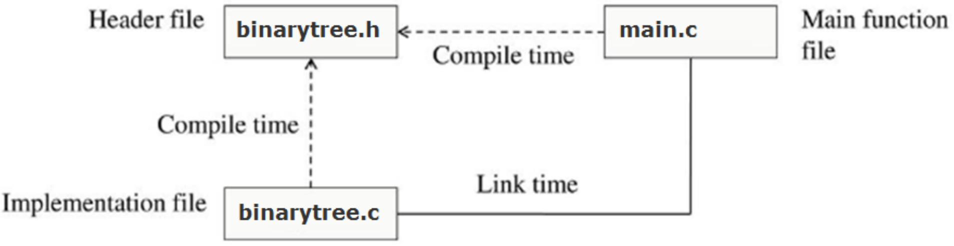
IBN KHALDOUN University –TIARET- Faculty of Mathematics and Computer Science
Department of Computer Science, Academic Year 2019/2020.
Course: Advanced Data Structures and Algorithm

Practical work : C Program to Construct a B Tree

Objective: The objective of this practical work is to build a C program to construct a B Tree. the following figure shows the *program structure*. The file **binarytree.c** and the file **main.c** can compiled in separate steps, by different people, in different organization. They each relay on the interface in **binarytree.h**.



In order to build this program follow the steps below:

Step 01:

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

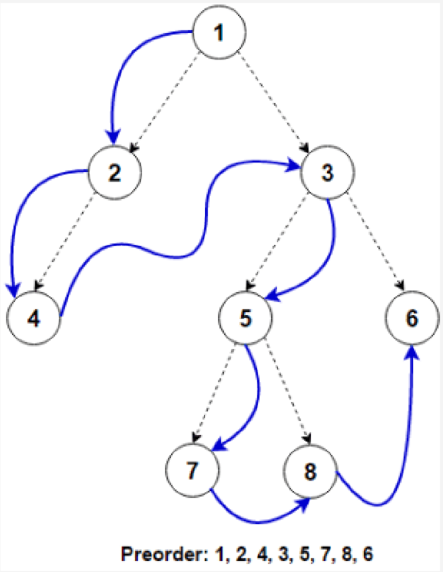
```c
typedef char DATA;
struct node
{
 DATA d;
 struct node *left;
 struct node *right;
};

typedef struct node NODE;
typedef NODE *BTREE;

BTREE newnode(void);
BTREE init_node(DATA d, BTREE p1, BTREE p2);
BTREE create_tree(DATA a[], int i, int size);
void preorder (BTREE root);
void inorder (BTREE root);
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

```c
#include <assert.h>
```

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

BTREE new_node()
{
    return ((BTREE)malloc(sizeof(NODE)));
}

BTREE init_node(DATA d1, BTREE p1, BTREE p2)
{
     BTREE t;

     t = new_node();
     t->d = d1;
     t->left = p1;
     t->right = p2;
     return t;
}

/* create a linked binary tree from an array */
BTREE create_tree(DATA a[], int i, int size)
{
     if (i >= size)
         return NULL;
     else
         return(init_node(a[i],
     create_tree(a, 2*i+1, size),
     create_tree(a, 2*i+2, size)));
}

/* preorder traversal */
void preorder (BTREE root)
{
}
```



Preorder: 1, 2, 4, 3, 5, 7, 8, 6

Step 03: Creating the main file (main.c) for the application

```c
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include " binarytree.h"
#define  ARRAY_SIZE 10
int main(void)
{  char a[ARRAY_SIZE] = {'g','d','i','b','f','h','j','a','c','e'};
    BTREE root;
    root = create_tree(a, 0, ARRAY_SIZE) ;
     assert(root != NULL);
     printf("PREORDER\n");
     preorder(root);}
```

```
/* preorder binary tree traversal
*/

void preorder (BTREE root)
{
   if (root != NULL) {
      printf("%c ", root->d);
      preorder(root -> left);
      preorder(root -> right);
   }
}

/* Inorder binary tree traversal
*/

void inorder (BTREE root)
{
   if (root != NULL) {
      inorder(root -> left);
      printf("%c ", root->d);
      inorder(root -> right);
   }
}

/* postorder binary tree traversal
*/

void postorder (BTREE root)
{
   if (root != NULL) {
      postorder(root -> left);
      postorder(root -> right);
      printf("%c ", root->d);
   }
}
```

```
// main.c

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

#include "binary.h"

#define  ARRAY_SIZE 10
int main(void)
{
   char a[ARRAY_SIZE] =
{'g','d','i','b','f','h','j','a','
c','e'};
   BTREE root;

   root = create_tree(a, 0,
ARRAY_SIZE) ;
   assert(root != NULL);

   printf("PREORDER\n");
   preorder(root);
   printf("\n");

   printf("INORDER\n");
   inorder(root);
   printf("\n");

   printf("POSTORDER\n");
   postorder(root);
   printf("\n");
}
```

```
==================================
==

makefile
--------

OBJS = main.o binary.o

list : $(OBJS)
      g++ -o binary $(OBJS)

main.o: main.c binary.h
      g++ -c main.c

binary.o: binary.c binary.h
      g++ -c binary.c

==================================
==

output
------

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
```