```
// Complexe.cc

#include <cmath>
#include <sstream>
#include "Complexe.h"

Complexe::Complexe(){
  m_reel = m_img = 0;
}

Complexe::Complexe(float a, float b){
  m_reel = a;
  m_img = b;
}

Complexe::~Complexe(){}

void Complexe::setReel(float a){ m_reel = a; }

void Complexe::setImg(float b){ m_img = b; }

bool Complexe::egal(const Complexe& autre) const{
  return m_reel == autre.m_reel && m_img == autre.m_img;
}

string Complexe::toString() const {
    ostringstream out;
    out << "(" << m_reel;

    if(m_img>=0) out << " + " << m_img << "i)";
    else out << "-" << -m_img << "i)";

    string s = out.str();

    return s;
}

void Complexe::additionner(const Complexe& autre, Complexe& somme) const{
  somme.m_reel = m_reel + autre.m_reel;
  somme.m_img = m_img + autre.m_img;
}

void Complexe::multiplier(const Complexe& autre, Complexe& produit) const{
  produit.m_reel = m_reel * autre.m_reel - m_img * autre.m_img;
  produit.m_img = m_reel * autre.m_img + m_img * autre.m_reel;
}

void Complexe::soustraire(const Complexe& autre, Complexe& difference) const{
  difference.m_reel = m_reel - autre.m_reel;
  difference.m_img = m_img - autre.m_img;
}

void Complexe::inverse(Complexe& inv) const{
// 1/(a + ib) = a/(a^2 + b^2) - i b/(a^2 + b^2)
  float module_carre = pow(m_reel,2) + pow(m_img,2);
  inv.m_reel = m_reel/module_carre;
  inv.m_img = -m_img/module_carre;
}

void Complexe::diviser(const Complexe& autre, Complexe& quotient) const{
  Complexe inv_autre;
  autre.inverse(inv_autre);
  multiplier(inv_autre, quotient);
}

void Complexe::conjugue(Complexe& z_barre) const{
// conjugué de a+ib = a - ib
```



```
  z_barre.m_reel = m_reel;
  z_barre.m_img = -m_img;
}

float Complexe::module() const{
// |z| = racine carrée de a^2 + b^2
  return sqrt(pow(m_reel,2) + pow(m_img,2));
}
```

```
//Complexe.h

#ifndef _COMPLEXE_H_
#define _COMPLEXE_H_

#include <string>

using namespace std;

class Complexe{
 private:
  float m_reel, m_img;

 public:
  Complexe();
  Complexe(float a, float b);
  ~Complexe();
  string toString() const;
  void setReel(float a);
  void setImg(float b);
  bool egal(const Complexe& autre) const;
  void additionner(const Complexe& z, Complexe& somme) const;
  void multiplier(const Complexe& z, Complexe& produit) const;
  void soustraire(const Complexe& z, Complexe& difference) const;
  void diviser(const Complexe& z, Complexe& quotient) const;
  void inverse(Complexe& inv) const;
  void conjugue(Complexe& z_barre) const;
  float module() const;
};
#endif
```