UNIVERSITÉ ANTONINE
Faculté d'ingénieurs en Informatique,
Multimédia, Réseaux & Télécommunications

Projet – Arbre Binaire

Matière: C et C++ avancée

Effectué par:	**NOM Prénom**	**INF#**	**Option**
	MATTA Elie et al.	Privacy applied	OGL

Table of Contents

I- Introduction

En informatique, un arbre binaire est une structure de données qui peut se représenter sous la forme d'une hiérarchie dont chaque élément est appelé nœud, le nœud initial étant appelé racine. Dans un arbre binaire, chaque élément possède au plus deux éléments fils au niveau inférieur, habituellement appelés gauche et droit. Du point de vue de ces éléments fils, l'élément dont ils sont issus au niveau supérieur est appelé père.

II- Les parties

On va diviser notre projet en trois parties :

- **Arbre.h** : Header file, contient l'initialisation des fonctions
- **Arbrebin.cpp** : Source file, contient le code et lest fonctions
- **Main.cpp** : Source file, contient le menu principale de l'arbre binaire

III- Code

Tout d'abord on commence par définir chaque partie du projet tout en expliquant chaque instruction par des commentaires

1) Arbre.h

```
#include <iostream>
using namespace std;

struct Arbre { //Declaration de la structure de l'arbre
int Noeud;     //L'entier a entrer
Arbre * FG;    //Fils gauche
Arbre * FD;    //Fils droit
}; //fin struct Arbre

class ArbreBinaire {

public:

ArbreBinaire(); //Constructeur vide
void AfficherArbreCroissant(Arbre *); //Affichage de l'arbre en ordre
croissant
Arbre *CreerNoeud(Arbre *,int); //Pour initialiser l'arbre et/ou creer un
nouveau element
```



```
void AfficherArbre(Arbre *); //Affichage de l'arbre souvent l'ordre de la
creation des entiers
void AfficherArbreDecroissant(Arbre *); //Affichage de l'arbre en ordre
decroissant
int Somme(Arbre *); //Fait la somme des elements de l'arbre
int CompteNoeud(Arbre *); //Compte le nombre de noeud tout en allant du fils
gauche au fils droit
Arbre *RechercheNoeud(Arbre *, int); //Recher le noeud demande et le retourne
(de type Arbre)
int hauteur (Arbre *); //la hauteur de l'arbre
Arbre *SuppNoeud(Arbre *, int); //Pour supprimer un noeud tout en remplacant
les valeurs convenables
}; //fin class ArbreBinaire
```

2) Arbrebin.cpp

```
#include "arbre.h" //pour utiliser les fonctions membres de arbre.h

ArbreBinaire::ArbreBinaire(){
}

Arbre *ArbreBinaire::CreerNoeud(Arbre *Racine,int valeur){
     if (Racine!=NULL){
              if (Racine->Noeud > valeur) //si la valeur a ajoute est plus
petite donc on la met
             {                            //au cote gauche de la racine
                    Racine->FG=CreerNoeud(Racine->FG,valeur);
              }
              else if (Racine->Noeud < valeur){
                    Racine->FD=CreerNoeud(Racine->FD,valeur);
              }
              else //si la valeur a ajoute n'est ni plus grande, ni plus
petite donc elle est egale
             {    //d'ou l'ambiguite et l'affichage d'une erreur est soumise
                    cout<<" ** Erreur: Valeur deja ajoute, SVP choisissez un
autre entier"<<endl;
              }
       }
       else //cette etape est parcourue toujours dans le cas ou on ajoute le
premier element dans l'arbre
      {    //ou dans le cas ou on veut ajoute un element, car ce nouveau
element dois avoir aussi un FD et FG
              Racine=new Arbre;
              Racine->Noeud=valeur;
              Racine->FD=NULL;
              Racine->FG=NULL;
       }
       return Racine; //on retourne tout le contenu de la racine (Noeud (qui
est la valeur), FD et FG)
}
void ArbreBinaire::AfficherArbre(Arbre *Racine){
```



```
      if (Racine!=NULL){ //si la Racine est devenue NULL donc on est arrive a
la fin de l'arbre
            cout<<" || "<< Racine->Noeud <<" || "<<endl;
            if (Racine->FD!=NULL || Racine->FG!=NULL){
                  AfficherArbre(Racine->FG); //on traverse vers la gauche
(valeurs inferieures), si il existe
                  AfficherArbre(Racine->FD); //et traverse vers la droite
(valeurs superieurs)
            }     }}

void ArbreBinaire::AfficherArbreCroissant(Arbre *Racine){
            if (Racine!=NULL){
             AfficherArbreCroissant(Racine->FG); //on traverse vers la gauche
(valeurs inferieures),
             cout<<" || "<< Racine->Noeud <<" || "; //si il existe car c'est
en ordre croissant
             AfficherArbreCroissant(Racine->FD); //quand on finit du fils
gauche, alors on comment par le fils droit
            }
}

void ArbreBinaire::AfficherArbreDecroissant(Arbre *Racine){
      if (Racine!=NULL){
            AfficherArbreDecroissant(Racine->FD); //on traverse vers la
droite au debut car c'est la valeure la plus grande
            cout<<" || "<< Racine->Noeud <<" || ";
            AfficherArbreDecroissant(Racine->FG);
            }
}

int ArbreBinaire::hauteur (Arbre *Racine){
      int hg=0; //hg = hauteur gauche
      int hd=0; //hd = hauteur droite
      if (Racine == NULL) //si elle NULL donc on considere sa hauteur comme
etant 0
            return 0;
      hg=hauteur(Racine ->FG); //hauteur de la part du fils gauche
      hd=hauteur(Racine ->FD); //hauteur de la part du fils droit
      if (hg>hd)
            return (hg+1); //si hg>hd on augmente la hauteur de la part du
fils gauche
      return (hd+1); //sinon on augment la hauteur du fils droit
}

int ArbreBinaire::Somme(Arbre *Racine){
int s=0;
if (Racine!=NULL){
s=Somme(Racine->FG); //on parcourt l'arbre du cote gauche au cote droite
s+=Racine->Noeud; //tout en ajoutant les valeurs
s+=Somme(Racine->FD);
}
return s;
}

int ArbreBinaire::CompteNoeud(Arbre *Racine){
int s=0;
if (Racine!=NULL){
```



```
s=CompteNoeud(Racine->FG); //le nombre de noeud du cote gauche
s++;
s+=CompteNoeud(Racine->FD); //puis le nombre de noeud au cote droit
}
return s;
}

Arbre *ArbreBinaire::RechercheNoeud(Arbre *Racine, int valeur){
      if (Racine!=NULL){
            if (Racine->Noeud > valeur){ //si la valeur a recherche est plus
petite donc on va au sens gauche
                  Racine=RechercheNoeud(Racine->FG,valeur);
            }
            else if (Racine->Noeud < valeur){ //sinon on va a droit
                  Racine=RechercheNoeud(Racine->FD,valeur);
            }
      }
      return Racine; //sinon ca veut dire c'est NULL ou bien c'est trouve, le
test est fait dans main
}

Arbre *ArbreBinaire::SuppNoeud(Arbre *Racine, int valeur){
      Arbre * NoeudASupprimer;
      if (Racine->Noeud==valeur){ // on a trouvé l'element a supprimer
            NoeudASupprimer=Racine;
            if (NoeudASupprimer->FG==NULL) //si il n'ya pa de FG, on retourne
FD
                  return NoeudASupprimer->FD;
            else{
                  Racine=NoeudASupprimer->FG; //sinon on recherche dans FG
l'endroit pour inserer le FD
                  while (Racine->FD!=NULL){ //tant que la Racine a un fils
droite on continue dans le meme sens
                        Racine=Racine->FD;
                  }
                  Racine->FD=NoeudASupprimer->FD; //le fils droit prend la
place du dernier fils droit de la racine qu'on souhaite supprime
                  return NoeudASupprimer->FG;
            }
            delete NoeudASupprimer;
      }
      else{
            if (Racine->Noeud > valeur){ //si la valeur a supprime est plus
petite que le noeud on va a gauche
                  Racine->FG=SuppNoeud(Racine->FG,valeur);
            }
            else
            {
                  Racine->FD=SuppNoeud(Racine->FD,valeur); //sinon on va a
droite
            }
      }
      return Racine;
}
```


3) Main.cpp

```
#include "arbre.h"

void main(){
int valeur;
ArbreBinaire *ab=new ArbreBinaire(); //creation d'un constructeur vide pour
acceder au methodes de la classe
Arbre *Racine;
Arbre *RepRecherche; //pour la fonction RechercheNoeud
Racine=NULL;
int Choix;
bool exit=false;

cout<<"    ASCII code ici mais on la pas imprimer car il est trop long \n";


while (!exit){

      cout<<"*******************************************************\n";
      cout<<"* Menu Principal : Manipulation d'une arbre binaire *\n";
      cout<<"* Project C et C++ avancee                          *\n";
      cout<<"* presente par Elie Matta et al.                    *\n";
      cout<<"*Copyright 2009, eliematta.com. All rights reserved  *\n";
      cout<<"*******************************************************\n";
      cout<<"I 1- Ajouter un noeud                                 I\n";
      cout<<"I 2- Afficher l'arbre                                 I\n";
      cout<<"I 3- Afficher l'arbre dans l'ordre croissant          I\n";
      cout<<"I 4- Afficher l'arbre dans l'ordre decroissant        I\n";
      cout<<"I 5- Chercher la hauteur                              I\n";
      cout<<"I 6- Somme des noeuds                                 I\n";
      cout<<"I 7- Nombre de noeuds                                 I\n";
      cout<<"I 8- Rechercher un noeud                              I\n";
      cout<<"I 9- Enlever un noeud                                 I\n";
      cout<<"I 10- Quitter                                         I\n";
      cout<<"I*****************************************************I\n";
      cout<<"\n\n\nChoix: ";
      cin >> Choix;
      cout<<endl; //pour retourner a la ligne

      while ((Choix!=10) && (Racine==NULL) && (Choix>1) && (Choix <10)){
            cout<<" ** Vous devez d'abord remplir l'arbre, choisissez un
choix entre 1 et 10 **\n\n";
            cout<<"Choix: ";
            cin>>Choix;
            cout<<endl; //pour retourner a la ligne
      }
            switch (Choix) {
case 1 : cout<<" Saisir un entier pour l'inserer dans l'arbre (-1 pour finir)
: ";
      cin>>valeur;
      while (valeur != -1){
            Racine=ab->CreerNoeud(Racine,valeur);
```



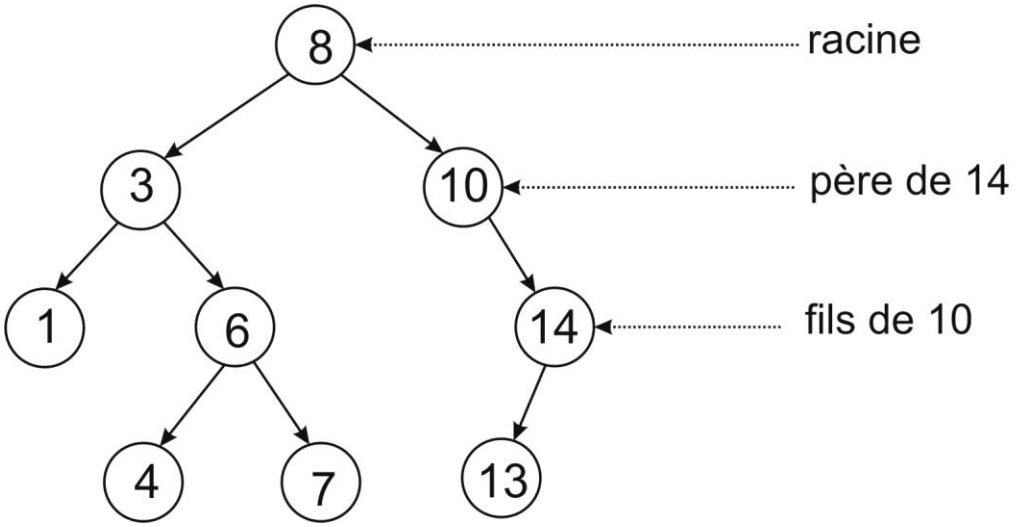
```
            cout<<" Saisir un entier pour l'inserer dans l'arbre (-1 pour
finir) : ";
            cin>>valeur;
      }
      break;

case 2 : ab->AfficherArbre(Racine);
      cout<<" \nAffichage de l'arbre est complet \n\n";
      break;

case 3 : ab->AfficherArbreCroissant(Racine);
      cout<<" \nAffichage de l'arbre en ordre croissant est complet \n\n";
      break;

case 4 : ab->AfficherArbreDecroissant(Racine);
      cout<<" \nAffichage de l'arbre en ordre decroissant est complet \n\n";
      break;

case 5 : ab->AfficherArbreCroissant(Racine);
      cout<<"\n La hauteur est: "<<ab->hauteur(Racine)<<"\n\n";
      break;

case 6 : ab->AfficherArbreCroissant(Racine);
      cout<<"\n La somme des noeuds est egale a "<<ab->Somme(Racine);
      cout<<"\n\n";
      break;

case 7 : ab->AfficherArbreCroissant(Racine);
      cout<<"\n Le nombre des noeuds est egale a "<<ab->CompteNoeud(Racine);
      cout<<"\n\n";
      break;

case 8 : cout<<" Saisir le nombre de noeud a rechercher : ";
      cin>>valeur;
      RepRecherche=ab->RechercheNoeud(Racine,valeur);
      if (RepRecherche==NULL){
            cout<<"\n L'entier a recherche n'est pas dans l'arbre\n\n";
            break;
      }
      if (RepRecherche->Noeud==valeur) {
            cout<<"\n L'entier est retrouve : "<<RepRecherche->Noeud<<endl;
      }
      break;

case 9 : cout<<" Choisissez la valeur du noeud a supprimer : \n";
      cin>>valeur;
      Racine=ab->SuppNoeud(Racine,valeur);
      cout<<"Supprimassion du noeud "<<valeur<<" acheve\n\n";
      break;

case 10 :
      cout<<" Merci pour votre temps! "<<endl;
      exit=true;
      break;

default : cout<<" *** Choix invalid, le choix "<<Choix<<" n'est pas
disponible ***\n\n";
```

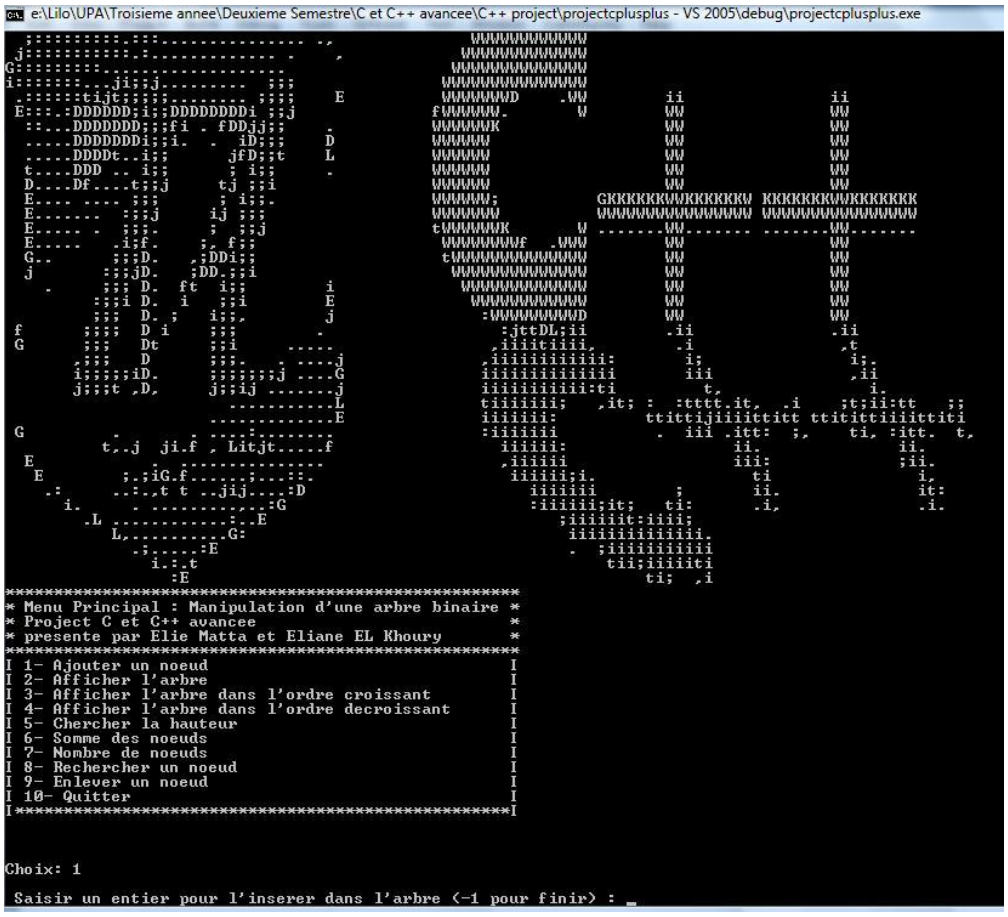


```
} } }
```


IV- Exemple

On va exécuter notre programme et utiliser chaque choix du menu principal tout en exécutant cet exemple :





Voyons l'erreur générer lorsqu'on choisit un choix impossible comme l'affichage de l'arbre quand il n'y a aucune valeur insérer dans l'arbre

```
Choix: 2

 ** Vous devez d'abord remplir l'arbre, choisissez un choix entre 1 et 10 **
```

1) Choix 1

Alors on a choisis le choix 1 pour ajouter les nœuds arbitrairement pour avoir :

```
Choix: 1

 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : 8
 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : 3
 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : 10
 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : 1
 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : 13
 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : 7
 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : 4
 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : 6
 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : 14
 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : 8
 ** Erreur: Valeur deja ajoute, SVP choisissez un autre entier
 Saisir un entier pour l'inserer dans l'arbre (-1 pour finir) : -1
```

On a insérer tout les valeurs et on peut y voir l'erreur lorsque le même élément est insérer plus qu'une fois

Voyons par suite les choix 2, 3 et 4 pour afficher l'arbre binaire respectivement par ordre d'insertion des éléments, croissant et décroissant

2) Choix 2

```
Choix: 2

  || 8 ||
  || 3 ||
  || 1 ||
  || 7 ||
  || 4 ||
  || 6 ||
  || 10 ||
  || 13 ||
  || 14 ||

Affichage de l'arbre est complet
```

3) Choix 3

```
Choix: 3

  || 1 ||   || 3 ||   || 4 ||   || 6 ||   || 7 ||   || 8 ||   || 10 ||   || 13 ||   || 14 ||
Affichage de l'arbre en ordre croissant est complet
```

4) Choix 4


```
Choix: 4

 || 14 ||  || 13 ||  || 10 ||  || 8 ||  || 7 ||  || 6 ||  || 4 ||  || 3 ||  || 1 ||
Affichage de l'arbre en ordre decroissant est complet
```

5) Choix 5
   Pour chercher la hauteur :

```
Choix: 5

 || 1 ||  || 3 ||  || 4 ||  || 6 ||  || 7 ||  || 8 ||  || 10 ||  || 13 ||  || 14 ||
 La hauteur est: 5
```

6) Choix 6
   Pour faire la somme des nœuds avec les fils gauche et droite :

```
Choix: 6

 || 1 ||  || 3 ||  || 4 ||  || 6 ||  || 7 ||  || 8 ||  || 10 ||  || 13 ||  || 14 ||
 La somme des noeuds est egale a 66
```

7) Choix 7
   Pour calculer les nombres de nœuds :

```
Choix: 7

 || 1 ||  || 3 ||  || 4 ||  || 6 ||  || 7 ||  || 8 ||  || 10 ||  || 13 ||  || 14 ||
 Le nombre des noeuds est egale a 9
```

8) Choix 8
   Pour rechercher un nœud :

```
Choix: 8

 Saisir le nombre de noeud a rechercher : 3

 L'entier est retrouve : 3
```

NB : Si on a choisi de rechercher un nombre qui n'existe pas dans l'arbre on aura l'erreur suivante

```
Choix: 8

 Saisir le nombre de noeud a rechercher : 30

 L'entier a recherche n'est pas dans l'arbre
```

9) Choix 9
   Pour supprimer un nœud et réarrangé tout l'arbre de nouveau

```
Choix: 9

 Choisissez la valeur du noeud a supprimer :
3
Supprimassion du noeud 3 acheve
```

10) Choix 10
    Pour quitter

