# 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 de vaccination, 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 créer

- `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 les villes dans une liste.

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

- `t_ville_elt *supprimerVille (t_ville_elt *liste, char* ville);`
  Cette fonction permet de supprimer les villes dans la liste villes et libére les ressources de la ville supprimer.

- `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 ue le nombre de vaccins a stocké 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 et celle stocker dans la liste de vaccins préalablement créer, 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 lessemaines planifiées.

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

## 2. Compléxiter

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

- `t_semaine_elt *creerSemaine(int num_semaine, int nb_vaccins);`
  Il n'y a que des instructions simple, 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 simple, 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 simple, 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 la fonction 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 deduireVaccinS 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_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/else nous 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 la complexité 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 la complexité est de O(1).

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

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

- `t_semaine_elt *ajouterSemaine (t_semaine_elt *liste, t_semaine_elt *semaine);`
  Cette fonction fait appelle à la fonction testSemaine qui est de complexité O(1) et parcours 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);`
  Il n'y a que des instructions simple, pas de boucle ni d'appel récursif donc la complexité est de O(1).

- `t_semaine_elt *supprimerSemaine (t_semaine_elt *liste, int semaine);`
  Cette fonction fait appelle à la fonction testSemaine ainsi que la fonction « desallouerSemain » qui sont de complexité O(1), de plus elle parcours toute la liste des semaines, donc 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 appelle à la fonction « desallouerVille » qui est de complexité O(1) et parcours toute la liste des villes, donc si m est le nombre de villes, la complexité de la fonction supprimerVille est en O(m).

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

- `bool testSemaine (int semaine, int nb_vaccins);`
  Il n'y a que des instructions simple, 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 parcours la liste des marques de vaccins, tout en la comparant à la marque saisit par l'utilisateur. Si l est le nombre de marque de vaccin, la complexité de rechercheTableau est en O(l).

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

- `void desallouerVille (t_ville_elt *villes_dispo);`

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

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

- `void desallouerListeVaccin (t_vaccin_elt**GESTION_VACCINS, int instance);`
  Dans cette fonction instance est incrémenter après chaque itération. Posons i le nombre d'itération, à chaque incrémentation on fait appelle à la fonction desallouerListeville qui est en O(q[t]) et desallouerVaccin en O(1). De ce fait, la complexité est de O(i*q[t]).

- `void desallouerListeVille (t_semaine_elt *semaines_planifiees);`
  Dans cette fonction, on parcours la liste de Villes, et à chaque fois que le premier élément de la liste est non nulle, on fait appelle à la fonction desallouerListeSem qui est en O(t) ainsi qu'à desallouerVille qui est de complexité O(1). De ce fait, la complexité de desallouerListeVille est de O(q[t]).

- `void desallouerListeSem (t_semaine_elt *semaines_planifiees);`
  `Cette fonction` parcours toute la liste des semaines, donc si t est l'élément premier de la liste semaine, la complexité de la fonction desallouerListeSem est en O(t).

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