

CALCULATRICE

Présentation

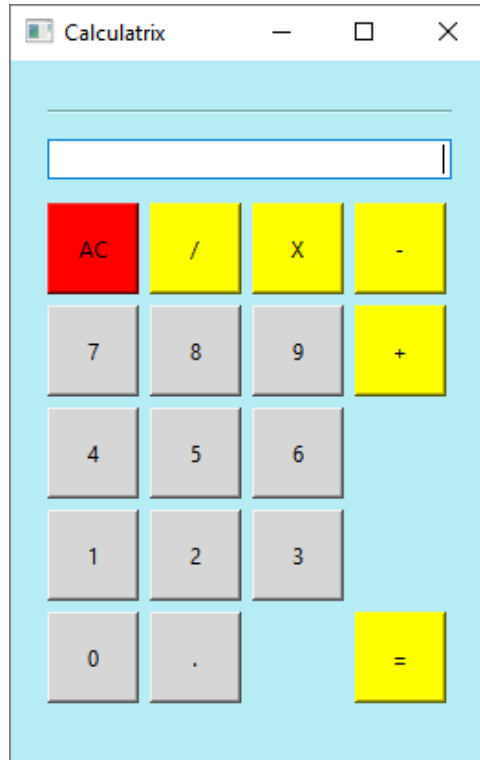
CALCULATRICE

I. Affichage graphique :.....	3
Programme en fonctionnement :.....	3
Création du squelette :.....	3
Élément graphique :.....	4
II. Programmation:.....	4
Création des <i>méthodes et variables</i> :.....	4
Créations des boutons :.....	5
Affichage des chiffres et virgule flottante :.....	5
Affichage des opérateurs :.....	6
Méthode init :.....	7
Méthode nombre :.....	7
Méthode calcul :.....	7
Méthode égal :.....	9

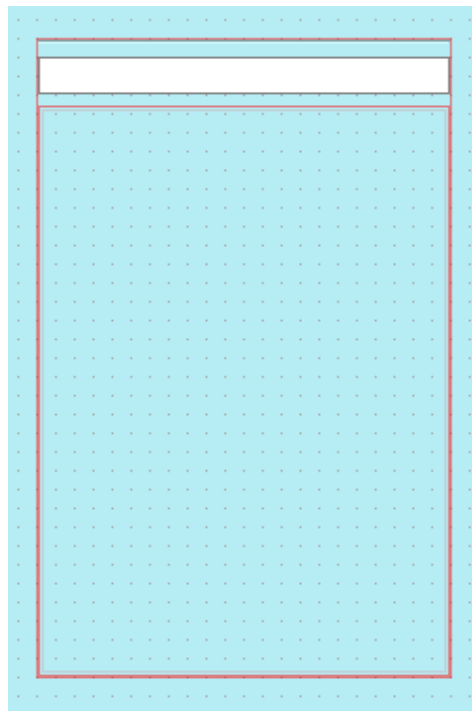
La calculatrice se présente comme ceci :

I. Affichage graphique :

Programme en fonctionnement :



Création du squelette :



Élément graphique :

Objet	Classe
fenetre	QMainWindow
centralwidget	QWidget
general	QVBoxLayout
grille	QGridLayout
affichage	QLineEdit
line	Line

II. Programmation:

Création des méthodes et variables :

```
#ifndef FENETRE_H
#define FENETRE_H

#include <QMainWindow>
#include <QString>
#include <QPushButton>

QT_BEGIN_NAMESPACE
namespace Ui { class fenetre; }
QT_END_NAMESPACE

class fenetre : public QMainWindow
{
    Q_OBJECT

private:
    Ui::fenetre *ui;
    QString operande1;
    QString operande2;
    QString operateur;
    QPushButton * touches;

public:
    fenetre(QWidget *parent = nullptr);
    ~fenetre();

public slots:
    void nombre();
    void init();
    void egal();
    void calcul();

};
#endif // FENETRE_H
```

Créations des boutons :

```
fenetre::fenetre(QWidget *parent) : QMainWindow(parent) , ui(new Ui::fenetre)
{
    ui->setupUi(this);
    // Méthode init() initialisée au démarrage de calculatrice
    init();
    // Déclaration de la variable touches de type QPushButton*
    touches = new QPushButton[17];

    // Création des boutons de la calculatrice et paramétrage de leur taille
    for(int i=0; i<17; i++)
    {
        touches[i].setFixedSize(50,50);
    }

    // Paramétrage de la couleur des boutons correspondant aux chiffres de 0 à 9 et au symbole "."
    for(int i=0; i<11; i++)
    {
        touches[i].setStyleSheet("background-color:lightgrey");
    }

    // Paramétrage de la couleur des boutons correspondant aux opérateurs (+, -, X, /)
    for(int i=11; i<16; i++)
    {
        touches[i].setStyleSheet("background-color:yellow");
    }
}
```

Affichage des chiffres et virgule flottante :

```
// chiffres
for(int i=0; i<10; i++)
{
    // Déclaration d'une variable n de type QString
    QString n;

    // Initialisation de la variable n par la valeur numérique i
    n.setNum(i);

    // Paramétrage du texte du bouton correspondant au chiffre
    touches[i].setText(n);

    // Connexion des boutons de nombres
    connect(&touches[i], SIGNAL(clicked()), this, SLOT(nombre()));

    // Formation des lignes et colonnes
    int l = ((9-i)/3)+1;
    int c = ((i-1)%3)+1;

    // Affiche les boutons dans la grille
    ui->grille->addWidget(&touches[i], l, c);
}
```

```

// Organisation des boutons dans la grille
// Affiche le bouton dans la grille
ui->grille->addWidget(&touches[0], 4, 1);

//
for(int i=0; i<17; i++)
{
    touches[i].setFixedSize(50,50);
}

```

Affichage des opérateurs :

```

// Paramétrage du texte du bouton correspondant à la virgule flottante
touches[10].setText(".");

// Connexion du bouton à la méthode nombre()
connect(&touches[10], SIGNAL(clicked()), this, SLOT(nombre()));

// Affiche le bouton dans la grille
ui->grille->addWidget(&touches[10], 4, 2);

// Paramétrage du texte du bouton correspondant à la division
touches[11].setText("/");
// Connexion du bouton à la méthode calcul()
connect(&touches[11], SIGNAL(clicked()), this, SLOT(calcul()));
// Affiche le bouton dans la grille
ui->grille->addWidget(&touches[11], 0, 2);

// Paramétrage du texte du bouton correspondant à la multiplication
touches[12].setText("X");
// Connexion du bouton à la méthode calcul()
connect(&touches[12], SIGNAL(clicked()), this, SLOT(calcul()));
// Affiche le bouton dans la grille
ui->grille->addWidget(&touches[12], 0, 3);

// Paramétrage du texte du bouton correspondant à la soustraction
touches[13].setText("-");
// Connexion du bouton à la méthode calcul()
connect(&touches[13], SIGNAL(clicked()), this, SLOT(calcul()));
// Affiche le bouton dans la grille
ui->grille->addWidget(&touches[13], 0, 4);

// Paramétrage du texte du bouton correspondant à l'addition
touches[14].setText("+");
// Connexion du bouton à la méthode calcul()
connect(&touches[14], SIGNAL(clicked()), this, SLOT(calcul()));
// Affiche le bouton dans la grille
ui->grille->addWidget(&touches[14], 1, 4);

// Paramétrage du texte du bouton correspondant à l'égal
touches[15].setText("=");
// Connexion du bouton à la méthode egal()
connect(&touches[15], SIGNAL(clicked()), this, SLOT(egal()));
// Affiche le bouton dans la grille
ui->grille->addWidget(&touches[15], 4, 4);

// Paramétrage du texte du bouton correspondant au reset
touches[16].setText("AC");
// Paramétrage de la couleur du bouton reset
touches[16].setStyleSheet("background-color:red");
// Connexion du bouton à la méthode init()
connect(&touches[16], SIGNAL(clicked()), this, SLOT(init()));
// Affiche le bouton dans la grille
ui->grille->addWidget(&touches[16], 0, 1);
}

```

Méthode init :

```
void fenetre::init()
{
    // Efface la valeur contenue dans la variable operande1
    operande1.clear();
    // Efface la valeur contenue dans la variable operande2
    operande2.clear();
    // Efface la valeur contenue dans la variable operateur
    operateur.clear();
    // Efface la valeur contenue dans le QLabel "affichage"
    ui->affichage->clear();
}
```

Méthode nombre :

```
void fenetre::nombre()
{
    // Déclaration d'une variable s de type QString
    // Initialisée par le texte correspondant au bouton envoyant le signal (0,1,2,3,4,5,6,7,8,9, ".")
    QString s = qobject_cast<QPushButton *>(sender())->text();
    // Ajoute dans le QLabel "affichage" la valeur contenue dans la variable s
    ui->affichage->setText(ui->affichage->text()+s);
}
```

Méthode calcul :

```
void fenetre::calcul()
{
    // Si le QLabel "affichage" est vide (aucun chiffre n'a été tapé)
    if(ui->affichage->text()=="")
    {
        // Renvoie un message d'erreur
        ui->affichage->setText("Error");
    }
    else
    {
        // On vérifie si le premier opérande est déjà initialisé :

        // 1 - Le premier opérande n'est pas initialisé
        if(operande1.isEmpty())
        {
            // On l'initialise avec le nombre écrit dans le QLabel "affichage"
            operande1=ui->affichage->text();
        }

        // 2 - La première opérande est déjà initialisée
        else
        {
            // On initialise la deuxième opérande avec le nombre écrit dans le QLabel "affichage"
            operande2=ui->affichage->text();
        }
    }
}
```

Calcul selon chaque opérateurs

```
// Si aucune des 2 opérands ne contiennent de virgule,  
// on les convertit en entier pour le calcul  
else  
{  
    // Choix de l'opération :  
  
    // 1 - Addition  
    if(operateur == "+")  
    {  
        // Le premier opérande est initialisé à nouveau  
        // et contient le résultat de l'addition  
        operande1.setNum(operande1.toInt() + operande2.toInt());  
    }  
  
    // 2 - Soustraction  
    if(operateur == "-")  
    {  
        // Le premier opérande est initialisé à nouveau  
        // et contient le résultat de la soustraction  
        operande1.setNum(operande1.toInt() - operande2.toInt());  
    }  
  
    // 3 - Multiplication  
    if(operateur == "X")  
    {  
        // Le premier opérande est initialisé à nouveau  
        // et contient le résultat de la multiplication  
        operande1.setNum(operande1.toInt() * operande2.toInt());  
    }  
  
    // 4 - Division  
    if(operateur == "/")  
    {  
        // Le premier opérande est initialisé à nouveau  
        // et contient le résultat de la division  
        operande1.setNum(operande1.toInt() / operande2.toInt());  
    }  
  
    }  
    // Efface la valeur contenue dans la deuxième opérande  
    operande2.clear();  
}  
// Efface le contenu du QLabel "affichage"  
ui->affichage->clear();  
}  
// La variable operateur est initialisée par le texte correspondant au bouton envoyant le signal (+, -, X, /)  
operateur = qobject_cast<QPushButton *>(sender()->text());  
}
```


Méthode égal :

```
void fenetre::egal()
{
    // Si le QLabel "affichage" est vide (aucun chiffre n'a été tapé)
    if(ui->affichage->text()=="")
    {
        // Renvoie un message d'erreur dans le QLabel "affichage"
        ui->affichage->setText("Error");
    }
    else
    {
        // On initialise la deuxième opérande avec le nombre écrit dans le QLabel "affichage"
        operande2=ui->affichage->text();

        // Si la première opérande et l'opérateur sont initialisées
        if(!operande1.isEmpty() && !opérateur.isEmpty())
        {
            // Déclaration d'une variable res de type QString
            QString res;

            // Si au moins 1 des 2 opérandes contient une virgule,
            // on les convertit en float pour le calcul
            if(operande1.contains('.') || operande2.contains('.'))
            {
                // Choix de l'opération :

                // 1- Addition
                if(opérateur == "+")
                {
                    // La variable res est initialisée
                    // et contient le résultat de l'addition
                    res.setNum(operande1.toFloat() + operande2.toFloat());
                }

                // 2 - Soustraction
                if(opérateur == "-")
                {
                    // La variable res est initialisée
                    // et contient le résultat de la soustraction
                    res.setNum(operande1.toFloat() - operande2.toFloat());
                }

                // 3 - Multiplication
                if(opérateur == "X")
                {
                    // La variable res est initialisée
                    // et contient le résultat de la multiplication
                    res.setNum(operande1.toFloat() * operande2.toFloat());
                }

                // 4 - Division
                if(opérateur == "/")
                {
                    // La variable res est initialisée
                    // et contient le résultat de la division
                    res.setNum(operande1.toFloat() / operande2.toFloat());
                }
            }
        }
    }
}
```

```

// Si aucune des 2 opérandes ne contiennent de virgule,
// on les convertit en entier pour le calcul
else
{
    // Choix de l'opération :

    // 1 - Addition
    if(operateur == "+")
    {
        // La variable res est initialisée
        // et contient le résultat de l'addition
        res.setNum(operande1.toInt() + operande2.toInt());
    }

    // 2 - Soustraction
    if(operateur == "-")
    {
        // La variable res est initialisée
        // et contient le résultat de la soustraction
        res.setNum(operande1.toInt() - operande2.toInt());
    }

    // 3 - Multiplication
    if(operateur == "X")
    {
        // La variable res est initialisée
        // et contient le résultat de la multiplication
        res.setNum(operande1.toInt() * operande2.toInt());
    }

    // 4 - Division
    if(operateur == "/")
    {
        // La variable res est initialisée
        // et contient le résultat de la division
        res.setNum(operande1.toInt() / operande2.toInt());
    }
}

// Affiche le résultat dans le QLabel "affichage"
ui->affichage->setText(res);
}
else
{
    // Renvoie un message d'erreur dans le QLabel "affichage"
    ui->affichage->setText("Error");
}
}
}

```