LANGAGE C

Exercices corrigés 1

TP1

Exercice 1 :

Ecrire un programme qui lit un caractère au clavier et affiche le caractère ainsi que son code numérique en employant getchar et printf,

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

        printf("introduire un caractère suivi de 'Enter'\n");
 C = getchar();
printf("Le caractère %c a le code ASCII %d\n", C, C);
        return 0;
}
```

Exercice 2 :

Ecrire un programme qui calcule et affiche la distance DIST (type double) entre deux points A et B du plan dont les coordonnées (XA, YA) et (XB, YB) sont entrées au clavier comme entiers.

```
#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 3 :

Ecrivez un programme qui calcule les solutions réelles d'une équation du second degré ax2+bx+c = 0 en discutant la formule.

Utilisez une variable d'aide D pour la valeur du discriminant b2-4ac et décidez à l'aide de D, si l'équation a une, deux ou aucune solution réelle. Utilisez des variables du type int pour A, B et C.

Considérez aussi les cas où l'utilisateur entre des valeurs nulles pour A; pour A et B; pour A, B et C. Affichez les résultats et les messages nécessaires sur l'écran.
Modifier le programme afin de considérer le cas des solutions complexes.

```
#include <stdio.h>
#include <math.h>
 main()
{
        /* Calcul des solutions réelles et complexes d'une équation du second degré */
        int A, B, C;
        double D; /* Discriminant */

        printf("Calcul des solutions réelles et complexes 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("%i %i %i", &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("Les solutions complexes de cette équation sont les suivantes :\n");
                printf("x1 = %.4f + i%.4f\n", (double)(-B),(double)(sqrt(-D)/(2*A))) ;
                printf("x2 = %.4f + i%.4f\n", (double)(-B),(double)(-sqrt(-D)/(2*A))) ;
        }

        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", (double)(-B+sqrt(D))/(2*A));
                printf(" x2 = %.4f\n", (double)(-B-sqrt(D))/(2*A));
        }
        return 0;
}
```

TP2 :

Exercice 1 :

Calculez la somme des N premiers termes de la série harmonique : $1 + 1/2 + 1/3 + ... + 1/N$

```
#include <stdio.h>
main()
{
        int N;     /* nombre de termes à calculer */
```

```
        int I;     /* compteur pour la boucle */
        float SOM; /* Type float à cause de la précision du résultat.   */

        do
        {
                printf ("Nombre de termes: ");
                scanf ("%d", &N);
        }while (N<1);
        for (SOM=0.0, I=1 ; I<=N ; I++)
                SOM += (float)1/I;
        printf("La somme des %d premiers termes est %f \n", N, SOM);
        return 0;
}
```

Exercice 2 :

Affichez un triangle isocèle formé d'étoiles de N lignes (N est fourni au clavier).

```
#include <stdio.h>
main()
{
        int LIG;   /* nombre de lignes        */
        int L;     /* compteur des lignes     */
        int ESP;   /* nombre d'espaces        */
        int I;     /* compteur des caractères */

        do
        {
                printf("Nombres de lignes : ");
                scanf("%d", &LIG);
        }while (LIG<1 || LIG>20);

        for (L=0 ; L<LIG ; L++)
        {
                ESP = LIG-L-1;
                for (I=0 ; I<ESP ; I++)
                        putchar(' ');
                for (I=0 ; I<2*L+1  ; I++)
                        putchar('*');
                putchar('\n');
        }
        return 0;
}
```

Exercice 3 :

a) Calculez la racine carrée X d'un nombre réel positif A par approximations successives en utilisant la relation de récurrence suivante:

$$XJ+1 = (XJ + A/XJ) / 2 \qquad X1 = A$$

La précision du calcul J est à entrer par l'utilisateur.

b) Assurez-vous lors de l'introduction des données que la valeur pour A est un réel positif et que J est un entier naturel positif, plus petit que 50.

c) Affichez lors du calcul toutes les approximations calculées :

La 1ère approximation de la racine carrée de ... est ...
La 2e approximation de la racine carrée de ... est ...
La 3e approximation de la racine carrée de ... est ...
. . .

```
#include <stdio.h>
main()
{
        double A;   /* donnée */
        double X;   /* approximation de la racine carrée de A */
        int N;      /* degré/précision de l'approximation     */
        int J;      /* degré de l'approximation courante      */

        do
        {
                printf("Entrer le réel positif A : ");
                scanf("%lf", &A);
        }while(A<0);
        do
        {
                printf("Entrer le degré de l'approximation : ");
                scanf("%d", &N);
   }
 while(N<=0 || N>=50);

 for(X=A, J=1 ; J<=N ; J++)
    {
     X = (X + A/X) / 2;
     printf("La %2d%s approximation de la racine carrée"
          " de %.2f est %.2f\n", J, (J==1)?"ère":"e", A, X);
    }
  return 0;
}
```

Exercice 4

Affiche la table des produits pour N variant de 1 à 10 :

```
 X*Y  I   0   1   2   3   4   5   6   7   8   9  10
-------------------------------------------------
  0   I   0   0   0   0   0   0   0   0   0   0   0
  1   I   0   1   2   3   4   5   6   7   8   9  10
  2   I   0   2   4   6   8  10  12  14  16  18  20
  3   I   0   3   6   9  12  15  18  21  24  27  30
  4   I   0   4   8  12  16  20  24  28  32  36  40
  5   I   0   5  10  15  20  25  30  35  40  45  50
  6   I   0   6  12  18  24  30  36  42  48  54  60
  7   I   0   7  14  21  28  35  42  49  56  63  70
  8   I   0   8  16  24  32  40  48  56  64  72  80
  9   I   0   9  18  27  36  45  54  63  72  81  90
 10   I   0  10  20  30  40  50  60  70  80  90 100
```

```
#include <stdio.h>
main()
{
 const int MAX = 10; /* nombre de lignes et de colonnes */
```

```
 int I;            /* compteur des lignes   */
 int J;            /* compteur des colonnes */

 /* Affichage de l'en-tête */
 printf(" X*Y I");
 for (J=0 ; J<=MAX ; J++)
       printf("%4d", J);
 printf("\n");
 printf("------");
 for (J=0 ; J<=MAX ; J++)
       printf("----");
 printf("\n");

 /* Affichage du tableau */
 for (I=0 ; I<=MAX ; I++)
     {
      printf("%3d  I", I);
      for (J=0 ; J<=MAX ; J++)
          printf("%4d", I*J);
      printf("\n");
     }
  return 0;
}
```

TP3 :

Exercice 1

Ecrire un programme qui saisit la dimension N d'un tableau de *int* (le tableau est initialement définit avec une taille maximum MAX que N ne doit pas excéder) remplit le tableau par des valeurs entrées au clavier et l'affiche.

Le programme doit ensuite effacer toutes les occurrences de la valeur 0 dans le tableau, tasser les éléments restants et afficher le tableau ainsi modifier.

Pour cela écrire les fonctions suivantes :

*void SaisirTableau (int *Tab, int N) ;*
*void AfficherTableau(int *Tab, int N) ;*
*int TasserTableau(int *Tab , int N) ;*

```
#include <stdio.h>
#define MAX 50
void SaisirTableau(int *, int ) ;
void AfficherTableau(int *, int) ;
int TasserTableau(int *, int) ;
main()
{
 /* Déclarations */
          int T[MAX]; /* tableau donné    */
          int N,M;     /* dimension        */

 /* Saisie de la dimension */
do
{
          printf("Dimension du tableau (max.%d) : ",MAX);
          scanf("%d", &N );
```

```
}while(N>MAX) ;

 /* Saisie des données */
        SaisirTableau(T,N) ;


 /* Affichage du tableau */
        AfficherTableau(T,N) ;

/*Tasser les elements du tableau */
        M = TasserTableau(T,N) ;

/* Edition des résultats */

        AfficherTableau(T ,M) ;
}

void SaisirTableau(int *Tab, int N)
{
        int i ;

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

void AfficherTableau(int *Tab, int N)
{
        int i ;
        printf("Tableau donné : \n");
        for (i=0; i<N; i++)
                printf("%d ", Tab[i]);
        printf("\n");
}


int TasserTableau(int * Tab, int N)
{
        int i,j ;

        /* 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++)
        {
                Tab[j] = Tab[i] ;
                if (Tab[i])
                        j++ ;
        }

 /* La nouvelle dimension du tableau est retournée */
        return j ;
}
```

Exercice 2

Ecrire un programme qui saisit la dimension N d'un tableau de *int* remplit le tableau par des valeurs entrées au clavier et l'affiche.

Copier ensuite toutes les composantes strictement positives dans un deuxième tableau Tpos et toutes les valeurs strictement négatives dans un tableau Tneg. Afficher Tpos et Tneg.

Ecrire la fonction suivante :

*int TrierTableau(int *, int *, int *,int)*

```
#include <stdio.h>
#define MAX 50
main()
{
 /* Déclarations */
 /* Les tableaux et leurs dimensions */
        int T[MAX], TPOS[MAX], TNEG[MAX];
        int N,M, Npos,     NNEG;
        int I; /* indice courant */


 /* Saisie de la dimension */
 do
{
        printf("Dimension du tableau (max.%d) : ",MAX);
        scanf("%d", &N );
}while(N>MAX) ;



 /* Saisie des données */
        SaisirTableau(T,N) ;

 /* Affichage du tableau */
        AfficherTableau(T,N) ;

/*Tasser les elements du tableau */
        M = TasserTableau(T,N) ;

/* Trier le tableau */
         Npos = TrierTableau(T,TPOS,TNEG,M) ;

/* Edition des resultats */
        printf("Elements positifs : \n") ;
        AfficherTableau(TPOS,Npos) ;
        printf("Elements négatifs : \n") ;
        AfficherTableau(TNEG,N-Npos) ;
}
int TrierTableau(int *T, int *TPOS, int *TNEG, int N)
{


        int npos=0, nneg=0;
        int i ;

 /* Transfert 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++;
                }
        }
        return npos ;
}
```

Exercice 3

Ecrire un programme qui calcul le produit scalaire de deux vecteurs d'entiers U et V de même dimension.

Ecrire la fonction suivante :

*long ProduitScalaire(int *U,int *V, int dimension)*

```
#include <stdio.h>
#define MAX 50
long ProduitScalaire(int *,int *, int) ;
main()
{
 /* Déclarations */
        int U[MAX], V[MAX];     /* tableaux donnés      */
        int N;                  /* dimension            */
        int I;                  /* indice courant       */
        long PS;                /* produit scalaire     */

 /* Saisie des données */
do
{
        printf("Dimension du tableau (max.%d) : ",MAX);
        scanf("%d", &N );
}while(N>MAX) ;
        printf("** Premier tableau **\n");
        for (I=0; I<N; I++)
        {
                printf("Elément %d : ", I);
                scanf("%d", &U[I]);
        }
        printf("** Deuxième tableau **\n");
        for (I=0; I<N; I++)
        {
                printf("Elément %d : ", I);
               scanf("%d", &V[I]);
        }
```

```
 /* Calcul du produit scalaire */

        PS = ProduitScalaire(U,V,N) ;

/* Edition du résultat */
        printf("Produit scalaire : %ld\n", PS);
}

long ProduitScalaire(int *U, int *V,int N)
{
        long ps ;
        int i ;

        for (ps=0, i=0; i<N; i++)
        ps += (long)U[i]*V[i];

        Return ps ;
}
```

TP 4 :

Exercice 1 :

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.
Pour cela on écrira les fonctions suivantes :

void RemplirTableau(void)
void AfficherTableau(void)

```
#include <stdio.h>
void RemplirTableau(void) ;
void AfficherTableau(void) ;
 int T[50][50];     /* tableau donné                          */
 int L, C;          /* dimensions                             */

main()
{
 /* Déclarations */
        long SOM;       /* somme des éléments - type long          */


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

        RemplirTableau() ;

        AfficherTableau() ;

  /* 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;
}

void RemplirTableau(void)
{
        int i,j ;

        for (i=0; i<L; i++)
        for (j=0; j<C; j++)
        {
                printf("Elément[%d][%d] : ",i,j);
                scanf("%d", &T[i][j]);
        }
}

void AfficherTableau(void)
{
        int i,j ;

        printf("Tableau donné :\n");
        for (i=0; i<L; i++)
        {
                for (j=0; j<C; j++)
                        printf("%d\t", T[i][j]);
                printf("\n");
        }
}
```

Exercice 2 :

Ecrire un programme qui réalise l'addition de deux matrices A et B de même dimension N x M (N et M sont saisies au clavier).

rappel :

a | b |	+	a' | b' |	=	a + a' | b + b' |
c | d |		c' | d' |		c + c' | d + d' |

```
#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;
}
```

Exercice 3 :

Ecrire un programme qui réalise le produit de deux matrices carrées de même dimension.

rappel :

a	b	*	a’	b’	=	a *a’ + b*c’	a*b’ + b*d’
c	d		c’	d’		c*a’ + d*c‘	c*b’ + d*d’

```
#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 ;           /* dimension des matrices (les matrices sont carrées)*/
        int i,j,k;        /* indices courants        */

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

        printf("*** Matrice A ***\n");
        for (i=0; i<N; i++)
                for (j=0; j<N; 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<N; 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<N; j++)
                        printf("%7d", A[i][j]);
                printf("\n");
        }
        printf("Matrice donnée B :\n");
        for (i=0; i<N; i++)
        {
                for (j=0; j<N; j++)
                        printf("%7d", B[i][j]);
                printf("\n");
        }


 /* Affectation du résultat du produit à C */
        for (i=0; i<N; i++)
        for (j=0; j<N; j++)
        {
                C[i][j] = 0 ;
                for(k = 0 ; k<N ; 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<N; j++)
                        printf("%7d", C[i][j]);
                printf("\n");
        }
}
```

TP 5 :

Exercice 1 :

Réécrire la fonction longueur (strln dans string.h) qui calcul la longueur d'une chaîne de caractères.
Prototype : *int longueur(char *)*

```
int longueur(char *chaine)
{
        int         i=0 ;

        while(chaine[i] != '\0')
                i++ ;

        return i ;
}
```

Exercice 2 :

En utilisant la précédence lexicographique écrire une fonction qui convertie les chaînes de caractères minuscules en chaînes de caractères majuscules.
Prototype : *void majuscule(char *)*

```
#include<stdio.h>
void majuscule(char *) ;
main()
{
    char chaine[] = "Ceci est une chaine !" ;

    majuscule(chaine) ;
    printf("%s\n",chaine) ;
}
void majuscule(char *chaine)
{
    int i=0;

    while(chaine[i] != '\0')
    {
        if ((chaine[i] >= 'a') && (chaine[i] <= 'z'))
            chaine[i] += (int)'A' - (int)'a' ;
        i++ ;
    }
}
```

Exercice 3 :

Ecrire un programme qui lit deux chaînes de caractères, et qui indique leur précédence lexicographique dans le code de caractères de la machine (ici: code ASCII). On écrira pour cela la fonction *precedence* qui récupère les deux chaînes en paramètre et qui retourne 1 si la première chaîne précède la deuxième, 2 si la deuxième précède la première, 0 si elle sont égale.
Prototype : *int precedence(char *,char *)*

```
#include <stdio.h>
int precedence(char *,char *) ;
main()
{
 /* Déclarations */
 char CH1[50], CH2[50]; /* chaînes à comparer */
int r      ;

        /* Saisie des données */
 printf("Entrez la première chaîne à comparer : ");
        gets(CH1);
        printf("Entrez la deuxième chaîne à comparer : ");
        gets(CH2);
        r = precedence (CH1,CH2) ;
        if(r==0)
                printf("\"%s\" est égal à \"%s\"\n", CH1, CH2);
        else if (r == 1)
                printf("\"%s\" précède \"%s\"\n", CH1, CH2);
        else
                printf("\"%s\" précède \"%s\"\n", CH2, CH1);
 }

int precedence (char *CH1,char *CH2)
{
         int I;              /* indice courant     */
        int r ;

        for (I=0; (CH1[I]==CH2[I]) && CH1[I] && CH2[I]; I++) ;

        if (CH1[I]==CH2[I])
                r = 0 ;
        else if (CH1[I]<CH2[I])
                r = 1 ;
        else
                r = 2 ;
  return r;
}
```

Exercice 4 :

Ecrire une procédure qui lit une chaîne de caractères et l'interprète comme un entier positif dans la base décimale. On écrira 2 fonctions :
La fonction *chaine2entier* qui récupère une chaîne de caractère et retourne un entier.
Prototype : *int chaine2entier(char *)*
La fonction *estentier* qui récupère un caractère et retourne 0 s'il ne correspond pas à un chiffre 1 s'il correspond à un chiffre.
Prototype : *int estentier(char) ;*

```
#include<stdio.h>
int estentier(char) ;
int chaine2entier(char *) ;
 main()
{
 /* Déclarations */
        char CH[100]; /* chaîne numérique à convertir */
        long N; /* résultat numérique */

        printf("Entrez un nombre entier et positif : ");
        gets(CH);

        printf("%s\n",CH) ;

        N = chaine2entier(CH) ;
        if(N<0)
                printf("%s ne représente pas correctement  un entier positif.\n", CH);
        else
                printf("La chaine %s a pour valeur %d\n" ,CH,N) ;
}

 int chaine2entier(char *CH)
{
        int    I;
        int     N = 0 ;
        int     OK = 1 ;

        for (I=0; OK && CH[I]; I++)
        if (estentier(CH[I]))
                N = N*10 + (CH[I]-'0');
        else
                OK=0;
        if (OK)
                return N ;
        else
                return -1 ;
}

int estentier(char c)
{
        if ((c>='0')&&(c<='9'))
                return 1 ;
        else
                        return 0 ;
}
```