2014/202015
SMI-S3
Travaux Dirigés de Langage C
Série N°7

Exercice 7.1

Ecrire la fonction ECRIRE_MATRICE à quatre paramètres MAT, L, C et CMAX qui affiche les composantes de la matrice de dimensions L et C.

Exercice 7.2

Ecrire la fonction SOMME_MATRICE du type **long** qui calcule la somme des éléments d'une matrice MAT du type **int**. Choisir les paramètres nécessaires. Ecrire un petit programme qui teste la fonction SOMME_MATRICE.

Exercice 7.3

Ecrire la fonction ADDITION_MATRICE qui effectue l'addition des matrices suivante:

MAT1 = MAT1 + MAT2

Choisir les paramètres nécessaires et écrire un petit programme qui teste la fonction ADDITION_MATRICE.

Exercice 7.4

Ecrire la fonction MULTI_MATRICE qui effectue la multiplication de la matrice MAT1 par un entier X:

MAT1 = X * MAT1

Choisir les paramètres nécessaires et écrire un petit programme qui teste la fonction MULTI_MATRICE.

Exercice 7.5

Ecrire la fonction TRANSPO_MATRICE à cinq paramètres MAT, L, LMAX, C, CMAX qui effectue la transposition de la matrice MAT en utilisant la fonction PERMUTER. TRANSPO_MATRICE retourne une valeur logique qui indique si les dimensions de la matrice sont telles que la transposition a pu être effectuée. Ecrire un petit programme qui teste la fonction TRANSPO_MATRICE.

Exercice 7.6

Ecrire la fonction MULTI_2_MATRICES qui effectue la multiplication de deux matrices MAT1 (dimensions N et M) et MAT2 (dimensions M et P) en une troisième matrice MAT3 (dimensions N et P):

MAT3 = MAT1 * MAT2

Supposez que les dimensions maximales des trois matrices soient toutes égales à 30 lignes et 30 colonnes. Ecrire un petit programme qui teste la fonction MULTI_2_MATRICES.

2014/202015
SMI-S3
Corrigé des Travaux Dirigés de Langage C
Série N°7

Exercice 7.1

```
void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX)
{
 /* Variables locales */
 int I,J;
 /* Affichage des composantes de la matrice */
 for (I=0; I<L; I++)
     {
      for (J=0; J<C; J++)
           printf("%7d", *(MAT + I*CMAX + J));
      printf("\n");
     }
}
```

Exercice 7.2

```
#include <stdio.h>
main()
{
 /* Prototypes des fonctions appelées */
 long SOMME_MATRICE  (int *MAT, int L, int C, int CMAX);
 void LIRE_DIM     (int *L, int LMAX, int *C, int CMAX);
 void LIRE_MATRICE   (int *MAT, int L, int C, int CMAX);
 void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX);

 /* Variables locales */
 int M[30][30]; /* Matrice d'entiers */
 int L, C;      /* Dimensions de la matrice */
 /* Traitements */
 LIRE_DIM (&L, 30, &C, 30);
 LIRE_MATRICE ( (int*)M, L,C,30);
  printf("Matrice donnée : \n");
 ECRIRE_MATRICE ( (int*)M, L,C,30);
  printf("Somme des éléments de la matrice : %ld\n",
                        SOMME_MATRICE( (int*)M, L,C,30));
 return 0;
}
long SOMME_MATRICE(int *MAT, int L, int C, int CMAX)
{
  /* Variables locales */
 int I,J;
 long SOMME = 0;
  /* Calcul de la somme */
 for (I=0; I<L; I++)
      for (J=0; J<C; J++)
           SOMME += *(MAT + I*CMAX + J);
 return SOMME;
}

void LIRE_DIM (int *L, int LMAX, int *C, int CMAX)
{
 . . .
}
```

```
void LIRE_MATRICE (int *MAT, int L, int C, int CMAX)
{
 . . .
}

void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX)
{
 . . .
}
```

Exercice 7.3

```
#include <stdio.h>
main()
{
 /* Prototypes des fonctions appelées */
 void ADDITION_MATRICE (int *MAT1, int *MAT2, int L, int C, int CMAX);
 void LIRE_DIM     (int *L, int LMAX, int *C, int CMAX);
 void LIRE_MATRICE   (int *MAT, int L, int C, int CMAX);
 void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX);
 /* Variables locales */
 /* Les matrices et leurs dimensions */
 int M1[30][30], M2[30][30];
 int L, C;
 /* Traitements */
 LIRE_DIM (&L,30,&C,30);
 printf("*** Matrice 1 ***\n");
 LIRE_MATRICE ((int*)M1,L,C,30 );
 printf("*** Matrice 2 ***\n");
 LIRE_MATRICE ((int*)M2,L,C,30 );
 printf("Matrice donnée 1 : \n");
 ECRIRE_MATRICE ((int*)M1,L,C,30);
 printf("Matrice donnée 2 : \n");
 ECRIRE_MATRICE ((int*)M2,L,C,30);
 ADDITION_MATRICE( (int*)M1 , (int*)M2 ,L,C,30);
  printf("Matrice résultat : \n");
 ECRIRE_MATRICE ((int*)M1,L,C,30);
 return 0;
}

void ADDITION_MATRICE (int *MAT1, int *MAT2, int L, int C, int CMAX)
{
  /* Variables locales */
 int I,J;
 /* Ajouter les éléments de MAT2 à MAT1 */
 for (I=0; I<L; I++)
      for (J=0; J<C; J++)
           *(MAT1+I*CMAX+J) += *(MAT2+I*CMAX+J);
}

void LIRE_DIM (int *L, int LMAX, int *C, int CMAX)
{
 . . .
}

void LIRE_MATRICE (int *MAT, int L, int C, int CMAX)
{
 . . .
}

void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX)
```

```
{
 . . .
}
```

Exercice 7.4

```
#include <stdio.h>
main(){
 /* Prototypes des fonctions appelées */
 void MULTI_MATRICE(int X, int *MAT, int L, int C, int CMAX);
 void LIRE_DIM     (int *L, int LMAX, int *C, int CMAX);
 void LIRE_MATRICE   (int *MAT, int L, int C, int CMAX);
 void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX);
 /* Variables locales */
 int M[30][30]; /* Matrice d'entiers */
 int L, C;      /* Dimensions de la matrice */
 int X;
 /* Traitements */
 LIRE_DIM (&L,30,&C,30);
 LIRE_MATRICE ((int*)M,L,C,30 );
 printf("Introduire le multiplicateur (entier) : ");
 scanf("%d", &X);
 printf("Matrice donnée : \n");
 ECRIRE_MATRICE ((int*)M,L,C,30);
 MULTI_MATRICE (X,(int*)M,L,C,30);
  printf("Matrice résultat : \n");
 ECRIRE_MATRICE ((int*)M,L,C,30);
 return 0;
}

void MULTI_MATRICE(int X, int *MAT, int L, int C, int CMAX)
{
  /* Variables locales */
 int I,J;
 /* Multiplication des éléments */
 for (I=0; I<L; I++)
      for (J=0; J<C; J++)
           *(MAT+I*CMAX+J) *= X;
}

void LIRE_DIM (int *L, int LMAX, int *C, int CMAX)
{
 . . .
}

void LIRE_MATRICE (int *MAT, int L, int C, int CMAX)
{
 . . .
}

void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX)
{
 . . .
}
```

Exercice 7.5

```
#include <stdio.h>
main()
{
 /* Prototypes des fonctions appelées */
 int TRANSPO_MATRICE (int *MAT, int *L, int LMAX, int *C, int CMAX);
```

```
 void LIRE_DIM     (int *L, int LMAX, int *C, int CMAX);
 void LIRE_MATRICE   (int *MAT, int L, int C, int CMAX);
 void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX);
 /* Variables locales */
 int M[30][30]; /* Matrice d'entiers */
 int L, C;      /* Dimensions de la matrice */
 /* Traitements */
 LIRE_DIM (&L,30,&C,30);
 LIRE_MATRICE ((int*)M,L,C,30 );
 printf("Matrice donnée : \n");
 ECRIRE_MATRICE ((int*)M,L,C,30);
 if (TRANSPO_MATRICE ((int*)M,&L,30,&C,30))
    {
     printf("Matrice transposée : \n");
     ECRIRE_MATRICE ((int*)M,L,C,30);
    }
 else
    printf("\aLa matrice n'a pas pu être transposée\n");
  return 0;
}
int TRANSPO_MATRICE (int *MAT, int *L, int LMAX, int *C, int CMAX)
{
  /* Prototypes des fonctions appelées */
 void PERMUTER(int *A, int *B);
 /* Variables locales */
 int I,J;
 int DMAX; /* la plus grande des deux dimensions */
 /* Transposition de la matrice */
 if (*L>CMAX || *C>LMAX)
     return 0;
 else
     {
      DMAX = (*L>*C) ? *L : *C;
      for (I=0; I<DMAX; I++)
          for (J=0; J<I; J++)
              PERMUTER (MAT+I*CMAX+J, MAT+J*CMAX+I);
      PERMUTER(L,C);  /* échanger les dimensions */
      return 1;
     }
}

void PERMUTER(int *A, int *B)
{
 . . .
}

void LIRE_DIM (int *L, int LMAX, int *C, int CMAX)
{
 . . .
}

void LIRE_MATRICE (int *MAT, int L, int C, int CMAX)
{
 . . .
}

void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX)
{
 . . .
}
```

Exercice 7.6

```
#include <stdio.h>

main()
{
 /* Prototypes des fonctions appelées */
 void MULTI_2_MATRICES (int *MAT1, int *MAT2, int *MAT3,
                        int N, int M, int P, int CMAX);
 void LIRE_DIM     (int *L, int LMAX, int *C, int CMAX);
 void LIRE_MATRICE   (int *MAT, int L, int C, int CMAX);
 void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX);
 /* Variables locales */
 /* Les matrices et leurs dimensions */
 int M1[30][30], M2[30][30], M3[30][30];
 int N, M, P;
 int DUMMY; /* pour la lecture de la première dimension de */
            /* MAT2 à l'aide de LIRE_DIM. */
 /* Traitements */
 printf("*** Matrice 1 ***\n");
 LIRE_DIM (&N,30,&M,30);
  LIRE_MATRICE ((int*)M1,N,M,30 );
 printf("*** Matrice 2 ***\n");
 LIRE_DIM (&DUMMY,30,&P,30);
  LIRE_MATRICE ((int*)M2,M,P,30 );
 printf("Matrice donnée 1 : \n");
 ECRIRE_MATRICE ((int*)M1,N,M,30);
 printf("Matrice donnée 2 : \n");
 ECRIRE_MATRICE ((int*)M2,M,P,30);
 MULTI_2_MATRICES ((int*)M1 , (int*)M2 , (int*)M3 , N,M,P,30);
  printf("Matrice résultat : \n");
 ECRIRE_MATRICE ((int*)M3,N,P,30);
 return 0;
}
void MULTI_2_MATRICES (int *MAT1, int *MAT2, int *MAT3,
                       int N, int M, int P, int CMAX)
{
  /* Variables locales */
 int I,J,K;
 /* Multiplier MAT1 et MAT2 en affectant le résultat à MAT3 */
 for (I=0; I<N; I++)
    for (J=0; J<P; J++)
       {
        *(MAT3+I*CMAX+J)=0;
        for (K=0; K<M; K++)
            *(MAT3+I*CMAX+J) += *(MAT1+I*CMAX+K) * *(MAT2+K*CMAX+J);
       }
}
void LIRE_DIM (int *L, int LMAX, int *C, int CMAX)
{
 . . .
}
void LIRE_MATRICE (int *MAT, int L, int C, int CMAX)
{
 . . .
}
void ECRIRE_MATRICE (int *MAT, int L, int C, int CMAX)
{
 . . .
}
```