esiea

Projet 3 : Analyseur de Fichier Texte

(Niveau intermédiaire)

3A FISE et FISA 2024-2025

Version	Date	Créé/modifié par	Commentaires
1.0	22/09/2024	Radhia GADDOURI	

Table des matières

esiea

1. Objectif du Projet

L'objectif de ce projet est de développer un programme en langage C qui analyse un fichier texte fourni par l'utilisateur et fournit diverses statistiques, telles que le nombre total de mots, de lignes, de caractères, ainsi que la fréquence de chaque mot. Le programme doit également être capable d'afficher les mots les plus fréquents et de sauvegarder les résultats dans un fichier de sortie.

2. Objectifs Pédagogiques

- **Manipulation de Fichiers et Analyse de Texte :**
    - Approfondir les compétences en lecture et écriture de fichiers en C, ainsi qu'en analyse de texte.
- **Structures de Données et Gestion des Chaînes :**
    - Comprendre l'utilisation des structures de données (tableaux, listes chaînées) et des chaînes de caractères en C.
- **Algorithmes de Recherche et de Comptage :**
    - Apprendre à implémenter des algorithmes efficaces de recherche et de comptage.
- **Gestion Dynamique de la Mémoire :**
    - Développer des compétences en gestion de la mémoire dynamique, y compris l'allocation et la libération de mémoire.

3. Fonctionnalités Requises

1. **Lecture du Fichier Texte :**
    - **Description :** Le programme doit lire un fichier texte dont le chemin est fourni par l'utilisateur.
    - **Consignes :**
        - Implémentez une fonction FILE* ouvrirFichierLecture(const char* chemin) qui ouvre le fichier en mode lecture.

- Vérifiez que le fichier a été ouvert avec succès. En cas d'erreur (fichier introuvable ou problème d'accès), affichez un message d'erreur approprié et terminez le programme.

2. **Compter le Nombre de Mots, Lignes et Caractères :**
   - **Description :** Le programme doit compter et afficher le nombre total de mots, lignes, et caractères présents dans le fichier.
   - **Consignes :**
     - Implémentez des fonctions distinctes pour chaque type de comptage :
       - int compterLignes(FILE* fichier) pour compter le nombre de lignes.
       - int compterMots(FILE* fichier) pour compter le nombre de mots.
       - int compterCaracteres(FILE* fichier) pour compter le nombre de caractères.
     - Assurez-vous que le programme traite correctement les espaces, tabulations, et sauts de ligne pour le comptage des mots et lignes.
3. **Calcul de la Fréquence de Chaque Mot :**
   - **Description :** Le programme doit calculer la fréquence d'apparition de chaque mot dans le fichier.
   - **Consignes :**
     - Utilisez une structure de données telle qu'un tableau de structures ou une liste chaînée pour stocker les mots et leur fréquence.
     - Implémentez une structure struct Mot avec des champs pour le mot (chaîne de caractères) et sa fréquence (entier).
     - Développez une fonction void ajouterMotOuIncrementer(char* mot, struct Mot* tableauMots, int* nombreMots) qui ajoute un nouveau mot ou incrémente la fréquence d'un mot existant.

4. **Affichage des Mots les Plus Fréquents :**
    - **Description :** Le programme doit afficher les mots les plus fréquents dans le fichier, dans l'ordre décroissant de fréquence.
    - **Consignes :**
        - Implémentez une fonction void trierMotsParFrequence(struct Mot* tableauMots, int nombreMots) pour trier les mots par fréquence décroissante.
        - Affichez les mots les plus fréquent. Vous pouvez limiter l'affichage aux 10 mots les plus fréquents, par exemple.
5. **Sauvegarde des Résultats dans un Fichier de Sortie :**
    - **Description :** Les résultats de l'analyse (nombre de mots, lignes, caractères, et fréquences de mots) doivent être sauvegardés dans un fichier de sortie.
    - **Consignes :**
        - Demandez à l'utilisateur de fournir le chemin du fichier de sortie.
        - Implémentez une fonction void sauvegarderResultats(char* cheminSortie, int nombreLignes, int nombreMots, int nombreCaracteres, struct Mot* tableauMots, int nombreMotsDistincts) qui écrit les résultats dans le fichier de sortie.
        - Vérifiez que le fichier de sortie a été créé et ouvert avec succès. En cas d'erreur, affichez un message d'erreur approprié.

4. Fonctionnalités Bonus suggérées

1. **Support Multilingue :**

- **Description :** Ajouter la prise en charge de plusieurs langues pour analyser des fichiers écrits dans différentes langues. Cette fonctionnalité pourrait inclure la gestion des caractères spéciaux (comme les accents) et des alphabets non latins.

- **Consignes :**
    - Implémentez une option pour l'utilisateur de spécifier la langue du texte.
    - Ajustez le programme pour reconnaître les caractères spéciaux dans différentes langues (par exemple, lettres accentuées en français).
    - Gérez correctement la casse (majuscule/minuscule) en tenant compte des particularités de chaque langue.

2. **Détection de Phrases et Analyse Sémantique de Base :**
    - **Description :** En plus de l'analyse des mots, le programme pourrait analyser les phrases pour détecter les plus longues, les plus courtes, et la longueur moyenne des phrases.
    - **Consignes :**
        - Implémentez une fonction pour détecter les fins de phrase (basée sur des délimiteurs comme ".", "!", "?"...).
        - Ajoutez une fonction pour calculer la longueur moyenne des phrases.
        - Identifiez la phrase la plus longue et la plus courte du fichier.
3. **Analyse Comparative entre Deux Fichiers :**
    - **Description :** Le programme pourrait comparer deux fichiers texte pour identifier les différences en termes de mots les plus fréquents ou de style (longueur des mots, des phrases, etc.).
    - **Consignes :**
        - Implémentez une option pour que l'utilisateur spécifie deux fichiers texte.
        - Analysez les deux fichiers et identifiez les mots communs ou distincts.
        - Affichez un rapport comparatif des deux fichiers, incluant les différences dans la fréquence des mots et la longueur des phrases.
4. **Mots Palindromes :**
    - **Description :** Ajouter une fonctionnalité pour détecter et afficher les mots palindromes (mots qui se lisent de la même manière dans les deux sens, comme "radar").

- **Consignes :**
    - Implémentez une fonction int estPalindrome(char* mot) qui vérifie si un mot est un palindrome.
    - Affichez la liste des palindromes détectés dans le fichier.

5. **Interface Graphique (Bonus avancé) :**

- **Description :** Intégrer une interface graphique à l'analyseur de texte à l'aide de la bibliothèque **GTK ou ncurses** pour afficher les résultats dans une interface plus conviviale.
- **Consignes :**
    - Implémentez une fenêtre graphique où l'utilisateur peut sélectionner le fichier à analyser, voir les résultats en direct, et sauvegarder les résultats dans un fichier de sortie.
    - Intégrez des boutons pour lancer l'analyse et visualiser les résultats de manière interactive.

5. Contraintes Techniques

- **Structures de Données :**
    - Utiliser des structures de données appropriées (tableaux dynamiques ou listes chaînées) pour stocker les mots et leurs fréquences.
    - Gérer dynamiquement la mémoire pour le stockage des mots et de leurs fréquences, en utilisant les fonctions malloc() et free().
- **Manipulation de Fichiers :**
    - Utiliser les fonctions de bibliothèque standard de C pour la manipulation des fichiers (fopen(), fclose(), fscanf(), fprintf(), etc.).
    - Assurer une gestion robuste des erreurs lors de l'ouverture, de la lecture, et de l'écriture des fichiers.
- **Manipulation de Chaînes :**
    - Utiliser des fonctions de manipulation de chaînes (strcpy(), strcmp(), strtok(), etc.) pour extraire et traiter les mots du fichier texte.

- Gérer correctement les différents caractères spéciaux, ponctuation, et casse (majuscule/minuscule) pour un comptage précis des mots.

6. Plan de Réalisation

1. **Initialisation du Projet :**
   - Créer les fichiers sources (main.c, analyseur.h, analyseur.c).
   - Définir les structures de données nécessaires (tableau de mots, structure Mot, etc.).
2. **Implémentation de la Lecture et de l'Analyse du Fichier :**
   - Implémenter les fonctions pour lire le fichier et compter les lignes, mots, et caractères.
   - Implémenter les fonctions pour extraire les mots et calculer leurs fréquences.
3. **Affichage des Résultats et Sauvegarde :**
   - Implémenter les fonctions pour trier les mots par fréquence et afficher les résultats.
   - Implémenter la fonction pour sauvegarder les résultats dans un fichier de sortie.
4. **Tests et Débogage :**
   - Tester chaque fonctionnalité individuellement pour s'assurer de son bon fonctionnement.
   - Utiliser plusieurs fichiers texte de test avec différentes structures (longueur, complexité) pour valider les résultats.
5. **Optimisation et Documentation :**
   - Optimiser le code pour des performances maximales (gestion efficace de la mémoire).
   - Ajouter des commentaires détaillés dans le code source et rédiger une documentation utilisateur.
6. **Développement des Fonctionnalités Bonus :**

7. Critères d'Évaluation

- **Fonctionnalité complète :** Le programme doit être entièrement fonctionnel avec toutes les fonctionnalités de base implémentées.
- **Gestion des erreurs et robustesse :** Le programme doit gérer correctement toutes les erreurs potentielles (ouverture de fichier, lecture, écriture).
- **Qualité du code :** Le code doit être bien structuré, clair, et documenté avec des commentaires appropriés.
- **Performance :** Le programme doit être performant, avec une gestion efficace de la mémoire.
- **Originalité et Créativité :** Des points bonus peuvent être accordés pour l'implémentation de fonctionnalités supplémentaires ou des améliorations créatives.

8. Livrables

- Code source complet (main.c, analyseur.h, analyseur.c).
- Fichiers de résultats de sortie générés par le programme.
- Rapport de projet détaillant l'implémentation, les défis rencontrés, les solutions apportées et la gestion de projet.
- (Optionnel) Démonstration vidéo montrant le fonctionnement du programme d'analyse.

9. Remarques importantes

- Composition des groupes projet et travail collaboratif : Vous pouvez travailler en **binôme ou en groupe de 3 personnes maximum**, **à condition que tous les membres de l'équipe appartiennent à la même classe TD (TD34, TD35, TD36, etc.)**. Assurez-vous bien de bien répartir les tâches et à collaborer de manière efficace.
- **Dépôt Livrable** : Les livrables doivent être déposés exclusivement sur la plateforme **GitLab**. Aucun dépôt ne sera accepté par mail, Teams ou tout autre moyen de communication. Voici le lien, accessible à tous via votre compte 365 : Gitlab.esiea.fr

- **Date limite pour le dépôt des livrables :** Jeudi 19 décembre 2024 à 23h59.
- **Date des soutenances du projet C :** lundi 6 et mardi 7 janvier 2025.