Portfolio C (2018-2019)

Guillaume Cau

Introduction :

Les travaux demandés sont séparé par TP et par exercice dans des répertoires. Dans chaque répertoire se trouve un makefile. Ce makefile permet de faire les commandes suivantes :

- make (compilation)

- make clean (suppression des binaires et de l'exécutable.)

- make run (lance l'exécutable.)

Malheureusement, si l'exécutable demande des paramètres, il faut le lancer manuellement.

TP n°2 :

Le but de ce TP était de ce familiariser avec la syntaxe du langage C et ses particularités. Même en ayant déjà fais du C, j'ai appris des chose en faisant ce TP.
Dans mon IUT, nos enseignants nous ont appris à utiliser les fonctions read/write/open et close qui sont les fonctions du système UNIX. Nous n'avions eu que peu d'exercices sur les fonctions utilisant le fonction FILE de la librairie stdio.h.
Ce TP m'a donc permis d'apprendre à utiliser ces fonctions qui doivent être utilisées en priorité.
Mon expérience d'utilisation des fonctions read/write etc m'ont permis de comprendre plus facilement le fonctionnement des fonctions de stdio.

TP n°3 :

Le but de ce TP était de s'entraîner à utiliser le C en utilisant des algorithmes. J'avais déjà implémenté ces algorithmes lors de mon DUT. Je n'ai donc pas eu trop de problèmes lors de ce TP.
Il y a cependant la recherche par dichotomie qui m'a posé problème. En effet, j'ai du mal à me visualiser comment couper le tableau en 2 sans dessins. J'ai donc perdu de ce temps de ce côté là.
Après, retourner le bon indice a été plus ardu que prévu. Les indices que je renvoyai lorsque l'élément était trouvé était celui du tableau local. Donc celui du sous-tableau et non du tableau principal. J'ai donc été obligé de faire la somme de l'indice trouvé avec la taille du tableau divisé par deux lorsque

l’élément se trouvait dans la partie de « droite » lors d’un appel récursif.
Lorsque l’élément ne se trouvait pas dans le tableau, la fonction renvoyait -1. Mais, la valeur -1 s’additionnait désormais avec la taille du tableau divisée par 2 lorsque la fonction était rappelé à droite du tableau. J’ai donc du faire une condition pour éviter d’effectuer une somme avec une valeur -1.

TP n°4 :

Le but de ce TP était d'apprendre à utiliser une pile (LIFO). Je n'ai pas vraiment eu de difficulté à créer ses fonctions associées.
Le seul problème était de savoir dans quel sens effectuer les opérations binaire (à deux arguments).

TP n°5 :

TP n°6 :

Je n'avais jamais fait de backtracking jusqu'à aujourd'hui. Ce TP m'a donc permis de comprendre comment procéder. De plus, j'ai appris une manipulation sympa. Lorsque l'on manipule un gros tableau, il est impossible de le placer dans une fonction car son existence même va faire exploser la pile. Il faut donc le placer en tant que variable globale. C'est peut être plus moche, mais ça fonctionne. Du moins, tant qu'on utilise pas le tas.

Sinon, dans l'exercice du sudoku, j'ai utilisé les divisions et les modulos pour déterminer les coordonnées plutôt qu'une boucle. De cette manière, je visualisait plus facilement l'algorithme parcourir le tableau case par case.
Dans l'exercice précédent, j'avais fait l'inverse. J'avais utilisé des boucles.

Avec le recul que j'ai pris, je ne suis pas certains d'avoir utiliser les méthodes les plus efficaces. J'ai utilisé les méthodes que je comprenais.

En terme de code, j'ai eu un problème lors du développement du sudoku. J'avais mis la vérification pour savoir si la case était vide dans la boucle des 9 chiffres. Le temps d'exécution était très lent. Je n'ai pas tout de suite aperçu ce problème. Je ne l'ai même jamais remarqué. Mon algorithme ne marchait pas, il ne terminait jamais . Je me suis dis que j'allais l'optimisé et c'est là que j'ai corrigé le problème sans réfléchir. La fonction s'est mise à marcher. Je ne comprends toujours pas pourquoi ça a fonctionner vu que c'est au plus une dizaine de comparaisons supplémentaires par récursion. Et pourtant, le temps d'exécution était bien plus que 10 fois plus lent. Enfin, je viens de comprendre en écrivant ces lignes… C'est plutôt exponentiel le nombre de comparaison en plus dans la récursion.

TP n°7 :

Malheureusement, je n’ai pas eu le temps de beaucoup travailler sur le sudoku. Je n’ai donc pas eu le temps de faire des fonctions propres pour toutes les fonctionnalités. Il en résulte que le code n’est pas très compréhensible.

Sur tous ce TP, je me suis basé sur des points de coordonnés choisi arbitrairement. Après, il suffisait de faire quelques calculs pour convertir les coordonnées de l’écran en coordonnées de tableau.

TP n°8 :

real 1m24,296s

user 1m24,293s

sys 0m0,000s