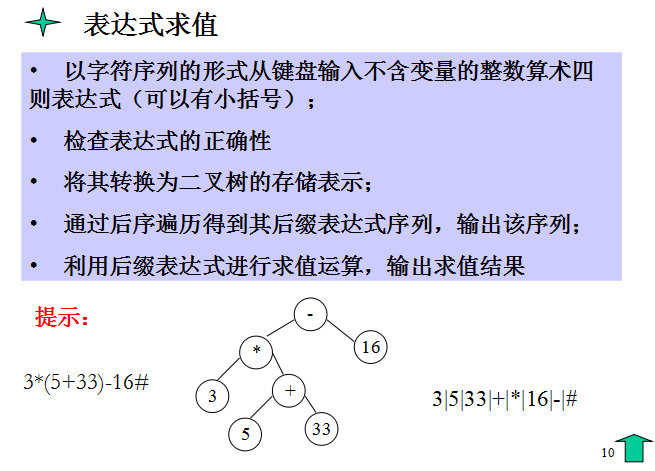
# 四则运算程序文档

* 班级： 2013211313
* 小组成员：彭俊喆 李雄伟 兰玉鹍
* 学号： 2013211535 2013211536 2013211540

## 题目要求及概述：



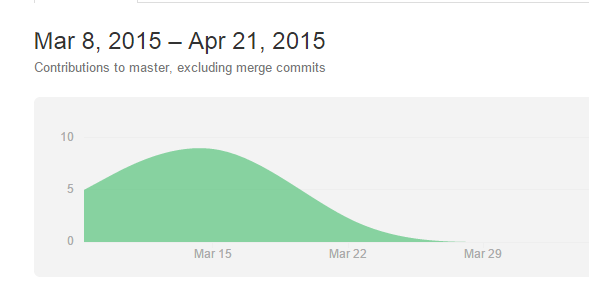
## 分工情况：

* 李雄伟：主要功能编写，自动机查错Check模块，Util模块，Suffix模块第二版，QT计算器界面设计
* 彭俊喆：代码重构，解决代码逻辑性问题，Cell模块编写，Suffix模块最终版，Expression类编写。
* 兰玉鹍：测试程序编写test.cpp，文档编写，Suffix模块第一版，增加Ans功能，Del功能。

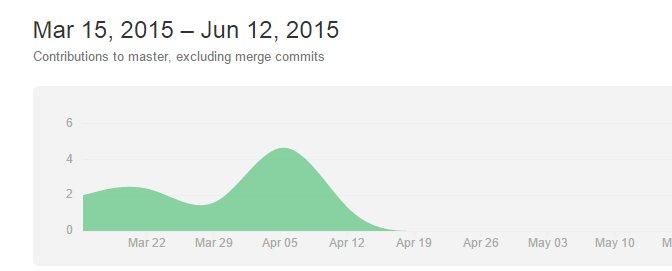
代码统计：

**Contribution**:

* 第一版本

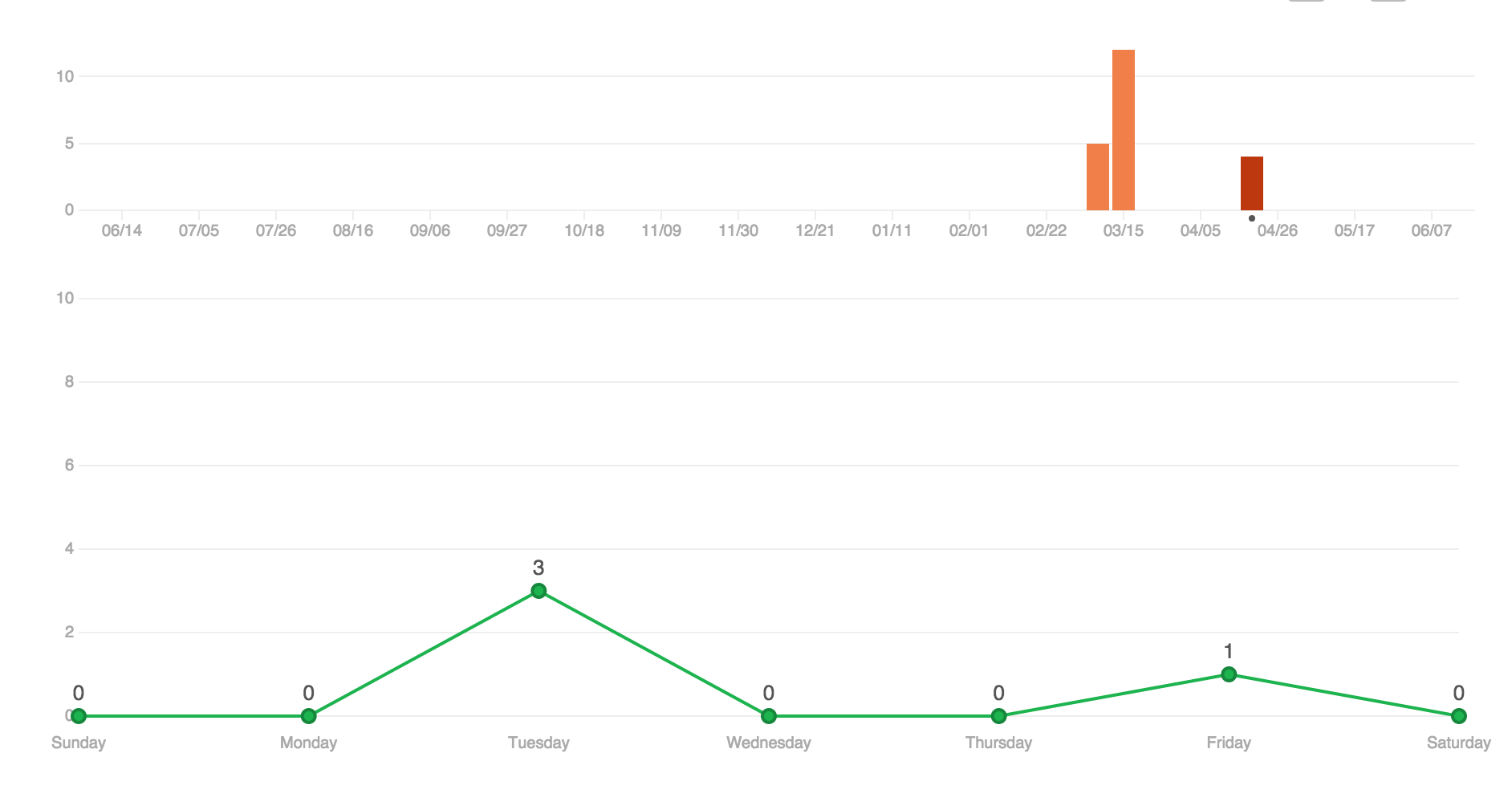


* 第二版本：

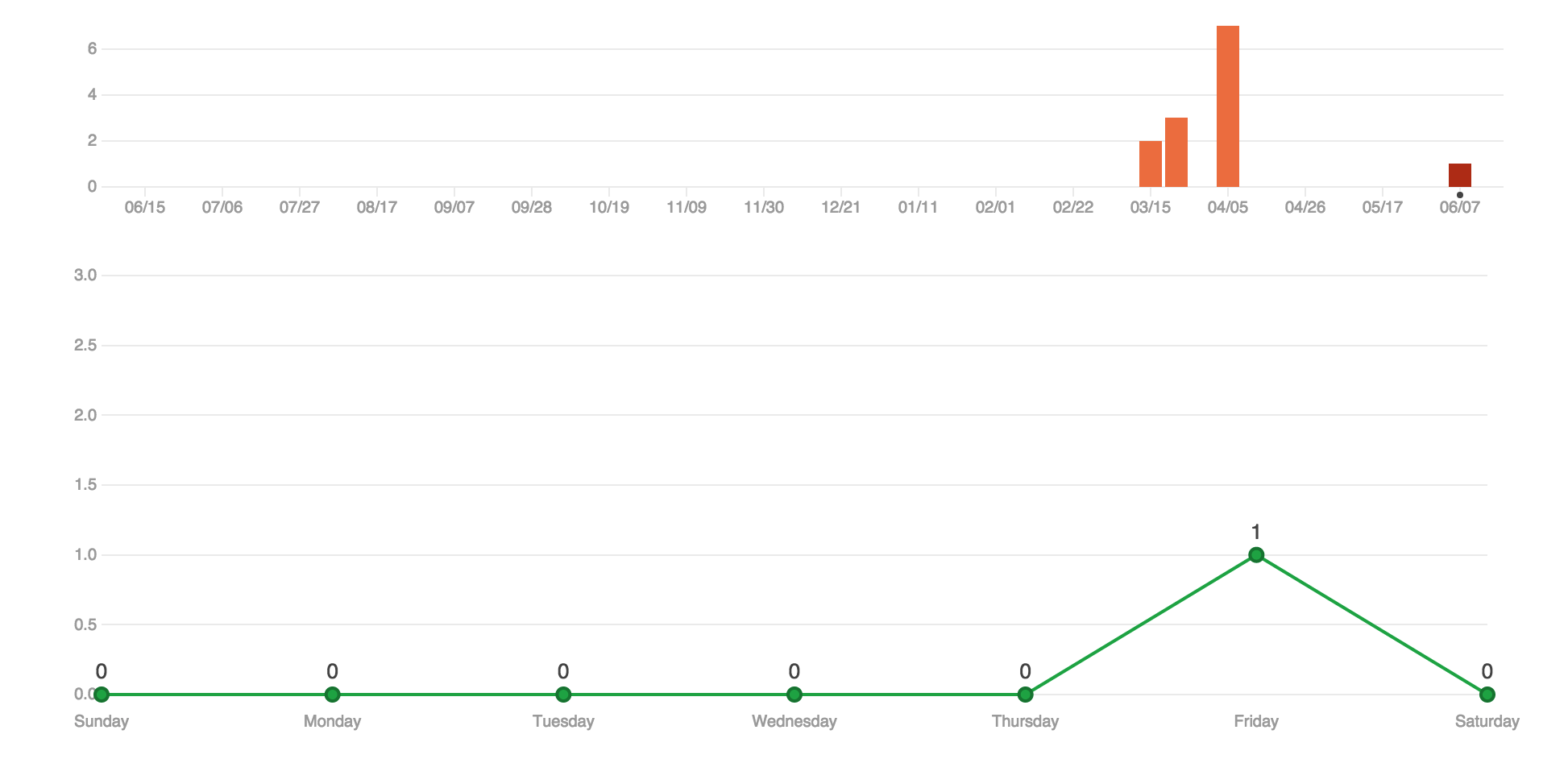


**Commit**：

* 第一版本：

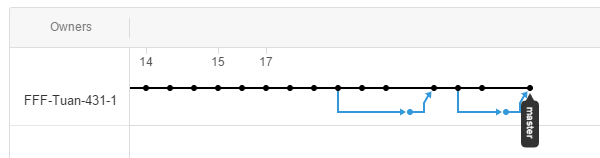


* 第二版本：

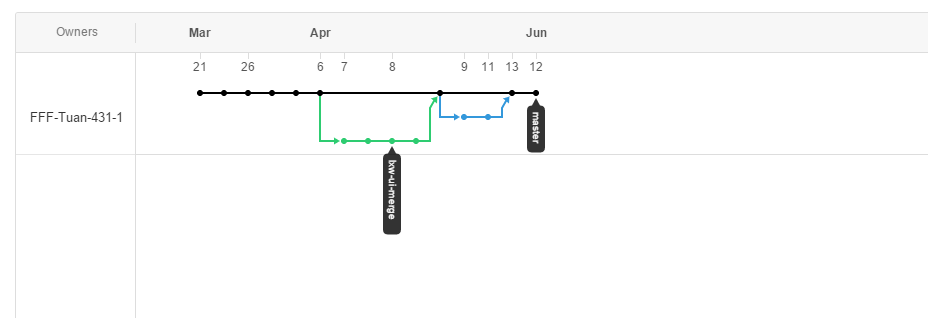


**Branch**：

* 第一版本：

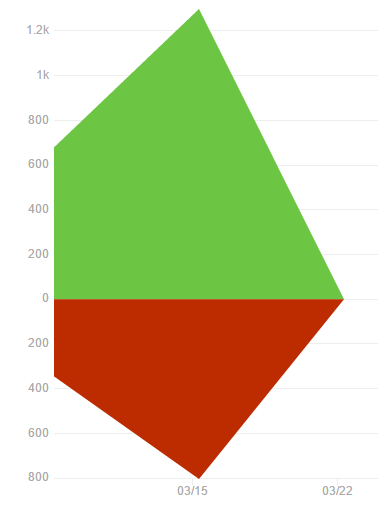


* 第二版本：



**Codes**：

* 第一版本：



* 第二版本：



## 功能需求说明及分析：

**功能需求**：输入中缀四则表达式，最终通过程序得出正确计算结果及后缀表达式，并最终实现程序图形化。

**要点分析**：

1. 表达式转义：通过二叉树将输入的中序顺序的表达式对树进行后序遍历得出后缀表达式。
2. 计算后缀表达式：借助栈并根据数学运算符优先级得出正确结果。
3. 图形化：使用QT Creator设计出计算器面板，并通过Expression类实现C++与QML的交互。

## 代码规范以及工作流：

**代码规范**：我们制定了共同的代码规范，这一点可以在我们的代码中体现，从变量的命名，到缩进，花括号的位置，无一不体现了我们对代码质量的极度追求。

**构建工具**： 我们使用了共同的IDE：Clion，同时，我们使用了跨平台开源的构建系统Cmake，使得我们的代码在任何一个主流平台都可以构建，运行。我们选用了Qt作为图形化的方案，同样，Qt也是一个全平台方案。

**代码管理：**我们采用git作为版本控制工具，使用GitHub作为我们的 代码仓库。这样可以使我们进行快速的分布式迭代开发，并且可以保存每一个阶段的代码，避免不必要的麻烦。

**开发模式：**我们的开发模式为『一次开发，多次迭代，渐进增强』的高效率开发，并且以『重构』为核心。我们的核心代码在最初的几次commit中就已经完成，后来逐步重构: 模块化、对象化、新功能开发、代码复用，都是在不断的重构迭代中完成的。

统计中，我们可以看出，我们新增代码行数为1.2k，删除的代码为1k。

最后，我们在几天内实现了多人合作开发，证明了我们的效率，也证明了我们开发模式的正确。

### 总体方案和设计说明：

总体设计方案：

1. 开发环境：编译器使用MinGW，IDE使用了Clion，以及Qt Creator
2. 总体结构：页面文件（QML）通过委托模式与核心模块（C++代码）交互。
3. 具体模块划分：Calculator.cpp, Cell.cpp, Check.cpp, Suffix.cpp, Util.cpp, Expression.cpp

### 数据结构及数据字典：

我们的核心模块是Suffix模块，这个模块的功能就是将中缀表达式转换成后缀表达式，来辅助计算，所以，我们主要介绍Suffix模块的数据结构。

我们构造了一颗二叉树，这棵二叉树用于以存储表达式，其中序遍历为中缀表达式，而后序遍历就是我们需要的后缀表达式。

在建树的过程中，我们使用了一个栈来辅助建树，因为优先级的不同，所以我们需要把特定的子树放在栈里缓存，而出栈的过程则表明了优先级的先后。

### 各模块设计说明：

1. Calculator类

Calculator类是『计算器』的抽象实现，接受一段表达式的输入，返回表达式结果等。

接口：

* setExpression(string s) 设置计算器需要计算的表达式
* getAnswer() 计算表达式结果
* isError() 判断表达式是否有误
* getSuffix() 获取后缀表达式

1. Expression 类

Expression 类是通过委托模式，沟通Calculator类与QML交互的桥梁，打通API。

接口：

* expression 只写，传入需要计算的表达式
* value 只读，表达式的结果
* suffix 只读，表达式的后缀表达式
* error 只读，表达式是否有错误

1. Check类

Check类用来判断表达式是否是合法的四则运算表达式。我们采用了有限状态自动机，共有四个状态：『数字』、『操作符』、『左括号』、『右括号』。只有在四个状态都顺利进行完毕时表达式为合法表达式，当某一个状态返回errorr则表达式错误。

接口：

* checkError() 判断表达式是否错误

1. Suffix类

用来将中缀表达式，转换成后缀表达式，我们通过二叉树来实现。此处插入二叉树的逻辑

接口：

* getSuffix() 获取后缀表达式

1. Cell类

Cell类是对数据的封装，辅助Suffix类进行转换。

1. Util类

Util类封装了其他类需要用到的公用helper方法，实现代码重用。

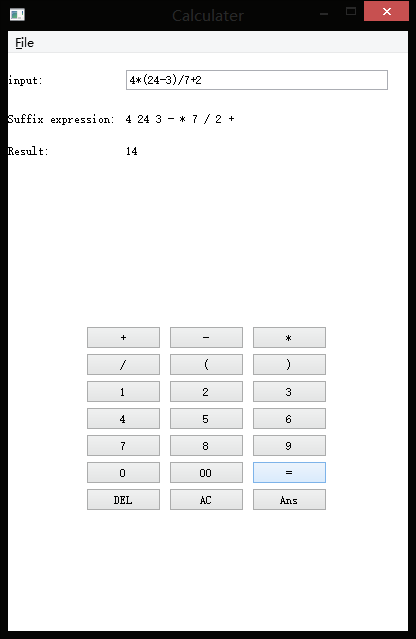
## 测试报告：

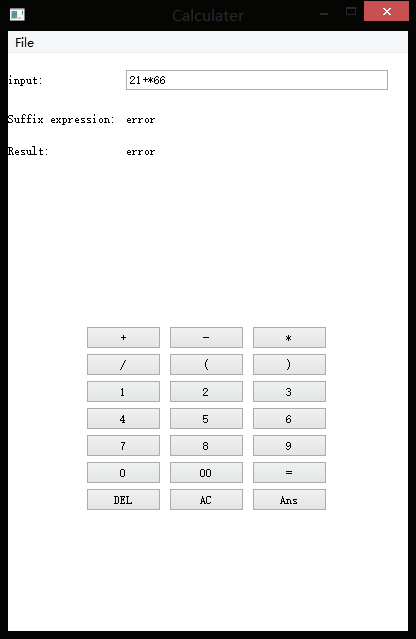
**第一版本：**

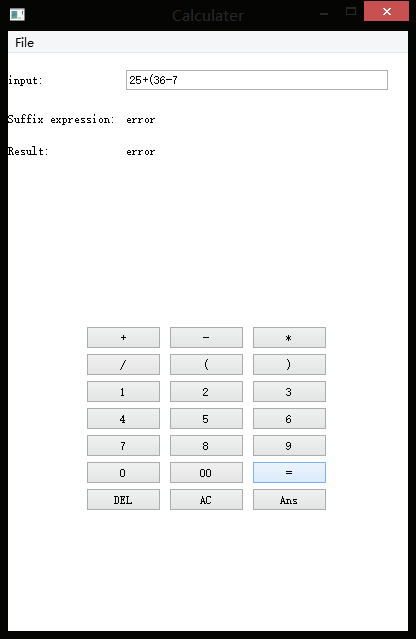
在第一版本中我们使用了断言（Assert）来进行测试，部分测试样例如下：

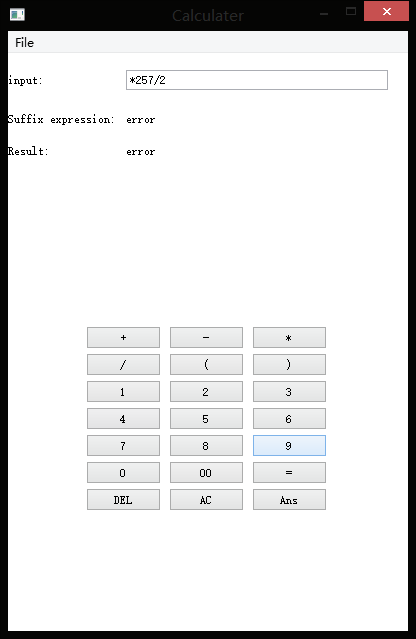
//不得使用括号的非法用法  
calculate.setExpression("(1");  
assert(calculate.isError());  
//任何以右括号为开头的表达式都是非法的  
calculate.setExpression(")");  
assert(calculate.isError());  
//一个合法表达式两边加任意数量合法的左右括号是合法的  
calculate.setExpression("((2+2))");  
assert(!calculate.isError());  
//括号中必须有合法表达式或者数字  
calculate.setExpression("()");  
assert(calculate.isError());  
//左括号之后的第一个字符必须是数字  
calculate.setExpression("(\*");  
assert(calculate.isError());  
//两个操作符连接在一起使用是非法的  
calculate.setExpression("+-");  
assert(calculate.isError());

**第二版本**

第二版本中，我们使用样例测试，部分测试样例如下：  








## 评价和改进意见

我们编写的代码已经实现了基本的四则运算求值，在检验输入的中缀表达式正确后可以进行中缀转后缀的操作和对后缀表达式进行求值。这些功能在第一版本的代码中已经成型。在第二版本的图形化界面里，使用Qt设计出一个类似于计算器的界面，并将QML界面模块与cpp中的功能模块进行衔接，从而实现二者的交互。在第二版本中，我们针对计算器的功能给我们的第二版本添加了DET和ANS两个按钮，甚至还有一个类似于小键盘'00'的按钮，尽量使得我们的图形界面更加友好与人性化。同时我们还将中缀转后缀之后的结果打印到屏幕上，方便用户理解。

但我们也有一些地方需要改进，比如说，我们使用的是QT自带的Botton,虽然看起来比较简洁，也没有什么功能上的缺陷，但是样式看起来还是有待美化的，可以自定义Mybotton样式。而且在一些比较麻烦的检验上也需要改进，比如/0的非法性和负数的合法性。

# 用户使用说明

## 引言

为了让任何人都能轻易的使用本程序，我们特意编写了此用户使用说明

## 软件概述

功能：中缀转后缀表达式和实现四则运算的计算

性能：所有正整数的加减乘除以及加括号运算，不可以运算浮点数和负数

灵活性：加入了ANS和DEL的类似于计算器的功能按钮，可以实现回删和记忆

## 运行环境

1. 在Windows/Linux/OS X 中，使用Qt Creator编译源码，运行程序
2. Windows环境中运行我们打包好的程序

## 使用说明

程序可以使用图形化界面输入，只要点击界面上的虚拟键盘就可以键入按钮所示的内容。

程序可以使用键盘输入，将鼠标移到input处，点击以聚焦，然后可以在键盘上输入所需要输入的数字，括号，以及运算符号，其余字符输入不会提示报错，但当用户试图获取结果时，会触发error事件。

程序关闭，可以点击图形化界面的File按钮，选择Exit，或者直接点击界面右上角的关闭图标。

## 运行说明

在运行环境下运行程序，然后按照使用说明进行操作即可