NF16 – A19 - TP 4 – Arbres binaires

Ce TP a pour objectif de se familiariser avec les arbres binaires et les différentes opérations nécessaires pour les manipuler, ainsi que leur variante : les arbres binaires d'intervalles.

Énoncé :

Les arbres d'intervalles sont des structures de données qui permettent de gérer une famille d'intervalles. Un arbre binaire d'intervalles est un arbre binaire dont les noeuds contiennent des intervalles (et non pas de simples valeurs numériques).

La clé utilisée pour comparer les nœuds est l'extrémité gauche des intervalles.

On dit que deux intervalles I et I' se chevauchent lorsque $I \cap I' \neq \emptyset$. Il est facile de constater qu'un couple d'intervalles (I, I') ne peut vérifier qu'une et une seule des trois propriétés suivantes :

(i) I est à gauche de I', exemple, I=[531,604], I'=[701,704];

(ii) I et I' se chevauchent, exemple : I=[531,604], I'=[601,605] ;

(iii) I est à droite de I', exemple : I=[531,604], I'=[124,128] .

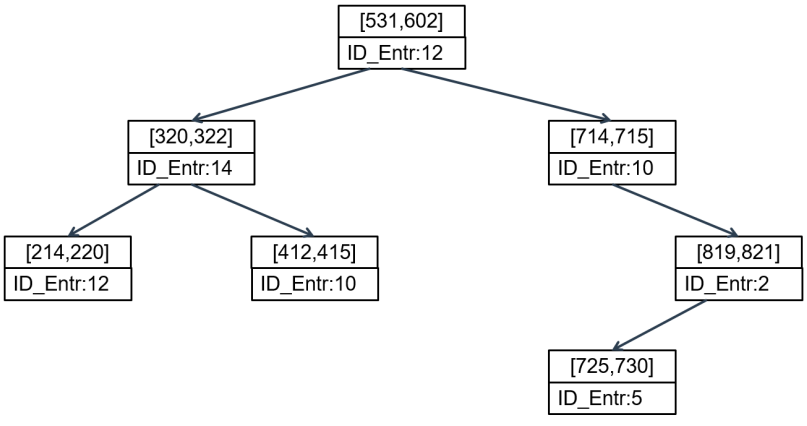
On désire gérer les réservations d'une salle des événements pour les entreprises, la gestion se fait sur une année. Il existe différentes structures pour effectuer ce genre de gestion efficacement. Durant ce TP, nous utilisons les arbres d'intervalles.

Une réservation caractérisé par:

- l'identifiant de l'entreprise de type int.
- intervalle de réservation.

Un intervalle est caractérisé par deux borne de type int. Les dates sont sous la forme MMJJ (ex : 602 pour le 02/06/2019). Le chevauchement de deux intervalles n'est pas autorisé.

Exemple :



A. Structures

1. Définir la structure Intervalle et le type correspondant **T_Inter.**

2. Définir la structure **Noeud** et le type correspondant **T_Noeud** qui représente un nœud d'un arbre d'intervalles. Un nœud représente une réservation. Elle contiendra entre autres un pointeur vers le fils gauche et le fils droit du nœud, mais **pas de pointeur vers le nœud parent**.

3. Définir le type **T_Arbre** - de type pointeur vers une structure **T_Noeud** - qui servira à représenter l'ABR.

B. Fonctions requises

1. Créer un noeud:

   ```
   T_Noeud* creer_noeud(int Id_entreprise,T_inter intervalle)
   ```

2. Ajouter un nœud :

   ```
   void ajouter_noeud(T_Arbre* abr,T_Noeud* noeud)
   ```

3. Rechercher une réservation :

   T_Noeud* recherche(T_Arbre abr, T_inter intervalle,int Id_entreprise)

   cette fonction renvoie un pointeur vers la réservation recherchée, sinon pointeur NULL.

4. Supprimer une réservation :

   void Supp_noeud(T_Arbre *abr, T_inter intervalle,int Id_entreprise)

5. Modifier une réservation :

   void modif_noeud(T_Arbre abr, T_inter intervalle,int Id_entreprise,T_inter NouvelIntervalle)

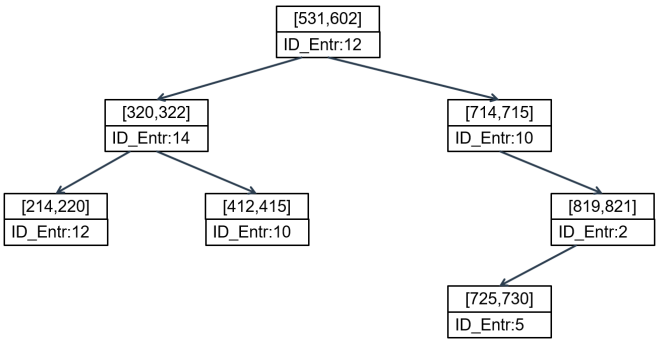
6. Afficher toutes les réservations dans l'arbre :

   void affiche_abr(T_Arbre abr)

   cette fonction affiche les réservations dans l'arbre par ordre croissant (choisissez le bon type de parcours des noeud).

L'affichage devra ressembler à ce qui est montré ci-dessous :

Arbre



Affichage

```
--> ID_Entr:12 -- [214,220]
--> ID_Entr:14 -- [320,322]
--> ID_Entr:10 -- [412,415]
--> ID_Entr:12 -- [531,602]
--> ...
```

7. Afficher les réservation d'une entreprise :
   void affiche_entr(`T_Arbre abr`,int Id_entreprise)
8. Implémenter la fonction **`void detruire_arbre(T_Arbre *abr)`** qui permet de détruire tous les nœuds d'un arbre binaire de recherche.

A. Programme Principal :

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

1. Créer un ABR
2. Afficher tous les réservations
3. Ajouter une réservation
4. Modifier une réservation
5. Supprimer une réservation
6. Afficher les réservations d'une entreprise
7. Supprimer l'arbre
8. Quitter

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.

L'organisation MINIMALE du projet est la suivante :

- Fichier d'en-tête tp4_abr.h contenant la déclaration des structures/fonctions de base ;
- Fichier source tp4_abr.c contenant la définition de chaque fonction,
- Fichier source main.c, contenant le programme principal.

- **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).