Département
Informatique

# Dossier de Réalisation

**TP-AC**
**Références Croisées**


Binôme 3213

Tristan Pourcelot
Jordan Vincent


3IF - Groupe 2

Année scolaire 2011-2012


**Institut National des Sciences Appliquées de Lyon**

Choix généraux

Listings du programme

```
1  /*************************************************************************
                           Fichier  -  description
3                            -------------------
     debut                : Nov. 2011
5    copyright            : (C) 2011 par Tristan Pourcelot & Jordan Vincent
*************************************************************************/
7
//-- Interface de la classe <Fichier> (fichier Fichier.h) --
9 #if ! defined ( FICHIER_H )
#define FICHIER_H
11
//------------------------------------------------------ Interfaces utilisees
13 #include <vector>
#include <string>
15 //------------------------------------------------------------ Constantes

17 //------------------------------------------------------------------ Types

19 //-------------------------------------------------------------------------
// Role de la classe <Fichier>
21 //
// La classe Fichier a pour role de representer les occurences d'un fichier
23 // pour un identificateur donne.
//
25 //-------------------------------------------------------------------------

27 class Fichier
{
29 //-------------------------------------------------------------- PUBLIC

31 public:
//----------------------------------------------------- Methodes publiques
33   void DisplayFichier() const;
   // Mode d'emploi : Affiche sur la sortie standard la liste des numeros
35   // de lignes associes au fichier.
   //
37
   bool operator==(Fichier const & unFichier) const;
39   // Mode d'emploi : renvoie vrai unFichier est egal ou this
   //
41   // Contrat : le test ne ce fait que sur l'attribut nomFic !
   //
43
   bool AddLigne( int numLigne );
45   // Mode d'emploi : Ajoute un numero de ligne au vecteur de numeros
   // de lignes. Incremente nbNumLignes.
47   // Renvoie faux si numero deja existant, vrai sinon.
   //
49   // Contrat : numLigne est strictement positif.
   //     les numeros sont supposes tries par ordre croissant
51   //     et numLigne est suppose etre plus grand que tous les
   //     numeros du vector
53

55 //------------------------------------------------- Constructeurs - destructeur

57   Fichier( const Fichier & unFichier );
   // Mode d'emploi (constructeur de copie) : construit une copie de
59   // unFichier
   // Contrat :
61
   Fichier ( string unNom );
63   // Mode d'emploi : constructeur
   // Initialise nomFic et met numLignes a NULL
65
     virtual ~Fichier ( );
67   // Mode d'emploi (destructeur) :
```

```
69 //------------------------------------------------------------------ PRIVE
   protected:
71 //----------------------------------------------------- Methodes protegees

73 //----------------------------------------------------- Attributs proteges

75   string nomFic;
     // nom du fichier
77
     vector <int> *numLignes;
79   // Pointeur sur un tableau dynamique de numeros de lignes tries
     // par ordre  croissant.
81 };

83 //--------------------------- Autres definitions dependantes de <Fichier>

85 #endif // FICHIER_H
```

../src/Fichier.h

```
1  /*************************************************************************
                           Fichier  -  description
3                            -------------------
       debut                : Nov. 2011
5      copyright            : (C) 2011 par Tristan Pourcelot & Jordan Vincent
   *************************************************************************/
7
   //-- Realisation de la classe <Fichier> (fichier Fichier.cpp) ------
9
   //------------------------------------------------------------------ INCLUDE
11
   //----------------------------------------------------- Include systeme
13 #include <iostream>
   #include <algorithm>
15 using namespace std;

17 //------------------------------------------------------ Include personnel
   #include "Fichier.h"
19
   //------------------------------------------------------------ Constantes
21
   //------------------------------------------------------------------ PUBLIC
23
   //----------------------------------------------------- Methodes publiques
25 void Fichier::DisplayFichier() const
   // Algorithme : Parcourt de numLignes et affichage des numeros de ligne
27 {
     cout << "\t" << nomFic;
29   for (int i=0; i<numLignes->size(); i++)
     {
31     cout << " " << numLignes->at(i);
     }
33 }

35 bool Fichier::AddLigne( int numLigne )
   // Algorithme : Ajoute une ligne si elle n'existe pas deja.
37 {
     if (numLignes == NULL)
39   {
       numLignes = new vector<int>;
41     numLignes->push_back(numLigne);

43     return 1;
     }
45   else if (! binary_search (numLignes->begin(), numLignes->end(), numLigne))

47   {
       numLignes->push_back(numLigne);
49     return 1;
```

```
  }
  else
  {
    return 0;
  }
} // ------ Fin de AddLigne

bool Fichier::operator==(Fichier const & unFichier) const
// Algorithme : Trivial
{
  if (nomFic == unFichier.nomFic) return true;
  else return false;
} // ------ Fin de operator==

//------------------------------------------------ Constructeurs - destructeur
Fichier::Fichier ( string unNom )
// Algorithme : Trivial
{
#if defined ( MAP )
  cout << "Appel du constructeur de Fichier" << endl;
#endif
  nomFic =    unNom;
  numLignes =   NULL;
} // ------ Fin de Fichier

Fichier::Fichier ( const Fichier & unFichier )
// Algorithme : Creation d'un nouveau vector et recopie des elements
{
#if defined ( MAP )
  cout << "Appel du constructeur de copie de Fichier" << endl;
#endif
  nomFic =    unFichier.nomFic;
  numLignes =   new vector<int>;
  *numLignes =  *unFichier.numLignes;
} // ------ Fin de Fichier (Constructeur de copie)

Fichier::~Fichier ( )
// Algorithme : Trivial
{
#if defined ( MAP )
  cout << "Appel du destructeur de Fichier" << endl;
#endif
  delete numLignes;
} // ------ Fin de ~Fichier

//------------------------------------------------------------------ PRIVE

//----------------------------------------------------- Methodes protegees
```

../src/Fichier.cpp

```
/*************************************************************************
                          Fichier  -  description
                             -------------------
    debut                : Nov. 2011
    copyright            : (C) 2011 par Tristan Pourcelot & Jordan Vincent
*************************************************************************/

//-- Realisation de la classe <Fichier> (fichier Fichier.cpp) -----

//------------------------------------------------------------ INCLUDE

//-------------------------------------------------- Include systeme
#include <iostream>
#include <algorithm>
using namespace std;

//-------------------------------------------------- Include personnel
#include "Fichier.h"

```

```
//------------------------------------------------------------------ Constantes

//------------------------------------------------------------------ PUBLIC

//------------------------------------------------------------ Methodes publiques
void Fichier::DisplayFichier() const
// Algorithme : Parcourt de numLignes et affichage des numeros de ligne
{
  cout << "\t" << nomFic;
  for (int i=0; i<numLignes->size(); i++)
  {
    cout << " " << numLignes->at(i);
  }
}

bool Fichier::AddLigne( int numLigne )
// Algorithme : Ajoute une ligne si elle n'existe pas deja.
{
  if (numLignes == NULL)
  {
    numLignes = new vector<int>;
    numLignes->push_back(numLigne);

    return 1;
  }
  else if (! binary_search (numLignes->begin(), numLignes->end(), numLigne))

  {
    numLignes->push_back(numLigne);
    return 1;
  }
  else
  {
    return 0;
  }
} // ----- Fin de AddLigne

bool Fichier::operator==(Fichier const & unFichier) const
// Algorithme : Trivial
{
  if (nomFic == unFichier.nomFic) return true;
  else return false;
} // ----- Fin de operator==

//-------------------------------------------- Constructeurs - destructeur
Fichier::Fichier ( string unNom )
// Algorithme : Trivial
{
#if defined ( MAP )
  cout << "Appel du constructeur de Fichier" << endl;
#endif
  nomFic =    unNom;
  numLignes =   NULL;
} // ----- Fin de Fichier

Fichier::Fichier ( const Fichier & unFichier )
// Algorithme : Creation d'un nouveau vector et recopie des elements
{
#if defined ( MAP )
  cout << "Appel du constructeur de copie de Fichier" << endl;
#endif
  nomFic =    unFichier.nomFic;
  numLignes =   new vector<int>;
  *numLignes =  *unFichier.numLignes;
} // ----- Fin de Fichier (Constructeur de copie)

Fichier::~Fichier ( )
// Algorithme : Trivial
{
#if defined ( MAP )
```

```
  cout << "Appel du destructeur de Fichier" << endl;
#endif
  delete numLignes;
} // ------ Fin de ~Fichier

//------------------------------------------------------------------ PRIVE

//------------------------------------------------------- Methodes protegees
```

../src/Fichier.cpp

```
/**************************************************************************
                          Fichier  -  description
                             -------------------
    debut                : Nov. 2011
    copyright            : (C) 2011 par Tristan Pourcelot & Jordan Vincent
**************************************************************************/

//-- Realisation de la classe <Fichier> (fichier Fichier.cpp) ------

//------------------------------------------------------------------ INCLUDE

//-------------------------------------------------------- Include systeme
#include <iostream>
#include <algorithm>
using namespace std;

//------------------------------------------------------ Include personnel
#include "Fichier.h"

//------------------------------------------------------------- Constantes

//------------------------------------------------------------------ PUBLIC

//------------------------------------------------------- Methodes publiques
void Fichier::DisplayFichier() const
// Algorithme : Parcourt de numLignes et affichage des numeros de ligne
{
  cout << "\t" << nomFic;
  for (int i=0; i<numLignes->size(); i++)
  {
    cout << " " << numLignes->at(i);
  }
}

bool Fichier::AddLigne( int numLigne )
// Algorithme : Ajoute une ligne si elle n'existe pas deja.
{
  if (numLignes == NULL)
  {
    numLignes = new vector<int>;
    numLignes->push_back(numLigne);

    return 1;
  }
  else if (! binary_search (numLignes->begin(), numLignes->end(), numLigne))

  {
    numLignes->push_back(numLigne);
    return 1;
  }
  else
  {
    return 0;
  }
} // ------ Fin de AddLigne

bool Fichier::operator==(Fichier const & unFichier) const
// Algorithme : Trivial
{
```

```
  if (nomFic == unFichier.nomFic) return true;
61  else return false;
} // ------ Fin de operator==
63
//------------------------------------------------- Constructeurs - destructeur
65 Fichier::Fichier ( string unNom )
// Algorithme : Trivial
67 {
#if defined ( MAP )
69  cout << "Appel du constructeur de Fichier" << endl;
#endif
71  nomFic =    unNom;
  numLignes =   NULL;
73 } // ------ Fin de Fichier

75 Fichier::Fichier ( const Fichier & unFichier )
// Algorithme : Creation d'un nouveau vector et recopie des elements
77 {
#if defined ( MAP )
79  cout << "Appel du constructeur de copie de Fichier" << endl;
#endif
81  nomFic =    unFichier.nomFic;
  numLignes =   new vector<int>;
83  *numLignes =  *unFichier.numLignes;
} // ------ Fin de Fichier (Constructeur de copie)
85
Fichier::~Fichier ( )
87 // Algorithme : Trivial
{
89 #if defined ( MAP )
  cout << "Appel du destructeur de Fichier" << endl;
91 #endif
  delete numLignes;
93 } // ------ Fin de ~Fichier

95 //------------------------------------------------------------------ PRIVE

97 //----------------------------------------------------------- Methodes protegees
```

../src/Fichier.cpp

```
1 /*************************************************************************
                          Fichier  -  description
3                             -------------------
    debut                : Nov. 2011
5   copyright            : (C) 2011 par Tristan Pourcelot & Jordan Vincent
*************************************************************************/
7
//-- Realisation de la classe <Fichier> (fichier Fichier.cpp) -----
9
//------------------------------------------------------------------ INCLUDE
11
//----------------------------------------------------- Include systeme
13 #include <iostream>
#include <algorithm>
15 using namespace std;

17 //----------------------------------------------------- Include personnel
#include "Fichier.h"
19
//------------------------------------------------------------ Constantes
21
//------------------------------------------------------------------ PUBLIC
23
//----------------------------------------------------- Methodes publiques
25 void Fichier::DisplayFichier() const
// Algorithme : Parcourt de numLignes et affichage des numeros de ligne
27 {
  cout << "\t" << nomFic;
29  for (int i=0; i<numLignes->size(); i++)
```

```
  {
31    cout << " " << numLignes->at(i);
  }
33 }

35 bool Fichier::AddLigne( int numLigne )
// Algorithme : Ajoute une ligne si elle n'existe pas deja.
37 {
  if (numLignes == NULL)
39  {
    numLignes = new vector<int>;
41    numLignes->push_back(numLigne);

43    return 1;
  }
45  else if (! binary_search (numLignes->begin(), numLignes->end(), numLigne))

47  {
    numLignes->push_back(numLigne);
49    return 1;
  }
51  else
  {
53    return 0;
  }
55 } // ------ Fin de AddLigne

57 bool Fichier::operator==(Fichier const & unFichier) const
// Algorithme : Trivial
59 {
  if (nomFic == unFichier.nomFic) return true;
61  else return false;
} // ------ Fin de operator==
63
//------------------------------------------------ Constructeurs - destructeur
65 Fichier::Fichier ( string unNom )
// Algorithme : Trivial
67 {
#if defined ( MAP )
69  cout << "Appel du constructeur de Fichier" << endl;
#endif
71  nomFic =    unNom;
  numLignes =   NULL;
73 } // ------ Fin de Fichier

75 Fichier::Fichier ( const Fichier & unFichier )
// Algorithme : Creation d'un nouveau vector et recopie des elements
77 {
#if defined ( MAP )
79  cout << "Appel du constructeur de copie de Fichier" << endl;
#endif
81  nomFic =    unFichier.nomFic;
  numLignes =   new vector<int>;
83  *numLignes =  *unFichier.numLignes;
} // ------ Fin de Fichier (Constructeur de copie)
85
Fichier::~Fichier ( )
87 // Algorithme : Trivial
{
89 #if defined ( MAP )
  cout << "Appel du destructeur de Fichier" << endl;
91 #endif
  delete numLignes;
93 } // ------ Fin de ~Fichier

95 //------------------------------------------------------------------ PRIVE

97 //----------------------------------------------------- Methodes protegees
```

../src/Fichier.cpp

```cpp
/*************************************************************************
                           Fichier  -  description
                             -------------------
    debut                : Nov. 2011
    copyright            : (C) 2011 par Tristan Pourcelot & Jordan Vincent
*************************************************************************/

//-- Realisation de la classe <Fichier> (fichier Fichier.cpp) ----

//---------------------------------------------------------------- INCLUDE

//-------------------------------------------------------- Include systeme
#include <iostream>
#include <algorithm>
using namespace std;

//------------------------------------------------------ Include personnel
#include "Fichier.h"

//------------------------------------------------------------ Constantes

//---------------------------------------------------------------- PUBLIC

//----------------------------------------------------- Methodes publiques
void Fichier::DisplayFichier() const
// Algorithme : Parcourt de numLignes et affichage des numeros de ligne
{
  cout << "\t" << nomFic;
  for (int i=0; i<numLignes->size(); i++)
  {
    cout << " " << numLignes->at(i);
  }
}

bool Fichier::AddLigne( int numLigne )
// Algorithme : Ajoute une ligne si elle n'existe pas deja.
{
  if (numLignes == NULL)
  {
    numLignes = new vector<int>;
    numLignes->push_back(numLigne);

    return 1;
  }
  else if (! binary_search (numLignes->begin(), numLignes->end(), numLigne))

  {
    numLignes->push_back(numLigne);
    return 1;
  }
  else
  {
    return 0;
  }
} // ------ Fin de AddLigne

bool Fichier::operator==(Fichier const & unFichier) const
// Algorithme : Trivial
{
  if (nomFic == unFichier.nomFic) return true;
  else return false;
} // ------ Fin de operator==

//-------------------------------------------- Constructeurs - destructeur
Fichier::Fichier ( string unNom )
// Algorithme : Trivial
{
#if defined ( MAP )
  cout << "Appel du constructeur de Fichier" << endl;
```

```
#endif
  nomFic =     unNom;
  numLignes =   NULL;
} // ------ Fin de Fichier

Fichier::Fichier ( const Fichier & unFichier )
// Algorithme : Creation d'un nouveau vector et recopie des elements
{
#if defined ( MAP )
  cout << "Appel du constructeur de copie de Fichier" << endl;
#endif
  nomFic =     unFichier.nomFic;
  numLignes =   new vector<int>;
  *numLignes =  *unFichier.numLignes;
} // ------ Fin de Fichier (Constructeur de copie)

Fichier::~Fichier ( )
// Algorithme : Trivial
{
#if defined ( MAP )
  cout << "Appel du destructeur de Fichier" << endl;
#endif
  delete numLignes;
} // ------ Fin de ~Fichier

//------------------------------------------------------------------ PRIVE

//----------------------------------------------------- Methodes protegees
```

../src/Fichier.cpp

```
/*************************************************************************
                          Fichier  -  description
                             -------------------
    debut                : Nov. 2011
    copyright            : (C) 2011 par Tristan Pourcelot & Jordan Vincent
*************************************************************************/

//-- Realisation de la classe <Fichier> (fichier Fichier.cpp) ------

//------------------------------------------------------------------ INCLUDE

//----------------------------------------------------- Include systeme
#include <iostream>
#include <algorithm>
using namespace std;

//----------------------------------------------------- Include personnel
#include "Fichier.h"

//------------------------------------------------------------ Constantes

//------------------------------------------------------------------ PUBLIC

//----------------------------------------------------- Methodes publiques
void Fichier::DisplayFichier() const
// Algorithme : Parcourt de numLignes et affichage des numeros de ligne
{
  cout << "\t" << nomFic;
  for (int i=0; i<numLignes->size(); i++)
  {
    cout << " " << numLignes->at(i);
  }
}

bool Fichier::AddLigne( int numLigne )
// Algorithme : Ajoute une ligne si elle n'existe pas deja.
{
  if (numLignes == NULL)
  {
```

```
    numLignes = new vector<int>;
41    numLignes->push_back(numLigne);

43    return 1;
   }
45 else if (! binary_search (numLignes->begin(), numLignes->end(), numLigne))

47 {
    numLignes->push_back(numLigne);
49    return 1;
   }
51 else
   {
53    return 0;
   }
55 } // ------ Fin de AddLigne

57 bool Fichier::operator==(Fichier const & unFichier) const
   // Algorithme : Trivial
59 {
   if (nomFic == unFichier.nomFic) return true;
61 else return false;
   } // ------ Fin de operator==
63
   //------------------------------------------------ Constructeurs - destructeur
65 Fichier::Fichier ( string unNom )
   // Algorithme : Trivial
67 {
   #if defined ( MAP )
69   cout << "Appel du constructeur de Fichier" << endl;
   #endif
71   nomFic =    unNom;
   numLignes =   NULL;
73 } // ------ Fin de Fichier

75 Fichier::Fichier ( const Fichier & unFichier )
   // Algorithme : Creation d'un nouveau vector et recopie des elements
77 {
   #if defined ( MAP )
79   cout << "Appel du constructeur de copie de Fichier" << endl;
   #endif
81   nomFic =    unFichier.nomFic;
   numLignes =   new vector<int>;
83   *numLignes =  *unFichier.numLignes;
   } // ------ Fin de Fichier (Constructeur de copie)
85
   Fichier::~Fichier ( )
87 // Algorithme : Trivial
   {
89 #if defined ( MAP )
   cout << "Appel du destructeur de Fichier" << endl;
91 #endif
   delete numLignes;
93 } // ------ Fin de ~Fichier

95 //------------------------------------------------------------------ PRIVE

97 //----------------------------------------------------- Methodes protegees
```

../src/Fichier.cpp

```
1 /*************************************************************************
                          Fichier  -  description
3                          -------------------
    debut                : Nov. 2011
5   copyright            : (C) 2011 par Tristan Pourcelot & Jordan Vincent
*************************************************************************/
7
   //-- Realisation de la classe <Fichier> (fichier Fichier.cpp) -----
9
```

```
1  //--------------------------------------------------------------------- INCLUDE
2
3  //---------------------------------------------------------- Include systeme
4  #include <iostream>
5  #include <algorithm>
6  using namespace std;
7
8  //---------------------------------------------------------- Include personnel
9  #include "Fichier.h"
10
11 //------------------------------------------------------------------- Constantes
12
13 //------------------------------------------------------------------------ PUBLIC
14
15 //---------------------------------------------------------- Methodes publiques
16 void Fichier::DisplayFichier() const
17 // Algorithme : Parcourt de numLignes et affichage des numeros de ligne
18 {
19   cout << "\t" << nomFic;
20   for (int i=0; i<numLignes->size(); i++)
21   {
22     cout << " " << numLignes->at(i);
23   }
24 }
25
26 bool Fichier::AddLigne( int numLigne )
27 // Algorithme : Ajoute une ligne si elle n'existe pas deja.
28 {
29   if (numLignes == NULL)
30   {
31     numLignes = new vector<int>;
32     numLignes->push_back(numLigne);
33
34     return 1;
35   }
36   else if (! binary_search (numLignes->begin(), numLignes->end(), numLigne))
37
38   {
39     numLignes->push_back(numLigne);
40     return 1;
41   }
42   else
43   {
44     return 0;
45   }
46 } // ------ Fin de AddLigne
47
48 bool Fichier::operator==(Fichier const & unFichier) const
49 // Algorithme : Trivial
50 {
51   if (nomFic == unFichier.nomFic) return true;
52   else return false;
53 } // ------ Fin de operator==
54
55 //-------------------------------------------- Constructeurs - destructeur
56 Fichier::Fichier ( string unNom )
57 // Algorithme : Trivial
58 {
59 #if defined ( MAP )
60   cout << "Appel du constructeur de Fichier" << endl;
61 #endif
62   nomFic =    unNom;
63   numLignes =   NULL;
64 } // ------ Fin de Fichier
65
66 Fichier::Fichier ( const Fichier & unFichier )
67 // Algorithme : Creation d'un nouveau vector et recopie des elements
68 {
69 #if defined ( MAP )
70   cout << "Appel du constructeur de copie de Fichier" << endl;
```

```
#endif
  nomFic =     unFichier.nomFic;
  numLignes =    new vector<int>;
  *numLignes =  *unFichier.numLignes;
} // ------ Fin de Fichier (Constructeur de copie)

Fichier::~Fichier ( )
// Algorithme : Trivial
{
#if defined ( MAP )
  cout << "Appel du destructeur de Fichier" << endl;
#endif
  delete numLignes;
} // ------ Fin de ~Fichier

//------------------------------------------------------------------------ PRIVE

//----------------------------------------------------------- Methodes protegees
```

../src/Fichier.cpp

Plan de test