计算器实验报告

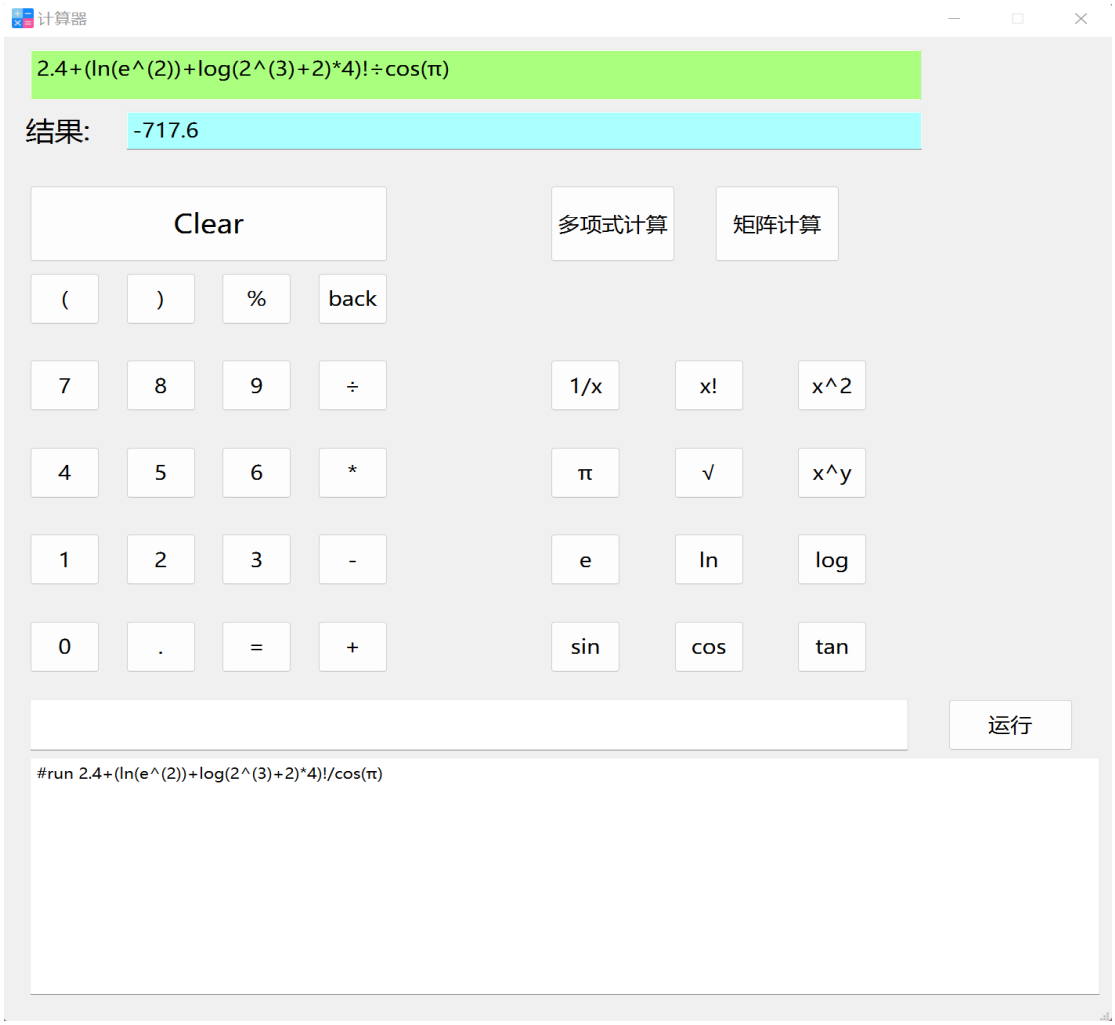
一、功能实现

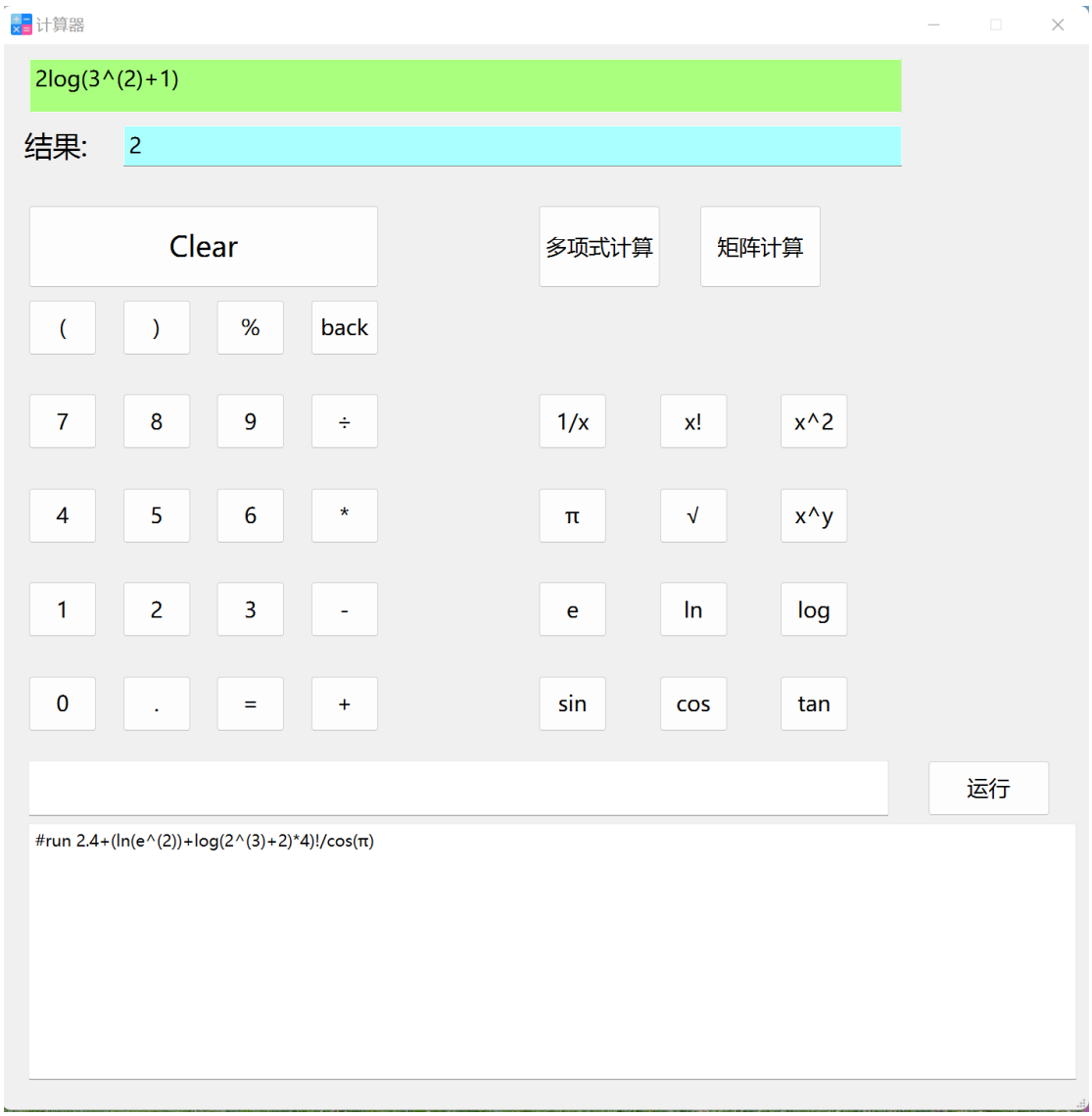
基础功能：实验要求的全部功能

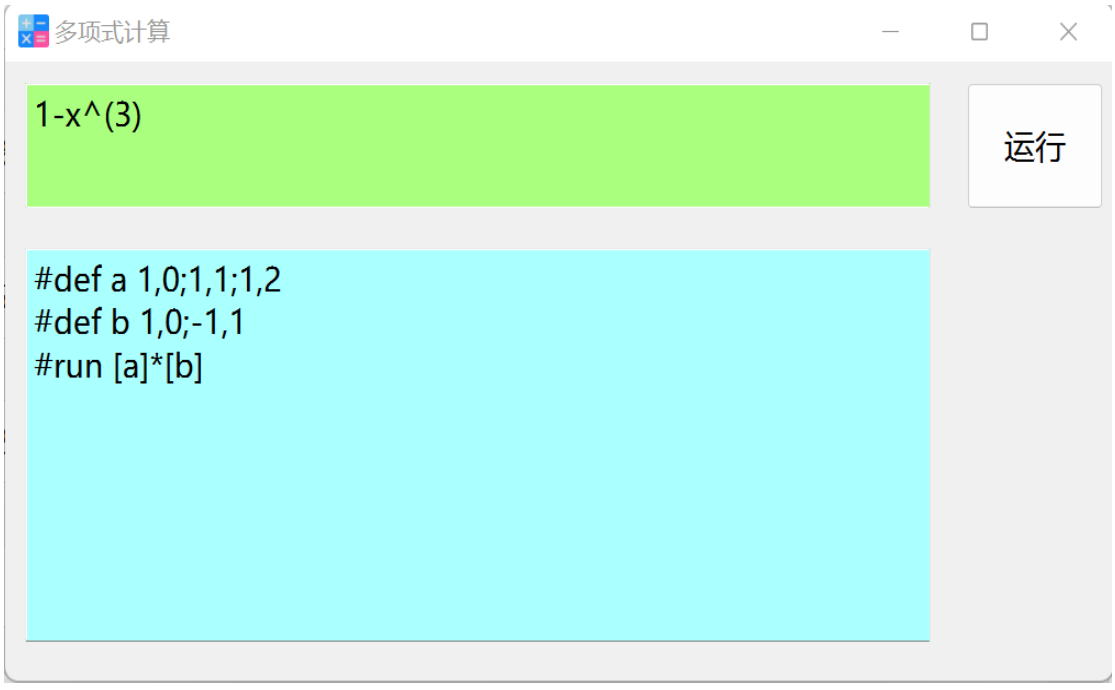
拓展功能：

1. Qt 可视化界面实现按键功能，且同时兼容命令行。
2. 允许较多高级运算功能（e、π、ln、log、根号、指数、倒数、阶乘、正弦、余弦、正切、百分号）。
3. 按键上允许省略部分表达式的乘号，如 2lg4,3sin(0.5)等。

二、运行截图




计算器
2log(3^(2)+1)
结果:
2
Clear
多项式计算
矩阵计算
(
)
%
back
7
8
9
÷
1/x
x!
x^2
4
5
6
*
π
√
x^y
1
2
3
-
e
ln
log
0
.
=
+
sin
cos
tan
运行
#run 2.4+(ln(e^(2))+log(2^(3)+2)*4)!/cos(π)


多项式计算
1-x^(3)
运行
#def a 1,0;1,1;1,2
#def b 1,0;-1,1
#run [a]*[b]

三、设计思路

（1）数据结构处理

1.自建链表 mylist.h 和栈 mystack.h，其中栈是链表的派生类。

2.自定义一个 expressions 类来表示所有能放入后缀表达式的符号，该类有两个私有成员：QString data 和 Buttons type。其中 Buttons 是一个枚举类型，包含了各种实际操作符的类型，如下代码：

```
enum Buttons
{
    Num,//数字
    Point,//点
```

```
    BinoOP,//双目操作符
    MonoOp,//单目运算符
    Special,//特殊符号
    LeftBra,//左括号
    RightBra,//右括号
    Percent,//百分号
    Index,//指数
    Back,//退格
    Equal,//等于
    Clear,//清除
};
```

变量和函数的结构详见 expression.h 代码，均为名称和值。

（2）计算思路

1.按键四则运算：

mainwindow.h 中主要定义了如下函数：

```
void buttons_connect();//绑定按钮
void onclicked(Buttons _type,QString _btn);//点击按钮
void translate();//中缀转后缀
bool checkBras();//检测中缀表达式中括号是否正确
int checkPoint(const QString& str);//返回数据中点的个数。
bool Operation();//运算
```

```
void write(const QString& newExprssion);//按键后对文本框的操作
bool checkLastChar();
void runcmd();//命令行
mylist <expressions> midexp;//中缀表达式
mylist <expressions> rpn;//后缀表达式
mylist <variants> varies;//变量
mylist <func> funcs;//函数
mystack <double> num;//数字
mystack <expressions> symbols;//符号
```

在点击按钮前会判断输入符号是否合法，否则点击后无效果。若输入符号合法，则根据按钮类型构造相应的 expressions 类并加入中缀表达式 midexp 中。当按下等号之后，先调用 translate 函数转换到 rpn 后缀表达式，最后用 operation 函数计算出结果。计算结果后需要 Clear 后才能重新进行计算。

2.命令行四则运算/单变量表达式运算/函数运算：

输入指令（详细见使用说明）后，先读取指令头部，再根据指令头处理后面的指令，#def、#let、#fuc 指令详见使用说明，读取 #run 指令后进行计算。

首先将对处理字符串，替换 sin，cos，tan，ln，log 等为 s，

c，t，n，g 单字符，接着读取@符号（函数标志）并将函数替换为相应表达式，然后遍历整个字符串，根据其类型构造不同的 expressions 并放入 midexp 中（以[]代表使用变量），之后调用检查函数检测括号匹配，通过检测后则调用 translate 函数转换到 rpn 后缀表达式，最后用 operation 函数计算出结果。

3.多项式计算：

首先用一个 struct 表示多项式的每一项，如下代码：

```
//表达式中每一项
struct chpol
{
    double coe;//系数
    int index;//指数
};
```

而表达式是由每一子项通过链表构成的，代码如下：

```
//表达式由每一项通过链表组成
struct pol
{
    mylist <chpol> fapol;//表达式
    QString name;//表达式名称
};
```

其它函数定义与四则运算一样，这里就把 pol 当做 Num 压入

midexp 中，整体运算过程与四则运算一致，不过针对多项式的各种计算函数被重载为了相应的操作。

4.矩阵计算：

操作格式详见说明手册，自定义了矩阵数据结构：

```
struct Matrix
{
    QString name;//名称
    int row;//行
    int col;//列
    double nums[10][10]={0.0};
};
```

整体操作流程同四则运算和多项式，针对矩阵的计算重载相应的计算函数。

四、参考资料

https://blog.csdn.net/ACdreamers/article/details/46431285

https://blog.csdn.net/weixin_44169941/article/details/107562037

https://blog.csdn.net/sjl20021006/article/details/117400261