Tableaux et fonctions

Programmation séquentielle en C, 2024-2025

Orestis Malaspinas (A401)

2024-10-08

Informatique et Systèmes de Communication, HEPIA

Rappel (1/3)

- Comment sont passés les arguments d'une fonction en `c`?

Rappel (1/3)

- Comment sont passés les arguments d'une fonction en `c`?
- **Toujours** par **copie**.

```c
void foo(int a) {
    a = 2;
}
void main() {
    int x = 1;
    foo(x);
    // Que vaut x ici?
}
```

Rappel (2/3)

- Les argument sont toujours **passés par copie**.

```
void foo(int a) {
    a = 2;
}
void main() {
    int x = 1;
    foo(x);
    // x vaut toujours 1
}
```

- Une nouvelle variable `int a` est créée lors de l'appel à `foo(a)`, et on lui assigne la valeur de `x`.
- `x` n'est donc jamais modifiée.

Rappel (3/3)

- Comment modifier un argument d'une fonction en `c`?

Rappel (3/3)

- Comment modifier un argument d'une fonction en `C`?
- L'argument doit être la **référence** vers la variable.

```c
void foo(int *a) { // le pointeur a
    *a = 2;        // déréférencement du pointeur a
}
void main() {
    int x = 1;
    foo(&x); // référence vers x
    // Ici x faut 2
}
```

- Une nouvelle variable `int *a` est créée lors de l'appel à `foo`, et on lui assigne la valeur de `&x`.
- `&x` n'est jamais modifiée **mais** `x` l'est.

Remarques

- On peut *retourner* une valeur depuis une fonction.

```
int foo(int a) {
    return a + 2;
}
void main() {
    int x = 1;
    int y = foo(x); // x n'est pas modifiée
                    // et y vaut x + 2
}
```

Les tableaux statiques

Qu'est-ce qu'un tableau statique?

Les tableaux statiques

Qu'est-ce qu'un tableau statique?

- Une **liste** ou un **ensemble**

Les tableaux statiques

Qu'est-ce qu'un tableau statique?

- Une **liste** ou un **ensemble**
- d'éléments du **même type**

Les tableaux statiques

Qu'est-ce qu'un tableau statique?

- Une **liste** ou un **ensemble**
- d'éléments du **même type**
- alloués de façon **contigüe** (en bloc) en mémoire

Les tableaux statiques

Qu'est-ce qu'un tableau statique?

- Une **liste** ou un **ensemble**
- d'éléments du **même type**
- alloués de façon **contigüe** (en bloc) en mémoire
- sur la **pile**

Les tableaux statiques

Qu'est-ce qu'un tableau statique?

- Une **liste** ou un **ensemble**
- d'éléments du **même type**
- alloués de façon **contigüe** (en bloc) en mémoire
- sur la **pile**
- dont la taille **ne peut pas** être changée.

Les tableaux statiques

Qu'est-ce qu'un tableau statique?

- Une **liste** ou un **ensemble**
- d'éléments du **même type**
- alloués de façon **contigüe** (en bloc) en mémoire
- sur la **pile**
- dont la taille **ne peut pas** être changée.

Remarques

- Les éléments d'un tableau sont accédés avec `[i]` où `i` est l'index de l'élément.
- Le premier élément du tableau à l'index `0`!
- Lorsqu'un tableau est déclaré, la taille de celui-ci doit toujours être spécifiée, sauf s'il est initialisé lors de sa déclaration.
- Un tableau local à une fonction ne doit **jamais être retourné**!

Syntaxe des tableaux

```
float tab1[5]; // tableau de floats à 5 éléments
               // ses valeurs sont indéfinies

int tab2[] = {1, 2, 3}; // tableau de 3 entiers,
                        // taille inférée

int val = tab2[1]; // val vaut 2 à présent

int w = tab1[5]; // index hors des limites du tableau
                 // comportement indéfini!
                 // pas d'erreur du compilateur
```

Itérer sur les éléments d'un tableau

```
int x[10];
for (int i = 0; i < 10; ++i) {
    x[i] = 0;
}
int j = 0;
while (j < 10) {
    x[j] = 1;
    j += 1;
}
int k = 0;
do {
    x[k] = -1;
    k += 1;
} while (k < 9);
```

Représentation des tableaux en mémoire

La mémoire est :

- ... contigüe,
- ... accessible très rapidement

Exemple:

```
char tab[4] = {79, 91, 100, 88};
```

char	79	91	100	88	….	….
addr	2000	2001	2002	2003	….	….

Qu'est-ce que `tab`?

```
tab;      // 2000, l'adress du 1er élément
&tab[0]; // 2000 == tab
tab[0];  // 79
sizeof(tab); // 4
```

Les tableaux comme arguments de fonctions

- Un tableau en argument est le pointeur vers sa première case.
- Pas moyen de connaître sa taille: `sizeof()` inutile.
- Toujours spécifier la taille d'un tableau passé en argument.

```c
void foo(int tab[]) { // équivalent à int *tab
    // que vaut sizeof(tab)?
    for (int i = 0; i < ?; ++i) { // taille?
        printf("tab[%d] = %d\n", i, tab[i]);
    }
}
void bar(int n, int tab[n]) { // [n] optionnel
    for (int i = 0; i < n; ++i) {
        printf("tab[%d] = %d\n", i, tab[i]);
    }
}
```

Quels sont les bugs dans ce code?

```c
#include <stdio.h>

int main(void) {
    char i;
    char a1[] = { 100, 200, 300, 400, 500 };
    char a2[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    a2[10] = 42;

    for (i = 0; i < 5; i++) {
        printf("a1[%d] = %d\n", i, a1[i]);
    }

    return 0;
}
```

Quels sont les bugs dans ce code?

```c
#include <stdio.h>

int main(void) {
    char i;
    // 200, .., 500 char overflow
    char a1[] = { 100, 200, 300, 400, 500 };
    char a2[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    a2[10] = 42; // [10] out of bounds

    for (i = 0; i < 5; i++) {
        printf("a1[%d] = %d\n", i, a1[i]);
    }

    return 0;
}
```