SDD : TP 3

Mathieu Boutin - Jérémy Morceaux

June 10, 2018

1 Présentation générale

- Ce TP à pour but de travailler sur la représentation des arbres en mémoire. Tout d'abord, il fallait utiliser la représentation lien vertical/horizontal. Puis de vérifier le résultat avec le débuggeur ddd. Ensuite, il fallait passer de cette représentation à celle de la représentation postfixée. Par la suite, il fallait créer une méthode d'insertion d'un nœud dans la représentation postfixée. Finalement, il fallait créer une copie de l'arbre initial et créer un autre type d'arbre où chaque noeud possède l'arbre du noeud père.

- Schéma de base :

Structure des fichiers de matrices : prenons par exemple une matrice n*m

1ère ligne fichier : <nombre de lignes> <nombre de colonne>
2ème ligne du fichier: $m_{0,1}$ $m_{0,2}$ ... $m_{0,m-1}$
...
n+1ème ligne du fichier : $m_{n-1,1}$ $m_{0,2}$ ... $m_{n-1,m-1}$

- Les fichiers sources se trouvent dans le dossier **src**.Les fonctions relatives au matrices sont dans le fichier **ZZ_matrix.c** et les fonctions relatives aux listes chainées sont dans le fichier **ZZ_linked_list.c**. Les entêtes sont dans le dossier **include**.

2 Détail de chaque fonction

2.1 createTree (char * formatage)

Principe : createTree (char * formatage)

On initialise une pile, des pointeurs de parcours, et des variables
Pour chaque caractère de notre chaine :
Si le caractère courant est une parenthèse ouvrante:
On le sauvegarde pour la prochaine itération
Si le caractère est une parenthèse fermante :
Si la pile est non vide:
On la dépile pour faire pointer le noeud de parcours sur un noeud père
Sinon:
On s'arrête
Si le caractère est une virgule:
On le sauvegarde pour la prochaine itération
Sinon [Si c'est une lettre]:
On ajoute un noeud au noeud courant : sur le le lien vertical si le caracètre pécèdent était une parenthèse ouvrante, sur le lien horizontale si le caractère précèdent était une virgule ou une parenthèse fermante.
On renvoie l'arbre
FIN

Lexique :

- Paramètre(s) de la fonction
  - formatage contient la chaîne de caractère décrivant l'arbre à créer.
- Variable(s) locale(s)
  - curr est le pointeur courant qui parcourt la liste.

Programme commenté :

Source code : findElt()

2.2 Représentation Postfixée

Principe : Représentation Postfixée

```
On initialise des pointeurs, une pile et des variables;
Si l'arbre est non vide alors:
    Tant qu'on n'a pas parcouru tout l'arbre faire:
        Si l'élément a déjà été traité alors :
            On affiche l'élément avec son nombre de fils;
            S'il a un frère alors:
                On accède à son frère;
            Sinon:
                On accède à son père s'il en a un;
        Sinon: [S'il n'a pas déjà été traité]
            Tant qu'il existe un fils faire :
                On accède au fils de l'élément;
            On affiche le dernier fils;
            S'il a un frère alors:
                On accède à son frère;
            Sinon: [il n'a pas de frère]
                On accède à son père si il existe;
```

FIN

Lexique :

- Paramètre(s) de la fonction
  - L'adresse de l'arbre
  - L'adresse d'une variable de Code Erreur
- Variables locales:
  - cur est le pointeur courant qui parcourt l'arbre
  - stack est la pile qui servira à stocker les élément lors du parcours et de remonter dans l'arbre
  - elmt est le type d'élément stocké dans la pile stack
  - wasInStack est un entier représentant un booléen afin de savoir si l'élément a déjà été empilé et donc traité
  - end est un entier traduisant un booléen afin de savoir si on a parcouru tout l'arbre

Programme Commenté :

2.3 Recherche avec parcours du 1er ordre par niveau et insertion

---

Principe : Recherche avec parcours 1er ordre par niveau

```
On initialise des pointeurs et des variables;
Si l'arbre est non Vide alors:
    Tant qu'on a pas parcouru tout l'arbre ou qu'on n'a pas trouvé la valeur v faire:
        Si l'élément a un fils alors:
            On stock l'élément dans la file;
        Si l'élément possède un frère alors:
            on accède à son frère;
        Sinon:
            Si la file est non vide alors:
                On retourne sur le premier élément enfiler;
            Sinon:
                On a parcouru tout l'arbre;
```

FIN

Lexique :

- Paramètre(s) de la fonction
  - L'adresse de l'arbre
  - L'adresse d'une variable de Code Erreur
  - La valeur v du nœud à chercher
- Variables locales:
  - cur est le pointeur courant qui parcourt l'arbre
  - queue est la file qui servira à stocker les élément lors du parcours de l'arbre selon le 1er ordre

Principe : Insertion

```
On initialise des pointeurs et des variables;
Si le père existe:
    S'il n'a pas de fils alors :
        On insère en tête le fils;
    Sinon: [il a au moins un fils]
        Si le premier fils est avant que le fils à insérer dans l'ordre alphabétique alors:
            Tant qu'on n'a pas trouvé la place où on insère le fils faire:
                On parcourt ses frères
            Si on est à la fin des fils alors:
                On insère le fils à la fin des fils;
            Sinon: [on a trouvé une place où insérer le fils entre 2 frères]
                Si le fils n'existe pas déjà alors:
                    On insère le fils;
                Sinon: [ le fils existe déjà pas besoin ]
                    On met fin au parcours;
        Sinon: [alors on l'insère en tête ]
            On insère le fils;
```

FIN

- end est un entier traduisant booléen

Programme Commenté :

Lexique :

- Paramètre(s) de la fonction
  - L'adresse du nœud où l'on doit insérer le fils
  - L'adresse d'une variable de Code Erreur
  - La valeur w du fils à insérer
- Variables locales:
  - cur est le pointeur courant qui parcourt l'arbre
  - prec est le pointeur qui pointe sur l'élément précédent, ici sur le frère précédent

Programme Commenté :

2.4 Représentation Postfixée avec la nouvelle structure

Principe :Représentation Postfixée Bis

```
On initialise des pointeurs, une pile et des variables;
Si l'arbre est non vide alors:
    Tant qu'on n'a pas parcouru tout l'arbre faire:
        Si l'élément a déjà été traité alors :
            On affiche l'élément avec son nombre de fils;
            S'il a un frère alors:
                On accède à son frère;
            Sinon:
                On accède à son père s'il en a un;
        Sinon: [S'il n'a pas déjà été traité]
            Tant qu'il existe un fils faire :
                On accède au fils de l'élément;
            On affiche le dernier fils;
            S'il a un frère alors:
                On accède à son frère;
            Sinon: [il n'a pas de frère]
                On accède à son père si il existe; FIN
```

Lexique :

- Paramètre(s) de la fonction
  - L'adresse de l'arbre
  - L'adresse d'une variable de Code Erreur
- Variables locales:
  - cur est le pointeur courant qui parcourt l'arbre
  - end est un entier traduisant un booléen afin de savoir si on a parcouru tout l'arbre
  - elmt est le type d'élément stocké dans la pile stack

– wasInStack est un entier représentant un booléen afin de savoir si l'élément a déjà été empilé et donc traité

Programme Commenté :

3 Compte rendu d'exécution

Makefile :

Makefile