Algorithmique et structures de données

Loïc Demange
loic.demange@etud.univ-paris8.fr

29 octobre 2021

TP précédent

Objectifs du TP :

- Créer une fonction factorielle qui renvoie la factorielle d'un entier sans boucle.
- Recoder le jeu "Plus ou moins" sans boucle, et constater la complexité asymptotique de l'algorithme.

TP précédent

Créer une fonction factorielle qui renvoie la factorielle d'un entier sans boucle.

```
#include <stdio.h>

int fact(int n)
{
    if(n == 0) // Condition d'arrêt
        return 1;

    return n * fact(n - 1); // Appel récursif
}

int main()
{
    printf("%d\n", fact(10));
    return 0;
}
```

TP précédent

Recoder le jeu “Plus ou moins” sans boucle, et constater la complexité asymptotique de l’algorithme.

```
#include <stdio.h>
int plusoumoins(int a, int b)
{
    if(a > b) // Condition d'arrêt
        return -1; // Tricherie

    char c;
    int res = (a+b)/2;
    printf("Votre nombre est-il %d ? ", res);
    scanf("%c", &c);
    getchar();
    switch(c)
    {
        case '+':
            return plusoumoins(res + 1, b); // Appel récursif (intervalle 2x plus petit)
        case '-':
            return plusoumoins(a, res - 1); // Appel récursif (intervalle 2x plus petit)
        case '=':
            return a;
        default:
            printf("Caractère non traité\n");
            return plusoumoins(a, b);
    }
}

int main()
{
    int res = plusoumoins(0, 100);
    if(res >= 0)... // Si res >= 0 alors res contient le nombre trouvé, sinon tricherie
    return 0;
}
```

Tableau

- Comment est stockée une variable ?

Tableau

- Comment est stockée une variable ?

Une variable est stockée dans une case mémoire, qui est déterminée par une adresse.
En C, on peut accéder à l'adresse d'une variable en ajoutant & devant celle-ci.

Exemple La variable x est stockée à l'adresse $\&x$, soit par exemple **@384647838**.

printf("%d\n", x) affiche x sous forme d'entier.
printf("%p\n", &x) affiche l'adresse de x.
scanf("%d", &x) stocke l'entrée utilisateur à l'adresse de x, soit dans x.

Tableau

- Qu'est-ce qu'un pointeur (en C) ?

Tableau

- Qu'est-ce qu'un pointeur (en C) ?

Une pointeur est une variable qui peut stocker une adresse.
En C, on utilise * pour signifier que c'est un pointeur.

Il est important de préciser le type de variable pour connaître sa taille.

Exemple
int x = 2; int *y = &x; y contient l'adresse de la variable x.

printf("%p\n", y) affiche l'adresse dans y, soit l'adresse de x.
printf("%p\n", &y) affiche l'adresse de y.
printf("%d\n", *y) affiche la valeur dans x.
scanf("%d", y) stocke l'entrée utilisateur dans y, soit dans l'adresse de x, soit dans x.

Tableau

```
#include <stdio.h>
int main()
{
    int a = 3;
    change(...);
    printf("%d\n", a); // doit afficher 4
    return 0;
}
```

Coder la fonction **change** pour que le contenu de la variable *a* soit modifié en 4.

Tableau

- Qu'est-ce qu'un tableau ?

Tableau

- Qu'est-ce qu'un tableau ?

Un tableau est une suite de cases mémoires successives.
En C, un tableau est déterminé par une adresse mémoire, une taille et un type.

On peut déclarer un tableau de deux façons : statique et dynamique.
Statique suppose qu'on connaisse la taille à la compilation, dynamique suppose qu'on connait la taille qu'à l'exécution (entrée utilisateur ou calcul).

On va pour le moment se concentrer sur la manière statique.

Tableau

int tab[4]; déclare un tableau d'entiers de 4 cases contiguës (qui se suivent).

tab est l'adresse de la première case du tableau (comme **&tab[0]**), il faut utiliser les opérateurs [] pour accéder à une valeur.

Exemple
printf("%p\n", tab) affiche l'adresse du tableau.
printf("%p\n", &tab) affiche l'adresse du tableau.
printf("%d\n", tab[1]) affiche la valeur dans la deuxième case du tableau.
printf("%p\n", &tab[1]) affiche l'adresse de la deuxième case du tableau.
scanf("%d", tab) stocke l'entrée utilisateur dans l'adresse du tableau, soit dans la première case.
scanf("%d", &tab[1]) stocke l'entrée utilisateur dans l'adresse de la deuxième case du tableau, soit dans la deuxième case.

Tableau

Chaque case d'un tableau se comporte comme une variable.
tab[3] = 2; attribue 2 à la quatrième case du tableau.

On peut initialiser un tableau case par case, ou alors de cette façon.
int tab[4] = {1,2,3,4};, où **tab[0]** vaut 1, **tab[1]** vaut 2, etc.

Remarque Pour initialiser toutes les cases à 0, on peut écrire
int tab[4] = {0};.

TP

Objectifs du TP :

- Créer un tableau et afficher l'adresse de la première et de la deuxième case. Constater pourquoi le type est important.
- Créer un tableau d'entiers de 10 cases et le remplir, puis afficher son contenu.
- Créer une fonction qui prend en paramètre un tableau d'entiers et sa taille, et qui affiche son contenu.