```
#include<iostream>
#include<cstdlib>

#include "complexe.h"

using namespace std;

/// ########### les accesseurs ################# ///

int Complexe::getRe()
{
        return my_re;
}

int Complexe::getIm()
{
        return my_im;
}

void Complexe::setRe(int r)
{
        my_re = r;
}

void Complexe::setIm(int i)
{
        my_im = i;
}

/// ########### les constructeur ############### ///

Complexe::Complexe()
{
        setRe( 0 );
        setIm( 0 );
// initialisation Ã  0
}

Complexe::Complexe(int r, int i)
{
        setRe( r );
        setIm( i );
}

/// ########### les methodes ################### ///

void Complexe::afficher()
{
        cout << getRe() << " + i" << getIm() << endl;
}


void Complexe::addition( Complexe c, Complexe & r )
{
        r.setRe( getRe() + c.getRe() );
        r.setIm( getIm() + c.getIm() );
}

void Complexe::soustraire( Complexe c, Complexe & r )
{
        r.setRe( getRe() - c.getRe() );
        r.setIm( getIm() - c.getIm() );
```



```
}

void Complexe::multiplier( Complexe c, Complexe & r )
{
        r.setRe( (getRe()*c.getRe()) - (getIm()*c.getIm()) );
        r.setIm( (getRe()*c.getIm()) + (getIm()*c.getRe()) );
}
```

fÃ©vr. 04, 11 17:57 **main.cc** Page 1/1

```
#include <iostream>
#include <cstdlib>

#include "rationnel.h"
#include "outil.h"

using namespace std;

int main()
{
        Rationnel p1(4,2);
        Rationnel p2(6,8);
        Rationnel rr;
        p1.affiche();
        p2.affiche();
        p1.multiplication(p2,rr);
        rr.affiche();
        rr.reducteur();
        rr.affiche();

        return 0;
}
```

fÃ©vr. 03, 11 20:06 **outil.cc** Page 1/1

```
#include<iostream>
#include<cstdlib>

#include "outil.h"

using namespace std;

void echange(int a, int b)
{
        a=a+b;
        b=a-b;
        a=a-b;
}

int pgcdrationnel( int a, int b )
{
        int r=0, q=0;

        if ( a < b )
                echange(a,b);
        do
        {
                r = a % b;
                q = (a-r)/b;
                if ( r!=0 )
                {
                        a = b;
                        b = r;
                }
        }
        while ( r != 0 );
        return b;
}
```

```
#include<iostream>
#include<cstdlib>

#include "rationnel.h"

using namespace std;

/// ########### les methodes ################### ///

void Rationnel::affiche() const
{
        cout << my_num << "/" << my_deno << endl;
}

void Rationnel::transforme()
{
        setNum( getNum() + getDeno() );
        setDeno( getNum() - getDeno() );
        setNum( getNum() - getDeno() );
}

bool Rationnel::egalite( Rationnel p)
{
        if ( getNum()/getDeno() == p.getNum()/p.getDeno() )
                {
                        cout << "les deux rationnels sont egaux" << endl;
                        return 1;
                }
        else
        {
                cout << "les deux rationnels ne sont pas egaux" << endl;
                return 0;
        }
}



void Rationnel::addition(Rationnel p, Rationnel & resultat)
{
        if ( getDeno() != p.getDeno() )
        {
                resultat.setNum( (getNum() * p.getDeno()) + (p.getNum() * getDen
o()) );
                resultat.setDeno( p.getDeno() * getDeno() );
        }
        else
        {
                resultat.setDeno( getDeno() );
                resultat.setNum( getNum() + p.getNum() );
        }
}

void Rationnel::soustraction(Rationnel p, Rationnel & resultat)
{
        if ( getDeno() != p.getDeno() )
        {
                resultat.setNum( (getNum() * p.getDeno()) - (p.getNum() * getDen
o()) );
                resultat.setDeno( p.getDeno() * getDeno() );
        }
        else
        {
```



```
                resultat.setDeno( getDeno() );
                resultat.setNum( getNum() + p.getNum() );
        }
}

void Rationnel::multiplication(Rationnel p, Rationnel & resultat)
{
        resultat.setNum( getNum() * p.getNum() );
        resultat.setDeno( getDeno() * p.getDeno() );
}

void Rationnel::division(Rationnel p, Rationnel & resultat)
{
        p.transforme();
        multiplication(p,resultat);
}

void Rationnel::reducteur()
{
        int n = pgcdrationnel( getDeno(), getNum() );
        setNum(getNum()/n);
        setDeno(getDeno()/n);
}

/// ########### les constructeurs ############# ///

Rationnel::Rationnel()
{
        setDeno(0);
        setNum(1);
}

Rationnel::Rationnel( int n, int d )
{
        setNum(n);
        setDeno(d);
}

/// ########### les accesseurs ############### ///

void Rationnel::setNum( int n )
{
        my_num = n;
}

void Rationnel::setDeno(int d )
{
        my_deno = d;
}

int Rationnel::getDeno()
{
        return my_deno;
}

int Rationnel::getNum()
{
        return my_num;
}
```

fÃ©vr. 04, 11 18:03 **testecomplexe.cc** Page 1/1

```
#include<iostream>
#include<cstdlib>

#include "complexe.h"

using namespace std;

int main()
{
        Complexe c1(3,5);
        Complexe c2(2,1);
        Complexe rc;

        c1.afficher();
        c2.afficher();

        cout << endl;

        c1.multiplier(c2,rc);
        rc.afficher();

        return 0;
}
```

fÃ©vr. 04, 11 17:30 **complexe.h** Page 1/1

```
#ifndef COMPLEXE_H
#define COMPLEXE_H

class Complexe
{
        private:
/// ########### les atributs ################### ///
                int my_re;
                int my_im;

        public:
/// ########### les methodes ################### ///
                void afficher();
                void addition( Complexe c, Complexe & r );
                void soustraire( Complexe c, Complexe & r );
                void multiplier( Complexe c, Complexe & r );
                void diviser( Complexe c, Complexe & r );

/// ########### les constructeur ################ ///
                Complexe();
                Complexe(int r, int i);

/// ########### les accesseurs ################## ///
                int getRe();
                int getIm();

                void setRe( int r );
                void setIm( int i );
};

#endif
```

fÃ©vr. 03, 11 20:06 **outil.h** Page 1/1

```
#ifndef OUTIL_H
#define OUTIL_H

int pgcdrationnel( int a, int b );

void echange(int a, int b);

#endif
```

fÃ©vr. 04, 11 19:04 **rationnel.h** Page 1/1

```
#ifndef RATIONNEL_H
#define RATIONNEL_H

#include "outil.h"

class Rationnel
{
        private:
/// ########### les atributs ################### ///
                int my_num;
                int my_deno;

        public:
/// ########### les methodes ################### ///
                void affiche() const;
                void transforme();
                bool egalite( Rationnel p);
        // les opÃ©rations
                void addition(Rationnel p, Rationnel & resultat);
                void soustraction(Rationnel p, Rationnel & resultat);
                void multiplication(Rationnel p, Rationnel & resultat);
                void division(Rationnel p, Rationnel & resultat);
        // les outils
                void reducteur();

/// ########### les constructeurs ############## ///
                Rationnel();
                Rationnel(int n, int d);

/// ########### les accesseurs ################ ///
                void setNum( int n );
                void setDeno( int d );
                int getNum() ;
                int getDeno() ;

};

#endif
```

```
cc = g++

### RATIONNEL ###

sourcerationnel = rationnel.cc main.cc outil.cc
objetrationnel = $(sourcerationnel:.cc=.o)

rationnel : $(objetrationnel)
		$(cc) $^ -o $@

### COMPLEXE ###

sourcecomplexe = testecomplexe.cc complexe.cc
objetcomplexe = $(sourcecomplexe:.cc=.o)

complexe : $(objetcomplexe)
		$(cc) $^ -o $@

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

rationnel.o : rationnel.cc rationnel.h outil.h
		$(cc) -c $<

main.o : main.cc rationnel.h outil.h
		$(cc) -c $<

outil.o : outil.cc outil.h
		$(cc) -c $<

testecomplexe.o : testecomplexe.cc complexe.h
		$(cc) -c $<


complexe.o : complexe.cc complexe.h
		$(cc) -c $<


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

clean :
		rm -vr *.o
```