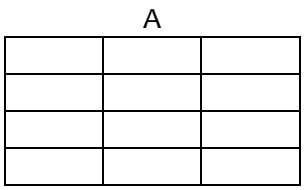
Rappel sur les tableaux à deux dimensions

Un tableau à deux dimensions est un tableau organisé en lignes et en colonnes. Voici un exemple de déclaration d'un tableau de deux dimensions de 4 lignes et de 3 colonnes de type float.

float A[4][3] ;

Ce tableau A peut être vue comme la figure suivante :



Au niveau de la RAM, les éléments du tableau A sont contigus, c'est-à-dire que les éléments du tableau A se suivent les uns après les autres. Voici la structure du tableau A dans la RAM :

Les trois premières cases représentent la première ligne, les 4, 5 et 6 représentent la deuxième ligne etc.

Pour parcourir un tableau de deux dimensions, nous utilisons deux indices. Un indice pour parcourir les lignes et un autre indice pour parcourir les colonnes.

Soit l'exemple suivant :

```
int M[4][10] = {{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9},
                {10,11,12,13,14,15,16,17,18,19},
                {20,21,22,23,24,25,26,27,28,29},
                {30,31,32,33,34,35,36,37,38,39}};
```

Il s'agit d'un tableau de deux dimensions de 4 lignes et de 10 colonnes initialisés par les valeurs ci-dessus. Voici les instructions permettant d'afficher le tableau M :

```
for(i=0 ; i<4 ;i++)
        for(j=0 ;j<10 ; j++){
                if (i%10==0) //condition pour revenir à la ligne lorsqu'une ligne est terminée
                        printf("\n")

                printf("%d\t", M[i][j]);
        }
```

Pointeurs et tableaux à deux dimensions

Soit l'exemple suivant pour mieux comprendre les pointeurs appliqués sur des tableaux à deux dimensions :

```
1.  int M[4][10] = {{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9},
                    {10,11,12,13,14,15,16,17,18,19},
                    {20,21,22,23,24,25,26,27,28,29},
                    {30,31,32,33,34,35,36,37,38,39}};
2.  int i;
3.  int *PM;
4.  PM = M;
5.  for(i=0; i<40;i++) {
            if (i%10==0)
                    printf("\n");
            printf("%d\t", *(PM+i));
    }
```

Dans l'instruction 1, nous déclarons un tableau d'entiers de deux dimensions appelé M avec 4 lignes et 10 colonnes. Ce tableau M est initialisé par des valeurs au moment de la déclaration.

Dans l'instruction 2, nous déclarons un indice nommé i pour parcourir le tableau M. Vous allez poser la question pourquoi un seul indice et pas deux indices. Effectivement, nous allons utiliser qu'un seul indice. Pourquoi ? Vous allez comprendre.

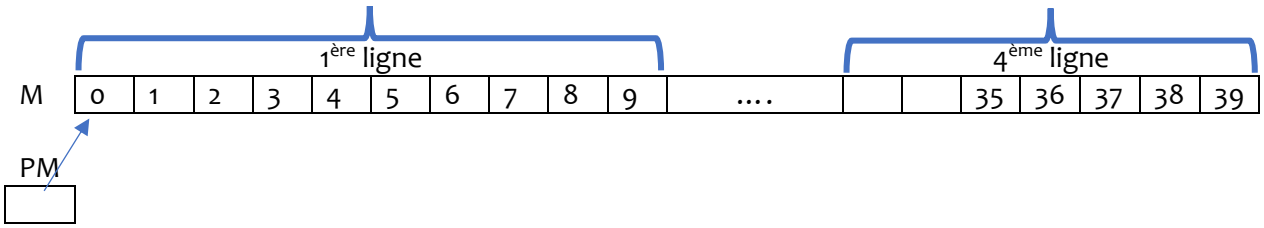
Dans l'instruction 3, nous déclarons un pointeur PM.

Dans l'instruction 4, nous affectons le tableau M au pointeur PM. M correspond à l'adresse du début du tableau M, et le début du tableau M est M[0][0] c'est-à-dire que nous affectons &M[0][0] au pointeur PM. Nous pouvons écrire maintenant :

PM = &M[0][0] ;

C'est le même principe dans un tableau à une seule dimension. Lorsque nous affectons un tableau T à un pointeur PT : PT = T, c'est-à-dire nous affectons au pointeur l'adresse du premier élément du tableau : PT= &T[0]

Ainsi, nous avons au niveau de la RAM les variables suivantes :

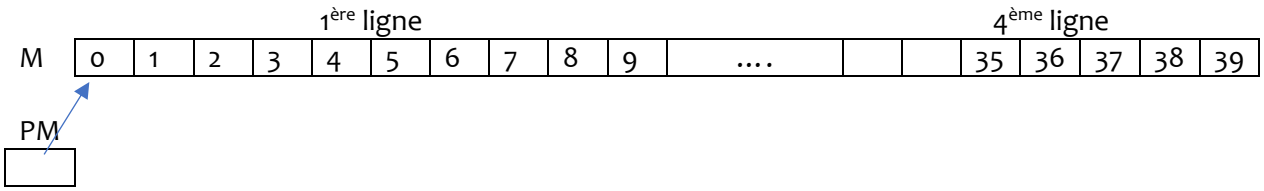


Ne pas oublier que le pointeur PM pointe sur le premier élément du tableau M et les éléments du tableau M sont contigus. Ainsi, nous pouvons considérer le tableau M de deux dimensions comme un

tableau d'une seule dimension ayant 40 éléments. C'est-à-dire le nombre de lignes que multiplie par le nombre de colonnes (4 * 10).

Et comme vous le savez pour parcourir un tableau d'une seule dimension, nous aurons besoin d'un seul indice. Voilà je viens de répondre à la question, pourquoi un seul indice.

Dans l'instruction 5, nous affichons le tableau M en utilisant le pointeur PM. L'instruction 5 comporte une seule boucle for avec un seul indice i. Aussi, vous constatez que dans l'instruction for et dans la condition i<40 nous avons mis la valeur 40, pourquoi ? Nous avons dit qu'un tableau de deux dimensions, ses éléments sont contigus et pour le parcourir avec un pointeur, il suffit de le considérer comme un tableau d'une seule dimension avec le nombre d'éléments qui est égale au nombre de ligne du tableau à deux dimensions multiplié par le nombre de colonne du même tableau à deux dimensions. Pour notre cas, nous avons 4 lignes et 10 colonnes, c'est-à-dire nous avons 40 éléments. C'est pour cela, que nous avons mis dans la condition i<40 afin que la boucle for parcourt 40 éléments.



Soit le tableau suivant récapitulant l'adressage et le contenu du tableau M avec la notation pointeur PM

&M[0][0] ⇔ PM + 0		M[0][0] ⇔ *(PM + 0)
&M[0][1] ⇔ PM + 1		M[0][1] ⇔ *(PM + 1)
&M[0][2] ⇔ PM + 2		M[0][2] ⇔ *(PM + 2)
…		…
&M[1][0] ⇔ PM + 10		M[1][0] ⇔ *(PM + 10)
&M[1][1] ⇔ PM + 11		M[1][1] ⇔ *(PM + 11)
…		
&M[3][0] ⇔ PM + 30		M[3][0] ⇔ *(PM + 30)
&M[3][1] ⇔ PM + 31		M[3][1] ⇔ *(PM + 31)

Ainsi de façon générale, pour accéder à un élément du tableau M en utilisant le pointeur PM, il suffit d'utiliser la formule suivante :

M[i][j] ⇔ PM + i * 10 + j (10 est le nombre de colonne)

Pour M[3][2] ⇔ PM + 3 * 10 + 2 = PM + 32

Donc, pour afficher les éléments du tableau M en utilisant le pointeur PM, nous avons cette instruction :

```
for(i=0; i<40;i++)
    printf("%d", *(PM+i));
```

Lors de l'affichage, nous pouvons séparer les éléments par des espaces en utilisant \t :

```
for(i=0; i<40;i++)
    printf("%d\", *(PM+i));
```

Pour afficher chaque 10 éléments dans une ligne, nous allons ajouter cette condition :

```
if (i%10==0)  printf("\n");
```

Voilà notre instruction for permettant d'afficher les éléments du tableau M en utilisant le pointeur PM. Après chaque affichage d'une ligne du tableau M, nous revenons à la ligne suivante.

```
for(i=0; i<40;i++)  {
            if (i%10==0)
                    printf("\n");
            printf("%d\t", *(PM+i));
    }
```

Voici le programme complet :

```
#include <stdio.h>
int main()
{
    int M[4][10] = {{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9},
                {10,11,12,13,14,15,16,17,18,19},
                {20,21,22,23,24,25,26,27,28,29},
                {30,31,32,33,34,35,36,37,38,39}};
    int i;
    int *PM;
    PM = M;
    for(i=0; i<40;i++)  {
       if (i%10==0)
          printf("\n");
       printf("%d\t", *(PM+i));
    }
}
```