BARRAULT Maxime
BELLEC Louison

Groupe12

COMPTE RENDU TP N°3

SDD– ZZ1

Le 29/05/2020

UNIVERSITÉ Clermont Auvergne

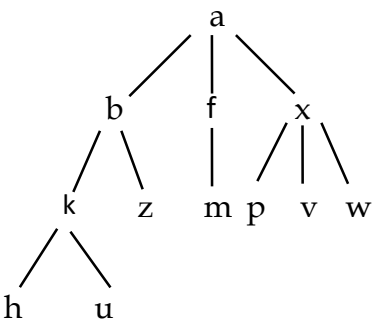
ISIMA
Institut Supérieur d'Informatique de Modélisation et de leurs Applications

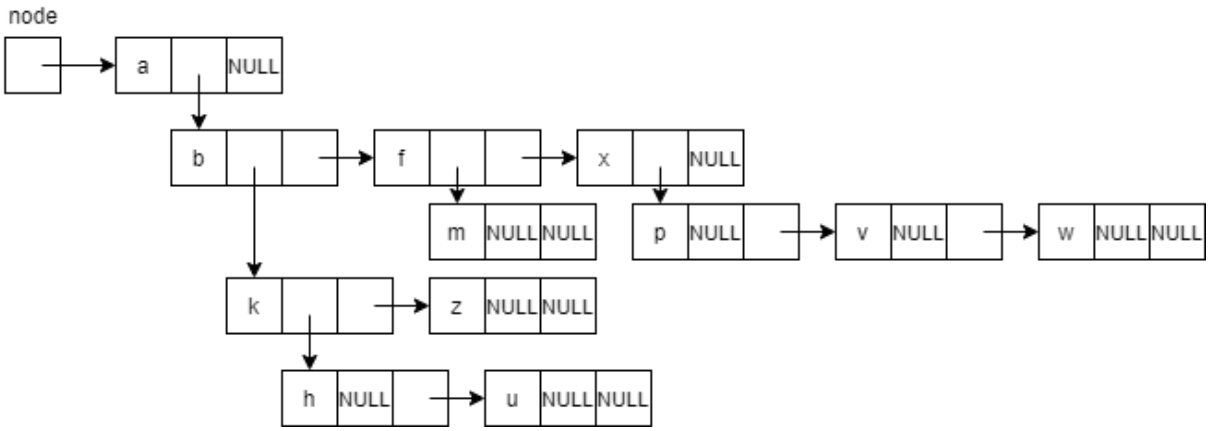
Table des matières

Objet Du TP

L'objet de ce TP consiste à écrire un programme de création d'un arbre en utilisant la représentation par lien vertical et lien horizontal à partir de sa notation algébrique. De plus deux sous-programmes permettront d'afficher l'ordre post fixé de l'arbre et d'insérer un fils à n'importe quel nœud de l'arbre.

Schéma de la structure de donnée





Fichier de donnée utilisé

Pour ce TP un seul fichier de donnée sera utilisé : un fichier texte contenant la notation algébrique de l'arbre à créer. Comme exemple voici la notation de l'arbre vu précédemment :

a×(b×(k×(h+u) +z) +f×m+x×(p+v+w)).

Organisation du code source

Dans le dossier trees se trouve le dossier src qui contient tous nos programmes et sous-programmes :

-dossier queue : contient les sous-programmes pour les files.
-dossier stack : contient les sous-programmes pour les piles.
-Tree.c et treeh : contient les sous-programmes des arbres.
-main.c : contient la fonction pour lire un fichier texte contenant la notation algébrique, et le jeu de test.
-file.txt : contient la notation de l'arbre algébrique à tester.

Remarques

-Durant le TP nous avons remarqué que nos sous-programmes sur les files ne fonctionnaient pas correctement, nous les avons modifiés en conséquence.

-Pour la question 1, nous considérons que la notation algébrique ne contient pas de parenthèse inutile comme par exemple : (a)x((b)xk … Cependant si notre programme devait traiter ce cas-là, le principe serait le suivant :

On empile tous les caractères de de la notation algébrique jusqu'à une parenthèse fermante. Puis on dépile jusqu'à une parenthèse ouvrante tout en créer les maillons et le chainage.

-De plus pour la notation algébrique les fois sont représentés par « * », elle n'accepte aucun espace et tous les autres caractères hormis « + », « * », «(» et «) » seront considérés comment le nom d'un nom a inséré dans l'arbre.

Jeux de test

Notre jeu de test se trouve dans la fonction main. Dans ce test nous créer un arbre, testons l'affichage postfix, la fonction search et enfin nous testons la fonction insertion en insérant l au nœud a, s au nœud k, i au nœud u. Puis on essaye d'insérer dans un nœud inexistant. Puis on test le tri par ordre alphabétique en insérant o, r et y au nœud.

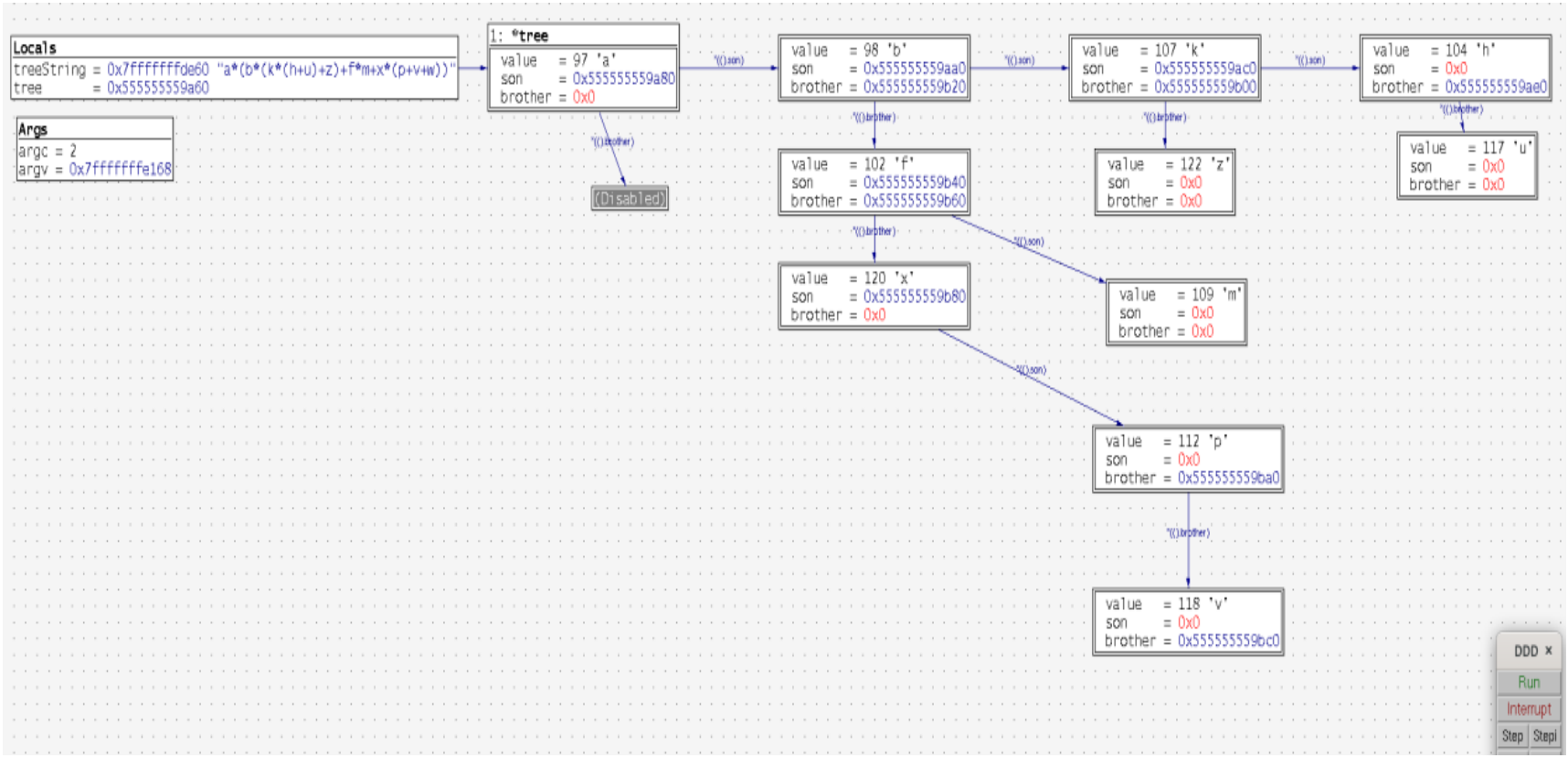
```
src: file.text
Chaine a transformer en arbre: a*(b*(k*(h+u)+z)+f*m+x*(p+v+w))
Ordre postfix:
hukzbcmfpvwxa
======TESTS======

=== TEST: creation arbre ===

=== TEST: ordre postfix ===
Ordre postfix:
hukzbmfpvwxa

=== TEST: fonction search ===
recherche du noeud a: a
recherche du noeud f: f
recherche du noeud u: u
recherche du noeud w: w

=== TEST: fonction insertion ===
insertion: racine
Ordre postfix:
hukzbmflpvwxa
insertion: milieu
Ordre postfix:
hsukzbmflpvwxa
insertion: feuille
Ordre postfix:
hsiukzbmflpvwxa
insertion: noeud inexistant
ERROR: Insertion dans un noeud inexistant
insertion: devant la fratrie
Ordre postfix:
hsiukzbmflopvwxa
insertion: milieu de la fratrie
Ordre postfix:
hsiukzbmfloprvwxa
insertion: fin de la fratrie
Ordre postfix:
hsiukzbmfloprvwyxa
```


Locals
treeString = 0x7fffffffde60 "a*(b*(k*(h+u)+z)+f*m+x*(p+v+w))"
tree = 0x555555559a60
Args
argc = 2
argv = 0x7fffffffe168
1: *tree
value = 97 'a'
son = 0x555555559a80
brother = 0x0
{Disabled}
value = 98 'b'
son = 0x555555559aa0
brother = 0x555555559b20
value = 107 'k'
son = 0x555555559ac0
brother = 0x555555559b00
value = 104 'h'
son = 0x0
brother = 0x555555559ae0
value = 117 'u'
son = 0x0
brother = 0x0
value = 102 'f'
son = 0x555555559b40
brother = 0x555555559b60
value = 122 'z'
son = 0x0
brother = 0x0
value = 120 'x'
son = 0x555555559b80
brother = 0x0
value = 109 'm'
son = 0x0
brother = 0x0
value = 112 'p'
son = 0x0
brother = 0x555555559ba0
value = 118 'v'
son = 0x0
brother = 0x555555559bc0
DDD
Run
Interrupt
Step
Stepi