Les arbres binaires

- sont utiles :
  - algorithmes de recherche efficace
  - algorithme de compression de Huffmann
- constituent une première approche des arbres
- sont faciles à parcourir

Deux points de vue

- Le point de vue itératif
  - arbre constitué de sommets
  - chaque sommet pointe sur deux sommets du niveau inférieur
- Le point de vue récursif
  - arbre comportant un sommet
  - ce sommet pointe sur deux sous-arbres construits de la même manière

Quel que soit le point de vue

- L'utilisateur d'un objet arbre binaire n'a pas à savoir comment cet objet est implémenté
  - pour que son programme reste indépendant de l'implémentation
  - pour que celle-ci puisse être modifiée et optimisée

Fonctionnalités de base

- Transfert vers ou depuis un fichier
- Ajout d'une nouvelle information
- Recherche d'une information
- Affichage des informations de l'arbre
- Indications diverses sur son état :
  - nombre d'informations
  - niveau d'équilibre, etc.

Exemple de programme utilisateur


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

int neutre() {return 0;}
#include "arbreb.C"

int main(){
ArbreBinaire <int, int> z;
int x, y;

cout << "Donnez une suite de valeurs terminee par -1" << endl;
cin >> x;
while (x!=-1){
        z.ajout(x);
        cin >> x;
        }
cout << "Voici l'arbre : " << z << endl;
cout << "Donnez des valeurs a chercher dedans (-1 pour arreter): " << endl;
cin >> x;
while (x!=-1){
        y = z.recherche(x);
        cout << ((y)? "Trouve" : "Pas trouve")  << endl;
        cin >> x;
        }

return 0;
}
```

Une implémentation possible

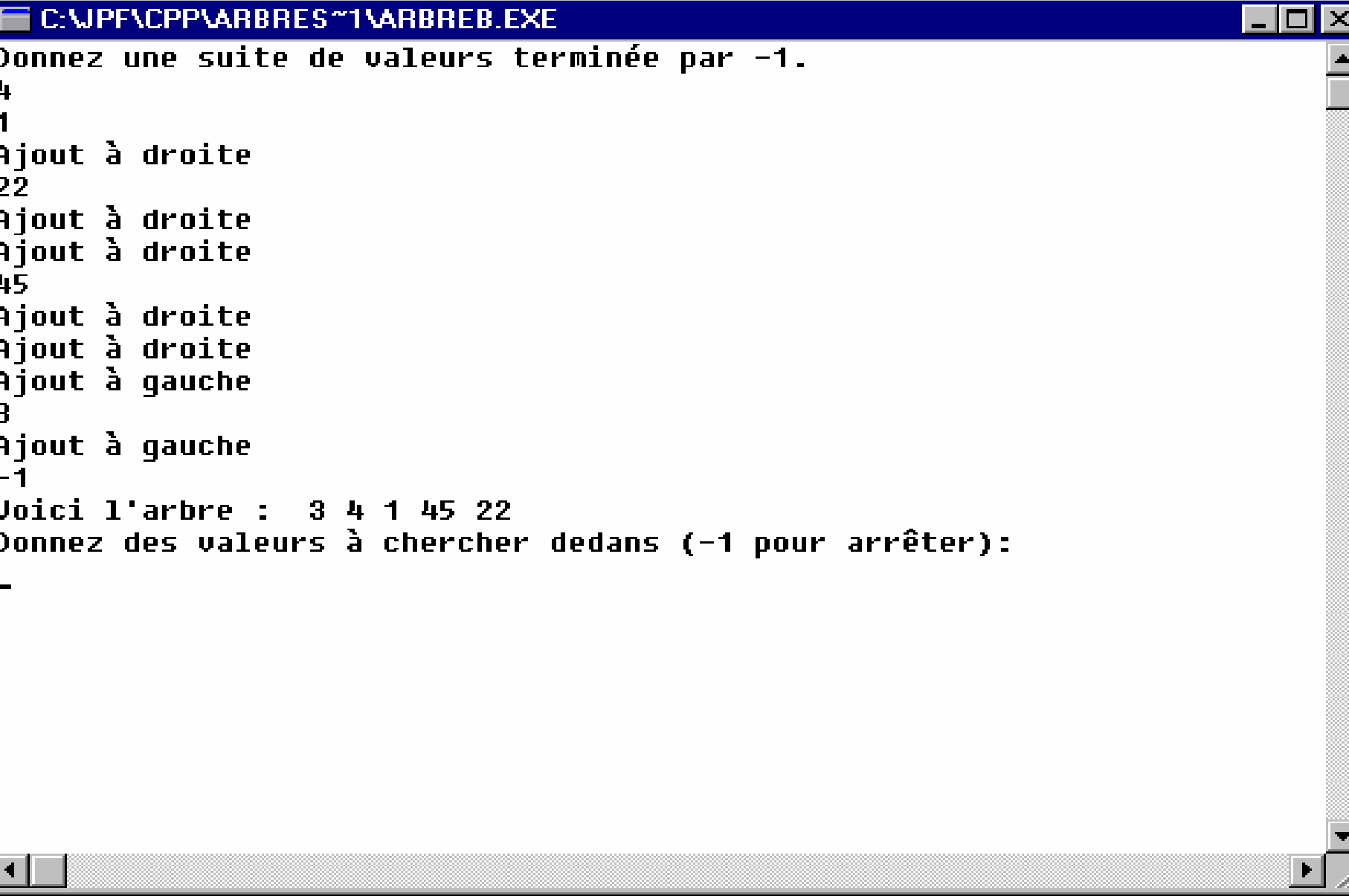

```
#ifndef ARBREB_H
#define ARBREB_H
#include <iostream>
using namespace std;

template <class Info, class Clef>
class ArbreBinaire {
protected :
   Info valeur;
   ArbreBinaire * fg;
   ArbreBinaire * fd;
private :
   ArbreBinaire(const ArbreBinaire &); // transmission par valeur
   interdite
```


```
public:
        ArbreBinaire(){
                valeur = neutre();
                fg = fd = NULL;
                }
        ~ArbreBinaire(){if (fd) delete fd; if (fg) delete fg;}
        virtual void ajout(const Info &);
        virtual const Info & recherche(const Clef &) const;
        friend ostream & operator << (ostream & arg1,
                const ArbreBinaire<Info, Clef> & arg2){
                // Parcours gauche - centre - droite
                if (arg2.fg) arg1 << *arg2.fg;
                arg1 << " " << arg2.valeur;
                if (arg2.fd) arg1 << *arg2.fd;
                return arg1;
                }
        };
#endif
```


```
template <class Info, class Clef>
void ArbreBinaire<Info, Clef>::ajout(const Info & arg){
        if (valeur == neutre())    {
                valeur = arg;
                return;
                }
        if (rand()% 2) {
                cout << "Ajout a gauche" << endl;
                if (!fg) fg = new ArbreBinaire;
                fg -> ajout(arg);
                }
                else {
                cout << "Ajout a droite" << endl;
                if (!fd) fd = new ArbreBinaire;
                fd -> ajout(arg);
                }
        return;
        }
```


```
emplate <class Info, class Clef>
onst Info & ArbreBinaire<Info, Clef>::recherche(const Clef & arg) const
      cout << "Entree dans la methode de recherche." << endl;
      if (valeur == arg) return valeur;
      if (fg) {
              const Info & infoTrouvee = fg -> recherche(arg);
              if (infoTrouvee) return infoTrouvee;
              }
      if (fd)
              return fd -> recherche(arg);
      return *new Info(neutre());
      }
```


C:\JPF\CPP\ARBRES~1\ARBREB.EXE
Donnez une suite de valeurs terminée par -1.
4
1
Ajout à droite
22
Ajout à droite
Ajout à droite
45
Ajout à droite
Ajout à droite
Ajout à gauche
3
Ajout à gauche
-1
Voici l'arbre : 3 4 1 45 22
Donnez des valeurs à chercher dedans (-1 pour arrêter):

C:\JPF\CPP\ARBRES~1\ARBREB.EXE

```
Donnez une suite de valeurs terminée par -1.
4
1
Ajout à droite
22
Ajout à droite
Ajout à droite
45
Ajout à droite
Ajout à droite
Ajout à gauche
3
Ajout à gauche
-1
Voici l'arbre :  3 4 1 45 22
Donnez des valeurs à chercher dedans (-1 pour arrêter):
45
Entrée dans la méthode de recherche.
Entrée dans la méthode de recherche.
Entrée dans la méthode de recherche.
Entrée dans la méthode de recherche.
Entrée dans la méthode de recherche.
Entrée dans la méthode de recherche.
Trouvé !
-1_
```