

MY CALCULATOR PROJECT

This is my first English [readme.md](#), and it's a really big challenge for me. Maybe there are some grammatical problems, never mind.

The main ui about my calculator.



0

This is **CKy's Calculator**
(v0.0).If you think it's not
good and have some problems,
contract me.
My Wechat ID: changkeyuan

1

2

3

+

±

0

.

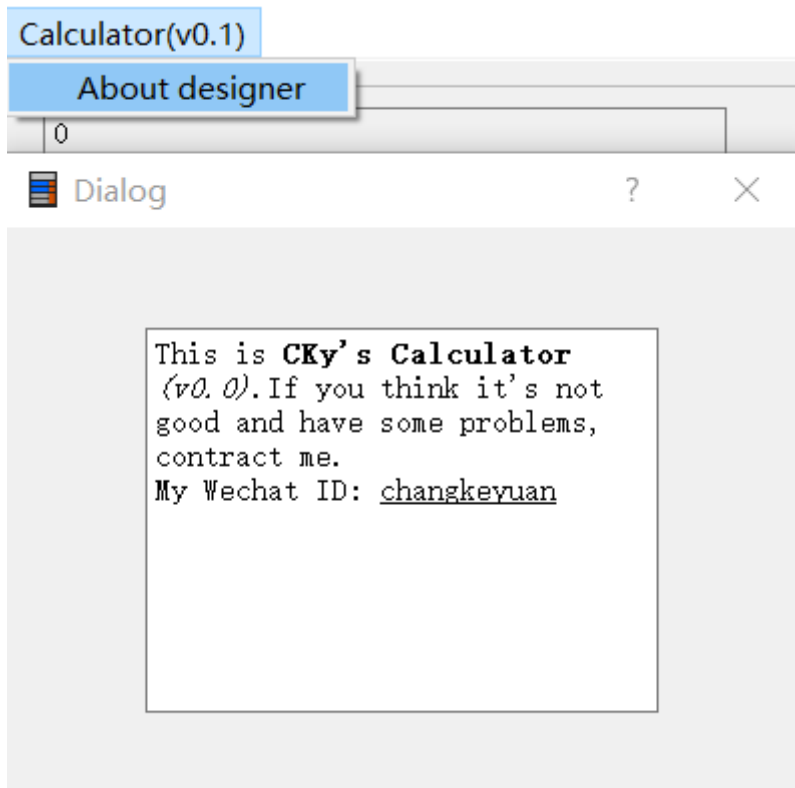
=



About the *extra function* of my calculator and its implementation

1. Having a menuBar and QAction which can see "About Designer"


Implementation: Create a new Dialog without Buttons and add a slot function to the QAction to show this new Dialog which contents something About Designer.



2. Adding a icon to my calculator

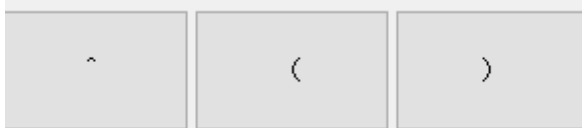
Implementation: Add the icon file to the .exe catalog and open it in the .pro file.



 basictwo

3. Having some more operator such as '^','(',')'.

Implementation: Regard them as higher priority operators.



4. using another textbrowser to show answer

Implementation: Just use the setText function.

```

void basictwo::on_equalbut_clicked()//点击等号的处理
{
    std::string temp;
    QString temp2;
    temp2=ui->label->text();
    temp=temp2.toStdString();
    convert(temp);
    QString answ;
    double ans;
    ans =calcu();
    answ=QString::number(ans);
    ui->textBrowser->setText(answ);
}

```

Some technical details

1. Using the QSignalMapper.

Implementation:First connect the button-clicked event with the map() in signalmapper,then use setmapping function.At last connect the map with slot function.

```

QSignalMapper* signalMapper = new QSignalMapper;
connect(ui->pushButton_0,SIGNAL(clicked()),signalMapper,SLOT(map()));
signalMapper->setMapping(ui->pushButton_0,QString("0"));
connect(ui->pushButton_1,SIGNAL(clicked()),signalMapper,SLOT(map()));
signalMapper->setMapping(ui->pushButton_1,QString("1"));
connect(ui->pushButton_2,SIGNAL(clicked()),signalMapper,SLOT(map()));
signalMapper->setMapping(ui->pushButton_2,QString("2"));

connect(signalMapper, SIGNAL(mapped( QString )),this, SLOT(clicknum(QString)));

QSignalMapper* signalMapper1 = new QSignalMapper;

connect(ui->pushButton_10,SIGNAL(clicked()),signalMapper1,SLOT(map()));
signalMapper1->setMapping(ui->pushButton_10,QString("f"));
connect(ui->pushButton_11,SIGNAL(clicked()),signalMapper1,SLOT(map()));
signalMapper1->setMapping(ui->pushButton_11,QString("p"));
connect(ui->pushButton_12,SIGNAL(clicked()),signalMapper1,SLOT(map()));
signalMapper1->setMapping(ui->pushButton_12,QString("g"));
connect(ui->pushButton_13,SIGNAL(clicked()),signalMapper1,SLOT(map()));
signalMapper1->setMapping(ui->pushButton_13,QString("d"));

connect(signalMapper1, SIGNAL(mapped( QString )),this, SLOT(clickdiv(QString)));

```

2. Handling of unary operators.

Determine the position of the last digit from the end of the string.Then use mid() function to get the

last digit, and use toDouble() function to turn a QString to double. At last deal with the number we get and use QString::number() function to turn a double to QString.

```
void basictwo::clickdiv( QString tex)//单目运算被点击
{
    double temp=0;
    QString tem;
    QString text=ui->label->text();
    int i;
    for( i=text.length()-1;i>=0;i++)
    {
        if(text[i]>='0'&&text[i]<='9')
        { if(i==0||(text[i-1]<'0' || text[i-1]>'9'))
            {
                tem=text.mid(i);
                text=text.left(i);
                temp=tem.toDouble();
                break;
            }
            else continue;
        }
    }
}
```

3. When number button and another button are clicked.

This is a slot function, just renew the QString and text.

```
void basictwo::clicknum( QString tex)//数字和双目，括号等被点击
{
    QString str1;
    QString text=ui->label->text();
    str1= (text=="0") ? "" : text ;
    str1+=tex;
    if(tex=='C')
        str1="0";
    ui->label->setText(str1);
}
```

4. Converting infix expression to postfix expression.

The main idea is using the stack data structure. We use a new class named Opera to store a number or an operator. The stack is a Opera class stack. The postfix expression is a Opera class queue. We start scanning from the beginning of the infix expression. When it's a number, we push it into the queue. If it's an operator we should judge the priority and push it into the operator stack. The '(' and ')' are the same, when we meet the ')', we should pop the operator until we get '('. Finally we push all operator into the postfix queue.

```

void basictwo::convert(std::string bef)
{
    map<char, int> pri;
    string ans;
    Opera te;
    pri['+'] = pri['-'] = 1; pri['*'] = pri['/'] = 2; pri['^'] = 3;
    for (int i = 0; i < bef.length(); i++) {
        if (bef[i] >= '0' && bef[i] <= '9') {
            te.flag = true;
            te.num = bef[i] - '0';
            while (i < bef.length() && bef[i] >= '0' && bef[i] <= '9') {
                te.num = te.num * 10 + (bef[i] - '0');
            }
            postfix.push(te);
        }
        else if (bef[i] == '(') {
            te.flag = false;
            te.ope = bef[i];
            rotor.push(te);
            i++;
        }
        else if (bef[i] == ')') {
            while (!rotor.empty() && rotor.top().ope != '(') {
                postfix.push(rotor.top());
                rotor.pop();
            }
            rotor.pop();
            i++;
        }
        else {
            te.flag = false;
            while (!rotor.empty() && pri[rotor.top().ope] >= pri[bef[i]]) {
                postfix.push(rotor.top());
                rotor.pop();
            }
            te.ope = bef[i];
            rotor.push(te);
            i++;
        }
    }
    while (!rotor.empty()) {
        postfix.push(rotor.top());
        rotor.pop();
    }
}

```

5. Dealing with the postfix expression.

The postfix expression is kept as a Opera class queue. We should scan from the beginning of this queue at first. If we get a number, we will push it into the number stack. And if we get a operator we should pop it up. The first operand is assigned to y, the last is assigned to x. Then we calculate and push the result into the number stack. We will get the last number in the stack, that is the answer.

```

double basictwo::calcu()
{
    double x, y;
    Opera now, temper;
    while (!postfix.empty()) {
        now = postfix.front();
        postfix.pop();
        if (now.flag == true) {
            tnum.push(now);
        }
        else {
            y = tnum.top().num;
            tnum.pop();
            x = tnum.top().num;
            tnum.pop();
            temper.flag = true;
            if (now.ope == '+') {
                temper.num = x + y;
            }
            else if (now.ope == '-') {
                temper.num = x - y;
            }
            else if (now.ope == '*') {
                temper.num = x * y;
            }
            else if (now.ope == '^')
            {
                temper.num = pow(x, y);
            }
            else {
                temper.num = x / y;
            }
            tnum.push(temper);
        }
    }
    return tnum.top().num;
}

```

About *.ui* and *ui_.h*

Firstly, `ui_*.h` is generated by `uic` (a Qt creator tool) after compiling the `.ui`.

When the content in the `.ui` is modified by Designer, `ui_.h` will also automatically update the content. The `ui_.h` file is converted from `.ui` to `c++` file.*

It mainly does the following work:

1. Defines a class `Ui_*` for encapsulating the visual design interface.
2. Defines the member variables of each component of the interface.
3. Defines a function named `setupUi` which is used to create individual interface components and sets their properties.
4. Define the namespace `Ui` and define a class `*` that inherits from `Ui_*`.

The problems I meet and solve.

1. QAction click event link to a new dialog display.
Solve: Adding a slot function in my `basictwo.cpp` and show the dialog.
2. Can not use debug (Unable to create a debugging engine.)
Solve: Download the windows kit and reload the debugger.
3. The process was ended forcefully.
Solve: Using a pointer without an assignment and assign it a value.
4. When I convert infix to postfix expression, I don't know how to indicate the priority of the operator.
Solve: Using `map` in C++ STL which can store elements in a mapped fashion, and each element has a key value and a mapped [value](#). So higher priority operator has a higher key value.

Reference

- About `.ui` and `ui_.h`
 - [Qt入门学习——Qt Creator 中 ui 文件和 Qt 代码关系](#)
 - [Qt项目界面文件（.ui）及其作用](#)
- About the implementation of the calculator algorithm
 - [Map in C++ Standard Template Library \(STL\)](#)
 - [中缀表达式转换成后缀表达式与后缀表达式的计算过程](#)
- About using `QSignalMapper` and some qt problems.
 - [QSignalMapper 信号映射](#)
 - [Qt学习笔记1——更高级Signals and Slots的用法](#)
 - [The process was ended forcefully.](#)