QT Calculator Documentation

Ömer Faruk Çelik

This calculator, which I implemented with the help of QT, performs addition and subtraction with hexadecimal values.

Implementation Details

1- Design

2- Functionality

1- Design

This design, consisting of 20 buttons and a label, was obtained from the xml file in the mainview.ui file.

Colors inspired by apple calculator design.

Border of the clicked button has been changed to white for a very short time to make it clear that the button has been pressed.

2- Functionality

When entering numbers into the calculator, you can only see the number you are entering on the screen.
When entering an operator into the calculator, you will see the results of the operations up to that point on the screen.
When you press the Clr key, you reset the calculator and go back to the beginning.

```
4   int tmpResult = 0;
5   int tmpOperand = 0;
6   bool lastCharWasOperator = false;
7   bool firstOperator = true;
8   bool lastOperatorIsPlus = false;
9   bool isNegative = false;
10  bool lastCharWasEqual = false;
```

Firstly, global variables responsible for various tasks are initialized.

tmpResult calculates the previous operations and keeps the result as the new operator is entered.
tmpOperand is responsible for keeping the last entered number.
lastCharWasOperator indicates that the last entered input is the operator.
firstOperator indicates that it is the first operator of expression .
lastOperatorIsPlus indicates that the last operator input is plus.
isNegative indicates that the last input value is negative.
lastCharWasOperator indicates that the last entered input is the equal sign.-

```
{
    ui->setupUi(this);


    connect(ui->pushButton_0,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_1,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_2,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_3,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_4,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_5,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_6,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_7,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_8,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_9,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_A,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_B,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_C,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_D,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_E,SIGNAL(released()),this,SLOT(onDigitPressed()));
    connect(ui->pushButton_F,SIGNAL(released()),this,SLOT(onDigitPressed()));


    connect(ui->pushButton_minus,SIGNAL(released()),this,SLOT(onMinusPressed()));
    connect(ui->pushButton_plus,SIGNAL(released()),this,SLOT(onPlusPressed()));

    connect(ui->pushButton_equal,SIGNAL(released()),this,SLOT(onEqualPressed()));
    connect(ui->pushButton_Clr,SIGNAL(released()),this,SLOT(onClrPressed()));

}
```

In this segment, all buttons are connected to the corresponding function.

```
//When digit is clicked
void MainView::onDigitPressed(){
    if(lastCharWasOperator || lastCharWasEqual){
        ui->display->clear();
    }
    QPushButton *pressed = (QPushButton*)sender();
    if(ui->display->text() == "0"){
        ui->display->setText(pressed->text());
    }else{
        ui->display->setText(ui->display->text() + pressed->text());
    }
    lastCharWasOperator = false;
    lastCharWasEqual = false;
}
```

When the number is clicked, if the previous input is operator or equal sign, the screen is cleared, if it is a number, the numbers are concatenated to each other.

```
//When minus is clicked
void MainView::onMinusPressed(){
    if(lastCharWasOperator){
        isNegative = !isNegative;
    }else{
        bool ok;
        if(isNegative){
            tmpOperand = ui->display->text().toLower().toInt(&ok,16) * -1 ;
            isNegative = false;
        }else{
            tmpOperand = ui->display->text().toLower().toInt(&ok,16);
        }
        if(firstOperator){
             tmpResult = tmpOperand;
             lastOperatorIsPlus = false;
             firstOperator = false;
        }else{
            if(lastOperatorIsPlus){
                tmpResult += tmpOperand;
            }else{
                tmpResult -= tmpOperand;
            }
             lastOperatorIsPlus = false;
        }
        ui->display->clear();
        ui->display->setText(QString::number(tmpResult,16).toUpper());
    }

      tmpOperand = 0;
      lastCharWasOperator = true;
      lastCharWasEqual = false;
}
```

```
//When plus is clicked
void MainView::onPlusPressed(){
    if(!lastCharWasOperator){
        bool ok;
        if(isNegative){
            tmpOperand = ui->display->text().toLower().toInt(&ok,16) * -1 ;
            isNegative = false;
        }else{
            tmpOperand = ui->display->text().toLower().toInt(&ok,16);
        }


        if(firstOperator){
            tmpResult = tmpOperand;
            lastOperatorIsPlus = true;
            firstOperator = false;
        }else{
            if(lastOperatorIsPlus){
                tmpResult += tmpOperand;
            }else{
                tmpResult -= tmpOperand;
            }
            lastOperatorIsPlus = true;
        }

        ui->display->clear();
        ui->display->setText(QString::number(tmpResult,16).toUpper());
    }
    tmpOperand = 0;
    lastCharWasOperator = true;
    lastCharWasEqual = false;
}
```

If lastCharWasOperator; when plus clicked ignore it when minus clicked change the sign of new input.
If operator has been used before keep the result of the operation so far in tmpresult and print it to the screen.

```
//When clear is clicked
//reset functionality
void MainView::onClrPressed(){
     ui->display->setText("0");
     tmpResult = 0;
     firstOperator = true;
     lastCharWasEqual = false;
     lastCharWasOperator = false;
     isNegative = false;
}
```

Program returns to initial conditions.

```
//When equal sign is clicked
void MainView::onEqualPressed(){
    //if enter clicked multiple times ignore it
    if(lastCharWasEqual){
        return;
    }

    if(lastCharWasOperator){
        ui->display->setText(QString::number(tmpResult,16));
    }else{
        bool ok;
        if(isNegative){
            tmpOperand = ui->display->text().toLower().toInt(&ok,16) * -1 ;
            isNegative = false;
        }else{
            tmpOperand = ui->display->text().toLower().toInt(&ok,16);
        }
        if(lastOperatorIsPlus){
            tmpResult += tmpOperand;
        }else{
            tmpResult -= tmpOperand;
        }
        ui->display->setText(QString::number(tmpResult,16).toUpper());

    }
    lastCharWasEqual = true;
    firstOperator = true;
    lastCharWasOperator = false;
}
```

When the button is clicked more than once, the clicks are ignored.
Evaluates and prints the calculation

You can continue to operate with the output by entering the operator.
If you enter a numeric value, you can operate from scratch.