Cours 8: tableau de tableaux

Guillaume Aubian

Ce document a été grandement inspiré par – et largement copié sur – le document correspondant du responsable précédent de ce cours, Juliusz Chroboczek, avec son accord.

1 Les tableaux de tableaux

Les tableaux que nous avons vus précédemment sont à une dimension : ils sont indexés par des entiers, et sont isomorphes aux suites finies ou aux vecteurs en mathématiques. Les tableaux à deux dimensions sont indexés par deux entiers, et correspondent donc aux matrices en mathématiques.

On peut représenter un tableau à deux dimensions par un tableau à une dimension (figure 3). Un tableau A de m lignes et n colonnes est représenté par un tableau a de $m \times n$ cases, et l'élément $A_{i,j}$ est stocké dans a_{ni+j}. Cette approche est facile à implémenter en C ; par exemple, la fonction suivante crée une table de multiplication :

```
int *table() {
    int *a = malloc(10 * 10 * sizeof(int));
    if(a == NULL) return NULL;
    for(int i = 0; i < 10; i++) {
    for(int j = 0; j < 10; j++) {
        a[10 * i + j] = (i + 1) * (j + 1);
    }
    }
    return a;
}
```

Cette approche est très efficace du moment que l'on prend soin de parcourir les données dans l'ordre dans lequel elles stockées en mémoire. Dans le fragment de code ci-dessous, j'ai parcouru

Figure 1: Les tableaux à l'ancienne.

a_0	a_1	a_2	a_3

Figure 2: Un tableau de tableaux.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

Figure 3: Un tableau de tableaux.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

la table de multiplication ligne par ligne justement dans le but de causer des accès séquentiels à la mémoire.

Le C sait gérer nativement les tableaux à deux dimensions. Un tableau à deux dimensions natif est déclaré avec la syntaxe `int a[lignes][colonnes];` et on accède à ses éléments à l'aide de `a[i][j]`.