# TP2 *Calculatrice*

Le but du TP est de réaliser une petite calculatrice fonctionnelle en C++ et Qt5 en dissociant le code de l'interface graphique de celui du moteur.

## Exercice 1 : Moteur

Il faudrait tout d'abord développer le « moteur » de notre calculatrice, c'est un exercice de programmation intéressant et non trivial. Mais comme nous nous intéressons plus particulièrement aux développements des interfaces graphique, vous vous contenterez de récupérer le code source du moteur dans votre espace moodle.

Voici les règles de fonctionnement de notre calculatrice :

1. L'ordre d'entrée des opérandes et des opérateurs est en notation infixe : Exemple :
```
"1 + 2" donne 3,
"1 * ( 2 + 3)" donne 5,
"1 + ^2 3" donne 10
```
2. Les priorités entre opérateurs sont respectées.
3. Attention ! Sur la majorité des calculatrices électroniques, les opérateurs binaires sont donnés en infixe et les opérateurs unaires en postfixe. Ce n'est pas le cas ici où les opérateurs unaires sont placés avant : "^2 3" ou "cos 4" par exemple.

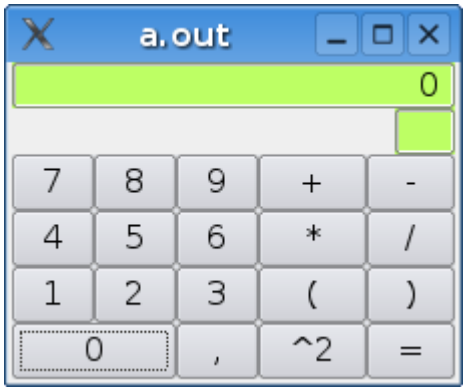
Le moteur consiste en une classe `Calc` gérant à la fois l'entrée des opérandes, des opérateurs, et **effectuant les calculs au fur et à mesure** lorsque les priorités sur les opérateurs le permettent. Notre classe `Calc` est capable de traiter l'exemple suivant :

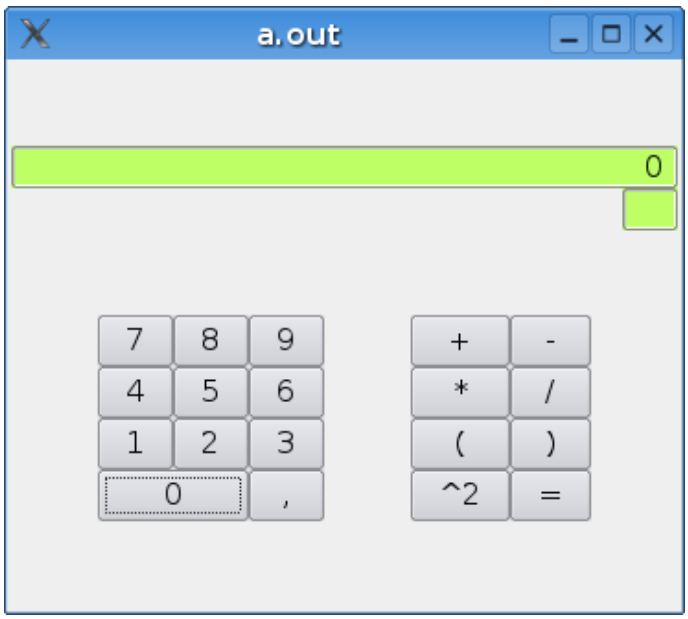
```
Calc calc;
calc.addChar('1');
calc.addChar('0');
calc.addChar('.');
calc.addChar('5');
calc.plus();
calc.square();
calc.bracketL();
calc.addChar('2');
calc.times();
calc.addChar('2');
calc.bracketR();
cout << calc.equal() << endl;
```

Prenez le temps de comprendre le code source du moteur. Passez ensuite aux exercices suivants.

## Exercice 2 : Construire l'interface

Nous allons maintenant réaliser l'interface graphique de notre calculatrice. Étudiez les captures d'écran suivantes de manière à bien disposer les différents éléments de l'interface :




Faites attention aux placements, aux espaces, aux élastiques, aux différents layouts et à la taille des widgets : votre interface doit être la même que celle présente sur les images ci-dessus.

Les écrans d'affichage de la calculatrice seront réalisés avec le widget QLineEdit en **lecture seule**.

Vous réaliserez cette interface avec les **widgets standards** fournis par la librairie Qt.

## Exercice 3 : Lier l'interface graphique au moteur via un traducteur

Il reste maintenant à modifier le code déjà écrit de manière à relier les événements et les diverses opérations réalisées. Notons néanmoins que :

1. La zone d'affichage des résultats se comporte comme dans une calculatrice habituelle :
   - affichage du chiffre en cours
   - résultats des calculs au fur et à mesure si connu par le moteur*
   - résultat final
2. La zone d'affichage de l'opérateur courant affiche l'opérateur en cours tant que l'opérateur suivant n'a pas été sélectionné

Vous écrirez une classe Traducteur qui se chargera de faire la liaison entre le moteur et l'interface graphique (cf. tp1 exercice 2).

**Le moteur (i.e. : la classe Calc) ne doit pas faire appel à la librairie Qt.**

* Pour ce faire, vous serez certainement amené à ajouter une(des) méthode(s) au moteur.

## Exercice 4 : Réaliser votre propre bouton

On peut constater que l'utilisation d'un bouton QPushButton n'est pas très aisé pour notre exercice. En effet, le signal clicked n'envoie aucune information. Imaginez que nous devions utiliser 500 boutons, serait-il judicieux d'écrire 500 slots connectés à chacun de ces boutons ?
Vous allez créer votre propre bouton héritant de QpushButton et définissant le signal suivant :

```
void sendValue(char) ; // ou int ou QString pour le paramètre ...
```

Vous remplacerez ensuite vos anciens boutons par ce nouveau bouton et vous modifierez les slots en conséquence. Est-ce que l'usage de tous ces slots est indispensable en Qt5 ? Et en Qt4 ?