Nom Prénom Etudiant 1 : CHIBANI Mahmoud
Nom Prénom Etudiant 2 : BELHANNACHI Moundji Hocine

Compte rendu partie 3

Contenu du fichier .tar :

Fichier compteRendu.pdf : que vous êtes en train de lire

- Dossier Programme_VTerminal qui contient les fichiers:
  - Sommet.cc Sommt.hh
  - ArbreB.cc ArbreB.hh
  - main.cc Makefile
  - huffman.hh huffman.cc
- Dossier Programme_VInterface qui contient :
  - Le dossier src qui contient les fichiers
    - Sommet.cc Sommt.hh
    - ArbreB.cc ArbreB.hh
    - huffman.hh huffman.cc
    - mainwindow.cc mainwindow.hh mainwindow.ui
    - main.cc Huffman.pro
  - Le fichier Makefile

VERSION Terminal :

Texte vers code :

-le programme prend un texte en anglais via l'entrée standard.
-le programme supporte tous les caractères dont le code ASCII appartient à l'intervalle : (32, 126).
Concrètement c'est toutes lettres de l'alphabet en majuscule et minuscule, les chiffres, les opérateurs binaires, les parenthèses et accolades …

code vers texte :

-le programme prend un code binaire via l'entrée standard.
-le programme vérifie la validité du code au niveau syntaxique (par exemple uniquement des 0 et des 1).
-le programme essaie de décoder le code binaire en utilisant l'arbre issue du dernier cryptage et retourne le texte décodé si le code binaire est complet et reconnue sinon il retourne une erreur.

Contenu :

-Contenu de la deuxième partie avec l'ajout de :
    -la fonction lecture_traitements_DE( ) dans le classe huffman.
    -la fonction start( ) dans la classe huffman.

Processus de cryptage :

En premier lieu le programme récupère le texte à codifier de l'entrée standard puis calcul le pourcentage d'occurrence de chaque caractère en appliquant cette formule : nombre d'occurrences d'un caractère * 100 / nombre de caractères du texte.
Puis le programme stocke le pourcentage de chaque caractère dans un Sommet vide qui sera à son tour insérer dans un arbre vide et tous les arbres crées pourront être accéder via un vecteur de pointeur d'arbres.

Ensuite le programme applique le principe de l'algorithme de huffman en fusionnant deux arbres présents dans le vecteur ayant les deux plus petits pourcentages d'occurrences dans le vecteur arbres, cette opération est répétée jusqu'à avoir un seul arbre dans le vecteur qui représentera le résultat de la fusion de tous les arbres selon l'algorithme de huffman.

Après avoir obtenu l'arbre finale le programme trouve pour chaque caractère son code huffman et ceci en parcourant l'arbre en profondeur tout en notant les 1 et les 0 qui compose le code du caractère à chaque fois que le programme passe par un sommet qui fais partie des sommets qui compose le chemin entre la racine et la feuille qui contient le caractère que le programme recherche.

Remarque : le code des caractères se lit de la feuille jusqu'à la racine (sens : ←) et de la racine jusqu' à la feuille (sens : →).

Finalement le programme parcourt le texte est envoie pour chaque caractère son code sur la sortie standard.

Processus de décryptage :

D'abord le programme récupère le code binaire et vérifie sa validité, ensuite il parcourt le code séquentiellement où il prend un 0 ou un 1 qu'il concatène à chaque itération et interprète le résultat de la concaténation de gauche à droite telle que pour chaque chiffre : 0 = aller vers le fils gauche et 1 = aller vers le fils droit et utilise ceci comme instructions à appliquer tout au long d'un parcours qui commencera de la racine de l'arbre du précédent cryptage et s'arrêtera si une feuille de l'arbre est trouvé et dans ce cas retourne le caractère de la feuille sinon il retourne une erreur.

Dans le cas où un caractère est retourné le code concaténé est effacer et le programme recommence à concaténer à la position suivante.
Tous les caractères trouvés sont concaténés et formeront le texte à retourner à la fin du programme.

Résultat terminale :

Les éléments affichés sur le terminal représentent le résultat du cryptage/décryptage d'un texte/code binaire rentré sur le même terminal.

Les éléments résultats d'un cryptage :

- l'alphabet (les caractères du texte).
- l'arbre résultat de l'application de l'algorithme de huffman où pour chaque sommet le pourcentage est affiché et pour les feuilles de l'arbre vous trouverez les caractères en plus.
- le code de chaque lettre de l'alphabet.
- le code du texte.

Les éléments résultats d'un décryptage :

-Liste des correspondances entre caractères et code binaire généré grâce au dernier cryptage
- texte reconnue par le programme en utilisant l'arbre du dernier cryptage

Comment faire marcher le programme :

On a implémenté un fichier **Makefile** qui gère la compilation et l'exécution de l'application, il suffit de taper la commande 'make'.
Vous pouvez également compiler puis exécuter séparément en tapant 'make compil' puis 'make run'.
Enfin pour supprimer tous les fichiers .o et l'exécutable tapez la commande 'make clean'.

VERSION INTERFACE :

Pour la version interface on crée une interface graphique en utilisons l'interface de programmation d'application Qt**.**
Il est nécessaire d'avoir téléchargé une version >= 4 de la Bibliothèque Qt sur votre machine pour faire marcher le programme.

On a utilisé les mêmes classes de la partie Terminal , et on a ajouter la classe : **MainWindow** pour gérer l'interface.

Notre Interface permet de lire un texte saisie et fait le même travail que la version terminal.

les objets principaux de notre interface sont:
- QLineEdit : où on peut saisir le texte qu'on veut crypter/décrypter.
- QPushButton : « coder » pour lancer notre cryptage
- QPushButton : « decoder » pour lancer notre decryptage
- QTextEdit : pour afficher les résultats ou les messages d'erreur.
- QTableWidget : tableau pour afficher le code huffman pour chaque caractère.
- QTextEdit : pour afficher l'arbre Binaire généré par le codage.

Comment faire marcher le programme :
- Positionnez-vous dans le dossier **Programme_VInterface**
tapez **make** pour compiler puis éxecuter.
- Pour **compiler et exécuter** séparément:
tapez **make compil.**
tapez **make run.**
- Pour supprimer tous les fichier générés par la compilation
Tapez **make clean.**