TD Machine #4 (xxième séance de TD) de CS101

- Durée 1h30
- Pas de compte-rendu. Il est fortement recommandé de prendre des notes personnelles.
- Version 2022.1
- *author* Laure Gonnord

Préparation

Relire le cours sur les tris.

À vos marques

- Le TDM est réalisé sur les machines de TP Linux de la salle B141. **utilisez la VM "Laure Gonnord-compilation"**.
- Essayer le plus possible de faire seul·e (on apprend moins vite en binôme)
- Ne pas hésiter à demander de l'aide (on apprend!)
- Créer un répertoire pour les TPs de C nommé CS101, et travailler dans un sous répertoire nommé TDM4.
- Note : vous pouvez utiliser cloud.esisar.grenoble-inp.fr pour sauvegarder vos programmes.

Prêts

- Récupérer sur Chamilo le fichier `tdm4.c` qui sera à remplir dans cette séance.
- Compiler le fichier, exécuter. Observer le source, qui est à éditer dans la suite.

Algorithmes de tableaux d'entiers

- Concevoir puis écrire une fonction qui permet de vérifier qu'un tableau est trié:

```
bool is_sorted(int t[N])
```

Utiliser les fonctions fournies pour tester.

- Écrire une version récursive de la recherche dichotomique d'un élément dans un tableau trié.

```
bool find_in_sortedtab(int t[N], int el)
```

On pourra s'inspirer de la façon de concevoir le trifusion dans le cours (CM-tris) et commencer par programmer une fonction:

```
bool find_in_sortedtab_aux(int t[N], int el, int left, int right)
```

Programmer, tester. Pour tester sur un tableau aléatoire, on pourra demander à l'utilisateur un ou plusieurs entiers à chercher (avec, par exemple:

```
int int_to_find;
  scanf("%d", &int_to_find)
```

=, après avoir imprimé le tableau aléatoire. Sinon, tester sur un tableau fixe marche aussi.

- Retrouver sur papier un pseudo-code d'un des tris "simples" du cours (insertion, sélection").

```
void insertsort_tab(int t[N]) // ou selectsort_tab
```

Programmer, tester.

Si vous avez du temps

- Reconcevoir et programmer l'algorithme du crible d'Eratosthene (voir le TD3)