Project2_算法表达式求值演示_实验报告

一. 实验目的

(1) 以字符序列的形式从终端输入语法正确的、不含变量的整数表达式。利用下表给出的算符优先关系，实现对算术混合运算表达式的求值，并仿照求值中运算符栈、运算数栈、输入字符和主要操作的变化过程。

θ1 \ θ2	+	-	*	/	(	)	#
+	>	>	<	<	<	>	>
-	>	>	<	<	<	>	>
*	>	>	>	>	<	>	>
/	>	>	>	>	<	>	>
(	<	<	<	<	<	=	---
)	>	>	>	>	---	>	>
#	<	<	<	<	<	---	=

注: θ1<θ2 表示 θ1 的优先级低于 θ2

(2) 扩充运算符集，如增加乘方、单目减、赋值等运算。

(3) 计算器的功能和仿真界面（可参考 Windows 计算器的功能）。

二.实验环境

编程语言和开发工具

编程语言：C++

开发工具：Visual Studio Code\QT

三.分析与设计

3.1 需求分析:对输入的表达式完成加减乘除等计算以及并输出相应的栈操作。

3.2 设计思路及细节：

1. 大概：先将可能出现的运算符进行优先级排序，在单独写出相应的运算函数，设立是 sym，num 两个栈，分别储存运算符和数字，通过 string 输入将运算的式子，再通过 while 循环逐个遍历 string 里的每个字符，若为数字则存入 num，字符优先级若大于栈顶字符则进行栈顶字符的运算，否则将字符存入 sym 栈中。
   输入：可用鼠标操作也可用键盘操作。
2. 排错：语法错误会有报错，报错程序在 QT 中实现。
3. 不足：仅实现了双目运算符的操作。
4. 优点：有良好 ui 界面，键鼠操作提供便利性，增加求 n 次方。

四.代码

1.判断优先级的函数

```
int Jugde(char a, char b){//判断各个运算符的优先级，优先级高的为1，低的为-1，同级为0（括号）
    int i=1;
    switch (a)
    {
    case '+':
        if((b == '+' )||(b== '-') ||(b== ')') ||(b== '#')){
            i= 1;
            break;
        }else{
            i= -1;
            break;
        }
        break;
    case '-':
        if((b == '+' )||(b== '-') ||(b== ')') ||(b== '#')){
            i= 1;
            break;
        }else{
            i= -1;
            break;
        }
        break;
    case '*':
        if((b == '(')||(b=='^')){
            i= -1;
            break;
        }else{
            i= 1;
            break;
        }
        break;
    case '/':
        if((b == '(')||(b=='^')){
            i= -1;
            break;
        }else{
            i= 1;
            break;
        }
        break;
    case '^':
        if(b == '('){
            i= -1;
            break;
        }else{
            i= 1;
            break;
        }
        break;
    case '(':
        if(b == ')'){
            i= 0;
            break;
        }else{
            i= -1;
            break;
        }
        break;
    case ')':
        i= 1;
        break;
    case '#':
        if(b == '#'){
                i= 0;
                break;
            }else{
                i= -1;
                break;
            }
            break;
    default:
        break;
    }
    return i;
}
```

2.运算符的实现

```
double operate(double a, char op, double b){//进行运算符相应的计算
    double temp=0;
    if(op == '+'){
        temp=a + b;
    }else if(op == '-'){
        temp= a - b;
    }else if(op == '*'){
        temp=a * b;
    }else if(op == '/'){
        temp=a / b;
    }else if(op == '^'){
        temp=pow(a,b);
    }
    return temp;
}
```

3. 进行运算符的判断并进行相应计算

```
double compute(string put){//进行输入式子的运算
    stack<char> sym;//存放运算符的栈
    stack<double> num;//存放数字的栈
    char bench[9] = "+-*/^()#";//判断输入式子的运算符，#为终止符
    sym.push('#');
    put=put+'#';//先在输入式子中加入终止符
    double num1,num2,result,value;//分别为参与双目运算的两个数字，运算结果，入栈用数字
    char op;//入栈用字符
    int i=0,j;//坐标
    while(true){
        j=put.find_first_of(bench,i);//查找第一个运算符
        if(j==string::npos)break;
        if(j!=i){//未找到则为数字，并将数字入栈
            string temp(put,i,j-i);
            value =atof(temp.c_str());
            num.push(value);
            cout<<value<<"入num栈"<<endl;
        }
        switch (Jugde(sym.top(),put[j])){//判断栈内运算符与正在操作的运算符比较优先级后进行相应操作
        case -1:
            sym.push(put[j]);//若栈内运算符优先级小，将正操作的入栈
            if(put[j]!='#'){
                cout<<put[j]<<"入sym栈"<<endl;
            }
            i=j+1;
            break;
        case 0://说明运算符为括号
            sym.pop();
            if(put[j]!='#'){
                cout<<put[j]<<"出sym栈"<<endl;
            }
            i=j+1;
            break;
        case 1://若栈内运算符优先级大，进行栈内运算符运算
            op=sym.top();
            sym.pop();
            if(put[j]!='#'){
                cout<<put[j]<<"出sym栈"<<endl;
            }
            num2=num.top();
            cout<<num2<<"出num栈"<<endl;
            num.pop();
            num1=num.top();
            num.pop();
            cout<<num1<<"出num栈"<<endl;
            result = operate(num1,op,num2);
            cout<<num1<<"与"<<num2<<"进行计算得出结果"<<result<<endl;
            num.push(result);//将运算结果入栈
            cout<<result<<"入num栈"<<endl;
            i=j;
            break;
        default:
            break;
        }
    }
    return num.top();
```

4.qt 实现函数

相应的控件

```
Widget::Widget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Widget)
{
    ui->setupUi(this);
    QFont lineEditFont ( "Microsoft YaHei", 33, 80);
    ui->lineEdit->setFont(lineEditFont);
    QMovie *movie = new QMovie("qqemoji.gif");
    ui->label_2->setMovie(movie);
    movie->start();
    connect(ui->pushButton_0,&QPushButton::clicked,[this](){pushButton_0();});
    connect(ui->pushButton_1,&QPushButton::clicked,[this](){pushButton_1();});
    connect(ui->pushButton_2,&QPushButton::clicked,[this](){pushButton_2();});
    connect(ui->pushButton_3,&QPushButton::clicked,[this](){pushButton_3();});
    connect(ui->pushButton_4,&QPushButton::clicked,[this](){pushButton_4();});
    connect(ui->pushButton_5,&QPushButton::clicked,[this](){pushButton_5();});
    connect(ui->pushButton_6,&QPushButton::clicked,[this](){pushButton_6();});
    connect(ui->pushButton_7,&QPushButton::clicked,[this](){pushButton_7();});
    connect(ui->pushButton_8,&QPushButton::clicked,[this](){pushButton_8();});
    connect(ui->pushButton_9,&QPushButton::clicked,[this](){pushButton_9();});
    connect(ui->pushButton_add,&QPushButton::clicked,[this](){pushButton_add();});
    connect(ui->pushButton_sub,&QPushButton::clicked,[this](){pushButton_sub();});
    connect(ui->pushButton_mul,&QPushButton::clicked,[this](){pushButton_mul();});
    connect(ui->pushButton_div,&QPushButton::clicked,[this](){pushButton_div();});
    connect(ui->pushButton_equ,&QPushButton::clicked,[this](){pushButton_equ();});
    connect(ui->pushButton_del,&QPushButton::clicked,[this](){pushButton_del();});
    connect(ui->pushButton_pow,&QPushButton::clicked,[this](){pushButton_pow();});
    connect(ui->pushButton_ls,&QPushButton::clicked,[this](){pushButton_ls();});
    connect(ui->pushButton_rs,&QPushButton::clicked,[this](){pushButton_rs();});
    connect(ui->pushButton_point,&QPushButton::clicked,[this](){pushButton_point();});
    connect(ui->pushButton_clear,&QPushButton::clicked,[this](){pushButton_clear();});
}
```

报错弹窗，相关按钮以及相应的排错的实现（数字格式相仿，+-*/格式相仿）

```
void Widget::pushButton_ls(){
    if(put.endsWith("1")||put.endsWith("2")||put.
        showError();//左括号的排错
    }else{
        put+="(";
        index=1;
        ui->lineEdit->setText(put);
    }
}
void Widget::pushButton_rs(){
    if(rse==0){
        showError();
    }else{
        if(put.endsWith("(")){
            showError();
        }else{
            put+=")";
            rse=1;
            ui->lineEdit->setText(put);
        }
    }
}
void Widget::pushButton_point(){
    if(index==0){
        if(perr==0){
            put+=".";
            index++;
            perr++;
        }else{
            showError();
        }
    }else{
        showError();
    }
    ui->lineEdit->setText(put);
}
```

```
void Widget::pushButton_9(){
    put+="9";
    rse++;
    if(index==1){
        index=0;
    }
    ui->lineEdit->setText(put);
}
void Widget::pushButton_add(){
    if(index==0){
        put+="+";
        index++;
        rse=0;
        if(perr==1){
            perr=0;
        }
    }else{
        showError();
    }
    ui->lineEdit->setText(put);
}
```

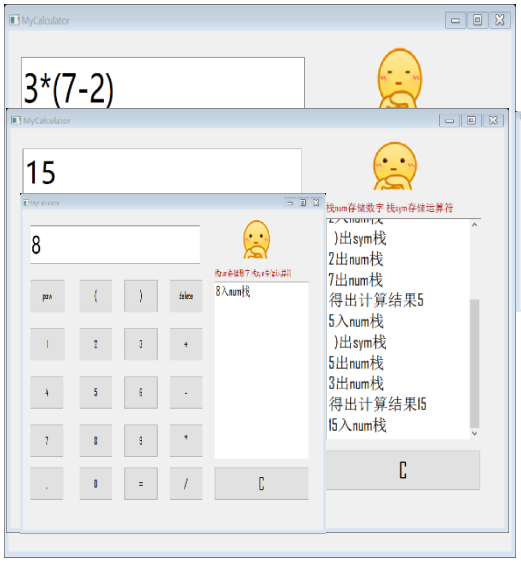
```
void Widget::pushButton_del(){
    ui->lineEdit->backspace();
    put.chop(1);
    if(index==1){
        index=0;
    }
    if(rse==0){
        rse=1;
    }
}
void Widget::pushButton_equ(){
    if(put.endsWith("+")||put.endsWith("-")|
        showError();
    }else{
        double temp=0;
        string temp1 =put.toStdString();
        temp=compute(temp1);
        put.clear();
        put= QString::number(temp,'g',6);
        ui->lineEdit->setText(put);
    }
}
void Widget::pushButton_clear(){
    put.clear();
    ui->lineEdit->setText(put);
    ui->textEdit->clear();
}
Widget::~Widget()
{
    delete ui;
}
```

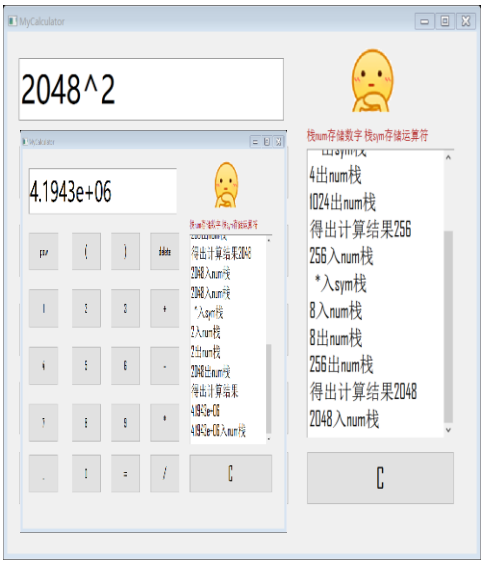
```
void showError(){
    QMessageBox message(QMessageBox::NoIcon, " ", "输入有问题呢");
    message.setIconPixmap(QPixmap("error.png"));
    message.exec();
}
```

五．实验结果

1.菜单

2.测试案例





3.排错

图 1 为“(”加“-”号的情况，实际上任何运算符都一样“(”内无数字直接加“)”也会报错

图 2 除“(”外，以其他运算符开头的式子也会报错。

图 3 为运算符后直接加“)”

图四不完整的式子进行计算也会报错

Tips：因为排错是在对应按钮按下或者点击是进行，有问题的输入不会被记录到栈中，截图难以分辨，欲实际了解排错请使用程序。