


République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et d'Informatique

Département Informatique

Filière : Informatique

Spécialité :

Informatique Académique

Rapport de projet
Vecteurs, Matrices, et listes chainées.

Proposé par : Mme boukheddouma

Chargé de TP : **Réalisé par :**

Mme Ouazar **Belkhir Selma**

Introduction:

En Algorithmique nous avons déjà appris comment manipuler et stocker des données en déclarant des variables simples : entiers, caractères, réels et booléens.

Lorsque les données sont nombreuses et de même nature, au lieu de manipuler un grand nombre de variables, il est plus pratique de ranger ces variables dans un tableau.
La notion de tableau est une notion courante très utilisée, pas seulement en informatique. Il s'agit la plupart du temps de tableaux à deux dimensions "matrice" avec des lignes et des colonnes. Une matrice est un tableau ou les éléments sont eux-mêmes des tableaux.
Mais on peut aussi considérer des tableaux à une seule dimension "vecteur" (ce sont plutôt des listes dans le langage courant). Un vecteur se compose d'un ensemble de variables hétérogènes appelées éléments du vecteur qui sont accessible via des indices.
Aussi, on a souvent besoin de manipuler un texte composé de lignes successives, donc une suite de chaînes, dite chaine de caractère qui sert à représenter des informations non numériques comme par exemple les noms, et les adresses.....
Cependant, les tableaux ont une structure fixe et leurs tailles sont limitées. A l'aide des pointeurs, nous pouvons représenter les éléments de même type avec une structure dynamique appelée la liste chaînée.
Cette dernière représente un ensemble d'éléments de même type reliés entre eux. Le principe est qu'à partir de chaque élément, nous pouvons accéder à l'élément suivant. Pour cela, chaque élément est composé de deux parties : une partie qui contient l'information de l'élément qui peut être de type simple ou composé et une partie qui contient l'adresse du prochain élément appelé aussi lien de chaînage.
Notre travail consiste donc à la manipulation des tableaux, de chaines de caractères et des listes chainées qui permettent de regrouper des données de même type.
A travers notre travail il est possible d'utiliser un menu qui permet à l'utilisateur de choisir une des options programmées.
Le travail comporte trois parties :
-Opérations arithmétiques sur les matrices.
-Autres opérations sur les matrices.
-Matrices, chaines de caractères, et listes chainées.

Partie 1 : Opérations Arithmétiques sur les matrices

Introduction

Dans la première partie intitulé opérations arithmétiques sur les matrices, on présente l'addition et la soustraction de deux matrices, la multiplication d'une matrice par un nombre, la multiplication de deux matrices, et l'inversion d'une matrice.
Pour l'addition et la soustraction de matrices elles ne sont possibles que lorsque les deux matrices sont de mêmes dimensions, c'est-à-dire qu'elles ont le même nombre de lignes et le même nombre de colonnes.
Pour faciliter la tâche on a utilisé une fonction pour l'allocation dynamique d'une matrice, une fonction pour la libérer, une fonction d'affichage pour vecteur et matrice et une autre fonction pour saisir une matrice, dont les prototypes sont les suivants :
float** Allocation_Matrice (int, int);
void Liberer_Matrice (float**, int);
void Affiche_Vecteur (float*, int);
void Affiche_Mat (float**, int, int);
void Saisi_Mat (float**, int, int);

-Addition et Soustraction : Pour additionner deux matrices, il suffit d'additionner les éléments occupant les mêmes positions dans chaque matrice. La somme obtenue est une nouvelle matrice.
Et pour soustraire deux matrices, il suffit de soustraire aux éléments de la première matrice les éléments occupant la même position dans la deuxième matrice. Tout comme dans l'addition, la soustraction de deux matrices résulte en une nouvelle matrice.
Le prototype de cette fonction :
void Add_Soust_Mat (char, float**, float** , int, int);
Code :

```
case 11:  // ****************La somme de deux matrices (Addition)**********************
{
    printf("\n Donnez les dimensions des matrices:\n");
    printf("Nombre de lignes:  ");      scanf("%d", &N);
    printf("Nombre de colonnes:  ");    scanf("%d", &M);

    matrice1 = Allocation_Matrice( N, M );
    matrice2 = Allocation_Matrice( N, M );

    printf("\n Saisissez les valeurs de La 1ere matrice:\n");
    Saisi_Mat( matrice1, N, M );
    printf("La 1ere Matrice est:\n\n");
    Affiche_Mat( matrice1, N, M );

    printf("\n Saisissez les valeurs de La 2eme matrice:\n");
    Saisi_Mat( matrice2, N, M );
    printf("La 2eme Matrice est:\n\n");
    Affiche_Mat( matrice2, N, M );

    Add_Soust_Mat( '+', matrice1, matrice2, N, M ); // l'additon
    Affiche_Mat( matrice1, N, M );

    Liberer_Matrice( matrice1, N );  // Liberer la memoire
    Liberer_Matrice( matrice2, N );
}
break;
```

```c
case 12:   // ***************La difference de matrices (Soustraction)******************
{
    printf("\n Donnez les dimensions des matrices:\n");
    printf("Nombre de lignes:  ");      scanf("%d", &N);
    printf("Nombre de colonnes:  ");    scanf("%d", &M);

    matrice1 = Allocation_Matrice( N, M );
    matrice2 = Allocation_Matrice( N, M );

    printf("\n Saisissez les valeurs de La 1ere matrice:\n");
    Saisi_Mat( matrice1, N, M );
    printf("La 1ere Matrice est:\n\n");
    Affiche_Mat( matrice1, N, M );

    printf("\n Saisissez les valeurs de La 2eme matrice:\n");
    Saisi_Mat( matrice2, N, M );
    printf("La 2eme Matrice est:\n\n");
    Affiche_Mat( matrice2, N, M );

    Add_Soust_Mat( '-', matrice1, matrice2, N, M ); // soustraction
    Affiche_Mat( matrice1, N, M );

    Liberer_Matrice( matrice1, N );
    Liberer_Matrice( matrice2, N );
}
break;
```

Exécution :

```
*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  autres operations
 4.Quitter
Choisissez 1 ou 2 ou 3 ou 4 :  1

 1. Somme de deux matrices
 2. Difference de deux matrices
 3. Multiplication d'une matrice par un nombre
 4. Produit de deux matrices
 5. Inverse d'une matrice
Choisissez un numero compris entre 1 et 5 du menu :  1

 Donnez les dimensions des matrices:
Nombre de lignes:  2
Nombre de colonnes:  2

 Saisissez les valeurs de La 1ere matrice:
matrice(0,0) :  3
matrice(0,1) :  7
matrice(1,0) :  8
matrice(1,1) :  2
La 1ere Matrice est:

3.000000        7.000000
8.000000        2.000000

 Saisissez les valeurs de La 2eme matrice:
matrice(0,0) :  0
matrice(0,1) :  1
matrice(1,0) :  3
matrice(1,1) :  2
La 2eme Matrice est:

0.000000        1.000000
3.000000        2.000000

La Somme des deux Matrices est:

3.000000        8.000000
11.000000       4.000000
```

```
*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  autres operations
 4.Quitter
Choisissez 1 ou 2 ou 3 ou 4 :  1

 1. Somme de deux matrices
 2. Difference de deux matrices
 3. Multiplication d'une matrice par un nombre
 4. Produit de deux matrices
 5. Inverse d'une matrice
Choisissez un numero compris entre 1 et 5 du menu :  2

 Donnez les dimensions des matrices:
Nombre de lignes:  3
Nombre de colonnes:  2

 Saisissez les valeurs de La 1ere matrice:
matrice(0,0) :  19
matrice(0,1) :  13
matrice(1,0) :  15
matrice(1,1) :  6
matrice(2,0) :  43
matrice(2,1) :  2
La 1ere Matrice est:

19.000000         13.000000
15.000000         6.000000
43.000000         2.000000

 Saisissez les valeurs de La 2eme matrice:
matrice(0,0) :  12
matrice(0,1) :  13
matrice(1,0) :  14
matrice(1,1) :  15
matrice(2,0) :  16
matrice(2,1) :  17
La 2eme Matrice est:

12.000000         13.000000
14.000000         15.000000
16.000000         17.000000

La Difference entre les deux Matrices est:

7.000000          0.000000
1.000000          -9.000000
27.000000         -15.000000
```

Multiplication d'une matrice par un nombre : Pour multiplier une matrice par un scalaire, il faut multiplier chaque élément de la matrice par ce scalaire. Le produit obtenu est donc une nouvelle matrice.

Le prototype de cette fonction :

void Multi_Mat_Nbr (float**, float, float, float);

Code :

```
//**********************Multiplication d'une matrice par un nombre**************************
void Multi_Mat_Nbr( float **mat, int nbr, int N, int M )
{
    int i, j;

    for( i=0; i<N; i++ )
        for( j=0; j<M; j++ )
            mat[i][j] *= nbr;
}
```

Exécution :

```
*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  autres operations
 4.Quitter
Choisissez 1 ou 2 ou 3 ou 4 :  1

 1. Somme de deux matrices
 2. Difference de deux matrices
 3. Multiplication d'une matrice par un nombre
 4. Produit de deux matrices
 5. Inverse d'une matrice
Choisissez un numero compris entre 1 et 5 du menu :  3

 Donnez les dimensions de la matrice:
Nombre de lignes:  2
Nombre de colonnes:  2

 Saisissez les valeurs de La matrice:
matrice(0,0) :  19
matrice(0,1) :  13
matrice(1,0) :  11
matrice(1,1) :  24
Voici votre Matrice initiale:

19.000000       13.000000
11.000000       24.000000

 Donnez un nombre:   2

La Matrice multipliÚe par 2.000000 est:

38.000000       26.000000
22.000000       48.000000
```

Produit de deux matrices : Le produit d'une matrice A par une matrice B est possible si et seulement si le nombre de colonnes de la matrice A est égal au nombre de lignes de la matrice B.

Le prototype de cette fonction :

float** Produit_Matriciel (float**, float**, int, int, int);

Code :

```
//*************************** Multiplication de deux matrices*****************************
float** Produit_Matriciel( float **mat1, float **mat2, int N, int M, int P )
{
    int i, k, j;
    float **mat3 = NULL;

    mat3 = Allocation_Matrice(N, P);

    /* produit matriciele : */
    for( i=0; i<N; i++ )
        for( j=0; j<P; j++ )
        {
            mat3[i][j] = 0; /* initialisation à 0 */
            for(k=0; k<M; k++)
                mat3[i][j] += mat1[i][k] * mat2[k][j];
        }

    return mat3;
}
```

Exécution :

```
*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  autres operations
 4.Quitter
Choisissez 1 ou 2 ou 3 ou 4 :  1

 1. Somme de deux matrices
 2. Difference de deux matrices
 3. Multiplication d'une matrice par un nombre
 4. Produit de deux matrices
 5. Inverse d'une matrice
Choisissez un numero compris entre 1 et 5 du menu :  4

 Les dimensions de la 1ere matrice:
Nombre de lignes:  2
Nombre de colonnes:  2

 Saisissez les valeurs de La 1ere matrice:
matrice(0,0) :  14
matrice(0,1) :  15
matrice(1,0) :  16
matrice(1,1) :  17
La 1ere Matrice est:

14.000000       15.000000
16.000000       17.000000

 Les dimensions de la 2eme matrice:
Nbr de lignes = 2 (obligatoirement saisi)
Nombre de colonnes:  3

 Saisissez les valeurs de La 2eme matrice:
matrice(0,0) :  14
matrice(0,1) :  15
matrice(0,2) :  16
matrice(1,0) :  17
matrice(1,1) :  18
matrice(1,2) :  19
La 2eme Matrice est:

14.000000       15.000000       16.000000
17.000000       18.000000       19.000000

Le Produit matriciel est:

451.000000      480.000000      509.000000
513.000000      546.000000      579.000000
```

Inverse d'une Matrice : La comatrice d'une matrice carrée A est une matrice carrée de même taille, dont les coefficients, appelés les cofacteurs de A, interviennent dans le développement du déterminant de A suivant une ligne ou une colonne, si A est une matrice inversible, sa comatrice intervient également dans une expression de son inverse.
La comatrice C d'une matrice carrée A d'ordre n, est la matrice obtenue en remplaçant chaque élément aij de la matrice A par son cofacteur.
On appelle cofacteur de l'élément aij le nombre (–1)i+jD.
Donc pour calculer l'inverse on a besoin d'une fonction qui enlève une ligne et une colonne précise de cette matrice, pour l'utiliser dans le calcul du déterminant (Le déterminant se calcule en multipliant les deux termes de la diagonale : a x d, puis les deux autres : b x c.), une fois calculé on écrit une fonction qui calcule la comatrice et la transposée de cette dernière qui intervient dans l'expression de l'inverse.

$$A^{-1} = \frac{{}^t\mathrm{com}(A)}{|A|} \quad (|A| \neq 0),$$

où ${}^t\mathrm{com}(A)$ est la transposée de la comatrice de A.

Comatrice

La **comatrice** d'une matrice carrée A est la matrice des cofacteurs de A. Si on la note com(A), elle vérifie :

$${}^t com(A)A = A{}^t com(A) = \det(A) I_n.$$

Dans le cas où A est inversible, la comatrice de A est reliée à l'inverse de A par la formule :

$$A^{-1} = \frac{1}{\det(A)} {}^t com(A).$$

Les prototypes utilisés :

float** Sous_Mat_Det(float** mat, int dim, int lign_a_sup, int col_a_sup)
float Determinant(float **mat, int N)
void Comatrice(float** mat, float** comat, int dim)
float** Inverse_Matrice(float **matrice, int dim)

Code :

```
/* Fonction qui enleve une ligne et une colonne precise ï¿½ une matrice,
   Pour l'utiliser dans le calcul du determinant :) */
float** Sous_Mat_Det( float** mat, int dim, int lign_a_sup, int col_a_sup )
{
    int i, j, l = 0, c = 0;
    float **sousMat;

    sousMat = Allocation_Matrice( dim-1, dim-1 );

    //parcourir chaque ligne
    for( i=0; i<dim; i++ )
        if( i != lign_a_sup ) /* si cet ligne n'est ps Ã  supprimer: */
        {
            for( j=0; j<dim; j++ )
                if( j != col_a_sup ) /*si ce n'est pas la colonne Ã  supprimer */
                {
                    sousMat[l][c] = mat[i][j];
                    c++;
                    if( c > dim-2 )
                    {
                        c = 0;
                        l++;
                    }
                }
        }

    return sousMat;
```

```c
float Determinant( float **mat, int N )
{
    int c, signe = 1;
    float det = 0.0;

    if( N == 1 )
        return mat[0][0];

    for( c=0; c<N; c++ )
    {
        det += signe * mat[0][c] * Determinant(Sous_Mat_Det(mat,N,0,c), N-1);
        signe *= (-1);
    }
//    printf("det est %f\n",det);
    return det;
}
```

```c
/* Fonction qui calcule la CoMatrice d'une matrice donnï¿½e,
   pour l'utiliser dans le calcul de l'inverse */
void Comatrice( float** mat, float** comat, int dim )
{
    int i, j;

    for( i=0; i<dim; i++ )
        for( j=0; j<dim; j++ )
            comat[i][j] = Determinant( Sous_Mat_Det( mat, dim, i, j ), dim-1 );
}

/* Fonction qui calcule l'inverse d'une matrice donnï¿½e, et retourne NULL si la
```

```c
/* Fonction qui calcule l'inverse d'une matrice donnée, et retourne NULL si la
   matrice est non inversible ( det == 0 ) */
float** Inverse_Matrice( float **matrice, int dim )
{
    float det = Determinant( matrice, dim );
    printf("det est %f\n",det);
    float **comatrice, **comatTransposed;
    if( det == 0 )
    {
        printf("\nLe determinant est null, cet matrice est non inversible\n\n");
        return NULL;
    }
    comatrice = Allocation_Matrice( dim, dim );
    comatTransposed = Allocation_Matrice( dim, dim );
    /* calculer la comatrice de la matrice */
    Comatrice( matrice, comatrice, dim );
    printf("\n\nLa comatrice:\n");
    Affiche_Mat(comatrice, dim, dim);
    /* calculer la comatrice transpose */
    Transpose_Mat( comatrice, comatTransposed,dim, dim );
    printf("\n\nLa comatrice transposÃ©:\n");
    Affiche_Mat(comatTransposed, dim, dim);
    /* calculer la matrice inverse selon la relation suivante:
       l'inverse d'un matrice = (1 / det) * comatriceTranspoed */
    Multi_Mat_Nbr( comatTransposed, (1.0/det), dim, dim );
    return comatTransposed;
}
/* La transposï¿½e d'une matrice */
```

Exécution :

```
Choisissez un numero compris entre 1 et 5 du menu :  5

 Donnez les dimensions de la matrice (carr´┐¢e):
Nombre de lignes == Nombre de colonnes : 3

 Saisissez les valeurs de La matrice:
matrice(0,0) :  -1
matrice(0,1) :  2
matrice(0,2) :  5
matrice(1,0) :  1
matrice(1,1) :  2
matrice(1,2) :  3
matrice(2,0) :  -2
matrice(2,1) :  8
matrice(2,2) :  10
Voici votre Matrice initiale:


|-1.000000      |2.000000       |5.000000       |
|1.000000       |2.000000       |3.000000       |
|-2.000000      |8.000000       |10.000000      |
det est 32.000000


La comatrice:

|-4.000000      |16.000000      |12.000000      |
|-20.000000     |0.000000       |-4.000000      |
|-4.000000      |-8.000000      |-4.000000      |


La comatrice transpos├®:

|-4.000000      |-20.000000     |-4.000000      |
|16.000000      |0.000000       |-8.000000      |
|12.000000      |-4.000000      |-4.000000      |
la matrice inverse est:

|-0.125000      |-0.625000      |-0.125000      |
|0.500000       |0.000000       |-0.250000      |
|0.375000       |-0.125000      |-0.125000      |
```

Partie 2 : Autres Opérations

Introduction :

Dans cette partie nous allons voir quelques algorithmes sur les matrices : transposée et tri d'une matrice, extraction de sous-matrices et calcul du vecteur Maxligne et Maxcolonne.

Transposée d'une Matrice : On obtient la matrice transposée en échangeant les lignes et les colonnes.

Le prototype de la fonction :

*void Transpose_Mat (float**, float**, int, int);*

Code :

```
/* La transposé d'une matrice */
void Transpose_Mat( float **mat1, float **mat2 ,int N, int M )
{
    int i,j;

    for( i=0; i<N; i++ )
        for( j=0; j<M; j++ )
            mat2[j][i] = mat1[i][j];
}
```

Exécution :

```
*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  autres operations
 4.Quitter
Choisissez 1 ou 2 ou 3 ou 4 :  2

 1. Transposee d'une matrice
  2. Tri d'une matrice
 3. Calcul du vecteur Maxligne
 4. Calcule du vecteur Maxcolonne
 5. Extraction de Sous-Matrice
Choisissez un numero entre 1 et 5 du menu :  1

 Donnez les dimensions de la matrice:
Nombre de lignes:  2
Nombre de colonnes:  2

 Saisissez les valeurs de La matrice:
matrice(0,0) :  32
matrice(0,1) :  4
matrice(1,0) :  1
matrice(1,1) :  6
Voici votre Matrice initiale:

32.000000          4.000000
1.000000           6.000000

La matrice transposee est:

32.000000          1.000000
4.000000           6.000000
```

Le tri d'une matrice : Trier un tableau c'est obtenir, à partir d'un tableau t, un tableau contenant les mêmes éléments mais rangés par ordre croissant ou décroissant.

Les prototypes de la fonction :

void Trie_Mat_Croissant (float**, int, int);

void Trie_Mat_Decroissant (float**, int, int);

Code :

```c
/* Trier la matrice selon l'ordre Decroissant (Trie par selection) */
void Trie_Mat_Decroissant( float **mat, int N, int M )
{
  int i, j, L, C, posColonMin, posLignMin;
  float tmp;

    printf("\nLa matrice Trie selon l'ordre decroissant est:\n\n");

    for ( i=0; i<N; i++ )
        for ( j=0; j<N; j++ )
        {
            /* Recherche du maximum aprés A[i][j] */
            posLignMin = i;
            posColonMin = j;

            C = j;

            for ( L=i; L<N; L++ )
            {
               if( L != i )
                 C=0;
               for ( ; C<M; C++ )
                if ( mat[L][C] > mat[posLignMin][posColonMin] )
                    {
                        posLignMin = L;
                        posColonMin = C;
                    }

            }

            /* Echange de A[I] avec le maximum */
            tmp = mat[i][j];
            mat[i][j] = mat[posLignMin][posColonMin];
            mat[posLignMin][posColonMin] = tmp;
        }
}
```

```c
/* Trier la matrice selon l'ordre Croissant (Tri par selection) */
void Trie_Mat_Croissant( float **mat, int N, int M )
{
    int i, j, L, C, posColonMin, posLignMin;
    float tmp;

    printf("\nLa matrice Trie selon l'ordre croissant est:\n\n");

    for ( i=0; i<N; i++ )
        for ( j=0; j<N; j++ )
        {
            /* Recherche du maximum aprés A[i][j] */
            posLignMin = i;
            posColonMin = j;
            C = j;

            for ( L=i; L<N; L++ )
            {
               if( L != i )
                 C = 0;
               for (; C<M; C++)
                if ( mat[L][C] < mat[posLignMin][posColonMin] )
                    {
                        posLignMin = L;
                        posColonMin = C;
                    }

            }

            /* Echange de A[I] avec le maximum */
            tmp = mat[i][j];
            mat[i][j] = mat[posLignMin][posColonMin];
            mat[posLignMin][posColonMin] = tmp;
        }
}
```

Exécution :

```
*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  Quitter
Choisissez 1 ou 2 ou 3 :  2

 1. Transposee d'une matrice
  2. Tri d'une matrice
 3. Calcul du vecteur Maxligne
 4. Calcule du vecteur Maxcolonne
 5. Extraction de Sous-Matrice
Choisissez un numero entre 1 et 5 du menu :  2
Entrer:
 D pour le trie Decroissant, Ou
 C pour l'ordre Croissant (en majuscule s'il vous plait'):
D

 Donnez la dimension de votre matrice:
Nombre de lignes:  2
Nombre de colonnes:  2

 Saisisez les valeurs de La matrice:
matrice(0,0) :  34
matrice(0,1) :  0
matrice(1,0) :  6
matrice(1,1) :  -8
Voici votre Matrice non trier:

34.000000        0.000000
6.000000         -8.000000

La matrice Trie selon l'ordre decroissant est:


la matrice aprÚs le trie :
34.000000        6.000000
0.000000         -8.000000
```

```
*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  autres operations
 4.Quitter
Choisissez 1 ou 2 ou 3 ou 4 :  2

 1. Transposee d'une matrice
  2. Tri d'une matrice
 3. Calcul du vecteur Maxligne
 4. Calcule du vecteur Maxcolonne
 5. Extraction de Sous-Matrice
Choisissez un numero entre 1 et 5 du menu :  2
Entrer:
 D pour le trie Decroissant, Ou
 C pour l'ordre Croissant (en majuscule s'il vous plait'):
C

 Donnez la dimension de votre matrice:
Nombre de lignes:  2
Nombre de colonnes:  2

 Saisisez les valeurs de La matrice:
matrice(0,0) :  34
matrice(0,1) :  1
matrice(1,0) :  -7
matrice(1,1) :  0
Voici votre Matrice non trier:

34.000000        1.000000
-7.000000        0.000000

La matrice Trie selon l'ordre croissant est:


la matrice aprÚs le trie :
-7.000000        0.000000
1.000000         34.000000
```

Le calcul du vecteur Maxligne : Il s'agit de construire un vecteur contenant pour chaque ligne i de la matrice, la plus grande valeur sur la ligne.

Le prototype de la fonction :

void Max_Ligne (float**, int, int);

Code :

```
/* Les elements max des lignes */
void Max_Ligne( float **mat, int N, int M )
{
    int i, j;
    float *maxLigne, Max;
    printf("\nLe vecteur des elements maximum des lignes est:\n\n");

    maxLigne = (float*) malloc( N * sizeof(float) );

    for(i=0; i<N; i++)
    {
        /* initialiser avec le 1er elt de la ligne */
        Max = mat[i][0];

        for(j=0; j<M; j++)
            if( Max < mat[i][j] )
                Max = mat[i][j];

        maxLigne[i] = Max;
    }

    Affiche_Vecteur( maxLigne, N );
}
```

Exécution :

```
 Donnez la dimension de votre matrice:
Nombre de lignes:  3
Nombre de colonnes:  2

 Saisissez les valeurs de La matrice:
matrice(0,0) :  19
matrice(0,1) :  13
matrice(1,0) :  26
matrice(1,1) :  4
matrice(2,0) :  29
matrice(2,1) :  26
Voici votre Matrice:

19.000000       13.000000
26.000000       4.000000
29.000000       26.000000

Le vecteur des elements maximum des lignes est:

19.000000       26.000000       29.000000
*** MENU ***
```

Le calcul du vecteur Maxcolonne : Il s'agit de construire un vecteur contenant pour chaque colonne j de la matrice, la plus grande valeur sur la colonne.

Le prototype de la fonction :

void Max_Colonne (float**, int, int);

Code :

```c
/* Les élements maximum des colonnes */
void Max_Colonne( float **mat, int N, int M )
{
    int i, j;
    float *maxColonne, Max;
    printf("\nLe vecteur des elements maximum des colonne est:\n\n");

    maxColonne = (float*) malloc( M * sizeof(float) );

    for(j=0; j<M; j++)
    {
        /* initialiser avec le 1er element de la colonne */
        Max = mat[0][j];

        for( i=0; i<N; i++ )
            if( Max < mat[i][j] )
                Max = mat[i][j];

        maxColonne[j] = Max;
    }

    Affiche_Vecteur( maxColonne, M );
}
```

Exécution :

```
 Donnez la dimension de votre matrice:
Nombre de lignes:  2
Nombre de colonnes:  2

 Saisissez les valeurs de La matrice:
matrice(0,0) :  12
matrice(0,1) :  5
matrice(1,0) :  67
matrice(1,1) :  8
Voici votre Matrice:

12.000000       5.000000
67.000000       8.000000

Le vecteur des elements maximum des colonne est:

67.000000       8.000000
```

Extraction de sous-matrices : Il s'agit d'extraire toutes les sous matrices s(k,l), k et l représentent les dimensions de la sous-matrice à extraire et seront données dans le programme.

Le prototype de la fonction :

void Extraction_Sous_Mat (float**, int, int, int, int);

Code :

```
/* Extarcation des sous matrices s_mat de taille s_N et s_M à partir de la
   matrice mat de dimentions N et M */
void Extraction_Sous_Mat( float **mat, int N, int M, int s_N, int s_M )
{
    int i, j, L=0, C=0;
    float **s_mat;
    while( L <= N-s_N )
    {
        /* allocation de la sous matrice */
        s_mat = Allocation_Matrice( s_N,  s_M);
/* remplissage de la sous matrice */
        for( i=0; i<s_N; i++ )
            for( j=0; j<s_M; j++ )
                s_mat[i][j] = mat[i+L][j+C];
/* si on à pas encor traiter la derniere sous matrice de cet etage */
        if( C < M-s_M )
            C++;
        else
        {
            C = 0;
            L++;
        }
   /* affichage de la sous matrice */
        Affiche_Mat(s_mat, s_N, s_M);
Liberer_Matrice(s_mat, s_N);
    }
 return;
}
```

Exécution :

```
*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  Quitter
Choisissez 1 ou 2 ou 3 :  2

 1. Transposee d'une matrice
  2. Tri d'une matrice
 3. Calcul du vecteur Maxligne
 4. Calcule du vecteur Maxcolonne
 5. Extraction de Sous-Matrice
Choisissez un numero entre 1 et 5 du menu :  5

 Donnez la dimension de votre matrice:
Nombre de lignes:  4
Nombre de colonnes:  4

 Saisissez les valeurs de La matrice:
matrice(0,0) :  13
matrice(0,1) :  19
matrice(0,2) :  26
matrice(0,3) :  11
matrice(1,0) :  -7
matrice(1,1) :  12
matrice(1,2) :  4
matrice(1,3) :  14
matrice(2,0) :  1
matrice(2,1) :  15
matrice(2,2) :  27
matrice(2,3) :  16
matrice(3,0) :  32
matrice(3,1) :  4
matrice(3,2) :  0
matrice(3,3) :  22

 Donnez la dimension des sous matrices a extraire:
Nombre de lignes:  2
Nombre de colonnes:  1
Voici votre Matrice initial:

13.000000       19.000000       26.000000       11.000000
-7.000000       12.000000       4.000000        14.000000
1.000000        15.000000       27.000000       16.000000
32.000000       4.000000        0.000000        22.000000
```

```
Voici toutes les sous matrices possibles:

13.000000
-7.000000
19.000000
12.000000
26.000000
4.000000
11.000000
14.000000
-7.000000
1.000000
12.000000
15.000000
4.000000
27.000000
14.000000
16.000000
1.000000
32.000000
15.000000
4.000000
27.000000
0.000000
16.000000
22.000000
```

Partie 3 : Matrices, chaines de caractères et listes chainées

Introduction :

Dans cette dernière partie on va traiter tout ce qui est chaines de caractères, matrice de chaine de caractère, les créations des structures de listes et leurs affichages et l'ajout et la suppression des chaines.

Saisie du texte : Permet à l'utilisateur de saisir un texte composé d'une succession de mots séparés par des espaces.

Code :

```c
case 30: // ********************saisie du texte*****************************
{
    printf("Entrez le texte : ");
    // pour enlever la saut de la ligne
    getchar();
    scanf("%[^\n]s", s);
}
break;
```

Exécution :

```
*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  autres operations
 4.Quitter
Choisissez 1 ou 2 ou 3 ou 4 :  3

 1. Saisie du texte
 2. Cr├®ation de la matrice de mots
 3. Affichage de la matrice
 4. Cr├®ation de la structure de listes
 5. Affichage de la structure
 6.Ajout d'un mot
 7.Suppression d'un mot
Choisissez un numero compris entre 1 et 7 du menu :  1
Entre le texte : BELKHIR SELMA ACAD A
```

Création de la matrice de mots : les chaînes de caractères peuvent être stockées sous la forme d'un tableau de « char » auquel est adjoint un entier pour indiquer sa longueur.

Le prototype de la fonction :

void TextToMat(const char *, char **, int, int);

Code :

```
void TextToMat(const char *msg, char **mat, int N, int M)
{
    int l = strlen(msg), x = 0;
    int i;
    int j;
    for ( i = 0; i < N; i++)
    {
        for (j = 0; j < M; j++)
        {
            if (x < l && !(msg[x] == ' ' || msg[x] == '\t'))
            {
                mat[i][j] = msg[x];
                x++;
            }
            else
                mat[i][j] = '\0';
        }
        while (x < l && (msg[x] == ' ' || msg[x] == '\t'))
        {
            x++;
        }
    }
    return;
}
```

Exécution :

```
*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  autres operations
 4.Quitter
Choisissez 1 ou 2 ou 3 ou 4 :  3

 1. Saisie du texte
 2. Cr├®ation de la matrice de mots
 3. Affichage de la matrice
 4. Cr├®ation de la structure de listes
 5. Affichage de la structure
 6.Ajout d'un mot
 7.Suppression d'un mot
Choisissez un numero compris entre 1 et 7 du menu :  2
la cr├®ation de la matrice pour ces mots : BELKHIR SELMA ACAD A
```

Affichage de la matrice : Après avoir saisi le texte composé d'une succession de mots séparés par des espaces, ces derniers seront récupérés dans une matrice dont l'affichage est le suivant.

Le prototype de la fonction :

void Affiche_MatC(char **, int, int);

Le code :

```
case 32: // ********************Affichage de la matrice**************************
{
    if (matrice3 != NULL)
        Affiche_MatC(matrice3, N, M);
    else
        printf("matrice pas encore creee\n");
}
break;
```

Exécution :

```
 1. Saisie du texte
 2. Cr├®ation de la matrice de mots
 3. Affichage de la matrice
 4. Cr├®ation de la structure de listes
 5. Affichage de la structure
 6.Ajout d'un mot
 7.Suppression d'un mot
Choisissez un numero compris entre 1 et 7 du menu :  3

| B | E | L | K | H | I | R |
| S | E | L | M | A |   |   |
| A | C | A | D |   |   |   |
| A |   |   |   |   |   |   |
```

Création de la structure de listes : Après avoir rangés les mots du texte dans la matrice ces derniers vont être organisés dans une structure.

Le prototype de la fonction :

void AjoutMot(VP *, const char *);

void InitStruct(VP *);

Le code :

```
void InitStruct(VP *listVP)
{
    int i;
    for ( i = 0; i < 26; i++)
    {
        listVP[i].head = NULL;
        listVP[i].car = (char)('A' + i);
    }
}
```

```
void AjoutMot(VP *listVP, const char *mot)
{
    int pos = (int)(toupper(mot[0]) - 'A');
    if (pos > 26)
        printf("il n'y a pas de liste pour cette lettre %c", mot[0]);
    else if (listVP[pos].head == NULL || strcmp(listVP[pos].head->mot, mot) > 0)
        listVP[pos].head = CreerUnElm(mot, listVP[pos].head);
    else
    {
        PList head = listVP[pos].head, next = head->next;
        while (next != NULL && (strcmp(next->mot, mot) > 0))
        {
            head = next;
            next = next->next;
        }
        head->next = CreerUnElm(mot, next);
    }
}
```

Exécution :

```
Choisissez 1 ou 2 ou 3 ou 4 :  3

 1. Saisie du texte
 2. Cr├®ation de la matrice de mots
 3. Affichage de la matrice
 4. Cr├®ation de la structure de listes
 5. Affichage de la structure
 6.Ajout d'un mot
 7.Suppression d'un mot
Choisissez un numero compris entre 1 et 7 du menu :  4
```

Affichage de la structure : La structure créé doit être affichée comme suit .

Le prototype de la fonction :

void afficherStructure(VP *);

Le code :

```c
void afficherStructure(VP *listVP)
{
    int i;
    for ( i = 0; i < 26; i++)
    {
        printf("affichage de list de carracter %c : ", listVP[i].car);
        PList head = listVP[i].head;
        if (head != NULL)
        {
            while (head->next != NULL)
            {
                printf("%s | ", head->mot);
                head = head->next;
            }
            printf("%s\n", head->mot);
        }
        else
            printf("[VIDE]\n", listVP[i].car);
    }
}
```

Exécution :

```
Choisissez 1 ou 2 ou 3 ou 4 :  3

 1. Saisie du texte
 2. Cr├®ation de la matrice de mots
 3. Affichage de la matrice
 4. Cr├®ation de la structure de listes
 5. Affichage de la structure
 6.Ajout d'un mot
 7.Suppression d'un mot
Choisissez un numero compris entre 1 et 7 du menu :  5
affichage de list de carracter A : A | ACAD
affichage de list de carracter B : BELKHIR
affichage de list de carracter C : [VIDE]
affichage de list de carracter D : [VIDE]
affichage de list de carracter E : [VIDE]
affichage de list de carracter F : [VIDE]
affichage de list de carracter G : [VIDE]
affichage de list de carracter H : [VIDE]
affichage de list de carracter I : [VIDE]
affichage de list de carracter J : [VIDE]
affichage de list de carracter K : [VIDE]
affichage de list de carracter L : [VIDE]
affichage de list de carracter M : [VIDE]
affichage de list de carracter N : [VIDE]
affichage de list de carracter O : [VIDE]
affichage de list de carracter P : [VIDE]
affichage de list de carracter Q : [VIDE]
affichage de list de carracter R : [VIDE]
affichage de list de carracter S : SELMA
affichage de list de carracter T : [VIDE]
affichage de list de carracter U : [VIDE]
affichage de list de carracter V : [VIDE]
affichage de list de carracter W : [VIDE]
affichage de list de carracter X : [VIDE]
affichage de list de carracter Y : [VIDE]
affichage de list de carracter Z : [VIDE]
```

Ajout d'un mot : Après avoir affiché le contenu de la structure, on nous demande d'ajouter un mot dans la structure et afficher le nouveau contenu

Le prototype de la fonction :

void AjoutMot(VP *listVP, const char *mot)

void afficherStructure(VP *listVP)

Le code :

```
void AjoutMot(VP *listVP, const char *mot)
{
    int pos = (int)(toupper(mot[0]) - 'A');
    if (pos > 26)
        printf("il n'y a pas de liste pour cette lettre %c", mot[0]);
    else if (listVP[pos].head == NULL || strcmp(listVP[pos].head->mot, mot) > 0)
        listVP[pos].head = CreerUnElm(mot, listVP[pos].head);
    else
    {
        PList head = listVP[pos].head, next = head->next;
        while (next != NULL && (strcmp(next->mot, mot) > 0))
        {
            head = next;
            next = next->next;
        }
        head->next = CreerUnElm(mot, next);
    }
}
```

Exécution :

```
Choisissez un numero compris entre 1 et 7 du menu :  6
ins├®rer le mot que vous voulez l'ajouter : KADER

*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  autres operations
 4.Quitter
Choisissez 1 ou 2 ou 3 ou 4 :  3

 1. Saisie du texte
 2. Cr├®ation de la matrice de mots
 3. Affichage de la matrice
 4. Cr├®ation de la structure de listes
 5. Affichage de la structure
 6.Ajout d'un mot
 7.Suppression d'un mot
Choisissez un numero compris entre 1 et 7 du menu :  5
affichage de list de carracter A : A | ACAD
affichage de list de carracter B : BELKHIR
affichage de list de carracter C : [VIDE]
affichage de list de carracter D : [VIDE]
affichage de list de carracter E : [VIDE]
affichage de list de carracter F : [VIDE]
affichage de list de carracter G : [VIDE]
affichage de list de carracter H : [VIDE]
affichage de list de carracter I : [VIDE]
affichage de list de carracter J : [VIDE]
affichage de list de carracter K : KADER
affichage de list de carracter L : [VIDE]
affichage de list de carracter M : [VIDE]
affichage de list de carracter N : [VIDE]
affichage de list de carracter O : [VIDE]
affichage de list de carracter P : [VIDE]
affichage de list de carracter Q : [VIDE]
affichage de list de carracter R : [VIDE]
affichage de list de carracter S : SELMA
affichage de list de carracter T : [VIDE]
affichage de list de carracter U : [VIDE]
affichage de list de carracter V : [VIDE]
affichage de list de carracter W : [VIDE]
affichage de list de carracter X : [VIDE]
affichage de list de carracter Y : [VIDE]
affichage de list de carracter Z : [VIDE]
```

Suppression d'un mot : Après avoir affiché le contenu de la structure, on nous demande de supprimer un mot de la structure et afficher le nouveau contenu

Le prototype de la fonction :

void SupprimerMot(VP *listVP, const char *mot)

void afficherStructure(VP *listVP)

Le code :

```
void SupprimerMot(VP *listVP, const char *mot)
{
    int pos = (int)(toupper(mot[0]) - 'A');
    if (pos > 26)
        printf("il n'y a pas de liste pour cette lettre %c", mot[0]);
    else if (listVP[pos].head == NULL)
        printf("this list '%c' is empty", toupper(mot[0]));
    else if (strcmp(listVP[pos].head->mot, mot) == 0)
    {
        PList p = listVP[pos].head;
        listVP[pos].head = listVP[pos].head->next;
        free(p);
    }
    else
    {
        PList head = listVP[pos].head, next = head->next;
        while (next != NULL && (strcmp(next->mot, mot) != 0))
        {
            head = next;
            next = next->next;
        }
        head->next = next->next;
        free(next);
    }
}
```

Exécution :

```
Choisissez un numero compris entre 1 et 7 du menu :  7
ins├®rer le mot que vous voulez le supprimer : ACAD

*** MENU: ***
 1.  Operations arithmetiques sur les matrices
 2.  Autres operations sur les matrices
 3.  autres operations
 4.Quitter
Choisissez 1 ou 2 ou 3 ou 4 :  3

 1. Saisie du texte
 2. Cr├®ation de la matrice de mots
 3. Affichage de la matrice
 4. Cr├®ation de la structure de listes
 5. Affichage de la structure
 6.Ajout d'un mot
 7.Suppression d'un mot
Choisissez un numero compris entre 1 et 7 du menu :  5
affichage de list de carracter A : A
affichage de list de carracter B : BELKHIR
affichage de list de carracter C : [VIDE]
affichage de list de carracter D : [VIDE]
affichage de list de carracter E : [VIDE]
affichage de list de carracter F : [VIDE]
affichage de list de carracter G : [VIDE]
affichage de list de carracter H : [VIDE]
affichage de list de carracter I : [VIDE]
affichage de list de carracter J : [VIDE]
affichage de list de carracter K : KADER
affichage de list de carracter L : [VIDE]
affichage de list de carracter M : [VIDE]
affichage de list de carracter N : [VIDE]
affichage de list de carracter O : [VIDE]
affichage de list de carracter P : [VIDE]
affichage de list de carracter Q : [VIDE]
affichage de list de carracter R : [VIDE]
affichage de list de carracter S : SELMA
affichage de list de carracter T : [VIDE]
affichage de list de carracter U : [VIDE]
affichage de list de carracter V : [VIDE]
affichage de list de carracter W : [VIDE]
affichage de list de carracter X : [VIDE]
affichage de list de carracter Y : [VIDE]
affichage de list de carracter Z : [VIDE]
```

Conclusion:

De notre point de vue, ce projet est objectivement une réussite. Tous les objectifs principaux ont été atteints : nous disposons de la solution de plusieurs problèmes avec différentes méthodes

Nous sommes globalement satisfaits de ce que nous avons réalisé.