CS205 C/C++ Programming - Project2 Report

Name: 罗皓予（Haoyu Luo）

SID：12112517

Part 0 - Catalogue

Part 1 - Description

GitHub链接

源码结构

源码概述

1.关于main源码的解释说明

2.关于BigDecimal源码头文件的解释说明

3.关于Calculator源码头文件的说明

Part 2 - Result & Verification

0.欢迎语句

1.不带括号的基本加减乘除

2.带括号的基本加减乘除

3自变量命名并计算

4.数学函数运算

5.无限精度运算

Part 3 - Difficulties&Solutions with code

Problem1:基本数学逻辑运算

1.0 逆波兰算法

1.1 表达式自定义标准格式化

1.2 四种运算规则的优先次序

1.3 后缀表达式的转化

1.4 后缀表达式的计算

1.5 除数为0的情况

Problem2:括号的基本处理

2.1添加了括号的符号栈的运算法则

Problem3:自定义变量

3.0 自定义变量实现思路

3.1 判断语句是否正在赋值

3.2 判断变量命名是否合法

3.3 阻止用户修改常量

3.4 保存变量名和其对应的数值

3.5 修改已定义的变量

3.6 对变量进行值代替

3.7 检查未定义的变量

Problem4:数学函数的支持实现

4.1 处理表达式中的数学函数

4.2 数学函数计算

4.3 TODO

Problem5:高精度的运算实现

Special1:对用户的友好界面提示

1.1 welcome

1.2 about

1.3 help

1.4 exit

1.5 clear

Special2:支持中英双语

Special3:自定义精度

Special4:注释

1.单行注释

2.多行注释判断

Special5:Calculator类

Part 1 - Description

GitHub链接

本次项目源码通过Github仓库提交，以下是仓库链接。

(https://github.com/jimmylaw21/CS205-C-C-Program-Design/tree/main/project2)

源码结构

本次作业共上交12个文件：

1.main.cpp

2.BigDecimal.cpp

3.BigDecimal.h

4.Calculator.cpp

5.Calculator.h

6.Variables.cpp

7.MathFunc.cpp

8.Helper.cpp

9.cmake_install.cmake

10.CMakeCache.txt

11.CMakeLists.txt

12.Makefile

一共2个h文件，6个cpp文件，3个cmake的文件，1个Makefile文件。

源码概述

1.关于main源码的解释说明

0.由于单独设计了Calculator类，并且将大部分功能集成在Calculator类中，所以main的行数很少，

简单调用几个Calculator类的方法即可实现满足所有本次项目已列出的要求的计算器。

1.将Calculator类实例化，此后主要调用Calculator对象cal的方法。

2.main方法的主要运行逻辑在while（true）循环中，仅判断是否退出和是否使用帮助信息，用getline方法获取输入，通过cal.helper()方法和cal.calculate()方法来实现功能，用户输入exit即可退出。

2.关于BigDecimal源码头文件的解释说明

0.【声明】本次作业中的BigDecimal.cpp和BigDecimal.h为Github上查找到的较好的源码库.

仓库链接：(https://github.com/Sam-bit/BigDecimal-CPP)

仓库作者：Sam-bit

本次作业使用其来实现无限精度小数的基本计算，感谢作者的代码贡献(已star)。

1.支持以多种类型包括但不限于unsigned int，long long，double，const char *，string和无类型来创建对象

2.支持将对象转成多种类型，比如string。

3.支持对象转化为人类可读的小数或者整数进行运算

4.支持任意长度的整数和小数的加减乘除运算

5.支持c++的double类型和long long类型的运算符重载

6.支持两个BigDecimal对象的比较大小和赋值

3.关于Calculator源码头文件的说明

0.Calculator类设计参考学习链接1：(https://blog.csdn.net/demo_yo/article/details/112340758)

基本表达式计算参考学习链接2：(https://blog.csdn.net/wanzhen4330/article/details/81952851)

数学函数和帮助信息参考学习链接3：(https://github.com/YeeTone/CS205-2020Fall/tree/main/CS205Assignment2)

1.该类用来运行计算器的几乎所有功能，其源码实现分4个cpp文件，分别负责基本表达式运算，变量赋值和变量表达式计算，数学函数和帮助信息。

2.该类将后缀表达式向量，符号栈，数字栈，变量map，自定义标准格式化表达式，和计算结果设置为private类型封装起来。其余成员属性和方法设置为public类型。

3.该类基本表达式运算部分，使用了后缀表达式向量，符号栈和数字栈，提供了后缀表达式转换，标准格式化表达式，计算后缀表达式等方法，并将这些方法集成在calculate方法中一键调用，支持基本数学加减乘除（Requirement1）和带括号的基本数学加减乘除（Requirement2）。

4.该类变量赋值和表达式运算部分，使用了map<string,string>，以符合c++命名规则的变量名为key，其值为value，保存变量。同时，支持对于不存在的字符自变量的检查并显示错误信息，支持带变量的表达式计算和表达式赋值（Requirement3）。

5.支持部分非复合数学函数的运算（Requirement4）（支持sqrt(a),max(a,b),abs(a),exp(a),min(a,b),random(),sin(x),cos(x),tan(x),ceil(x),floor(x),log(x)，特别声明：指数计算，阶乘和绝对值功能作为运算符 ^, !, | |设计在基本表达式运算部分）【注意：以上函数是基于math.h头文件的相应函数进行设计，因此结果有可能受到计算精确度的限制】

6.支持理论无限精度的数字运算（Requirement5），用户可以自定义输出精度（默认为100）。

7.支持显示欢迎语句，帮助文档和项目详细信息，帮助用户熟悉本计算器具有的功能

9.支持中英双语，支持用户进行两种语言的相互转换

10.支持用户正常退出计算器

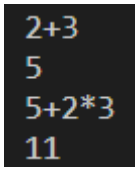
11.支持用户清空冗余的命令行信息

Part 2 - Result & Verification

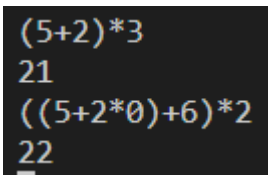
```
【Testcase#0】欢迎语句
Input:nothing
Output:Several sentences for welcome
```

```
--------------------------------------------------------------------------------
Please enjoy your calculation!
If you need help, please print help to get some information.
If you need welcome, please print welcome to get this information again.
If you need exit, please print exit to get exit information.
如果需要显示中文，请输入Chinese
--------------------------------------------------------------------------------
```

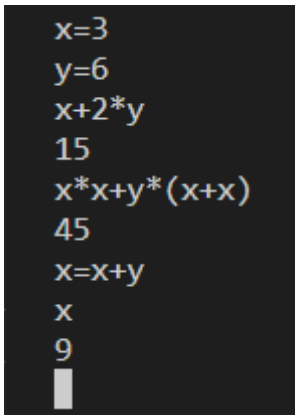
```
【Testcase#1】不带括号的基本加减乘除
Input:2+3 5+2*3
Output:5 11
```



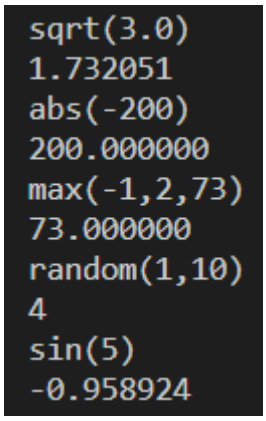
【Testcase#2】带括号的基本加减乘除
Input:(5+2)*3 ((5+2*0)+6)*2
Output:21 22



【Testcase#3】自变量命名并计算
Input:x=3 y=6 x+2*y x*x+y*(x+x) x=x+y x
Output:15 45 9



【Testcase#4】数学函数运算
Input:sqrt(3.0) abs(-200) max(-1,2,73) random(1,10) sin(5)
Output:1.732051 200.000000 73.000000 4 -0.958924



【Testcase#5】无限精度运算
Input:10000000000000000000000000.99999999999999999999999999999999999+1.0
scale = 5
10.0/7
Output:10000000000000000000000001.99999999999999999999999999999999999
1.42857

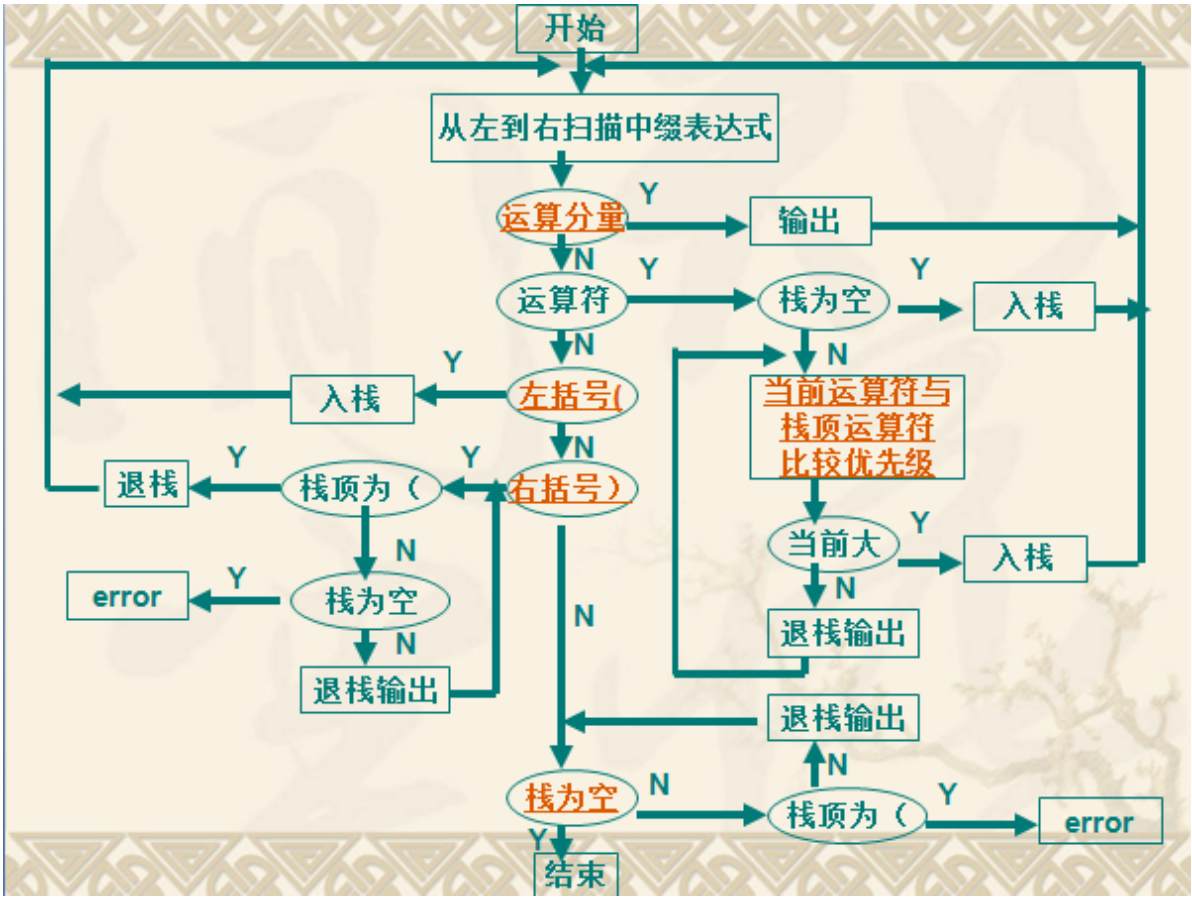
```
10000000000000000000000000.999999999999999999999999999999999+1.0
10000000000000000000000001.999999999999999999999999999999999
scale=5
10.0/7
1.42857
```

Part 3 - Difficulties&Solutions with code

Problem1:基本数学逻辑运算

1.0 逆波兰算法

我们日常所用的数学表达式（如5+3*2）都是中缀表达式，中缀表达式是人容易理解的表达式。后缀表达式又叫做逆波兰表达式，对计算机来说，计算中缀表达式是很困难的，但是计算后缀表达式却非常容易，所以我们先把中缀表达式转化成后缀表达式来计算。下面的算法流程图可以很好的演示整个转换的过程：



（算法流程图源于学习链接2）

该算法将数字和符号分别拆开并压入栈，是本计算机实现表达式计算的核心步骤。

1.1 表达式自定义标准格式化

一, 在处理表达式的过程中，我们会遇到诸如-12，-45.34832的负数和诸如+73847的不寻常的带符号正数，它们会干扰我们生成后缀表达式，所以从左到右扫描表达式infix，将左边为空或括号的正负数转换为0+/0-的形式，赋值给标准格式化表达式stdInfix以便后续处理。

二，我们还需要清除表达式中多余的空格，为后面自定义变量和数学函数支持作铺垫。

三，我们还要防止用户输入为空并输出错误信息。

具体方法：

```
void getFormat();
void clearEmptySpace();
void trim(string &s);
void checkInfix();
```

1.2 四种运算规则的优先次序

构建prior函数和PRIO_LV枚举类：传入类型为char类型，返回类型为PRIO_LV枚举类型， *在传入!的时候返回PRIO_LV4, 在传入*%或者^的时候返回PRIO_LV3，在传入乘号*或者/的时候返回PRIO_LV2，在传入+或者-的时候返回PRIO_LV1，其余返回PRIO_LV0即可。

具体方法：

```
enum PRIO_LV {
  PRIO_LV0 = 0,
  PRIO_LV1 = 1,
  PRIO_LV2 = 2,
  PRIO_LV3 = 3,
  PRIO_LV4 = 4,
};
int getPrior(char c);
```

1.3 后缀表达式的转化

将表达式stdInfix由中缀表达式转化成后缀表达式。

应用自带库：c++的库，以及c++的库

将读取到的数字存入数字栈，特别地将小数.xxx转换成0.xxx存入，将读入到的运算符存入符号栈，然后依据优先级取出并存入后缀表达式的vector对象，同时传入中缀表达式的string对象，返回后缀表达式的vector对象。

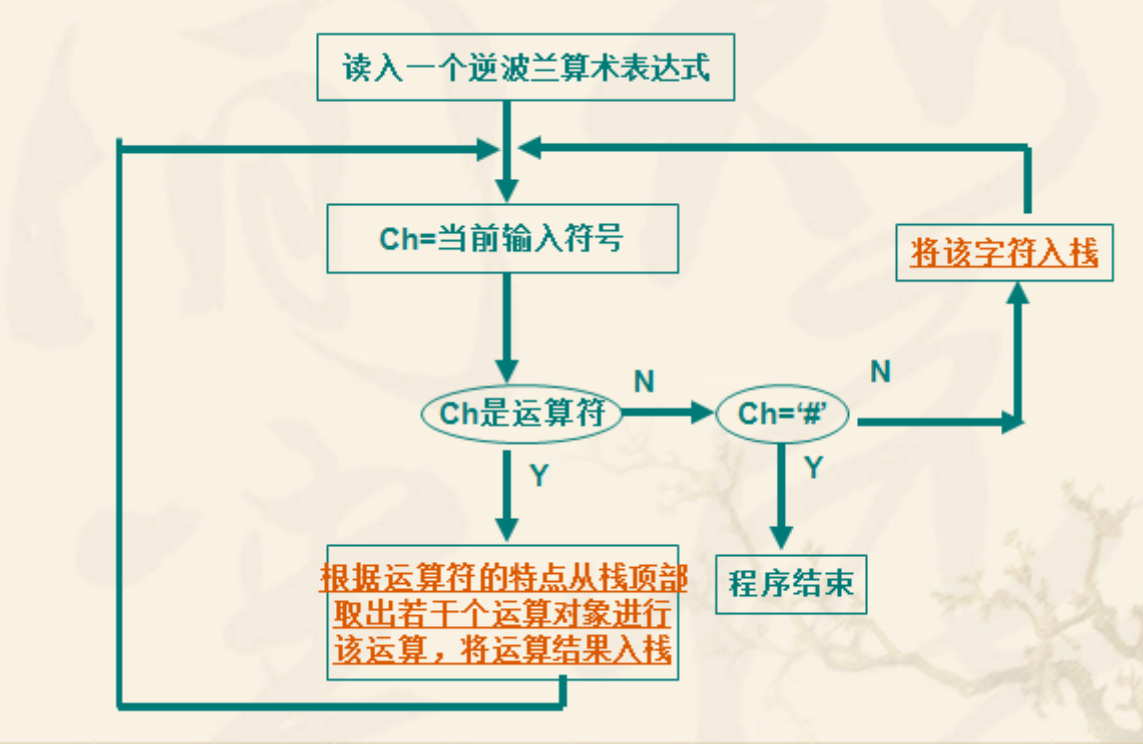
具体方法：

```
void calResult();
```

1.4 后缀表达式的计算

遍历后缀表达式的vector对象postfix，读到运算符就从数字栈中取出string，以数字string初始化BigDecimal对象进行计算，将计算结果转回string压入数字栈，最终取出栈顶的元素即可，将其赋值给private封装的result，用getResult()取出result输出结果。

```
void calResult();
BigDecimal getResult();
```



1.5 除数为0的情况

BigDecimal.cpp文件已经针对除数为0的情形进行了处理，因此直接调用BigDecimal.cpp的除法进行运算即可。

```
【Testcase#6】除数为0情形
Input:5/0
Output:Division By ZERO
```

```
5/0
Division By ZERO
```

Problem2:括号的基本处理

在确保左右括号数目匹配的前提下，在基本的运算法则中，括号括起来的表达式的优先级将被提升，将会优先计算。

添加了括号的符号栈的运算法则：

1.遇到运算符则判断和栈顶的运算等级，如果栈顶等级高则取出栈顶元素计算，并将结果压入栈内，直到栈为空或者栈顶等级不高于运算符

2.遇到左括号直接压栈，此后对于除了右括号的任意运算均直接取出存入向量postfix内；

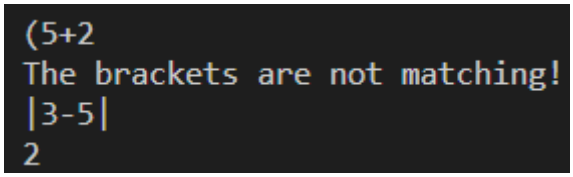
3.遇到右括号则持续出栈运算，直到遇到左括号，将左括号出栈丢弃；

4.最后将栈中剩余符号加入后缀表达式即可；

5.特别地，可以处理( ) [ ] {} 以及绝对值符号| |。

```
void checkBracketMatching();    //在helper部分
void calResult();
```

```
【TestCase#7】左右括号数目不匹配和绝对值运算符计算
Input:(5+2 |3-5|
Output:The brackets are not matching! 2
```



Problem3:自定义变量

3.0 自定义变量实现思路

在方程问题中，我们对表达式的计算是将自变量与其对应的数值进行替换，从而得到无自变量的表达式来计算。SampleQuestion:

已知x=7，请问2*x+4=?

SampleAnswer:

①将所要计算的式子中的x用3替换掉，得到2*7+4

②计算2*7+4=18，得到计算结果为18

基于Sample的思路，实现自变量命名可以使用字符串替代为真正数值的方法来解决该问题。

3.1 判断语句是否正在赋值

赋值表达式有三个基本元素：

1.等号左侧的合法变量名；

2.等号；

3.等号右侧的可计算的表达式

而一般的可计算表达式中不含等号，因此可以通过是否有等号来区别赋值和计算。

利用string类中的find函数寻找等号，若找到等号返回符号下标，若没找到返回-1.

具体方法：

```
bool isAssigning(string expression)
```

3.2 判断变量命名是否合法

首先，利用string类中的find函数寻找等号，截取等号左边的string作为变量名，进行判断其是否合法命名。

C++的合法命名规则为：首个字符必须是下划线或者字母，并且后续组成必须由下划线、数字和字母组成。

其中，如果变量名不合法则终止赋值，并返回错误信息。

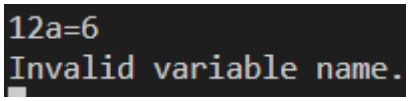
具体方法：

```
bool isVariableNameValid(string expression)
```

```
【TestCase#8】自变量命名不合法

Input: 12a=6

Output: Invalid variable name.
```



3.3 阻止用户修改常量

除了判断用户输入的自变量是否符合命名规则以外，还要判断ta是否在修改常量诸如PI和E，这些常数在Calculator类初始化时就已经存入已保存的变量名了，所以判断输入的变量名是否与常数名相同即可。

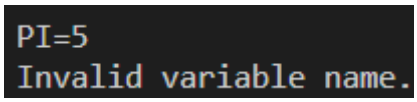
具体方法：

```
bool isVariableNameValid(string expression)
```

```
【TestCase#9】不合法修改常数

Input: PI=5

Output: Invalid variable name.
```



3.4 保存变量名和其对应的数值

由于变量名和变量值有明显的两两对应关系，故我选择使用c++自带的库，通过map结构以变量名为key，变量值为value来储存变量名和变量值。

同时，map有一种类似数组的初始化方式：variables[varName] = varValue;

这种方法可以自动找到map中的key进行赋值，若没有则创建新的key进行赋值，非常simple and beautiful。

具体方法：

```
void Calculator::addVariable(string expression);
```

3.5 修改已定义的变量

修改已定义的变量分为两种情况，一种是修改赋值式如x=x+y。

首先，使用迭代器遍历变量map查看是否先前定义过该变量，然后，存在的话则找对应的下标，然后修改变量名string对应的变量值string即可。

具体实现：

```
string replaceVar(string expression);
string replaceAll(string str, string oldStr, string newStr);
```

3.6 对变量进行值代替

另一种是修改计算式如x*2+y，通过找等号区分两种情况，若是赋值式则只修改等号右侧的变量。

具体实现：

```
string replaceVar(string expression);
string replaceAll(string str, string oldStr, string newStr);
```

3.7 检查未定义的变量

对未定义的变量的检查是在将所有已知变量进行值替换后进行的，所以此时只要检测是否有非法字符存在于表达式即可。具体实现是使用C++自带的库，通过正则表达式进行匹配判断，发现非法字符即返回错误信息"The variable does not exist!"。

具体实现：

```
void checkUnknownVar(string expression);
bool checkArithmeticExpressionValid(string expression);
```

Problem4:数学函数的支持实现

在C++语言中，我们有很多的数学函数可供我们使用，在C/C++的math.h头文件的支持下，实现输入数学表达式的求值。

4.1 处理表达式中的数学函数

先基于find函数检测有无指定的数学函数，然后取出函数体部如sqrt（2.0），对函数中的值如2.0使用math.h中的函数进行计算，最后将函数体替换为结果值，将函数左侧+结果值+函数右侧作为表达式返回。

具体实现：

```
bool containsMathFunc(string expression, string funcName);
string replaceMathFunc(string expression);
```

4.2 数学函数计算

使用stod方法将函数中的值转为double类型，然后调用相应的数学函数计算，最后使用to_string函数转回string类型进行拼接回去即可。

具体实现：

```
string getSqrt(string expression);
```

4.3 TODO

1.复合函数

当前运算不能完全处理复合数学函数处理的情形，只能处理单一数学函数的运算。

现在构想的实现方式有二：

一是改造符号栈由stack变为stack，将数学函数视为运算符，此时处理复合函数原理与处理括号相似；

二是在replaceMathFunc函数内嵌入calculate函数，并进行递归计算，用表达式中不再存在数学函数作为递归终止条件，该方法对时间消耗较大。

2.提高精度

当前运算依靠math.h进行，而使用该库只能得到六位精度的结果。

现在构想的实现方式有一：

改造大数类，增加数学函数的重载，使用大数的计算方法通过数学知识实现无限精度的数学函数计算，

比如sin（），可以将string转为bigDeciaml后，运用泰勒公式得到结果值

Problem5:高精度的运算实现

该运算方法是基于BigDecimal头文件实现的。该类支持：

1.支持以多种类型包括但不限于unsigned int，long long，double，const char *，string和无类型来创建对象

2.支持将对象转成多种类型，比如string。

3.支持对象转化为人类可读的小数或者整数进行运算

4.支持任意长度的整数和小数的加减乘除运算

5.支持c++的double类型和long long类型的运算符重载

6.支持两个BigDecimal对象的比较大小和赋值

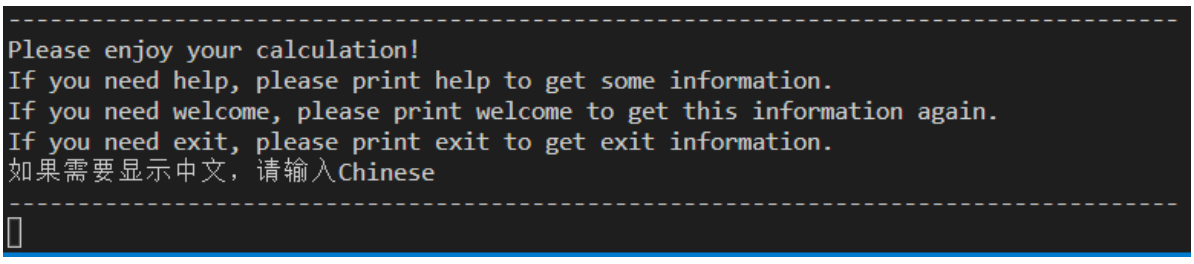
Special1:对用户的友好界面提示

1.1 welcome

当用户初次打开计算器或者输入welcome的时候，会显示对用户的欢迎信息，具体方法实现：

```
void welcome()
```

实现效果如下：



1.2 about

当用户输入about的时候，会显示对项目的相关信息，具体方法实现：

```
void about();
```

实现效果如下：

```
about
--------------------------------------------------------------------------------
This calculator is mainly for the Assignment2 in CS205(C/C++) course.
This calculator supports basic operations including exponents and factorials,
as well as variable assignment calculations and mathematical functions
Course Instructor: Shiqi Yu
Author: Jimmy Luo
Version: 1.00
--------------------------------------------------------------------------------
```

1.3 help

当用户输入help的时候，会显示对用户的帮助信息，具体方法实现：

```
void help();
```

实现效果如下：

```
0. The constant value in this calculator:
E=2.718281828459
PI=3.1415926535898

Supported functions:
1. Basic mathematical calculation: + - * / % ^ ! ||
example-1.1:
3*4
12
example-1.2:
3.45*4.32
14.904

2. calculation with brackets: ( ) [ ] { }
example-2.1:
(2+5)*9.9
69.3
example-2.2:
(4.5+6.8)*1.2
13.56

3. Definition with single char variable: x, y, z
example-3.1:
x=3
y=6
x+2*y
21
example-3.2:
x=1
y=x*2
x=3
x+x*y
9
example-3.3:
x=3
y=4
x=x+y
x
7

4. Some single math functions:
example-4.1:
sqrt(3)
1.732051
example-4.2:
sin(2)
0.909297
sqrt(x), pow(a,b), max(a,b), abs(x), exp(x),
min(a,b), random(), sin(x), cos(x), tan(x), ceil(x), floor(x), log(x),
is supported
However, the compound calculation is not supported.
If input a compound calculation, the calculator will exist.
such as: pow(pow(2,3),4)

5. Arbitrary Precision:
Any length of integer or decimal value are supported in this calculator.
example-5.1:
5.444444444444444444444444444444444444444*2+1.111111111111111111111111111111111111111
```

1.4 exit

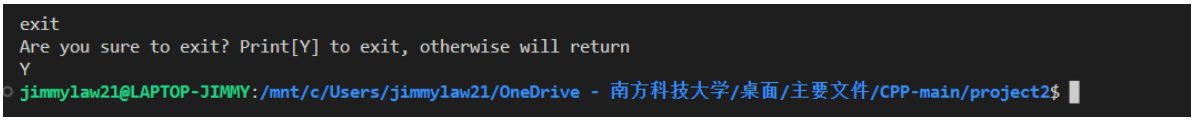
当用户输入exit的时候，会显示是否确认退出。

通过用bool变量来终止main中的while循环，如果是处于要退出的状态则按Y退出，否则取消要退出的状态。

具体实现：

```
void exit();
```

实现效果如下：

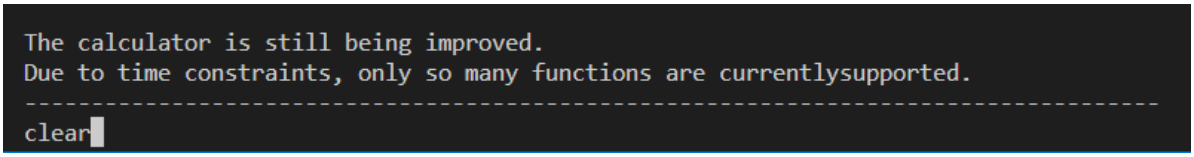


1.5 clear

当用户输入clear的时候，会调用system(clear)，清空命令行中的信息，具体方法实现：

```
void clear();
```

实现效果如下：



Special2:支持中英双语

本计算器适用的人群应当包括英语水平没有那么好的用户，因此应当实现中文、英文两种语言乃至更多语言的切换。

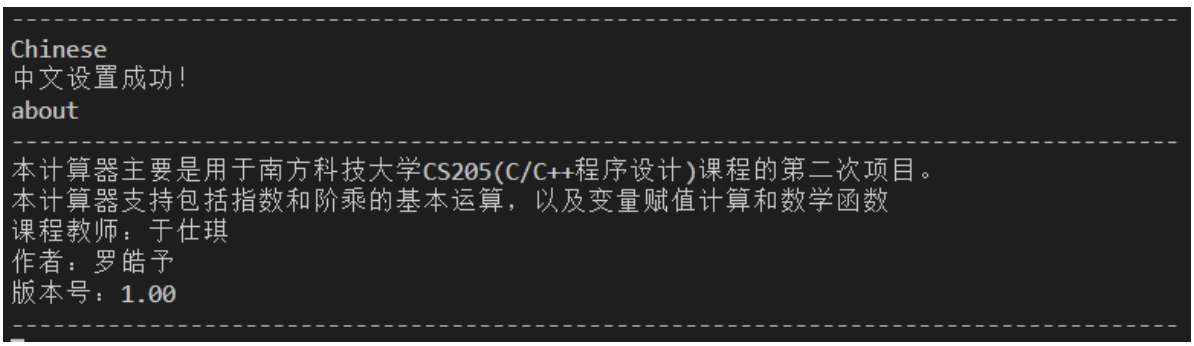
本计算器的中文辅助部分曾经参考过其它中文计算器的语言设置，但并未遇到中文乱码等问题，故没有特殊处理

参考链接：https://www.cnblogs.com/roadwide/p/10533594.html

具体实现：

```
void setLanguage();
```

实现效果如下，处于篇幅考虑，仅展示部分情况：



Special3:自定义精度

本计算器可以实现无限精度，但是在现实使用中用户很可能不需要过于准确的输出，因此提供给用户自定义精度的方法供用户使用，它通过修改大数类中的静态变量_scale来实现，用户输入scale=？即可修改精度。精度默认为100位。

具体实现：

```
void setscale();
```

实现效果如下：

```
2/7
0.2857142857142857142857142857142857142857142857142857142857142857142857142857142857142857142857142857
scale=10
2/7
0.2857142857
```

Special4:注释

本计算器允许用户输入单行注释和多行注释

1.单行注释判断：

前2个字符是否都为/，如果是则isHelp = true，不再继续向下读。

具体方法实现：

```
bool isSingleAnnotation();
```

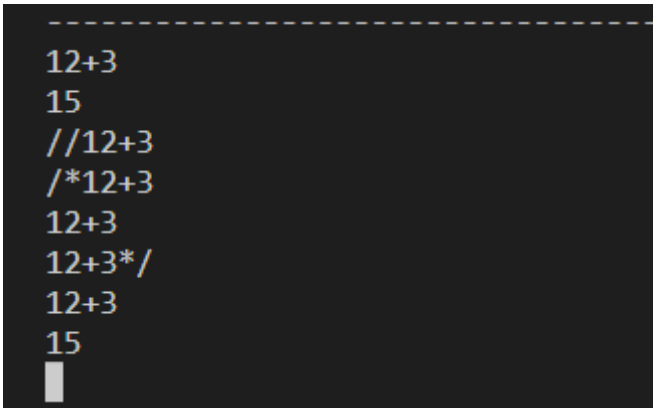
2.多行注释判断：

infix的前2个字符是否为/*，*则isAnnotating=true，若infix*的后两个字符为*/，则isAnnotating=false。isAnnotating=true，则isHelp=true，期间所有读入的内容全部视为注释。

具体方法实现：

```
 bool isMultiAnnotationStart();
 bool isMultiAnnotationEnd();
```

实现效果如下：



Special5:Calculator类

本计算器代码实现上设计并使用了类和对象，创造了calculator类，其由Calculator.h，Calculator.cpp，Variables.cpp，MathFunc.cpp四个文件组成，使得代码结构更加清晰美观,但是在代码组织上仍有不足.

TODO：

1.部分方法以表达式缓存infix为核心进行设计，如void getPostfix()，其余方法则与类的静态变量没有直接联系，如void addVariable(string expression），应当将处理表达式的方法尽可能设计为处理表达式缓存infix的void方法。‘

2.部分方法内部实现重复度很高，可以再抽象一层来减少冗余代码，如以string getSqrt(string expression)为代表的数学函数处理方法。

以下为Calculator.h：

```
// Calculator.h: 头文件
#include "BigDecimal.h"
#include <cmath>
#include <map>
#include <stack>
#include <string>
#include <vector>
using namespace std;

// 计算器类
class Calculator {
public:
  Calculator(); // 构造函数

  // 基本表达式计算
  void getFormat();       // 表达式自定义标准格式化
  int getPrior(char c);   // 获取算术符号优先级
  void getPostfix();      // 后缀表达式转换
  void calResult();       // 计算后缀表达式
  void calculate();       // 计算方法
  BigDecimal getResult(); // 获取结果

  // 变量表达式计算
  bool isAssigning(string expression);          // 检查是否为赋值表达式
  bool isVariableNameValid(string expression); // 检查变量名是否合法
  void addVariable(string expression);          // 添加变量
  string replaceVar(string expression);         // 替换变量
  void checkUnknownVar(string expression); // 检查表达式中是否有未知变量
  void pretreat(string expression);        // 预处理表达式
  void Assign();                           // 赋值方法
```

```
// 数学函数计算
bool containsMathFunc(string expression,
                      string funcName); // 检查表达式中是否包含某函数
string replaceMathFunc(string expression); // 替换表达式中所有数学函数

string getSqrt(string expression);    // 处理sqrt函数
string getMax(string expression);     // 处理max函数
string getMin(string expression);     // 处理min函数
string getRandom(string expression); // 处理random函数
string getExp(string expression);     // 处理exp函数
string getAbs(string expression);     // 处理abs函数
string getSin(string expression);     // 处理sin函数
string getCos(string expression);     // 处理cos函数
string getTan(string expression);     // 处理tan函数
string getCeil(string expression);    // 处理ceil函数
string getFloor(string expression);   // 处理floor函数
string getLog(string expression);     // 处理log函数

// 工具函数
vector<string> split(const std::string &strIn, char delim); // 字符串分割
void trim(string &s); // 去除string所有空格
string replaceAll(string str, string oldStr,
                  string newStr); // 替换str中所有相同的oldStr为newStr

// 小助手
void helper();       // 帮助系统
void welcome();      // 欢迎界面
void help();         // 帮助信息
void about();        // 关于信息
void exit();         // 退出程序
void setLanguage(); // 设置语言
void setscale();     // 设置精度
void clear();        // 清空屏幕
```

```
    void checkInfix();              // 检查表达式是否合法
    void clearEmptySpace();         // 清除表达式中的空格
    void checkBracketMatching(); // 检查括号是否匹配
    bool checkArithmeticExpressionValid(string expression);

    void printWelcomeEnglish(); // 打印欢迎界面英文版
    void printWelcomeChinese(); // 打印欢迎界面中文版
    void printHelpEnglish();    // 打印帮助信息英文版
    void printHelpChinese();    // 打印帮助信息中文版
    void printAboutEnglish();   // 打印关于信息英文版
    void printAboutChinese();   // 打印关于信息中文版

    bool isSingleAnnotation();      // 检查是否为单行注释
    bool isMultiAnnotationStart(); // 检查是否为多行注释开始
    bool isMultiAnnotationEnd();   // 检查是否为多行注释结束
    void annotationCheck();         // 检查是否为注释

    // 数据成员
    string operatorSym; // 运算符号
    string infix;       // 表达式缓存
    bool isExit;        // 是否退出
    bool isHelp;        // 是否帮助
    bool isAnnotating;  // 是否注释

private:
    int language;                   // 语言选择
    vector<string> postfix;         // 后缀表达式向量
    stack<char> symStack;           // 符号栈
    stack<BigDecimal> figStack;     // 数字栈
    map<string, string> variables; // 变量表
    string stdInfix;                // 自定义标准格式化表达式
    BigDecimal result;              // 最终计算结果
};
```

Part 4 - Conclusion

本次project2作为一个c++新手项目，实现难度和质量上下差很大，考察的是多方面的综合能力：

1. 代码架构能力：类和对象的设计与使用；

2. c++的string类的处理操作：复刻了一些java里有而c++里没有的方法，如trim，split，replaceAll；

3. 静态成员的声明与使用；

4. c++中stack，vector，map数据结构的使用；

5. 产品设计能力：考虑用户的体验；

6. 异常情形的识别与处理；

7.编码能力和Debug能力；

8.正则表达式的使用；