Rapport TP3

Listes chaînées - NF16

DÉBUREAUX Anaïs/ SLIMANI Nada- 18 avril 2021

```
printf("================================================================================\n");
printf("||    Bienvenu au menu principal                                              ||\n");
printf("================================================================================\n");
printf("||  1. Initialiser la liste des marques de vaccin disponibles.                ||\n");
printf("||  2. Ajouter et planifier un stock.                                         ||\n");
printf("||  3. Retirer un nombre de vaccins du stock.                                 ||\n");
printf("||  4. Afficher le stock d'un vaccin.                                         ||\n");
printf("||  5. Afficher la planification pour une semaine.                            ||\n");
printf("||  6. Fusionner deux stocks.                                                 ||\n");
printf("||  7. Quitter.                                                               ||\n");
printf("================================================================================\n\n");
```

Introduction

- Objectif :

Le département de l'Oise souhaite mettre en place un système pour gérer son stock de vaccins contre la Covid-19 en temps réel.
Pour ce faire, on va enregistrer pour chaque commune possédant un centre devaccination, le nombre de vaccins dont la commune dispose.

En outre, le nombre de vaccins disponibles pour une commune sera réparti sur plusieurs semaines : cela correspond aux prévisions de vaccins à administrer chaque semaine.

- L'organisation MINIMALE du projet est la suivante :

  - Fichier d'en-tête tp3.h, contenant la déclaration des structures/fonctions de base.
  - Fichier source tp3.c, contenant la définition de chaque fonction.
  - Fichier source main.c, contenant le programme principal,

1. Fonctions créées

```
t_semaine_elt *ajouterSemaine (t_semaine_elt *liste, t_semaine_elt
*semaine) ;
```

Cette fonction permet de répertorier le numéro des semaines dans une liste, tout en les classant par ordre croissant.

- `t_ville_elt *ajouterVille (t_ville_elt *liste, t_ville_elt *ville, t_semaine_elt *l_semaine);`
  Cette fonction permet de répertorier des villes dans une liste.

- `t_semaine_elt *supprimerSemaine (t_semaine_elt *liste, int semaine);`
  Cette fonction permet de supprimer des semaines dans une liste semaine tout en conservant l'ordre croissant de cette dernière et en libérant les ressources des semaines supprimées.

- `t_ville_elt *supprimerVille (t_ville_elt *liste, char* ville);`
  Cette fonction permet de supprimer des villes dans une liste de villes et de libérer les ressources des villes supprimées.

- `bool testSemaine (int semaine, int nb_vaccins);`
  Cette fonction permet de vérifier que le numéro de semaine saisit par l'utilisateur est correcte, c'est à dire compris entre 1 et 53 ainsi que le nombre de vaccins à stocker qui doit être strictement positif.

- `t_vaccin_elt *rechercheTableau(char *marqueV, t_vaccin_elt *GESTION_VACCINS[10], int instance);`
  Cette fonction permet de comparer la marque du vaccin saisit par l'utilisateur à celles stockées dans le tableau de vaccins préalablement créé pour la gestion des vaccins.

- `void desallouerVaccin (t_vaccin_elt **vaccin_ville);`
  Cette fonction permet de libérer les ressources des vaccins, dont la marque et les villes disponibles.

- `void desallouerVille (t_ville_elt *villes_dispo);`
  Cette fonction permet de libérer les ressources des villes, dont le nom et les semaines planifiées.

- `void desallouerSemaine (t_semaine_elt *semaines_planifiees);`
  Cette fonction permet de libérer les ressources des semaines.

- `void desallouerListeVaccin (t_vaccin_elt**GESTION_VACCINS, int instance);`
  Cette fonction permet de libérer les ressources des listes de vaccins.

- `void desallouerListeVille (t_ville_elt**ElementPrem);`
  Cette fonction permet de libérer les ressources des listes de villes.

- `void desallouerListeSem (t_semaine_elt**ElementPrem) ;`

Cette fonction permet de libérer les ressources des listes de semaines.

- `t_ville_elt *trierVilles (t_ville_elt *liste);`
  Cette fonction permet de calculer le nombre total de vaccin par ville et de le mettre à jour. Elle fera appel à copieListe afin de copier les éléments de la liste de villes actuelle dans une nouvelle liste en les classant par ordre croissant du total de vaccins.

- `t_ville_elt *copieListe (t_ville_elt *liste, t_ville_elt *ville);`

Cette fonction copie la liste de villes passée en argument dans une nouvelle liste en triant les villes par ordre croissant du total de vaccins.

2. Complexité

```
/* ========== FONCTIONS ========== */
```

- `t_semaine_elt *creerSemaine(int num_semaine, int nb_vaccins);`

Il n'y a que des instructions simples, pas de boucle ni d'appel récursif donc la complexité est de O(1).

- `t_ville_elt *creerVille(char *ville);`

Il n'y a que des instructions simples, pas de boucle ni d'appel récursif donc la complexité est de O(1).

- `t_vaccin_elt *creerVaccin(char *marque);`

Il n'y a que des instructions simples, pas de boucle ni d'appel récursif donc la complexité est de O(1).

- `t_semaine_elt *ajouterVaccinS(t_semaine_elt *liste, int semaine, int nb_vaccins);`

Cette fonction fait appelle à la fonction testSemaine, creerSemaine qui sont de complexité O(1), de plus on fait appelle à la fonction ajouterSemaine qui est en O(n) et on parcours toute la liste des semaines, donc si p est le nombre de semaine, la complexité de lafonction ajouterVaccinS est en O(n+p), car max(O(1), O(n+p)) = O(n+p).
Etant donné que n et p représente le nombre de semaine la complexité et donc de O(2n) = O(n).

- `t_semaine_elt *deduireVaccinS(t_semaine_elt *liste, int semaine, int nb_vaccins);`

Cette fonction fait appelle à la fonction testSemaine qui est de complexité O(1), de plus on fait appelle à la fonction supprimerSemaine qui est en O(n) et on parcours toute la liste des semaines, donc si p est le nombre de semaine, la complexité de la fonction deduireVaccinSest en O(n+p), car max(O(1), O(n+p)) = O(n+p).
Etant donné que n et p représente le nombre de semaine la complexité et donc de O(2n) = O(n).

- `t_ville_elt *ajouterVaccinV(t_ville_elt *liste, char* ville, int semaine, int nb_vaccins);`

La complexité de ajouterVaccinV est de O(m+n) car, dans la troisième boucle if/elsenous faisons appelle à ajouterVille O(1) et ajouterVaccinS O(n) de ce fait max(O(1),O(n)) = O(n).Nous parcourons également toute la liste des villes, donc si m est le nombre de villes, la complexité de la fonction ajouterVaccinV est en O(n+m).

- `t_ville_elt *deduireVaccinV(t_ville_elt *liste, char* ville, int semaine, int nb_vaccins);`

La complexité de deduireVaccinV est de O(m*n) car nous faisons appelle à la fonction deduireVaccinS qui est de complexité O(n) ainsi que supprimerVille de O(m). De plus, nous parcourons la liste de toutes les villes, on a donc une complexité de O(m+m*n) = O(m*n).

- `void afficherStock(t_vaccin_elt *vaccin);`

Il n'y a que des instructions simple, pas de boucle ni d'appel récursif donc lacomplexité est de O(1).

- `void afficherPlanification(t_vaccin_elt *vaccin, int semaine);`

Il n'y a que des instructions simple, pas de boucle ni d'appel récursif donc lacomplexité est de O(1).

- `t_vaccin_elt *fusionnerStocks(t_vaccin_elt *vaccinA, t_vaccin_elt *vaccinB);`

On parcourt les deux listes à fusionner, l'une après l'autre. Soit n le nombre d'éléments de la première liste et m le nombre d'éléments de la deuxième liste. La complexité est en O(n+m).

`/* ========== FONCTIONS AJOUTÉES========== */`

- `t_semaine_elt *ajouterSemaine (t_semaine_elt *liste, t_semaine_elt *semaine);`

Cette fonction fait appel à la fonction `testSemaine` qui est de complexité O(1) et parcourt toute la liste des semaines, donc si n est le nombre de semaine, la complexité de la fonction `ajouterSemaine` est en O(n).

- `t_ville_elt *ajouterVille (t_ville_elt *liste, t_ville_elt *ville, t_semaine_elt *l_semaine);`

L'ajout se fait en tête de liste, on ne parcourt pas la liste. Il n'y a que des instructions simples, pas de boucle ni d'appel récursif donc la complexité est en O(1).

- `t_semaine_elt *supprimerSemaine (t_semaine_elt *liste, int semaine);`

Cette fonction fait appel à la fonction `testSemaine` ainsi qu'à la fonction `desallouerSemaine` qui sont de complexité O(1). De plus, elle parcourt toute la liste des semaines. Si n est le nombre de semaine, la complexité de la fonction `supprimerSemaine` est en O(n).

- `t_ville_elt *supprimerVille (t_ville_elt *liste, char* ville);`

Cette fonction fait appel à la fonction `desallouerVille` qui est de complexité O(1) et parcourt toute la liste des villes. Si m est le nombre de villes, la complexité de la fonction `supprimerVille` est en O(m).

- `bool testSemaine (int semaine, int nb_vaccins);`

Il n'y a que des instructions simples, pas de boucle ni d'appel récursif donc la complexité est de O(1).

- `t_vaccin_elt *rechercheTableau(char *marqueV, t_vaccin_elt *GESTION_VACCINS[10], int instance);`

Dans cette fonction, on parcourt le tableau des marques de vaccins, tout en le comparant à la marque saisit par l'utilisateur. Si n est le nombre de marque de vaccin qui est stocké dans `instance` et vaut au maximum 10, la complexité de `rechercheTableau` est en O(n).

- `t_ville_elt* trierVilles (t_ville_elt *liste);`

Pour calculer le nombre de vaccins total, on parcourt le chaque semaine de chaque ville. Soient n le nombre de semaines et m le nombre de villes, la complexité de ce calcule est en O(m*n).

Pour ranger les villes dans l'ordre, on parcourt à nouveau la liste de villes en O(m). On fait appel à la fonction copieListe equi ets en O(n). La complexité est en O(n*m).

Complexité globale = O(2*n*m) = O(n*m).

- `t_ville_elt *copieListe (t_ville_elt *liste, t_ville_elt *ville);`

  Pour insérer une ville dans l'ordre croissant dans une liste, on parcourt la liste de n semaines de la ville pour réaliser la copie. La complexité est en O(n).

```
/* ------- LIBÉRER LES RESSOURCES --------------*/
```

- `void desallouerVaccin (t_vaccin_elt **vaccin_ville);`
  Il n'y a que des instructions simples, pas de boucle ni d'appel récursif donc lacomplexité est de O(1).

- `void desallouerVille (t_ville_elt *villes_dispo);`

Il n'y a que des instructions simples, pas de boucle ni d'appel récursif donc la complexité est de O(1).

- `void desallouerSemaine (t_semaine_elt *semaines_planifiees);`
  Il n'y a que des instructions simples, pas de boucle ni d'appel récursif donc lacomplexité est de O(1).

- `void desallouerListeVaccin (t_vaccin_elt**GESTION_VACCINS, int instance);`

  Dans cette fonction, `instance` permet la condition d'arrêt de la boucle. Soit i le nombre d'itération, à chaque incrémentation de i on fait appel à la fonction `desallouerListeville` qui est en O(n*m) et `desallouerVaccin` de complexité O(1). De ce fait, la complexité est de O(i*n*m).

- `void desallouerListeVille (t_semaine_elt *semaines_planifiees);`
  Dans cette fonction, on parcourt la liste de villes. A chaque fois que l'élément de la liste est non nul, on fait appel à la fonction `desallouerListeSem` qui est en O(m) ainsi qu'à `desallouerVille` qui est de complexité O(1). Soit n le nombre de villes contenu dans la liste, la complexité de `desallouerListeVille` est de O(n*m).

- `void desallouerListeSem (t_semaine_elt *semaines_planifiees);`
  Cette fonction parcourt toute la liste des semaines, donc si m est le nombre de semaines de la liste, la complexité de la fonction `desallouerListeSem` est en O(m).