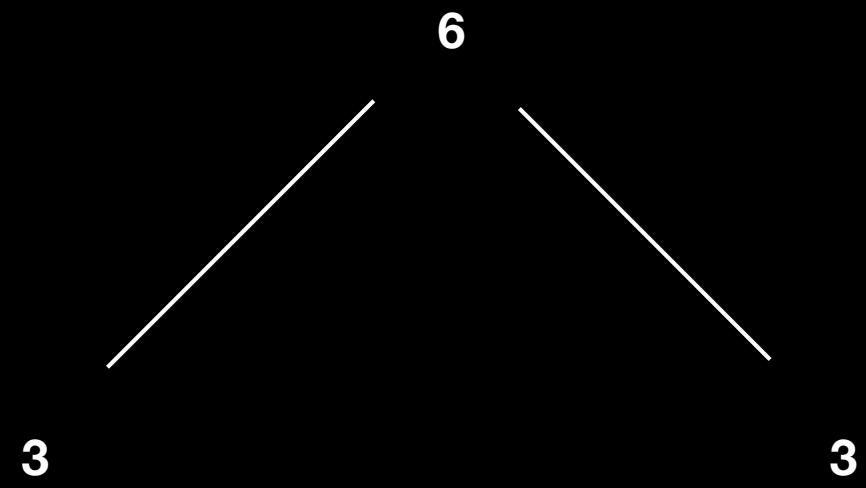
Les arbres binaires

Dorian.H Mekni

26 | MAR | 2020

Les arbres binaires



Les arbres binaires

- Concept née des sciences mathématiques
- Les arbres comme les listes chaînées servent à mémoriser des données. Ils sont constitués d'éléments que l'on appelle souvent des nœuds (node).

arbresBinaires.h-> Header

```c
#ifndef arbresBinaires_h
#define arbresBinaires_h

#include <stdio.h>
#include <stdlib.h>

typedef struct Tree
{
    int element;
    // Binaire donc deux branches
    struct Tree *tleft;
    struct Tree *tright;
    // Binaire donc chaque branche aura une descendance de données
    // Utile pour tout retour en arrière.
    struct Tree *parent;
}Tree;

// Prototypes

Tree *new_tree(int x);
Tree *join_tree(Tree *left, Tree *right, int node);
void clean_tree(Tree *tr);
void print_tree_prefix(Tree *tr);
int count_tree_nodes(Tree *tr);


#endif /* arbresBinaires_h */

```

*new_tree()

Cette fonction initialise un arbre binaire.

```
4
5  Tree *new_tree(int x)
6  {
7      // Allocation dynamique
8      Tree *tr = malloc(sizeof(*tr));
9
10     if(tr == NULL)
11     {
12         fprintf(stderr, "Error in dynamic allocation.\n");
13         exit(EXIT_FAILURE);
14     }
15
16     tr->element = x;
17     tr->tleft = NULL;
18     tr->tright = NULL;
19     tr->parent = NULL;
20
21     // An extra inch of help here
22     printf("We just created %d\n", tr->element);
23
24     return tr;
25 }
```

*join_tree()

Elle permet de joindre deux arbres à un parent

```
43
44 Tree *join_tree(Tree *left, Tree *right, int node)
45 {
46     Tree *tr = new_tree(node);
47
48     tr->tleft = left;
49     tr->tright = right;
50
51     if(left != NULL)
52         left->parent = tr;
53
54     if(right != NULL)
55         right->parent = tr;
56
57         return tr;
58 }
59
```

count_tree_nodes()

Elle compte combien de noeuds l'arbre compte

```
82
83  int count_tree_nodes(Tree *tr)
84  {
85      if(tr == NULL)
86          return 0;
87
88      return (count_tree_nodes(tr->tleft) + count_tree_nodes(tr->tright) +1);
89      // Le 1 étant la racine
90  }
91
```

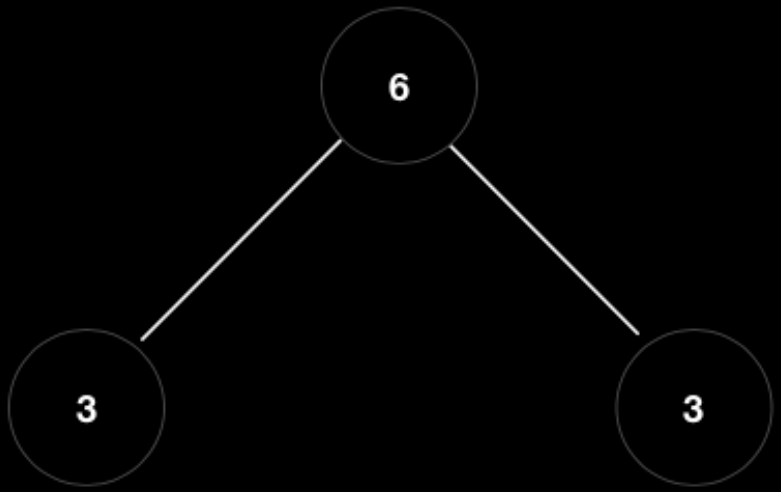
Test de Compilation 1

```c
#include <stdio.h>
#include <stdlib.h>
#include "arbresBinaires.h"

int main(void)
{
    Tree *root = join_tree(new_tree(3), new_tree(3), 6);

    clean_tree(root);



    return 0;
}

```

```
We just created 3
We just created 3
We just created 6
Delete 6
Delete 3
Delete 3
Program ended with exit code: 0
```

Test de Compilation 1

1er exemple avec notre arbre sur schéma représenté(voir post précédent) via nos fonctions sur notre console. Les enfants sont crées d'abord puis notre code remonte à la racine de l'arbre : 6.




```
We just created 3
We just created 3
We just created 6
Delete 6
Delete 3
Delete 3
Program ended with exit code: 0
```

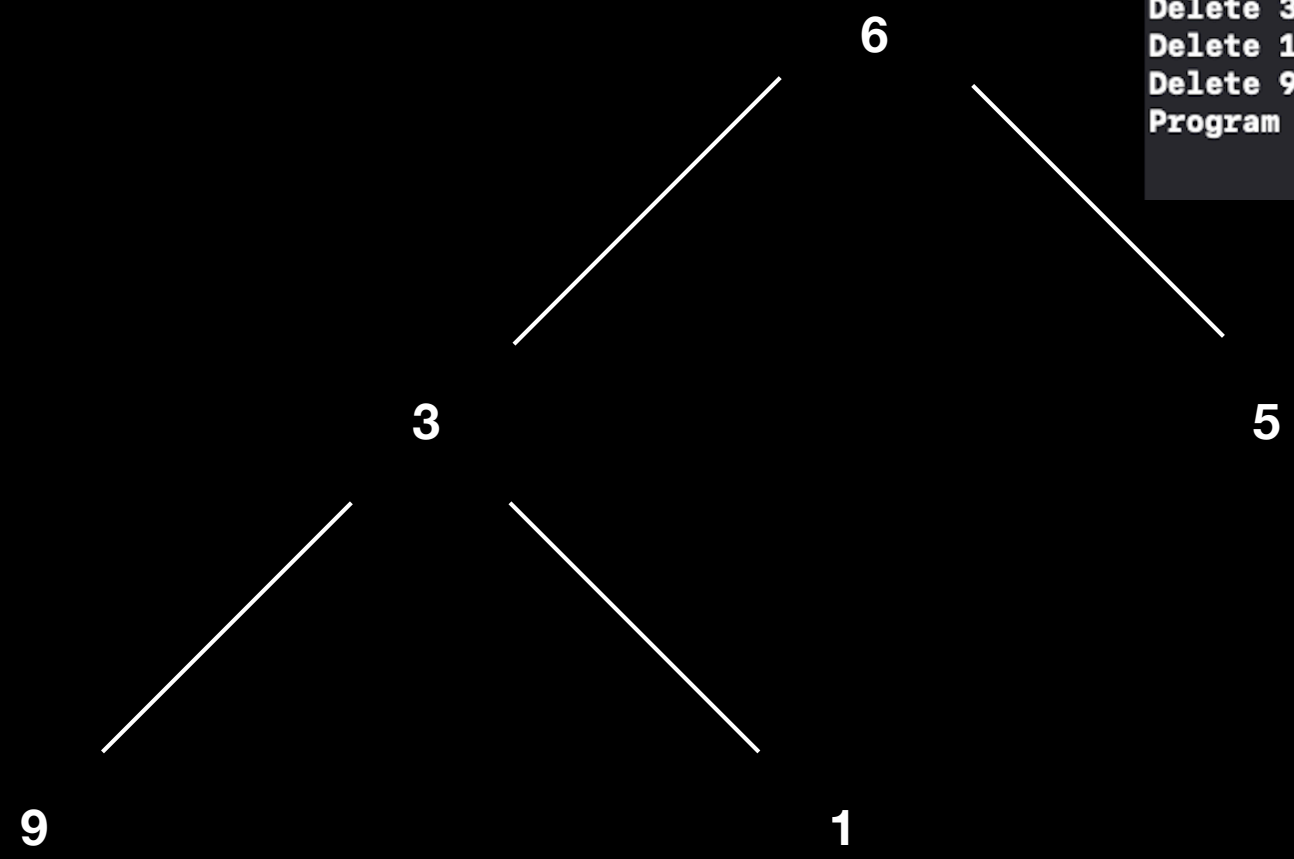
Test de Compilation 2

```c
#include <stdio.h>
#include <stdlib.h>
#include "arbresBinaires.h"

int main(void)
{
    Tree *root = (join_tree(join_tree(new_tree(9), new_tree(1), 3), new_tree(5), 6));

    print_tree_prefix(root);

    clean_tree(root);



    return 0;
}

```

```
We just created 9
We just created 1
We just created 3
We just created 5
We just created 6
(6).
(6) -> 3
(3) -> 9
(3) -> 1
(6) -> 5
Delete 6
Delete 5
Delete 3
Delete 1
Delete 9
Program ended with exit code: 0
```

Test de Compilation 2

```
We just created 9
We just created 1
We just created 3
We just created 5
We just created 6
(6).
(6) -> 3
(3) -> 9
(3) -> 1
(6) -> 5
Delete 6
Delete 5
Delete 3
Delete 1
Delete 9
Program ended with exit code: 0
```



Test de Compilation 3

```c
#include <stdio.h>
#include <stdlib.h>
#include "arbresBinaires.h"

int main(void)
{
    Tree *root = (join_tree(join_tree(new_tree(9), new_tree(1), 3), new_tree(5), 6));

    print_tree_prefix(root);
    printf("The binary tree contains %d nodes in total.\n", count_tree_nodes(root));
    clean_tree(root);

    return 0;
}
```

```
We just created 9
We just created 1
We just created 3
We just created 5
We just created 6
(6).
(6) -> 3
(3) -> 9
(3) -> 1
(6) -> 5
The binary tree contains 5 nodes in total.
Delete 6
Delete 5
Delete 3
Delete 1
Delete 9
Program ended with exit code: 0
```

Test de Compilation 3

```
We just created 9
We just created 1
We just created 3
We just created 5
We just created 6
(6).
(6) -> 3
(3) -> 9
(3) -> 1
(6) -> 5
The binary tree contains 5 nodes in total.
Delete 6
Delete 5
Delete 3
Delete 1
Delete 9
Program ended with exit code: 0
```