Partie 3 : Stockage de contacts dans une liste à niveaux

On souhaite utiliser une telle liste pour stocker des contacts et des rendez-vous, en utilisant des recherches / insertions / modifications rapides, ainsi qu'une aide à la saisie (complétion automatique)

Pour ceci, il est nécessaire de créer des structures pour stocker les informations suivantes :

Pour un contact

nom et prénom, sous forme de chaînes de caractères (tableaux de caractères) dynamiques

Pour un rendez-vous

une date de rendez-vous, avec les informations suivantes

jour mois année

L'heure de rendez-vous :

Heure minute

Durée du rendez-vous

Heure minute

L'objet du rendez-vous

Tableau dynamique de caractères

Pour les saisies de tableaux dynamiques de caractères, il est très fortement suggéré / conseillé d'écrire une fonction `char *scanString(void);` qui effectue la saisie et retourne un tableau dynamique stockant uniquement les caractères nécessaires

Pour une entrée de l'agenda

Un contact pourra avoir 0, 1 ou plusieurs rendez-vous stockés : on utilisera donc la structure suivante, pour une entrée de l'agenda :

Un contact

Une liste simplement chaînée de rendez-vous

La liste à niveaux stockera ainsi des entrées dans l'agenda.

Principe de stockage sur plusieurs niveaux

On fera le tri des entrées à partir des informations de contact : on utilisera la chaine ‘nom_prenom’ comme valeur pour rechercher ou insérer des entrées dans l’agenda.

L’ordre de tri et de comparaison est l’ordre lexicographique (ordre du dictionnaire). La chaîne devra être convertie en minuscules pour ces opérations.

Ainsi le contact ‘Nicolas Flasque’ (chaîne « **f**lasque_nicolas ») sera classé après le contact ‘William Dupont’ (chaîne « **d**upont_william »), mais avant le contact ‘Albert Petit’ (chaîne « **p**etit_albert »)

On utilisera une liste à 4 niveaux

Constitution des niveaux : les niveaux correspondent aux lettres successives des chaînes de nom.

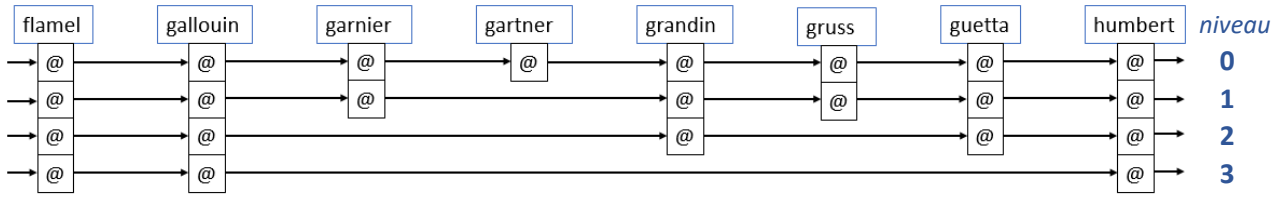
Au niveau le plus élevé (3), on effectue le chaînage sur la première lettre de la chaine (on relie deux cellules si la première lettre de leur chaîne est différente)

Au niveau 2, on effectue le chaînage si la première lettre des chaînes des cellules est la même mais la deuxième est différente

Au niveau 1, on effectue le chaînage si les deux premières lettres des chaînes des cellules sont les mêmes mais la troisième est différente

Au niveau 0, on chaîne toutes les cellules

Illustration



Cahier des charges

1) Réaliser une application de gestion d'agenda

Cette application doit proposer un menu pour

1. Rechercher un contact, et proposer une complétion automatique à partir de la 3ème lettre entrée pour le nom (il faudra donc faire la saisie du nom de recherche caractère par caractère) ;
2. Afficher les rendez-vous d'un contact ;
3. Créer un contact (avec insertion dans la liste) ;
4. Créer un rendez-vous pour un contact (avec insertion dans la liste si le contact n'existe pas) ;
5. Supprimer un rendez-vous ;
6. Sauvegarder le fichier de tous les rendez-vous ;
7. Charger un fichier de rendez-vous ;
8. Fournir les temps de calcul pour une insertion de nouveau contact : voir point 2) ci –dessous.

2) Tracer la complexité de l'application :

Comme pour la liste à niveau comportant des entiers, vous devez produire un graphique de comparaison de temps d'exécution entre :

i) Les opérations de recherche / insertion faites uniquement au niveau 0
ii) Ces mêmes opérations faites à partir du niveau le plus élevé

Lors de la soutenance / démonstration, il vous sera demandé d'exécuter en direct cette partie.

Contraintes techniques

L'application doit être réalisée en mode 'console'. Pas besoin de couleurs ou d'interface graphique.

Les saisies doivent être sécurisées pour les dates et les heures.

Vous devez créer une liste contenant initialement au moins 1000 contacts : pour ce faire, vous pouvez créer un fichier de contacts à partir des deux fichiers 'noms' et 'prénoms' fournis.