

# Algorithmique et Structures de données 1

## L2 2021-2022 Travaux Pratiques 3

---

Site du cours : <https://defelice.up8.site/algo-struct.html>

Les exercices marqués de (@) sont à faire dans un second temps.

Un fichier écrit en langage C se termine conventionnellement par `.c`.

Une commande de compilation est `gcc fichier_source1.c fichier_source2.c fichier_source3.c`.

Voici des options de cette commande.

- `-o nom_sortie` pour donner un nom au fichier de sortie (par défaut `a.out`).
- `-Wall` pour demander au compilateur d'afficher plus de Warnings
- `-Wextra` pour demander au compilateur d'afficher plus de Warnings
- `-std=c11` pour compiler selon la norme C11

Exemple : `gcc -Wall fichier1.c -o monprogramme`

---

### Exercice 1. Temps d'exécution

- Écrire un programme `tempsExecut.c` qui mesure le temps d'utilisation processeur d'une boucle `for` vide de  $n$  iterations, ensuite donner un bilan : nombre d'opération, temps d'exécution et nombre d'iterations moyen par milliseconde. Dans le programme recommencer la mesure en multipliant par 1,1 le nombre d'iterations et ce jusqu'à l'infini.
- Aide : utiliser la fonction `clock` de `time.h` qui retourne le nombre de <<ticks>> utilisé par le processus et utiliser la constante `CLOCKS_PER_SEC` définie dans `time.h` qui indique le nombre de <<ticks>> par seconde.

### Exercice 2. Fibonacci 2

Comparer le temps d'exécution de l'algorithme de fibonacci du td3 en comparant les deux méthodes (itératif et récursif) du td3. Laquelle semble la meilleur ? Illustrer votre propos par un programme `fibo.c` qui affiche ces différences.

### Exercice 3. Puissance

A partir de  $a$  on peut calculer  $a^n$  avec  $O(n)$  multiplications. ( $a \times a \dots \times a = a^n$ ). Trouver un algorithme qui utilise  $O(\log_2(n))$  opérations et implantez-le en C : `int puissance(int a, int n)`.

### Exercice 4. Quicksort

Pour trier (de façon croissante) un tableau selon l'algorithme du **quicksort** :

1. Un tableau de taille 1 ou 0 est trié, sinon :
2. On choisit une valeur quelconque dans le tableau : le pivot.
3. On permute les cases du tableau de façon que les cases à gauche du pivot contiennent des valeurs inférieures au pivot et que les cases à droite soient supérieures.
4. On trie la partie à gauche du pivot puis on trie la partie à droite (toujours selon l'algorithme du quicksort).

Écrire une fonction `void triRapide(int n, int* tab)` qui tri selon l'algorithme du Quicksort.

### Exercice 5. Quicksort probabiliste

1. Pour l'algorithme du quicksort précédent donner le pire des cas et sa complexité.
2. Implanter à nouveau l'algorithme du quicksort dans une fonction `triRapideP` mais en choisissant le pivot aléatoirement dans le tableau.
3. Quel est l'avantage d'un tel algorithme ?

### Exercice 6. Le compte est bon 2

Implanter la fonction `int compteAf(int tailleE, int E[], int s)` (qui fait ce que fait `compte` de l'exercice << Le compte est bon >> du TD3) qui en plus imprime la solution lorsqu'il y en a une.