École de technologie supérieure Hugues Saulnier, Eric Thé

INF-147 Travail Pratique #2 Remise: fin de session

Propagation d'un virus grippal (partie 1) Travail en équipe

Objectifs

Vous avez réalisé la simulation de la propagation du virus dans une population fermée.

Dans le TP2, vous allez simuler la propagation du virus dans plusieurs villes simultanément, on ajoute de la complexité avec la possibilité d'échanges de personnes entre les villes.

Ces échanges de personnes seront très régulés : les déplacements seront unidirectionnels et à temps prédéfinis de transit dans chaque ville. Avec par exemple 6 villes en simulation, un `t_migrant` quittant la ville 3 pour la ville 1 passera **nécessairement** par les villes 4 , 5, 0 et 1, un `t_migrant` quittant la ville 2 pour la ville 5 passera donc par les villes 3, 4 et 5

Chaque ville aura sa population dans une `t_liste_personnes` et une liste chainée de `t_migrant` pour les personnes migrantes qui doivent y transiter ou s'y arrêter.

Chaque ville possèdera un temps de transit bien défini que devra réaliser tout migrant qui intègre la liste chainée des migrants de cette ville. C'est valable autant pour ceux qui finissent leur voyage que pour ceux qui y font escale. Les migrants ont toutes et tous l'obligation de passer un nombre fixé d'heures dans la liste chainée avant soit d'intégrer la population de la ville soit de quitter cette liste chainée pour intégrer la liste chainée de la ville suivante.

La définition de la structure `t_ville` vous sera donnée à la fin de cet énoncé 😊

Mais avant d'aller plus loin

De plus petits modules à conserver, à modifier, à développer

Conserver `alea.pop` tel quel

Développer le tout nouveau module `m_ensemble_noms`. Ce module offre une structure spécialisée qui permet d'insérer et de conserver le nom de chaque ville de la simulation. Chaque ville aura ainsi un **nom** et une **position** dans cette structure, ce qui permettra au migrant de respecter le parcours spécifique décrit dans la partie *«objectifs»* précédente.

On vous donne l'interface `(.h)` complète du module et une implémentation `(.c)` partielle avec certaines définitions de fonctions à compléter. Vous aurez partout du commentaire et même un `main` de test.
Faire que `m_ensemble_noms` soit testé et en bon état puisque ce module est primordial à toute simulation à plusieurs villes.

Modifier le module `m_personne`, les modifications à faire et à tester cette semaine sont d'ajouter deux mutatrices

`void modifier_position_personne(t_personne * lui, t_R2 position )` qui force une position à une `t_personne`. Elle sera nécessaire lorsqu'un migrant intègre la population d'une ville

`void modifier_vitesse_personne(t_personne * lui, t_R2 vitesse )` qui force une vitesse à une `t_personne`. Elle sera nécessaire lorsqu'un migrant intègre la population d'une ville

Modifier le module `m_liste_personnes`, les modifications à faire et à tester cette semaine sont au minimum d'ajouter deux mutatrices et une constante

Ajouter la constante EXEDENT_TAB qui assure que le tableau dynamique de `t_personne` dans une `t_liste_personnes` sera capable de s'allonger avec un `realloc` pour ainsi pouvoir ajouter des personnes à la population (70 < EXEDENT_TAB < 150) sans déborder du tableau dynamique.

`int ajouter_une_personne(t_liste_personne * liste, const t_personne * src);` va ajouter une personne dans la population, si le nombre de personnes vivantes dans la ville occupe tout le tableau, on doit l'agrandir (`realloc + assert`) du tableau dynamique de `EXEDENT_TAB` cases avant d'y ajouter la personne. Ajuster les compteurs de la `t_liste_personnes`. Retourner 1 (insertion réussie) ou 0 (peu importe la raison)

`int enlever_une_personne(t_liste_personne * liste, t_personne * dest)` choisir au hasard une personne vivante (malade ou pas) dans la population et la copier dans la référence, remplacer dans la liste la personne extraite par la dernière personne vivante du tableau. Ajuster les compteurs. Retourner 1 (extraction réussie) ou 0 (peu importe la raison)

Si vous le voulez, votre ancienne fonction `ajouter_des_personnes` ne fera plus qu'utiliser `init_personne` et la nouvelle `ajouter_une_personne`.

Développer le module `m_migrant` dont je vous donne l'interface, comme le membre principal de ce nouveau type étant une `t_personne`, ce travail est loin d'être difficile et reste parfaitement naturel.

Terminer la première partie avec le début du développement du module `m_ville`. Voici une version écourtée de son interface .h avec la définition de la structure et le constructeur dont vous pourrez réaliser l'implémentation et ajouter quelques méthodes dont les plus évidentes (les get_)

```
// concaténé au nom de la ville pour obtenir le nom du fichier de résultats
#define EXTENSION_FICHIER    "_log.txt"
#define MAX_NOM_VILLE 50


typedef struct ville * t_ville;


struct ville{
   char nom_ville[MAX_NOM_VILLE]; // nom de la ville
   int largeur, hauteur;        // dimensions de la ville
   double proportion_confinement;   // proportion de désiré dans cette ville
   double prob_emigrer;  // probabilité de vouloir quitter la ville
   t_liste_personnes population;  // les habitants de la ville
   t_liste  migrants ;      // SD des t_migrants, liste chainée quelconque 😊
   int nb_hre_transit;     //  nombre d'heures de transit d'un migrant dans la SD
   //les compteurs du transit de personnes
```

```
    int nb_migrants_out;        //  qui ont quitté la ville
     int nb_migrants_in;        // qui ont rejoint la ville
    int nb_morts_transit;      // nb. de migrants qui sont morts en transit
   // le fichier de log des résultats de la simulation
   FILE * logfile;            // lien au fichier texte des résultats
};
```

`t_ville  init_ville (const char*,int ,int ,int ,double , double ,int);`
Le constructeur recoit sept paramètres : le nom de la ville, hauteur, largeur, taille de la population initiale, proportion de confinement, probabilité d'émigrer et nombre d'heures de transit à passer dans sa liste de migrants.

description de son comportement :

- la structure est obtenue dynamiquement
- les paramètres hauteur, largeur, les deux probabilités, le nombre d'heures de transit sont assignés à leurs champs respectifs
- les trois compteurs de transit mis à 0
- la population est initialisée des valeurs précédentes
- la liste chainée de migrants est initialisée
- le nom de la ville est copié dans la structure ville et ajouté dans `m_ensemble_noms`
- `EXTENSION_FICHIER est concaténé` au nom de la ville et un fichier d'écriture est ouvert (fopen) le FILE* obtenu est conservé dans la structure

Le module `m_ville` offrira des méthodes nombreuses en plus du constructeur, qui vont simuler une heure de pandémie dans la population, insérer des migrants dans la population, extraire des personnes de la population pour en faire des migrants, faire transiter des migrants de cette ville à la ville suivante….*etcetera.*

Module avancé de simulation

Le niveau maximal de synthèse viendra dans la partie 2 du TP avec le module `m_groupe_villes` qui implémentera la structure `t_groupe_villes` dont une variable va contenir toutes les villes et permettre l'échange de migrants entre elles.

BON TRAVAIL!