****

**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**

**MINISTERE DE L’ENSEIGNEMENT ET DE LA RECHERCHE SCIENTIFIQUE**

**ECOLE NATIONALE SUPERIEURE D’INFORMATIQUE.**

**CPI 1 ANNEE SCOLAIRE : 2016/2017**

**TP EN ALGORITHMIQUE ET STRUCTURE DE DONNEES DYNAMIQUES**

**REALISE PAR :**

**• MAHLAL Mehenna**

**• IDRES Aghiles**

**SECTION : A**

**GROUPE : 01**

**Avec : Mme S. KACET**

**SEMESTRE : II**

Table des matières

[1.Découpage modulaire et la structure utilisée : 2](#_Toc480147127)

[A) Structure utilisée : 2](#_Toc480147128)

[B) Découpage modulaire : 2](#_Toc480147129)

# 1.Découpage modulaire et les structures utilisées :

## A) Structures utilisées :

typedef struct \_str\_mot{

char \* mot; p\_mot\_next est le pointeur sur le mot suivant dans la liste

struct \_str\_mot \*p\_mot\_next; des mots, et p\_mot\_prev est le pointeur sur le mot précédent

struct \_str\_mot \*p\_mot\_prev; p\_lg\_mot\_next pointe la prochaine longueur sur la liste des

} str\_mot; longueurs et p\_lg\_mot\_prev le précédent, p\_mot\_first pointe

typedef struct \_str\_lg\_mot{ le tout premier mot de la liste des mots d’une longueur L et

short longueur; p\_mot\_last pointe le dernier mot d’une longueur L de la liste

struct \_str\_lg\_mot \*p\_lg\_mot\_next; des mots .

struct \_str\_lg\_mot \*p\_lg\_mot\_prev; la structure str\_intervalle contient 3 champs : un champ pour

str\_mot \* p\_mot\_first; un tableau de deux cases où la première va contenir la prem-

str\_mot \* p\_mot\_last; ière lettre de l’intervalle et la deuxième pour la dernière.

} str\_lg\_mot; P\_lg\_mots\_first pointe le premier maillon de la liste des

typedef struct \_str\_intervalle{ longueurs et p\_lg\_mots\_last pointe le dernier maillon de la

int intervalle[2]; liste des longueurs.

struct \_str\_lg\_mot \*p\_lg\_mots\_first;

struct \_str\_lg\_mot \*p\_lg\_mots\_last;

} str\_intervalle;

## B) Découpage modulaire :

• Pour la gestion des intervalles, on aura besoin de 4 fonctions primordiales :

void init\_intervalle (int nb\_car\_interv) : Nous donne le nombre d’intervalles à partir du nombre de caractères donné et les initialise.

void maj\_intervalle (str\_intervalle \*p\_str\_intervalle, str\_lg\_mot \*p\_str\_lg\_mot).

str\_intervalle \* chercher\_intervalle (char \* mot ) : Recherche l’intervalle où se trouve le mot à partir de sa première lettre.

str\_intervalle \* chercher\_intervalle\_par\_bornes (int interv\_min, int interv\_max ) : Recherche l’intervalle à partir de sa borne inférieure et sa borne supérieure.

• pour gérer les listes linéaires chainées, on aura besoin de ces fonctions :

* La liste des longueurs :

void changer\_maillon\_tete\_longueur (str\_lg\_mot \* p\_str\_lg\_mot\_first\_new, str\_lg\_mot \* p\_str\_lg\_mot\_first\_old ) : Mettre à jour le nouveau maillon de la nouvelle et l’ancien (de l’ancienne liste ).

void inserer\_maillon\_longueur( str\_lg\_mot \* p\_str\_lg\_mot, str\_lg\_mot \* p\_str\_lg\_mot\_prev ) : Insérer un nouveau maillon dans la liste des longueurs.

str\_lg\_mot \* creer\_maillon\_longueur( str\_lg\_mot \* p\_str\_lg\_mot\_prev, int lg\_mot ) : Créer un nouveau maillon de longueur.

void supprimer\_maillon\_longueur( str\_lg\_mot \* p\_str\_lg\_mot ) : Supprimer un maillon dans la liste des longueurs.

str\_lg\_mot\* chercher\_maillon\_longueur(str\_intervalle \* p\_str\_intervalle, int lg\_mot ) : Chercher un maillon dans la liste de longueurs.

str\_lg\_mot \* trier\_liste\_longueurs( str\_lg\_mot \* p\_str\_lg\_mot ) : Trier la liste des longueurs en créant une nouvelle liste, la nouvelle liste va avoir le premier maillon de l’ancienne liste comme tête et puis à chaque itération lorsque on parcourt l’ancienne liste on fait des permutations et on s’arrête une fois la queue de l’ancienne liste atteinte.

* La liste des mots :

void changer\_maillon\_tete\_mot( str\_mot \* p\_str\_mot\_first\_new, str\_mot \* p\_str\_mot\_first\_old ) : Changer la tête de la liste des mots.

void inserer\_maillon\_mot( str\_mot \* p\_str\_mot, str\_mot \* p\_str\_mot\_prev ) : insérer un nouveau maillon dans la liste des mots.

str\_mot \* creer\_maillon\_mot(str\_mot \* p\_str\_mot\_first, str\_mot \* p\_str\_mot\_prev, char \* mot ) : Créer un maillon dans la liste des mots.

void supprimer\_maillon\_mot( str\_mot \* p\_str\_mot ) : supprimer un maillon dans liste des mots et libération de la mémoire qu’il occupe.

str\_mot \* chercher\_maillon\_mot(str\_mot \* p\_str\_mot\_start, char \* mot ) : chercher le maillon qui contient le mot inséré par l’utilisateur.

str\_mot \* trier\_liste\_mots( str\_mot \* p\_str\_mot ) : Trier la liste des mots en créant une nouvelle liste, la nouvelle liste aura le premier maillon de l’ancienne liste comme tête et puis à chaque itération lorsque on parcourt l’ancienne liste on fait des permutations (en comparant les lettres des mots, s’il y a égalité dans la première lettre, on compare la deuxième de chacun des mots et ainsi de suite…) et on s’arrête une fois la queue de l’ancienne liste atteinte.

• Les fonctions nécessaires pour la gestion d’index :

int creer\_index(int nb\_car\_interv) : Une fois la longueur de l’intervalle est donnée par l’utilisateur, le tableau d’intervalles est créé (tab\_intervalles), et on ouvre le fichier le fichier avec la fonction fopen suivie de l’option read ‘r’, on utilise la fonction fgetc pour extraire les caractères du texte et on sait qu’on a construit un mot une fois que l’on rencontre un des caractères déclarés dans le programme(‘,.,…) et on indexe le mot trouvé, on s’arrête une fois on atteint la fin du fichier.

void afficher\_index() : On accède au tableau d’index, en accédant au champ intervalle (qui est un tableau de deux cases) on affiche la borne inférieure et supérieure de l’intervalle et on affiche ensuite la première longueur et tous les mots de la liste, une fois la fin de la liste des mots est atteinte on passe à la longueur suivante et on refait ce processus jusqu’à ce que la liste des longueur soit entièrement parcourue, ensuite on passe vers la prochaine case du tableau d’index et on refait ce qui précède jusqu’à ce que l’on parcoure tout le tableau.

void sauvegarder\_index() : On ouvre le fichier où l’on veut sauvegarder l’index avec la fonction fopen suivie de l’option write ‘w’ pour pouvoir modifier son contenu, on accède au tableau qui contient les intervalles et les pointeurs vers les listes des longueurs qui chaque maillon à son tour contient un pointeur vers la liste des mots, on le parcoure et à chaque fois on écrit la borne inférieure de l’intervalle dans le fichier suivie d’un point-virgule, puis la borne supérieure ; puis la longueur du mot ; et les mots séparés avec des points virgules. On s’arrête une fois on atteint la fin : du tableau, de la liste des longueurs et de la liste des mots.

int creer\_index\_avec\_sauvegarde() : L’index doit être sauvegardé dans un fichier vide, et on choisit la structure suivante qui nous aide (info1 ;info2 ;info3 ;info4,info5…) : info1 représente la borne inférieure de l’intervalle, info 2 représente la borne supérieure de l’intervalle et info3 représente la longueur du mot et les autres informations ( info4,info5…) représentent les mots.

int indexer\_un\_mot( char \* mot ) : On récupère le premier caractère du mot et on cherche l’intervalle qui le contient, une fois l’intervalle est trouvé on accède à la liste des longueurs et on cherche si la longueur du mot existe déjà, si oui on cherche dans la liste des mots s’il n’existe pas on l’ajoute sinon un nouveau maillon sera créé dans la liste des mots, et si la longueur du mot n’existe pas alors un nouveau maillon dans la liste des longueurs est créé et le mot sera dans la tête de la liste des mots de même longueur que lui.

int supprimer\_un\_mot\_dans\_index( char \* mot ) : On extrait du mot sa première lettre, on cherche l’intervalle dans lequel il y est, une fois l’intervalle trouvé on cherche dans la liste de longueurset si sa longueur n’y est pas alors le mot n’existe pas sinon on accède à la liste des mots et on cherche le mot, si on le trouve on va le supprimer en modifiant le chainage entre les maillons ( on va affecter l’adresse de son précédant vers celle de son suivant ).

void liberer\_memoire\_index() : On parcourt la tableau principal ( qui contint les intervalles et les pointeurs vers la liste des longueurs, et on va accéder vers la liste des mots et à chaque fois on sauvegarde le suivant du mot et on libère l’adresse qui contient le mot(pointeur courant), ensuite le pointeur courant recevra le suivant.

void reinit\_var() : initialise le nombre d’intervalles à 0 et le nombre de mots dans le fichier à 0 et le nombre de mots inutiles ( insignifiants ) et le nombre des doublons à 0.

•Les fonctions utilisateurs :

int chercher\_mot\_dans\_intervalle( int lg\_mot, int interv\_min, int interv\_max, int nb\_mots ) : On recherche d’abord l’intervalle puis on recherche le maillon longueur dans l’intervalle, si la longueur du mot du mot existe alors on va le chercher dans la liste des mots sinon on affiche un message à l’écran disant qu’on a pas trouvé le mot.

int chercher\_mot\_dans\_index( int lg\_mot, int nb\_mots ) : On cherche l’intervalle où appartient le mot puis on effectue une recherche dans un intervalle grâce à la fonction précédente.

int chercher\_mot\_min\_max( int type\_recherche ) .

•La gestion du menu :

Pour la gestion on doit utiliser des fonctions ask, c’est-à-dire des fonctions qui vont contenir les fonctions précédentes et nous permettre d’entrer leurs paramètres (valeurs, mots…) pour l’exécution.