NF16 - TP 3 – Listes chaînées

Ce TP est un nouveau problème. Il a pour objectif de se familiariser avec les listes chaînées et les différentes opérations nécessaires pour les manipuler.

Énoncé:

Pendant la période épidémique, le système médical est surchargé et les patients doivent attendre longtemps pour obtenir un rendez-vous médical en raison d'une mauvaise gestion et du manque de personnel. Dans ce TP, nous proposons de mettre en place une application pour planifier les rendez-vous des patients en utilisant les listes chaînées et un algorithme glouton donné afin de minimiser le temps d'attente total des patients.

L'application que nous proposons gère les entités suivantes (toutes stockées dans des listes linéaires chaînées) :

1. **Les soigneurs :** chaque soigneur est caractérisé par :
   - Un identifiant unique (de type int, positif)
   - Nom (Max 40 Caractères)
   - Prénom (Max 40 Caractères)
   - Une liste des intervalles de temps durant lesquels il est disponible (initialisée avec un seul intervalle [0, inf[ )

2. **Des intervalles de temps** : pour représenter la disponibilité des soigneurs. Chaque intervalle de temps est caractérisé par :
   - La date de début (de type int, en minutes)
   - La date de fin (de type int, en minutes)

3. **Les patients** : un patient est caractérisé par :
   - Un identifiant unique (de type int, positive)
   - Nom (Max 40 Caractères)
   - Prénom (Max 40 Caractères)
   - Une liste de rendez-vous médicaux

4. **Les rendez-vous médicaux** : chaque rendez-vous médical est caractérisé par :
   - Une description brève (Max 125 Caractères)
   - La date de début souhaitée (de type int, en minutes)
   - La date de fin souhaitée (de type int, en minutes)
   - La date de début affectée (de type int, en minutes, supérieure ou égale à la date de début souhaitée)
   - La date de fin affectée (de type int, en minutes)
   - Un identifiant du soigneur associé (de type int, positif)

Partie I : Structures et fonctions de base

A. Structures

1 Définir les structures suivantes :

- la structure **Intervalle** et le type correspondant **T_Intervalle**.
- la structure **RendezVous** et le type correspondant **T_RendezVous**.
- la structure **Soigneur** et le type correspondant **T_Soigneur.**
- la structure **Patient** et le type correspondant **T_Patient**.

Remarque : chaque structure représente un maillon dans une liste linéaire chaînée, n'oubliez donc pas de rajouter les champs nécessaires.

B. Fonctions et procédures de base

1 Ecrire une procédure qui permette d'ajouter un nouveau soigneur dans une liste :

T_Soigneur* ajouterSoigneur(T_Soigneur* listeSoigneurs, int idSoi, char* nom, char* prenom) ;

Cette fonction renvoie un pointeur vers la tête de la liste de soigneurs après l'ajout d'un soigneur. La liste des intervalles de temps disponibles pour un nouveau soigneur est initialisée avec un seul intervalle [0, inf[, l'identifiant du patient correspondant est -1.

2 Ecrire une procédure qui permette d'ajouter un nouveau patient dans une liste :
T_Patient* ajouterPatient(T_Patient* listePatients, int idPat, char* nom, char* prenom) ;

Cette fonction renvoie un pointeur vers la tête de la liste de patients après l'ajout d'un patient. La liste de rendez-vous médicaux pour un nouveau patient est initialement vide.

3 Ecrire une procédure qui permette d'ajouter un nouveau rendez-vous médicale dans une liste :
T_RendezVous* ajouterRendezVous(T_RendezVous* listeRendezVous, int idSoi, int dateDebut-Souaitee, int dateFinSouaitee, int tempsDeplacement, char* desc) ;

Cette fonction renvoie un pointeur vers la tête de la liste de rendez-vous après l'ajout d'un rendez-vous.

4 Ecrire une procédure pour modifier les informations d'un rendez-vous médical (les dates ou la description) :

void modifierRendezVous(T_RendezVous* listeRendezVous, int idSoi, int dateDebutSouaitee, int dateFinSouaitee, int tempsDeplacement, int char* desc) ;

5 Ecrire une procédure pour supprimer un rendez-vous médical d'un patient en donnant l'identifiant du soigneur correspondant :

T_RendezVous* supprimerRendezVous(T_RendezVous* listeRendezVous, int idSoi) ;

Cette fonction renvoie un pointeur vers la tête de la liste de RdV médicaux après une suppression.

Partie II : Ordonnancement des rendez-vous

L'ordonnancement ou Scheduling en anglais – désigne le procédé par lequel des priorités successives sont données à des tâches différentes.
Une tâche est une entité élémentaire de travail :

- localisée dans le temps par une date de début et de fin,
- dont la réalisation est caractérisée par une durée.

Dans le domaine industriel, l'ordonnancement consiste à organiser dans le temps la réalisation d'une suite de tâches, en prenant en compte les contraintes de production :

- Temporelles, délais requis, retards, priorités
- Techniques, contraintes d'enchaînement, technologie machines
- Capacitaires, disponibilité des ressources
- etc.

Pour une visualisation facile, le diagramme de GANTT est un graphique (chrono gramme) qui consiste à placer les tâches chronologiquement en fonction des contraintes techniques de succession (contraintes d'antériorités). L'axe horizontal des abscisses représente le temps et l'axe vertical des ordonnées les tâches. On représente chaque tâche par un segment de droite dont la longueur est proportionnelle à sa durée. Un outil de visualisation d'un solution d'ordonnancement est implémenté en Python et vous offerte.

1. Définir la structure **Ordonnancement** contenant une date de création en forme "AAAA-MM-JJ", la liste des soigneurs et la listes de patients, et le type correspondant **T_Ordonnancement**.

2. Implémenter une fonction permettant de créer une instance de l'ordonnancement en important les données à partir d'un fichier. Le format du fichier est décrit dans le fichier "instance1.txt" fourni avec le TP.
   T_Ordonnancement* creerInstance(char* filename) ;

Cette fonction renvoie un pointeur vers l'instance.

3. Implémenter une procédure qui permette d'affecter un rendez-vous à un soigneur en fonction de ses disponibilités (ne pas oublier de mettre à jour la date de début affectée et la date de fin affectée du rendez-vous) :
   void affecterRendezVous (T_RendezVous* rdv, T_Soigneur* soigneur) ;

4. Implémenter une procédure pour ordonnancer les rendez-vous des patients en fonction des disponibilités des soigneurs en minimisant la somme des temps d'attente des patients (le temps d'attente est calculé par la date de début affectée – la date de début souhaitée) :
   void Ordonnancer(T_Ordonnancement* solution) ;

Algorithme d'ordonnancement : l'algorithme glouton d'ordonnancement en minimisant la somme du temps d'attente des patients se construit comme suit :

- Étape 1 : Trier les patients par ordre décroissant de durée totale des rendez-vous (la durée d'un rendez-vous est calculée par la date de fin souhaitée – la date de début souhaitée)
- Étape 2 : Affecter les rendez-vous des patients dans l'ordre ci-dessus aux intervalles de temps disponible des soigneurs.

5 Ecrire une procédure qui permette d'exporter la solution d'un ordonnancement dans un fichier texte. Le format du fichier est décrit dans le fichier "solution.txt" fourni avec le TP.

void ExportSolution(T_Ordonnancement* solution, char* filename) ;

Le nom du fichier exporté a sufix de la date de création d'un ordonnancement, par exemple filename='solution.txt', et la date de creation est '2020-10-13', donc le nom de ficher exporté est 'solution.txt.2020-10-13.txt'.

Travail attendu

A. Programme Principal :

Utiliser les fonctions précédentes pour proposer à l'utilisateur le menu interactif suivant :

1 Créer une instance à partir d'un fichier
2 Afficher tous les patients et leurs rendez-vous
3 Afficher tous les soigneurs et leurs intervalles de temps disponibles
4 Afficher un rendez-vous en indiquant l'identifiant du patient et le soigneur correspondant
5 Modifier un rendez-vous en indiquant l'identifiant du patient et celui du soigneur correspondant
6 Supprimer un rendez-vous en indiquant l'identifiant du patient et celui du soigneur correspondant
7 Ordonnancer
8 Exporter la solution d'un ordonnancement
9 Quitter_

B. Consignes générales :

- **Sources**

À la fin du programme, les blocs de mémoire dynamiquement alloués doivent être proprement libérés. Vous devrez également être attentifs à la complexité des algorithmes implémentés. Toutes fonctions avec un prefix 'provided_' est offert et implémenté dans la libraire libTP3_provided.so. Pour compiler le programme: ouvrir une terminale, allez dans le repertoire du fichier "CmakeLists.txt", lancer "cmake . && make" .

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 libraire libTP3_provided.so, lequel est déjà offert,
- Fichier source main.c, contenant le programme principal,
- Fichier compilation CmakeLists.c, contient la configuration de compilation.

➢ Rapport

Votre rapport de quatre pages maximum contiendra :

- La liste des structures et des fonctions supplémentaires que vous avez choisi d'implémenter et les raisons de ces choix.
- Un exposé succinct de la complexité de chacune des fonctions implémentées.

Votre rapport et vos fichiers source feront l'objet d'une remise de devoir sur Moodle dans l'espace qui sera ouvert à cet effet quelques jours suivant votre démonstration au chargé de TP (un seul rendu de devoir par binôme).

➢ Annexes

Format d'un ficher importé :
NbPatients NbSoigneurs
Id_Pat NbRdV Nom Prénom
Id_Soi Date_Début_souhaitée Date_Fin_souhaitée Temps_déplacement_précédent Une_description
……
Id_Soi Date_Début_souhaitée Date_Fin_souhaitée Temps_déplacement_précédent Une_description
Id_Pat NbRdV Nom Prénom
……
Id_Soi Nom Prénom
……
Id_Soi Nom Prénom

Format d'une solution exportée :
NbPatients NbSoigneurs
Id_Pat NbRdV
Id_Soi Date_Début_affectée Date_Fin_affectée Temps_déplacement_précédent
……
Id_Soi Date_Début_affectée Date_Fin_affectée Temps_déplacement_précédent
Id_Pat NbRdV
……

Un exemple :
instance1.txt

```
2 3
1 2 PAA PBB
1 20 50 0 traitementA
2 70 100 20 traitementB
2 3 PCC PDD
1 25 60 0 traitementC
2 80 100 20 traitementD
3 125 145 25 traitementE
1 SAA SBB
2 SCC SDD
3 SEE SFF
```

L'algorithme glouton va ordonnancer les traitements demandés par les patients, un exemple de la solution exportée est :
solution.txt

```
2 3
1 2
1 60 90 0
2 110 140 20
2 3
1 25 60 0
2 80 100 20
3 125 145 25
```

La visualisation de l'instance et de la solution (lancer la commande dans une terminale : python OrdonnancementVisualisation.py, le programme est fourni en annexe) :

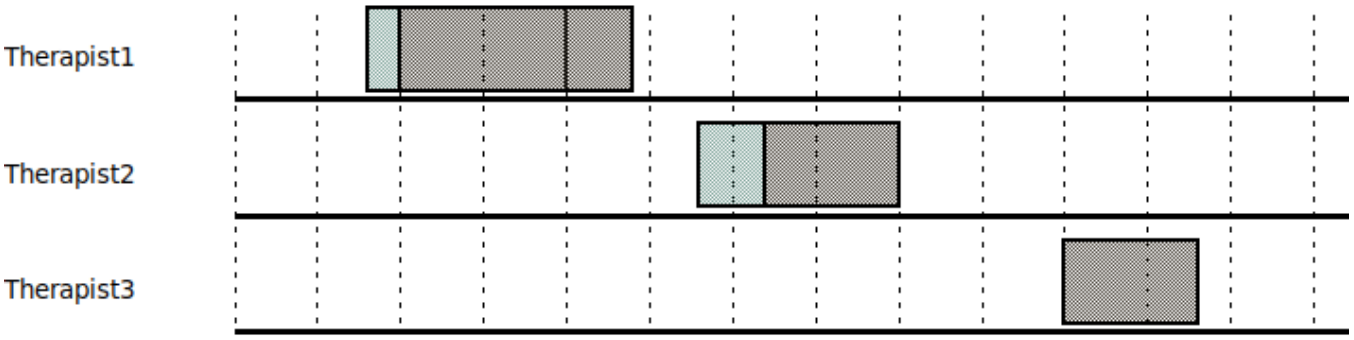

Fig.1 La visualisation de l'instance

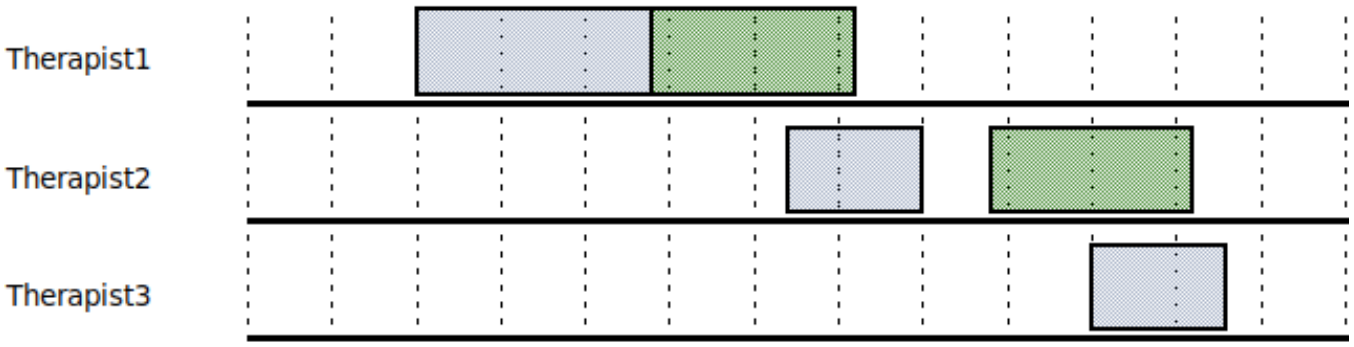

Fig.2 La visualisation de la solution

Tips :
1. L'application est défilante et traînable, on peut également zoomer et dézoomer pour la visualisation.
2. Lorsque la souris est placée sur le rendez-vous, les détails s'affichent
3. Focus sur un patient en faisant un clic gauche sur le rendez-vous, puis tous les rendez-vous associés du patient sélectionné seront affichés, les autres seront masqués. Cliquez avec le bouton droit sur n'importe quel rendez-vous visibles, vous retournerez à la vue principale de la planification des rendez-vous.