# 面向对象程序设计大程实验报告

小组成员：赵浚源 3120103425

胡亮泽 3120102116

(联系方式：18868103152)

王艺 3120101996

周天 3120101895

指导老师：陈琦

## 开发环境

本小组在这次的程序设计中以VS为开发环境，相比于过去的VC版本，能够更简单的使用一些语句，功能也更加齐全。另外，小组成员对于VS的使用环境比较熟悉，对程序的设计开发有很大的帮助。

## 概要设计

本小组这次程序设计的目标是实现一款数学计算工具软件系统。

其主要功能包括：

1、矩阵乘法的运算；

2、对n个数据的排序；

3、方程的求解；

4、任意曲线的积分运算；

**下面将介绍各个模块的设计思路，主要是对头文件的定义和某些函数的功能进行说明，具体的实现方式详见各成员负责部分文档的说明。**

1. 矩阵乘法的运算

该部分主要包括matrix.cpp文件，Matrix\_Mul.cpp文件及他们相应的头文件

头文件的定义：

Matrix.h

|  |
| --- |
| #ifndef \_Matrix\_H\_  #define \_Matrix\_H\_  #include <string>  #include <iostream>  class Matrix  {  int line;//行数  int row;//列数  double \*a;//存放元素  bool IsDes;//析构标志  //所有对元素的访问均从1起  public:  Matrix(int l,int r = 1);  Matrix();  Matrix(const Matrix& a);  ~Matrix();  bool setm(int l,int r);//设置矩阵大小  bool set(int i,int j,double n);//设置矩阵元素  double get(int i,int j)const;//获取矩阵元素  bool print()const;//打印矩阵  Matrix operator +(const Matrix& Matrix1)const;  Matrix operator -(const Matrix& Matrix1)const;  Matrix operator\*(const double d);//数乘  friend Matrix operator\*(const double d, const Matrix c);//数乘  //上面这一段是为了满足交换律  friend Matrix operator/(const Matrix c,const double d);//数除  Matrix operator \*(const Matrix& m)const;//矩阵乘  void operator =(const Matrix& m);//赋值运算  void ChangeLine(const int a,const int b);//交换行  void ChangeRow(const int a,const int b);//交换列  std::ostream& operator<< (std::ostream& stream)const;//重载输出流  std::istream& operator>> (std::istream& in);//重载输入流  friend std::ostream& operator<< (std::ostream& stream, const Matrix other);//重载输出流  friend std::istream& operator>> (std::istream& stream, Matrix other);//重载输入流，输入格式为先输入行数再输入列数，接下来输入矩阵  };  class NotMatch//不匹配错误类  {  public:  std::string error;  int al;//第一个矩阵的行数  int ar;//列数  int bl;//第二个矩阵行数  int br;//列数  NotMatch(std::string s, int a,int b,int c,int d)  {  error = s;  al = a;  ar = b;  bl = c;  br = d;  }  ~NotMatch(){};  };  class NotMatrix  {  public:  int line;  int row;  NotMatrix(int l, int r)  {  line = l;  row = r;  }  ~NotMatrix(){};  };  #endif |

Matrix\_Mul.h

|  |
| --- |
| #pragma once  #ifndef \_Matrix\_Mul\_h\_  #define \_Matrix\_Mul\_h\_  #include "Function.h"  #include "Class Matrix\matrix.h"  class Matrix\_Mul : public Function  {  Matrix m1;  Matrix m2;  public:  Matrix\_Mul() : m1(), m2(){};  ~Matrix\_Mul(){};  void function();  };  #endif //\_Matrix\_Mul\_h\_ |

头文件中的变量定义和函数作用已经在注释中展现了。下面分析设计和实现思路。

设计思路：

定义一个matrix的类，以及一个Matrix\_Mul的类，后者主要是关于矩阵的乘法的类。

实现思路：

对于一些操作符的重载，如乘法，加法等，不多做赘述。主要过程是通过调用setm函数先设置矩阵的大小，再通过调用get和set函数分别输入矩阵元素和赋值矩阵元素。其实实现思路比较简单，这个模块的难点关键在于实现相应具体函数的功能。

1. 对n个数据的排序

该模块主要包括sort.h和sort.cpp文件

首先是头文件的定义

|  |
| --- |
| #pragma once  #ifndef \_SORT\_H\_  #define \_SORT\_H\_  #include "Function.h"  class Sort : public Function  {  double \*m;  int mem;  void merge(double\*data, int a, int b, int length, int n);  public:  Sort(){};  Sort(double \*a, int n);  ~Sort();  void setmem(double \*a, int n);  void function();  };  #endif//\_SORT\_H\_ |

设计思路：定义一个Sort的类，并在输入一定数量的数字后利用归并排序的方法获得排序结果。

实现思路：在类Sort中，m指向一连串数字的起点，而mem代表数字的个数。调用function输入setmem函数需要的数字个数n和数字首地址a后，调用setmem函数对相应变量进行赋值，最后调用merge函数利用归并排序的方法对这些数字进行排序并且输出。

1. 方程的求解

本模块主要包括one\_y\_one\_, one\_y\_two\_, one\_y\_three\_, two\_y\_one\_, three\_y\_one\_的头文件和cpp文件。其主要目的分别是实现一元一次，一元二次，一元三次，二元一次，三元一次方程组的求解。

其实所有方程类型的实现和设计思路都是差不多的，只是具体的实现过程有比较大的区别，下面就以一元一次方程的实现思路作为例子：

头文件为：

|  |
| --- |
| #pragma once  #ifndef ONE\_Y\_ONE\_C\_H  #define ONE\_Y\_ONE\_C\_H  #include "Equation.h"  class one\_y\_one\_c : public Equation  {  private:  double a;  double b;  public:  one\_y\_one\_c():a(0),b(0){};  virtual ~one\_y\_one\_c(){};  one\_y\_one\_c(const one\_y\_one\_c& other){};  void getCoeff();  double\* solve();  };  #endif |

设计思路：给出一定的模板，在输入相应变量前面的系数后计算相应的结果。

实现思路：由于所有的方程的实现过程是有一定共同性的，所以所有的方程类型都有一个共同的积累Equation。而在所有的方程组类型中，都有共同的两个函数：

getCoeoff和solve，分别是用于输入相应的系数和解方程的，先后调用这两个函数就可以实现解方程的功能了。

1. 任意曲线的积分运算

本模块主要包括calculus.h和calc.cpp文件。

Calculus.h头文件定义如下：

|  |
| --- |
| #pragma once  #ifndef CALCULUS\_H\_  #define CALCULUS\_H\_  #include "Function.h"  struct stack{ //利用堆栈实现函数表达式运算优先级  char func[100]; //原函数表达式字符串  double output[50]; //用以存储输出的数字  char stack\_out[10]; //用以存储堆栈中的运算符  double Func\_Calc(const double& x);  };  class Calculus : public Function{  private:  const double high = 2000000000; //运算的上限  const double low = -2000000000; //运算的下限  const double distance = 0.0001; //离散点之间的x向距离  double x\_from, x\_to; //积分的x范围  stack pre; //堆栈  double Calc(double x); //计算某个x值下的函数值，即某个点的函数值  void GetFunction(); //读取函数表达式，需要输入，输入关于x的函数体，中间不能有空格等符号  void GetRange(); //读取积分范围  public:  Calculus(double from = 0, double to = 1000){  x\_from = from;  x\_to = to;  }  double GetResult();  void function();  };  #endif |

设计思路：头文件中的大多数定义已经有相应的注释了，不再赘述。主要涉及思路参照了matlab中积分的方法。即对于任意曲线，在曲线上取大量的点，并通过计算这些点的面积并相加得到曲线面积，所以本模块最后的计算结果实际上是一个估计值，误差大约在0.00002左右。

实现思路：

定义一个calculus的类和stack的结构。Calculus类中有包括积分的区域x\_from, x\_to等变量，主要是描述积分的时候的一些内容。还有一个关于堆栈的结构变量，堆栈结构存在的主要意义就是对输入的字符串函数体进行处理分析，包括优先级和计算等，其中的成员函数Func\_Calc就是用来计算某个点的横坐标为x时该点的纵坐标的。

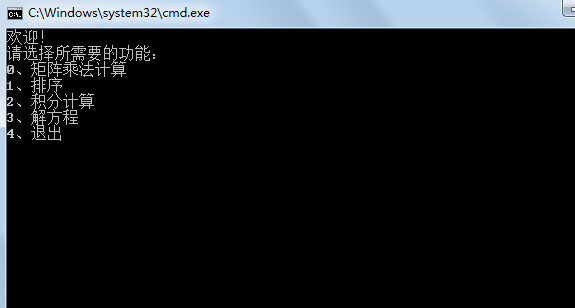
执行过程主要包括：

调用calculus类中的GetFunction函数，主要作用是得到函数体字符串，该字符串会通过pre变量的调用输入到pre中。

调用GetRange函数，主要作用是获得积分范围，可以反向积分，即x\_from的输入大于x\_to的值。

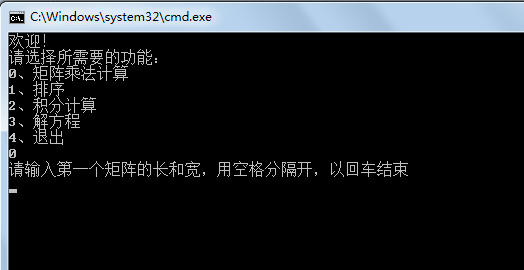
上面两个函数都在调用GetResult函数的时候自动调用，主要是获得积分结果，该函数通过不断循环调用Calc函数（主要功能为计算某个点的面积值），获得许多点的函数值并求和，获得最后的积分结果。

## 用户使用手册及系统操作截图

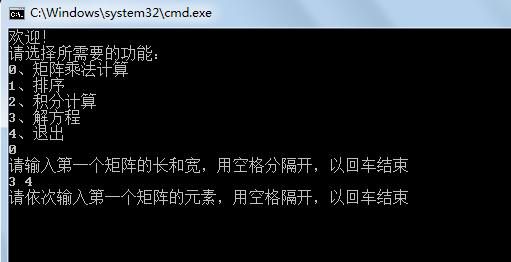
运行程序，进入功能选择界面，如图：

4种主要功能和退出对应的功能号都显示在界面中，输入对应的功能号就可以进行相应的操作了。

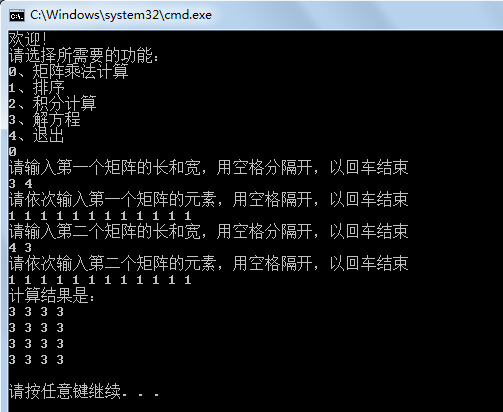
首先，输入0，进入矩阵乘法计算模块，得到以下结果：



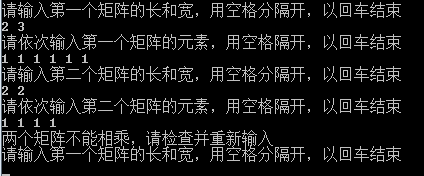
按照提示输入第一个矩阵的长和宽即可，得到：



在此输入第一个矩阵的元素，第二个矩阵的输入格式完全相同

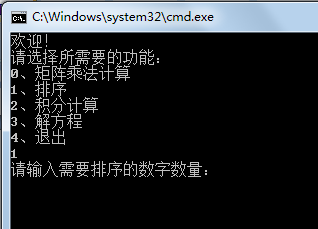


输入完成后，如果输入正确，则会输出正确的计算结果，否则，会输出：

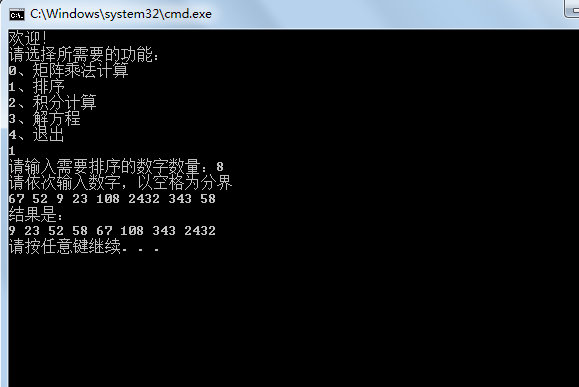


以上是矩阵乘法计算模块的使用方法，接下来是排序模块的使用：

同理，进入界面后，输入1进入排序功能：

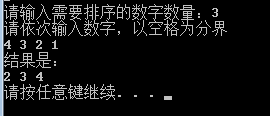


按照要求输入相应的数字数量



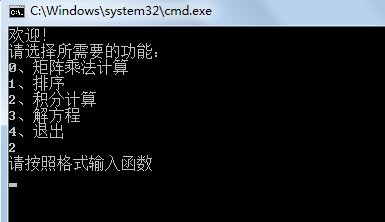
然后按照要求输入相应的数字后会输出相应的排序结果

如果输入的数字个数超过原本的数量，则超出部分的数字不会进入排序计算

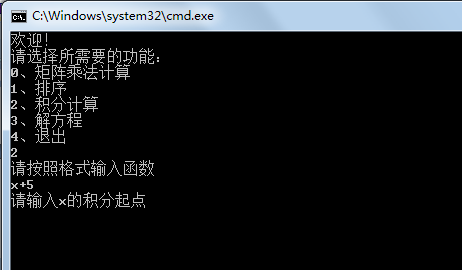


以上是排序功能，接下来是积分功能的模块：

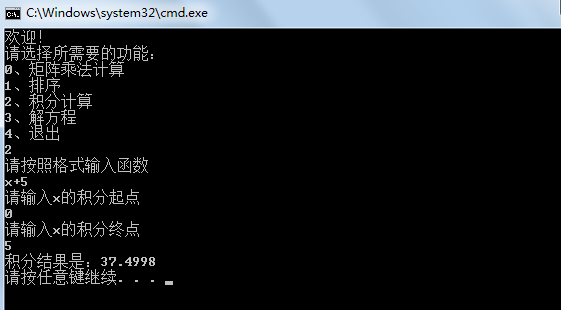
同理进入积分功能模块，出现



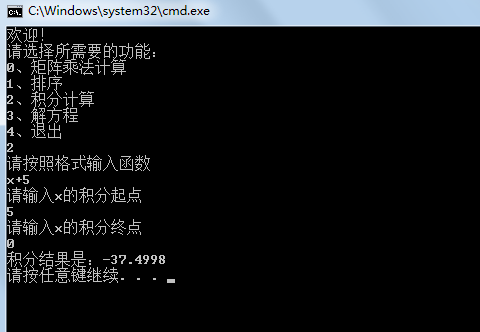
这里只需要输入函数体就可以了，比如输入x+5



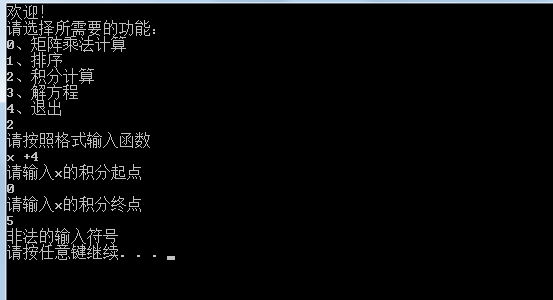
接下来按照提示输入积分范围，即可得到计算结果



当积分方向相反时，输出相反的积分结果

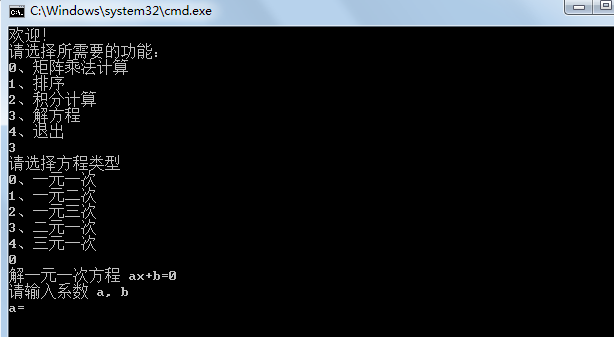


另外，输入函数中不允许出现空格等非法符号，除加(+)减(-)乘(\*)除(/)指数(^)符号，其中带负号必须加上括号，还有e和x以外，所有的符号都是非法符号，一旦输入

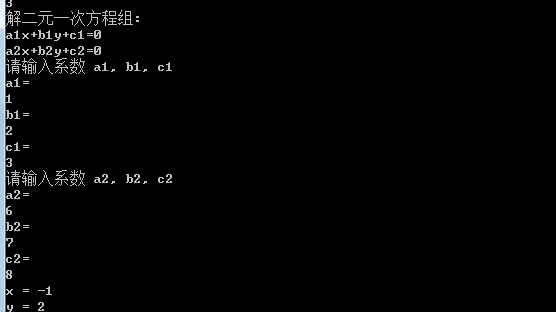


以上是积分模块，下面是解方程模块：

选择该功能后，可以选择相应的方程方式，如



按照提示输入相应的系数，就可以输出结果



以上就是所有功能的使用方法，谢谢！

## 具体分工

赵浚源：矩阵模块的设计、后期测试与整合、文档撰写

胡亮泽：积分模块的设计、后期测试、文档撰写

王艺：排序模块的设计、后期测试、文档撰写

周天：解方程模块的设计、后期测试、文档撰写