# TP2 : Attaquer le code de César

**Dans ce TP, vous devez répondre aux questions posées en écrivant les fonctions demandées avec le bon prototype. Tout manquement à ces instructions entrainera une perte de points lors du rendu.**

Dans ce TP, nous verrons comment casser le code de Cesar.

## 1 Le TP1 : dernière chance pour le corriger et le compléter (45mn maximum)

Vous trouverez, sur Moodle, le résultat des évaluations automatiques du TP1, sur le codage de Cesar. Si votre programme a été noté comme ayant des problèmes, vous pouvez le corriger dès à présent, et effectuer un nouveau dépôt de votre travail (ce sera le dernier concernant le TP1). N'oubliez pas de suivre à la lettre les instructions de rendu données dans le TP1. Vous devrez effectuer votre rendu avant 14h30.

## 2 Le chiffre de César (2h environ)

Comme vous l'avez déjà vu, le chiffre de César consiste à décaler, dans un message à transmettre, toutes les lettres d'un certain nombre de places. Par exemple, si le décalage vaut trois, alors la lettre 'a' devient 'd', 'b' devient 'e', ..., 'z' devient 'c'. Le destinataire du message doit connaître ce décalage afin de pouvoir l'appliquer (de façon inversée) sur le message chiffré et obtenir le message

La faiblesse de ce chiffrage est son nombre limité de possibilités... Si quelqu'un souhaite casser ce code, il peut essayer tous les décalages possibles (26 en tout) sur une partie du message, et conserver le décalage qui fait apparaître des mots intelligibles. Cette méthode aurait été utilisée par Jules César pour échanger des correspondances secrètes : l'analphabétisme courant à l'époque permettait, avec une méthode aussi simple, de protéger son message. Dans ce TP, nous utiliserons une méthode plus automatique pour décrypter le message.

Vous trouverez, joints à ce TP, deux textes codés avec la méthode de César. Les deux textes chiffrés sont des messages en français, la lettre qui y apparaît le plus souvent doit donc être le 'e' du message déchiffré (en français, la lettre 'e' est celle qui apparaît le plus souvent). Vous devrez trouver, dans les deux messages, quelle lettre apparaît le plus souvent et calculer le décalage à appliquer. Par exemple, si vous trouvez que le message chiffré contient une majorité de 'o', cela signifie que toutes les lettres du message d'origine ont été décalées de 10 places dans l'alphabet, et que le 'e' est ainsi devenu le 'o'.

Votre programme doit lire le fichier chiffré d'entrée, trouver la lettre qui y apparait le plus souvent, et en déduire le décalage inverse à appliquer au texte pour retrouver le message d'origine. Notez bien que cette méthode, comme toutes les méthodes de déchiffrage d'un message, ne fonctionne que si le message est assez long...

Voici les étapes à suivre pour construire votre programme :

**Questions**

1. (45mn) Faîtes une fonction **char prochaineLettre(FILE* in)** qui prend en paramètre un fichier et renvoie la prochaine lettre du fichier (elle devra donc renvoyer les lettres et sauter les espaces, les ponctuations, les chiffres, ...) du fichier. Réfléchissez à ce que votre fonction devrait renvoyer dans le cas d'une erreur (fichier non ouvert, fin de fichier atteinte, ...) ?

2. (45mn) Faîtes une fonction **char lettreLaPlusFrequente(FILE *in)** qui prend en paramètre un fichier et renvoie la lettre qui y apparaît le plus souvent.

3. (30mn) Écrivez une fonction **void decodageCesar(FILE *in, FILE *out)** qui prend en paramètre un fichier crypté par la méthode de César et écrit le message décrypté dans le fichier de sortie.

# 3 Rendu du décodage de César

Vous devez rendre le travail réaliser pour le TP2 sur Moodle ou par mail, à l'adresse

chaussard.laga+idb@gmail.com

Ce rendu doit suivre le format qui vous avait été précisé précédemment, à savoir que votre programme devra s'exécuter à l'aide de la commande

```
./cesar_decodage mon_fichier_code.txt mon_fichier_decode.txt
```

De plus, votre programme devra compiler à l'aide de la commande

```
gcc cesar_decodage.c -o cesar_decodage
```

Si votre ligne de compilation est plus complexe, vous devrez alors la spécifier dans un *Makefile*. Tous vos fichiers devront être directement placés dans un fichier zip (et ne pas être dans un sous dossier du fichier zip), dont le nom sera

```
VotreNom_VotrePrenom_VotreNumeroEtudiant.zip
```

Les formats de compression acceptés sont .zip, .tgz, .tar.gz, .7z, .rar.

Si vous avez un binôme, il vous faudra écrire son nom, prénom et numéro d'étudiant séparés par des espaces, en plus des vôtres, dans un fichier *binome.txt*, dont le format sera

```
Nom1 Prenom1 Numero_etudiant1
Nom2 Prenom2 Numero_etudiant2
```

Vous ne devez rendre le fichier qu'une seule fois par binôme.