Algorithmique et Structures de données

L2 2021-2022

Travaux Pratiques 5 - Arbres

`gcc mon_fichier.c -std=c11 -Wall -Wextra -o mon_programme`
Les exercices marqués d'une étoile sont optionnels et à faire dans un second temps.

Dans ce tp on travaille avec des arbres binaires étiquetés. On utilisera les définitions de types suivantes.

```
#include<stdlib.h>
typedef struct noeud
{
   int val;
   struct noeud* g;
   struct noeud* d;
} Noeud;
typedef Noeud* Arbre
```

Un arbre vide sera défini par

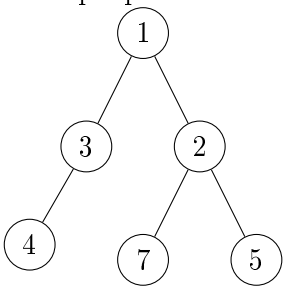
```
Arbre a=NULL;  // a est un arbre vide
```

Exercice 1. *Opérations de bases*

Définir les fonctions suivantes :

- `Arbre anul()` qui renvoie un arbre vide.
- `Arbre ass(Arbre gauche,Arbre droit,int etiquette)`. Qui assemble deux sous-arbres pour produire un nouvel arbre. Attention à la gestion de la mémoire.
- `void liberer(Arbre a)` qui libère la place allouée pour construire l'arbre `a`.

Par exemple pour construire l'arbre :



On peut utiliser l'appel suivant :

`ass(ass(ass(anul(),anul(),4),anul(),3),ass(ass(anul(),anul(),7),ass(anul(),anul(),5),2),1)` (Les arbres vides sont implicites et ne sont pas représentés sur la figure)

Exercice 2. *Parcours !*

Écrire les fonctions suivantes :

1. `void parcours_postfixe(Arbre a)` qui affiche dans l'ordre postfixé les étiquettes de l'arbre.
2. `void parcours_infixe(Arbre a)` qui affiche dans l'ordre infixe les étiquettes de l'arbre avec des parenthèses.
3. * `void parcours_prefixe(Arbre a)` qui affiche dans l'ordre prefixe le parcours de l'arbre.

Exercice 3. *Construction par parcours préfixe*

1. Construire une fonction `int construire_arbre(Arbre* A)` permettant de construire un arbre à partir d'entiers positifs lus au clavier. Les nœuds seront décrits par un parcours en en profondeur préfixe, un fils vide sera codé par -1. Ainsi la suite
   `1 3 4 -1 -1 -1 2 7 -1 -1 5 -1 -1`
   décrit l'arbre du début du sujet.
   Aide : On pourra utiliser `scanf` pour lire les nombres entrés.

2. * Construire la fonction `int construire_arbref(Arbre* A,FILE* f)` qui construit l'arbre à partir d'un fichier ouvert en lecture.
3. *`int construire_arbret(Arbre* A,int* tab,int* place)` qui construit l'arbre à partir d'un tableau de nombre.
4. * `int construire_arbrep(Arbre* A,int** tab)` même exercice mais utilisant un double pointeur. Exemple d'utilisation :

```
char T[]={2,-1,-1};
char* t=T;
Arbre a=NULL;
construire_arbrep(&a,&t);
```

Exercice 4. *Compte toutes les feuilles sans te tromper !*
Écrire les fonctions suivantes.

1. `int nb_noeud(Arbre a)` qui renvoie le nombre de nœuds (étiqueté) d'un arbre.
2. `int nb_feuilles(Arbre a)` qui renvoie le nombre de feuilles d'un arbre. (Une feuille est un noeud qui n'a que des fils vide).
3. *`int nb_noeudI(Arbre a)` qui renvoie le nombre de nœuds interne (Un nœud interne est un noeud qui n'est pas une feuille).