利用抽象向量类派生数学向量及应用

2024年11月2日

**1 利用抽象向量类派生数学向量项目概述**

**1.1 项目内容及要求**

1.用先前写好的抽象向量类作为基类派生出一个数学向量类

2.为数学向量类进行重载运算符即设计其它功能函数的编码

3.用数学向量类作为组合成员实现一个矩阵类，并对矩阵类进行一定的函数设计

4.用专门的测试程序测试类的运行情况

**1.2 研究人员及分工**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 学号 | 姓名 | 角色及具体贡献 |
| 1 | 23121942 | 刁文涵 | 其它项目 |
| 2 | 23120740 | 焦艺峰 | 程序设计、测试、完善与PPT制作与项目汇报 |
| 3 | 23120736 | 潘子扬 | 其它项目 |
| 4 | 23120771 | 刘威龙 | 矩阵类设计 |

**2 类的设计**

**2.1 MathVector类的设计**

**2.1.1数据成员设计**

**数据成员从基类继承：**

Alloc alloc; //内存管理器

    T \*data; //向量值资源

    size\_t size; //向量维度

    size\_t volume; //向量预留空间大小

**2.1.2 成员函数（友元函数）原型设计**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 函数名 | 作用 | |
| 构造函数、析构函数 | | | |
| 1 | MathVector():AbstractVector<T,Alloc>(){}; | 默认的构造函数，直接使用基类的构造函数 | |
| 2 | MathVector(size\_t n):AbstractVector<T,Alloc>() | 指定维度不赋值的构造函数 | |
| 3 | MathVector(size\_t n,const T &x):AbstractVector<T,Alloc>() | 指定维度并统一赋值的构造函数 | |
| 运算符重载 | | | |
| 4 | AbstractVector<T, Alloc>& operator+=(const AbstractVector<T, Alloc>& c)noexcept override | 重载+=，使得两个向量可以直接相加并把相加的值赋值给左值。 | |
| 5 | MathVector<T,Alloc> operator+(const MathVector<T, Alloc>& c) const noexcept | 重载+，使得两个向量可以直接相加 | |
| 6 | MathVector<T, Alloc>& operator-=(const MathVector<T, Alloc>& c)noexcept | 重载-，用于向量减法 | |
| 7 | MathVector<T, Alloc>& operator-=(const MathVector<T, Alloc>& c)noexcept | 重载-=，用于向量减法后直接赋值 | |
| 8 | MathVector<T,Alloc> operator-()const noexcept | 重载-，表示向量的负值 | |
| 9 | T operator\*(const MathVector<T, Alloc>& c)const noexcept | 重载\*，表示向量内积 | |
| 10 | MathVector<T,Alloc> operator\*(const T& c)const noexcept | 重载\*，表示向量数乘 | |
| 11 | MathVector<T, Alloc>& operator\*=(const T& c)noexcept | 重载\*=，表示向量数乘后赋值 | |
| 12 | MathVector<T,Alloc> operator^(const MathVector<T,Alloc>& c)const noexcept | 重载^,表示向量外积，只允许三维向量 | |
| 13 | MathVector<T,Alloc>& operator^=(const MathVector<T,Alloc>& c)noexcept | 重载^=,表示向量外积后赋值 | |
| 14 | bool operator==(const MathVector<T,Alloc>& c)const noexcept | 重载==，表示向量相等 | |
| 15 | bool operator!=(const MathVector<T,Alloc>& c)const noexcept | 重载!=,表示向量不相等 | |
| 其他功能函数 | | | |
| 16 | void resize(const size\_t& n=0) | | 重构向量，清空并重新设置大小 |
| 17 | void clearzero() | | 将向量值清零 |
| 18 | double modoules()const noexcept | | 计算向量模长 |
| 19 | double angle(const MathVector<T,Alloc>& c)const noexcept | | 计算向量夹角 |
| 20 | void Rand() | | 为向量随机赋值 |
| 21 | AbstractVector<T,Alloc>& plus\_equal(const AbstractVector<T, Alloc>& c) | | 实现带异常检查的+= |
| 22 | MathVector<T,Alloc> plus(const MathVector<T,Alloc>& c)const | | 实现带异常检查的向量加法 |
| 23 | MathVector<T,Alloc>& minus\_equal(const MathVector<T, Alloc>& c) | | 实现带异常检查的向量-= |
| 24 | MathVector<T,Alloc> minus(const MathVector<T,Alloc>& c)const | | 实现带异常检查的向量减法 |
| 25 | T dot(const MathVector<T,Alloc>& c)const | | 实现带异常检查的向量内积 |
| 26 | MathVector<T,Alloc> times(const MathVector<T,Alloc>& c)const | | 实现带异常检查的向量外积 |
| 27 | MathVector<T,Alloc>& times\_equal(const MathVector<T,Alloc>&c) | | 实现带异常检查的向量^= |
| 28 | double angle\_err(const MathVector<T,Alloc>& c)const | | 实现带异常检查的向量夹角 |
| 其它友元函数 | | | |
| 29 | inline MathVector<Ts> operator\*(const Ts &c, const MathVector<Ts> &v) | | 实现数字在左的向量数乘 |
| 30 | double angle(const MathVector<Ts> &a, const MathVector<Ts> &b)noexcept | | 实现外部函数式计算量向量夹角 |
| 31 | double angle\_err(const MathVector<Ts> &a, const MathVector<Ts> &b) | | 实现外部函数式计算量向量夹角并做异常检查 |

**2.2 Matrix类的设计**

**2.2.1数据成员设计**

int m=0;//矩阵行数（行向量个数）

int n=0;//矩阵列数（行向量分量个数）

MathVector<T>\* a= nullptr;//矩阵中向量的资源(指针)

**2.2.2成员函数（友元函数）原型设计**

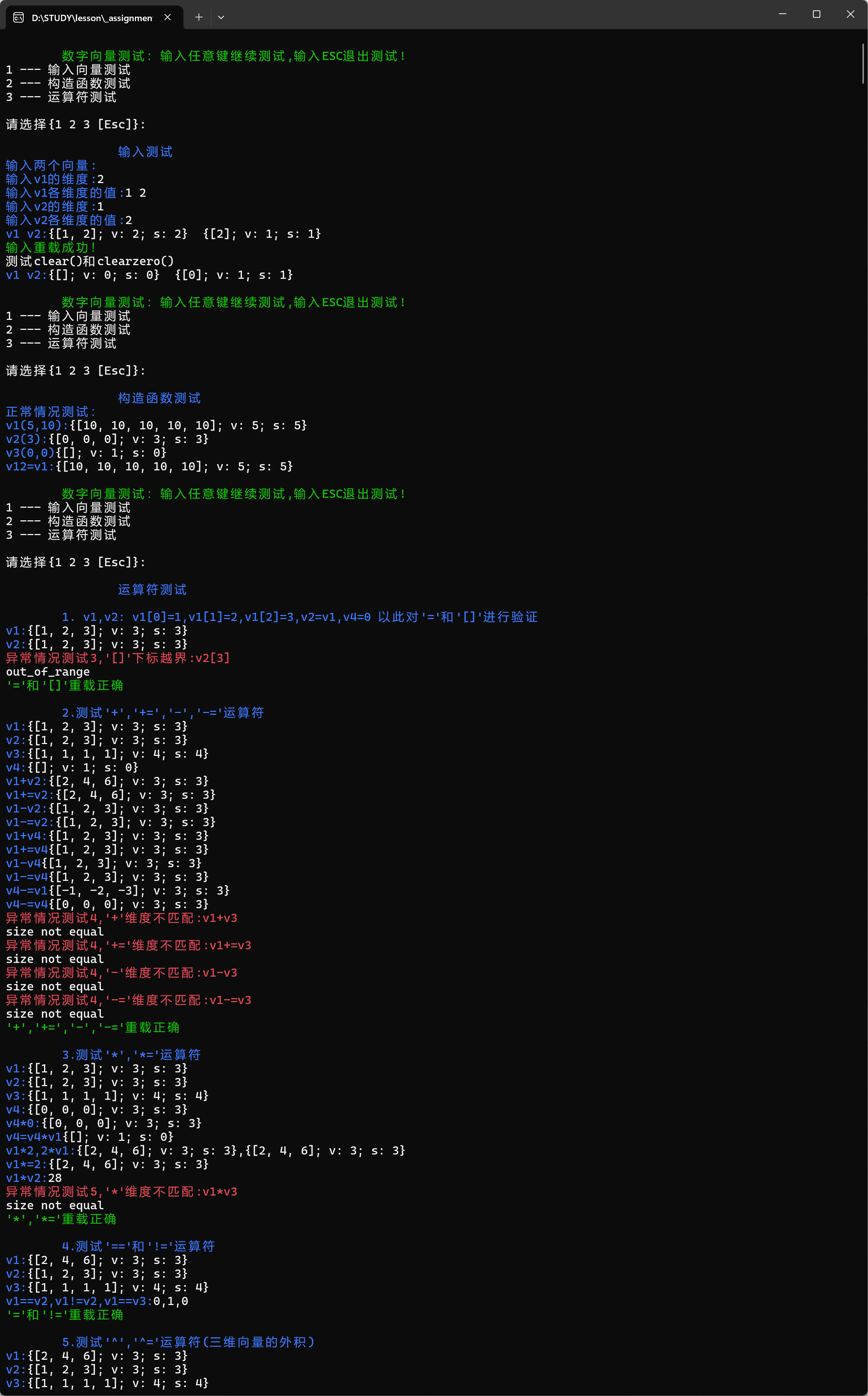
|  |  |  |
| --- | --- | --- |
| 序号 | 函数名 | 作用 |
| 构造函数、析构函数 | | |
| 1 | Matrix(); | 默认的构造函数 |
| 2 | Matrix(const int& m, const int& n); | 有参构造函数 |
| 3 | Matrix(const Matrix& copy); | 拷贝构造函数 |
| 4 | Matrix(const MathVector<T>& vec) | 用向量构造的特殊拷贝构造函数 |
| 5 | ~Matrix() | 析构函数 |
| 运算符重载 | | |
| 6 | MathVector<T>& operator[](int i)const | 重载[]，表示某一行向量 |
| 7 | Matrix operator+(const Matrix& b)const | 重载+,表示矩阵加法 |
| 8 | Matrix operator-(const Matrix& b)const; | 重载-，表示矩阵减法 |
| 9 | Matrix operator-()const; | 重载-，表示负矩阵 |
| 10 | Matrix operator\*(const Matrix& b)const; | 重载\*，表示矩阵乘法 |
| 11 | template<class Tb> Matrix<T> operator\*(const Tb& b)const | 重载\*，实现矩阵数乘 |
| 12 | Matrix<T>& operator=(const Matrix<T>& copy) | 重载=，进行矩阵赋值 |
| 13 | template<class T> std::ostream& operator<<(std::ostream& out, const Matrix<T>& b) | 重载矩阵输出 |
| 功能类函数 | | |
| 14 | MathVector<T>\* begin()const | 返回矩阵首地址 |
| 15 | void Rand(); | 实现矩阵随机化 |
| 16 | Matrix<T> transpose()const | 实现矩阵转置 |
| 17 | bool judge\_zero\_matrix()const; | 判断矩阵是否为0矩阵 |
| 18 | bool judge\_square\_matrix()const; | 判断矩阵是否为方阵 |
| 19 | T determinant(MathVector<T>\* a, int n)const; | 求方阵行列式 |
| 20 | void adjugate\_matrix(MathVector<T>\* a, Matrix<T>& adjugate\_matrix\_ptr, int n); | 求矩阵伴随矩阵 |
| 21 | Matrix<double> inverse\_matrix(); | 求逆矩阵 |

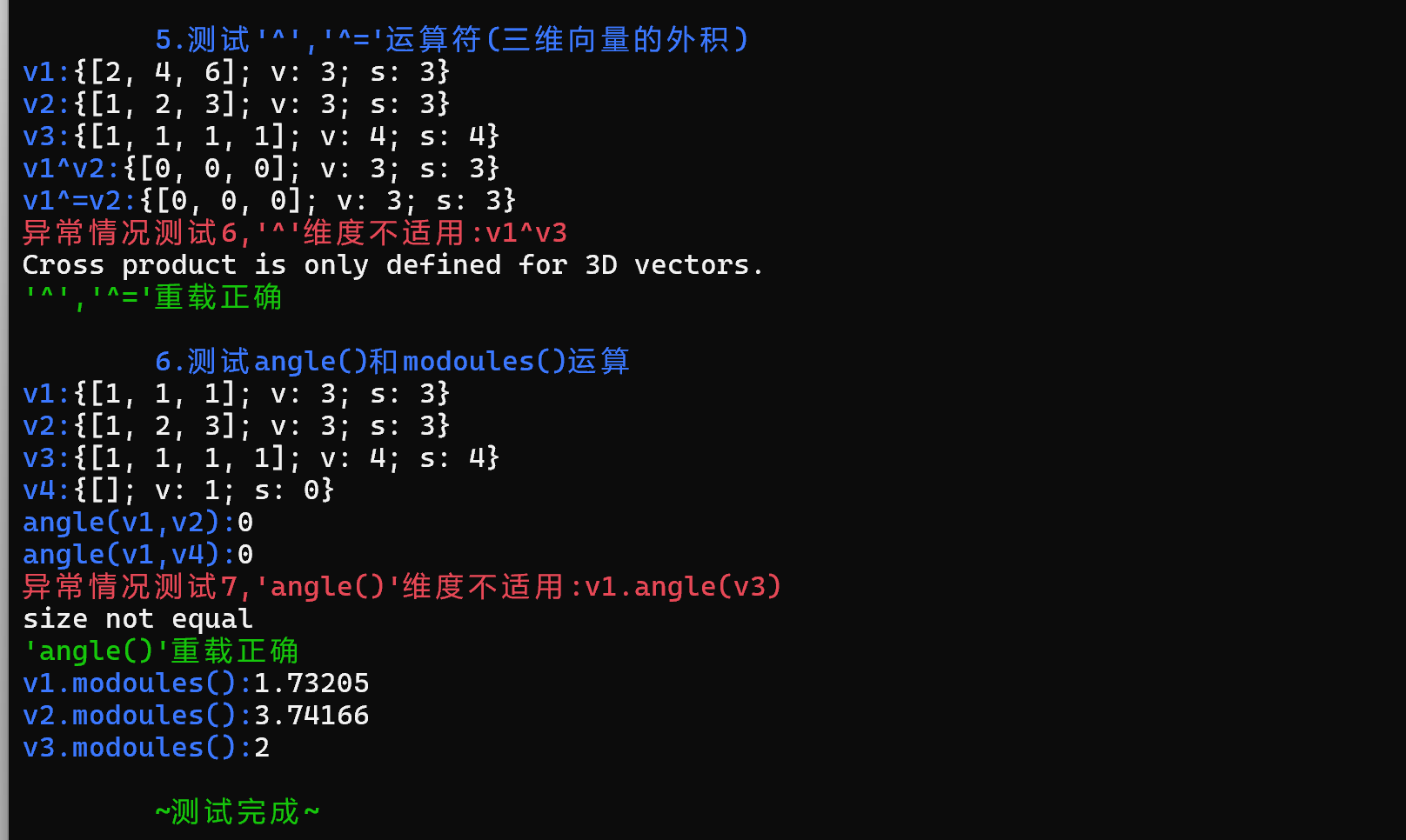
**3 测试情况**

**3.1 测试样例设计**

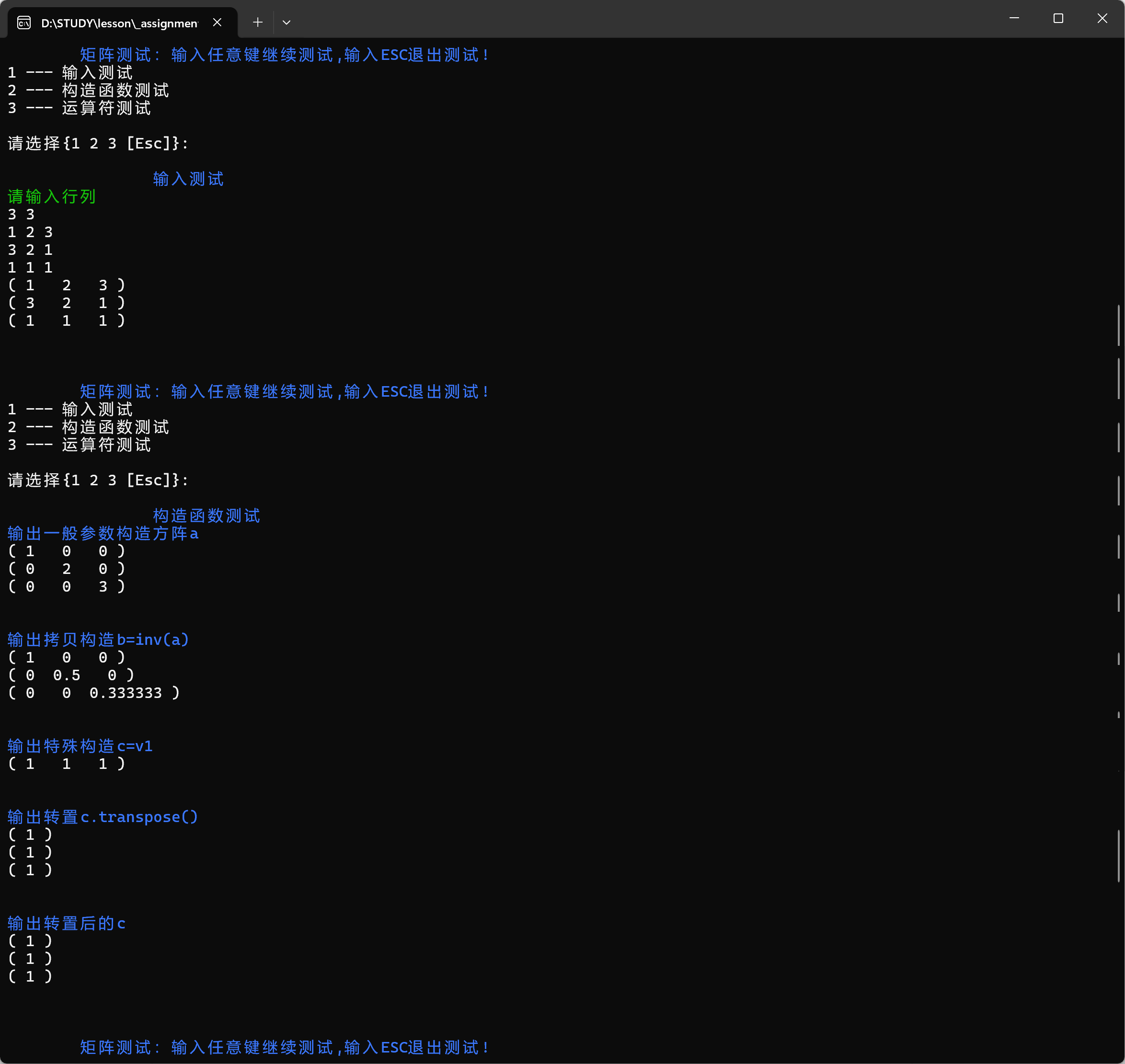
**3.1.1 基本功能测试**

写了专门的测试函数进行各项函数功能测试，如下图，分别进行了输入输出，构造和运算符重载及其它功能的实现测试。

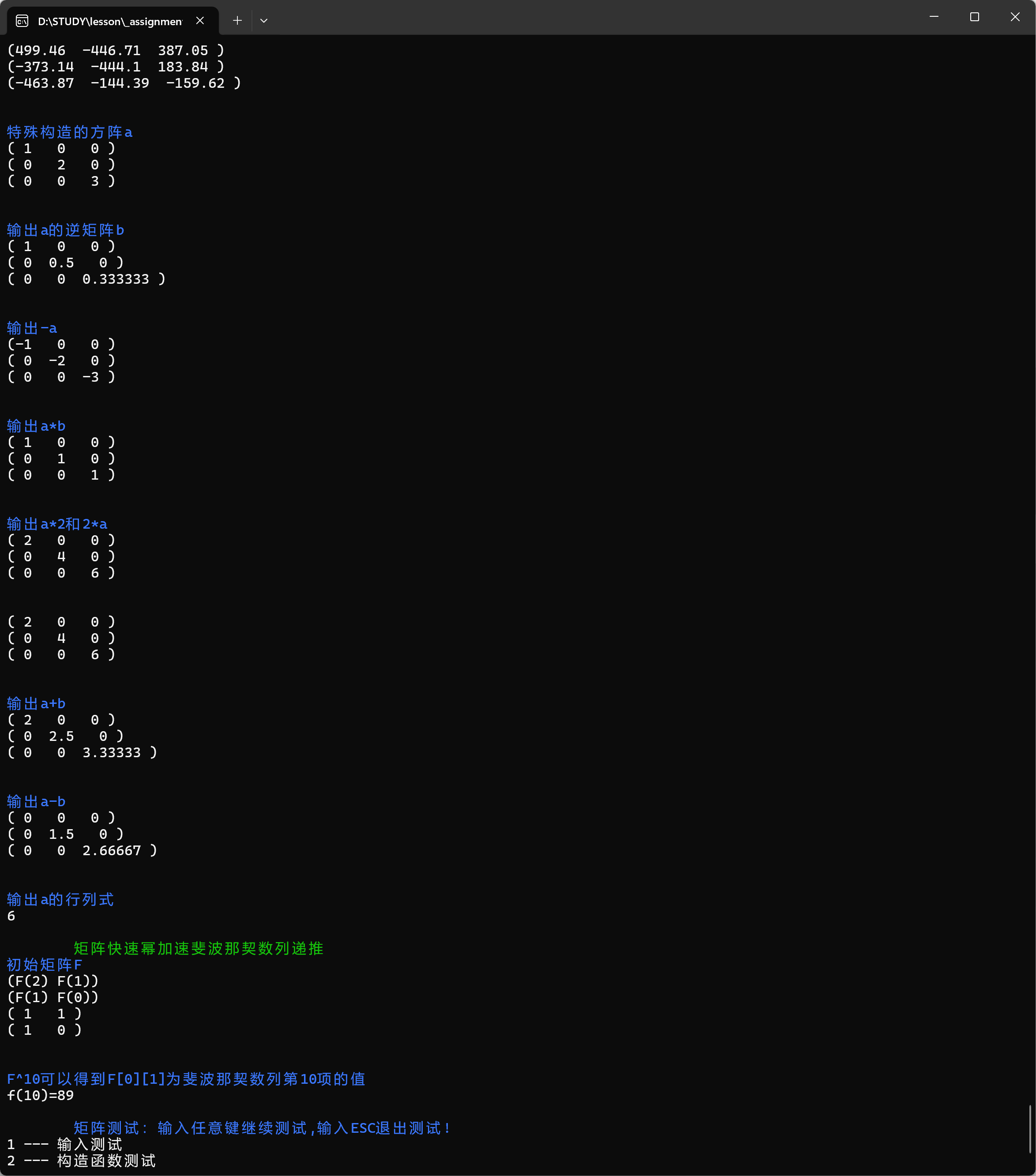




对于矩阵类，除了进行基本的测试，还设计了一个矩阵快速幂加速地推的小应用场景。（一下为输入输出和构造测试结果）



以下为运算符重载测试(第一个为随机化的方阵)：



**3.2 测试结果对程序的改进情况**

本程序的重点实现一个功能完善的数学向量，应用数学向量实现矩阵，此后使用矩阵再进行各项运算功能。从改进上说，它还可以进一步实现更多更复杂的数学运算。