Adrien ROY, Oussama ACHKOUK | Groupe : TD3 TP5

COMPTE RENDU
Programmation Orienté Objet C++
TP4 et TP5

Présentation et choix :

Pour ce TP, nous avons décidé de présenter les TPs 4 et 5 ensemble car nous avons initialement commencé par le TP 3 et le TP 5, mais nous avons ensuite décidé de rassembler nos projets pour un travail plus collaboratif. Pour ces TPs, nous avons choisi d'utiliser Replit plutôt que GitHub car nous nous sommes retrouvés sur de nombreuses machines différentes et mettre en place notre environnement de développement à chaque fois semblait prendre plus de temps que ce que nous ferait gagner le versioning de GitHub. De plus, Replit permet de travailler sur la même section en même temps, ce qui encourage la collaboration. Nous avons bien entendu testé la compilation de notre projet sur WSL avec Clang avant de pousser la partie finale sur GitHub.

TP4 et ses fonctionnalités :

Le TP4 va implémenter différentes méthodes de chiffrement et permettra de gérer l'interaction entre plusieurs classes ainsi que l'héritage. La classe principale utilisée sera "Encrypt" qui servira de classe de base pour toutes les autres classes. Les variables **"_encrypt"** et **"_plain"** seront définies en **"protected"** pour permettre aux classes héritant de **"Encrypt"** de les modifier sans problème, ce qui simplifiera le code. Nous allons également définir toutes les fonctions nécessaires dans cette classe, comme une fonction pour lire un fichier d'entrée et une fonction pour écrire un fichier de sortie. Il y aura également des getters pour "encrypt" et "decrypt" qui, bien qu'ils n'ont pas été utilisés, pourraient théoriquement être utiles si on faisait un historique des derniers codages et décodages. Enfin, il y aura deux fonctions virtuelles pures qui forceront toutes les classes héritées à les définir.

Nous avons également créé une classe **"BasicEncrypt"** qui copie simplement le texte chiffré dans le texte déchiffré et vice-versa. Le fichier "main.cpp" est très épuré et permet d'appeler facilement n'importe quelle fonction "encrypt/decrypt" comme ceci :

```
<classe utilisée> instance;

instance.read();
instance.encode/decode();
instance.write();
```

Maintenant que cette partie est terminée, nous pouvons nous concentrer sur les fonctions d'encryption et de décryption des différentes classes demandées. Nous avons implémenté les codes **caesar, caesar2, vigenere** et pour le TP5, Enigma.

Pour tous ces codes, nous analysons un string qui est un tableau de caractères. Chaque caractère n'est modifié que par la clé et sa position. Nous pouvons donc simplement itérer sur le string et appliquer les opérations nécessaires à chaque caractère pour encoder et décoder.

Le code de César est le plus simple, on prend le caractère ASCII "a" et on y ajoute 3 (‘a’+3 => d). Cependant, certaines exceptions doivent être prises en compte, par exemple, nous voulons que ‘z’+3 => c. Cependant, si nous testons informatiquement, nous remarquons que ‘z’+3 vaut '}'.

Pour chiffrer un caractère, on utilise la formule (caractère - 'a' + 3) % 26 + 'a'. Cela permet de décaler la valeur ASCII du caractère de 3 places vers la droite, en utilisant l'opérateur modulo pour éviter les dépassements de 26. Par exemple, pour chiffrer la lettre 'z', on utilise la formule (122 - 97 + 3) % 26 + 97, ce qui donne 99 (correspondant à la lettre 'c').

Pour déchiffrer un caractère chiffré avec l'algorithme de César, on utilise la formule (caractère - 'a' - 3 + 26) % 26 + 'a'. Cela permet de décaler la valeur ASCII du caractère de 3 places vers la gauche, en utilisant l'opérateur modulo pour éviter les dépassements de 26. Et pour éviter de retrouver le modulo d'un nombre négatif, on rajoute 26. Il suffit maintenant de bien relier les liens et l'algorithme de César est prêt à l'emploi.
Pour utiliser cette méthode avec des caractères de la langue française, on peut utiliser la même formule en remplaçant 'a' par '!' et 26 par 93. Pour chiffrer un caractère, on utilise la formule (caractère - '!' + 3) % 93 + '!'. Et pour déchiffrer un caractère, on utilise la formule (caractère - '!' - 3 + 93) % 93 + '!'.

le code de vigegnére reprend le code cesar cependant la clé ne vas pas être fixe il est donc nécessaire d’avoir deux itérateurs

le premier itère le string que l’on veut encoder . Le second itère la clés d’encodage
supposons que on a la clé « clé » et le message « abc »

on prend le premier caractère du message ‘a’ et on prend le premier caractère de la clé ‘c’ qui vaut 3

on vas donc avoir (‘a’ – ’a’+3)%26 + ‘a’ exactement comme pour césar

on itère donc nos deux string ce qui fait ‘b’ et ‘l’

on as donc (‘b’- ‘a’ +12)%26 et quand on arrive a la fin de notre clé on retourne sur le premier caractère de celle ci.

L’opération de décodage est donc triviale et ne fait que réutiliser ce qui a été fait auparavant.

TP5 :

le TP5 reprend exactement les même base que l’exercice 4 d’où la combinaison des deux

la ou les codages du TP4 était des opérations mathématiques simples et facilement réversible Enigma est plus complexe

interessont nous a la machine physique elle a une entrée ,des rotors effectuant une transformation lettre par lettre préenregistrée et un brouillage amené par les décalages successif de ses roues

on as donc crée ses rotors qui ont pour propriétés un décalage de base , une clé de transformation et ,un niveau indiquant si elle tourne a chaque lettre toute les 26 lettres ou les 676 lettres

le rotor prend une lettre calcule son index y rajoute le décalage de base et le nombre de fois ou elle a tourné fait un module par 26 ce qui donne un indice entre 0 et 25

on cherche cette position sur la cle de transformation et on as la lettre de sortie

il suffit de répéter cette opération 3 fois en faisant attention de bien passer la bonne suite de caractère et nous avons donc le message codé par notre pseudo système enigma.

La ou Enigma diffère des autres systèmes est que l’opération mathématique de renversement de cette opération n’est pas triviale la méthode utilisée n’est probablement pas optimale et pourrait être améliorée

on crée une fonction rotorinverse qui prend une lettre du message codé

rotor inverse vas chercher le string contenant la clé de transformation pour savoir a quel indice se trouve cette lettre

après cela on peut retirer a l’indice de la lettre le décalage de position dans lequel serait le rotor et on obtient donc la lettre que le rotor initial a transformé

on répété cette opération 3 fois avec les mêmes clés de transformation que l’encryptage et le message est décodé.

Conclusion:

En conclusion, le TP4 implémente différentes méthodes de chiffrement en utilisant la classe "Encrypt" comme classe de base pour toutes les autres classes. Les variables **"encrypt"** et **"plain"** sont définies en **"protected"** pour permettre aux classes héritant de **"Encrypt"** de les modifier facilement. Nous avons également défini toutes les fonctions nécessaires pour lire et écrire des fichiers d'entrée et de sortie. Nous avons implémenté les codes caesar, caesar2, vigenere et pour le TP5, Enigma. Pour chiffrer et déchiffrer, nous avons utilisé des formules mathématiques simples qui utilisent la clé et la position du caractère pour encoder et décoder. Ces méthodes peuvent également être utilisées pour chiffrer plus de caractères en remplaçant 'a' par un autre caractère de début de la séquence et 26 par le nombre de caractères dans la séquence.
L'Utilité des TP est assez différentes pour les deux membres du groupes l'un plutôt habitué au C pur a été surpris de l'utilité des classes pour la réutilisation et l'autre a appris a mieux manipuler tout ce qui est tableau et conteneurs

malheureusement pour Adrien le grand défaut du travail dans ce TP est l'obstination a faire enigma d'une certaine manière (sans avoir a chercher une lettre dans le tableau original)c'est un objectif qui n'as pas aboutit et un gâchis de temps de travail pour gagner quelques instants sur le temps processeur qui n'est tout simplement pas évalué dans l'exercice.

pour Oussama le défaut est l'indécision plutôt que prendre le problème qui empêche la compilation et essayer de le traiter maintenant quitte a demander de l'aide le problème est ignorée et se retrouve a être géré plus tard dans des conditions moin avantageuses.

Projet github :https://github.com/barresonn/projeitC313