Taille d'un tableau en C

Pour déterminer la dimension d'un tableau `t` en C, le cours conseille d'utiliser la fonction `sizeof` et un quotient :

```
sizeof t / sizeof *t;
```

Le rapport du total de bytes occupés par le tableau sur la taille de son premier élément donne en effet le nombre d'éléments.

Toutefois, cette méthode marche bien dans le bloc d'instructions où est défini le tableau mais n'est pas robuste vis à vis du passage d'un tableau en paramètres.

Considérons le code :

```
# include <stdio.h>

int tailletab (int t[]){// taille d'un tableau, vraiment ?
  return sizeof t / sizeof *t;
}

int main(){
  int t[5] = {1,2,3,4,5};
  printf("taille t : %ld bytes\n", (sizeof t / sizeof *t));
  printf("tailletab(t) = %d bytes\n", tailletab(t));
  printf("sizeof(int)=%ld bytes\n", sizeof(int));
  printf("sizeof(int *)=%ld bytes\n", sizeof(int *));
  return 0;
}
```

Après compilation et exécution, on obtient :

```
$ ./a.out
taille t : 5 bytes
tailletab(t) = 2 bytes
sizeof(int)=4 bytes
sizeof(int *)=8 bytes
```

Notre méthode par quotient indique bien le nombre d'éléments de `t` (5 bytes) dans le bloc d'instructions où il est déclaré. Le quotient calcule le nombre de bytes occupés par `t` (5×4 bytes) divisée par la taille de `*t`, c'est à dire la taille du premier élément de `t` [1]. Or, un entier occupe 4 bytes. Ainsi 20/4 donne 5, ce qu'on veut.

Mais lorsqu'on passe `t` en paramètre de la fonction `tailletab`, on trouve que la taille de `t` est 2 bytes.

Que s'est-il passé ?

1. On peut préférer écrire `sizeof t[0]` plutôt que `sizeof *t` : ça revient au même

Les arguments sont transmis par valeurs en C. Ce qui fait qu'une fonction travaille en fait avec des copies des arguments.

Dans le cas particulier d'un tableau passé en argument d'une fonction, celle-ci travaille en fait avec un pointeur sur le premier élément [2].

Dans l'appel `tailletab(t)`, `t` est donc un objet de type `int *`, un pointeur sur `t[0]`. La taille d'un pointeur (donc d'une adresse) est 8 bytes. Ainsi la taille calculée est :

```
taille de int * / taille de t[0]
```

soit 8/4 donc 2 bytes.

Il n'est donc JAMAIS possible de transmettre à une fonction tous les éléments d'un tableau et donc d'en connaître la taille.

On retient ♡ On peut obtenir la dimension d'un tableau `t` dans le bloc où il a été déclaré en calculant le quotient :

```
1 sizeof t / sizeof t[0]
```

Dans le bloc de déclaration, `sizeof t` désigne bien le nombre de bytes occupés par le tableau (il est connu du compilateur).

Mais cette méthode ne fonctionne plus lorsque le tableau est passé en argument d'une fonction. Cette dernière travaille en fait non avec `t`, mais avec un pointeur sur son premier élément. Ainsi `sizeof t` désigne la taille d'une adresse, laquelle est une quantité fixe.

2. Les autres éléments du tableau (mais pas leur nombre) sont accessibles dans la fonction car ils occupent des positions contiguës en mémoire