

Université M'Hamed BOUGARA - Boumerdes
Faculté des sciences – Département d'Informatique

Algorithmique et structure de données

Présentée par

Iza lyla

lylaiza@yahoo.fr

Contenu du module

- ▶ Introduction
- ▶ Historique
- ▶ Généralités et concepts de base
- ▶ Algorithmes séquentiels simple
- ▶ Les structures conditionnelles (en langage algorithmique et en C)
- ▶ Les boucles (en langage algorithmique et en C)
- ▶ **Les tableaux et les chaînes de caractères**
- ▶ Les types personnalisés

Tableaux bidimensionnels (matrice)

Plan du cours 6: Les tableaux bidimensionnels (les matrices)

- ▶ Introduction sur les matrices
- ▶ Déclaration et mémorisation
- ▶ Opérations sur les matrices
- ▶ Matrices particulières

Introduction sur les tableaux bidimensionnels (matrice) (1)

L'algorithmique ou la programmation en générale, nous offre la possibilité de déclarer et d'utiliser des tableaux dans lesquels les valeurs ne sont pas repérées par un seul indice comme les tableaux à une seule dimension, mais par deux indices (coordonnées), le premier indice sert à représenter les lignes et le second indice pour les colonnes.

un tableau a deux dimensions peut être interpréter comme un tableau à une dimension dont le nombre d'éléments est L où chaque composante (élément de ce tableau) est un tableau (unidimensionnel) de dimension C tel que :

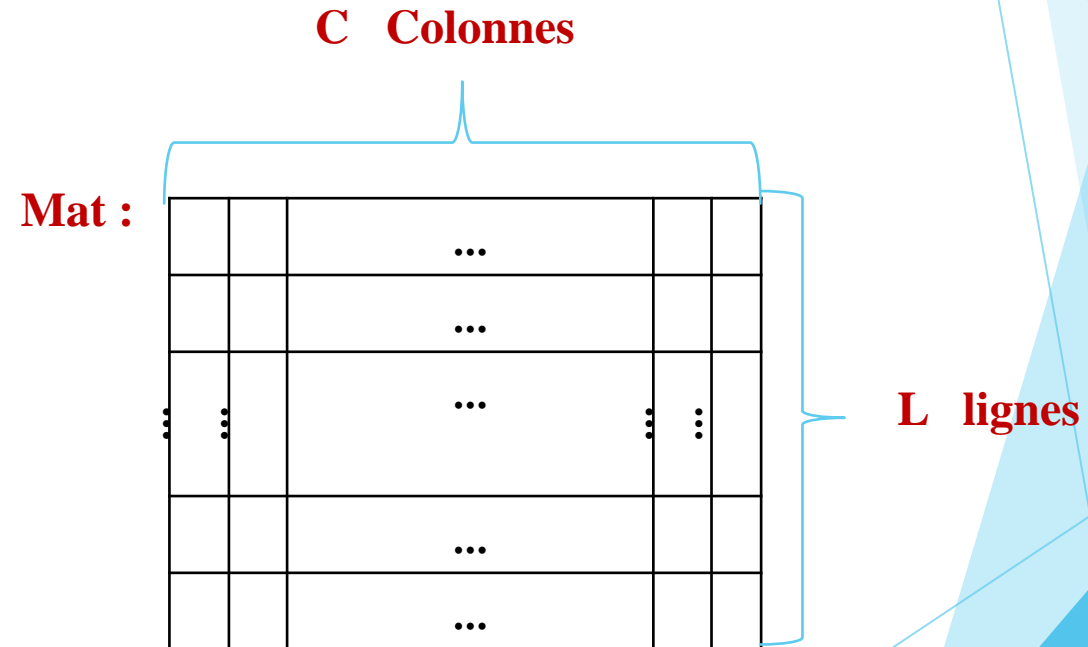
L est le nombre de lignes et C est le nombre de colonnes du tableau. L et C sont alors les deux dimensions du tableau. Un tableau à deux dimensions contient donc $L * C$ composantes (éléments).

Une matrice ou tableau à deux dimensions est une structure de données composée d'un ensemble de cases mémoires, où :

- ▶ Chaque matrice doit avoir un nom unique. Par exemple, la matrice T .
- ▶ Une case mémoire représente un élément de la matrice.
- ▶ Tous les éléments de la matrice sont de même type.
- ▶ Le nom de chaque élément est composé du nom de la matrice avec deux indices (la ligne et la colonne). Ces derniers indiquent la position de l'élément dans la matrice. Par exemple, l'élément $T[3, 2]$.
- ▶ Les indices sont indiqués entre crochets.

Tableaux bidimensionnels (matrice)(2)

Le schéma ci-dessous représente un tableau à deux dimensions.



Exemple

Considérons un tableau notes à une dimension pour mémoriser les notes de 20 étudiants d'un groupe dans un test :

```
float note[20] = {15,...,12,10,5};
```

Pour mémoriser les notes des étudiants dans les 10 tests d'un semestre, nous pouvons rassembler plusieurs de ces tableaux, unidimensionnels dans un tableau notes à deux dimension : float notes [10][20] (ou 10 est le nombre de lignes et 20 est le nombre de colonnes) tel que :

Dans une lignes nous retrouvons les notes de tous les étudiants dans un test. Dans une colonne, nous retrouvons toutes les notes d'un étudiant. Le schéma suivant représente un tableau à deux dimensions qui présente deux entrées : *E_i* pour les étudiant et l'entrée *note_testi* pour les note des test de chaque étudiant

	<i>E1</i>	<i>E2</i>		<i>En</i>
<i>note_test1</i>				
<i>note_test2</i>				
<i>note_testn</i>				

Déclaration et mémorisation

La déclaration d'un tableau à deux dimensions ou matrice se fait donc par un *identificateur*, le mot clé *tableau*, *le nombre de lignes* et *le nombre de colonnes* du tableau et le *type de données* qu'il contient.

La syntaxe de la déclaration d'une matrice est :

1. En algorithmique :

```
Var <nom_matrice> : tableau [nb_lignes,nb_colonnes] de <type >;
```

Exemple :

```
Var notes : tableau [10][20] de reel;
```

2. En c :

```
<type> <nom_matrice> [nb_lignes][nb_colonnes];
```

Exemple :

```
float notes[10][20];
```

Un élément d'une matrice est désigné par le nom du tableau (ou matrice) et ses indices de lignes et colonnes respectivement, pouvant être manipulé exactement comme une variable.

Exemple : note [2][3] en c et note [2,3] en algorithmique ; qui est un élément de la matrice note qui se trouve à la croisement de la ligne 2 et de la colonne 3.

Remplissage d'une matrice

Le remplissage d'un tableau à deux dimensions ou matrice consiste à entrer les valeurs qu'elle doit contenir. Pour ce faire il existe trois méthodes :

1. Par l'initialisation;
2. Par l'instruction de lecture qui consiste à entrer directement les éléments du tableau à partir du clavier ou ;
3. Par le calcul de ses éléments à partir d'une expression,

Opérations sur les matrices

Remplissage d'une matrice par initialisation

1. Initialisation

Lors de la déclaration d'un tableau à deux dimensions (comme pour les tableaux à une dimension), on peut initialiser les composants du tableau en indiquant la liste des valeurs respectives entre accolades. Où les composantes de chaque ligne du tableau sont comprise entre accolades.

Exemple:

```
Int Mat [3][5] = { {1,2,3},{6,5,4},{4,8,7},{3,2,5}};
```

Lors de l'initialisation, les valeurs sont affectées ligne par ligne en passant de gauche à droite. Nous ne devons pas nécessairement indiquer toutes les valeurs : Les valeurs manquantes seront initialisées à zéro. Il est cependant défendu d'indiquer trop de valeurs pour un tableau.

Exemple :

```
int mat [3][2]={ {1,2,3}}; les deux autres lignes seront initialisées à zéro.
```

```
int mat1[2][3] = { {1},{3}}; les deux autre colonnes seront initialisées également à zéro.
```

```
Int mat [2][3] = { {2,3,6,1}}; erreurs ( car le nombre de colonnes est 3 et dans l'initialisation est de 4).
```

2. Réserve automatique :

Si le nombre de lignes n'est pas indiquée explicitement lors de l'initialisation, l'ordinateur réserve automatiquement le nombre d'octets nécessaires.

Le remplissage d'une matrice par lecture de ses composants

Exemple 2 :

```
Algorithme remplissage_lecture;  
var mat : tableau [3,5] d'entier;  
    i,j : entier;  
debut  
pour (i allant de 1 à 3)faire  
    pour (j allant de 1 à 5) faire  
        lire (mat[i,j]);  
    fait;  
fait;  
Fin.
```



En c

```
#include <stdio.h>  
int main ()  
{  
    int mat [3][5], i,j;  
    for(i=1;i<=3;i++)  
        for (j=1;j<=5;j++)  
            scanf ("%d", &mat[i][j]);  
    return 0;  
}
```

Le remplissage d'une matrice par le calcul de ses éléments

Exemple 1 : dans ce cas la valeur de `mat [i][j]` doit être au fonction de `i` et `j`.

```
Algorithme remplissage_calcul;  
var mat : tableau [3,5] d'entier;  
  i,j : entier;  
debut  
  pour (i allant de 1 à 3)faire  
    pour (j allant de 1 à 5) faire  
      mat[i,j] = i+j*2;  
    fait;  
  fait;  
Fin.
```

*Indice des
lignes*

En c

```
#include <stdio.h>  
int main ()  
{  
  int mat [3][5], i,j;  
  for(i=1;i<=3;i++)  
    for (j=1;j<=5;j++)  
      mat [i][j] = i+j*2;  
  
  return 0;  
}
```

*Indice des
colonnes*

Accès aux composants (éléments) d'une matrice

On accède (en lecture ou en écriture) aux éléments d'une matrice (ou d'un tableau à deux dimensions) en utilisant la syntaxe suivante :

Syntaxe en algorithmique :

<nom_mat> [indice_ligne,indice_colonne];

Exemple :

Var mat : tableau [2,5] d'entier ;

Mat [1,3] : représente l'élément dont le numéro de ligne est 1 et le numéro de colonne est 3.

Syntaxe en c :

<nom_mat> [indice_lignes][indice_colonnes];

Exemple :

Int Mat [2][5];

Int x ;

X=mat [1][3] : signifie que x prend la valeur de l'intersection de la première ligne et de la troisième colonnes.

Affichage d'un tableau bidimensionnel (matrice)

L'écriture d'une matrice est l'opération qui permet d'afficher son contenu à l'écran, pour cela il faut parcourir les lignes et les colonnes pour écrire les éléments.

1. L'action qui permet l'affichage en algorithmique est :

```
Pour (i allant de 1 à n) faire  
    pour (j allant de 1 à m) faire  
        écrire (mat[i,j]; fait; fait;
```

2. L'action qui permet l'affichage en c est :

```
for (i=1;i<=n; i++){  
    for (j=1;j<=m ; j++){  
        printf ("" %d'',&mat[i][j]); }}
```


Autres opérations sur les matrices(1)

rechercher un élément dans une matrice en algorithmique :

La recherche dans un tableau à deux dimensions s'effectue généralement en utilisant un parcours séquentiel ;

```
Algorithme recherche ;  
Var mat: tableau [4,5] d'entier;  
Var b : booleen  
    elem,n, m : entier;  
Begin  
Ecrire ('remplir la matrice');  
Pour (i allant de 1 a 4) faire  
    Pour (j allant de 1 a 5) faire  
        Lire (mat[i,j]; fait; fait;  
    Lire (elem);  
    b ← faux;  
    Pour(i de 1 à n) faire  
        pour (j de 1 à m) faire  
            Si elem =mat [i,j] alors  
                debut  
                    ecrire ('element trouve');  
                    b→ vrai ;  
            Finsi;  
        Si b = faux alors  
            Ecrire ('l element n existe pas dans la matrice');  
    Fin.
```

rechercher un élément dans une matrice en C :

```
#include <stdio.h>
#include<stdbool.h>
int main ()
{
int mat[4][5],elem,i,j ;
bool  b;
printf ("remplir la matrice \n");
for (i=1; i<=3;i++)
for (j=1; j<=2;j++)
scanf ("%d",&mat[i][j]);
printf ("saisir la valeur a chercher \n");
scanf("%d",&elem);
b = false;
for (i=1; i<=3;i++)
for (j=1; j<=2;j++)
if (elem ==mat [i][j])
    b = true ;
    if (b == false)
printf ("l element n existe pas dans la matrice"); else printf ("element trouve");
return 0;
}
```

Autres opérations sur les matrices (2)

2. **Trace d'une matrice :** La trace d'une matrice carrée est la somme des éléments de la diagonale.

$S = T[0, 0] + T[1, 1] + T[2, 2] + T[3, 3] = 12 + 0 + -1 + 6 = 17$ Donc, la somme est donnée par l'équation suivante: $Tr = \sum_{i=1}^3 T[i, i];$

▶ Algorithme qui permet de calculer la trace d'une matrice

pour calculer la trace d'une matrice on doit avoir :

- ▶ les dimensions de la matrice. Comme la matrice est carrée, une seule dimension est suffisante puisque $M = N$.
- ▶ Remplir la matrice
- ▶ Parcourir la matrice diagonalement, en n'utilisant qu'un seul indice, pour calculer la trace.
- ▶ Afficher la trace. Ceci nous donnera l'algorithme suivant :

```
Algorithme trace ;
  Var tr : entier ;
n, i , j : Entier ;
mat : Tableau [ 50,50 ] d'entier ;
Début
Tant que ((n>=1) et (n<=50)) faire
  Ecrire ( ' Nombre de lignes de T <= 50 ? ' ) ;
  Lire ( n ) ;
fait;
Pour i allant de 1 à n Faire
  Pour j allant de 1 à n Faire
    Début
    Ecrire ( ' Donnez la valeur de T[' ,i,',',j,']= ' ) ;
    Lire (T[i,j]) ;
  Finpour;
/*/ Une seule boucle est suffisante pour parcourir la diagonale
  tr  $\leftarrow$  0;
Pour i allant de 1 à n Faire
  Tr  $\leftarrow$  tr + mat [ i , i ] ;
  Ecrire ( ' La Trace = ' , tr ) ;
Fin.
```

Les matrices particulière (1)

1. **Les matrices carrées :** une matrice est dite carrée si le nombre de lignes égale au nombre de colonnes.

Exemple : soit mat la matrice carrée tel que :

Var mat : tableau [2,2] d'entier ;

2. **Matrice identité :** est une matrice où la valeur, de tous les éléments de la diagonale, est égale à (1) et le reste des éléments égaux à zéro (0).

Exemple : soit mat la matrice carrée tel que: var mat : tableau [3,3] d'entier ; $\text{mat} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

L'algorithme qui permet de construire une matrice identité

Pour réaliser cette algorithme on doit avoir:

- ▶ les dimensions de la matrice. Comme la matrice est carrée, une seule dimension est suffisante puisque $M = N$.
- ▶ Remplir la matrice par le calcul de ses éléments en affectant un (1) aux éléments de la diagonale ($\text{Mat}[i,i]$) et zéro (0) aux autres éléments.
- ▶ Afficher la matrice Ceci nous donnera l'algorithme suivant :

```
Algorithme valeur ;
Var n , i , j : Entier ;
T : Tableau [50,50] de réel ;
  debut
Tantque ((n>=1) et (n<=50)) faire
Ecrire ('Nombre de lignes de T <= 50 ?');
  Lire (n)
fintantque;
  Pour (i allant de 1 à n) Faire
  debut
Pour j 1 à n Faire
  T[ i , j ]=0 ;
  T[ i , i ]=1 ;
  Fin ;
  Pour (i allant de 1 à n) Faire
  Pour (j allant de 1 à n) Faire
  Ecrire ('T[',i,',',j,',']= ',T[i,j]);finpour;finpour;
  Fin .
```

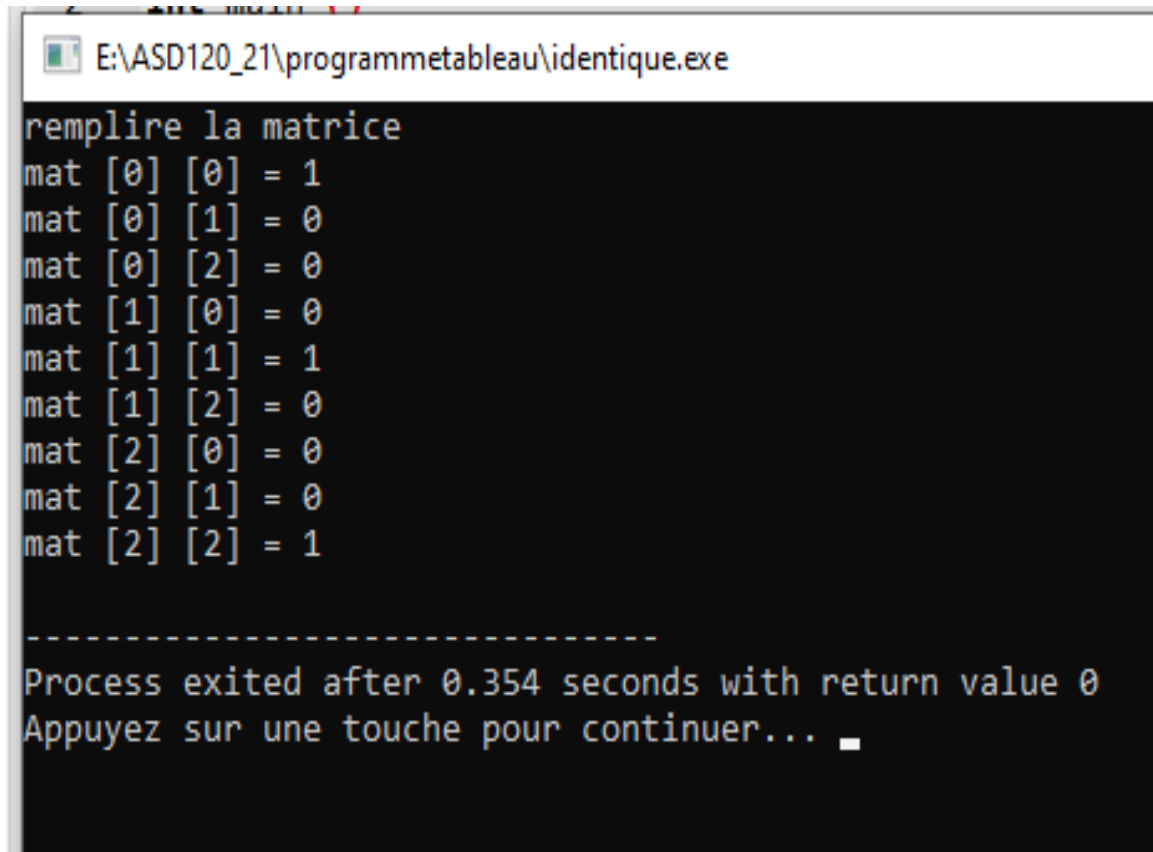
Le programme en c qui permet de construire une matrice identité est :

```
#include<stdio.h>

int main ()
{   int i,j;
    int mat[3][3];
    printf ("remplire la matrice \n");
    for (int i=0;i<3;i++){
    for (int j=0;j<3;j++){
        mat[i][j]=0;
        mat[i][i]=1;
    }

    }
    for (i=0;i<3;i++){
    for (j=0;j<3;j++){
        printf("mat [%d] [%d] = %d \n",i,j,mat[i][j]);
    }
    }
    return 0;
}
```

Lors de l'exécution, le programme donne les résultats suivant :



```
E:\ASD120_21\programmetableau\identique.exe  
remplire la matrice  
mat [0] [0] = 1  
mat [0] [1] = 0  
mat [0] [2] = 0  
mat [1] [0] = 0  
mat [1] [1] = 1  
mat [1] [2] = 0  
mat [2] [0] = 0  
mat [2] [1] = 0  
mat [2] [2] = 1  
  
-----  
Process exited after 0.354 seconds with return value 0  
Appuyez sur une touche pour continuer... █
```


Les matrices particulière (2)

3. **Transposé d'une matrice :** Pour réaliser une matrice transposée, il suffit d'inter-changer entre les lignes et les colonnes :

L'algorithme qui permet de construire une matrice identité

Pour réaliser cette algorithme on doit avoir:

- ▶ La première ligne devient la première colonne, la deuxième ligne devient la deuxième colonne, et ainsi de suite.
- ▶ Donc, au fur et à mesure que la matrice A est remplie, on remplit la matrice B en inversant les indices. $B[j, i] = A[i, j]$.

```
Algorithme transposé ;
Var n , m , i , j : Entier ;
  A : Tableau [50,50] d entier ;
  trA : tableau [50,50] d entier;
Début
  repeter
  Ecrire (' Nombre de lignes de T <= 50 ?');
  Lire ( n ) ;
jusqu a ((n<1) et (n>50));
  repeter
  Ecrire ('Nombre de colonnes de T <= 50 ?') ;
  Lire ( m ) ;
Jusqu a ((m<1) et (m>50));
  Pour (i allant de 1 à n) Faire
Pour (j allant de 1 à m ) Faire
  Début
Ecrire (' Donnez la valeur de A [ ' , i , ' , ' , j , ' ] = ' ) ;
Lire ( A [ i , j ] ) ;
  trA [ j , i ] = A [ i , j ] ;
  Finpour ;
  Pour i 1 à m Faire
  Pour j 1 à n Faire
    Ecrire ( ' B [ ' , i , ' , ' , j , ' ] = ' , B [ i , j ] ) ;
Fait; fait;
Fin.
```

Le programme c qui permet calculer la transposé d'une matrice est :

```
#include <stdio.h>

int main()
{
int n , m , i , j ,mat[50][50], t1[50][50] ;
do {
printf("Nombre de lignes de T <= 50 ? \n");
scanf ("%d", &n);
}while ((n<1) & (n>50));
do {
printf ("Nombre de colonnes de T <= 50 ?\n") ;
scanf ("%d", &m);
} while ((m<1) & (m>50));
for(int i=0; i<n;i++) {
for (int j=0; j<m;j++)
{
printf ("Donnez la valeur de mat[%d][%d] = \n",i,j ) ;
scanf("%d", &mat[i][j]);
t1[j][i] = mat[i][j] ;
}}
for(i=0; i<m;i++)
for (j=0; j<n;j++)
printf("t1 [%d][%d] = %d \n",i,j,t1[i][j]) ;
return 0;
```

Les matrices particulière (3)

4. **La matrice symétrique :** est une matrice carré où : $\text{Mat}[i,j] = \text{Mat}[j,i]$
:

Exemple : $\text{Mat} = \begin{bmatrix} 3 & 3 & 2 \\ 3 & 4 & 1 \\ 2 & 1 & 2 \end{bmatrix}$

Pour réaliser l'algorithme qui permet de vérifier la symétrie, on doit placer un compteur qui compte le nombre de fois où on a $\text{Mat}[i,j] = \text{Mat}[j,i]$.

```
Algorithme symetrique;
Var mat : tableau [10,10] d entier;
Debut
Ecrire ('remplire la matrice');
Pour (i allant de 1 a 3)
Pour (j allant de 1 a 3)
Lire (mat[i,j]);
comp  $\leftarrow$  0;
Pour (i allant de 1 a 3)faire
Pour (j allant de 1 a 3) faire
si mat[i,j]=mat[j,i] alors
Comp  $\leftarrow$  comp + 1 ; finsi;
Fait;fait;
si (comp = 0) alors
Ecrire ('la matrice est symetrique');
Sinon
Ecrire ('la matrice mat n est pas symetrique');
```

Le programme c qui permet de vérifier la symétrie d'une matrice est :

```
#include <stdio.h>

int main () {
    int i, j , mat [10][10], comp;
    printf ("remplir la matrice \n");
    for (i=0;i<3;i++)
        for (j=0;j<3;j++)
            scanf ("%d", &mat[i][j]);
    comp = 0;
    for (i=0;i<3;i++){
        for (j=0;j<3;j++){
            if (mat[i][j]!=mat[j][i])
                comp++ ;
        }
    }
    if(comp == 0)
        printf("la matrice est symetrique\n");
    else
        printf ("la matrice mat n est pas symetrique\n");
    for (i=0;i<3;i++)
        for (j=0;j<3;j++)
            printf("mat [%d][%d] = %d \n", i,j , mat[i][j]);
    return 0 ; }
```

Lors de l'exécution, le programme donne les résultats suivant :

```
E:\ASD120_21\programmetableau\symetrie.exe
remplire la matrice
7
4
8
5
9
6
1
3
5
la matrice mat n est pas symetrique
mat [0][0] = 7
mat [0][1] = 4
mat [0][2] = 8
mat [1][0] = 5
mat [1][1] = 9
mat [1][2] = 6
mat [2][0] = 1
mat [2][1] = 3
mat [2][2] = 5
-----
Process exited after 6.696 seconds with return value 0
Appuyez sur une touche pour continuer...
```

```
E:\ASD120_21\programmetableau\symetrie.exe
remplire la matrice
3
3
2
3
4
1
2
1
2
la matrice est symetrique
mat [0][0] = 3
mat [0][1] = 3
mat [0][2] = 2
mat [1][0] = 3
mat [1][1] = 4
mat [1][2] = 1
mat [2][0] = 2
mat [2][1] = 1
mat [2][2] = 2
-----
Process exited after 11.33 seconds with return value 0
Appuyez sur une touche pour continuer... _
```