UNIVERSITÉ PARIS 13

# Projet Structures de données et algorithmes
# Codage binaire des caractères

---


Binôme

Mohand Lounis **BENSEKHRI** 11710457

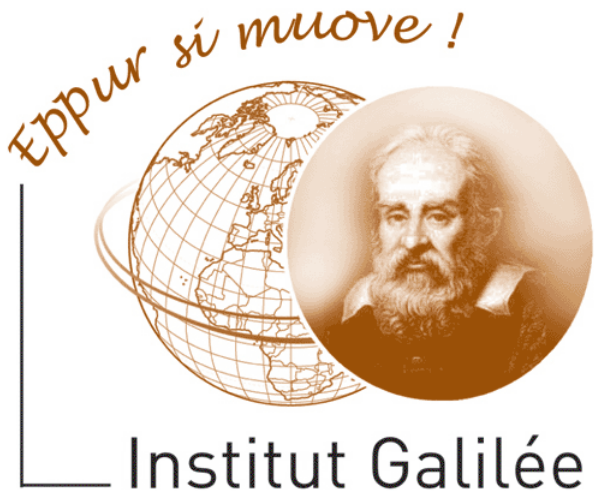
Sofiane **HAMMOUM** 11508506


Année Universitaire : 2018 / 2019

***L2 informatique***


**M. HAMIDI**

**M. VALENCIA**

# TABLE DES MATIERES

# Introduction

Au cours de ce projet de Structures de données et algorithme nous avons pu mettre en pratique toutes les nouvelles connaissances que nous avons acquises au cours de ce premier semestre sur les arbres binaires et les tas et aussi performer les anciennes connaissances acquises.

Ce projet ne nous a pas pris beaucoup de temps pour le mettre en œuvre, c'est pour cela que nous avons choisi d'envoyer une première version de ce que nous avons fait qui est nommé 1.1 qui est le mini projet demandé qui a eu un seul fichier.c et un rapport.

Dans cette deuxième version nous avons pu performer le logiciel de codage en effet maintenant il est apte de coder un fichier entré par l'utilisateur par le biais de son, et aussi nous lui avons rajouté la possibilité d'exécuter la version 1.1 du projet. Le logiciel peut notamment décoder un fichier qu'il a codé. Il y a aussi la possibilité de coder une chaine de caractères passé en argument.

Tout cela est effectivement mentionné dans le menu de l'exécution du logiciel.

Nous avons rencontré plusieurs difficultés surtout pour la manipulation des fichiers dont on parlera un peu plus loin dans le rapport.

# Explications sur le code

A propos du code nous l'avons performé au mieux pour éviter tout bug de la part du logiciel. Le programme en tout est constitué de 8 fichiers qui sont :

codage.h / codage.c / fichier.h / fichier.c / prog.h / prog.c / main.c / Makefile

Tous les fichiers sont commentés et en haut de chaque fichier.c on peut trouver la description de ce dernier et de ses fonctions.

- Dans les fichiers codage nous pouvons trouver les fonctions basiques qui permettent le codage (création du tas, de l'arbre de codage, et impression des codes).
- Dans les fichiers fichier nous trouverons les fonctions qui permettent la manipulation des fichiers et ainsi le codage et le décodage de ces derniers.
- Dans les fichiers prog nous trouverons les fonctions qui permettent une saisie sécurisée et les fonctions qui offrent la possibilité d'exécuter tel ou tel code.
- Le fichier main comporte la fonction principale main.
- Le fichier Makefile permet la compilation de ces différents fichiers tous ensemble et ainsi générer un fichier ./codage qui est l'exécutable du programme.

**Explication A Propos de l'exécution du programme :**

Après compilation du programme le terminal nous génère un fichier exécutable nommé codage.
On l'exécute et le logiciel se lance.

1. Le logiciel affiche un message de bienvenu accompagné d'un menu, nous devons saisir un des nombres qui désigne chacun une fonctionnalité spécifié.
2. Si l'on essaye de taper une autre chose que ce qui est demandé le logiciel nous demande une nouvelle saisie.
3. Après une saisie acceptée :

Nous avons cinq possibilités de choix qui sont :

a. Exécuter la version 1 :
   i. Le logiciel nous demande le nombre de caractères que notre alphabet contient il faut que ce nombre soit strictement positif sinon il ne le redemande.
   ii. Ensuite il demande de saisir chaque char avec sa fréquence si la fréquence ; il faut que la fréquence soit strictement positive sinon il nous la redemande.
   iii. Après la saisie de tous les chars avec leurs fréquences respectives le logiciel nous affiche le tas initial, l'arbre de codage et enfin le code de tous les chars entrés générés par l'arbre de codage.

b. Coder une chaine de char :
   i. Le logiciel nous demande d'entrée une chaine de char.
   ii. Ensuite il nous affiche le code de chaque char de l'alphabet de cette chaine, et enfin le code complet de cette chaine.
   NB : l'alphabet c'est-à-dire les symboles différents.
   Exe : l'alphabet de « BONJOUR » est : BONJUR.

c. Coder un fichier :
   i. Le logiciel nous demande le chemin du fichier.
   ii. Si le fichier donné par ce chemin n'existe pas il nous redemande un nouveau chemin.
   iii. Ensuite le logiciel nous affiche le contenu du fichier à coder, le code de tout l'alphabet du fichier à coder. Et enfin le contenu du fichier.cds créé.

iv. Si le fichier à coder a une extension par exemple .txt le fichier codé enlève cette extension et met à la place .cds si le fichier à coder n'a pas d'extension il lui rajoute .cds.

NB : Un fichier codé est reconnu par son extension qui est .cds

d. Décoder un fichier codé :
    i. Le logiciel nous demande le chemin du fichier à décoder il faut que ce dernier soit déjà créé et qu'il ait l'extension « .cds », sinon il nous demande un nouveau chemin.
    ii. Après cela il nous demande d'entrer le chemin du fichier qui a permis d'avoir le fichier codé ; et cela seulement pour récupérer l'alphabet utilisé et le nombre d'occurrence de cette dernière.
    iii. Le fichier ne doit pas être d'extension .cds ou .dcd et il faut qu'il existe sinon il nous demande un nouveau chemin.
    iv. Le radical des deux fichiers .cds et standard doit être identique sinon le décodage a échoué
    v. Après cela, le logiciel nous affiche ce que contiennent le fichier codé et le fichier .dcd créé qui est le décodage du fichier .cds
    vi. Un fichier décodé par le logiciel est reconnu grâce à son extension qui est .dcd
e. Quitter le logiciel :
    i. Le logiciel affiche un message indiquant sa fermeture.

Nous avons notamment utilisé le logiciel valgrind pour nous faciliter la détection des erreurs de segmentations et des fuites de mémoires (le logiciel à 0 fuite de mémoire).

# PROBLEMES RENCONTRES

Nous avons rencontré plusieurs erreurs et difficultés durant cette deuxième version du projet en effet nous avons pu corriger la totalité, mais il en reste moins que nous n'avons pas pu corriger certains bugs dont nous pensons connaître la cause.

Le problème que nous avons rencontré est que quand nous essayons de coder et décoder plusieurs fichiers en une seule exécution du logiciel ça arrive qu'il décode mal les deux derniers fichiers. Nous pensons que l'erreur vient des rewind() qui manquent pour remettre le curseur parcourant le fichier à 0, car si nous essayons de coder les fichier séparément (c'est-à-dire chacun après une exécution du logiciel) les codes sont BON.

La deuxième faute que nous avons pu détecter aussi c'est que si nous essayons de décoder un fichier codé ensuite on code le fichier standard, le programme s'arrête à cause d'une erreur de segmentation. Nous pensons que c'est dû au même problème que pour le premier bug car coder un fichier seul est une fonctionnalité qui fonctionne.