《面向对象程序设计实验》任务书

# 实验五 面向对象的矩阵运算编程

**一、实验目的与要求**

1．掌握类模板、模板实例化等面向对象概念；

2．理解运算符在模板中定义和调用方法；

3．理解与掌握移动语义编程技巧；

4．掌握防止内存泄漏的安全编程方法。

**二、实验内容**

矩阵MAT是行列定长的二维数组。常见的矩阵运算包括矩阵的加、减、乘、转置和赋值等运算。

请对矩阵MAT类中的所有函数成员编程，并对随后给出的main( )函数进行扩展，以便完成矩阵及其重载的所有运算符的测试。

输出矩阵元素时整数用”%6ld” 或”%6lld”打印，浮点数用”%8f”或”%8lf”打印，最后一行用换行符结束”\n”。

**至少要测试两种实例类MAT<int>和MAT<long long>。**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iomanip>

#include <exception>

#include <typeinfo>

#include <string.h>

using namespace std;

template <typename T>

class MAT {

T\* const e; //指向所有整型矩阵元素的指针

const int r, c; //矩阵的行r和列c大小

public:

MAT(int r, int c); //矩阵定义

MAT(const MAT& a); //深拷贝构造

MAT(MAT&& a)noexcept; //移动构造

virtual ~MAT()noexcept;

virtual T\* const operator[ ](int r); //取矩阵r行的第一个元素地址，r越界抛异常

virtual MAT operator+(const MAT& a)const; //矩阵加法，不能加抛异常

virtual MAT operator-(const MAT& a)const; //矩阵减法，不能减抛异常

virtual MAT operator\*(const MAT& a)const; //矩阵乘法，不能乘抛异常

virtual MAT operator~()const; //矩阵转置

virtual MAT& operator=(const MAT& a); //深拷贝赋值运算

virtual MAT& operator=(MAT&& a)noexcept; //移动赋值运算

virtual MAT& operator+=(const MAT& a); //“+=”运算

virtual MAT& operator-=(const MAT& a); //“-=”运算

virtual MAT& operator\*=(const MAT& a); //“\*=”运算

virtual char\* print(char\* s)const noexcept; //打印时列用空格隔开，行用回车结束

};

int main(int argc, char\* argv[ ]) //请扩展main()测试其它运算

{

MAT<int> a(1, 2), b(2, 2), c(1, 2);

char t[2048];

a[0][0] = 1; //类似地初始化矩阵的所有元素

a[0][1] = 2; //等价于“\*(a.operator[ ](0)+1)=2;”即等价于“\*(a[0]+1)=2;”

a.print(t); //初始化矩阵后输出该矩阵

b[0][0] = 3; b[0][1] = 4; //调用T\* const operator[ ](int r)初始化数组元素

b[1][0] = 5; b[1][1] = 6;

b.print(t);

c = a \* b; //测试矩阵乘法运算

c.print(t);

(a + c).print(t); //测试矩阵加法运算

c = c - a; //测试矩阵减法运算

c.print(t);

c += a; //测试矩阵“+=”运算

c.print(t);

c = ~a; //测试矩阵转置运算

c.print(t);

return 0;

}