面向对象程序设计

课程设计报告

**班级：231202**

**学号：20201000128**

**姓名：刘瑾瑾**

**2022.04.25**

1. **课程设计内容和要求**

**(一)设计内容：**

基于第八章实现的计算器，参照Windows中的科学计算器，利用Qt实现计算器的界面。

**（二）设计要求：**

1、计算器界面清楚，功能完整，交互良好；

2、支持加、减、乘、除、求余、括号、次幂等功能；

3、支持sin、cos、tan、sqrt等功能；

4、能够对异常输入进行处理，例如未实现的运算功能、不匹配的括号输入等。

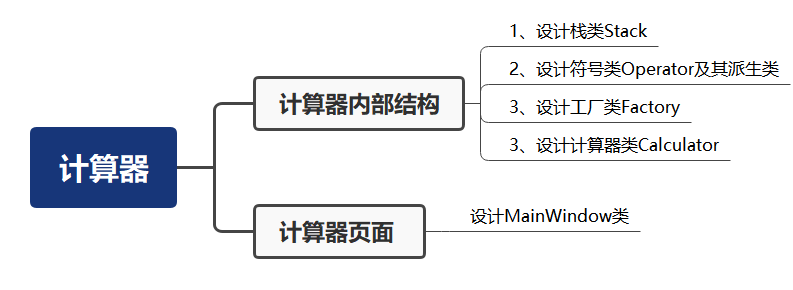
1. **需求分析**

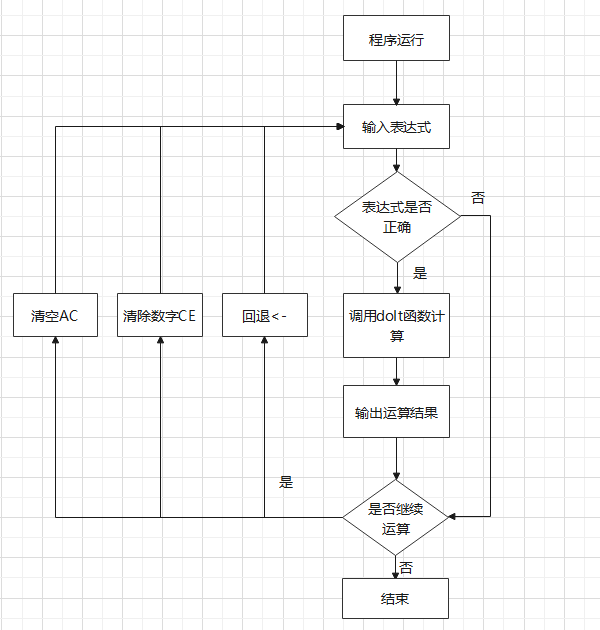
1、问题描述：基于第八章实现的计算器，参照Windows中的科学计算器，利用Qt实现计算器的界面，可以实现相关的运算功能，对异常运算进行处理，界面清楚，功能完整，交互良好。

2、系统环境：Qt5.14.2

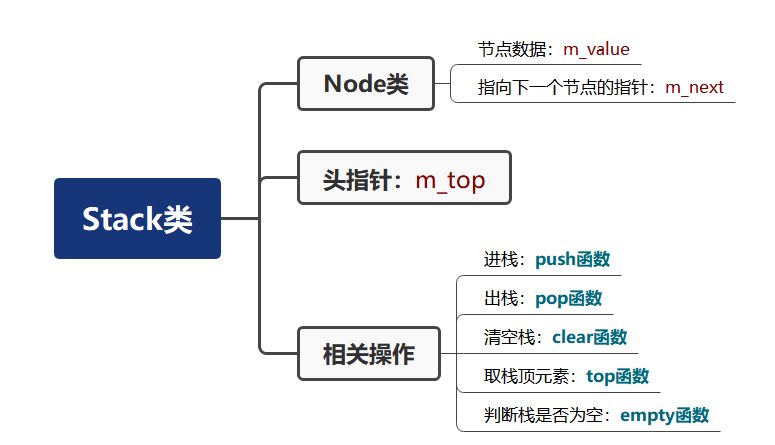
3、运行要求：可以安全高效的实现上述功能。

1. **概要设计**
2. 系统流程设计

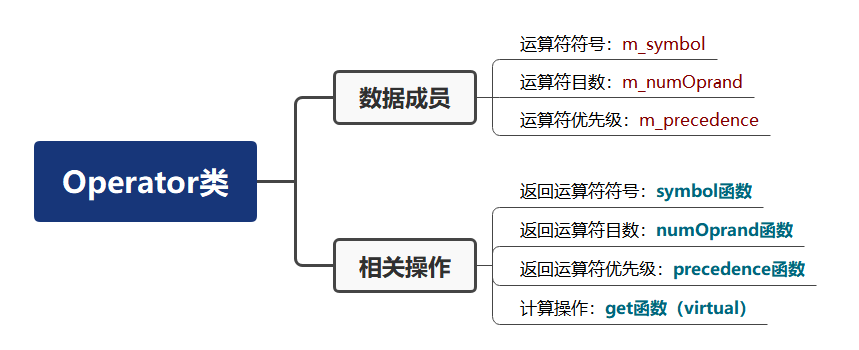


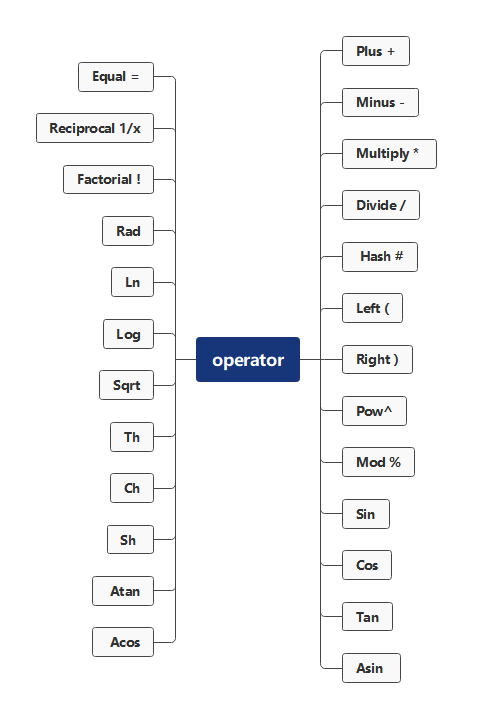


1. 系统模块设计：
2. Stack类：

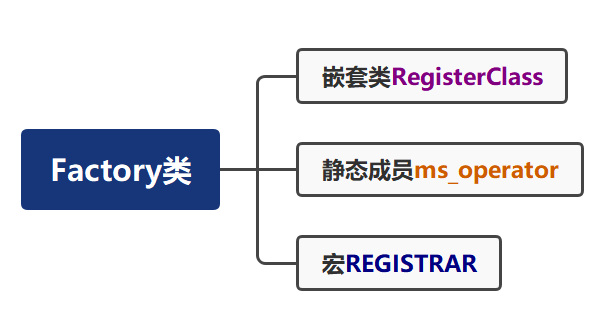


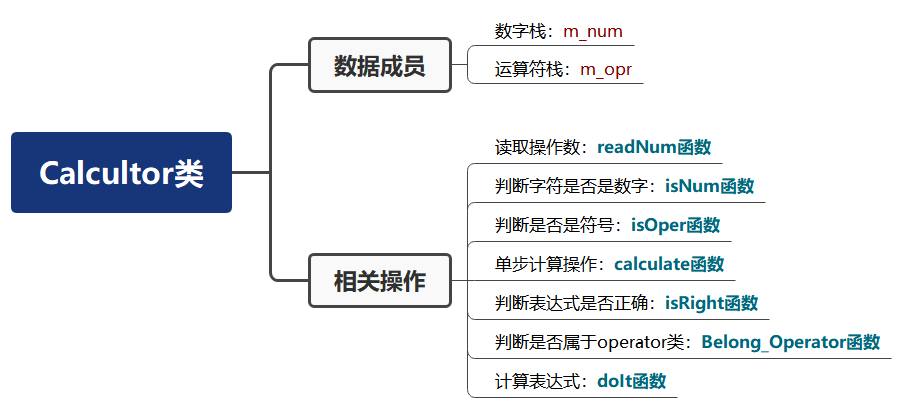
1. Operator类及其派生类:

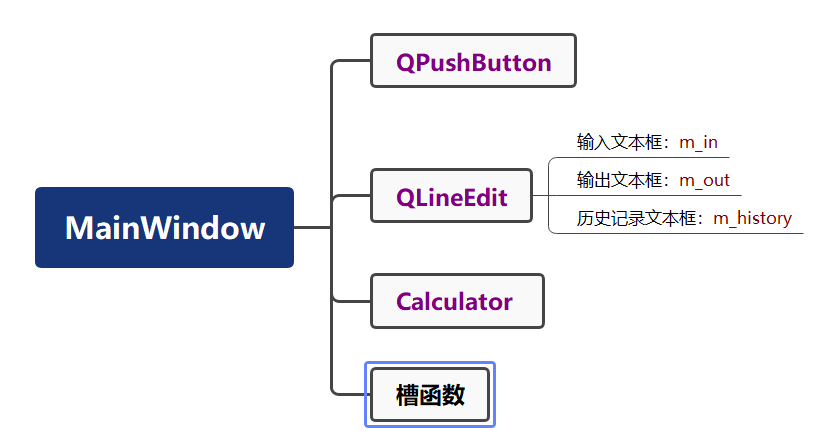




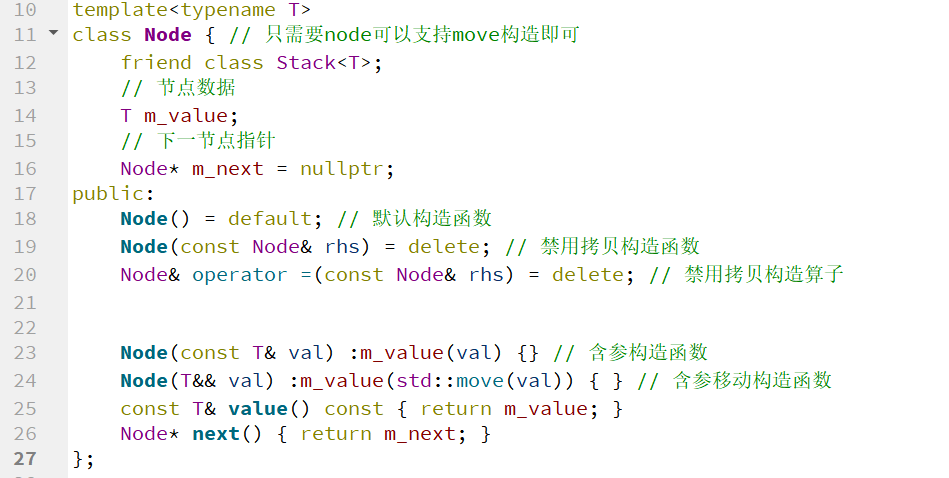
1. Factory类



1. Calculator类
2. MainWindow类



1. **详细设计**
   1. 设计Stack类

(1)设计一个结点类Node，包含节点数据m\_vlaue和指向下一个结点的指针：m\_next。在该类里面禁止使用拷贝构造函数以及拷贝构造算子。

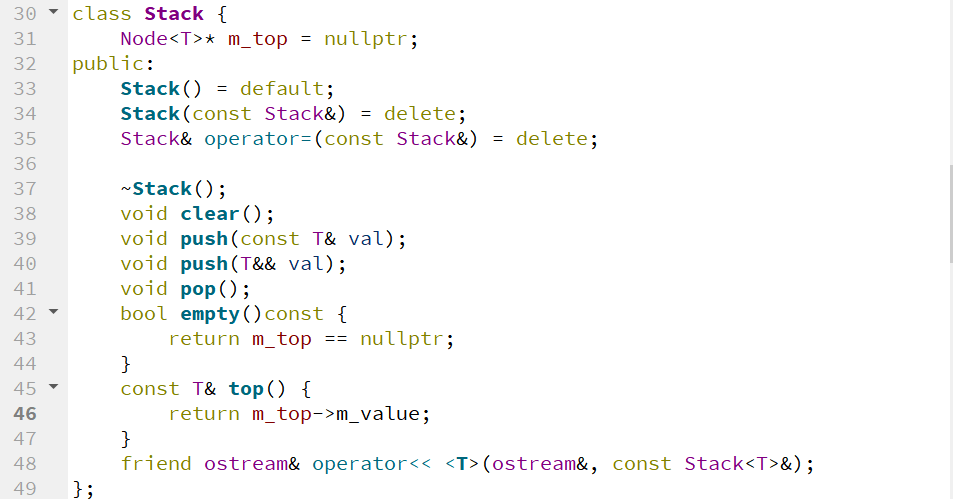
（2）设计Stack类，包含数据成员头指针以及栈相关操作。

push进栈操作：创建一个新节点，将节点压栈，再修改栈顶指针，使其指向新的栈顶结点

pop出栈操作：先将栈顶元素地址保存起来，修改栈顶指针，使其指向新的栈顶元素，最后通过保存的指针释放栈顶元素的内存。

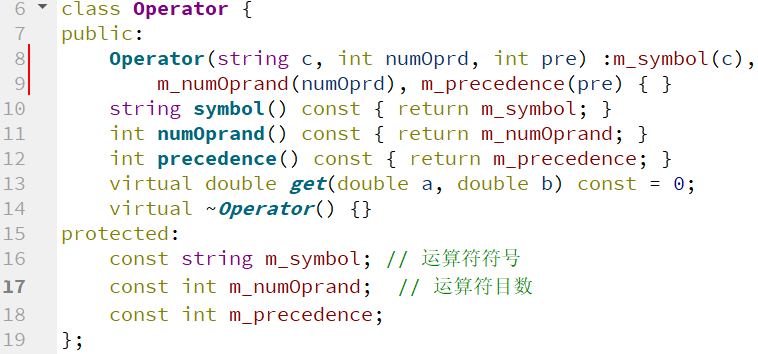
clear清空操作：利用出栈的操作，逐个释放每个元素的内存空间。

top取栈顶元素：返回栈顶元素的const引用。

empty判断是否为空栈：如果栈为空，返回true，否则返回flase。

* 1. 设计Operator类及其派生类

（1）设计Operator类

为了更加方便的表示各个运算符类，我把每一种运算符的共有性质抽象出来，形成公共基类Operator。该公共基类包括三个数据成员m\_symbol、m\_numOprand、m\_precedence,分别表示运算符的符号、目数和优先级。此外，再基类中声明了一个包含两个形参的纯虚函数get,用来实现运算符的计算操作，此函数在基类中无意义，它的实现取决于派生类的具体定义。

（2）设计Operator 类的派生类

派生类要公有继承公共基类Operator,并在其中实现其构造函数，完成纯虚函数get的定义，实现运算符的运算功能。

以Plus类为例：

class Plus : public Operator {

public:

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

double get(double a, double b) const {

return a + b;

}

};

* 1. 设计Factory类

Factory类可以根据运算符的名字自动创建相应运算符类对象。实现一个类注册机制：设计者在完成运算符类的定义后能够以运算符的名字来注册这个类，然后使用者根据运算符的名字来选择运算符类。使用如下数字结构实现：

map<char,function<shared\_ptr<Operator>()>>ms\_operator;

（1）对象工厂Factory通过其嵌套类RegisterClass的构造函数实现对静态成员ms\_operator的类型注册功能。通过定义这样一个成员可以实现string类型到智能指针的一个映射关系，只要调用和string绑定的function对象，便可以自动生成一个指向该运算符的shared智能指针。

(2)为了方便类型的注册，通过宏REGISTRAR创建一个嵌套类RegisterClass的全局对象，同时完成类型T 的注册。

例如：利用以下代码注册Plus类：

REGISTRAR（Plus,“+”）；

（3）##用来连接两个语言符号，产生一个对象名。至此,完成了Plus类的自动注册。然后，利用同样的方法完成其他类的注册。

（4）在Factory类的成员函数create中，it->second()用来调用与形参opr相关联的lamada表达式，进而返回一个已注册类型的对象。如果形参opr是未注册的运算符，则返回一个nullptr。



* 1. 设计Calcultor类

1. 为了计算表达式的值，使用两个栈:一个数据栈和一个运算符栈。

数据栈暂存操作数，运算符栈暂存运算符。

(2)shared\_ptr的使用

由于利用了继承和多态的思想，所以运算符栈保存的使基类Operator指针，用来实现动态绑定。为了使基类指针指向的派生类动态对象能自动释放内存，使用shared\_ptr。其实，使用unique\_ptr更好，但由于本项目仅支持C++11标准，make\_unique函数是由C++14引入的库函数，所以使用shared\_ptr。

(3)isNum函数

如果表达式中含有数字1-9以及小数点“.”，函数返回真，否则为假。

(4)readNum函数

先利用isNum判断是否为数字，然后通过循环将完整数字串读入，存储到一个string类型的对象中，最后通过C++11中新添加的stod函数，将其转变为double类型的数字。

(5)isOper函数

判断是否为“+”、“-”、“\*”、“/”、“%”、“^”等二元运算符。

(6)Belong\_Operator函数

判断运算符属于哪个派生类，方面通过Factory类实现相应运算符的shared\_ptr的生成

(7)calculate函数

通过运算符的优先级判断应当出栈的操作数的数量，将操作数出栈，然后调用绑定函数对象进行表达式的计算，并将计算结果压栈，最后将指向相应的运算符指针出栈。

(8)isRight函数

isRight函数用于判断表达式是否正确，主要判断以下几种情况：

* 表达式开头不能有二元运算符
* 表达式结尾“=”之前不能有二元运算符
* 左括号之后不能有二元运算符
* 右括号之前不能有二元运算符
* 左括号之前不能有数字
* 右括号之后不能有数字
* 左右括号数量不能不等
* 右括号不能在左括号之前
* 二元运算符之后不能为二元运算符
* 除数不能为0
* 数字不能出现在sin,cos等函数之前

除此之外，若变量超出函数的定义域，会输出“nan”,即“not a number”,或者输出“error”。

如：asin,acos的定义域为[-1，1]，

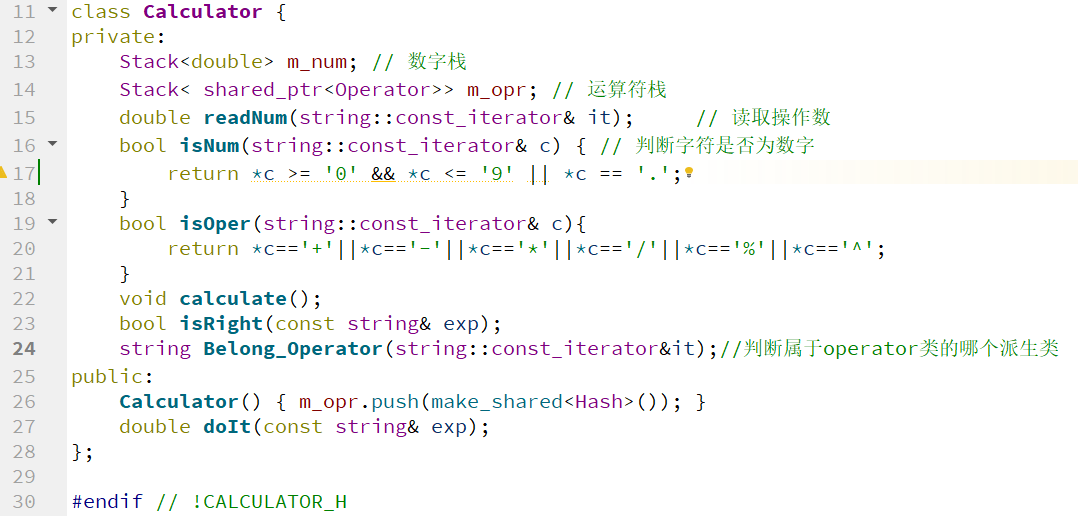
Log,ln的定义域为[0，+]。

tan的定义域为{x|x}

(9)doIt函数

1. 先调用isRight函数判断表达式是否是正确的
2. 利用迭代器扫描整个表达式，遇到操作数（如果是e,通过e=1+1/1!+...+1/n!计算e的值；如果是pi,通过(atan1)\*4计算它的值）时，将其压入操作数栈;遇到运算符时，与当前运算符栈的栈顶运算符比较优先级，若高于栈顶运算符的优先级或运算符栈为空，则将其压入运算符栈;否则将栈顶运算符弹出，并根据所弹出运算符的目数，弹出运算数栈中相应数目的操作数，然后做运算并将运算的结果压入操作数栈。重复这一过程，直到当前运算符入栈。
3. 继续扫描表达式，并执行相应的操作，直至表达式结束。
4. 在此过程中，如若遇到左括号，则将其压入符号栈中，遇到右括号，调用calculate重复计算过程，直至遇到左括号，再将左括号出栈。
5. 最后，如果两个栈非空，则逐个弹出运算符栈顶元素和操作数栈中相应数目的运算数，并执行相应运算，将结果压入操作数栈，重复此过程直到运算符栈为空，最后一次运算的结果便是表达式的值。

注：为了方便，在从左往右扫描表达式时，都会在运算符栈初始化一个优先级很低的符号#。



* 1. 设计界面

mainWindow类中包含QpushButton\*、QlineEdit、Calculator类的对象以及槽函数。通过在界面QWidget上添加 QPushButton按键，QLineEdit显示文本框并定义和连接相关的槽函数即可达到计算器计算的的功能。

1. 设置Qwidget界面。
2. 设置运算符和函数的QPushButton按钮。
3. 设置QlineEdit文本框m\_in、m\_out、m\_hist。
4. 安排按钮和文本框的大小及布局。
5. 通过信号与槽机制调用函数，实现界面和计算机类内部的交互。
6. 设置部分按键的快捷键，可实现通过键盘直接输入。
7. 设置背景图片。

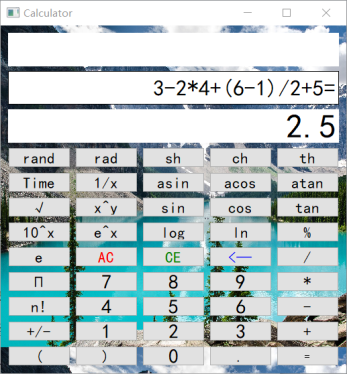
除了基本的运算功能外，还在界面添加以下功能：

* Time：显示当前时间。
* rand：产生随机数。
* AC：清除输入文本框的内容。
* CE：删除末尾数字。
* Delete：逐个删除输入文本框的内容。
  1. 异常检测部分

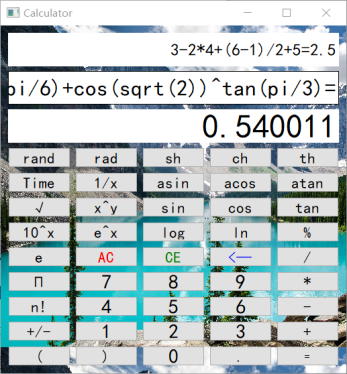
通过异常处理机制try,throw,catch来进行异常检测：

* 表达式等号“=”输入后，try检测异常。
* 如果表达式不正确，throw将抛出“error”。
* “error”将会由catch捕获，将其显示在文本框上面。

1. **测试**
2. 输入表达式“3-2\*4+（6-1）/2+5=”



1. 输入表达式“sin(pi/6)+cos(sqrt(2))^tan(pi/3)=”



3、验证各个运算符功能：

(1)+ （2）- （3）\*

|  |  |  |
| --- | --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\S1@CMB@BCLTE(W}]UUOSYDD.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\S98%H]WD6N_R7K2@X)1X3H7.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\PAOTXM{5UR4EBY{BF2DE$VA.png |

（4）/ （5）% （6）^(e^x,10^x皆属于此功能)

|  |  |  |
| --- | --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\4%93)5{O9}R~VF~TX1%IZHY.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\0W}YICV4]]AFGGYLC2KTT1U.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\QX))_~~%QJI(J9~35LU[3YM.png |

(7)sin （8）cos （8）tan

|  |  |  |
| --- | --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\IXSOO{X~[F~YAZP48`$TBTO.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\RAD_RK74QT8H}Y@QBFJ$VMU.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\LW0NC9(]FKJAH1BG[ES2AEV.png |

（10）sqrt (11)asin (12)acos

|  |  |  |
| --- | --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\GMNKPPIWWFP}(8_8CP1@WAI.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\~95YZ%}NFU6$6)QA4NS4ZSS.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\4G1XDZTD2MB~ZZMUEC{J9YJ.png |

（13）atan (14)sh (15)ch

|  |  |  |
| --- | --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\DWS{64`5PP$JYP%_BT)KMU7.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\ROB)PJWMUV~%LIO(_BCBRN5.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\YG`%E1Z@P327HS~$ZDPBN)P.png |

（16）th (17)log (18)ln

|  |  |  |
| --- | --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\W_$QG8IJYSLPE]]3R_OSA{W.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\Y11KT`R47L_0X]K%[CQZW`6.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\(T~E%F8~U]1AB@L1H$IQ}7R.png |

（19）n! （20）1/x （21）rad

|  |  |  |
| --- | --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\2M{KH68Y~GI6USR14DU4}54.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\_U3KHG65JX%D`~$F@9BA$PC.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\`T(EDW4H[L)U%B$_EQ3D4_O.png |

（22）rand （23）Time

|  |  |  |
| --- | --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\RPKHJ`%77`]2LHC3F}X7K)Q.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\7KG788UEKG_4ZREI@V843$T.png |  |

4、验证判断表达式是否错误的功能

(1)表达式开头不能有二元运算符 (2) 表达式结尾“=”之前不能有二元运算符

|  |  |
| --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\U`OD{_PDSLORBP`7~717[5L.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\D)Z2[V(I1{F]]09~93XFKPX.png |

（3）左括号之后不能有二元运算符 （4）右括号之前不能有二元运算符

|  |  |
| --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\RGS2QHTLK)9DRA]258Z950V.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\YL`T[JU4P8]{C8FTR0]8$7E.png |

（5）左括号之前不能有数字 （6）右括号之后不能有数字

|  |  |
| --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\Q{TQ42UT7$21LVW_9H91WUT.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\@U[E8@TK5B{{ZKRSX8CB]B6.png |

（7）左右括号数量不能不等 （8）右括号不能在左括号之前

|  |  |
| --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\DXJ{~FYL(AIH{I5U(XO4WMD.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\W$}]PG}5YX437$JHG%BT)3O.png |

（9）二元运算符之后不能为二元运算符 （10）除数不能为0

|  |  |
| --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\BOC6){4$AJ]6$TOHBV6$RBW.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\5F)QD(LA%4Y$5$B2E}D71P7.png |

（11）数字不能出现在sin,cos等函数之前

|  |  |
| --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\{GS0U[JHLMX4[RGP$%L%{60.png |  |

（12）不能超出定义域

|  |  |  |
| --- | --- | --- |
| d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\AV_ILH_9~OXVS_VT(%ODBY6.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\SK~I8O[()O}`DA3C){XO`1T.png | d:\Documents\Tencent Files\1377005145\FileRecv\MobileFile\Image\A@P%~TVI2L]YSPSOOA5{[5M.png |

**六、结论**

对系统开发的总结：在这次系统开发的过程中，我受到了很多启发:

1、开发系统需要明确的思路，对于系统的框架要首先搭建好。计算器的内部结构方面：先写出节点类Node,再由Node得到栈类Stack;抽象出运算符的共性作为Operator类，具体的运算符如“+”，“-”，“\*”，“/”皆作为其派生类；用Factory类注册各个运算符；最后通过运算符号栈和操作数栈搭出计算器类的基本框架，完善相应的函数。计算器的界面：利用Qt在界面QWidget上添加 QPushButton按键，QLineEdit显示文本框并定义和连接相关的槽函数达到计算器计算的功能，其中信号与槽机制，使我更加理解了界面与内部结构之间的交流传输。

2、在程序设计时，我们要尽可能充分考虑各种情况，没有考虑到的情况，也要及时修补。如：完善判断表达式是否正确的函数时，表达式错误的情况有多种，但在初期我仅能考虑到几种情况，后续经常出现程序运行中断，需要进行修复。

3、在程序设计时，如果某一部分程序比较长，使用率也比较高，可以将它写做一个单独的函数，需要时调用即可，这样可以提高代码安全性和效率。

4、在使用Qt时,由于之前并没有接触过，会遇到很多问题，解决这些问题需要自己找资料，一步一步试验。

存在的不足：

1、计算机类使用shared\_ptr来构建符号栈，其实使用unique\_ptr更好，但由于本项目仅支持C++11标准，make\_unique函数是由C++14引入的库函数，所以只能使用shared\_ptr。shared\_ptr可以进行赋值或者复制，可能会造成一定的安全问题。

2、计算器的界面是使用Qt的函数进行大小和布局安排，代码比较冗长，如果直接拖拽界面的话，可能会更简练。

**七、附录**

1、stack.h

#ifndef STACK\_H

#define STACK\_H

#include<iostream>

using namespace std;

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

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

// 节点类

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);

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)); // val is a lvalue, so I should move it to a rvalue

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

2、operator.h

#ifndef OPERATOR\_H

#define OPERATOR\_H

#include<string>

#include<math.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 {

if(b==0){throw string("error");}

return a / b;

}

};

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

public:

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

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

return a; // 本身#不需要进行任何运算, 但是仍然需要返回值

}

};

class **Left** :public Operator { // 符号(

public:

**Left**() :Operator("(", 0, 6) {}

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

return a;

}

};

class **Right** :public Operator { // 运算符)

public:

**Right**() :Operator(")", 0, 0) {}

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

return a; // 本身#不需要进行任何运算, 但是仍然需要返回值

}

};

class **Pow** :public Operator { // 运算符^

public:

**Pow**() :Operator("^", 2, 4) {}

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

return pow(a,b);

}

};

class **Mod** :public Operator { // 运算符%

public:

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

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

return int(a)%int(b);

}

};

class **Sin** :public Operator { // 运算符sin

public:

**Sin**() :Operator("sin", 1, 5) {}

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

return sin(b);

}

};

class **Cos** :public Operator { // 运算符cos

public:

**Cos**() :Operator("cos", 1, 5) {}

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

return cos(b);

}

};

class **Tan** :public Operator { // 运算符tan

public:

**Tan**() :Operator("tan", 1, 5) {}

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

if(b==(std::atan(1.0) \* 4)/2)throw string("error");

return tan(b);

}

};

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

public:

**Asin**() :Operator("asin", 1, 5) {}

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

return (asin(b)\*180/(atan(1)\*4));

}

};

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

public:

**Acos**() :Operator("acos", 1, 5) {}

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

return (acos(b)\*180/(atan(1)\*4));

}

};

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

public:

**Atan**() :Operator("atan", 1, 5) {}

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

return (atan(b)\*180/(atan(1)\*4));

}

};

class **Sh** :public Operator { // 运算符sh

public:

**Sh**() :Operator("sh", 1, 5) {}

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

return sinh(b);

}

};

class **Ch** :public Operator { // 运算符ch

public:

**Ch**() :Operator("ch", 1, 5) {}

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

return cosh(b);

}

};

class **Th** :public Operator { // 运算符th

public:

**Th**() :Operator("th", 1, 5) {}

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

return tanh(b);

}

};

class **Rad** :public Operator { // 运算符rad

public:

**Rad**() :Operator("rad", 1, 5) {}

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

return (b\*180/(atan(1)\*4));

}

};

/\*class Rand :public Operator { // 运算符rand

public:

Rand() :Operator("rand", 1, 5) {}

double get(double a, double b) const {

return tanh(b);

}

};\*/

class **Sqrt** :public Operator { // 运算符sqrt

public:

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

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

return sqrt(b);

}

};

class **Log** :public Operator { // 运算符log

public:

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

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

return log10(b);

}

};

class **Ln** :public Operator { // 运算符ln

public:

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

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

return log(b);

}

};

class **Reciprocal** :public Operator { //求倒数

public:

**Reciprocal**() :Operator("rec", 1, 5) {}

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

return 1/b; //

}

};

class **Factorial** :public Operator { // 运算符!

public:

**Factorial**() :Operator("!", 1, 5) {}

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

int i=0,sum=1;

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

sum\*=i;

}

return sum;

}

};

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

public:

**Equal**() :Operator("=", 2, 0) {}

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

return a;

}

};

#endif // OPERATOR\_H

3、factory.h

#ifndef OPERATORFACTORY\_H

#define OPERATORFACTORY\_H

#include<string>

#include<map>

#include<functional>

#include<memory>

#include"operator.h"

// 注册Object的宏声明

#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\_shared<T>(); });

}

};

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

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

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

return it->second();

}

static map<string, function<shared\_ptr<Operator>()>> ms\_operator;

};

#endif // OPERATORFACTORY\_H

4、calculator.h

#pragma once

#ifndef CALCULATOR\_H

#define CALCULATOR\_H

#include <vector>

#include <memory>

#include "operator.h"

#include "stack.h"

#include"factory.h"

#include<QObject>

class **Calculator** {

private:

Stack<double> m\_num; // 数字栈

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

double **readNum**(string::const\_iterator& it); // 读取操作数

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

return \*c >= '0' && \*c <= '9' || \*c == '.';

}

bool **isOper**(string::const\_iterator& c){

return \*c=='+'||\*c=='-'||\*c=='\*'||\*c=='/'||\*c=='%'||\*c=='^';

}

void **calculate**();

bool **isRight**(const string& exp);

string **Belong\_Operator**(string::const\_iterator&it);//判断属于operator类的哪个派生类

public:

**Calculator**() { m\_opr.push(make\_shared<Hash>()); }

double **doIt**(const string& exp);

};

#endif // !CALCULATOR\_H

5、mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include<string>

#include<calculator.h>

#include"QPushButton"

#include"QLineEdit"

#include<QString>

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

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

~***MainWindow***();

public slots:

void **button\_0**();

void **button\_1**();

void **button\_2**();

void **button\_3**();

void **button\_4**();

void **button\_5**();

void **button\_6**();

void **button\_7**();

void **button\_8**();

void **button\_9**();

void **button\_pi**();//Π

void **button\_e**();

void **button\_dot**();//.

void **button\_plus**();

void **button\_minus**();

void **button\_multiply**();

void **button\_divide**();

void **button\_pow**();

void **button\_mod**();//%

void **button\_sin**();

void **button\_cos**();

void **button\_tan**();

void **button\_sqrt**();

void **button\_log**();

void **button\_ln**();

void **button\_reciprocal**();//倒数

void **button\_factorial**();//阶乘

void **button\_time**();//时间

void **button\_equal**();

void **button\_left**();

void **button\_right**();

void **button\_clear**();

void **button\_clearnum**();

void **button\_delete**();

void **button\_arcsin**();

void **button\_arccos**();

void **button\_arctan**();

void **button\_exp**();

void **button\_ten**();

void **button\_negative**();

void **button\_sh**();

void **button\_ch**();

void **button\_th**();

void **button\_rad**();

void **button\_rand**();

void **setBackgroundImage**();

private:

QPushButton\* m\_0;

QPushButton\* m\_1;

QPushButton\* m\_2;

QPushButton\* m\_3;

QPushButton\* m\_4;

QPushButton\* m\_5;

QPushButton\* m\_6;

QPushButton\* m\_7;

QPushButton\* m\_8;

QPushButton\* m\_9;

QPushButton\* m\_pi;

QPushButton\* m\_e;

QPushButton\* m\_dot;

QPushButton\* m\_plus;

QPushButton\* m\_minus;

QPushButton\* m\_multiply;

QPushButton\* m\_divide;

QPushButton\* m\_pow;

QPushButton\* m\_mod;

QPushButton\* m\_sin;

QPushButton\* m\_cos;

QPushButton\* m\_tan;

QPushButton\* m\_sqrt;

QPushButton\* m\_log;

QPushButton\* m\_ln;

QPushButton\* m\_reciprocal;

QPushButton\* m\_factorial;

QPushButton\* m\_time;

QPushButton\* m\_equal;

QPushButton\* m\_left;

QPushButton\* m\_right;

QPushButton\* m\_clear;

QPushButton\* m\_clearnum;

QPushButton\* m\_delete;

QPushButton\* m\_arcsin;

QPushButton\* m\_arccos;

QPushButton\* m\_arctan;

QPushButton\* m\_exp;

QPushButton\* m\_ten;

QPushButton\* m\_negative;

QPushButton\* m\_sh;

QPushButton\* m\_ch;

QPushButton\* m\_th;

QPushButton\* m\_rad;

QPushButton\* m\_rand;

QLineEdit \*m\_in;

QLineEdit \*m\_out;

QLineEdit \*m\_history;

Calculator m\_cal;

};

#endif // MAINWINDOW\_H

6、factory.cpp

#include"factory.h"

map<string, function<shared\_ptr<Operator>()>> Factory::ms\_operator;

7、calculator.cpp

#include "calculator.h"

#include"operator.h"

#include "factory.h"

#include<string>

using namespace std;

// 读取操作数

double Calculator::**readNum**(string::const\_iterator& it) {

string t;

while (isNum(*it*))

t += \*it++;

return stod(t);

}

// 动作, 使用运算符栈和操作数栈顶的元素进行计算

void Calculator::**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();

}

double Calculator::**doIt**(const string& exp) {

if(!isRight(exp)) throw string("error");

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 if(\*it=='e'){

double e=1.0,n=1.0;

int k=1;

while(1/n>1e-10){//通过e=1+1/1!+...+1/n!计算e的值

e+=1/n;

++k;

n=k\*n;

}

++it;

m\_num.push(e);

}

else if (\*it == 'p') {

auto temp = it + 1;

if (\*temp == 'i') {

m\_num.push(std::atan(1.0) \* 4);

it += 2;

}

}

else if (\*it == ')') {

while (m\_opr.top()->symbol()!="(") {

calculate();

}

m\_opr.pop();

++it;

}

else {// 如果是运算符, 压栈或计算

auto oo=Factory::create(Belong\_Operator(*it*));

// 如果栈顶优先级高, 则计算

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

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

break;

calculate();

}

// "="从不入栈

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

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

}

}

double result = m\_num.top();

m\_num.pop();

return result;

}

bool Calculator::**isRight**(const string& exp){

bool b=1;

auto it = exp.begin();

if(isOper(*it*)) b=0;//计算式开头不能为符号，括号除外

auto it1=exp.end();

auto temp1=it1-2;

if(isOper(*temp1*))b=0;//=前不能有符号，括号除外

int count=0;//记录没有与之匹配的左括号或右括号

for (; it != exp.end();++it) {

auto temp=it+1;

auto temp2=it-1;

if(isOper(*it*)){

if(isOper(*temp*))b=0;//符号后不能有符号，括号除外

}

if(\*it=='(') {

if(isOper(*temp*))b=0;

if(isNum(*temp2*))b=0;

++count;//左括号后不能有符号

}

else if(\*it==')'){

if(isOper(*temp2*))b=0;

if(isNum(*temp*))b=0;

--count;//右括号前不能有符号

}

else if(\*it=='/'){

if(\*temp=='0')b=0;//除数不能为0

}

else if(\*it=='t'){

if(\*temp=='a'){

auto temp3=it+4;

string str;

for(int i=0;i<4;++i){

str+=\*temp3++;

}

if(str=="pi/2") b=0;

}

}//tanx中x!=pi/2

else if(isNum(*it*)){

if(\*temp>='a'&&\*temp<='z') b=0;

auto temp4=it+2;

if(\*temp4=='e'||\*temp4=='a')b=1;

}

if(count<0)b=0;//右括号不能出现在左括号之前

}

if(count!=0){

b=0;//左右括号的数量应当相等

}

return b;

}

string Calculator::**Belong\_Operator**(string::const\_iterator&it){

string s;

if (\*it == '+') {

s+=\*it++;

}

else if (\*it == '-') {

s+=\*it++;

}

else if (\*it == '\*') {

s+=\*it++;

}

else if (\*it == '/') {

s+=\*it++;

}

else if (\*it == '=') {

s+=\*it++;

}

else if (\*it == '%') {

s+=\*it++;

}

else if (\*it == '^') {

s+=\*it++;

}

else if(\*it== '(') {

s+=\*it++;

}

else if(\*it=='!'){

s+=\*it++;

}

else if (\*it == 's') {

auto temp = it + 1;

if (\*temp == 'i' ) {

for(int i=0;i<3;++i){

s+=\*it++;

}

}

else if (\*temp == 'q') {

for(int i=0;i<4;++i){

s+=\*it++;

}

}

else if(\*temp=='h'){

for(int i=0;i<2;++i){

s+=\*it++;

}

}

}

else if(\*it=='c'){

auto temp=it+1;

if(\*temp=='o'){

for(int i=0;i<3;++i){

s+=\*it++;

}

}

else if(\*temp=='h'){

for(int i=0;i<2;++i){

s+=\*it++;

}

}

}

else if(\*it=='t'){

auto temp=it+1;

if(\*temp=='a'){

for(int i=0;i<3;++i){

s+=\*it++;

}

}

else if(\*temp=='h'){

for(int i=0;i<2;++i){

s+=\*it++;

}

}

}

else if(\*it=='l'){

auto temp=it+1;

if(\*temp=='o'){

for(int i=0;i<3;++i){

s+=\*it++;

}

}

else if(\*temp=='n'){

for(int i=0;i<2;++i){

s+=\*it++;

}

}

}

else if(\*it=='r'){

auto temp=it+1;

if(\*temp=='e'){

for(int i=0;i<3;++i){

s+=\*it++;

}

}

else if(\*temp=='a'){

for(int i=0;i<3;++i){

s+=\*it++;

}

}

}

else if(\*it=='a'){

for (int i = 0; i < 4; ++i) {

s+=\*it++;

}

}

return s;

}

8、mainwindow.cpp

#include "mainwindow.h"

#include <QLayout>

#include <QVector>

#include <QString>

#include <QToolTip>

#include <QStackedLayout>

#include <QDateTime>

string text;

string history;

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(parent)

{

Factory::RegisterClass<Hash> reg\_Hash("#");

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

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

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

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

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

REGISTRAR(**Pow**,"^")

REGISTRAR(**Mod**,"%")

REGISTRAR(**Sin**,"sin")

REGISTRAR(**Tan**,"tan")

REGISTRAR(**Cos**,"cos")

REGISTRAR(**Sqrt**,"sqrt")

REGISTRAR(**Log**,"log")

REGISTRAR(**Ln**,"ln")

REGISTRAR(**Reciprocal**,"rec")

REGISTRAR(**Factorial**,"!")

REGISTRAR(**Left**,"(")

REGISTRAR(**Right**,")")

REGISTRAR(**Asin**,"asin")

REGISTRAR(**Atan**,"atan")

REGISTRAR(**Acos**,"acos")

REGISTRAR(**Sh**,"sh")

REGISTRAR(**Ch**,"ch")

REGISTRAR(**Th**,"th")

REGISTRAR(**Rad**,"rad")

m\_0=new QPushButton("0",this);

m\_1=new QPushButton("1",this);

m\_2=new QPushButton("2",this);

m\_3=new QPushButton("3",this);

m\_4=new QPushButton("4",this);

m\_5=new QPushButton("5",this);

m\_6=new QPushButton("6",this);

m\_7=new QPushButton("7",this);

m\_8=new QPushButton("8",this);

m\_9=new QPushButton("9",this);

m\_pi=new QPushButton("Π",this);

m\_e=new QPushButton("e",this);

m\_dot=new QPushButton(".",this);

m\_plus=new QPushButton("+",this);

m\_minus=new QPushButton("-",this);

m\_multiply=new QPushButton("\*",this);

m\_divide=new QPushButton("/",this);

m\_pow=new QPushButton("x^y",this);

m\_mod=new QPushButton("%",this);

m\_sin=new QPushButton("sin",this);

m\_cos=new QPushButton("cos",this);

m\_tan=new QPushButton("tan",this);

m\_sqrt=new QPushButton("√",this);

m\_log=new QPushButton("log",this);

m\_ln=new QPushButton("ln",this);

m\_reciprocal=new QPushButton("1/x",this);

m\_factorial=new QPushButton("n!",this);

m\_time=new QPushButton("Time",this);

m\_equal=new QPushButton("=",this);

m\_left=new QPushButton("(",this);

m\_right=new QPushButton(")",this);

m\_clear=new QPushButton("AC",this);

m\_clearnum=new QPushButton("CE",this);

m\_delete=new QPushButton("<—",this);

m\_arcsin=new QPushButton("asin",this);

m\_arccos=new QPushButton("acos",this);

m\_arctan=new QPushButton("atan",this);

m\_exp=new QPushButton("e^x",this);

m\_ten=new QPushButton("10^x",this);

m\_negative=new QPushButton("+/-",this);

m\_sh=new QPushButton("sh",this);

m\_ch=new QPushButton("ch",this);

m\_th=new QPushButton("th",this);

m\_rad=new QPushButton("rad",this);

m\_rand=new QPushButton("rand",this);

m\_in=new QLineEdit(this);

m\_out=new QLineEdit(this);

m\_history=new QLineEdit(this);

m\_in->setReadOnly(true);

m\_in->setAlignment(Qt::AlignRight);

m\_in->setFrame(false);

m\_out->setReadOnly(true);

m\_out->setAlignment(Qt::AlignRight);

m\_out->setFrame(false);

m\_history->setReadOnly(true);

m\_history->setAlignment(Qt::AlignRight);

m\_history->setFrame(false);

m\_clear->setStyleSheet("color:red");

m\_clearnum->setStyleSheet("color:green");

m\_delete->setStyleSheet("color:blue");

m\_0->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_1->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_2->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_3->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_4->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_5->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_6->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_7->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_8->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_9->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_pi->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_e->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_plus->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_minus->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_multiply->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_divide->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_pow->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_mod->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_sin->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_cos->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_tan->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_sqrt->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_log->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_ln->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_reciprocal->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_factorial->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_time->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_equal->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_left->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_right->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_dot->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_clear->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_clearnum->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_delete->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_arcsin->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_arccos->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_arctan->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_exp->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_ten->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_negative->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_sh->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_ch->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_th->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_rad->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

m\_rand->setSizePolicy(QSizePolicy::Minimum, QSizePolicy::Minimum);

//设置快捷键，可以从键盘直接输入

m\_0->setShortcut(QKeySequence(Qt::Key\_0));

m\_1->setShortcut(QKeySequence(Qt::Key\_1));

m\_2->setShortcut(QKeySequence(Qt::Key\_2));

m\_3->setShortcut(QKeySequence(Qt::Key\_3));

m\_4->setShortcut(QKeySequence(Qt::Key\_4));

m\_5->setShortcut(QKeySequence(Qt::Key\_5));

m\_6->setShortcut(QKeySequence(Qt::Key\_6));

m\_7->setShortcut(QKeySequence(Qt::Key\_7));

m\_8->setShortcut(QKeySequence(Qt::Key\_8));

m\_9->setShortcut(QKeySequence(Qt::Key\_9));

m\_e->setShortcut(QKeySequence(Qt::Key\_E));

m\_clear->setShortcut(QKeySequence(Qt::Key\_C));

m\_dot->setShortcut(QKeySequence(Qt::Key\_Period));

m\_plus->setShortcut(QKeySequence(Qt::Key\_Plus));

m\_minus->setShortcut(QKeySequence(Qt::Key\_Minus));

m\_multiply->setShortcut(QKeySequence(Qt::Key\_Asterisk));

m\_divide->setShortcut(QKeySequence(Qt::Key\_Slash));

m\_equal->setShortcut(QKeySequence(Qt::Key\_Equal));

m\_left->setShortcut(QKeySequence(Qt::Key\_ParenLeft));

m\_right->setShortcut(QKeySequence(Qt::Key\_ParenRight));

m\_delete->setShortcut(QKeySequence(Qt::Key\_Backspace));

m\_sin->setShortcut(QKeySequence(Qt::Key\_S));

m\_tan->setShortcut(QKeySequence(Qt::Key\_T));

//设置按钮的大小

m\_in->setFixedHeight(50);

m\_out->setFixedHeight(50);

m\_history->setFixedHeight(50);

m\_0->setFixedHeight(30);

m\_1->setFixedHeight(30);

m\_2->setFixedHeight(30);

m\_3->setFixedHeight(30);

m\_4->setFixedHeight(30);

m\_5->setFixedHeight(30);

m\_6->setFixedHeight(30);

m\_7->setFixedHeight(30);

m\_8->setFixedHeight(30);

m\_9->setFixedHeight(30);

m\_pi->setFixedHeight(30);

m\_e->setFixedHeight(30);

m\_plus->setFixedHeight(30);

m\_minus->setFixedHeight(30);

m\_multiply->setFixedHeight(30);

m\_divide->setFixedHeight(30);

m\_pow->setFixedHeight(30);

m\_mod->setFixedHeight(30);

m\_sin->setFixedHeight(30);

m\_cos->setFixedHeight(30);

m\_tan->setFixedHeight(30);

m\_sqrt->setFixedHeight(30);

m\_log->setFixedHeight(30);

m\_ln->setFixedHeight(30);

m\_reciprocal->setFixedHeight(30);

m\_factorial->setFixedHeight(30);

m\_time->setFixedHeight(30);

m\_equal->setFixedHeight(30);

m\_left->setFixedHeight(30);

m\_right->setFixedHeight(30);

m\_dot->setFixedHeight(30);

m\_clear->setFixedHeight(30);

m\_clearnum->setFixedHeight(30);

m\_delete->setFixedHeight(30);

m\_arcsin->setFixedHeight(30);

m\_arccos->setFixedHeight(30);

m\_arctan->setFixedHeight(30);

m\_exp->setFixedHeight(30);

m\_ten->setFixedHeight(30);

m\_negative->setFixedHeight(30);

m\_sh->setFixedHeight(30);

m\_ch->setFixedHeight(30);

m\_th->setFixedHeight(30);

m\_rad->setFixedHeight(30);

m\_rand->setFixedHeight(30);

QFont num\_font;

num\_font.setFamily("黑体");

num\_font.setPointSize(18);

num\_font.setWeight(50);

m\_0->setFont(num\_font);

m\_1->setFont(num\_font);

m\_2->setFont(num\_font);

m\_3->setFont(num\_font);

m\_4->setFont(num\_font);

m\_5->setFont(num\_font);

m\_6->setFont(num\_font);

m\_7->setFont(num\_font);

m\_8->setFont(num\_font);

m\_9->setFont(num\_font);

QFont opr\_font;

opr\_font.setFamily("黑体");

opr\_font.setPointSize(14);

opr\_font.setWeight(20);

m\_pi->setFont(opr\_font);

m\_e->setFont(opr\_font);

m\_plus->setFont(opr\_font);

m\_minus->setFont(opr\_font);

m\_multiply->setFont(opr\_font);

m\_divide->setFont(opr\_font);

m\_clear->setFont(opr\_font);

m\_log->setFont(opr\_font);

m\_ln->setFont(opr\_font);

m\_mod->setFont(opr\_font);

m\_sin->setFont(opr\_font);

m\_cos->setFont(opr\_font);

m\_tan->setFont(opr\_font);

m\_sqrt->setFont(opr\_font);

m\_delete->setFont(opr\_font);

m\_reciprocal->setFont(opr\_font);

m\_factorial->setFont(opr\_font);

m\_clearnum->setFont(opr\_font);

m\_time->setFont(opr\_font);

m\_left->setFont(opr\_font);

m\_right->setFont(opr\_font);

m\_dot->setFont(opr\_font);

m\_pow->setFont(opr\_font);

m\_arcsin->setFont(opr\_font);

m\_arccos->setFont(opr\_font);

m\_arctan->setFont(opr\_font);

m\_exp->setFont(opr\_font);

m\_ten->setFont(opr\_font);

m\_sh->setFont(opr\_font);

m\_ch->setFont(opr\_font);

m\_th->setFont(opr\_font);

m\_rad->setFont(opr\_font);

m\_rand->setFont(opr\_font);

m\_negative->setFont(opr\_font);

QFont line1\_font;

line1\_font.setFamily("黑体");

line1\_font.setPointSize(20);

line1\_font.setWeight(40);

m\_in->setFont(line1\_font);

QFont line2\_font;

line2\_font.setFamily("黑体");

line2\_font.setPointSize(30);

line2\_font.setWeight(50);

m\_out->setFont(line2\_font);

QFont line3\_font;

line3\_font.setFamily("黑体");

line3\_font.setPointSize(15);

line3\_font.setWeight(25);

m\_history->setFont(line3\_font);

//水平布局，安排各个按钮的位置

QHBoxLayout \*bottom\_1=new QHBoxLayout;

bottom\_1->addWidget(m\_time);

bottom\_1->addWidget(m\_reciprocal);

bottom\_1->addWidget(m\_arcsin);

bottom\_1->addWidget(m\_arccos);

bottom\_1->addWidget(m\_arctan);

QHBoxLayout \*bottom\_2=new QHBoxLayout;

bottom\_2->addWidget(m\_sqrt);

bottom\_2->addWidget(m\_pow);

bottom\_2->addWidget(m\_sin);

bottom\_2->addWidget(m\_cos);

bottom\_2->addWidget(m\_tan);

QHBoxLayout \*bottom\_3=new QHBoxLayout;

bottom\_3->addWidget(m\_ten);

bottom\_3->addWidget(m\_exp);

bottom\_3->addWidget(m\_log);

bottom\_3->addWidget(m\_ln);

bottom\_3->addWidget(m\_mod);

QHBoxLayout \*bottom\_4=new QHBoxLayout;

bottom\_4->addWidget(m\_e);

bottom\_4->addWidget(m\_clear);

bottom\_4->addWidget(m\_clearnum);

bottom\_4->addWidget(m\_delete);

bottom\_4->addWidget(m\_divide);

QHBoxLayout \*bottom\_5=new QHBoxLayout;

bottom\_5->addWidget(m\_pi);

bottom\_5->addWidget(m\_7);

bottom\_5->addWidget(m\_8);

bottom\_5->addWidget(m\_9);

bottom\_5->addWidget(m\_multiply);

QHBoxLayout \*bottom\_6=new QHBoxLayout;

bottom\_6->addWidget(m\_factorial);

bottom\_6->addWidget(m\_4);

bottom\_6->addWidget(m\_5);

bottom\_6->addWidget(m\_6);

bottom\_6->addWidget(m\_minus);

QHBoxLayout \*bottom\_7=new QHBoxLayout;

bottom\_7->addWidget(m\_negative);

bottom\_7->addWidget(m\_1);

bottom\_7->addWidget(m\_2);

bottom\_7->addWidget(m\_3);

bottom\_7->addWidget(m\_plus);

QHBoxLayout \*bottom\_8=new QHBoxLayout;

bottom\_8->addWidget(m\_left);

bottom\_8->addWidget(m\_right);

bottom\_8->addWidget(m\_0);

bottom\_8->addWidget(m\_dot);

bottom\_8->addWidget(m\_equal);

QHBoxLayout \*bottom\_9=new QHBoxLayout;

bottom\_9->addWidget(m\_rand);

bottom\_9->addWidget(m\_rad);

bottom\_9->addWidget(m\_sh);

bottom\_9->addWidget(m\_ch);

bottom\_9->addWidget(m\_th);

//竖直布局

QVBoxLayout \*bottom=new QVBoxLayout;

bottom->addLayout(bottom\_9);

bottom->addLayout(bottom\_1);

bottom->addLayout(bottom\_2);

bottom->addLayout(bottom\_3);

bottom->addLayout(bottom\_4);

bottom->addLayout(bottom\_5);

bottom->addLayout(bottom\_6);

bottom->addLayout(bottom\_7);

bottom->addLayout(bottom\_8);

QVBoxLayout \*top=new QVBoxLayout;

top->addWidget(m\_history);

top->addWidget(m\_in);

top->addWidget(m\_out);

QVBoxLayout \*mainbox=new QVBoxLayout;

mainbox->addLayout(top);

mainbox->addLayout(bottom);

//创建一个窗口，设置窗口的布局

QWidget\* widget=new QWidget(this);

widget->setLayout(mainbox);

//对主窗口设置新的中心窗口，也就是布局好的窗口

this->setCentralWidget(widget);

setWindowTitle("Calculator");

connect(m\_0,&QPushButton::clicked,this,&MainWindow::button\_0);

connect(m\_1,&QPushButton::clicked,this,&MainWindow::button\_1);

connect(m\_2,&QPushButton::clicked,this,&MainWindow::button\_2);

connect(m\_3,&QPushButton::clicked,this,&MainWindow::button\_3);

connect(m\_4,&QPushButton::clicked,this,&MainWindow::button\_4);

connect(m\_5,&QPushButton::clicked,this,&MainWindow::button\_5);

connect(m\_6,&QPushButton::clicked,this,&MainWindow::button\_6);

connect(m\_7,&QPushButton::clicked,this,&MainWindow::button\_7);

connect(m\_8,&QPushButton::clicked,this,&MainWindow::button\_8);

connect(m\_9,&QPushButton::clicked,this,&MainWindow::button\_9);

connect(m\_pi,&QPushButton::clicked,this,&MainWindow::button\_pi);

connect(m\_e,&QPushButton::clicked,this,&MainWindow::button\_e);

connect(m\_dot,&QPushButton::clicked,this,&MainWindow::button\_dot);

connect(m\_plus,&QPushButton::clicked,this,&MainWindow::button\_plus);

connect(m\_minus,&QPushButton::clicked,this,&MainWindow::button\_minus);

connect(m\_multiply,&QPushButton::clicked,this,&MainWindow::button\_multiply);

connect(m\_divide,&QPushButton::clicked,this,&MainWindow::button\_divide);

connect(m\_pow,&QPushButton::clicked,this,&MainWindow::button\_pow);

connect(m\_mod,&QPushButton::clicked,this,&MainWindow::button\_mod);

connect(m\_sin,&QPushButton::clicked,this,&MainWindow::button\_sin);

connect(m\_cos,&QPushButton::clicked,this,&MainWindow::button\_cos);

connect(m\_tan,&QPushButton::clicked,this,&MainWindow::button\_tan);

connect(m\_sqrt,&QPushButton::clicked,this,&MainWindow::button\_sqrt);

connect(m\_log,&QPushButton::clicked,this,&MainWindow::button\_log);

connect(m\_ln,&QPushButton::clicked,this,&MainWindow::button\_ln);

connect(m\_reciprocal,&QPushButton::clicked,this,&MainWindow::button\_reciprocal);

connect(m\_factorial,&QPushButton::clicked,this,&MainWindow::button\_factorial);

connect(m\_time,&QPushButton::clicked,this,&MainWindow::button\_time);

connect(m\_equal,&QPushButton::clicked,this,&MainWindow::button\_equal);

connect(m\_left,&QPushButton::clicked,this,&MainWindow::button\_left);

connect(m\_right,&QPushButton::clicked,this,&MainWindow::button\_right);

connect(m\_clear,&QPushButton::clicked,this,&MainWindow::button\_clear);

connect(m\_clearnum,&QPushButton::clicked,this,&MainWindow::button\_clearnum);

connect(m\_delete,&QPushButton::clicked,this,&MainWindow::button\_delete);

connect(m\_arcsin,&QPushButton::clicked,this,&MainWindow::button\_arcsin);

connect(m\_arccos,&QPushButton::clicked,this,&MainWindow::button\_arccos);

connect(m\_arctan,&QPushButton::clicked,this,&MainWindow::button\_arctan);

connect(m\_exp,&QPushButton::clicked,this,&MainWindow::button\_exp);

connect(m\_ten,&QPushButton::clicked,this,&MainWindow::button\_ten);

connect(m\_negative,&QPushButton::clicked,this,&MainWindow::button\_negative);

connect(m\_sh,&QPushButton::clicked,this,&MainWindow::button\_sh);

connect(m\_ch,&QPushButton::clicked,this,&MainWindow::button\_ch);

connect(m\_th,&QPushButton::clicked,this,&MainWindow::button\_th);

connect(m\_rad,&QPushButton::clicked,this,&MainWindow::button\_rad);

connect(m\_rand,&QPushButton::clicked,this,&MainWindow::button\_rand);

}

MainWindow::~***MainWindow***()

{

delete m\_0;

delete m\_1;

delete m\_2;

delete m\_3;

delete m\_4;

delete m\_5;

delete m\_6;

delete m\_7;

delete m\_8;

delete m\_9;

delete m\_pi;

delete m\_e;

delete m\_dot;

delete m\_plus;

delete m\_minus;

delete m\_multiply;

delete m\_divide;

delete m\_pow;

delete m\_mod;

delete m\_sin;

delete m\_cos;

delete m\_tan;

delete m\_sqrt;

delete m\_log;

delete m\_ln;

delete m\_reciprocal;

delete m\_factorial;

delete m\_time;

delete m\_clear;

delete m\_clearnum;

delete m\_delete;

delete m\_equal;

delete m\_left;

delete m\_right;

delete m\_arcsin;

delete m\_arccos;

delete m\_arctan;

delete m\_exp;

delete m\_ten;

delete m\_sh;

delete m\_ch;

delete m\_th;

delete m\_rad;

delete m\_rand;

delete m\_in;

delete m\_out;

delete m\_history;

}

void MainWindow::**button\_0**(){

text+="0";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_1**(){

text+="1";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_2**(){

text+="2";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_3**(){

text+="3";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_4**(){

text+="4";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_5**(){

text+="5";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_6**(){

text+="6";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_7**(){

text+="7";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_8**(){

text+="8";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_9**(){

text+="9";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_pi**(){

text+="pi";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_e**(){

text+="e";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_dot**(){

text+=".";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_plus**(){

text+="+";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_minus**(){

text+="-";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_multiply**(){

text+="\*";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_divide**(){

text+="/";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_pow**(){

text+="^";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_mod**(){

text+="%";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_sin**(){

text+="sin(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_cos**(){

text+="cos(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_tan**(){

text+="tan(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_sqrt**(){

text+="sqrt(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_log**(){

text+="log(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_ln**(){

text+="ln(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_reciprocal**(){

text+="rec";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_factorial**(){

text+="!";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_time**(){

QDateTime time = QDateTime::currentDateTime();

QString qstr = time.toString("yyyy-MM-dd hh:mm:ss dddd");

text= qstr.toStdString();

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_arcsin**(){

text+="asin(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_arccos**(){

text+="acos(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_arctan**(){

text+="atan(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_exp**(){

text+="e^";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_ten**(){

text+="10^";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_negative**(){

text+="(0-";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_sh**(){

text+="sh(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_ch**(){

text+="ch(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_th**(){

text+="th(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_rad**(){

text+="rad";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_rand**(){

double a=rand();

QString b;

string c;

b=QString::number(a,10,2);//Qstring::number(a,基底，精度)

c=b.toStdString();

text+=c;

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_equal**(){

text+="=";

double a;

string b;

QString c;

m\_in->setText(QString::fromStdString(text));

try {

a=m\_cal.doIt(text);

m\_out->setText(QString::number(a));

history=text;

history+=c.setNum(a).toStdString();

}

catch (string a) {//抛出throw错误被捕获

m\_out->setText(QString::fromStdString(a));

QToolTip::showText(m\_in->mapToGlobal(QPoint(100, 0)), "输入格式错误！");

m\_in->setStyleSheet("QLineEdit{border:1px solid red }");

}

}

void MainWindow::**button\_left**(){

text+="(";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_right**(){

text+=")";

m\_in->setText(QString::fromStdString(text));

}

void MainWindow::**button\_clear**(){

m\_in->setStyleSheet("QLineEdit{border:1px solid black }");

if(text.size()>0){

string s,r;

r+='0';

text=s;

m\_in->setText(QString::fromStdString(text));

m\_out->setText(QString::fromStdString(r));

m\_history->setText(QString::fromStdString(history));}

}

void MainWindow::**button\_clearnum**(){

m\_in->setStyleSheet("QLineEdit{border:1px solid black }");

if(text.size()>0){

while(1){

if(text[text.size()-1]>='0'&&text[text.size()-1]<='9'||text[text.size()-1]=='.'){

string s;

for(int i=0;i<text.size()-1;++i){

s+=text[i];

}

text=s;

}

else{

break;

}

}

m\_in->setText(QString::fromStdString(text));

}

}

void MainWindow::**button\_delete**(){

m\_in->setStyleSheet("QLineEdit{border:1px solid black }");

if(text.size()>0){

string s;

for(int i=0;i<text.size()-1;++i){

s+=text[i];

}

text=s;

m\_in->setText(QString::fromStdString(text));

}

}

void MainWindow::**setBackgroundImage**()

{

QPixmap pixmap = QPixmap("D:/Qt/Qtcode/MyNewCalculator/image.jpg").scaled(this->size());

QPalette palette(this->palette());

palette.setBrush(QPalette::Background, QBrush(pixmap));

this->setPalette(palette);

}

9、main.cpp

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

MainWindow w;

w.setBackgroundImage();

w.show();

return a.exec();

}

**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，并愿意承担由此产生的后果。

作者签字：

时间：2022.04.25

报告成绩

指导老师签字：