4.5.5 Les tableaux en Caml

En Caml, un tableau dont les éléments sont de type `T` *est de type* `T array`. *On alloue un tableau avec l'expression* `Array.make u v` *où* `u` *est une expression dont la valeur est un entier* `n` *et* `v` *une expression de type* `T`. *Le tableau créé a la taille* `n` *et chaque champ reçoit comme valeur initiale la valeur de l'expression* `v`.

Si la valeur de l'expression `t` *est une référence associée dans la mémoire à un tableau et la valeur de l'expression* `u` *est un entier* `k`, *la valeur de l'expression* `Array.get t u` *est la valeur contenue dans le* $\texttt{k}^{\text{ème}}$ *champ de ce tableau.*

Pour affecter le $\texttt{k}^{\text{ème}}$ *champ d'un tableau, on utilise une nouvelle instruction* `Array.set t u v` *où* `t` *est une expression dont la valeur est une référence associée dans la mémoire à un tableau,* `u` *est une expression dont la valeur est un entier* `k` *et* `v` *une expression du même type que les éléments du tableau. Quand on exécute cette instruction, le* $\texttt{k}^{\text{ème}}$ *champ du tableau reçoit la valeur de l'expression* `v`.

Ainsi, le programme

```
let t = Array.make 10 0
in let k = 5
in Array.set t k 4; print_int (Array.get t k)
```

affiche 4.

4.5.6 Les tableaux en C

En C, les tableaux, comme les enregistrements, ne sont pas alloués. De ce fait, la taille d'un tableau ne peut pas être déterminée au moment de son allocation et elle fait partie de son type.

Un tableau de taille `n` *dont les éléments sont de type* `T` *est de type* `T[n]`. *On déclare une variable* `t` *de type* `T[n]` *ainsi*

```
T t [n];
```

Par exemple

```
int t [10];
```

La taille du tableau n'est pas nécessairement une constante, mais peut être n'importe quelle expression entière. Le tableau créé a la taille `n` *et chaque champ reçoit une valeur initiale quelconque.*

En C, comme en Java et en Caml, la valeur de la variable `t` *est une référence associée dans la mémoire à un tableau. Les tableaux sont donc assez différents*