

Programmation séquentielle

Série 8 - Pointeurs et mémoire dynamique

22.11.2022

Buts

- Utilisation de pointeurs.
- Utilisation de la mémoire dynamique.

Énoncé

Dans ce travail pratique, vous allouerez **tous** les tableaux dynamiquement, à l'aide de la fonction `malloc`. Ainsi un tableau d'entiers de taille 10, nommé `tab`, sera déclaré et alloué de la façon suivante:

```
int *tab = malloc(10 * sizeof(int));  
int *tab = malloc(10 * sizeof(*tab)); // alternative
```

La mémoire ainsi allouée doit toujours être **explicitement libérée** à l'aide de la fonction `free`.

```
free(tab);
```

Il est obligatoire d'utiliser un `memory sanitizer` lors de votre compilation. Cela s'effectue en ajoutant les options `-g -fsanitize=address -fsanitize=leak` lors de la compilation.

Exercice 1 - Les pointeurs

Il se peut qu'une partie de ces programmes produise des erreurs. Identifiez les et quand cela s'y prête, tentez de corriger le code.

Une fois que vous pensez que votre code est correct, essayez de prédire ce qu'il va afficher, puis compilez le avec

```
$ gcc programme.c -o programme -g -fsanitize=address -fsanitize=leak
```

et exécutez le. Est-ce que cela correspond à votre prédiction ?

Programme 1

```
#include <stdio.h>

void foo(int a) {
    a += 3;
}

void bar(int *a) {
    *a += 3;
}

void baz(int *a) {
    a += 3;
}

int main() {
    int a = 5;
    printf("%i\n", a);

    foo(a);
    printf("%i\n", a);

    bar(&a);
    printf("%i\n", a);

    baz(&a);
    printf("%i\n", a);
}
```

Programme 2

```
#include <stdio.h>
#include <stdlib.h>

int *foo(int a) {
    int tab[a];
    for (int i = 0; i < a; ++i) {
        tab[i] = 2;
    }
    return tab;
}

int *bar(int a) {
    int *tab = malloc(a);
    for (int i = 0; i < a; ++i) {
        tab[i] = 2;
    }
    return tab;
}

void baz(int *tab, int b) {
    tab = malloc(b*sizeof(int));
    for (int i = 0; i < b; ++i) {
        tab[i] = 2;
    }
}

int main() {
    int *a = foo(4);
    printf("%d\n", a[3]);

    int *b = bar(4);
    printf("%d\n", b[3]);

    int *c = malloc(4*sizeof(int));
    baz(c, 4);
    printf("%d\n", c[3]);

    free(a);
    free(b);
    free(c);
}
```

Exercice 2 - Mémoire dynamique

Écrire un programme *modulaire* (pensez à utiliser des fonctions) qui réalise les tâches suivantes.

- Demander à l'utilisateur d'entrer la valeur de `size` pour la création d'un tableau de `int`.
- Remplir le tableau `tab` de toutes les valeurs entre 0 et `size-1` dans l'ordre.
- Afficher le tableau.
- Effectuer une permutation aléatoire de tous les éléments de `tab`.
- Effectuer une permutation cyclique de tous les éléments de `tab` d'un nombre `n` d'éléments vers la droite.
- Placer le plus petit élément de `tab` en fin de tableau en procédant à un échange de place.
- Allouer un second tableau, `tab_two` de longueur `size` et le remplir de façon similaire à `tab`, et effectuer la somme de `tab` avec `tab_two` élément par élément. Stocker le résultat dans un troisième tableau préalablement alloué `tab_three`.