1 Fonctions utilitaires

1.1 afficher_ligne

— Signature : `void afficher_ligne(int num_ligne)`
— Description :
Affiche une seule ligne. La ligne est affichée en trois parties le numéro, le séparateur et le contenu de la ligne.

Le numéro : le numéoro de la ligne aligné à droite. La largeur du numéro est toujours assez grande pour afficher le dernier numéro du tampon en conservant alignés les séparateurs.

Le séparateur : « : » ou « :* » s'il s'agit de la ligne courante.

Le contenu : Toute la chaîne de caractère qui représente une ligne, suivie d'un retour de chariot.

Exemple :

```
0 : contenu de la ligne

ou

0 :*contenu de la ligne

s'il s'agit de la ligne courante.
```

— Paramètres :

1. num_ligne : Le numéro de la ligne dans le tampon à afficher.

— Retour : void.

1.2 afficher

— Signature : `void afficher()`
— Description : Affiche toutes les lignes du tampon grâce à la fonction afficher_lignes.
— Paramètres : aucun
— Retour : void.

1.3 lire_fichier

— Signature : `int lire_fichier(char** destination, const char* nom_fichier)`
— Description : Lit le contenu d'un fichier donné en paramètre et en place chacune des lignes dans le tableau de chaînes destination. Si le fichier n'existe pas ou est illisible, un tampon vide est créé.
— Paramètres :

1. destination. Tableau de chaînes de caractères prêt à recevoir le contenu du fichier.
2. nom_fichier. Chaîne de caractère contenant le chemin d'accès absolu ou relatif au répertoire de travail du programme.

1.4 ecrire_fichier

— Signature : `bool ecrire_fichier(const char* nom_fichier, char ** const lignes, int nb_lignes)`
— Description : Écrit toutes les lignes de *lignes* dans un fichire nommé *nom_fichier*. Si le fichier n'existe pas il est créé, s'il existe il est écrasé.
— Paramètres :

1. nom_fichier : Chaîne de caractère contenant le chemin d'accès absolu ou relatif au répertoire de travail du programme.
2. lignes : Tableau de chaînes de caractères à écrire dans le fichier.
3. nb_lignes : Le nombre total d'éléments dans *lignes*

1.5 est_numerique

— Signature : `bool est_numerique(const char* chaine)`
— Description : Détermine si une chaîne de caractère ne contient que des caractères numériques.
— Paramètres :

1. chaine : Chaîne de caractère à analyser.

— Retour : Vrai si et seulement si tous les caractères de *chaine* sont l'un des caractères «0123456789».

1.6 compter_mots

— Signature : `int compter_mots(const char* source)`
— Description : Compte le nombre de mots dans une chaîne de caractères. *Un mot est délimité par le début de la chaîne, une ou plusieurs espaces ou la fin de la chaîne.*
— Paramètres :

1. source : La chaîne de caractère dont compter les mots.

— Retour : Le nombre de mots contenus dans la chaîne (>=0).

1.7 changer_casse

— Signature : `char *changer_casse(char* dest, const char* source, int casse)`
— Description : Modifie la casse de lettres d'une chaîne de caractères selon le paramètre *casse*. Si casse vaut :
  — CASSE : :MAJ : Toutes les lettres sont transformées en majuscules.
  — CASSE : :MIN : Toutes les lettres sont transformées en minuscules.
  — CASSE : :CAP : Toutes les lettres sont transformées en minuscules sauf la toute première qui est transformée en majuscule.
— Paramètres :

1. dest : Paramètre de sortie. Pointeur vers la chaîne de caractères modifiée.
2. source : La chaîne de caractères originale.
3. casse : L'une des constantes de CASSE.

— Retour : Un pointeur vers la chaîne modifiée.

1.8 separer_mots

— Signature : `int separer_mots(char** mots, const char* source)`
— Description : Sépare une chaîne ses composantes. Copie chaque mot de *source*, dans *mots* puis retourne le nombre de mots trouvés. La définition d'un mot est la même que celle utilsée par *compter_ mots*.
— Paramètres :

1. mots : Tableau de chaînes de caractères devant contenir la liste des mots trouvés. Le tableau doit compter suffisamment d'espace pour tous les mots de la source.

2. source : La chaîne de caractère à analyser.

— Retour : Le nombre de mots trouvés et copiés dans *mots*.

1.9 chercher

— Signature : `int chercher(const char* source, const char* cible)`
— Description : Recherche une sous-chaîne dans une chaîne.
— Paramètres :

1. source : La chaîne de caractère dans laquelle chercher la cible.
2. cible : La sous-chaîne recherchée.

— Retour : L'indice du premier caractère de la sous-chaîne dans *source* s'il existe, -1 sinon.

1.10 remplacer

— Signature : `char *remplacer(char* dest, const char* source, const char* cible, const char* remplacement)`
— Description : Recherche une sous-chaîne et, si elle existe, la remplace par une autre. La chaîne telle que modifiée est placée dans le paramètre de sortie *dest*. L'originale est inchangée. Si *cible* ne se trouve pas dans *source*, *dest* est une copie inchangée de *cible*.
— Paramètres :

1. dest : Paramètre de sortie. Pointeur vers la chaîne modifiée.
2. source : La chaîne originale.
3. cible : La sous-chaîne recherchée
4. remplacement : Le remplacement si la sous-chaîne est trouvée.

— Retour : Un pointeur vers la chaîne destination.