AI01/NF16 – TP4 : 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.

Enoncé :

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 nœuds 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]

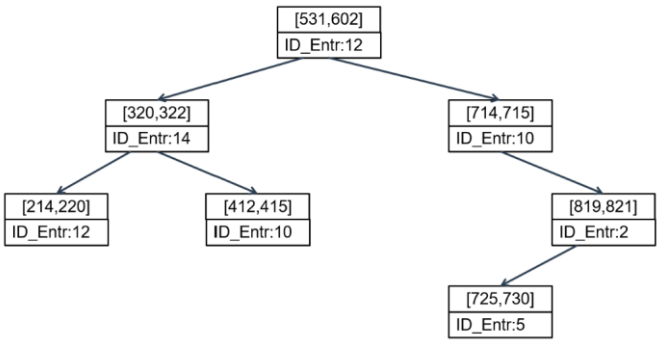
On désire gérer les réservations d'une salle d'événements pour les entreprises au moyen d'arbres binaires d'intervalles. La gestion se fait sur une année.

Une réservation est caractérisée par :

- L'identifiant de l'entreprise de type ***int***
- L'objet de la réservation de type ***char****
- L'intervalle de réservation

Un intervalle est caractérisé par deux bornes de type ***int***.
Les dates sont sous la forme MMJJ (ex : 602 pour le 02/06/2024, 1123 pour le 23/11/2024).
Le chevauchement de deux intervalles n'est pas autorisé.

Exemple :



A. Structures de données :

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. Il 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 à implémenter :

1. Créer un nœud :

```
T_Noeud *creer_noeud(int id_entr, char *objet, T_inter intervalle)
```

2. Ajouter une réservation :

```
void ajouter(T_Arbre *abr, int id_entr, char *objet, T_inter intervalle)
```

3. Rechercher une réservation :

```
T_Noeud *rechercher(T_Arbre abr, T_inter intervalle, int id_entr)
```

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

4. Supprimer une réservation :

```
void supprimer(T_Arbre *abr, T_inter intervalle, int id_entr)
```

5. Modifier les dates d'une réservation :

```
void modifier(T_Arbre abr, int id_entr, T_inter actuel, T_inter nouveau)
```

6. Afficher toutes les réservations présentes dans l'arbre :

```
void afficher_abr(T_Arbre abr)
```

*Cette fonction affichera les réservations **par ordre croissant** sous la forme :*

```
14/02/2024 au 20/02/2024 : entr. 12 – Colloque
20/03/2024 au 22/03/2024 : entr. 14 – Séminaire
12/04/2024 au 15/04/2024 : entr. 10 – Salon
31/05/2024 au 02/06/2024 : entr. 12 – Exposition
14/07/2024 au 15/07/2024 : entr. 10 – Colloque
25/07/2024 au 30/07/2024 : entr. 5 - Salon
19/08/2024 au 21/08/2024 : entr. 2 – Séminaires
```

7. Afficher les réservations d'une entreprise :

```
void afficher_entr(T_Arbre abr, int id_entr)
```

*Cette fonction affichera les réservations **par ordre croissant**.*

8. Afficher toutes les réservations sur une période :

```
void afficher_periode(T_Arbre abr, T_inter periode)
```

*Cette fonction affichera les réservations **par ordre croissant**.*

On prendra en compte les réservations dont l'intervalle chevauche en partie la période en paramètre (commence avant la période et se termine pendant la période, ou commence pendant la période et se termine après la période) afin de voir toutes les disponibilités de la salle entre les deux dates données.

9. Détruire tous les nœuds d'un arbre binaire :

```
void detruire_arbre(T_Arbre *abr)
```

C. Interface :

Utiliser les fonctions précédentes pour écrire un programme permettant de manipuler les matrices creuses. Le programme propose un menu qui contient les fonctionnalités suivantes :

1. Afficher toutes les réservations
2. Afficher les réservations d'une entreprise
3. Afficher les réservations sur une période
4. Ajouter une réservation
5. Modifier une réservation
6. Supprimer une réservation
7. Quitter

***NB** : on considère que l'on travaille sur l'année en cours, 2024 par exemple, toutes les réservations seront donc comprises entre le 01/01/2024 et le 31/12/2024*

Consignes générales :

- Sources

À la fin du programme, **les blocs de mémoire dynamiquement alloués doivent être proprement libérés**.

L'organisation MINIMALE du projet est la suivante :

- Fichier d'en-tête tp4.h, contenant la déclaration des structures/fonctions de base,
- Fichier source tp4.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 trois fichiers feront l'objet d'une remise de devoir sur **Moodle** dans l'espace qui sera ouvert à cet effet (un seul rendu de devoir par binôme).