实 验 报 告

实验人：吴晨康 学号：16337247 日期： 2017/10/8

院（系）：数据科学与计算机学院 班级：计科7班

实验题目：Project 1 —— 一个多项式计算器的实现

1. 实验目的

锻炼Coding能力，熟悉项目的开发过程

1. 实验环境

Visual Studio

1. 实验内容

设计一个简单的多项式计算器

计算器的功能菜单如下：

1. 输入一个多项式
2. 显示一个多项式
3. 两个多项式相加
4. 两个多项式相减
5. 两个多项式相乘
6. 判断两个多项式是否相等
7. 求多项式的导数
8. 求多项式带入某点的值
9. 退出系统
10. 显示菜单
11. 分析与设计

4.1 需求分析

设计一个简单的多项式计算器，该计算器具有如下功能

1. 输入一个多项式
2. 显示一个多项式
3. 多项式相加
4. 多项式相减
5. 多项式相乘
6. 多项式相等判断
7. 多项式求导
8. 多项式求值
9. 显示菜单

系统功能图：

多项式计算器

多项式求值

显示菜单

多项式求导

多项式比较

多项式相乘

多项式相减

多项式相加

显示多项式

输入多项式

储存结果？

该项目的目的为制作一个多项式计算器。依照功能可分为九大模块，功能分别为输入多项式、显示多项式、多项式相加、多项式相减、多项式相乘、多项式比较、多项式求导、多项式求值和显示菜单。

输入多项式模块支持以小括号内的系数-幂值对来输入多项式，也可以同时输入多项式的名称。在输入时，该模块会同时检测输入多项式及其名称（如果有输入）的合法性，若输入不合法，则不做任何操所而跳出；若输入合法，则在屏幕上打印出多项式，并询问是否存储改多项式，并根据用户选择做出相应操所

多项式的各个运算模块分别从用户处取得多项式并进行相应的运算，并询问是否存储运算结果。注意这里的多项式既可以是系数-幂值对形式也可以是已存储的有名称的多项式。

显示菜单模块打印出计算器的功能菜单。

4.2 类结构设计

多项式计算器

多项式

项

class Calculator {

public:

/\*The main process of the calculator

used for obtaining options from user and call related functions\*/

void mainProcess(); //The main process of the calculator

private:

map<string, Polynomial> storedPoly; //A map used to store polynomials

void welcomeMenu()const; //Print one piece of welcome message when the program is launched

void helpMenu()const; //Print the help menu including the functions of the calculator

void inputPoly(); //The module to input a polynomial

void polyAdding(); //The module to add up two polynomial

void polySubtracting(); //The module to subtract one polynomial from another

void polyMultipleWithConst(); //The module to multiple a polynomial by a constant value

void calculateValue(); //The module to calculate the value of a polynomial in a particular point

void showStoredPoly(); //The module to show all stored polynomials

void polyMultipleWithPoly(); //The sub module to multiple a polynomial with another

void polyCompare(); //The sub module to compare whether tow polynomials are equal

void polyDerivation(); //The sub module to calculate the derivation of a polynomial

void getOption(int& option); //A background service called to get user’s option

void callModule(int option); //A background service called to call main modules

void storePoly(string, const Polynomial&); //A background service called to store a polynomial with a particular name

inline Polynomial recognizePoly(string); //A background service called to translate a string to a valid polynomial

string nameInput(); //A background service to obtain a valid polynomial name from the user

bool nameCheck(string); //A background service to judge whether a polynomial name is valid

};

4.3 接口设计

该计算器程序的主要接口类为Calculator类，要使用该多项式计算器只需实例化一个Calculator对象并调用它的mainProcess()方法。

class Calculator {

public:

/\*The main process of the calculator

used for obtaining options from user and call related functions\*/

void mainProcess(); //The main process of the calculator

private:

map<string, Polynomial> storedPoly; //A map used to store polynomials

void welcomeMenu()const; //Print one piece of welcome message when the program is launched

void helpMenu()const; //Print the help menu including the functions of the calculator

void inputPoly(); //The module to input a polynomial

void polyAdding(); //The module to add up two polynomial

void polySubtracting(); //The module to subtract one polynomial from another

void polyMultipleWithConst(); //The module to multiple a polynomial by a constant value

void calculateValue(); //The module to calculate the value of a polynomial in a particular point

void showStoredPoly(); //The module to show all stored polynomials

void polyMultipleWithPoly(); //The sub module to multiple a polynomial with another

void polyCompare(); //The sub module to compare whether tow polynomials are equal

void polyDerivation(); //The sub module to calculate the derivation of a polynomial

//Other background service not list here.

}；

4.4 数据成员设计

主要数据成员为一个以string（多项式名称）为Key，Polynomial为值的map。Polynomial类的设计如下：

class Polynomial {

public:

//Constructor of the class Polynomial

Polynomial::Polynomial() {}

Polynomial(vector<pair<int, double> >& poly);

//The operator overloading of polynomial adding subtracting and multipling

Polynomial operator+(const Polynomial&)const;

Polynomial operator-(const Polynomial&)const;

Polynomial operator\*(const Polynomial&)const;

Polynomial operator\*(double multiCoffic)const;

//The overloading of operator "=" on the polynomial class

const Polynomial& operator=(const Polynomial&);

//The overloading of operator "[]" to get the cofficient for power i of the polynomial class

double operator[](int i)const;

//The operator overloading to judge whether two polynomials are equal

bool operator==(const Polynomial&);

bool operator!=(const Polynomial&);

Polynomial derivative()const; //The function to calculate the derivative of the polynomial

double value(double base)const; //The function to calculate the value of the polynomial in a particular point

bool empty()const; //The function to judge whether the polynomial is a zero polynomial

string printableString()const; //The function to return a printable string representing this polynomial

private:

vector<pair<int, double> > myPoly; //The array to store the cofficients of the polynomial

};

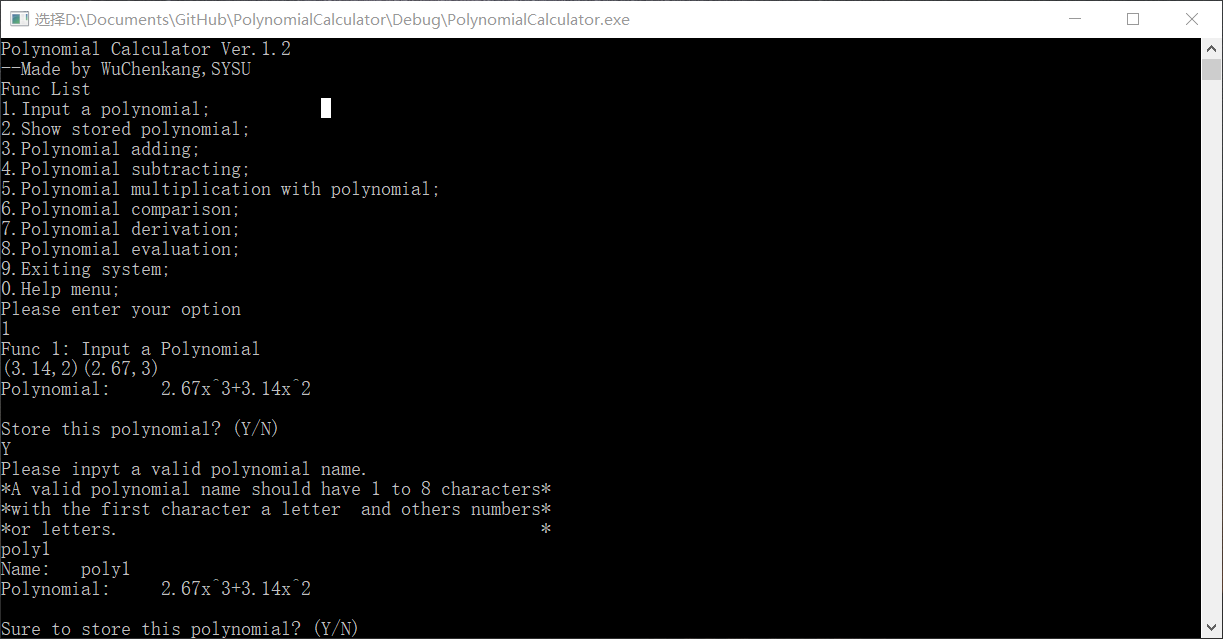
//The operator overloading to calculate a polynomial multiple a constant value

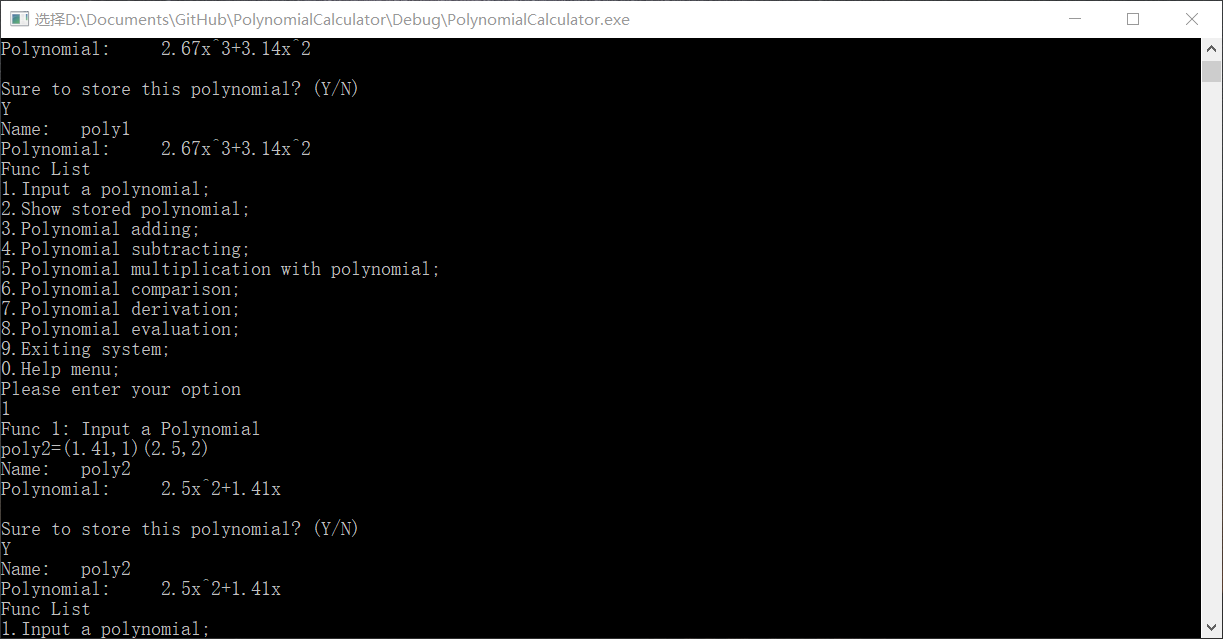
Polynomial operator\*(double multiCoffic, const Polynomial&);

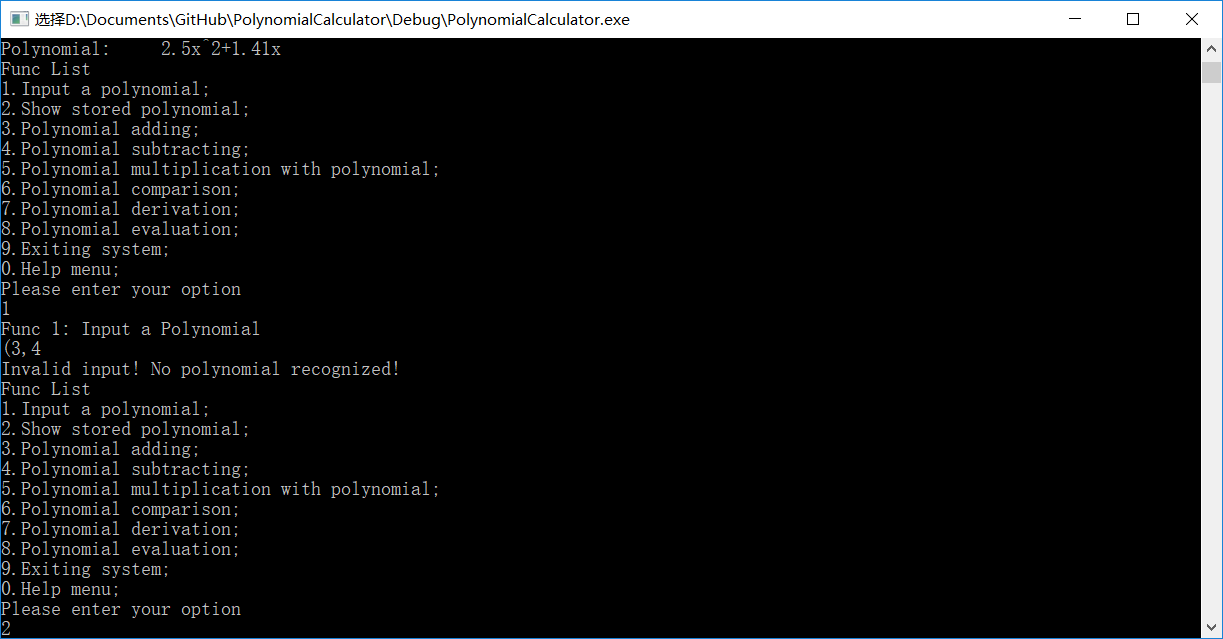
1. 实验结果

设计有代表性的输入数据，并分析测试结果以及针对测试中发现的问题进行修正和改进

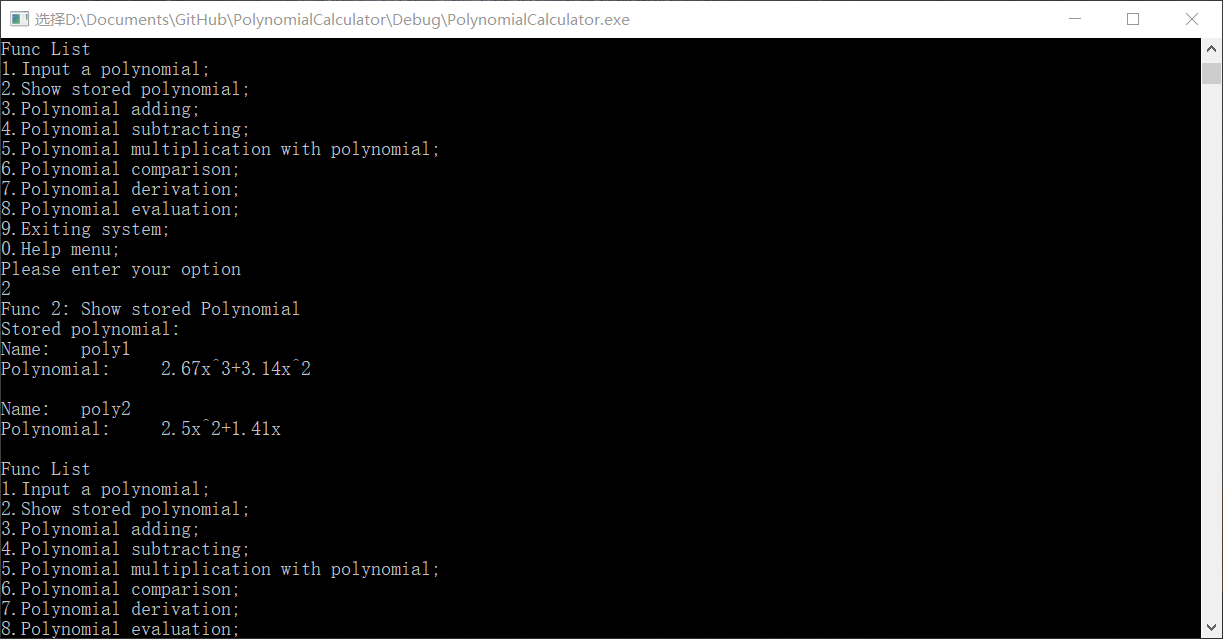
1. 输入多项式 无误，并可对错误输入给出提示



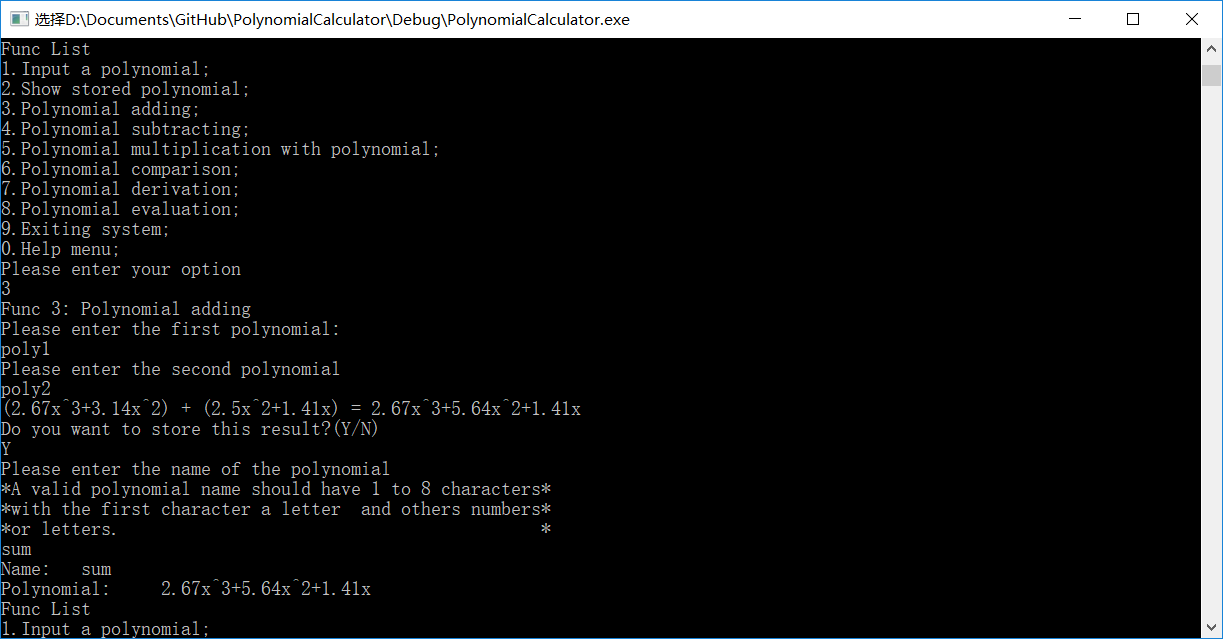




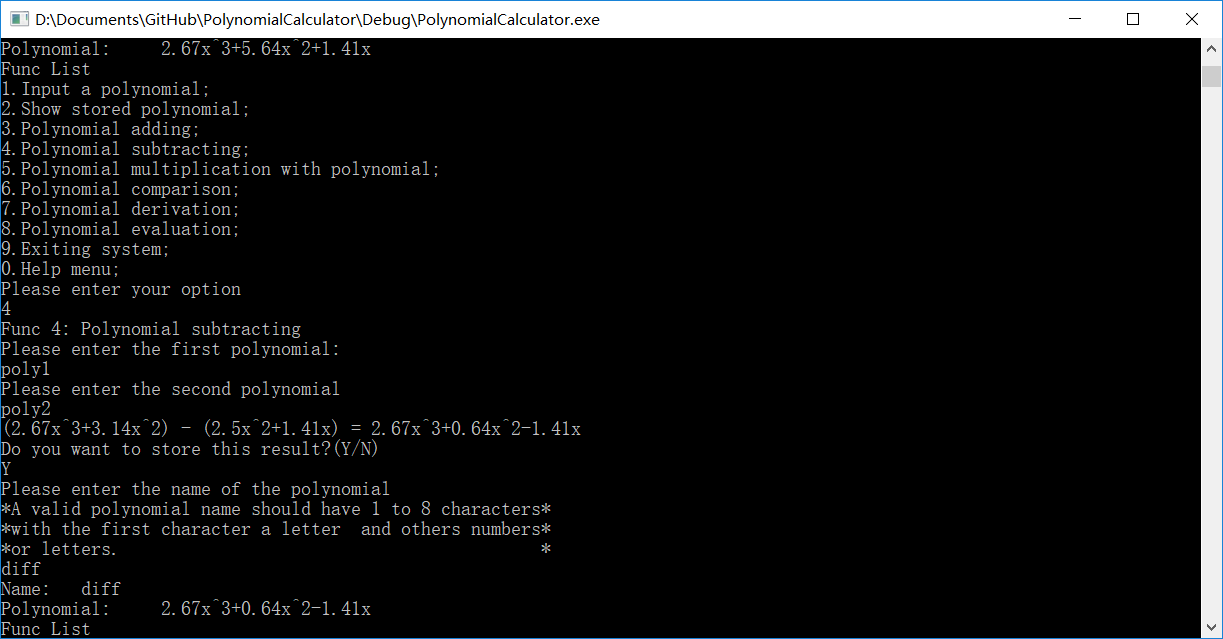
1. 显示多项式 无误，正确的显示出所有存储的多项式



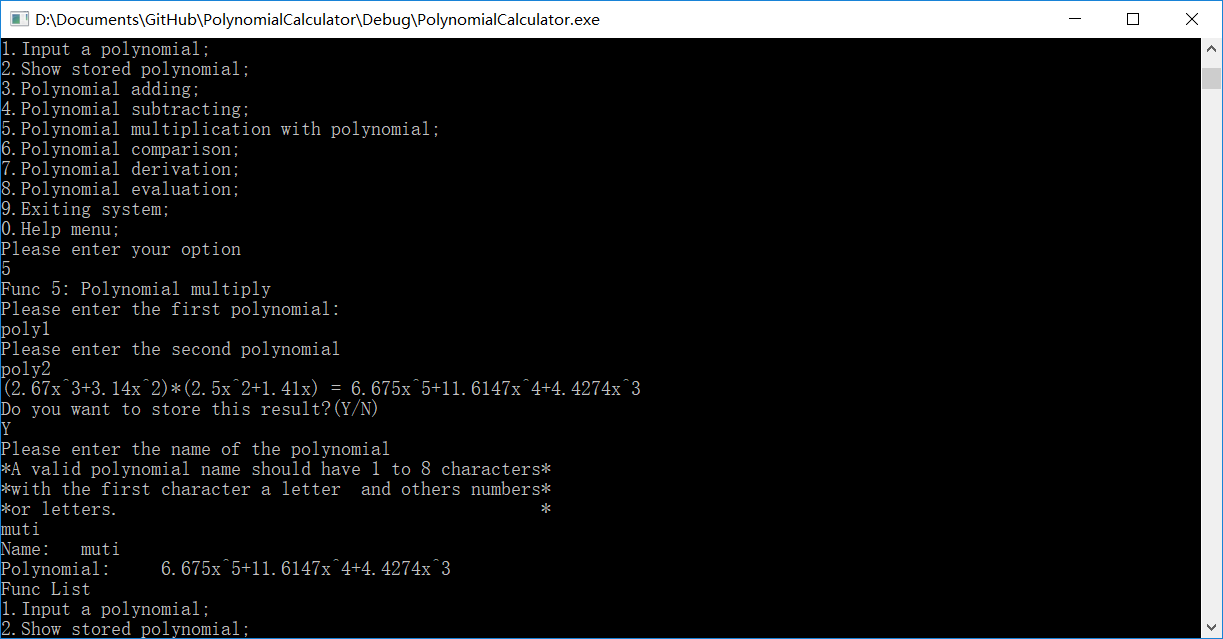
1. 多项式加法 无误



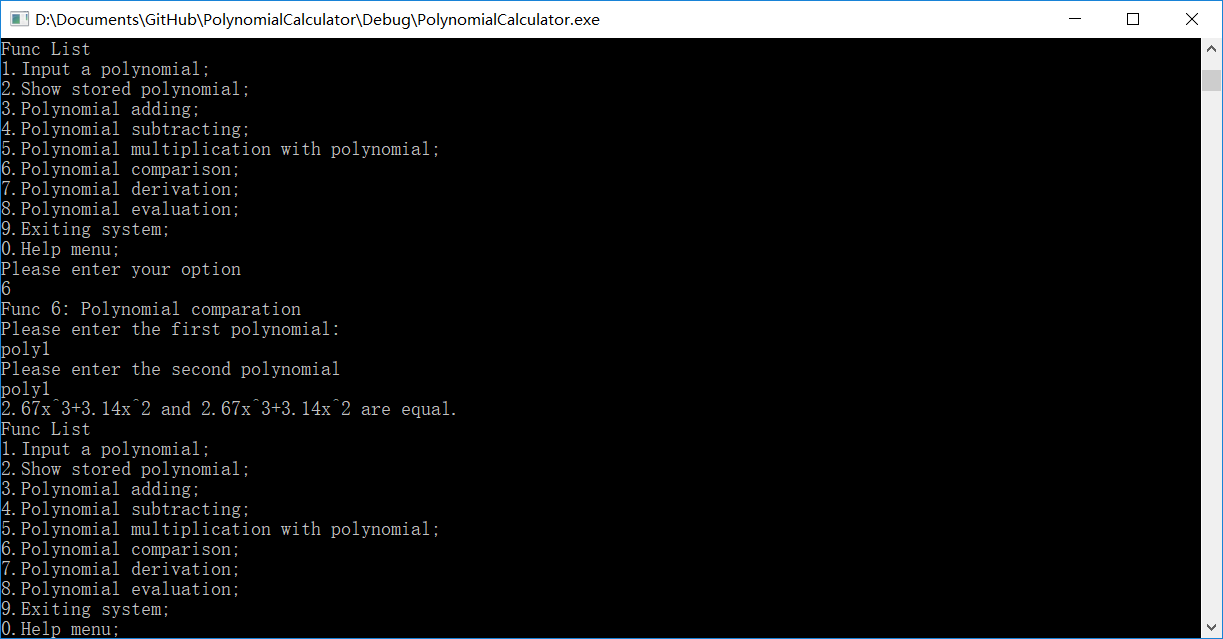
1. 多项式减法 无误

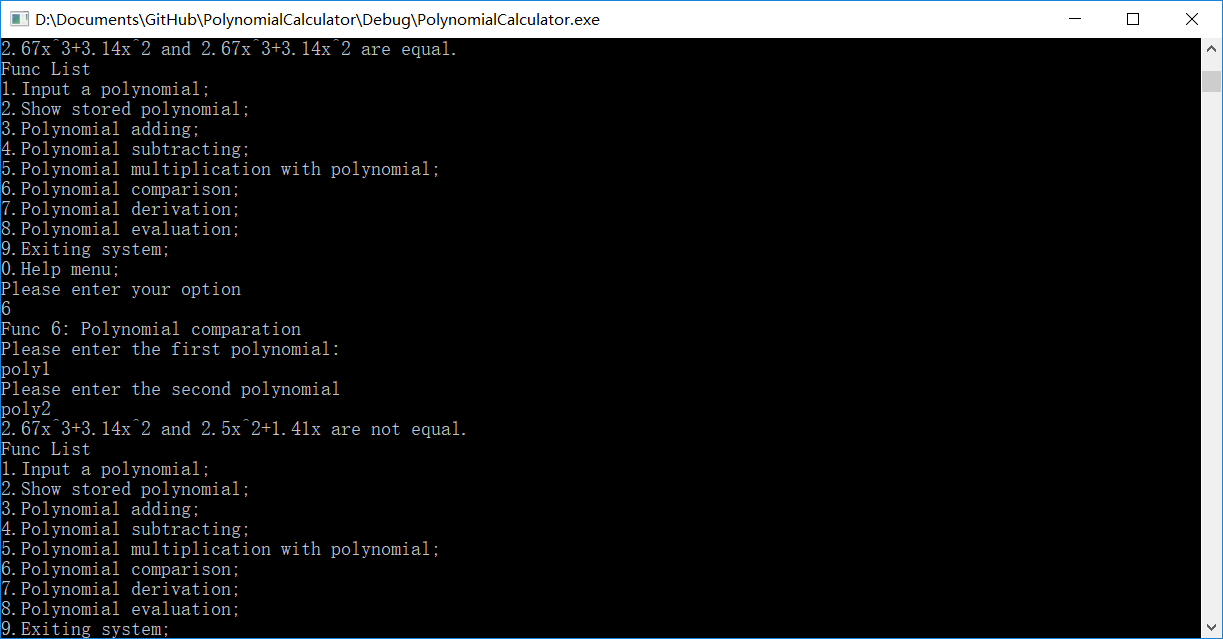


1. 多项式乘法 无误

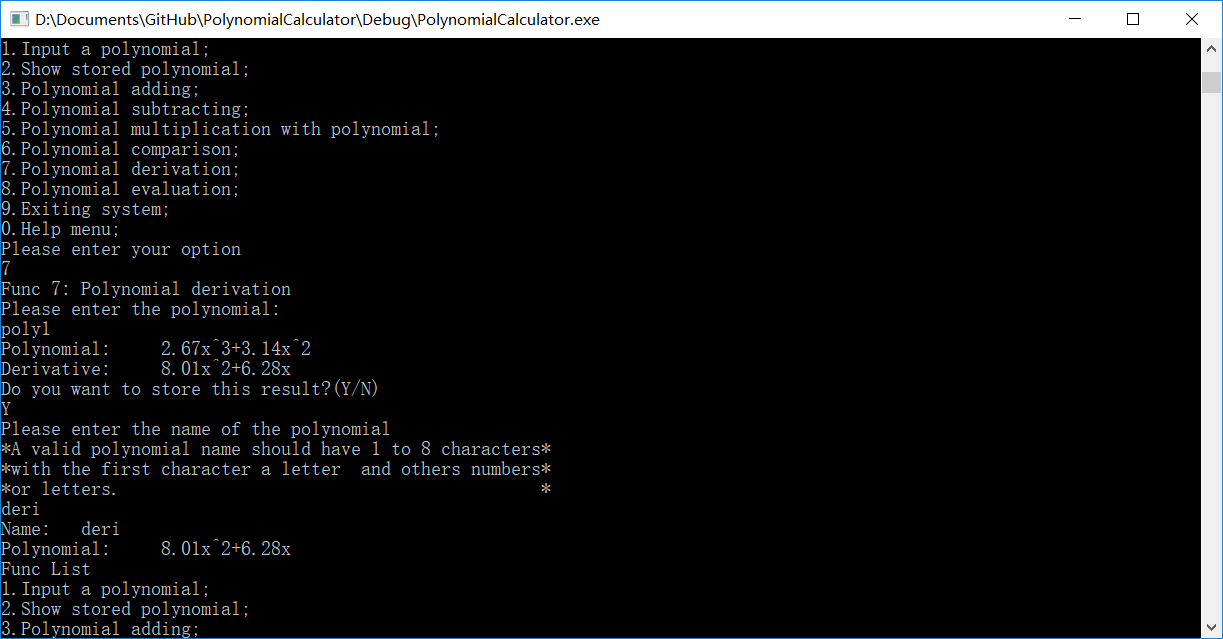


1. 多项式比较无误

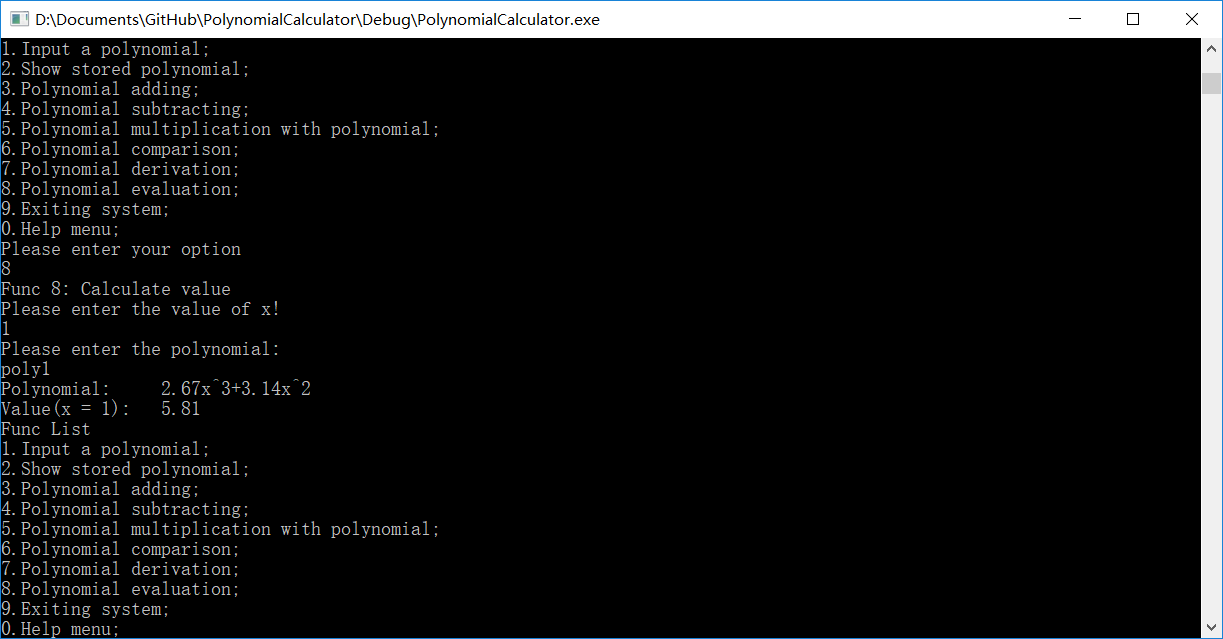




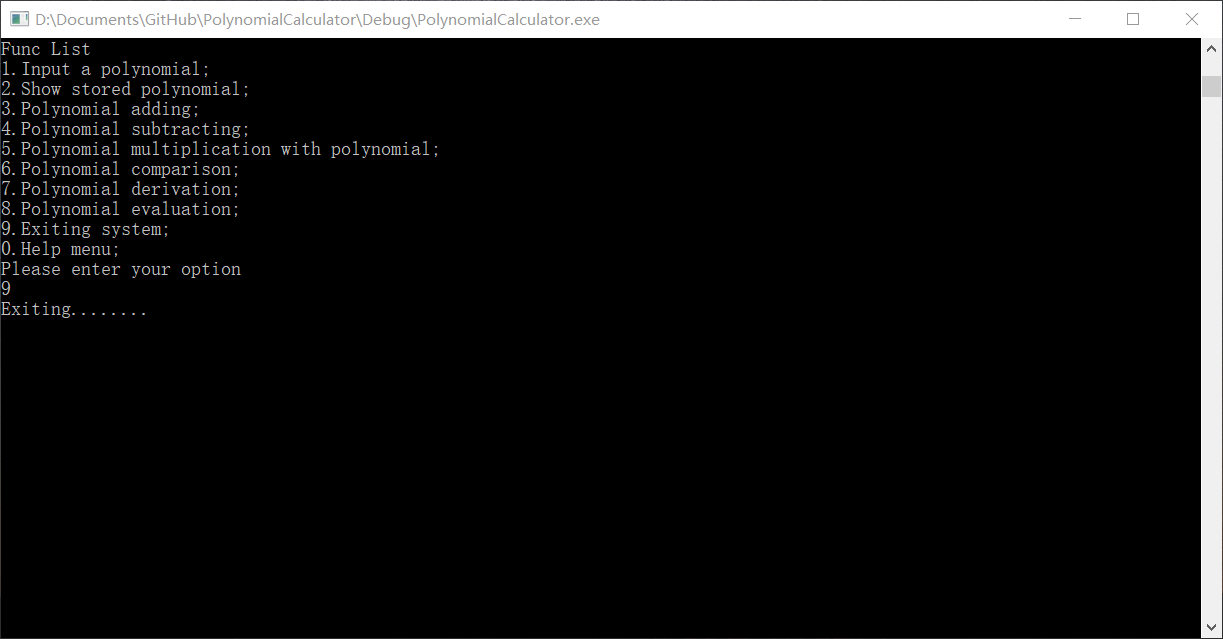
1. 多项式求导 无误



1. 多项式求值 无误



1. 退出 无误，正确退出



1. 设计心得

因为项目的需求是做一个多项式计算器，故很自然的将其分为两个类，一类作为接口和交互类，为计算器（Calculator）类，一类作为数据类，为多项式（Polynomial）类。 首先设计多项式类。

先设计多项式类的数据成员，由顾客的制定选择vector<pair<int, double> >类型，可根据多项式的项数自动分配大小。

设计多项式类的成员函数。发现若是多项式类单纯作为数据类，则有些单薄且遗留给计算器类的工作将会很多，会有头重脚轻之嫌，故决定将计算作为多项式类的成员函数，而让计算器类专注于与用户的交互和逻辑处理。因为部分运算可用运算符来自然表示，如加减乘除和比较赋值等，故对该部分函数采用运算符重载的设计，使对多项式类的使用更加自然。并且在设计其他的多项式计算器时，该Polynomial类可重用。

其次设计计算器类

先设计计算器类的数据成员。其实则是如何将多项式储存起来。因为多项式是以名字来进行区分的，采用 vector 或数组时，要加上搜索多项式的函数，想起之前老师讲的 map，可以以“键”来区分成员，若是以多项式名称的字符串作为键，多项式作为成员，则在储存和取出多项式时都较为方便。于是采用了 map<string, Polynomial>作为数据成员来储存多项式。

然后则是计算器类成员函数的设计。因为计算已经包含在多项式类中了，故计算器类成员函数设计的重点就在于与用户间的信息交互。其中我认为最为困难的在于从用户取得多项式输入并将之转换为 Polynomial 类，因为用户的输入既可能是单一的多项式（如（3.4，5）（2.5，1）（4，0）），也可能是带有名字的多项式（如 poly1=（3，4）（2.5,1）），故要对两种情况加以区分。在经过反复的思考和好几次的失败测试后，我找到了一种可行的方法：考虑在输入第一个左括号之前是否输入“=”即可。第一个左括号之后的内容则将之储存在一个字符串中由另一成员函数处理。因为输入若是合法，则这个字符串的内容相对有一定规律，识别相对较容易。设计中遇到的另一个问题则是输入时遗留在缓冲区的换行符的问题，这些换行符会在单个字符从缓冲区取得输入时造成一定影响。

故需要清空缓冲区。刚开始考虑使用cin.clear()和cin.sync()组合，但发现该方法在dev-c++和Visual Studio 2015 可用，但在Visual Studio 2017 和 Linux G++环境中无法正常清空缓冲区。经查资料发现，C++标准并未强制cin.sync()实现任何功能，故使用该函数是不可靠的。于是采用cin.clear()加cin.ignore()组合来清空缓冲区，并通过了测试，可正常清空缓冲区。

在项目的设计过程中，我遇到了如上的问题和挑战，虽然具有一定困难，但这也是对我能力的一种提高，经过这次项目设计，我对于 C++程序设计也加深了理解。