# 面向对象程序设计

# 课程报告

**班级：**

**学号：**

**姓名：**

**原创性声明：**

本人声明报告中的内容和程序为本人独立完成，引用他人的文献、数据、图件、资料均已明确标注出。除标注内容外，不包含任何形式的他人成果，无侵权和抄袭行为，并愿意承担由此而产生后果。

作者签字：

时间：

**报告成绩**

指导教师签字：

目录

[面向对象程序设计 0](#_Toc9221)

[课程报告 0](#_Toc25697)

[一 课程设计题目与要求 3](#_Toc18463)

[二 需求分析 3](#_Toc14944)

[三 概要设计 4](#_Toc10082)

[（一） 工作流程设计 4](#_Toc126)

[（二） 系统模块设计 5](#_Toc4719)

[五 测试 11](#_Toc19391)

[六 结论 16](#_Toc15772)

[七 意见与建议 17](#_Toc3342)

[八 附录 17](#_Toc29770)

#### 一 课程设计题目与要求

仿照Windows系统的计算器软件，为教材第12.4 节通用计算器设计界面，开发一款实用的计算器软件。

除实现加减乘除以外，需仿照以下计算器实现应有的功能，并设计界面。



#### 二 需求分析

界面是人机交互的重要媒介，首先需要在Qt中为计算器设计界面，并将界面与实现各种功能的槽函数进行连接；其次，需要有合理的运算逻辑和正确的框架，以便于计算器可以快速准确的响应用户需求。

#### 三 概要设计

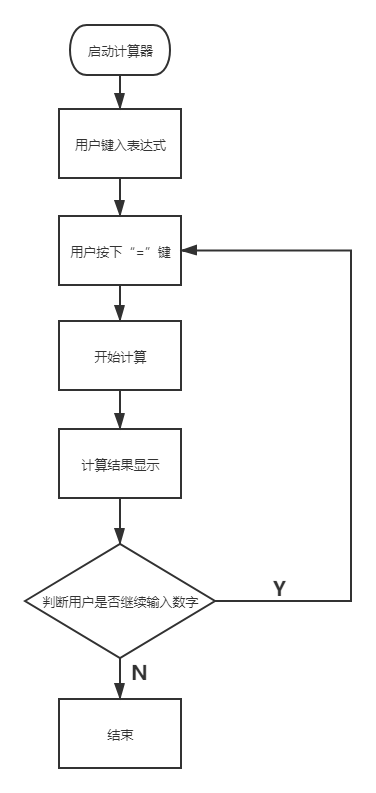
##### 工作流程设计

1.程序开始运行，界面弹出，界面的上方为显示表达式的textBrowser文本框，文本框下方有显示时间专用的textBrowser窗口，右侧的textBrowser文本框用于显示历史记录。按键排列在表达式窗口的下方，可在标准与科学界面之间切换。  


2.用户按照需求在界面中输入表达式，输入结束后点击等号进行计算。

3.计算器读取界面中的表达式，并调用相关函数进行计算，并将计算结果换行显示在表达式窗口中，并在历史窗口中记录。

4.计算结束后，用户可以继续输入以进行下一次计算，或选择退出界面。



##### 系统模块设计

1.运算符模块：定义基类Operator，并提供symbol（）、numOprand（）、Precedence（）函数用于访问运算符的符号、目数、优先级；并且提供了get（）纯虚函数，用于派生类的计算。其他运算符均继承于Operator类。

2.对象工厂模块：Factory 类在其嵌套结构体 RegisterClass 的构造函数中实现对静态成员变量 ms\_operator的类型注册功能，使得代码更加简洁，增强了程序的可扩展性，实现了代码重用。

3.输入模块：通过按键输入表达式，将表达式写入Dialog类的数据成员m\_show、m\_epx中，m\_show用于表达式在文本框中的显示，m\_epx用于计算。

4.显示模块：每一次按键都会出发相应的槽函数，每个槽函数中都有更新显示的语句，以便实时将输入的表达式显示在文本框中。每次计算结束后会执行显示语句，将运算结果放入表达式框和历史记录框中。

5.判断模块：定义issin（）、isNum（）、isopr（）函数，用于判断表达式中是否存在字母、操作数、运算符。

6.读取模块：定义readsin（）、readNum（）、readnum（）、readdot（）函数，用于读取表达式中的sin类型的字符串、正方向读取操作数、反方向读取操作数、读取运算符。

7.计算模块：先使用doIt（）函数进行操作数压栈、运算符压栈、运算符优先级判断等操作。读取与栈顶运算符相等目数的操作数，送入相应的get（）函数中进行运算。

8.特殊功能模块：CE可以实现操作数的退格、C可以实现表达式清零、退格按键可实现运算符的识别（如识别到sin退三格）。

9.保护模块：每个按键的槽函数中均设计了保护模块，以防止部分运算符的连续输入，造成计算器崩溃（如“+”只允许连续输入一个）。

10.时间显示模块：当按下“当前时间”按键时，自动调用时间显示函数。但时间不能自动更新，每点击一次按键则更新一次。

11.用户自定义模块：系统界面提供了三个radio bottom按钮，分别对应了红色、蓝色和黑色的字体，用户可以根据喜好选择字体颜色。

#### 四 详细设计 **（一）类的函数成员和数据成员设计及其他模块的实现**

###### **Node**

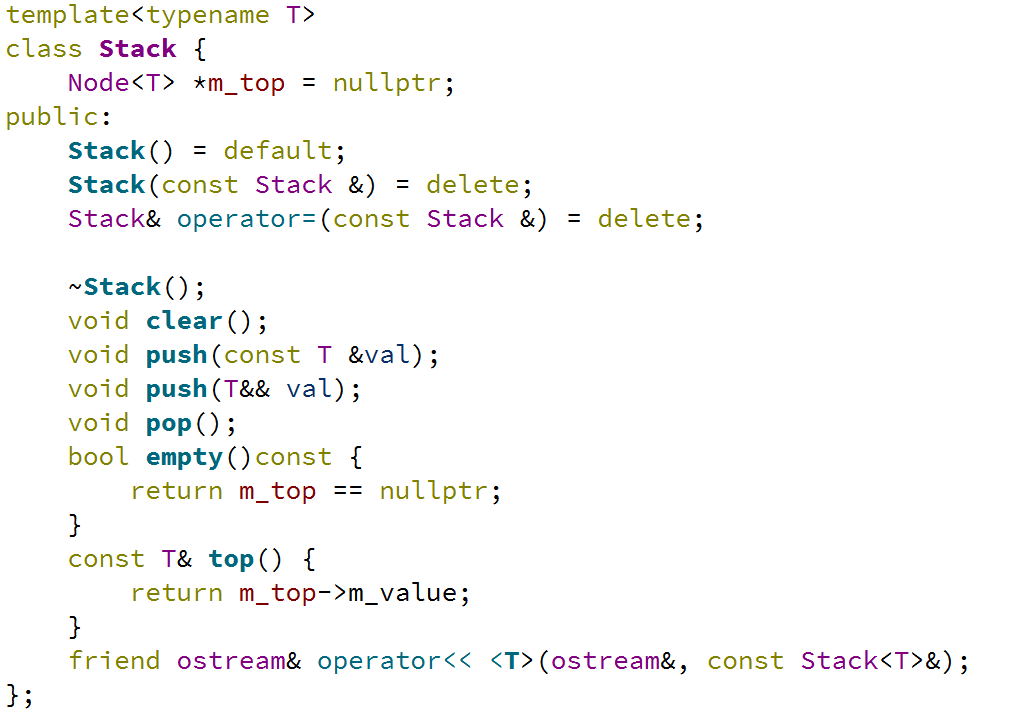


Node类为stack类的实现奠定了基础，Node类是栈中的节点，需要将stack声明为Node类的友元，以便stack类访问Node类。

由于后续函数需要用到unique\_ptr，而unique\_ptr不支持复制，故在Node类中提供了移动构造函数，使用move（）函数将左值转换为右值供unique\_ptr的初始化。含参构造函数用于左值构造。

Value（）、next（）函数是接口函数，用于在Node类的外部访问Node的数据成员。

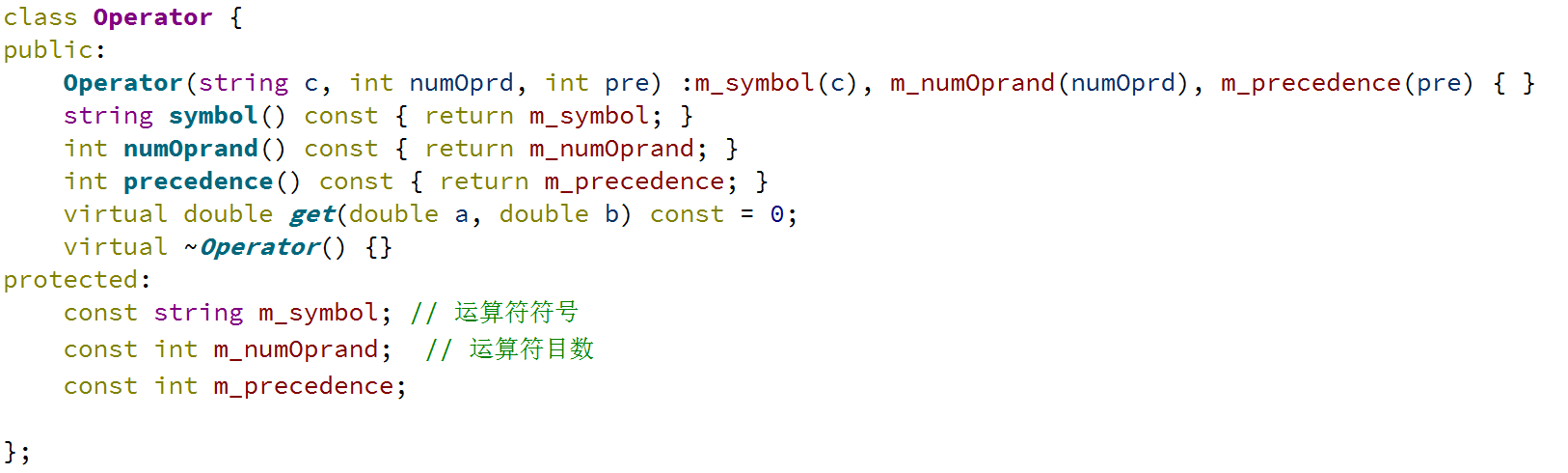
###### Stack



Stack类中有Node指针类型的数据成员m\_top。函数push（）、pop（）用于入栈和出栈操作，其中push(const T &val)用于左值入栈、push(T&& val)用于右值入栈操作。

empty（）函数用于判断栈是否为空；top（）函数是接口函数，便于外部访问stack类数据成员m\_top；clear（）函数用于清空栈。

###### Operator

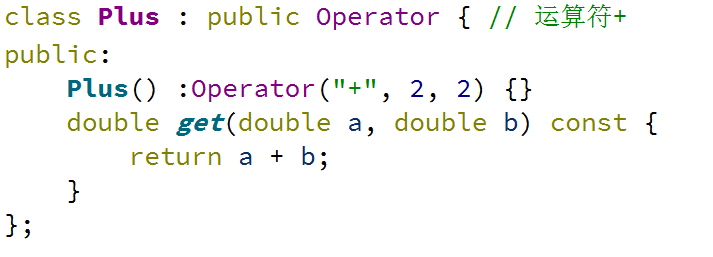


Operator类有m\_symbol、m\_numOprand、m\_precedence三个数据成员，分别表示运算符的符号、目数和优先级。

函数symbol（）、numOprand（）、precedence（）均为接口函数，用于在外部访问Operator类的数据成员。

get（）函数为纯虚函数，在Operator类中只负责接口声明，而接口的定义由派生类来负责。

Operator的派生类举例



Plus类是Operator的派生类，公有继承Operator。Plus类的符号为“+”，目数为2，优先级为2。

###### OperatorFactory



为了用户可以快速方便的注册运算符类，创建对象工厂Factory。对象工厂Factory通过嵌套类RegisterClass的构造函数可以实现对静态成员ms\_operator的类型注册功能。

语句#define REGISTRAR(T, Key) Factory::RegisterClass<T> reg\_##T(Key);通过宏REGISTRAR创建一个嵌套类全局对象，同时完成T的注册。##用来连接两个语言符号，并产生一个对象名。

Dialog



Dialog类本质上是12.4计算器中的calculator类，但是增加了槽函数，以实现传递信号的功能。

Isopr（）、issin（）都是判断表达式中字符类型的函数。Isopr（）使用right（）函数读取表达式最右端的字符是否为运算符，该函数主要的功能是防止运算符的连续输入，造成计算器崩溃。Issin（）用于判断字符是否为字母，内部通过迭代器实现表达式中字符的逐个判断，该函数一般在doIt（）函数中使用。

如图，在当按下“X”号之后，m\_show中加入的是“+”，而m\_epx中加入的是“\*”。这也是需要用不同的对象来储存表达式的原因。除此之外，槽函数中还有对应的显示语句用于m\_show的显示。

Slots中的函数均为槽函数，槽函数分为操作数槽函数、运算符槽函数、功能键槽函数。当操作数按键被按下，相关联的槽函数将被执行，并向m\_show和m\_epx中加入相应的数字（其中m\_show为显示在界面中的表达式，m\_epx为用于计算的表达式）；当运算符键被按下，相应的运算符会被加入m\_show和m\_epx中。部分运算符槽函数中，用于显示的符号和加入运算表达式的符号不同，如下图输入到m\_show的符号为“×”，而输入到m\_epx中的符号为“\*”，这是为给用户展现更加美观的表达式。当功能键被按下时，相应的功能函数将会被调用，如current\_time.toString将用于时间的设置。



keyPressEvent(QKeyEvent \*e)用于实现键盘按键与界面中虚拟按键的关联。该函数使用QKeyEvent类型的指针，指向电脑键盘的按键，并用if语句判断键盘的按键是否被按下。当电脑键盘的按键被按下时，界面中相应的虚拟按键会自动按下。用户在使用该系统时，可以自行选择电脑键盘输入或者虚拟键盘输入，提升用户体验。

doIt(QString &question)是该系统的核心函数，当用户按下“=”键，doIt（）函数将被调用。doIt函数执行时，先通过toStdString()将QString转化为字符串，接下来使用迭代器扫描表达式，使用isNum和issin函数进行判断，如果是数字则压进操作数栈，如果是符号则直接执行下一步，如果是字母则读取字母后进行下一步操作。获取运算符后，判断这个运算符的优先级是否大于运算符栈栈顶的优先级，若小于，则先计算之前的表达式再进行入栈操作；若大于，则直接入栈。等号的优先级最小，故扫描到等号时完成计算，且等号不入栈。运算结束后，运算结束标识符ended置1。

Calculate（）函数中两个空间存放操作数，分别为a[1]、a[0]。执行该函数时，操作数以此出栈，栈顶操作数放入a[0]，第二个操作数放入a[1]。完成操作数的出栈后，调用相应的get（）函数进行计算，计算结束后将结果压入操作数栈。

CE按键的实现：CE按键为单个操作数清除键，在CE的槽函数中，使用迭代器从表达式的最右端开始扫描，扫描一个清除一个，当扫描到运算符时停止。

退格按键的实现：在退格按键中使用right函数判断表达式最右端字符的类型，如果最右端为“sin”则删除最右端的三个元素，而不是删除“n”。这样的设计可以避免因为退格产生错误的表达式。

字体改色按键：使用函数setColor（）可以改变textBowser中字体的颜色。当改色按键被按下，相应的槽函数被执行，字体颜色发生相应的变化。

时间函数：加入头文件QTime，使用其中的封装函数即可显示当前时间，但是文本框无法动态显示，所以时间只能显示点击时刻的时间。

##### （二）界面设计



界面布局大致分为三部分，分别为表达式窗口、历史记录窗口和按键窗口。表达式窗口用于显示表达式和运算结果，历史记录窗口用于显示先前的表达式和运算结果，同时还具有清楚历史记录的功能。按键部分分为标准按键和科学按键两部分，使用Tab widget实现切换；标准部分由25个按键组成，除“=”号外均为栅格排列；科学部分由12个按键组成，均为栅格排列。时间窗口和改色按键位于表达式窗口和按键之间，当按下“当前时间”按钮，时间会显示在时间窗口中。

界面中所有的显示窗口均为textBrowser；按键中，除了改色按键为radio buttom外，其余按键均为pushbottom。

#### 五 测试

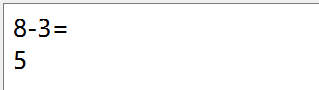
电脑键盘按键响应的测试不便于在报告中展示，将会在现场向老师展示。

##### 基本运算符测试

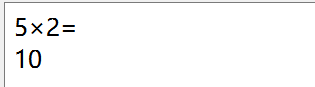
加法：



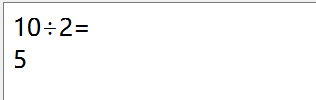
减法：



乘法：



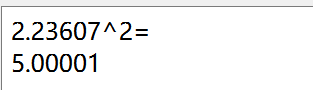
除法：



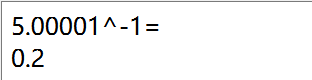
开方：



平方：



倒数：



求余：



Sin：



acos：



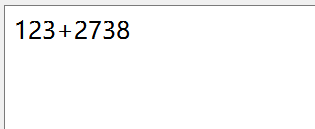
log：

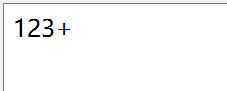


e^：

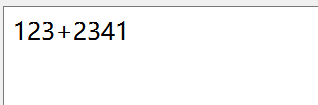


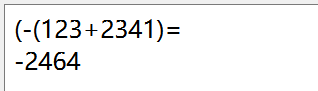
CE：





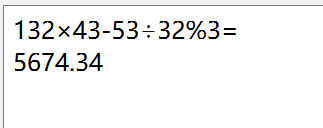
+/-：



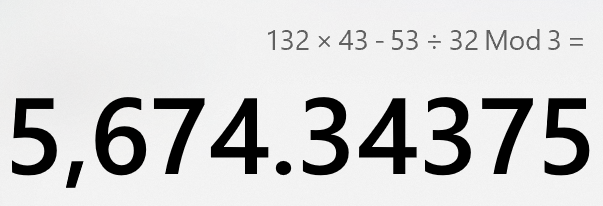


##### 复杂表达式测试

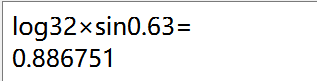
本计算器：



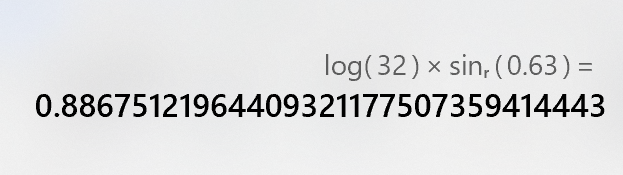
联想计算器：



本计算器：



联想计算器：



##### 时间功能测试



##### 字体变色功能测试

黑色：



红色：

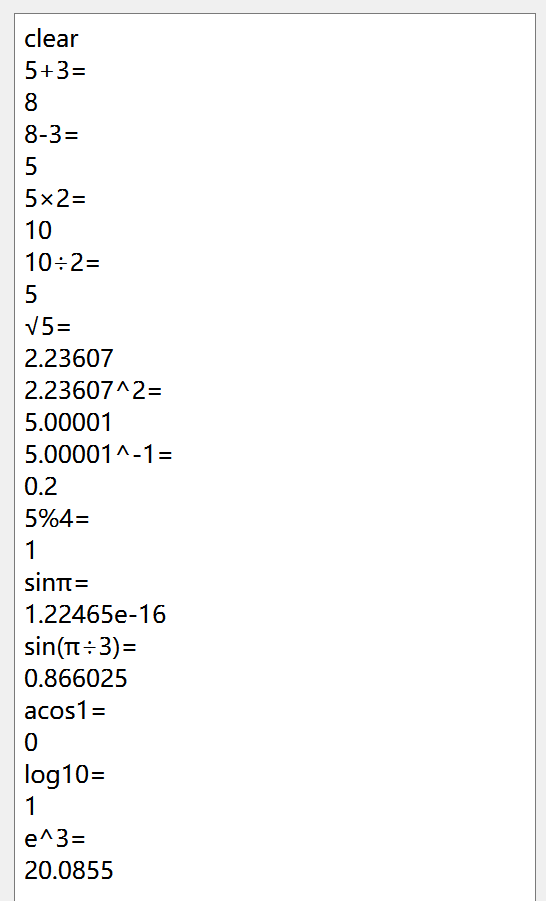


蓝色：

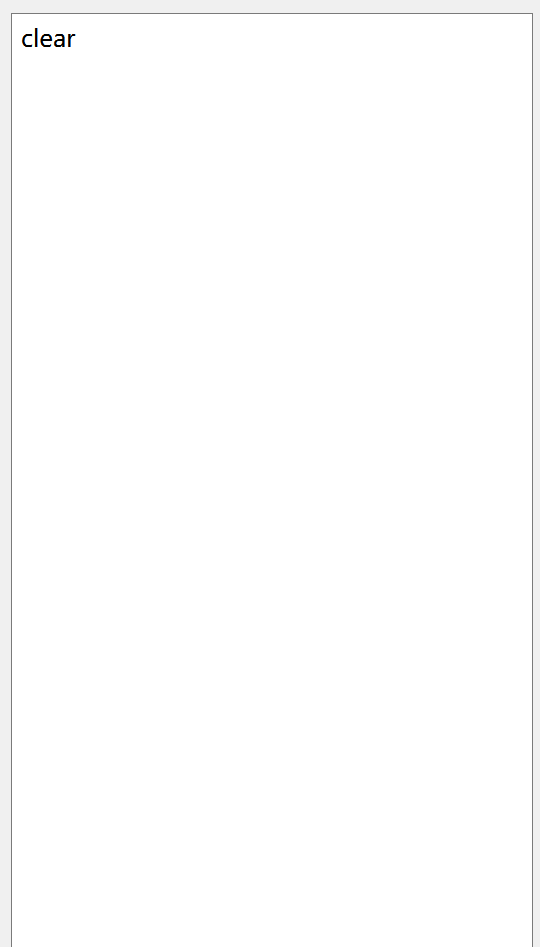


##### 历史记录功能测试

自动记录功能：



清空历史记录功能：



#### 六 结论

1. 系统开发总结  
    该系统主要通过vs2019设计程序，Qt设计界面实现。由于用户仅能通过界面和计算机交互，故设计一个简单明了的计算机界面极为重要。在界面设计的过程中，我逐渐深入了解了Qt可视化界面开发的知识，并且巩固了本学期的知识，可以说是收获颇丰。同时我也深深感受到了Qt的便利性，在Qt中有许多官方的视频和例程供我们学习，在遇到问题时可以按F1键进入帮助界面查看文献。

在设计系统时还需要考虑用户误操作的情况，例如在用户连续输入两个“+”或两个“.”后，doIt（）函数将会出现运算错误导致程序崩溃。故在编写程序时，在每个运算符的槽函数中都设置了保护语句，防止用户错误输入，以减少程序崩溃的情况。

该系统的内核程序主要是通过识别表达式中的操作数和运算符并实行入栈、出栈、计算等一系列操作实现的。其中操作符类定义了虚函数，可以使后续运算符的识别和运算更加便捷。在计算的过程中，左括号与其他运算符一样，需要压栈进行运算，如果左括号不进行压栈运算会导致括号前的字符串也开始计算，最终导致程序报错；而左括号的get（）函数仅仅只是读取数据并返回，相当于左括号出栈，数字栈不发生改变，不影响运算结果。  
 要保证系统正常运算，运算符优先级的设置是非常重要的。优先级的设置直接影响了运算的顺序，可能会导致部分输入不完整的表达式开始计算，导致运算符过少，无法正确计算。但一般这种问题可以通过加括号和加“0”解决。例如“+/-”运算符，我最开始的想法是在整个表达式的前段加入一个“-”，但运行后发现“-”是双目运算符，故doIt（）无法正常运算。最终的解决办法是，在表达式前段加入“0-（”，在表达式后端加入“）”。

另外，表达式中操作数和运算符的识别主要是通过迭代器it指针实现的，因此it指针的位置是正确读取表达式的关键。在该系统中，提供了正向读取函数readNum（）和反向读取函数readnum（），以便在需要时合理移动it指针的位置。

系统界面中提供了历史记录窗口和时间显示窗口。历史记录窗口可供用户在连续计算时查看先前的表达式及结果，并且提供了清空历史记录的功能。时间窗口可以显示时间，但时间不能自动更新，仅在按键被点击时更新一次。除此之外，系统界面还提供了字体改色的功能，可以迎合用户的喜好。

1. 不足之处  
    在编写程序时，对用户的错误操作可能考虑不周，部分表达式的计算可能无法实现，例如“3√3”的计算需要改成“3\*√3”才能正确运算。虽说该计算器已满足基本的计算需求，但是在遇到未能考虑到的表达式时，有可能会出现意想不到的错误。

按键和显示窗格的大小没能做到随Dialog窗口大小的改变而改变；界面的设计缺乏美感；时间不能自动更新，这些都是需要改进的。

这些仅仅是我在使用该系统的过程中发现的漏洞，对于操作符和运算符，可以通过排列组合组成无数种表达式，在这其中可能会出现更多的问题，这需要在日后使用该系统时不断的进行优化。

#### 七 意见与建议

##### 学习方法总结

在第一次动手编写Qt程序时，我感觉到无从下手。但是在观看了长江雨课堂的视频和ppt中推荐的视频后，对Qt的程序编写有了初步的理解。在编写计算器系统时，我能充分利用Qt自带的“帮助“模块、CSDN论坛上的资源，自行解决问题。在遇到实在无法解决的问题时，我会向学长请教，一般都会得到他们耐心的解答。

在设计计算器系统之初，每当我想到某种功能，就一股脑的写完，导致程序易崩溃且不能正确运行，最终需要重写功能，非常浪费时间。因此我认为，在编写程序时需要边写边调试，以确保程序按照自己的意愿进行。

##### 对课程的建议

长江雨课堂上上传的视频太过不清晰，建议老师可以上传清晰的视频；周四下午的大课时间过长，讲解的速度较快，同学们容易走神、跟不上。

#### 八 附录

**Stack.h**

#ifndef STACK\_H

#define STACK\_H

#include<string>

#include<map>

#include<functional>

#include<memory>

#include<utility>

#include<string>

#include<math.h>

#include<iostream>

using namespace std;

template<typename T> class **Stack**;

template<typename T> ostream& operator<<(ostream&, const Stack<T>&);

template<typename T>

class **Node** { // 只需要node可以支持move构造即可

friend class Stack<T>;

// 节点数据

T m\_value;

// 下一节点指针

Node \*m\_next = nullptr;

public:

**Node**() = default; // 默认构造函数

**Node**(const Node &rhs) = delete; // 禁用拷贝构造函数

Node & operator =(const Node &rhs) = delete; // 禁用拷贝构造算子

**Node**(const T &val) :m\_value(val) {} // 含参构造函数

**Node**(T &&val) :m\_value(std::move(val)) { } // 含参移动构造函数

const T& **value**() const { return m\_value; }

Node \***next**() { return m\_next; }

};

template<typename T>

class **Stack** {

Node<T> \*m\_top = nullptr;

public:

**Stack**() = default;

**Stack**(const Stack &) = delete;

Stack& operator=(const Stack &) = delete;

~**Stack**();

void **clear**();

void **push**(const T &val);

void **push**(T&& val); // 我重载了呀? yes, I have overloaded it, and it's enough. Something wrong happend somewhere else

void **pop**();

bool **empty**()const {

return m\_top == nullptr;

}

const T& **top**() {

return m\_top->m\_value;

}

friend ostream& operator<< <**T**>(ostream&, const Stack<T>&);

};

template<typename T>

Stack<T>::~**Stack**() {

clear();

}

template<typename T>

void Stack<T>::**push**(const T &val) {

Node<T> \*node = new Node<T>(val);

node->m\_next = m\_top;

m\_top = node;

}

// 支持move的push函数

template<typename T>

void Stack<T>::**push**(T &&val) {

Node<T> \*node = new Node<T>(move(val));

node->m\_next = m\_top;

m\_top = node;

}

template<typename T>

void Stack<T>::**clear**() {

Node<T> \*p = nullptr;

while (m\_top != nullptr) {

p = m\_top;

m\_top = m\_top->m\_next;

delete p;

}

}

template<typename T>

void Stack<T>::**pop**() {

Node<T> \*p = m\_top;

m\_top = m\_top->m\_next;

delete p;

}

template<typename T>

ostream& operator<<(ostream &os, const Stack<T>& s) {

Node<T> \*p = s.m\_top;

while (p != nullptr) {

os << p->value() << " ";

p = p->next();

}

return os;

}

#endif // STACK\_H

**Operator.h**

#ifndef OPERATOR\_H

#define OPERATOR\_H

#include"stack.h"

#include"dialog.h"

#include"OperatorFactory.h"

using namespace std;

class **Operator** {

public:

**Operator**(string c, int numOprd, int pre) :m\_symbol(c), m\_numOprand(numOprd), m\_precedence(pre) { }

string **symbol**() const { return m\_symbol; }

int **numOprand**() const { return m\_numOprand; }

int **precedence**() const { return m\_precedence; }

virtual double ***get***(double a, double b) const = 0;

virtual ~***Operator***() {}

protected:

const string m\_symbol; // 运算符符号

const int m\_numOprand; // 运算符目数

const int m\_precedence;

};

class **Plus** : public Operator { // 运算符+

public:

**Plus**() :Operator("+", 2, 2) {}

double ***get***(double a, double b) const {

return a + b;

}

};

class **Minus** :public Operator { // 运算符-

public:

**Minus**() :Operator("-", 2, 2) {}

double ***get***(double a, double b) const {

return a - b;

}

};

class **Multiply** :public Operator { // 运算符\*

public:

**Multiply**() :Operator("\*", 2, 3) {}

double ***get***(double a, double b) const {

return a\*b;

}

};

class **Divide** :public Operator { // 运算符/

public:

**Divide**() :Operator("/", 2, 3) {}

double ***get***(double a, double b) const {

return a / b;

}

};

class **mode** :public Operator { // 运算符/

public:

**mode**() :Operator("%", 2, 3) {}

double ***get***(double a, double b) const {

int c=a/b;

return a-c\*b;

}

};

class **power** :public Operator { // 运算符/

public:

**power**() :Operator("^", 2, 5) {}

double ***get***(double a, double b) const {

return pow(a, b);

}

};

class **dealsqrt** :public Operator { // 运算符/

public:

**dealsqrt**() :Operator("sqrt", 1, 5) {}

double ***get***(double a, double b) const {

return sqrt(b);

}

};

class **dealsin** :public Operator { // 运算符/

public:

**dealsin**() :Operator("sin", 1, 6) {}

double ***get***(double a, double b) const {

return sin(b);

}

};

class **dealcos** :public Operator { // 运算符/

public:

**dealcos**() :Operator("cos", 1, 6) {}

double ***get***(double a, double b) const {

return cos(b);

}

};

class **dealtan** :public Operator { // 运算符/

public:

**dealtan**() :Operator("tan", 1, 6) {}

double ***get***(double a, double b) const {

return tan(b);

}

};

class **dealasin** :public Operator { // 运算符/

public:

**dealasin**() :Operator("asin", 1, 6) {}

double ***get***(double a, double b) const {

return asin(b);

}

};

class **dealacos** :public Operator { // 运算符/

public:

**dealacos**() :Operator("acos", 1, 6) {}

double ***get***(double a, double b) const {

return acos(b);

}

};

class **dealatan** :public Operator { // 运算符/

public:

**dealatan**() :Operator("atan", 1, 6) {}

double ***get***(double a, double b) const {

return atan(b);

}

};

class **kleft** :public Operator { // 运算符#

public:

**kleft**():Operator("(", 1, 1) {}

double ***get***(double a, double b) const {

return b; // 本身#不需要进行任何运算, 但是仍然需要返回值

}

};

class **kright** :public Operator { // 运算符#

public:

**kright**():Operator(")", 1, 2) {}

double ***get***(double a, double b) const {

return b; // 本身#不需要进行任何运算, 但是仍然需要返回值

}

};

class **negative** :public Operator { // 运算符#

public:

**negative**():Operator("$", 1, 2) {}

double ***get***(double a, double b) const {

return -b; // 本身#不需要进行任何运算, 但是仍然需要返回值

}

};

class **bln** :public Operator { // 运算符#

public:

**bln**():Operator("ln", 1, 5) {}

double ***get***(double a, double b) const {

return log(b); // 本身#不需要进行任何运算, 但是仍然需要返回值

}

};

class **blog** :public Operator { // 运算符#

public:

**blog**():Operator("log", 1, 5) {}

double ***get***(double a, double b) const {

return log10(b); // 本身#不需要进行任何运算, 但是仍然需要返回值

}

};

class **reciprocal** :public Operator { // 运算符#

public:

**reciprocal**():Operator("1/x", 1, 5) {}

double ***get***(double a, double b) const {

return 1/b; // 本身#不需要进行任何运算, 但是仍然需要返回值

}

};

class **factorial** :public Operator { // 运算符#

public:

**factorial**():Operator("!", 1, 3) {}

double ***get***(double a, double b) const {

int result=1;

for(int i=1;i<=b;i++)

{

result\*=i;

}

return result;

}

};

class **dealexp** :public Operator { // 运算符#

public:

**dealexp**():Operator("em", 1, 5) {}

double ***get***(double a, double b) const {

return exp(b);

}

};

class **Hash** :public Operator { // 运算符#

public:

**Hash**() :Operator("#", 1, 0) {}

double ***get***(double a, double b) const {

return b; // 本身#不需要进行任何运算, 但是仍然需要返回值

}

};

class **Equal** :public Operator { //表达式结束符=

public:

**Equal**() :Operator("=", 2, -1) {}

double ***get***(double a, double b) const {

return a;

}

};

#endif // OPERATOR\_H

**OperatorFactory.h**

#ifndef OPERATORFACTORY\_H

#define OPERATORFACTORY\_H

#include"Operator.h"

#include"dialog.h"

#define REGISTRAR(T, Key) Factory::RegisterClass<T> reg\_##T(Key);

using namespace std;

class **Factory**{

public:

template<typename T>

struct **RegisterClass** {

**RegisterClass**(string opr) {

Factory::ms\_operator.emplace(opr, []{return make\_unique<T>();});

}

};

static unique\_ptr<Operator> **create**(string opr) {

auto it = ms\_operator.find(opr);

if (it != ms\_operator.end())

return it->second();

}

//private:

static map<string, function<unique\_ptr<Operator>()>> ms\_operator; // 存储已注册运算符名及对应构建函数指针的map

};

#endif // OPERATORFACTORY\_H

**dialog.h**

#ifndef DIALOG\_H

#define DIALOG\_H

#include <QDialog>

#include<OperatorFactory.h>

#include"ui\_dialog.h"

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **Dialog**; }

QT\_END\_NAMESPACE

class **Dialog** : public QDialog

{

Q\_OBJECT

public:

**Dialog**(QWidget \*parent = nullptr);

~***Dialog***();

int **precedence**(const QString& s) const;//获取优先级,没有用到

string **readsin**(string::iterator& it);

bool **issin**(string::iterator& c) const {

return \*c >= 'a' && \*c <= 'z';

}

bool **isopr**(QString s);

void **calculate**();

void ***keyPressEvent***(QKeyEvent \*e);

private slots:

void **on\_num1\_clicked**();

void **on\_num2\_clicked**();

void **on\_num3\_clicked**();

void **on\_num4\_clicked**();

void **on\_num5\_clicked**();

void **on\_num6\_clicked**();

void **on\_num7\_clicked**();

void **on\_num8\_clicked**();

void **on\_num9\_clicked**();

void **on\_num0\_clicked**();

void **on\_point\_clicked**();

void **on\_plus\_clicked**();

void **on\_pi\_clicked**();

void **on\_sin\_clicked**();

void **on\_cos\_clicked**();

void **on\_tan\_clicked**();

void **on\_asin\_clicked**();

void **on\_acos\_clicked**();

void **on\_atan\_clicked**();

void **on\_lkuo\_clicked**();

void **on\_rkuo\_clicked**();

void **on\_gantan\_clicked**();

void **on\_min\_clicked**();

void **on\_mul\_clicked**();

void **on\_div\_clicked**();

void **on\_qiuyu\_clicked**();

void **on\_pinfang\_clicked**();

void **on\_kaifang\_clicked**();

void **on\_back\_clicked**();

void **on\_cle\_clicked**();

void **on\_equ\_clicked**();

void **on\_clc\_clicked**();

void **on\_ce\_clicked**();

void **on\_zhengfu\_clicked**();

void **on\_qlog\_clicked**();

void **on\_qln\_clicked**();

void **on\_daoshu\_clicked**();

void **on\_em\_clicked**();

void **on\_time\_clicked**();

void **on\_red\_clicked**();

void **on\_blue\_clicked**();

void **on\_black\_clicked**();

private:

Stack<double> m\_num; // 数字栈

Stack<unique\_ptr<Operator>> m\_opr; // 运算符栈

Ui::Dialog \*ui;

QString m\_epx;

QString m\_show;

int ended;

double **readNum**(string::iterator &it); // 读取操作数

double **readnum**(string::iterator& it);

bool **readdot**(QString s);

bool **isNum**(string::iterator &c) { // 判断字符是否为数字

return (\*c >= '0'&&\*c <= '9' )|| \*c == '.'||\*c=='p'||\*c=='i';

}

double **doIt**(QString &question);

};

#endif // DIALOG\_H

**dialog.cpp**

#include "dialog.h"

#include "ui\_dialog.h"

#include<QTime>

#include<QKeyEvent>

Dialog::**Dialog**(QWidget \*parent)

: QDialog(*parent*)

, ui(new Ui::Dialog)

{

ui->setupUi(this);

ended=0;

m\_opr.push(make\_unique<Hash>());

}

Dialog::~***Dialog***()

{

delete ui;

}

void Dialog::**on\_num1\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="1";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_num2\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="2";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_num3\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="3";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_num4\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="4";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_num5\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="5";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_num6\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="6";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_num7\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="7";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_num8\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="8";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_num9\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="9";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_num0\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="0";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_point\_clicked**()

{

QString btm=".";

if(readdot(m\_epx)){

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

}

void Dialog::**on\_plus\_clicked**()

{

QString btm="+";

if(readdot(m\_epx)){

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

}

void Dialog::**on\_pi\_clicked**()

{

if(ended==1){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

}

QString btm="(pi)";

QString sbtm="π";

m\_show+=sbtm;

m\_epx+=btm;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_sin\_clicked**()

{

QString btm="sin";

if(ended==1){

m\_show.push\_front(btm);

m\_epx.push\_front(btm);

}

else{

m\_show+=btm;

m\_epx+=btm;

}

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_cos\_clicked**()

{

QString btm="cos";

if(ended==1){

m\_show.push\_front(btm);

m\_epx.push\_front(btm);

}

else{

m\_show+=btm;

m\_epx+=btm;

}

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_tan\_clicked**()

{

QString btm="tan";

if(ended==1){

m\_show.push\_front(btm);

m\_epx.push\_front(btm);

}

else{

m\_show+=btm;

m\_epx+=btm;

}

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_asin\_clicked**()

{

QString btm="asin";

if(ended==1){

m\_show.push\_front(btm);

m\_epx.push\_front(btm);

}

else{

m\_show+=btm;

m\_epx+=btm;

}

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_acos\_clicked**()

{

QString btm="acos";

if(ended==1){

m\_show.push\_front(btm);

m\_epx.push\_front(btm);

}

else{

m\_show+=btm;

m\_epx+=btm;

}

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_atan\_clicked**()

{

QString btm="atan";

if(ended==1){

m\_show.push\_front(btm);

m\_epx.push\_front(btm);

}

else{

m\_show+=btm;

m\_epx+=btm;

}

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_lkuo\_clicked**()

{

QString btm="(";

if(ended==1){

m\_show.clear();

m\_epx.clear();

m\_num.clear();

}

m\_epx+=btm;

m\_show+=btm;

ui->textBrowser->setText(m\_show);

ended=0;

}

void Dialog::**on\_rkuo\_clicked**()

{

QString btm=")";

if(ended==1){

m\_show.clear();

m\_epx.clear();

m\_num.clear();

}

m\_epx+=btm;

m\_show+=btm;

ui->textBrowser->setText(m\_show);

ended=0;

}

void Dialog::**on\_gantan\_clicked**()

{

QString btm="!";

m\_show+=btm;

m\_epx+=btm;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_min\_clicked**()

{

if(readdot(m\_epx)){

QString btm="-";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

}

void Dialog::**on\_mul\_clicked**()

{

if(readdot(m\_epx)){

QString btm="\*";

QString sbtm="×";

m\_show+=sbtm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

}

void Dialog::**on\_div\_clicked**()

{

QString btm="/";

QString sbtm="÷";

if(readdot(m\_epx)){

m\_show+=sbtm;

ended=0;

m\_epx+=btm;

ui->textBrowser->setText(m\_show);

}

}

void Dialog::**on\_qiuyu\_clicked**()

{

if(readdot(m\_epx)){

QString btm="%";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

}

void Dialog::**on\_pinfang\_clicked**()

{

if(readdot(m\_epx)){

QString btm="^";

m\_show+=btm;

m\_epx+=btm;

ended=0;

ui->textBrowser->setText(m\_show);

}

}

void Dialog::**on\_kaifang\_clicked**()

{

QString btm="sqrt";

QString sbtm="√";

if(ended==1){

m\_show.push\_front(sbtm);

m\_epx.push\_front(btm);

}

else{

m\_show+=sbtm;

m\_epx+=btm;

}

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_back\_clicked**()

{

if(m\_show.right(1)=="π"){

m\_show.chop(1);

m\_epx.chop(4);

}

else if(m\_show.right(4)=="asin"||m\_show.right(4)=="acos"||m\_show.right(4)=="atan"){

m\_show.chop(4);

m\_epx.chop(4);

}

else if(m\_show.right(3)=="sin"||m\_show.right(3)=="cos"||m\_show.right(3)=="tan"||m\_show.right(3)=="log"){

m\_show.chop(3);

m\_epx.chop(3);

}

else if(m\_show.right(3)=="sin"){

m\_show.chop(3);

m\_epx.chop(2);

}

else if(m\_show.right(2)=="ln"||m\_show.right(2)=="e^"){

m\_show.chop(2);

m\_epx.chop(2);

}

else{

m\_show.chop(1);

m\_epx.chop(1);

}

ui->textBrowser->setText(m\_show);

//ui->textabc->setText(m\_epx);

}

void Dialog::**on\_cle\_clicked**()

{

m\_epx.clear();

m\_show.clear();

m\_num.clear();

//m\_opr.clear();

ui->textBrowser->setText("0");

}

double Dialog::**doIt**(QString &epx) {

unique\_ptr<Operator> oo;

ui->his\_show->append(m\_show);

string exp;

exp= epx.toStdString();

for (auto it = exp.begin(); it != exp.end();) {

// 如果是操作数, 压栈之

if (isNum(*it*))

m\_num.push(readNum(*it*));

else {// 如果是运算符, 压栈或计算

// 将readOpr得到的string转化为Operator

if(issin(*it*))

oo=Factory::create(readsin(*it*));

else

{

string s;

s+=\*it++;

oo=Factory::create(s);

}

// 如果栈顶优先级高, 则计算

if(oo->symbol()!="(")

{

while (oo->precedence() <= m\_opr.top()->precedence()) {

if (m\_opr.top()->symbol() == "#")

break;

calculate();

}

}

// "="从不入栈

if (oo->symbol() != "=")

m\_opr.push(std::move(*oo*));

}

}

double result = m\_num.top();

double a=m\_num.top();

ui->textBrowser->append(QString::number(a));

ui->his\_show->append(QString::number(a));

epx.clear();

m\_show.clear();

m\_epx+=QString::number(a);

m\_show+=QString::number(a);

return result;

}

double Dialog::**readNum**(string::iterator& it){

string t;

while (isNum(*it*))

t += \*it++;

if (t == "pi")

{

return acos(-1);

}

return stod(t);

}

double Dialog::**readnum**(string::iterator& it){

string t;

while (isNum(*it*))

t += \*it--;

if (t == "ip")

{

return acos(-1);

}

return stod(t);

}

bool Dialog::**readdot**(QString s){

QString t;

string z;

QString a;

t=s;

z=t.toStdString();

a=m\_epx.right(1);

if(a=="."||a=="+"||a=="-"||a=="\*"||a=="/"||a=="^"){

return 0;

}

else if(m\_show.length()==0){

return 0;

}

else{

return 1;

}

}

bool Dialog::**isopr**(QString s){

QString t;

string z;

string a;

t=s;

z=t.toStdString();

a=\*(z.end()-1);

if(a=="√"||a=="+"||a=="-"||a=="\*"||a=="/"||a=="^"){

return 0;

}

else{

return 1;

}

}

string Dialog::**readsin**(string::iterator& it){

string t;

while (issin(*it*)&&t!="em"&&t!="sin"&&t!="cos"&&t!="sqrt"&&t!="log"&&t!="ln"&&t!="reciprocal"&&t!="asin"&&t!="acos"&&t!="atan"&&t!="tan")

t += \*it++;

return t;

}

void Dialog::**calculate**() {

// 操作数出栈并传入Operator进行计算, 在此只考虑单目和双目运算符

double a[2] = { 0 };

for (auto i = 0; i < m\_opr.top()->numOprand(); ++i) {

a[i] = m\_num.top();

m\_num.pop();

}

m\_num.push(m\_opr.top()->*get*(a[1], a[0]));

m\_opr.pop();

}

void Dialog::**on\_equ\_clicked**()

{

QString btm="=";

m\_epx+=btm;

m\_show+=btm;

ui->textBrowser->setText(m\_show);

doIt(*m\_epx*);

ended=1;

}

void Dialog::**on\_clc\_clicked**()

{

QString t="clear";

ui->his\_show->setText(t);

}

void Dialog::**on\_ce\_clicked**()

{

string exp;

exp= m\_epx.toStdString();

for(auto it=(exp.end()-1);;){

if((exp.end()-1)==exp.begin()){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

break;

}

if(isNum(*it*)){

--it;

m\_show.chop(1);

m\_epx.chop(1);

if(it==exp.begin()){

m\_epx.clear();

m\_show.clear();

m\_num.clear();

break;

}

}

else{

break;

}

}

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_zhengfu\_clicked**()

{

m\_show.push\_front("(-(");

m\_epx.push\_front("0-(");

m\_show.push\_back(")");

m\_epx.push\_back(")");

ended=0;

ui->textBrowser->setText(m\_show);

ui->textabc->setText(m\_epx);

}

void Dialog::**on\_qlog\_clicked**()

{

QString btm="log";

if(ended==1){

m\_show.push\_front(btm);

m\_epx.push\_front(btm);

}

else{

m\_epx+=btm;

m\_show+=btm;

}

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_qln\_clicked**()

{

QString btm="ln";

if(ended==1){

m\_show.push\_front(btm);

m\_epx.push\_front(btm);

}

else{

m\_epx+=btm;

m\_show+=btm;

}

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_daoshu\_clicked**()

{

QString btm="ds";

m\_epx+=btm;

m\_show+="^-1";

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_em\_clicked**()

{

QString btm="em";

if(ended==1){

ended=0;

m\_show.push\_front("e^");

m\_epx.push\_front(btm);

}

else{

m\_epx+=btm;

m\_show+="e^";

}

ended=0;

ui->textBrowser->setText(m\_show);

}

void Dialog::**on\_time\_clicked**()

{

QDateTime current\_time = QDateTime::currentDateTime();

//显示时间，格式为：年-月-日 时：分：秒 周几

QString StrCurrentTime = current\_time.toString("yyyy-MM-dd hh:mm:ss ddd");

//时间显示格式可自由设定，如hh:mm ddd 显示 时：分 周几

//具体格式见函数QString QDateTime::​toString(const QString & format) const

ui->textabc->setText(StrCurrentTime);

}

void Dialog::***keyPressEvent***(QKeyEvent \*e){

if(e->key()==Qt::Key\_0){

Dialog::on\_num0\_clicked();

}

else if(e->key()==Qt::Key\_1){

Dialog::on\_num1\_clicked();

}

else if(e->key()==Qt::Key\_2){

Dialog::on\_num2\_clicked();

}

else if(e->key()==Qt::Key\_3){

Dialog::on\_num3\_clicked();

}

else if(e->key()==Qt::Key\_4){

Dialog::on\_num4\_clicked();

}

else if(e->key()==Qt::Key\_5){

Dialog::on\_num5\_clicked();

}

else if(e->key()==Qt::Key\_6){

Dialog::on\_num6\_clicked();

}

else if(e->key()==Qt::Key\_7){

Dialog::on\_num7\_clicked();

}

else if(e->key()==Qt::Key\_8){

Dialog::on\_num8\_clicked();

}

else if(e->key()==Qt::Key\_9){

Dialog::on\_num9\_clicked();

}

else if(e->key()==Qt::Key\_A){

Dialog::on\_plus\_clicked();

}

else if(e->key()==Qt::Key\_Minus){

Dialog::on\_min\_clicked();

}

else if(e->key()==Qt::Key\_M){

Dialog::on\_mul\_clicked();

}

else if(e->key()==Qt::Key\_Slash){

Dialog::on\_div\_clicked();

}

else if(e->key()==Qt::Key\_Period){

Dialog::on\_point\_clicked();

}

else if(e->key()==Qt::Key\_Backspace){

Dialog::on\_back\_clicked();

}

else if(e->key()==Qt::Key\_Equal){

Dialog::on\_equ\_clicked();

}

else if(e->key()==Qt::Key\_T){

Dialog::on\_time\_clicked();

}

else if(e->key()==Qt::Key\_Y){

Dialog::on\_qiuyu\_clicked();

}

else if(e->key()==Qt::Key\_AsciiCircum){

Dialog::on\_pinfang\_clicked();

}

else if(e->key()==Qt::Key\_G){

Dialog::on\_gantan\_clicked();

}

else if(e->key()==Qt::Key\_R){

Dialog::on\_rkuo\_clicked();

}

else if(e->key()==Qt::Key\_L){

Dialog::on\_lkuo\_clicked();

}

}

void Dialog::**on\_red\_clicked**()

{

QPalette plet=ui->textBrowser->palette();

plet.setColor(QPalette::Text,Qt::red);

ui->textBrowser->setPalette(plet);

ui->his\_show->setPalette(plet);

ui->textabc->setPalette(plet);

}

void Dialog::**on\_blue\_clicked**()

{

QPalette plet=ui->textBrowser->palette();

plet.setColor(QPalette::Text,Qt::blue);

ui->textBrowser->setPalette(plet);

ui->his\_show->setPalette(plet);

ui->textabc->setPalette(plet);

}

void Dialog::**on\_black\_clicked**()

{

QPalette plet=ui->textBrowser->palette();

plet.setColor(QPalette::Text,Qt::black);

ui->textBrowser->setPalette(plet);

ui->his\_show->setPalette(plet);

ui->textabc->setPalette(plet);

}

**main.cpp**

#include "dialog.h"

#include"OperatorFactory.h"

#include"Operator.h"

#include <QApplication>

using namespace std;

map<string, function<unique\_ptr<Operator>()>> Factory::ms\_operator; // 定义存储ObjectCreator的静态map

// 使用宏对Operator进行注册

REGISTRAR(**Plus**, "+");

REGISTRAR(**Minus**, "-");

REGISTRAR(**Multiply**, "\*");

REGISTRAR(**Divide**, "/");

REGISTRAR(**Equal**, "=");

REGISTRAR(**mode**, "%");

REGISTRAR(**power**, "^");

REGISTRAR(**dealsqrt**, "sqrt");

REGISTRAR(**dealsin**, "sin");

REGISTRAR(**dealcos**, "cos");

REGISTRAR(**dealtan**, "tan");

REGISTRAR(**dealasin**, "asin");

REGISTRAR(**dealacos**, "acos");

REGISTRAR(**dealatan**, "atan");

REGISTRAR(**kleft**, "(");

REGISTRAR(**kright**, ")");

REGISTRAR(**negative**, "$");

REGISTRAR(**bln**, "ln");

REGISTRAR(**blog**, "log");

REGISTRAR(**reciprocal**, "ds");

REGISTRAR(**factorial**, "!");

REGISTRAR(**dealexp**, "em");

int **main**(int argc, char \*argv[])

{

QApplication a(*argc*, *argv*);

Dialog w;

w.show();

return a.exec();

}

#### 报告打印要求

A4双面打印，正文用五号字体，但涉及到程序代码的地方请用小五打印。