TP 1
CORRECTEUR DE TEXTE

Introduction

Pour le premier devoir, vous allez construire un correcteur de texte. Les logiciels d'OCR (Optical Character Recognition : reconnaissance optique de caractère) sont utilisés pour traduire une image d'un texte en un fichier de texte. Cette reconnaissance des caractères n'est pas toujours correcte. Votre programme va recevoir un texte contenant des fautes dû à la reconnaissance de texte incorrecte et les corriger. Cette correction d'erreur sera faite en comparant les mots du texte avec les mots d'un dictionnaire. Votre logiciel va calculer une métrique de « distance » (nombre d'erreurs) entre un mot du texte et les mots du dictionnaire. Le mot ayant le moins d'erreurs (plus courte distance) sera choisi comme mot corrigé.

Description des entrées

Votre logiciel va recevoir, sur la ligne de commande, le nom d'un fichier contenant le texte à corriger. Ce sera donc un fichier ayant l'extension « txt ». Aussi, l'utilisateur peut activer l'utilisation d'une métrique secondaire en cas d'égalité. Le sémaphore « -s » est utilisé pour activer la métrique secondaire. Le logiciel va utiliser un dictionnaire de mot (anglais) pour les comparaisons. Ce dictionnaire sera dans un fichier contenant des mots en ordre alphanumérique à raison d'un par ligne. Votre logiciel doit ouvrir ce fichier et le charger en mémoire. Vous pouvez télécharger le fichier words.txt à l'adresse suivante : https://github.com/dwyl/english-words. Votre logiciel doit lire ce fichier sur le disque dur, dans le répertoire courant.

Lecture d'un fichier

```
#include <fstream>

using namespace std;

int main() {
    ifstream fichier( nom );
    char c;

    do {
        fichier.get( c );
        cout << c;
    } while( ! fichier.eof() );
    return 0;
}
```

Traitement

Lorsque démarré, votre logiciel va lire les mots du fichier, les corriger si nécessaire et afficher les mots résultants. Un mot est une suite continue de caractères : lettre minuscule « a..z », lettre majuscule « A..Z » et apostrophe « ‘ ». Tout autre caractère est considéré comme un séparateur de mot et sera simplement recopié à la sortie.

Une fois qu'un mot m a été lu, vous devez le transformer en minuscule. Ensuite, chercher le dictionnaire pour vérifier s'il existe. Utilisez une recherche binaire pour accélérer le processus. Si le mot est présent tel quel dans le dictionnaire, alors il est affiché à l'écran, dans sa typographie originale. Si le mot n'est pas présent, alors vous devez trouver le mot t_i du dictionnaire D qui est le plus similaire. La similarité entre deux mots est représentée par un petit nombre d'erreurs (nombre de différences entre les deux mots). Soit deux mots m et t_i de taille (nombre de caractères) $|m|$ et $|t_i|$, le nombre de différences est calculé par l'appel à la fonction $L_{(m,t_i)}(|m|, |t_i|)$. Voici la description récursive de cette fonction :

$$L_{(m,t_i)}(0, j) = j$$

$$L_{(m,t_i)}(i, 0) = i$$

$$L_{(a,b)}(i,j) = \min \begin{cases} L_{(a,b)}(i, j-1) + 1 \\ L_{(a,b)}(i-1, j) + 1 \\ L_{(a,b)}(i-1, j-1) + f(a[i] \neq b[j]) \end{cases}$$

Où f est une fonction indicatrice. Une fonction indicatrice est une fonction qui, dans notre contexte, accepte un booléen et le transforme en une valeur numérique :

$$f(true) = 1$$
$$f(false) = 0$$

Lorsque votre logiciel a comparé tous les mots du dictionnaire au mot à corriger, il suffit de prendre la liste des mots qui ont obtenu un nombre d'erreur minimum. Le mot de la liste qui est le premier dans l'ordre alphabétique sera le mot choisi. Dans le cas où l'utilisateur a choisi d'utiliser une fonction secondaire, votre logiciel ne choisira pas le premier mot de la liste. Dans ce cas, vous devez construire une fonction qui prend les mots de la liste et fait un choix « acceptable » pour corriger l'erreur. C'est à vous d'inventer la fonction secondaire. Cette fonction doit avoir un commentaire justifiant la méthode de sélection utilisée.

Lorsque le « bon » mot a été trouvé, il est affiché à l'écran. Si le mot original était en majuscule, il doit rester en majuscule. Si le mot original commençait par une lettre majuscule, le mot corrigé doit commencer par une majuscule.

Production des sorties

Votre logiciel doit réécrire le contenu du fichier d'entrées à l'écran. Les mots, corrigés ou non, sont réécrits à l'écran et tout autre caractère est simplement réécrit à l'écran tel quel.

Optimisation

Les optimisations suivantes doivent être présentes dans le code :

1. Fouille binaire pour vérifier si le mot est présent dans le dictionnaire.
2. Lorsque le mot n'est pas présent, commencez la recherche avec les mots débutant avec la même lettre que le mot cherché.
3. Optimisez la fonction L de comparaison des mots afin qu'elle se termine si elle dépasse le minimum courant. Puisque nous cherchons les mots ayant un minimum d'erreur, si un mot ayant 2 erreurs est déjà trouvé, il est inutile de continuer le calcul lorsque la comparaison entre deux mots est plus grande que 2.

L'optimisation suivante peut être ajoutée, elle peut aussi remplacer l'optimisation 3 :

1. le calcul de la fonction de comparaison L peut être accéléré grandement. Ce calcul récursif répété le calcule de plusieurs valeurs. Il est possible d'éviter cette répétition en conservant les valeurs déjà calculées dans une table.

Directive

1. Le tp est à faire seul ou en équipe de deux (maximum).
2. Commentaire :
    a. Commentez l'entête de chaque fonction. Ces commentaires doivent contenir la description de la fonction et le rôle de ces paramètres.
    b. Une ligne contient soit un commentaire, soit du code, pas les deux.
    c. Utilisez des noms d'identificateur significatif.
    d. Utilisez le français.
3. Code :
    a. Pas de `goto`, `continue`.
    b. Les `break` ne peuvent apparaître que dans les `switch`.
    c. Un seul `return` par méthode.
4. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.

Remise

Remettre le tp par l'entremise de Moodle. Remettez votre fichier `'*.cpp'`, nous utiliserons un seul fichier de code pour le premier tp. Le tp est à remettre avant le 9 juin 23:59.

Évaluation

- Fonctionnalité (9 pts) : Votre tp doit compiler sans erreur (il peut y avoir des warnings). J'utilise la ligne de compilation suivante : `g++ nomTp.cpp -lm -std=c++11`

- Structure (2 pt) : Il faut avoir **plusieurs** fonctions. Construisez un code bien structuré.
- Lisibilité (4 pts) : commentaire, indentation et noms d'identificateur significatif.