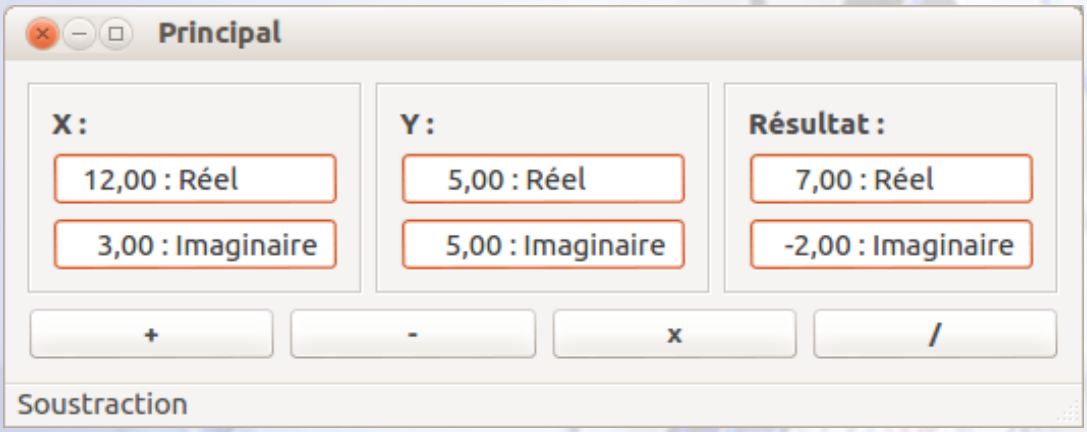
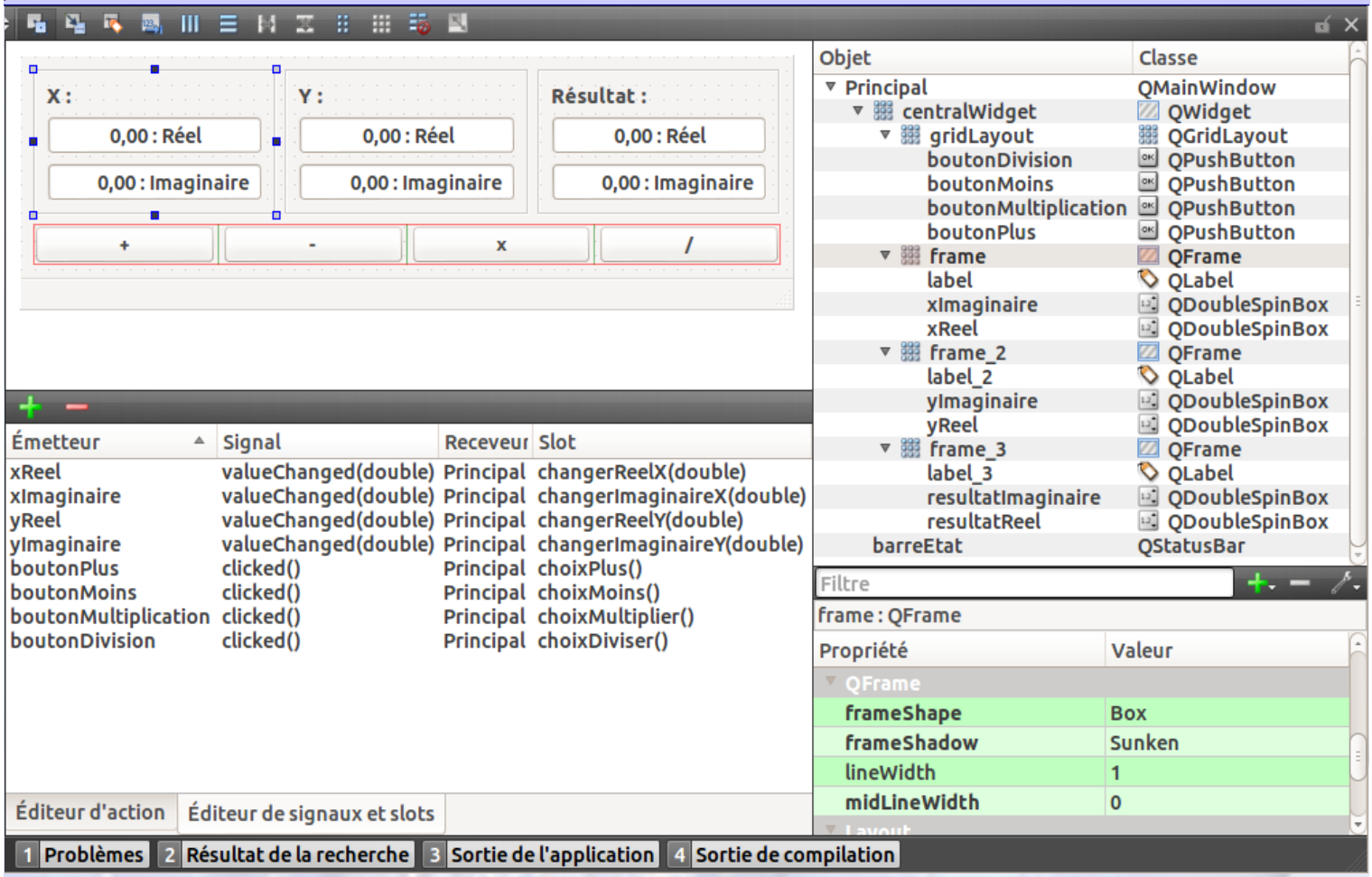
Ce projet permet de valider le cours concernant l'utilisation de fonctions afin de réduire la complexité au travers de la mise en œuvre du projet sur le calcul de nombres complexes.

*Chaque nombre complexe est ici représenté par une **structure**.*



x PRINCIPAL.UI



PRINCIPAL.H

```
#ifndef PRINCIPAL_H
#define PRINCIPAL_H

#include <QMainWindow>
#include "ui_principal.h"

//------------------------------ Déclarations personnelles --------------------
struct Complexe
{
    double reel, imaginaire;
};

Complexe addition(const Complexe &c1, const Complexe &c2);
Complexe soustraction(const Complexe &c1, const Complexe &c2);
Complexe multiplication(const Complexe &c1, const Complexe &c2);
Complexe division(const Complexe &c1, const Complexe &c2);

double module(const Complexe &c);
double argument(const Complexe &c);

enum Operation {Plus, Moins, Multiplier, Diviser};

//------------------- Déclaration de la classe représentant la fenêtre -------
class Principal : public QMainWindow, public Ui::Principal
{
    Q_OBJECT

public:
    Principal(QWidget *parent = 0);
private:
    Complexe x, y, resultat;
    Operation operation;
private:
    void calcul();
    void message();

private slots:
    void changerReelX(double valeur);
    void changerImaginaireX(double valeur);
    void changerReelY(double valeur);
    void changerImaginaireY(double valeur);
    void choixPlus();
    void choixMoins();
    void choixMultiplier();
    void choixDiviser();
};

#endif // PRINCIPAL_H
```

PRINCIPAL.CPP

```
#include <math.h>
#include "principal.h"

Principal::Principal(QWidget *parent) : QMainWindow(parent)
{
    setupUi(this);
    x.reel = x.imaginaire = 0.0;
    y.reel = y.imaginaire = 0.0;
    operation = Plus;
    barreEtat->showMessage("Addition");
}
```

```
void Principal::changerReelX(double valeur)         { x.reel = valeur; calcul(); }
void Principal::changerImaginaireX(double valeur) { x.imaginaire = valeur; calcul(); }
void Principal::changerReelY(double valeur)         { y.reel = valeur; calcul(); }
void Principal::changerImaginaireY(double valeur) { y.imaginaire = valeur; calcul(); }

void Principal::choixPlus()               { operation = Plus; calcul(); }
void Principal::choixMoins()              { operation = Moins; calcul(); }
void Principal::choixMultiplier()         { operation = Multiplier; calcul(); }
void Principal::choixDiviser()            { operation = Diviser; calcul(); }

void Principal::calcul()
{
     switch (operation)
     {
          case Plus : resultat = addition(x, y); barreEtat->showMessage("Addition"); break;
          case Moins : resultat = soustraction(x, y); barreEtat->showMessage("Soustraction"); break;
          case Multiplier : resultat = multiplication(x, y); barreEtat->showMessage("Multiplication"); break;
          case Diviser : resultat = division(x, y); barreEtat->showMessage("Division");  break;
     }
     resultatReel->setValue(resultat.reel);
     resultatImaginaire->setValue(resultat.imaginaire);
}

Complexe addition(const Complexe &c1, const Complexe &c2)
{
    Complexe c;
    c.reel = c1.reel + c2.reel;
    c.imaginaire = c1.imaginaire + c2.imaginaire;
    return c;
}

Complexe soustraction(const Complexe &c1, const Complexe &c2)
{
    Complexe c;
    c.reel = c1.reel - c2.reel;
    c.imaginaire = c1.imaginaire - c2.imaginaire;
    return c;
}

Complexe multiplication(const Complexe &c1, const Complexe &c2)
{
    Complexe c;
    double mod = module(c1) * module(c2);
    double arg = argument(c1) + argument(c2);
    c.reel = mod*cos(arg);
    c.imaginaire = mod*sin(arg);
    return c;
}

Complexe division(const Complexe &c1, const Complexe &c2)
{
    Complexe c;
    double mod = module(c1) / module(c2);
    double arg = argument(c1) - argument(c2);
    c.reel = mod*cos(arg);
    c.imaginaire = mod*sin(arg);
    return c;
}

double module(const Complexe &c) {  return sqrt(c.reel*c.reel + c.imaginaire*c.imaginaire);}

double argument(const Complexe &c)
{
     if (c.imaginaire == 0.0) return 0.0;
     if (c.reel == 0.0) return c.imaginaire>0.0 ? M_PI_2 : -M_PI_2;
     return atan(c.imaginaire / c.reel);
}
```