Initiation à la Programmation C - Projet

Nombre d'occurrences des mots d'un texte
Comparaison d'algorithmes

L2-MI - Semestre 3 - 2025

1 Introduction

L'objectif de ce projet est de réaliser une application qui prend en paramètre un fichier texte et répertorie les mots les plus courants présents dans ce texte ainsi que leur nombre d'occurrences. On défini un mot comme étant une suite de lettres majuscules ou minuscules, éventuellement séparées par un tiret. Ces mots devront être affichés à l'utilisateur classés par nombre d'occurrences décroissant.

Plus spécifiquement, il vous est demandé de mettre en œuvre différents algorithmes capables de réaliser cet objectif et de les comparer en termes de vitesse et d'utilisation de la mémoire.

L'utilisateur doit pouvoir choisir le fichier à analyser, le nombre de mots à afficher ainsi que l'algorithme à utiliser.

2 Planning et organisation

Ce projet doit être réalisé en **binôme**. Les membres du binôme peuvent être issus de deux groupes de TP différents. Les monômes et trinômes ne sont pas autorisés. Il vous faut **renseigner votre binôme sur e-learning** le plus tôt possible et **avant le 10 décembre 2025**.
En cas de difficulté à vous associer à un-e camarade, utilisez le canal `adopt-un-binome-projet` sur discord. En dernier recours, contactez un des enseignants de l'UE.

Le **rendu** de votre production se fera via **E-learning** et avant le 6 janvier 2026 à 23h59. Tout retard, même minime, sera sanctionné par une baisse de la note.

L'évaluation finale s'appuiera sur votre rendu ainsi que sur une soutenance.
Durant cette soutenance, vous présenterez vos programmes, vos choix, votre organisation **mais surtout les résultats de votre étude**. Celle-ci s'appuiera sur l'exploitation de vos programmes
La soutenance, d'une durée de 15mn par binôme, s'articulera en deux temps :

- Présentation en groupe de votre démarche et de vos résultats sous forme d'une analyse comparative structurée.
- Dialogue individuel avec chacun des deux membres sur la méthodologie de travail ou l'implantation des algorithmes.

À l'issue de cette soutenance, chaque membre du binôme recevra une note individuelle.

Les soutenances seront organisées durant la semaine d'examens du mois de janvier, soit entre le 12 et 16 janvier 2026. Une **inscription à la soutenance**, via e-learning, devra obligatoirement être **réalisée entre le 15 et le 19 décembre**. Une absence à la soutenance entraîne la note 0 au projet.

3 Production attendue

Votre travail doit contenir au minimum les fichiers suivants :

- Le rapport au format `.pdf` ;
- Un fichier `ReadMe.txt` expliquant comment lancer et compiler correctement vos programmes ;
- Pour le programme C, les différents fichiers sources nécessaires à la compilation ;
- Le(s) script(s) python utilisés pour la visualisation.

→ Le rapport, les fichiers sources, le `ReadMe.txt` et les scripts python seront compressés sous forme d'une archive au format `.zip`. Si nécessaire, il conviendra d'organiser le contenu de votre archive à l'aide d'une arborescence de répertoires pour en faciliter la manipulation et la compréhension.

→ L'utilisation d'un "makefile" est possible mais pas impérative.

3.1 Fonctionnalités imposées du programme

Votre programme doit être capable de :

- Prendre en argument un fichier texte ;
- Afficher le résultat si demandé ;
- Écrire le résultat dans un fichier texte si demandé.
  Le format du fichier contenant le résultat est le suivant : une ligne par mot différent dans le texte, chaque ligne doit contenir d'abord le mot, puis son nombre d'occurrences. De plus, les mots doivent être dans l'ordre décroissant d'occurrences.
- Permettre le choix du nombre de mots à afficher ;
- Permettre le choix entre au moins trois algorithmes ;
- Mesurer et afficher les performances : espace mémoire utilisé et temps d'exécution ;
- Écrire les performances dans un fichier. Le formatage des données est libre.

De plus, au moins une amélioration est attendue (c.f. partie 4).

3.2 Recommandations et contraintes de programmation imposées

Vous pourrez utiliser **Gallica**, la bibliothèque numérique de la BnF, pour récupérer des fichiers texte d'ouvrages libres de droits : `https://gallica.bnf.fr/`.

Afin de pouvoir mesurer l'espace mémoire utilisé lors de l'exécution de votre programme, il vous est demandé d'utiliser la structure suivante :

```
typedef struct {
   size_t cumul_alloc;      // champ obligatoire : cumul de l'espace mémoire alloué
   size_t cumul_desalloc;   // champ obligatoire : cumul de l'espace mémoire désalloué
   ...                   // d'autres champs qui sembleraient utiles
} InfoMem;
```

Ainsi que les fonctions suivantes :

- `void* myMalloc(size_t size, InfoMem* infoMem);`
- `void* myRealloc(void* ptr, size_t new_size, InfoMem* infoMem, size_t old_size);`
- `void myFree(void* ptr, InfoMem* infoMem, size_t old_size);`

Ces fonctions seront utilisées à la place de `malloc`, `realloc` et `free` afin de calculer l'espace mémoire utilisé tout au long du programme. L'idée est de passer en paramètre l'adresse d'une structure InfoMem qui conservera, à minima et à tout moment de l'exécution, le nombre total d'octets alloués et désalloués par le programme.

La structure et les fonctions associées pourront être écrites dans un fichier `gererMem.c` mais ce n'est pas impératif.

Afin de vous permettre de valider cette partie de votre code, une activité de **test** sera mise en place sur la plateforme **Platon**.

Le recours à une IA générative est fortement déconseillée dans le cadre d'une utilisation non éclairée. Ce projet a pour vocation première de vous permettre de développer vos compétences et savoir-faire. Dans tous les cas, le code produit et les algorithmes devront être parfaitement maîtrisés lors de la soutenance. Dans le cas contraire, il y aura plagiat reconnu, quel qu'en soit la source.

3.3 Contenu du rapport

Un partie non négligeable de l'évaluation se basera sur le rapport. Celui-ci doit contenir, à minima, les points suivants :

- L'architecture du contenu de votre fichier `.zip` avec l'utilité de chaque dossier ou fichier contenu.
- Une explication précise des algorithmes implémentés et de leur fonctionnement, notamment les structures de données utilisées.
  Vous pourrez faire figurer dans votre rapport les structures et prototypes des fonctions principales mais sans en détailler le code.
- Une étude comparative des performances des algorithmes en termes d'espace mémoire et de vitesse d'exécution.
- Votre organisation en termes de recherche et de conception d'algorithmes.
- Votre organisation en tant que groupe, notamment :
  - → La contribution approximative, en %, de chaque membre du binôme au travail collectif.
  - → le-s algorithme-s implémenté-s par chaque membre. **L'évaluateur s'appuiera sur cet item pour diriger le dialogue individuel** lors de la soutenance. Chaque membre doit donc avoir implémenté au moins un des algorithmes de recherche demandés et en maîtriser tous les aspects.

Il n'est pas nécessaire que les algorithmes implémentés possèdent une structure complètement différente. Par exemple, il vous est possible de comparer le même algorithme mais utilisant des structures de données différentes (structure triée vs structure non triée).

Une attention particulière sera portée sur les visualisations présentes dans votre rapport pour chaque algorithme ainsi que sur les commentaires et discussions autour des résultats mettant notamment en parallèle complexité théorique et complexité empirique (en mémoire et en temps).
Vous devrez présenter, par exemple, des courbes de l'évolution des performances de vos algorithmes en fonction du nombre de mots différents du texte, du nombre de mots total du texte et autres facteurs pouvant impacter l'efficacité.

Pour ce faire, votre programme doit être capable de lancer un algorithme particulier sur une liste de différents fichiers textes de tailles variées (ces noms pourront être passé en paramètre lors du lancement du programme, ou via un autre fichier texte) et d'y inscrire les performances dans un fichier (avec un format adapté).

Il vous est également demandé de réaliser un script Python lisant les fichiers de performances produits par votre programme et générant des courbes en utilisant la bibliothèque matplotlib.

4 Améliorations

Les fonctionnalités décrites précédemment sont obligatoires. Au moins une amélioration doit être implémentée dans votre programme afin de valider votre projet. Les améliorations supplémentaires permettront d'envisager une note supérieure en fonction de leur complexité.

Les améliorations suivantes vous sont proposées. Leur complexité est indiquée par un nombre d'étoiles ($\star$).

- Permettre de donner des mots interdits qui ne seront pas pris en compte par le programme ($\star$).
- Permettre de donner une liste de mots pour ne chercher que ceux-ci dans le fichier ($\star$).
- Une option permettant à l'utilisateur de ne demander que les mots d'au moins k lettres (ou k est spécifié par l'utilisateur) ($\star$).
- Pouvoir générer le graphique des mots les plus courants ($\star$).
- Implémenter quatre algorithmes ou plus ($\star\star$).
- Une option permettant d'étudier la fréquence d'apparition de certains mots sur des portions glissantes d'un fichier texte volumineux.
  Par exemple, si "`joie`" est un mot fréquent dans un fichier, on s'intéresse à sa fréquence d'apparition sur les n premiers mots, puis les n suivants, etc. On peut ainsi obtenir une visualisation de la densité d'apparition de ce mot sur un gros fichier texte ($\star\star$).
- Sur le principe précédent, on peut aussi choisir de faire glisser la fenêtre de taille n d'un delta δ de quelques mots et observer la fréquence d'apparition sur cette fenêtre glissante ($\star\star\star$).

Toute amélioration qui n'est pas présente dans la liste sera bienvenue dés lors qu'elle sera pertinente. N'hésitez pas à interroger votre chargé de TP à ce sujet.

Amusez-vous et ne procrastinez pas car il y a du travail... Bonne chance à chacun-e !