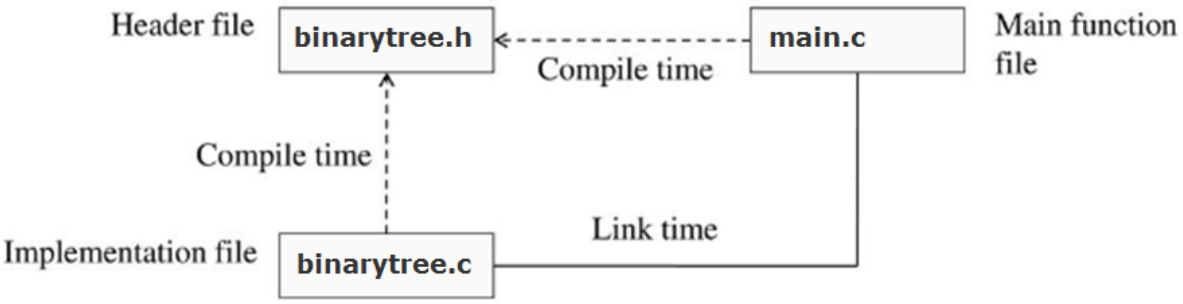
Practical work : C Program to Construct a B Tree

Objectif : The objective of this TP is to build a c program to construct a B Tree . the following figure shows the **Structure of the program.** The file binarytree.c and the file main.c can compiled in separate steps, by diferent people, in diffrenet organization. They each relay on the interface in binarytree.h.



In order to do so follow the steps below **:**

Step 01 :

Create a file (binarytree.h) which contains the definition of the structure and the tree manipulation functions:

```
1. typedef char DATA;
2. struct node
3. {
4.  DATA d;
5.  struct node *left;
6.  struct node *right;
7. };
8.
9. typedef struct node NODE;
10. typedef NODE *BTREE;
11.
12. BTREE newnode(void);
13. BTREE init_node(DATA d, BTREE p1, BTREE p2);
14. BTREE create_tree(DATA a[], int i, int size);
15. void preorder (BTREE root);
16. void inorder (BTREE root);
17. void postorder (BTREE root);
```

Step 02:

- Create a file (binarytree.c) which contains the implementation of the tree handling functions.
- Add the code of *Inorder* and postorder binary tree traversal

```
1. #include <assert.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include "binarytree.h"
5.
6. BTREE new_node()
7. {
8.     return ((BTREE)malloc(sizeof(NODE)));
9. }
10.
11. BTREE init_node(DATA d1, BTREE p1, BTREE p2)
12. {
13.     BTREE t;
14.
15.     t = new_node();
16.     t->d = d1;
17.     t->left = p1;
18.     t->right = p2;
19.     return t;
20. }
21.
22. /* create a linked binary tree from an array */
23. BTREE create_tree(DATA a[], int i, int size)
24. {
25.     if (i >= size)
26.         return NULL;
27.     else
28.         return(init_node(a[i],
29.     create_tree(a, 2*i+1, size),
30.     create_tree(a, 2*i+2, size)));
31. }
32.
33. /* preorder traversal */
34. void preorder (BTREE root)
35. {
36. }
37.
38. /* Inorder traversal */
39. void inorder (BTREE root)
40. {
41. …….
42. }
43.
44. /* postorder binary tree traversal */
```

```
45.
46. void postorder (BTREE root)
47. {
48.
49. }
```

Step 03:

Creating the main file (main.c) for the application

```
1. int main(void)
2. {
3.     char a[ARRAY_SIZE] = {'g','d','i','b','f','h','j','a','c','e'};
4.     BTREE root;
5.
6.     root = create_tree(a, 0, ARRAY_SIZE) ;
7.     assert(root != NULL);
8.     printf("PREORDER\n");
9.     preorder(root);
10.     printf("\n");
11.     printf("INORDER\n");
12.     inorder(root);
13.     printf("\n");
14.
15.     printf("POSTORDER\n");
16.     postorder(root);
17.     printf("\n");
18. }
```

The result of our program:

```
PREORDER
g d b a c f e i h j
INORDER
a b c d e f g h i j
POSTORDER
a c b e f d h j i g
```