MIL;
        }
```

```
 /* 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;
}
```

TP 5 – Tableaux numériques et pointeurs (Partie 1)

Exercice 1
Ecrire un programme qui lit la dimension N d'un tableau T du type entier (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 entier (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 entier (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 entier (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
Soit P un pointeur qui "pointe" sur un tableau A :

```
int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};
int *P;
P = A;
```

Quelles valeurs ou adresses fournissent les expressions suivantes : ?

a- *P+2
b- *(P+2)
c- &P+1
d- &A[4]-3
e- A+3
f- &A[7]-P
g- P+(*P-10)
h- *(P+*(P+8)-A[7])

Exercice 6
Écrire un programme qui lit un entier X et un tableau A de type entier au clavier et élimine toutes les occurrences de X dans A en tassant les éléments restants. Le programme utilisera deux pointeurs P_1 et P_2 pour parcourir le tableau.

TP5 – Corrections

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) *P+2 => la valeur 14
b) *(P+2) => la valeur 34
c) &P+1 => l'adresse du pointeur derrière le pointeur P (rarement utilisée)
d) &A[4]-3 => l'adresse de la composante A[1]
e) A+3 => l'adresse de la composante A[3]
f) &A[7]-P => la valeur (indice) 7
g) P+(*P-10) => l'adresse de la composante A[2]
h) *(P+*(P+8)-A[7]) => la valeur 23

Exercice 6

```
#include<stdio.h>
#include<conio.h>
main()
{
int *p1,*p2,j;
int x,t[10],n;

p1=t;

printf("donner la dimension du tableau");
scanf("%d",&n);

for(p1=t;p1<t+n;p1++)
{
printf("donner la valeur de t[%d]",p1-t);
scanf("%d",p1);
}

printf("dooner la valeur");
scanf("%d",&x);

j=0;
p1=t;



for(p1=t;p1<t+n;p1++)
```

```
{   if(*p1==x)
    {  for(p2=p1;p2<t+n;p2++)
        {   if(*p2!=x)
            { *p1=*p2;
              *p2=x;
               p2=t+n-1;
               j=j+1;
            }
        }
    }
}

p1=t;
for(p1=t;p1<t+j;p1++) printf(" %d \t",*p1);

getch();
} // fin main
```

TP 6 – Tableaux numériques et pointeurs (Partie 2)

Exercice 1
Écrire un programme qui range les éléments d'un tableau A de type entier dans l'ordre inverse. Le programme utilisera deux pointeurs P_1 et P_2 et une variable numérique AIDE pour la permutation des éléments.

Exercice 2
Ecrire un programme qui lit la dimension N d'un tableau T du type entier (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. Utiliser le formalisme pointeur.

Exercice 3
Écrire un programme qui implémente une méthode de recherche de l'indice de la valeur minimale d'un intervalle d'un tableau d'entiers compris entre les indices deb et fin inclus. Utiliser le formalisme pointeur.

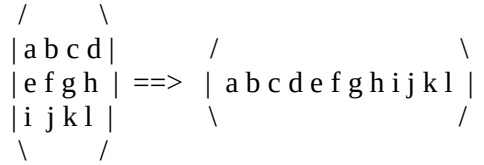
Exercice 4
Soit T un tableau de dimension N de type entier (dimension maximale : 10). Remplir le tableau par des valeurs entrées au clavier et afficher le tableau. Donner un code pour Trier les données de T à l'aide du pointeur. Utiliser le formalisme pointeur.

Exercice 5
Ecrire un programme qui lit les dimensions L et C d'un tableau T à deux dimensions du type int (dimensions maximales : 50 lignes et 50 colonnes). Remplir le tableau par des valeurs entrées au clavier et afficher le tableau ainsi que la somme de tous ses éléments.

Exercice 6
Ecrire un programme qui lit les dimensions L et C d'un tableau T à deux dimensions du type int (dimensions maximales : 50 lignes et 50 colonnes). Remplir le tableau par des valeurs entrées au clavier et afficher le tableau ainsi que la somme de chaque ligne et de chaque colonne en n'utilisant qu'une variable d'aide pour la somme.

Exercice 7
Ecrire un programme qui transfère un tableau M à deux dimensions L et C (dimensions maximales : 10 lignes et 10 colonnes) dans un tableau V à une dimension L*C.

Exemple :



TP6 – Corrections

Exercice1:

```
#include<stdio.h>
#include<conio.h>
main()
{
int t[10],*p1,*p2,n,aide,i;

printf("donnez la dimension du tab");
scanf("%d",&n);

p1=t;

for(i=0;i<n;i++)
{
printf("donnez les valeurs t[%d]=",i);
scanf("%d",p1+i);
}

p2=t+n-1;

while (p1<p2)
{
aide=*p1;
*p1=*p2;
*p2=aide;
p1++;
p2--;
}

for(i=0;i<n;i++) printf("t[%d]=%d",i,t[i]);

getch();
} // fin main
```

Exercice2:

```
#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. */
int *p ; // pointeur pour manipuler le tableau T
```

```
 /* Saisie des données */
 printf("Dimension du tableau (max.50) : ");
 scanf("%d", &N );

for (p = T; p < T + N; p++)
    {
     printf("Elément %d : ", p - T);
     scanf("%d", p);
    }
  /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (p = T; p < T + N; p++)
     printf("%d ", *p);
 printf("\n");
  /* Calcul de la somme */
 for (SOM = 0, p = T; p < T + N; p++)
     SOM += *p;
  /* Edition du résultat */
 printf("Somme de éléments : %ld\n", SOM);
getch();
}
```

Exercice3:

```
#include<stdio.h>
#include<conio.h>

main()
{
int t[10],*p,deb,fin,val,a,b;

p=t;

printf("donnez la dimension du tab");
scanf("%d",&a);

for(p=t;p<t+a;p++)
{
printf("donnez les valeurs t[%d]=",p-t+1);
scanf("%d",p); }
printf("donnez le début");
scanf("%d",&deb);

printf("donnez la fin");
scanf("%d",&fin);

printf("donnez la valeur à rechercher");
scanf("%d",&val);

for(p=t+deb;p<=t+fin;p++)
```

```
{
if(*p=val)
{
b=p-t+deb;
}
}

printf( "la position %d",b);

getch();
} // fin main
```

Exercice 4:

```
#include<stdio.h>
#include<conio.h>

main()
{
int t[10],aide,*p,*k,a;

p=t;

printf("donnez la dimension du tableau \n");
scanf("%d",&a);

for(p=t;p<t+a;p++)
{
printf("donner la valeuer de t[%d]=",p-t+1);
scanf("%d",p);
}

printf ("le tableau est :\n");

for(p=t;p<t+a;p++)
{
for(k=p+1;k<t+a;k++)
{
if(*p>*k)
{
aide=*p;
*p=*k;
*k=aide;
}
}
}

for(p=t;p<t+a;p++) printf("%d\t",*p);

getch();
```

```
} // fin main
```

Exercice 5

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int T[50][50]; /* tableau donné */
 int L, C;  /* dimensions        */
 int I, J;  /* indices courants  */
 long SOM;  /* somme des éléments - type long à cause */
           /* de la grandeur prévisible du résultat. */
 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &L );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &C );
 for (I=0; I<L; I++)
    for (J=0; J<C; J++)
        {
         printf("Elément[%d][%d] : ",I,J);
         scanf("%d", &T[I][J]);
        }
  /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<L; I++)
    {
     for (J=0; J<C; J++)
         printf("%7d", T[I][J]);
     printf("\n");
    }
  /* Calcul de la somme  */
 for (SOM=0, I=0; I<L; I++)
     for (J=0; J<C; J++)
         SOM += T[I][J];
  /* Edition du résultat */
 printf("Somme des éléments : %ld\n", SOM);
 return 0;
}
```

<u>Exercice 6</u>

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int T[50][50]; /* tableau donné    */
 int L, C;      /* dimensions       */
 int I, J;      /* indices courants */
 long SOM;  /* somme des éléments - type long à cause   */
           /* de la grandeur prévisible des résultats. */
```

```
 /* Saisie des données */
 printf("Nombre de lignes   (max.50) : ");
 scanf("%d", &L );
 printf("Nombre de colonnes (max.50) : ");
 scanf("%d", &C );
 for (I=0; I<L; I++)
   for (J=0; J<C; J++)
       {
        printf("Elément[%d][%d] : ",I,J);
        scanf("%d", &T[I][J]);
       }
 /* Affichage du tableau */
 printf("Tableau donné :\n");
 for (I=0; I<L; I++)
   {
    for (J=0; J<C; J++)
        printf("%7d", T[I][J]);
    printf("\n");
   }
 /* Calcul et affichage de la somme des lignes */
 for (I=0; I<L; I++)
    {
     for (SOM=0, J=0; J<C; J++)
           SOM += T[I][J];
     printf("Somme - ligne %d : %ld\n",I,SOM);
    }
 /* Calcul et affichage de la somme des colonnes */
 for (J=0; J<C; J++)
    {
     for (SOM=0, I=0; I<L; I++)
           SOM += T[I][J];
     printf("Somme - colonne %d : %ld\n",J,SOM);
    }
  return 0;
}
```

Exercice 7

```
#include <stdio.h>
main()
{
 /* Déclarations */
 int M[10][10]; /* tableau à 2 dimensions */
 int V[100];    /* tableau à 1 dimension  */
 int L, C;      /* dimensions       */
 int I, J;      /* indices courants */
 /* Saisie des données */
 printf("Nombre de lignes   (max.10) : ");
 scanf("%d", &L );
 printf("Nombre de colonnes (max.10) : ");
 scanf("%d", &C );
```

```
 for (I=0; I<L; I++)
   for (J=0; J<C; J++)
       {
        printf("Elément[%d][%d] : ",I,J);
        scanf("%d", &M[I][J]);
       }
 /* Affichage du tableau 2-dim */
 printf("Tableau donné :\n");
 for (I=0; I<L; I++)
   {
    for (J=0; J<C; J++)
         printf("%7d", M[I][J]);
    printf("\n");
   }
 /* Transfer des éléments ligne par ligne */
 for (I=0; I<L; I++)
     for (J=0; J<C; J++)
           V[I*C+J] = M[I][J];
  /* Affichage du tableau 1-dim */
 printf("Tableau résultat : ");
 for (I=0; I<L*C; I++)
    printf("%d ", V[I]);
 printf("\n");
 return 0;
}
```

TP 7 – Les chaînes de caractères

Exercice 1
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 2
Ecrire un programme qui lit 5 mots, séparés par des espaces et qui les affiche ensuite dans une ligne, mais dans l'ordre inverse. Les mots sont mémorisés dans 5 variables M1, ..., M5.

Exercice 3
Ecrire un programme qui lit une ligne de texte (ne dépassant pas 200 caractères) la mémorise dans une variable TXT et affiche ensuite :

a) la longueur L de la chaîne.
b) le nombre de **'e'** contenus dans le texte.
c) toute la phrase à rebours, sans changer le contenu de la variable TXT.
d) toute la phrase à rebours, après avoir inversé l'ordre des caractères dans TXT. Exemple :

voici une petite phrase !
! esarhp etitep enu iciov

Exercice 4
Ecrire un programme qui lit un texte TXT (de moins de 200 caractères) et qui enlève toutes les apparitions du caractère 'e' en tassant les éléments restants. Les modifications se feront dans la même variable TXT. Exemple :

Cette ligne contient quelques lettres e.
Ctt lign contint qulqus lttrs .

Exercice 5
Ecrire un programme qui demande l'introduction du nom et du prénom de l'utilisateur et qui affiche alors la longueur totale du nom sans compter les espaces. Employer la fonction **strlen**.

Exercice 6
Ecrire un programme qui lit deux chaînes de caractères CH1 et CH2, les compare lexicographiquement et affiche le résultat. Exemple :

Introduisez la première chaîne: ABC
Introduisez la deuxième chaîne: abc
"ABC" précède "abc"

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

TP7 – Corrections

Exercice 1

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 2

```
#include <stdio.h>
main()
{
 char M1[30], M2[30], M3[30], M4[30], M5[30];
 printf("Entrez 5 mots, séparés par des espaces :\n");
 scanf ("%s %s %s %s %s",  M1, M2, M3, M4, M5);
 printf("%s %s %s %s %s\n",M5, M4, M3, M2, M1);
  return 0;
}
```

Exercice3

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char TXT[201]; /* chaîne donnée       */
 int I,J;  /* indices courants         */
 int L;    /* longueur de la chaîne    */
 int C;    /* compteur des lettres 'e' */
 int AIDE; /* pour l'échange des caractères */

 /* Saisie des données */
 printf("Entrez une ligne de texte (max.200 caractères) :\n");
 gets(TXT); /* L'utilisation de scanf est impossible pour */
 /* lire une phrase contenant un nombre variable de mots. */

 /* a) Compter les caractères */
 /* La marque de fin de chaîne '\0' est */
 /* utilisée comme condition d'arrêt. */
 for (L=0; TXT[L]; L++)
    ;
 printf("Le texte est composé de %d caractères.\n",L);

 /* b) Compter les lettres 'e' dans le texte */
 C=0;
 for (I=0; TXT[I]; I++)
    if (TXT[I]=='e') C++;
 printf("Le texte contient %d lettres \'e\'.\n",C);

 /* c) Afficher la phrase à l'envers */
 for (I=L-1; I>=0; I--)
    putchar(TXT[I]);  /* ou printf("%c",TXT[I]); */
 putchar('\n');        /* ou printf("\n"); */

 /* d) Inverser l'ordre des caractères */
 for (I=0,J=L-1 ; I<J ; I++,J--)
   {
    AIDE=TXT[I];
    TXT[I]=TXT[J];
    TXT[J]=AIDE;
   }
 puts(TXT);  /* ou printf("%s\n",TXT); */
  return 0;
}
```

Exercice4

```
#include <stdio.h>
main()
{
 /* Déclarations */
 char TXT[201]; /* chaîne donnée    */
 int  I,J;      /* indices courants */

 /* Saisie des données */
 printf("Entrez une ligne de texte (max.200 caractères) :\n");
 gets(TXT);
 /* Eliminer les lettres 'e' et comprimer :  */
 /* Copier les caractères de I vers J et incrémenter J */
 /* seulement pour les caractères différents de 'e'.   */
 for (J=0,I=0 ; TXT[I] ; I++)
```

```
   {
    TXT[J] = TXT[I];
    if (TXT[I] != 'e') J++;
   }
 /* Terminer la chaîne !! */
 TXT[J]='\0';
  /* Edition du résultat */
 puts(TXT);
  return 0;
}
```

Exercice5

```
#include <stdio.h>
#include <string.h>
 main()
{
  char NOM[40], PRENOM[40];
  printf("Introduisez votre nom et votre prénom: \n");
  scanf("%s %s", NOM, PRENOM);
  printf("\nBonjour %s %s !\n", NOM, PRENOM);
  printf("Votre nom est composé de %d lettres.\n",
                          strlen(NOM) + strlen(PRENOM));
 /* ou bien
  printf("Votre nom est composé de %d lettres.\n",
                          strlen(strcat(NOM,PRENOM)));
 */
 return 0;
}
```

Exercice 6

```
#include <stdlib.h>
#include <string.h>
 main()
{
 /* Déclarations */
 char CH1[200], CH2[200]; /* chaînes entrées */
 int RES;  /* résultat de la fonction strcmp */

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

 /* Comparaison et affichage du résultat */
 RES = strcmp(CH1,CH2);
 if (RES<0)
    printf("\"%s\" précède \"%s\"\n",CH1 ,CH2);
 else if (RES>0)
    printf("\"%s\" précède \"%s\"\n",CH2 ,CH1);
 else
    printf("\"%s\" est égal à \"%s\"\n",CH1, CH2);
  return 0;
}
```

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;
}
```

TP 8 – Les fonctions

Exercice 1
Écrire un programme qui implémente une fonction qui retourne la somme des entiers pairs inférieurs ou égaux à un entier n donné (passé en argument). Le programme principal doit afficher le résultat final.

Exercice 2
Écrire un programme qui implémente une fonction qui lit les données relatives à un étudiant appartenant à une institution universitaire telles que son nom, son prénom, son CNE ainsi que les notes obtenues dans 12 modules. Le programme principal doit afficher toutes ces données lues par la fonction ainsi que la moyenne des notes.

Exercice 3
Écrire une fonction qui ne renvoie aucune valeur et qui détermine la valeur maximale et la valeur minimale d'un tableau d'entiers (à un indice) de taille quelconque. Il faudra donc prévoir 4 arguments : le tableau, sa dimension, le maximum et le minimum. Écrire un programme d'essai.

Exercice 4
Écrire une fonction permettant de trier par ordre croissant les valeurs entières d'un tableau de taille quelconque (transmise en argument). Le tri pourra se faire par réarrangement des valeurs au sein du tableau lui-même.

Exercice 5
Ecrire la fonction NCHIFFRES du type **int** qui obtient une valeur entière N (positive ou négative) du type **long** comme paramètre et qui fournit le nombre de chiffres de N comme résultat. Ecrire un programme qui teste la fonction NCHIFFRES.

Exemple :
Introduire un nombre entier : 6457392
Le nombre 6457392 contient 7 chiffres.

Exercice 6
Écrire une fonction calculant la somme de deux matrices dont les éléments sont de type double. Les adresses des trois matrices et leurs dimensions (communes) seront transmises en argument.

Exercice 7
Soit T un tableau de N éléments de type entier. Écrire un programme qui implémente une fonction **fct3 ( )** qui calcule le nombre d'éléments pairs **np**, le nombre d'éléments impairs **nimp** et le nombre d'éléments égaux à 0 **nbz**, du tableau T. Le programme principal doit afficher le résultat final.

Exercice 8
Ecrire un programme qui implémente une fonction qui effectue la multiplication de deux matrices d'entiers A de dimensions (N, M) et B de dimensions (M, P). Le résultat de la multiplication sera mémorisé dans une troisième matrice C qui sera ensuite affichée.

TP8 – Corrections

Exercice 1

```
# include <stdio.h>

int fct1 (int n) {

int i, somme = 0;

for(i=0 ; i<=n ; i++) {
        if (i % 2 == 0) somme += i ; }

return somme;

}

main ( ) {

int Som, N;

printf (“Entrer N : ”) ;
scanf(“%d”,&N);

Som = fct1 (N); // Appel de la fonction

printf (“La somme est %d : \n”, Som) ;

getch();

}
```

Exercice 2

```
#include <stdio.h>
#include <string.h>


void fct2 (char *nom ,char *pnom, int *age , float t[50], int n ){

    int i;

    printf("lire nom : ");
    gets(nom);

    printf("lire prenom : ");
    gets(pnom);

    printf("lire age : ");
    scanf("%d",age);

    for (i=0; i<n; i++){
```

```
            printf("t[%d] = ",i);

scanf("%f",&t[i]);
            }

}

main ( )
{
  int i, N, Age;
  float Moy = 0.0, T[50];
  char Nom[50], Pnom[50];


   printf("lire dim : ");
   scanf("%d",&N);
   printf("dim = %d\n", N);

   fct2 (Nom,Pnom,&Age,T,N); // Appel de la fonction

   printf("nom : %s\n",Nom);

  printf("prenom : %s\n",Pnom);

   printf("age = %d\n",Age);

   printf("\n");

   for (i=0; i<N; i++){
          printf("T[%d] = %f \n",i,T[i]);
          Moy += T[i];
  }

          printf("La moyenne est  %f \n",Moy/N);

getch();

}
```

Exercice 3

```
void maxmin (int t[], int n, int * admax, int * admin)
{ int i, max, min ;
max = t[0] ;
min = t[0] ;
for (i=1 ; i<n ; i++)
{ if (t[i] > max) max = t[i] ;
if (t[i] < min) min = t[i] ;
}
*admax = max ;
*admin = min ;
```

```
}
#include <stdio.h>
main()
{ void maxmin (int [], int, int *, int *) ;
int t[8] = { 2, 5, 7, 2, 9, 3, 9, 4} ;
int max, min ;
maxmin (t, 8, &max, &min) ;
printf ("valeur maxi : %d\n", max) ;
printf ("valeur mini : %d\n", min) ;
}
```

Exercice4

```
void tri (unsigned char c[] , int nc)
{ int i,j ;
char ct ;
for (i=0 ; i<nc-1 ; i++)
for (j=i+1 ; j<nc ; j++)
if ( c[i]>c[j] )
{ ct = c[i] ;
c[i] = c[j] ;
c[j] = ct ;
}
}
```

Exercice5

```
#include <stdio.h>
main()
{
 /* Prototypes des fonctions appelées */
 int NCHIFFRES (long N);
 /* Variables locales */
 long A;
 /* Traitements */
 printf("Introduire un nombre entier : ");
 scanf("%ld", &A);
 printf("Le nombre %ld a %d chiffres.\n",A ,NCHIFFRES (A));
 return 0;
}


int NCHIFFRES (long N)
{
  /* Comme N est transmis par valeur, N peut être */
 /* modifié à l'intérieur de la fonction. */
 int I;
 /* Conversion du signe si N est négatif */
 if (N<0)
     N *= -1;
  /* Compter les chiffres */
 for (I=1; N>10; I++)
     N /= 10;
```

```
 return I;
}
```

Exercice6

```
void sommat (double * a, double * b, double * c, int n, int p)
{ int i ;
for (i=0 ; i<n*p ; i++)
*(c+i) = *(a+i) + *(b+i) ;
}
```

Exercice7

```
#include <stdio.h>

void fct3 (int t[50], int n, int *np, int *nimp, int *nbz) {

 int i;
for(i=0; i<n; i++){
        if(t[i]==0) *nbz=*nbz+1;
        else if(t[i]%2==0) *np=*np+1;
        else *nimp=*nimp+1;
}
}

main() {

 int N, i;
int NP=0,NIMP=0,NBZ=0;
int T[50];

("Entrer la dimension de T : ");
scanf("%d",&N);

for(i=0; i<N ; i++){
        printf("Entrer T[%d] : ", i);
        scanf("%d", &T[i]);
}

fct3 (T, N, &NP, &NIMP, &NBZ); // Appel de la fonction

printf("Le nombre de zéros : %d \n",NBZ);
printf("Le nombre d'éléments pairs : %d \n",NP);
printf("Le nombre d'éléments impairs : %d \n",NIMP);

getch();

}
```

Exercice8

```
#include <stdio.h>
```

```
// Prototype de la fonction prod_mat
void prod_mat (int a[50][50], int n, int m, int b[50][50], int p, int c[50][50]) ;

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");
    }

 // Appel de la fonction prod_mat
 prod_mat (A, N, M, B, P, C);

  /* 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");
    }

    getch();

}

void prod_mat (int a[50][50], int n, int m, int b[50][50], int p, int c[50][50]) {

int i, j, k ;

 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];
         }

}
```

TP 9 – Les structures

Exercice 1

Écrire un programme qui définit trois structures Point, Cercle et Rectangle. Le programme doit lire et afficher les champs respectifs des variables de type structure Point, Cercle et Rectangle (par exemple : 2 points p1 et p2, 2 cercles c1 et c2, 2 rectangles R1 et R2).

Exercice 2

Définir une structure **date** contenant trois champs de type entier pour identifier le jour, le mois et l'année. Initialiser une variable de type structure date. Afficher cette structure à l'aide de la variable et à l'aide d'un pointeur.

Exercice 3

On reprend la structure date de l'exercice 2.

1) Définir un tableau de structures date.
2) Définir un pointeur sur ce tableau.
3) Initialiser ce tableau.
4) Afficher le contenu du tableau.

Exercice 4

Définir une structure de données Heure permettant de représenter une heure au format hh/mm/ss, puis écrire les fonctions suivantes :

1) conversion d'un élément de type Heure en nombre de secondes (entier).
2) conversion d'un nombre de secondes (entier) en un élément de type Heure
3) addition de deux éléments de type Heure

Exercice 5

Définir une structure Etudiant contenant les champs CNE (entier long), nom (chaine), prenom (chaine), date (structure), adresse (structure), email (chaine), phone (chaine) et site_web (chaine). Ecrire un programme qui lit et affiche les données relatives à 100 étudiants en utilisant un tableau de structure appelé Contact.

Exercice 6

Écrire un programme qui lit au clavier des informations dans un tableau de structures du type point défini comme suit : struct point { int num ; float x ; float y ; } ;

Le nombre d'éléments du tableau sera fixé par une instruction #define et le programme doit afficher l'ensemble des informations précédentes.

Réaliser le même traitement en utilisant cette fois-ci une fonction pour la lecture des informations et une autre pour l'affichage.

Exercice 7

On considère la structure suivante : typedef struct {

```
char *nom;
int age ;
float taille ;
}Personne ;
```

Donner le code des fonctions suivantes :
Personne * Creer_personne (char *nom, int age, float taille).
void Affiche_personne (Personne *w).
void Detruire_personne (Personne *w).

TP9 – Corrections

Exercice 1

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

struct Point {
        float x;
        float y ;
} ;

struct Cercle {
        float xc;
        float yc ;
        float rayon;
} ;

struct Rectangle {
        float L;
        float l ;
} ;


int main ( )
{


struct Point p1, p2 ; // déclaration de 2 variables de type Point
struct Cercle c1, c2 ; // déclaration de 2 variables de type Cercle
struct Rectangle R1, R2 ; // déclaration de 2 variables de type Rectangle


printf ( "Entrer les champs du point p1 : \n") ;
scanf ( "%f %f",&p1.x,&p1.y) ;
printf ( "Afficher les champs du point p1 : %f %f\n", p1.x, p1.y) ;

printf ( "\n") ;
printf ( "Entrer les champs du point 2 : \n") ;
scanf ( "%f %f",&p2.x,&p2.y) ;
printf ( "Afficher les champs du point p2 : %f %f\n", p2.x, p2.y) ;

printf ( "\n") ;
printf ( "Entrer les champs du cercle c1 : \n") ;
scanf ( "%f %f %f",&c1.xc,&c1.yc,&c1.rayon) ;
printf ( "Afficher les champs du cercle c1 : %f %f %f\n", c1.xc, c1.yc,c1.rayon) ;

printf ( "\n") ;
printf ( "Entrer les champs du cercle c2 : \n") ;
scanf ( "%f %f %f",&c2.xc,&c2.yc,&c2.rayon) ;
printf ( "Afficher les champs du cercle c1 : %f %f %f\n", c2.xc, c2.yc,c2.rayon) ;
```

```
printf ( "\n") ;
printf ( "Entrer les champs du rectangle R1 : \n") ;
scanf ( "%f %f",&R1.L,&R1.l) ;
printf ( "Afficher les champs du rectangle R1 : %f %f\n", R1.L, R1.l) ;

printf ( "\n") ;
printf ( "Entrer les champs du rectangle R2 : \n") ;
scanf ( "%f %f",&R2.L,&R2.l) ;
printf ( "Afficher les champs du rectangle R2 : %f %f\n", R2.L, R2.l) ;



getch();

return 0 ;
}
```

Exercice 2

```
#include <stdio.h>

struct date      /* déclaration du modèle de structure date */
{
  int jour;
  int mois;
  int annee;
};

int main ()
{
  struct date dat, *ptdate = &dat;

  /* saisie de la date */
  printf (" Entrez une date ( jj mm aaaa ) :");
  scanf ("%d %d %d", &dat.jour, &dat.mois, &dat.annee);
  /* impression de la date */
  printf (" La date est : %d\t%d\t%d\n", dat.jour, dat.mois, dat.annee);
  /* impression de la date avec le pointeur */
  printf ("La date est : %d\t%d\t%d\n", ptdate->jour, ptdate->mois, ptdate->annee);
  return 0;
}
```

Exercice 3

```
#include <stdio.h>

/* déclaration du modèle de structure date */
struct date{ int jour, mois, annee;};

int main ()
{
```

```
 struct date dat[5], *ptdate=dat;
 int i = 0;

 /* remplissage du tableau */
 printf ("Entrez 5 dates au format ( jj mm aaaa )\n");
 while (i < 5){
    scanf ("%d %d %d", &dat[i].jour, &dat[i].mois, &dat[i].annee);
    i++;
 }

 /* impression du tableau */
 for (i = 0; i < 5; i++, ptdate++) {

    /* sans pointeur */
    printf ("Date numero      %d:%d\t%d\t%d\n", i + 1, dat[i].jour, dat[i].mois, dat[i].annee);

   /* avec pointeur */
printf ("Date numero%d:%d\t%d\t%d\n", i + 1, ptdate->jour, ptdate->mois, ptdate->annee
);
  }

 return 0;
}
```

Exercice4

```
typedef struct {
int hh;
int mm;
int ss;
} Heure;

int HeureEnSecondes(Heure h) {
return (h.hh*3600 + h.mm*60 + h.ss);
}

Heure SecondesEnHeure(int sec) {
Heure h;
h.hh = sec/3600;
sec = sec%3600;
h.mm = sec/60;
sec = sec%60;
h.ss = sec;
return h;
}

Heure AddHeures(Heure h1, Heure h2) {
Heure h3;
return (SecondesEnHeure(HeureEnSecondes(h1)+HeureEnSecondes(h2)));
}
```

Autre solution :

```
Heure AddHeures2(Heure h1, Heure h2) {

Heure h3;
h3.hh=h1.hh+h2.hh;
h3.mm=h1.mm+h2.mm;
h3.ss=h1.ss+h2.ss;

if (h3.ss > 59) {
h3.ss -= 60;
h3.mm += 1;
}

if (h3.mm > 59) {
h3.mm -= 60;
h3.hh +=1;
}

return h3;

}
```

Exercice5

```
struct date {
                                        int jour ;
                                        char mois [10] ;
                                         int annee ;
} ;
struct adresse {
                                        int no_rue ;
                                        char rue [100] ;
                                        int code_postal ;
                                        char ville [30] ;
                                        char pays [30] ;
} ;

struct Etudiant {
                                        long CNE;
                                        char nom [30] ;
                                        char prenom [30] ;
                                        struct date date_naiss ;
                                        struct adresse ads ;
                                        char email [30] ;
                                        char phone [14] ;
                                        char site_web [30] ;
                                    } ;
struct Etudiant Contact [100] ;
```

Exercice6

```
#include <stdio.h>
#define NPOINTS 5

main()
{ struct point { int num ;
float x ;
float y ;
} ;
struct point courbe[NPOINTS] ;
int i ;
for (i=0 ; i<NPOINTS ; i++)
{ printf ("numéro : ") ; scanf ("%d", &courbe[i].num) ;
printf ("x : ") ; scanf ("%f", &courbe[i].x) ;
printf ("y : ") ; scanf ("%f", &courbe[i].y) ;
}
printf (" **** structure fournie ****\n") ;
for (i=0 ; i<NPOINTS ; i++)
printf ("numéro : %d x : %f y : %f\n",
courbe[i].num, courbe[i].x, courbe[i].y) ;
}

// Traitement avec les fonctions

#include <stdio.h>
#define NPOINTS 5
struct point { int num ;
float x ;
float y ;
} ;
void lit (struct point []) ; /* ou void lit (struct point *) */
void affiche (struct point []) ; /* ou void lit (struct point *) */
main()
{
struct point courbe[NPOINTS] ;
lit (courbe) ;
affiche (courbe) ;
}
void lit (struct point courbe []) /* ou void lit (struct point * courbe) */
{
int i ;
for (i=0 ; i<NPOINTS ; i++)
{ printf ("numéro : ") ; scanf ("%d", &courbe[i].num) ;
printf ("x : ") ; scanf ("%f", &courbe[i].x) ;
printf ("y : ") ; scanf ("%f", &courbe[i].y) ;
}
}
void affiche (struct point courbe []) /* ou void affiche
(struct point * courbe) */
```

```
{
int i ;
printf (" **** structure fournie ****\n") ;
for (i=0 ; i<NPOINTS ; i++)
printf ("%d %f %f\n", courbe[i].num, courbe[i].x, courbe[i].y) ;
}
```

Exercice 7

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>


typedef struct {
            char *nom;
            int age ;
            float taille ;
}Personne ;

Personne * Creer_personne (char *nom , int age , float taille ){

        Personne * w;

        w = (Personne *)malloc(sizeof(Personne));

        w -> nom = nom;
         w -> age = age;
          w -> taille = taille;

          return w;

        }

        void Afficher_personne ( Personne *w){

            printf("%s\n",w -> nom);
            printf("%d\n",w -> age);
            printf("%f\n",w -> taille);

            }

void Detruire_personne ( Personne *w){

            free(w);

            }

int main ( int argc , char * argv [ ] )
{
Personne *mounir = Creer_personne ( "Mounir " , 32 , 175 ) ;
```

```
Personne *badr = Creer_personne ( " Badr " , 20 , 183 ) ;

printf ( "Mounir est à l’adresse mémoire %p : \n" , mounir ) ;
Afficher_personne (mounir ) ;

printf ("\n");

printf ( "Badr est à l ’adresse mémoire %p : \n" , badr ) ;
Afficher_personne ( badr ) ;

printf ("\n");

mounir->age += 10 ;
Afficher_personne (mounir ) ;

printf ("\n");

badr->age += 20;
Afficher_personne ( badr ) ;

printf ("\n");

Detruire_personne (mounir ) ;
Detruire_personne ( badr ) ;

getch();

return 0 ;
}
```

TP 10 – Les fichiers

Exercice 1

Définir une structure Etudiant contenant les champs CNE (entier long), nom (chaine), prenom (chaine), date (structure), adresse (structure), email (chaine), phone (chaine) et site_web (chaine). Ecrire un programme qui enregistre dans un fichier « Annuaire » les données relatives à 100 étudiants en utilisant un tableau de structure appelé Contact. On utilisera la fonction **fprintf ( )** pour la sauvegarde des données.

Exercice 2

Ecrire un programme qui implémente une fonction qui effectue la transposition t_A d'une matrice A d'entiers de dimensions L et C en une matrice de dimensions C et L qui sera ensuite **sauvegardée dans fichier**. La matrice A sera transposée par permutation de ses éléments (Dans le programme, **on n'utilisera qu'une seule matrice** : la matrice A). On utilisera la fonction **fprintf ( )** pour la sauvegarde des données.

Exercice 3

Ecrire un programme qui implémente une fonction qui effectue la multiplication de deux matrices d'entiers A de dimensions (M, N) et B de dimensions (N, P). Le résultat de la multiplication sera mémorisé dans une troisième matrice C et **sauvegardé dans un fichier**. On utilisera la fonction **fprintf ( )** pour la sauvegarde des données.

TP10 – Corrections

Exercice5

Même chose que l'exercice 5 de la série 9. On rajoute juste l'écriture dans le fichier « Annuaire ».

```
FILE *fp ;
….
fp = fopen (« Annuaire », « a ») ;
…..
Dans boucle for : fprintf (fp, « …… », …..) ;
…..
fclose (fp) ;
….


struct date {
                                        int jour ;
                                        char mois [10] ;
                                         int annee ;
} ;
struct adresse {
                                        int no_rue ;
                                        char rue [100] ;
                                        int code_postal ;
                                        char ville [30] ;
                                        char pays [30] ;
} ;

struct Etudiant {
                                        long CNE;
                                        char nom [30] ;
                                        char prenom [30] ;
                                        struct date date_naiss ;
                                        struct adresse ads ;
                                        char email [30] ;
                                        char phone [14] ;
                                        char site_web [30] ;
                                    } ;
```

struct Etudiant Contact [100] ;

Exercice6

Même chose que l'exercice 6 de la série 8. On rajoute juste l'écriture dans un fichier.

```
…..
FILE *fp ;
….
fp = fopen (« fichier », « a ») ;
…..
fprintf (fp, « …… », …..) ;
…..
fclose (fp) ;
….
```

Exercice7

Même chose que l'exercice 7 de la série 8. On rajoute juste l'écriture dans un fichier.

```
…..
FILE *fp ;
….
fp = fopen (« fichier », « a ») ;
…..
fprintf (fp, « …… », …..) ;
…..
fclose (fp) ;
….
```

TP 11 – Structures et fichiers

Problème de synthèse

On souhaite gérer une base de données d'inscriptions pour l'organisation d'un congrès qui dure une journée. Les organisateurs proposent aux participants de s'inscrire pour des repas, ainsi que pour l'hébergement en hôtel. Un participant peut s'inscrire indépendamment aux 2 repas proposés : déjeuner (150 dhs) et/ou dîner (350 dhs) ou aucun. Il n'est pas obligé de prendre un hôtel. S'il en prend un, il peut choisir parmi 2 types d'hôtels différents : 2 étoiles (250 dhs) ou 3 étoiles (400 dhs). Un participant peut venir accompagné de son conjoint. Dans ce cas, la réservation d'hôtel est identique mais lorsqu'un repas est sélectionné alors il faut en compter 2.

1) Créer un nouveau type Participant qui inclut son nom (chaîne de caractères : tableau de 20 caractères), son prénom (chaîne de caractères : tableau de 30 caractères), ainsi que toutes les autres informations nécessaires à son inscription selon les critères définis ci-dessus. On privilégiera une structure contenant un nombre minimal de champs.

2) Créer un nouveau type TabPart qui est un tableau de 100 éléments de type Participant.

3) Créer une fonction Nb2Etoiles qui, pour un argument de type TabPart donné, affiche le nom et le prénom des personnes qui ont choisi de réserver un hôtel 2 étoiles. On pourra utiliser la fonction printf avec pour format %s pour l'affichage des chaînes de caractères nom et prénom.

4) Créer une fonction NbDej qui, pour un argument de type TabPart donné, retourne le nombre de déjeuners à prévoir.

5) Créer une fonction Montant qui calcule, pour un Participant donné en argument, le montant de sa facture.

6) Sauvegarder toutes les informations relatives à chaque participant y compris le montant de sa facture dans un fichier.

TP11 – Corrections

```
typedef struct
{
char nom [20];
char prenom [30];
int dejeuner;   // 0=non, 1=oui
int diner;   // 0=non, 1=oui
int hotel;   // 1=pas d'hotel, 2=2etoiles, 3=3etoiles
int seul;   // 0=non, 1=oui
} Participant;

typedef Participant TabPart [100];

void Nb2Etoiles(TabPart tab)
{
int i;
for (i=0; i<100; i++)
{
    if (tab[i].hotel==2) {
        printf("nom: %s, prenom:%s\n", tab[i].nom,
        tab[i].prenom);
    }
}
}

int NbDej(TabPart tab)
{
int nb=0, i;
for (i=0; i<100; i++)
 {
   if (tab[i].dejeuner==1) {
    nb+=1;
   if (tab[i].seul==0)
   nb+=1;
  }
}

return nb;
}

int Montant(Participant p)
{
int total=0;

if (p.dejeuner==1)
total += 150;
```

```
if (p.diner==1)
total += 350;

if (p.seul==0)
total *= 2;

if (p.hotel==2)
total += 250;

if (p.hotel==3)
total += 400;

return total;
}
```