* Le nom du projet ?
* Qui a réalisé le projet ? (quand il n'a pas été fait seul...)
* Qu'avez vous appris ?
* Comment avez-vous appris (méthode/démarche) ?
* Quelles actions vous ont-elle permis de développer vos compétences ?
* Quels éléments factuels illustrent votre maîtrise des notions mobilisées ?
* Quels choix fondamentaux principaux avez-vous faits durant vos projets ?
* Une fois vos productions établies (un projet fini), une critique réflexive des choix que vous avez faits ?

**Vue d’ensemble**

Emilie Marti – INFO1, G2

Chargé de TP : FRANCIS Nadime

Tous les TP sont gérés sous le gestionnaire de versions git, avec github. Le lien d’accès (voire de git clone) est le suivant : <https://github.com/hyliancloud/INFO_C.git> . Tout est pushé sur la branche master, les branches ne sont créées que si je veux avoir deux versions d’un projet ou TP (pour l’instant j’en ai pas eu besoin). Ceci me permet de travailler sur n’importe quelle machine et avoir une sauvegarde de sécurité de mon travail.

Ma démarche pour apprendre à résoudre les TP est généralement la suivante :

* Si je connais l’algorithme, je tente ma méthode en premier. Si elle marche, je la tente sur plusieurs cas pour être sûre qu’elle marche toujours (ou pour déterminer ses limites).
* Si je ne connais pas l’algorithme, je fais des recherches sur internet, notamment openclassroom qui a de bons tutoriels, puis je mets en pratique en C. Je consulte aussi d’autres sites comme stackoverflow.
* Les erreurs de compilation sont corrigées en prenant connaissance du type d’erreur (si je ne le reconnais pas je cherche) et en corrigeant.
* Si j’ai des erreurs de comportement, j’essaie de debugger avec des printf qui témoignent du devenir de la fonction à chaque pas. Plus tard, lors de projets plus conséquents, j’essaierais d’apprendre à utiliser les debugger tels que gdb.
* Si je suis bloquée, je demande au chargé de TP ou à un collègue.
* J’essaie d’implémenter les conseils d’optimisation du chargé de TP, tout en gardant les versions antérieures pour comparer.

Pour le semestre 1, les TP ont été faits de façon individuelle.

Pour le premier semestre, j’ai plus ou moins suivi le fil rouge. Les TP réalisés sont les suivants, pour un total de 51 points :

* TP3 (exos)
* TP4 (pile d’entiers)
* TP5 (tableaux fin -1)
* TP6 (backtracking)
* TP7 (sudoku graphique)
* TP9 (exos malloc)

**TP3 – Quelques exercices**

Emilie Marti

Ce TP contient des exercices différents et indépendants les uns des autres ; on aurait pu les faire dans des fichiers indépendants, cependant j’ai choisi de les faire sous forme de modules qui vont être appelés par le même main.c, en rajoutant une forme modulaire à mon programme. Aussi, le premier exercice étant l’implantation de la fonction puissance, elle a pu être réutilisée plus tard lors des conversions de base, ce qui montre qu’elle marche plutôt bien.

Au niveau de la puissance, j’ai implémenté une version optimisée de la fonction de puissance récursive à la suite du conseil de M. Francis.

J’essaie de factoriser mon code ; c’est plus facile à comprendre et à debugger si besoin (chaque fonctionnalité a sa propre fonction, on évite de tout faire dans une fonction).

De façon générale, j’ai pris connaissance de certains algorithmes (la puissance en récursif, les tris dichotomique et à bulles et la conversion des bases) et à les mettre en place en C, ainsi qu’à trouver une manière d’optimiser une fonction récursive (pour qu’elle trouve un résultat « intelligemment » en ne passant pas par tous les cas).

Les compétences suivantes étaient à valider :

* I/O

Compétence mise en place à travers les entrées sur la sortie standard (avec des scanf), contrôlés avec des conditions pour ne pas prendre en compte n’importe quel input donné par l’utilisateur. Le flux de sortie utilisé est principalement stdout.

* Type

Les types utilisés sont les entiers et les caractères, soit sous sa forme, soit sous la forme de pointeurs.

* Programme

Les fonctions sont utilisées et testées dans le main.

* Compilation

Un makefile est créé pour compiler facilement avec la simple commande « make ». Les modules sont convertis en fichiers objets .o puis utilisés pour créer l’exécutable de sortie (qui fonctionne sur linux).

* Récursivité

La récursivité est utilisée sur la puissance et sur la recherche dichotomique. Pour la puissance, deux fonctions récursives sont proposées ; l’une est plus optimisée par rapport à l’autre car elle va faire moins de tours pour calculer la puissance que la première.

* Tableaux

Les tableaux 1D sont utilisés pour les chaînes de caractères qu’il faut traiter et convertir puis pour les tableaux d’entiers qu’il va falloir trier lors des deux derniers exercices.

Les compétences supplémentaires mises en œuvre sont les suivantes :

* Bibliothèque

Les bibliothèques utilisées sont : stdio, stdlib (pour les fonctions standard), string pour les chaînes de caractères, math pour des fonctions mathématiques puis time pour avoir le nombre de secondes depuis le début de l’existence de l’informatique (pour changer la seed lorsqu’on veut avoir des nombres random).

* Module

Pour rendre le rendu plus propre, j’ai décidé de faire un seul main qui testera les fonctions de tous les exercices, mais celles-ci seront placées dans un fichier source différent, qui sera rappelé sur le main via son fichier header (.h).

**TP4 – Pile d’entiers**

Emilie Marti

Sur ce TP, on apprend à utiliser la pile pour stocker des valeurs en mémoire et les récupérer dans l’ordre. Il s’agira d’un programme modulaire dont le header (.h) est fourni ; les fonctions sont déclarées, et on doit les coder en conservant le prototype. En gros on codera la pile qui va être ensuite utilisée par une calculatrice graphique (codée avec MLV) qui va utiliser ses fonctions. Finalement, certaines fonctionnalités vont être rajoutées à la calculatrice.

Pour calculer les modulo, il faut taper « m » ; en effet, le symbole du module, % , n’est pas géré par la librairie graphique MLV et taper ce symbole au clavier fait directement planter le programme.

Les compétences à acquérir pour ce TP sont les suivantes :

* I/O

Les input des utilisateurs sont gérés via les fonctions de la librairie MLV.

* Type

Les types mis en œuvre sont les entiers, les flottants et les caractères.

* Programme

Les fonctions sont codées dans son modules et utilisées dans un main qui va aussi gérer les fonctions graphiques pour que le programme marche sur la calculatrice graphique.

* Module

Le programme est sous la forme modulaire ; toutes les fonctions sont dans des fichiers spécifiques qui dépendent de sa fonctionnalité.

* Compilation

Un makefile est mis en place pour une compilation simple ; un seul « make » est à écrire sur le terminal et le makefile créera les fichiers objets qui seront utilisés pour créer l’exécutable de sortie. Les fichiers objets sont ensuite éliminés car non utilisés pour plus de clarté dans le dossier.

* Bibliothèque

Pour ce TP, la bibliothèque graphique MLV a été utilisée ; les fonctions principales étant données, il a fallu les adapter pour mettre en œuvre les fonctionnalités. Les fonctions de la pile peuvent être associées à une mini-bibliothèque utilisée sur la calculatrice.

**TP5 – Tableaux fin -1**

Emilie Marti

Sur ce TP, on apprend à manipuler des tableaux d’entiers, triés ou pas, qui ont la particularité de se terminer par -1, pour simuler le caractère de fin de chaîne ‘\0’.

**TP6 – Backtracking**

Emilie Marti

Sur ce TP,

**TP7 – Sudoku graphique**

Emilie Marti

Sur ce TP

**TP9 – Exos malloc**

Emilie Marti

Sur ce TP,