École de technologie supérieure (été 2025) Hugues Saulnier, Eric Thé

# INF-147 Travail Pratique #1

Remise à venir sur moodle

Propagation d'un virus grippal (partie 2)

Travail en équipe

## 0 un ajout au module `m_personne`

La fonction mutatrice `modifier_etat_personne` est définie pour forcer son premier paramètre dans l'état défini par son deuxième paramètre et modifie la probabilité de bouger avec son troisième paramètre.

Vous allez ajouter la mutatrice `modifier_prob_deplacer` qui force la détermination d'une nouvelle probabilité de bouger à une `t_personne`. C'est une fonction essentielle qui servira lorsqu'on désire pour des raisons de salubrité publique augmenter ou diminuer les déplacements dans une population sujette à la maladie (population dans une `t_liste_personnes` )

Je vous donne cette nouvelle interface.

## 1 terminer le module `m_liste de personnes`

Dans le module `m_liste_personnes`, vous avez déjà réalisé l'initialisation, l'ajout initial de personnes, la création du patient zéro et diverses fonctions informatrices de la liste.

Votre fonction `ajouter_des_personnes` a ajouté K personnes saines aux K premiers indices du tableau dynamique de la liste (si ce n'est fait ainsi alors vous devez rectifier).

Pour une certaine vraisemblance biologique, un malade tout juste infecté ne peut transmettre aussitôt la maladie. Vous ajoutez une constante dans l'interface de votre module, par exemple ici fixée arbitrairement à 48 heures, elle pourra être modifiée lors de vos tests

```
// minimum d'heures de maladie pour la transmission
#define NB_HRS_TRANSMISSION  2*24
```

On ajoute maintenant le destructeur et différentes fonctions mutatrices qui permettent la simulation d'une contagion communautaire dans une population fermée.
Toutes ces fonctions ont encore une référence sur une `t_liste_personnes` en premier paramètre.

Le destructeur, la fonction `liberer_liste` reçoit une `t_liste_personnes*`, va faire le `free` du tableau dynamique puis mettre NULL le pointeur et à 0 sa taille et tous ses compteurs. La liste est redevenue vide.

La fonction de gestion `deplacer_un_mort` reçoit une `t_liste_personnes *` et un indice entier. Si cet indice correspond à une `t_personne` de la population et que cette personne est morte, vous devez permuter la personne avec la dernière personne vivante du tableau dynamique. Les `t_personne` mortes doivent toujours venir après les `t_personne` vivantes

La fonction `assurer_temps_maladie` reçoit une `t_liste_personnes*` La fonction traverse itérativement le tableau de la liste et déclenche l'incrémentation des heures de maladie des malades présents. La fonction retourne le nombre de malades ayant atteint `NB_HRS_MALADIE`

La fonction `terminer_maladie` reçoit une `t_liste_personnes*` et la proportion de confinement à appliquer aux survivants en changement d'état. La fonction traverse itérativement le tableau de la liste pour tester le rétablissement ou la mort de celles qui terminent la maladie pour déclencher alors la modification de leurs propriétés (n'oubliez pas qu'un mort sera permuté en fin de liste juste après les vivants et de tester le vivant qui s'est potentiellement déplacé). Les compteurs de la liste sont mis à date. La fonction retourne le nombre de morts apparus lors de cet appel.

La fonction `deplacer_les_personnes` reçoit trois paramètres : une `t_liste_personnes*` et les dimensions de l'espace de simulation. Elle déclenche `deplacer_personne` pour chaque `t_personne` dans la liste. La fonction retourne le nombre de `t_personne` qui ont effectivement bougé.

La fonction `traiter_contacts` reste la plus importante, elle assure la propagation. Elle reçoit en paramètres une `t_liste_personnes*, la` proportion de confinement à appliquer lors de l'apparition d'un nouveau malade. Elle boucle sur toutes les `t_personne` vivantes du tableau dynamique : Pour chaque personne on doit itérativement obtenir la distance entre la personne et **chacune de ses suivantes** et si la distance obtenue est inférieure à `DISTANTE_CONTACT` et si le contact est celui d'une personne malade capable de transmettre la maladie et d'une personne en santé on doit

1. tester la contagion de la personne saine et déclencher si nécessaire le changement d'état chez le nouveau malade et l'incrémentation du nombre de maladies transmises chez le malade.
2. qu'il y ait transmission ou pas, déclencher la fonction `inverser_les_vitesses` à ces deux personnes.

La fonction retourne le nombre total de personnes qui viennent d'être infectés en une heure dans la population

La fonction `simuler_une_heure_pandemie` reçoit quatre paramètres : une `t_liste_personnes*, la` proportion de confinement à appliquer et les dimensions de l'espace de simulation. Elle agit à haut niveau et ne fait que déclencher en séquence les fonctions mutatrices précédentes. Je laisse à chaque équipe de définir et de bien commenter la séquence d'appels choisie. La fonction retourne le nombre de malades actuellement dans la liste.

## 2 le programme principal capable de donner vos premières simulations

Le fichier `pandémie.c` va inclure l'interface de votre module `m_liste_personnes` et pourra simuler dans un main les heures de pandémie d'une population jusqu'à la guérison ou la mort du dernier malade ou la mort de tous les habitants.

Votre main cette semaine va donc

- Initialiser un compteur d'heures
- Définir `la` proportion de confinement voulue
- Initialiser une liste de personnes aux conditions initiales choisies
- Créer le patient zéro
- Déclencher une boucle qui
    a. Incrémenter le compteur d'heures
    b. Déclencher la fonction `simuler_une_heure_pandemie`
    c. Périodiquement afficher le nombre d'heures et un court résumé de l'état de la liste de personnes

Tant que vos conditions ne sont pas satisfaites 😊 Afficher alors les résultats de la simulation

## 3 Les conditions initiales

Remarque primaire pour la région, la zone de contact et de contagion couvre 38.5 mètres², pensez à laisser suffisamment de surface à chaque personne, mettons entre 100 et 200 m²

Par exemple, vous simulez une population de 800 personnes, la surface de la région (largeur x hauteur) peut facilement couvrir 160,000 m² , si on pense à une région carrée, ça correspond à 400 mètres de coté.

Testez pour différentes proportions de confinement { 0.6 , 0.75 , 0.85 , 0.95 }

Testez pour différentes probabilités initiales de contagion, y'a pas que

```
#define PROB_INFECTION  0.4    // probabilité initiale de contracter la maladie
```

Dès qu'une simulation devient plus crédible et intéressante, vous notez toutes ses conditions initiales et les résultats obtenus, avant tout : le nombre de passage dans la boucle (la durée en heures) et les différentes statistiques en fin de pandémie

les **contagions** : nombre total, maximum de contagions /heure, minimum de contagions/ heure, nombre moyen de contagions/heure

même chose pour les **décès**

## 4 Présentation

D'ici le 25 juin lundi de l'ÉTS, votre équipe devra me montrer en présentiel (toute l'équipe est présente avec moi) une exécution de votre logiciel pour ensuite me donner un très court rapport contenant un minimum de 6 exécutions dont les résultats vous semblent variés et intéressants selon vos conditions initiales choisies

Vous m'en donnerez les résultats sur Excel ou tabulées à la main 😊

Cette courte présentation représente le tiers des points du TP1

Une fois la présentation terminée et acceptée, vous déposez dans un .zip unique tous les fichiers du projet (tous les .h, et tous les .c et c'est tout). C'est cet envoi qui fera foi d'une présentation complétée. La qualité du code ne sera jugée qu'avec la remise finale.

Voilà pour la deuxième semaine

## Bon travail