Initiation à l'algorithmique
Les tableaux

Mohamed MESSABIHI

mohamed.messabihi@gmail.com

Université de Tlemcen
Département d'informatique
1ère année MI

https://sites.google.com/site/informatiquemessabihi/

Introduction 1/3

Exercice 1

Écrire un programme permettant de saisir les 8 notes des examens du S1 d'un étudiant en MI puis les afficher avec sa moyenne.

Introduction 1/3

Exercice 1

Écrire un programme permettant de saisir les 8 notes des examens du S1 d'un étudiant en MI puis les afficher avec sa moyenne.

Solution

```
void main()
{
int note=0, somme= 0, moyenne=0, i=0;
for (i=0 ; i<10 ; i++)
{       printf ("donnez la note numero %d : ", i+1) ;
        scanf ("%d", &note) ;
        somme+=note;
}
moyenne = somme/10;
printf("La moyenne est : %d", moyenne);
}
```

Introduction 2/3

Exercice 2

Écrire un programme qui permet de saisir les moyennes de tous les 450 étudiants inscrit en MI puis calculer la moyenne générale.

Introduction 2/3

Exercice 2

Écrire un programme qui permet de saisir les moyennes de tous les 450 étudiants inscrit en MI puis calculer la moyenne générale.

Solution :

```
void main()
{
int M=0, somme= 0, moyenne=0, i=0;
for (i=0 ; i<450 ; i++)
{       printf ("donnez la moyenne numero %d : ", i+1) ;
        scanf ("%d", M) ;
        somme+=M;
}
moyenne = somme/450;
printf("La moyenne est : %d", moyenne);
}
```

Introduction 3/3

Exercice 3

Écrire un programme qui permet de saisir les moyennes de tous les 450 étudiants inscrit en MI puis de déterminer combien d'entre elles sont supérieures à la moyenne de la classe.

Introduction 3/3

Exercice 3

Écrire un programme qui permet de saisir les moyennes de tous les 450 étudiants inscrit en MI puis de déterminer combien d'entre elles sont supérieures à la moyenne de la classe.

Solution :

```
void main()
{
int M1, M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,.....
...
MoyenneGenerale = (M1+M2+M3+M4+M5+M6+M7+M8+M9+M10+M11+M12+
    M13+...)/450 ;
printf("La moyenne est : %d", Moyenne);

Euuuuuuuuuuuuhhhhh !!!!
...
}
```

Introduction 3/3

Exercice 3

Écrire un programme qui permet de saisir les moyennes de tous les 450 étudiants inscrit en MI puis de déterminer combien d'entre elles sont supérieures à la moyenne de la classe.

Solution :

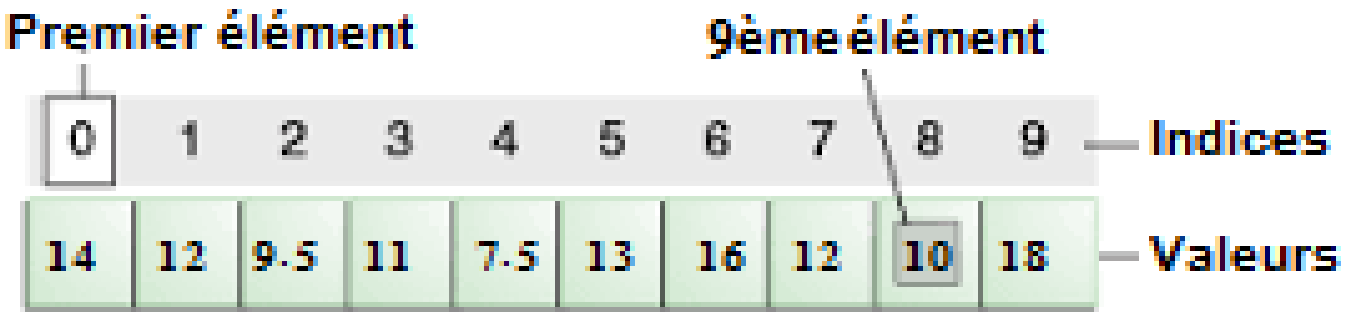
```
void main()
{
int M1, M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,.....
...
MoyenneGenerale = (M1+M2+M3+M4+M5+M6+M7+M8+M9+M10+M11+M12+
    M13+...)/450 ;
printf("La moyenne est : %d", Moyenne);

Euuuuuuuuuuuuhhhhh !!!!
...
}
```

N'y a t-il pas un moyen plus simple et plus élégant pour écrire ça ?

Bien sûr que si : notion de tableaux

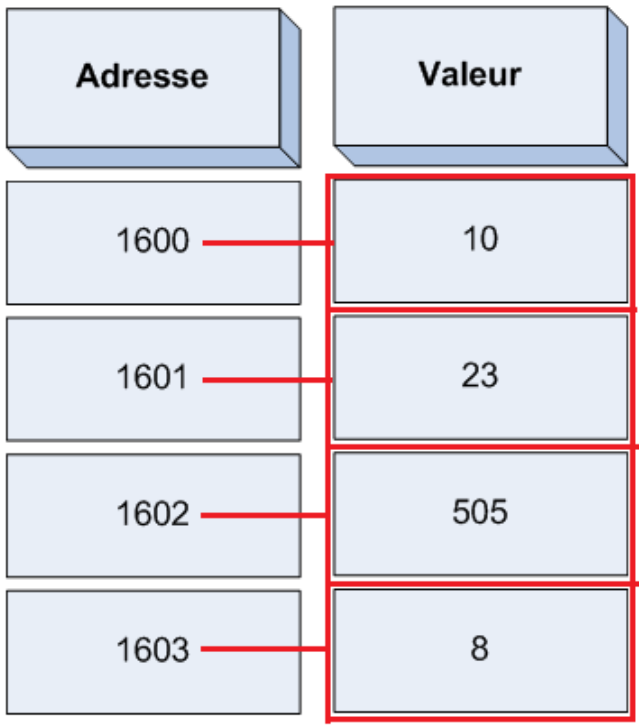
Définition

- Un ensemble de valeurs portant le même nom de variable et repérées par un nombre, s'appelle un tableau, ou encore une variable indicée
- Le nombre qui, au sein d'un tableau, sert à repérer chaque valeur s'appelle l'indice
- Chaque élément du tableau est désigné par le nom du tableau, suivi de l'indice de l'élément, entre crochets



Le tableau en mémoire

- Un schéma d'illustration d'un tableau de 4 cases en mémoire qui commence à l'adresse 1600.
- Lorsqu'un tableau est créé, il prend un espace contigu en mémoire : les cases sont les unes à la suite des autres.
- Toutes les cases d'un tableau sont du même type. Ainsi, un tableau de int contiendra uniquement des int, et pas autre chose.



Déclaration d'un tableau et l'accès à ses éléments

Exemple

```
int notes[10];
...
notes[0] = 14;
notes[1] = 8;
notes[2] = 12;
notes[3] = 17;
...
```

1. Il suffit donc de rajouter entre crochets le nombre de cases que vous voulez mettre dans votre tableau. pas de limite.
2. Pour accéder à chaque élément du tableau, il faut écrire le nom du tableau suivi de l'indice de l'élément concerné entre crochets.

Attention

Un tableau commence à l'indice numéro 0 ! Notre tableau **notes** de 10 **int** a donc les indices 0, 1, 2,... et 9. Il n'y a pas d'indice 10 dans un tableau de 10 cases ! C'est une source d'erreurs très courantes pour les débutants.

Déclaration d'un tableau et l'accès à ses éléments

Exemple

```
int notes[10];
...
notes[0] = 14;
notes[1] = 8;
notes[2] = 12;
notes[3] = 17;
...
```

1. Il suffit donc de rajouter entre crochets le nombre de cases que vous voulez mettre dans votre tableau. pas de limite.
2. Pour accéder à chaque élément du tableau, il faut écrire le nom du tableau suivi de l'indice de l'élément concerné entre crochets.

Attention

Un tableau commence à l'indice numéro 0 ! Notre tableau **notes** de 10 **int** a donc les indices 0, 1, 2,... et 9. Il n'y a pas d'indice 10 dans un tableau de 10 cases ! C'est une source d'erreurs très courantes pour les débutants.

Initialiser un tableau 1/2

- Il faut savoir qu'il existe une autre façon d'initialiser un tableau un peu plus automatisée en C.
- Elle consiste à placer les valeurs une à une entre accolades, séparées par des virgules.

Exemple :

```
void main()
{
    int tableau[4] = {0, 0, 0, 0};

    double A[3] = {1.6, 3.78, 14.892};
}
```

Initialiser un tableau 2/2

- on peut également définir les valeurs des premières cases du tableau, toutes celles que vous n'aurez pas renseignées seront automatiquement mises à 0.

Exemple :

```
void main()
{
    int tab1[4] = {0, 0, 0, 0}; // 0, 0, 0, 0
    int tab2[6] = {10, 23};     // 10, 23, 0, 0, 0, 0
    int tab3[4] = {0};          // 0, 0, 0, 0
    int tab4[5] = {1};          // 1, 0, 0, 0, 0, 0,
}
```

Attention

Dans le tableau tab4, on n'initialise pas toutes les cases à 1 : seule la première case sera à 1, toutes les autres seront à 0.

Initialiser un tableau 2/2

- on peut également définir les valeurs des premières cases du tableau, toutes celles que vous n'aurez pas renseignées seront automatiquement mises à 0.

Exemple :

```
void main()
{
    int tab1[4] = {0, 0, 0, 0}; // 0, 0, 0, 0
    int tab2[6] = {10, 23};     // 10, 23, 0, 0, 0, 0
    int tab3[4] = {0};          // 0, 0, 0, 0
    int tab4[5] = {1};          // 1, 0, 0, 0, 0, 0,
}
```

Attention

Dans le tableau tab4, on n'initialise pas toutes les cases à 1 : seule la première case sera à 1, toutes les autres seront à 0.

Parcourir un tableau

- Supposons qu'on veuille maintenant afficher les valeurs de chaque case du tableau.
- On aurait pu faire autant de printf qu'il y a de cases. Mais ce serait répétitif et lourd, et imaginez un peu la taille de notre code si on devait afficher le contenu de chaque case du tableau une à une !
- Le mieux est de se servir d'une boucle qui est très pratique pour parcourir un tableau :

Exemple

```c
#include<stdio.h>
int main(void)
{
    double notes[8]={14, 2, 15.5, 13, 4, 19, 17.5, 16};
    for (int i = 0 ; i < 8 ; i++)
        printf("%f\n", notes[i]);
}
```

Parcourir un tableau

- Supposons qu'on veuille maintenant afficher les valeurs de chaque case du tableau.
- On aurait pu faire autant de printf qu'il y a de cases. Mais ce serait répétitif et lourd, et imaginez un peu la taille de notre code si on devait afficher le contenu de chaque case du tableau une à une !
- Le mieux est de se servir d'une boucle qui est très pratique pour parcourir un tableau :

Exemple

```
#include<stdio.h>
int main(void)
{
    double notes[8]={14, 2, 15.5, 13, 4, 19, 17.5, 16};
    for (int i = 0 ; i < 8 ; i++)
        printf("%f\n", notes[i]);
}
```

Parcourir un tableau

- Supposons qu'on veuille maintenant afficher les valeurs de chaque case du tableau.
- On aurait pu faire autant de printf qu'il y a de cases. Mais ce serait répétitif et lourd, et imaginez un peu la taille de notre code si on devait afficher le contenu de chaque case du tableau une à une !
- Le mieux est de se servir d'une boucle qui est très pratique pour parcourir un tableau :

Exemple

```
#include<stdio.h>
int main(void)
{
    double notes[8]={14, 2, 15.5, 13, 4, 19, 17.5, 16};
    for (int i = 0 ; i < 8 ; i++)
        printf("%f\n", notes[i]);
}
```

Parcourir un tableau

- Supposons qu'on veuille maintenant afficher les valeurs de chaque case du tableau.
- On aurait pu faire autant de printf qu'il y a de cases. Mais ce serait répétitif et lourd, et imaginez un peu la taille de notre code si on devait afficher le contenu de chaque case du tableau une à une !
- Le mieux est de se servir d'une boucle qui est très pratique pour parcourir un tableau :

Exemple

```
#include<stdio.h>
int main(void)
{
    double notes[8]={14, 2, 15.5, 13, 4, 19, 17.5, 16};
    for (int i = 0 ; i < 8 ; i++)
        printf("%f\n", notes[i]);
}
```

Parcourir un tableau

```c
#include<stdio.h>
int main(void)
{
    double notes[8]={14, 2, 15.5, 13, 4, 19, 17.5, 16};
    for (int i = 0 ; i < 8 ; i++)
        printf("%f\n", notes[i]);
}
```

- La boucle parcourt le tableau à l'aide d'une variable appelée i (c'est le nom le plus souvent utilisé pour parcourir un tableau !).
- Notez qu'on peut mettre une variable entre crochets pour « parcourir » le tableau, c'est-à-dire accéder à ses valeurs.
- Attention à ne pas tenter d'afficher la valeur de notes[10] !Sinon vous aurez soit n'importe quoi, soit une belle erreur, l'OS interrompra votre programme car il aura tenté d'accéder à une adresse ne lui appartenant pas.

Parcourir un tableau

```c
#include<stdio.h>
int main(void)
{
    double notes[8]={14, 2, 15.5, 13, 4, 19, 17.5, 16};
    for (int i = 0 ; i < 8 ; i++)
        printf("%f\n", notes[i]);
}
```

- La boucle parcourt le tableau à l'aide d'une variable appelée i (c'est le nom le plus souvent utilisé pour parcourir un tableau !).
- Notez qu'on peut mettre une variable entre crochets pour « parcourir » le tableau, c'est-à-dire accéder à ses valeurs.
- Attention à ne pas tenter d'afficher la valeur de notes[10] !Sinon vous aurez soit n'importe quoi, soit une belle erreur, l'OS interrompra votre programme car il aura tenté d'accéder à une adresse ne lui appartenant pas.

Parcourir un tableau

```c
#include<stdio.h>
int main(void)
{
    double notes[8]={14, 2, 15.5, 13, 4, 19, 17.5, 16};
    for (int i = 0 ; i < 8 ; i++)
        printf("%f\n", notes[i]);
}
```

- La boucle parcourt le tableau à l'aide d'une variable appelée i (c'est le nom le plus souvent utilisé pour parcourir un tableau !).
- Notez qu'on peut mettre une variable entre crochets pour « parcourir » le tableau, c'est-à-dire accéder à ses valeurs.
- Attention à ne pas tenter d'afficher la valeur de notes[10] !Sinon vous aurez soit n'importe quoi, soit une belle erreur, l'OS interrompra votre programme car il aura tenté d'accéder à une adresse ne lui appartenant pas.

Et si on revenait à notre exercice de départ ?

Solution

```
#include <stdio.h>
int main(void)
{
int i, som, nbm ;
double moy ;
int t[450] ;
for (i=0 ; i<450 ; i++)
{printf ("donnez la note de l'etudiant numero %d : ", i+1) ;
 scanf ("%d", &t[i]) ;
}
for (i=0, som=0 ; i<450 ; i++) som += t[i] ;
moy = som / 450 ;
printf ("\n\n moyenne de la promo : %f\n", moy) ;
for (i=0, nbm=0 ; i<450 ; i++ )
          if (t[i] > moy) nbm++ ;
printf ("%d etudiants  ont plus que cette moyenne", nbm) ;
}
```

Passage de tableaux à une fonction

Exemple :

```
void saisir(int tab[], int tailleTab)
{
    int i;
    for (i = 0 ; i < tailleTab ; i++)
        scanf("%d", &tab[i]);
}
void afficher(int tab[], int tailleTab)
{
    int i;
    for (i = 0 ; i < tailleTab ; i++)
        printf("%d\n", tab[i]);
}
void main()
{
    int tableau[4] = {0};
    saisir(tableau, 4);
    afficher(tableau, 4);
}
```

Le nom d'un tableau est un pointeur

```c
int notes[10];
printf("%d", notes);
printf("%d", notes[0]);
printf("%d", *notes);
```

- Au début, on affiche l'adresse où se trouve le tableau **notes** : 1600
- En revanche, si on indique l'indice d'une case du tableau **notes** entre crochets, on obtient sa valeur : par exemple notes[0] = 14. De même pour les autres indices.
- Le nom du tableau **notes** est un pointeur vers la première case du tableau notes, on peut donc utiliser le symbole * pour connaître la valeur de la première case : *notes = 14
- Il est aussi possible d'obtenir la valeur de la seconde case avec *(notes + 1) (adresse de tableau + 1). notes[1] et *(notes + 1) sont donc équivalents.
- De manière générale, notes[0] est la valeur qui se trouve à l'adresse notes + 0 (1600). notes[1] est la valeur se trouvant à l'adresse notes + 1 (1601), et ainsi de suite.

Le nom d'un tableau est un pointeur

```c
int notes[10];
printf("%d", notes);
printf("%d", notes[0]);
printf("%d", *notes);
```

- Au début, on affiche l'adresse où se trouve le tableau **notes** : 1600
- En revanche, si on indique l'indice d'une case du tableau **notes** entre crochets, on obtient sa valeur : par exemple notes[0] = 14. De même pour les autres indices.
- Le nom du tableau **notes** est un pointeur vers la première case du tableau notes, on peut donc utiliser le symbole * pour connaître la valeur de la première case : *notes = 14
- Il est aussi possible d'obtenir la valeur de la seconde case avec *(notes + 1) (adresse de tableau + 1). notes[1] et *(notes + 1) sont donc équivalents.
- De manière générale, notes[0] est la valeur qui se trouve à l'adresse notes + 0 (1600). notes[1] est la valeur se trouvant à l'adresse notes + 1 (1601), et ainsi de suite.

Le nom d'un tableau est un pointeur

```
int notes[10];
printf("%d", notes);
printf("%d", notes[0]);
printf("%d", *notes);
```

- Au début, on affiche l'adresse où se trouve le tableau **notes** : 1600
- En revanche, si on indique l'indice d'une case du tableau **notes** entre crochets, on obtient sa valeur : par exemple notes[0] = 14. De même pour les autres indices.
- Le nom du tableau **notes** est un pointeur vers la première case du tableau notes, on peut donc utiliser le symbole ***** pour connaître la valeur de la première case : *notes = 14
- Il est aussi possible d'obtenir la valeur de la seconde case avec *(notes + 1) (adresse de tableau + 1). notes[1] et *(notes + 1) sont donc équivalents.
- De manière générale, notes[0] est la valeur qui se trouve à l'adresse notes + 0 (1600). notes[1] est la valeur se trouvant à l'adresse notes + 1 (1601), et ainsi de suite.

Le nom d'un tableau est un pointeur

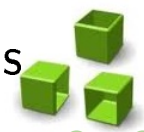
```c
int notes[10];
printf("%d", notes);
printf("%d", notes[0]);
printf("%d", *notes);
```

- Au début, on affiche l'adresse où se trouve le tableau **notes** : 1600
- En revanche, si on indique l'indice d'une case du tableau **notes** entre crochets, on obtient sa valeur : par exemple notes[0] = 14. De même pour les autres indices.
- Le nom du tableau **notes** est un pointeur vers la première case du tableau notes, on peut donc utiliser le symbole ***** pour connaître la valeur de la première case : *notes = 14
- Il est aussi possible d'obtenir la valeur de la seconde case avec *(notes + 1) (adresse de tableau + 1). notes[1] et *(notes + 1) sont donc équivalents.
- De manière générale, notes[0] est la valeur qui se trouve à l'adresse notes + 0 (1600). notes[1] est la valeur se trouvant à l'adresse notes + 1 (1601), et ainsi de suite.

Le nom d'un tableau est un pointeur

```c
int notes[10];
printf("%d", notes);
printf("%d", notes[0]);
printf("%d", *notes);
```

- Au début, on affiche l'adresse où se trouve le tableau **notes** : 1600
- En revanche, si on indique l'indice d'une case du tableau **notes** entre crochets, on obtient sa valeur : par exemple notes[0] = 14. De même pour les autres indices.
- Le nom du tableau **notes** est un pointeur vers la première case du tableau notes, on peut donc utiliser le symbole ***** pour connaître la valeur de la première case : *notes = 14
- Il est aussi possible d'obtenir la valeur de la seconde case avec *(notes + 1) (adresse de tableau + 1). notes[1] et *(notes + 1) sont donc équivalents.
- De manière générale, notes[0] est la valeur qui se trouve à l'adresse notes + 0 (1600). notes[1] est la valeur se trouvant à l'adresse notes + 1 (1601), et ainsi de suite.



Passage de tableaux à une fonction : pointeur 1/2

Exemple :

```
void saisir(int *tableau, int tailleTableau)
{
    int i;
    for (i = 0 ; i < tailleTableau ; i++)
        scanf("%d", &tableau[i]);
}
void afficher(int *tableau, int tailleTableau)
{
    int i;
    for (i = 0 ; i < tailleTableau ; i++)
        printf("%d\n", tableau[i]);
}
void main()
{
    int tab[4] = {0};
    saisir(tab, 4);
    afficher(tab, 4);
}
```

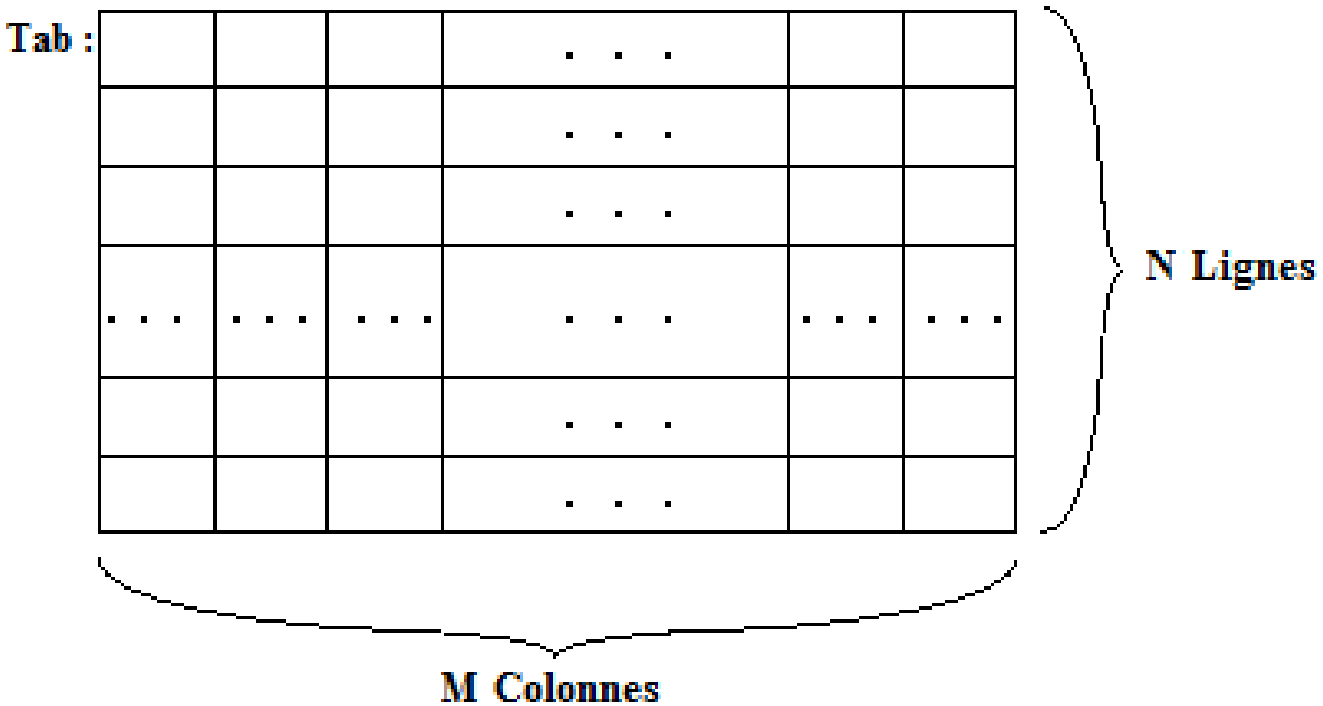
Passage de tableaux à une fonction : pointeur 2/2

Exemple :

```c
#include<stdio.h>
void saisir(int *tableau, int tailleTableau)
{
    int i;
    for (i = 0 ; i < tailleTableau ; i++)
        scanf("%d", tableau+i);
}
void afficher(int *tableau, int tailleTableau)
{
    int i;
    for (i = 0 ; i < tailleTableau ; i++)
        printf("%d\n", *(tableau+i));
}
void main()
{
    int tab[4] = {0};
    saisir(tab, 4);
    afficher(tab, 4);
}
```

Tableaux à deux dimensions

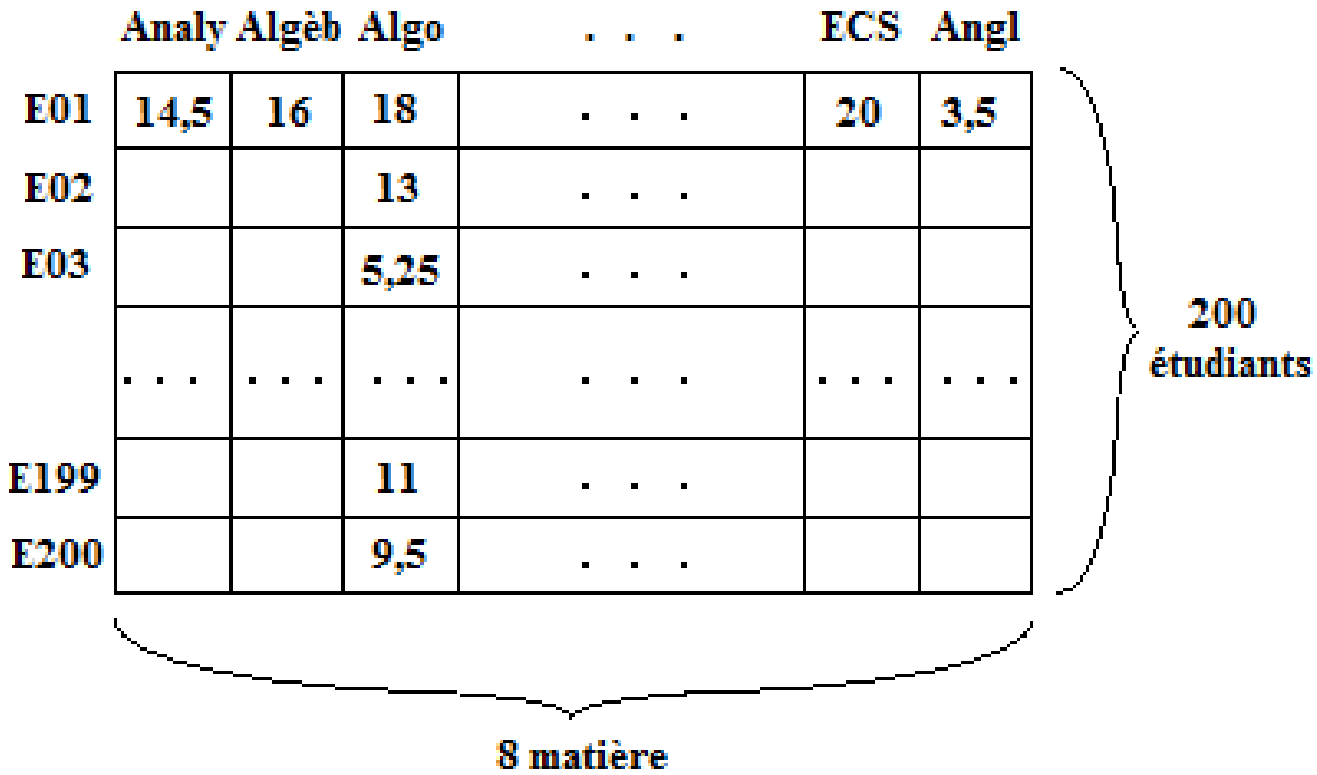
- Un tableau à deux dimensions est à interpréter comme un tableau de dimension N dont chaque élément est un tableau de dimension M.



- On appelle N le nombre de lignes et M le nombre de colonnes du tableau Tab. N et M sont alors les deux dimensions du tableau.
- Un tableau à deux dimensions contient donc N*M éléments.

Tableaux à deux dimensions

- Un tableau à deux dimensions est à interpréter comme un tableau de dimension N dont chaque élément est un tableau de dimension M.



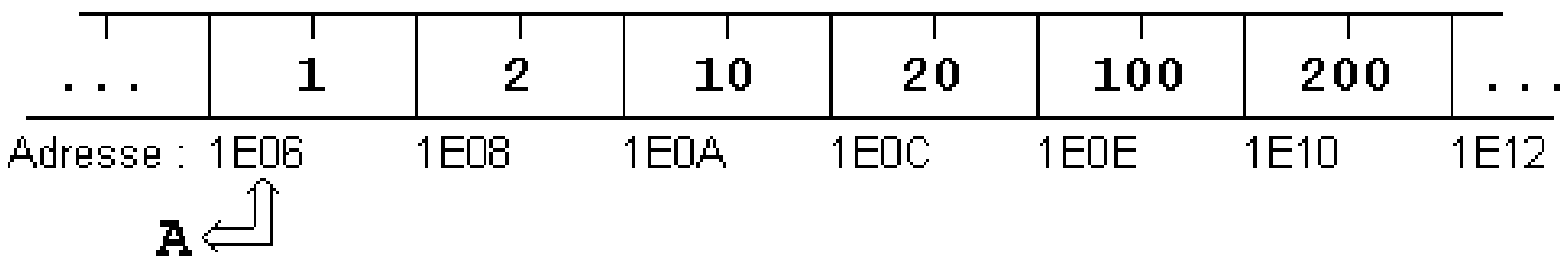
- On appelle N le nombre de lignes et M le nombre de colonnes du tableau Tab. N et M sont alors les deux dimensions du tableau.
- Un tableau à deux dimensions contient donc N*M éléments.

Déclaration et mémorisation des tableaux à deux dimensions

- Comme pour les tableaux à une dimension, le nom d'un tableau est le représentant de l'adresse du premier élément du tableau (c.-à-d. l'adresse de la première ligne du tableau).

```
int A[3][2] =    {{1,   2  },
                  {10,  20 },
                  {100, 200}};
```

- Les éléments d'un tableau à deux dimensions sont stockées ligne par ligne dans la mémoire.



Initialisation des tableaux à deux dimensions

- Lors de la déclaration d'un tableau, on peut initialiser les éléments du tableau, en indiquant la liste des valeurs respectives entre accolades.
- À l'intérieur de la liste, les éléments de chaque ligne du tableau sont encore une fois comprises entre accolades.
- Pour améliorer la lisibilité des programmes, on peut indiquer les composantes dans plusieurs lignes.

```
int  A[3][10] ={{ 0,10,20,30,40,50,60,70,80,90},
                {10,11,12,13,14,15,16,17,18,19},
                { 1,12,23,34,45,56,67,78,89,90}};
double B[3][2] = {{-1.05,    -1.10  },
                  {86e-5,     87e-5 },
                  {-12.5E4, -12.3E4}};
```

- 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 par zéro.

Accès aux éléments d'un tableau à deux dimensions

Considérons un tableau A de dimensions N et M.

```
int A[][10] = {{ 0,10,20,30,40,50,60,70,80,90},
               {10,11,12,13,14,15,16,17,18,19},
               { 1,12}};
```

- Les indices du tableau varient de 0 à N-1, respectivement de 0 à M-1.
- L'élément de la $i^{ième}$ ligne et $j^{ième}$ colonne est noté : A[i-1][j-1]
- A[0][0] : 0, A[1][0] : 10, A[1][2] :12, A[2][9] :0.

Parcours d'un tableaux à deux dimensions

```c
#include <stdio.h>
#include <stdlib.h>
void afficherTableau(int tableau[2][2]);
int main(void)
{
    int tableau[2][2] = {{10, 20} , {15, 35}};
    afficherTableau(tableau);
        return 0;
}
void afficherTableau(int tableau[2][2])
{
  int i = 0;
  int j = 0;
  for (i = 0; i < 2; i++) {
    for(j = 0; j < 2; j++) {
      printf("Tableau[%d][%d] = %d\n", i, j, tableau[i][j]);
    }
  }
}
```