TP2 - LES TABLEAUX

Année 2021/2022

1. Définition

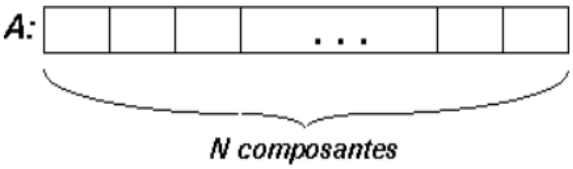
Les tableaux sont certainement les variables **structurées** les plus populaires. Ils sont disponibles dans tous les langages de programmation et servent à résoudre une multitude de problèmes. Dans une première approche, le traitement des tableaux en C ne diffère pas de celui des autres langages de programmation. Le langage C permet un accès encore plus direct et plus rapide aux données d'un tableau.

Les chaînes de caractères sont déclarées en C comme tableaux de caractères et permettent l'utilisation d'un certain nombre de notations et de fonctions spéciales.

2. Les tableaux à une dimension

2.1 Définitions

Un tableau (uni-dimensionnel) "A" est une variable structurée formée d'un nombre entier **N** de variables simples du **même type**, qui sont appelées les composantes du tableau. Le nombre de composantes N est alors la **dimension du tableau**. En faisant le rapprochement avec les mathématiques, on dit encore que "A est un vecteur de dimension N".



Exemple : `int JOURS[12]={31,28,31,30,31,30,31,31,30,31,30,31};`

La déclaration définit un tableau du type **int** de dimension **12**. Les 12 composantes sont initialisées par les valeurs respectives 31, 28, 31, ... , 31. On peut accéder à la première composante du tableau par **JOURS[0],** à la deuxième composante par **JOURS[1],** . . . , à la dernière composante par **JOURS[11].**

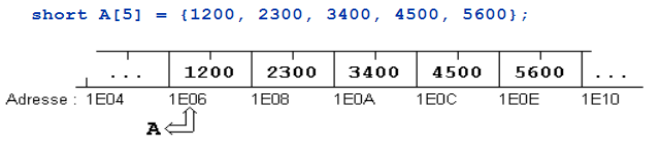
2.1.1 Déclaration et mémorisation

Déclaration : **<TypeSimple> <NomTableau>[<Dimension>];**

Exemples :

```
int A[25];        ou bien  long A[25];        ou bien  ...
float B[100];     ou bien  double B[100];     ou bien  ...
char D[30];
```

Mémorisation : En C, le nom d'un tableau est le représentant de l'adresse du premier élément du tableau. Les adresses des autres composantes sont calculées (automatiquement) relativement à cette adresse. ***Exemple*** :



Si un tableau est formé de N composantes et si une composante a besoin de M octets en mémoire, alors le tableau occupera de N*M octets.

2.1.2 Initialisation et réservation automatique

Initialisation : Lors de la déclaration d'un tableau, on peut initialiser les composantes du tableau, en indiquant la liste des valeurs respectives entre accolades.

Exemples :
```
int A[5]   = {10, 20, 30, 40, 50};
float B[4] = {-1.05, 3.33, 87e-5, -12.3E4};
```
Il faut évidemment veiller à ce que le nombre de valeurs dans la liste corresponde à la dimension du tableau. Si la liste ne contient pas assez de valeurs pour toutes les composantes, **les composantes restantes sont initialisées par zéro.**

Réservation automatique : Si la dimension n'est pas indiquée explicitement lors de l'initialisation, alors l'ordinateur réserve automatiquement le nombre d'octets nécessaires.

Exemples : `int A[] = {10, 20, 30, 40, 50};` réservation de 5*sizeof(int) octets (dans notre cas: 10 octets)

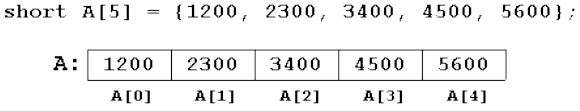
2.1.3 Accès aux composantes

En déclarant un tableau par : `int A[5];` Nous avons défini un tableau A avec cinq composantes, auxquelles on peut accéder par : `A[0], A[1], ... , A[4]`

Exemple : - l'accès au premier élément du tableau se fait par `A[0]`
- l'accès au dernier élément du tableau se fait par `A[N-1]`

2.1.4 Affichage et affectation

La structure **for** se prête particulièrement bien au travail avec les tableaux. La plupart des applications se laissent implémenter par simple modification des exemples-types de l'affichage et de l'affectation.



Affichage du contenu d'un tableau :Traduisons le programme **AFFICHER** du langage algorithmique en C:

```
main()
{
 int A[5];
 int i;  /* Compteur */
   for (i=0; i<5; i++)
     { printf("%d ", A[i]);}
 return 0;
 printf("\n");
}
```

Affectation avec des valeurs provenant de l'extérieur : Traduisons le programme **REMPLIR** du langage algorithmique en C :

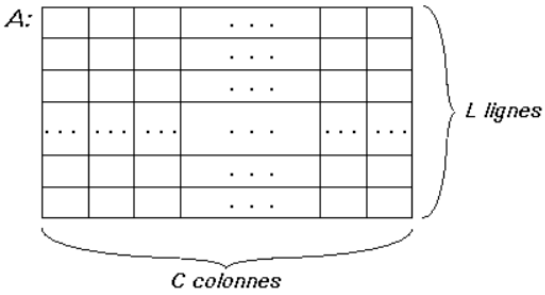
```
main()
{
 int A[5];
 int i;  /* Compteur */
  for (i=0; i<5; i++)
        {scanf("%d", &A[i]);}
 return 0;
}
```

Remarques :

- Comme **scanf** a besoin des adresses des différentes composantes du tableau, il faut faire précéder le terme A[i] par l'opérateur adresse '**&**'.
- La commande de lecture **scanf** doit être informée du type exact des données à lire. (Ici: %d ou %i pour lire des valeurs du type **int**)

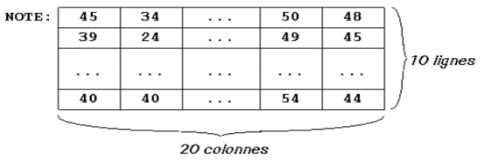
3. Les tableaux à deux dimensions

Définitions : En C, un tableau à deux dimensions A est à interpréter comme un tableau (uni-dimensionnel) de dimension L dont chaque composante est un tableau (uni-dimensionnel) de dimension C. On appelle **L** le **nombre de lignes** du tableau et **C** 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**.



Exemple :

```
int NOTE[10][20] = {{45, 34, ... , 50, 48},
                    {39, 24, ... , 49, 45},
                    ...   ...     ...
                    {40, 40, ... , 54, 44}};
```



3.1 Déclaration de tableaux à deux dimensions en C : `<TypeSimple> <NomTabl>[<DimLigne>][<DimCol>];`

Exemples

```
int A[10][10];        ou bien long A[10][10];   ou bien ...
float B[2][20];       ou bien double B[2][20];  ou bien ...
char D[15][40];
```

Initialisation : Lors de la déclaration d'un tableau, on peut initialiser les composantes du tableau, en indiquant la liste des valeurs respectives entre accolades. A l'intérieur de la liste, les composantes 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.

Exemples **:**
```
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}};
```

Accès à un tableau à deux dimensions en C : **`<NomTableau>[<Ligne>][<Colonne>]`**

En C :

- Les indices du tableau varient de 0 à L-1, respectivement de 0 à C-1.
- La composante de la Nième ligne et Mième colonne est notée **`: A[N-1][M-1]`**

Affichage et affectation : Lors du travail avec les tableaux à deux dimensions, nous utiliserons deux indices (p.ex: i et j), et la structure for, souvent imbriquée, pour parcourir les lignes et les colonnes des tableaux.

```
main()
{
 int A[5][10];
 int i,j;
 /* Pour chaque ligne ... */
 for (i=0; i<5; i++)
   {
    /* ... considérer chaque composante */
    for (j=0; j<10; j++)
          printf("%7d", A[i][j]);
       printf("\n");  //Retour à la ligne
   }
 return 0;
}
```

Remarques : Pour obtenir des colonnes bien alignées lors de l'affichage, il est pratique d'indiquer la largeur minimale de l'affichage dans la chaîne de format. Pour afficher des matrices du type **int** (valeur la plus 'longue': -32768), nous pouvons utiliser la chaîne de format **`"%7d"`** : **Exemple** : **`printf("%7d", A[I][J]);`**

Affectation avec des valeurs provenant de l'extérieur :

```
main()
{
 int A[5][10];
 int i,j;
 /* Pour chaque ligne ... */
 for (i=0; i<5; i++)
   /* ... considérer chaque composante */
   for (j=0; j<10; j++)
         scanf("%d", &A[i][j]);
 return 0;
}
```

Exercices :

Exercice 1 : Ecrire un programme qui lit la dimension **N** d'un tableau T du type **int** (dimension maximale : 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau. Calculer et afficher ensuite la somme des éléments du tableau.

Exercice 2 : Ecrire un programme qui lit la dimension **N** d'un tableau T du type **int** (dimension maximale : 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau. Effacer ensuite toutes les occurrences de la valeur 0 dans le tableau T et tasser les éléments restants. Afficher le tableau résultant.

Exercice 3 : Ecrire un programme qui lit la dimension N d'un tableau T du type int (dimension maximale : 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau. Copiez ensuite toutes les composantes strictement positives dans un deuxième tableau TPOS et toutes les valeurs strictement négatives dans un troisième tableau TNEG. Afficher les tableaux TPOS et TNEG.

Exercice 4 : Ecrire un programme qui lit la dimension **N** d'un tableau T du type **int** (dimension maximale : 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau. Ranger ensuite les éléments du tableau T dans l'ordre inverse sans utiliser de tableau d'aide. Afficher le tableau résultant.

***Idée** : Echanger les éléments du tableau à l'aide de deux indices qui parcourent le tableau en commençant respectivement au début et à la fin du tableau et qui se rencontrent en son milieu.*

Exercice 5 : 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.

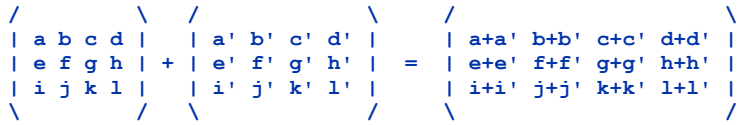
Exercice 6 : Ecrire un programme qui lit les dimensions L et C d'un tableau T à deux dimensions du type int (dimensions maximales : 50 lignes et 50 colonnes). Remplir le tableau par des valeurs entrées au clavier et afficher le tableau ainsi que la somme de chaque ligne et de chaque colonne en n'utilisant qu'une variable d'aide pour la somme.

Exercice 7 : Ecrire un programme qui transfère un tableau M à deux dimensions L et C (dimensions maximales : 10 lignes et 10 colonnes) dans un tableau V à une dimension L*C.

Exemple:

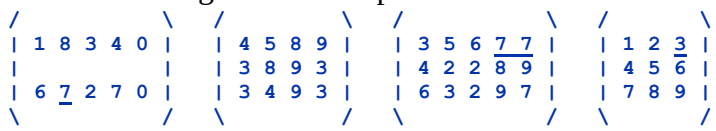
Exercice 8 : Addition de deux matrices : Ecrire un programme qui réalise l'addition de deux matrices A et B de mêmes dimensions N et M. Le résultat de l'addition sera mémorisé dans une troisième matrice C qui sera ensuite affichée.

Rappel:



Exercice 9 : Recherche de 'points-cols' : Rechercher dans une matrice donnée A les éléments qui sont à la fois un maximum sur leur ligne et un minimum sur leur colonne. Ces éléments sont appelés des points-cols. Afficher les positions et les valeurs de tous les points-cols trouvés.

Exemples: Les éléments soulignés sont des points-cols:



Méthode: Etablir deux matrices d'aide MAX et MIN de même dimensions que A, telles que:

MAX[i,j] = / 1 si A[i,j] est un maximum sur la ligne i
\ 0 sinon

MIN[i,j] = / 1 si A[i,j] est un minimum sur la colonne j
\ 0 sinon