TP 1 : les tableaux statiques à double dimension, les tableaux dynamiques à double dimension (tableaux de pointeurs)

1 Tableaux statiques à double dimension

Dans cet exercice, nous allons envisager le problème des tableaux statiques à deux dimensions et essayer de comprendre comment les manipuler. La déclaration d'un tableau à deux dimensions peut se faire de la façon suivante :

```
int tableau[DIMENSION1][DIMENSION2];
```

Un tableau à deux dimensions est ainsi stocké en mémoire :

```
tableau[0][0], tableau[0][1], ..., tableau[0][DIMENSION2-1],
tableau[1][0], tableau[1][1], ..., tableau[1][DIMENSION2-1],
...
tableau[DIMENSION1-1][0], tableau[DIMENSION1-1][1], ...,
tableau[DIMENSION1-1][DIMENSION2-1]
```

1. Connaissant la méthode de stockage en mémoire d'un tableau à deux dimensions, décrire quels sont les éléments indispensables que le compilateur doit connaître pour calculer l'adresse d'un élément `(i,j)`.
2. Ecrire une fonction `Saisie` qui permet de saisir un tableau à deux dimensions. Cette fonction prend en paramètre le tableau lui-même ainsi que le nombre de lignes et de colonnes. Il ne s'agit ici que de saisie, l'allocation mémoire est faite dans le programme principal.
3. Ecrire la fonction d'affichage correspondante `Affiche` et un petit programme principal correspondant.
4. Compte-tenu de la représentation mémoire d'un tel tableau, on se rend compte que la variable tableau n'est au fond que l'adresse en mémoire du premier entier de la première ligne et qu'on doit pouvoir aisément copier cette adresse dans un pointeur de type `int *`

. Ecrivez donc la fonction (appelée `Saisiep`) qui réalise à nouveau la saisie du tableau mais en passant le tableau en paramètre sous la forme d'un pointeur d'entier. Comment accéder à l'élément (i,j) dans ce cas à l'intérieur de la fonction ?

5. Vérifiez que les fonctions `Saisiep` et `Affiche` sont bien compatibles, c'est-à-dire que vous pouvez bien afficher avec la fonction `Affiche` le tableau qui a été saisie avec la fonction `Saisiep`.

2 Tableaux dynamiques à double dimension (tableaux de pointeurs)

Il y a parfois confusion en langage C entre tableaux de pointeurs et tableaux muti-dimensionnels statiques car on peut réaliser les mêmes choses en utilisant ces deux notions (et notamment utiliser dans les deux cas la notation `tab[i][j]`). Nous allons donc à présent utiliser des tableaux de pointeurs pour implémenter une matrice d'entiers.

La représentation est composée de plusieurs tableaux : 1) chaque ligne de la matrice est représentée par un tableau d'entiers et 2) un tableau de pointeurs quant à lui, contient les adresses de tous ces tableaux. La figure 1 vous représente cette structure.

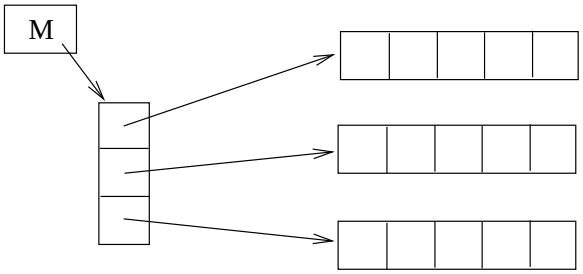

FIGURE 1 – Tableaux de pointeurs

1. Quelle est la déclaration que vous allez faire pour une telle matrice (quel est le type de M en somme) ?
2. Définir le code permettant d'allouer de la mémoire pour une matrice de 2 lignes et 3 colonnes.
3. Ecrivez la fonction `Saisie2` qui permet de saisir une telle matrice. Quelle différence par rapport à la fonction `Saisie` de la section 1 ?

4. Ecrivez la fonction `Saisie2p` dont les paramètres seront exactement les mêmes que ceux de `Saisie2` mais dans laquelle nous utiliserez uniquement des déplacements de pointeurs pour accéder à l'élément (i,j) (autrement dit, toute utilisation de notation type [i][j] est interdite). Comparez avec la fonction `Saisiep` et retrouvez bien la différence d'implémentation mémoire entre `int tab[][]` et `int ** tab`.
5. Ecrire très simplement la fonction d'affichage `Affiche2` pour une telle matrice.
6. Vérifier qu'une matrice saisie par la fonction `Saisie2p` est bien affichable par la fonction `Affiche2`.
7. Pour terminer cette partie, créez une fonction `Init` qui retourne `void` et qui permet d'initialiser une matrice telle que M. En particulier, le nombre de lignes et de colonnes seront lues dans la fonction et l'allocation mémoire sera réalisée dans la fonction.