SORBONNE UNIVERSITÉ

TD : Programmation en C — semaine 8

6 avril 2018

Objectif(s)

- ✯ Lecture d'un code existant.
- ✯ *Refactoring* de code existant.
- ✯ Renforcement sur la notion de modularité.

— **Note 1** : Pour tous vos programmes, vous utiliserez la commande `make` pour compiler.
— **Note 2** : Pour tous vos programmes, vous ferez des fonctions de 5 lignes maximum (sans comptez les commentaires, accolades seules ou en-têtes de fonctions).
— **Note 3** : Pour tous vos programmes, vous ferez des fichiers de 50 lignes maximum (sans comptez les commentaires).

Exercice 1 – Bataille navale

Vous venez d'arriver dans votre nouvelle entreprise, *Titanic Inc.*, et on vous demande de reprendre le code de l'ancien stagiaire. Il n'était pas très doué et le code est un fou..., un bazar sans nom. Pour votre première mission, on vous demande de reprendre le code et de faire plusieurs petites tâches indiquées ci-dessous.

En entreprise, on part rarement de zéro : on arrive sur un projet existant et on doit le modifier. Par conséquent, la lecture du code d'autrui est une compétence très importante.

1. Découvrez le code `navalBattle.c` disponible sur le site du cours, ainsi que la documentation accompagnant ce code. En vous basant sur le Makefile de la leçon précédente, vous l'adaptez pour compiler ce nouveau fichier, et puis vous l'exécuter et découvrez les fonctionnalités du programme. *Note* : La documentation contient trop d'information, à vous de lire seulement ce qui vous intéresse.

   Le plus important quand on arrive sur un nouveau projet est de ne pas essayer de tout comprendre dès le départ. Sur des projets conséquents, cela est impossible de toutes manières.
2. Une astuce pour comprendre comment un code fonctionne est d'essayer de le modifier en ajoutant une petite extension. On vous demande d'ajouter un menu qui permet à l'utilisateur de choisir entre plusieurs mode de jeu :
   1. Joueur VS. Ordinateur (le mode implémenté pour le moment).
   2. Ordinateur VS. Ordinateur (un jeu automatique).
   3. Joueur VS. Joueur

Exercice 2 – Cap refactoring mon capitaine !

Quand les personnes vous précédent n'ont pas été très soigneuse dans leur code, il faut entamer une étape de *refactoring*. Cela consiste à modifier le code afin de le rendre plus lisible (et potentiellement d'éliminer des bugs au passage).

1. Séparer le code en plusieurs fichiers tel que aucun fichier ne fait plus de 50 lignes.
2. Refactorer les fonctions tels qu'elles fassent au maximum 5 lignes (commentaires, accolades et entête de fonctions exclus).

Exercice 3 – Une IA à la conquête des océans

1. L'intelligence artificielle dans notre jeu est un peu stupide : elle tire au hasard tout le temps. Vous êtes en charge de l'implémentation de la nouvelle intelligence artificielle, afin de battre même ceux qui ont de la bouteille. Une première idée d'amélioration est de ne pas tirer deux fois au même endroit.
2. Pour améliorer votre IA, réfléchissez quoi faire quand vous trouvez un bateau. Vous devrez probablement utiliser des structures supplémentaires pour que l'IA stock des informations entre deux coups.
3. Améliorer librement votre IA. Vu que votre code est extrêmement modulaire, il vous sera facile de défier un camarade et de confronter vos deux intelligences artificielles.