Projet Langages Avancés

Cryptage & Décryptage

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

Breton Maximilien & Fostier Nicolas

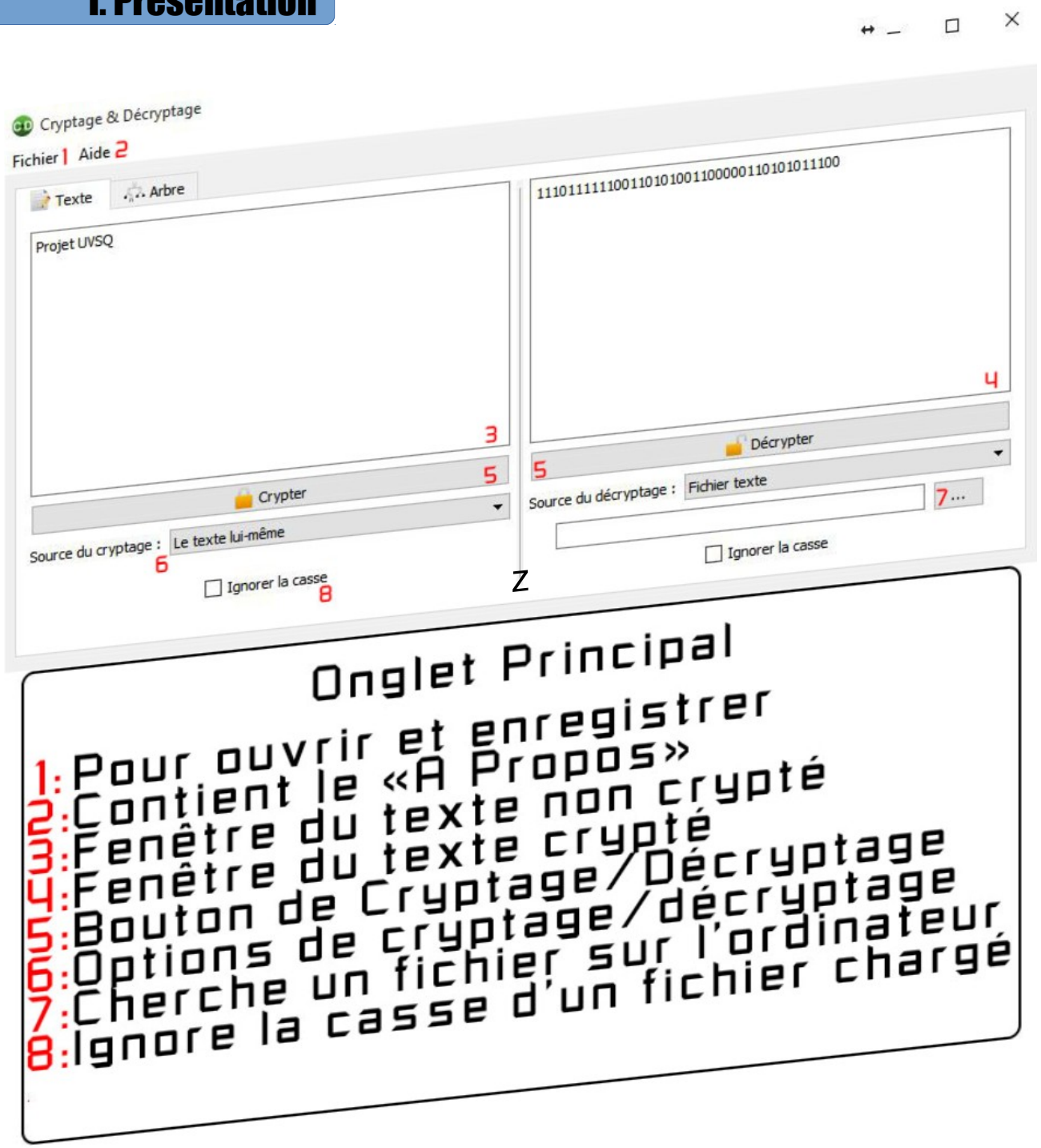
Projet réalisé dans le cadre de l'UE « Langages Avancés » de Mme L.Kloul

I. Présentation générale du projet

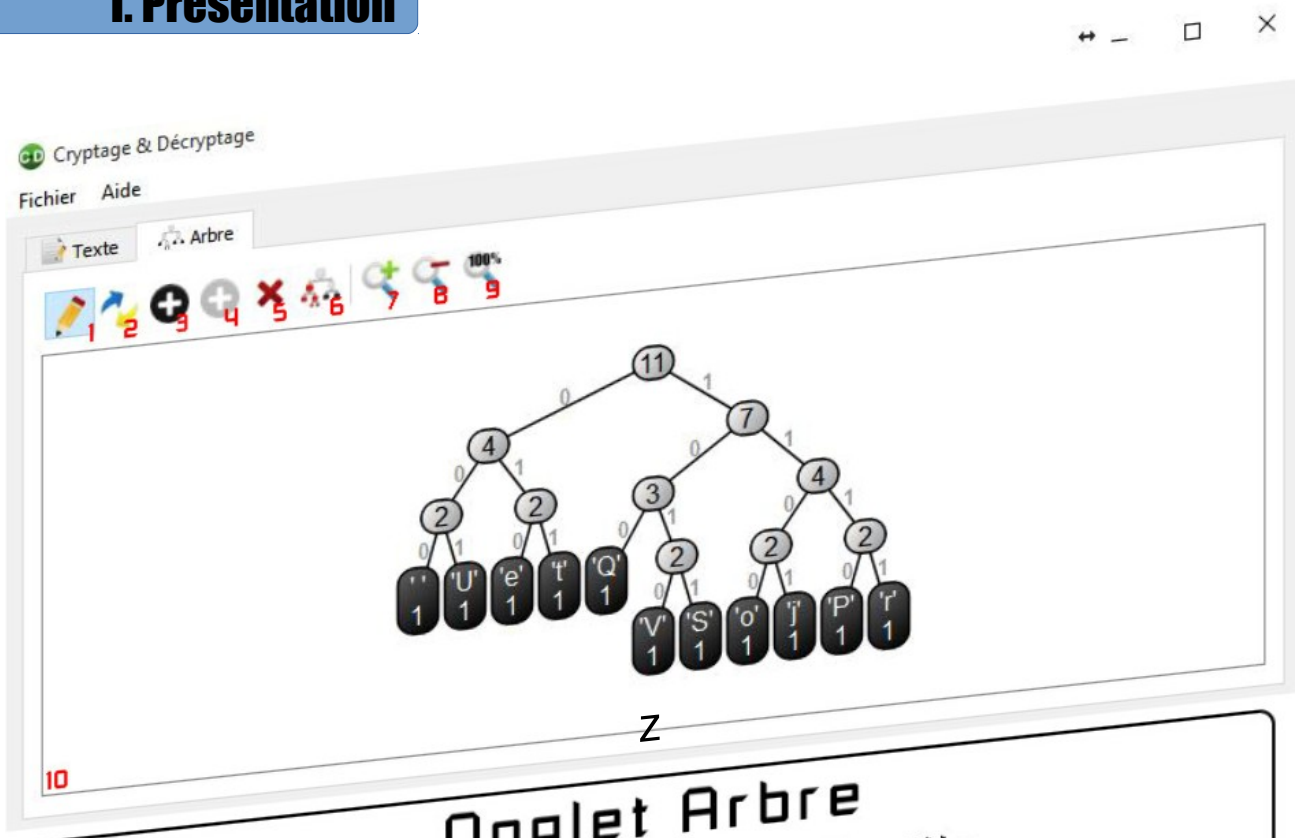
Principe du projet :

Le projet a pour but de coder et décoder des textes écrits dans différents langages.
On utilisera pour le codage de Huffmann pour parvenir à notre objectif.
Egalement, le projet doit être capable de vérifier si un texte a été codé avec un arbre binaire d'Huffman donné, et décrypter un texte codé selon le codage de Huffman.
Une interface graphique sera également nécessaire et utilisée tout au long de ce projet. Elle sera réalisée par nos soins, à l'aide de la bibliothèque QT 5.5.1

Fenêtre Principale



Fenêtre de l'Arbre d'Huffman



Onglet Arbre

1: Modifier les valeurs d'une feuille
2: Echanger des noeuds de position
3: Ajouter des feuilles dans l'arbre
4: Ajouter des noeuds dans l'arbre
5: Supprimer un noeud inutile ou une feuille
6: Supprimer un sous-arbre
7: Zoom
8: Dézoom
9: Retour au zoom d'origine de l'arbre
10: Fenêtre de visualisation de l'arbre actuel

II.Les classes utilisées

Chaque classe possède ses .h et .cpp.
Chaque méthode est commentée et expliquée en détail dans le programme.

ArbreHuff

Il s'agit de la classe gérant le stockage, et l'affichage de l'arbre binaire de Huffman.

Elle permet également, grâce à quelques méthodes qu'on qualifiera d' « algorithmiques » de savoir un arbre est vide, où l'on se trouve sur un arbre en sélectionnant un nœud de celui-ci (une feuille, la racine, ou bien un nœud intermédiaire). On pourra également connaître la profondeur de notre arbre, les ancêtres d'un nœud et modifier le nombre d'occurrences d'un caractère tout en adaptant l'arbre en conséquence.

♦ *Hérite de QGraphicsWidget*

CListe

Il s'agit de notre classe représentant une liste. On peut y ajouter un élément en début ou fin de liste, savoir si la liste est vide et enfin, on peut supprimer des éléments de cette liste.

On a également des méthodes liées à la recherche dans la liste :
On peut chercher un élément et renvoyer son adresse, l'adresse de son noeud ou sa valeur et même chercher la valeur minimale de la liste.

CNoeud

Il s'agit probablement de la classe la plus classique du projet de Cryptage & Décryptage. Celle-ci contient uniquement des attributs, constructeurs et destructeurs caractérisant un nœud d'une CListe. Il sera ensuite très largement utilisé par les autres classes.

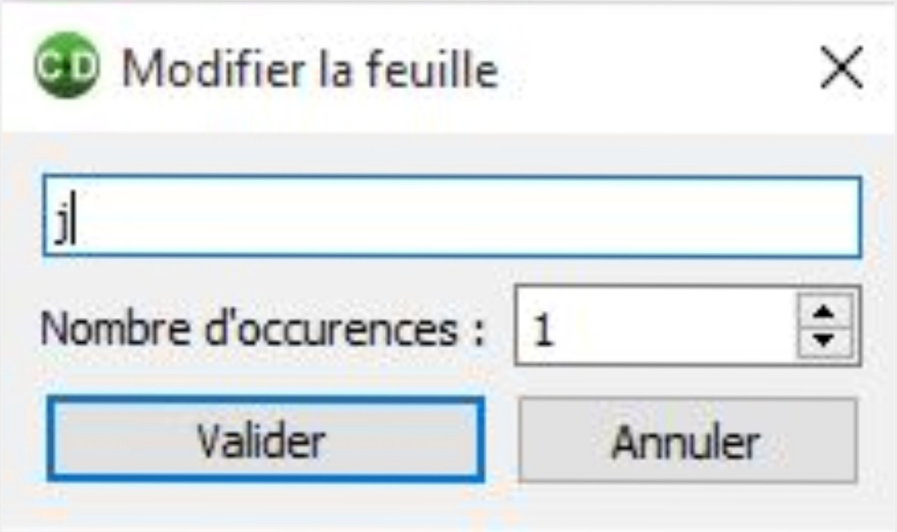
Codage

C'est dans cette classe que nous noterons les codages de chaque caractère en valeurs binaires. La classe est donc composées d'attributs, de constructeurs, destructeur, getters et setters, mais également d'une surcharge des opérateurs == et !=.

DialogModifFeuille

Il s'agit de la petite fenêtre s'ouvrant lors du clic à l'aide de l'outil crayon sur une feuille. C'est cette fenêtre qui permet de modifier les valeurs des feuilles : le caractère qu'elle représente, le nombre d'occurrences du caractère. Elle permet ainsi de modifier l'arbre binaire du second onglet comme bon nous semble.

♦ *Hérite de QDialog*



FenetrePrincipale

C'est cette classe qui contient tous les attributs d'affichage des textes cryptés/décryptés et de l'arbre binaire d'Huffman. C'est la classe qui permettra de voir le programme tel qu'on le voit et qui permet les interactions avec celui ci !

Le détail des méthodes est entièrement commenté dans le programme, instruction par instruction (quand cela n'est pas trivial)

♦ *Hérite de QMainWindow*

Huffman

Contient toute la partie de cryptage des textes.
Ces méthodes permettent de calculer l'occurrence de chaque caractère, de crypter un texte ou bien de le décrypter grâce aux méthodes crypter() et décrypter().
Il générera également l'arbre d'Huffman à partir des occurrences du texte passé en argument grâce à sa méthode genererArbreHuff().

Occurences

Cette classe permet de stocker chaque caractère présent dans un texte, ainsi que son nombre d'occurences au cours du texte passé en argument. Ces informations seront alors ré-utilisées entre autre par la classe Codage afin de les interpréter correctement sous la forme d'un arbre binaire de Huffman.

III.Les soucis rencontrés

On va parler des points délicats du programme sur laquelle il aura fallu revenir à deux fois, et comment le problème a été résolu.