МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Липецкий Государственный Технический Университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа

по программированию №2

“Конструкторы и деструктор. Перегрузка операторов”

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Станиславчук С. М.

(подпись, дата)

Группа АС-21-1

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_ Ведищев В. В.

(подпись, дата)

Липецк 2022 г.

Содержание

2. Задание

3. Цель

4. Код программы

5. Вывод

2. Задание

Реализовать на языке C++ класс, который обладает возможностью выполнения соответствующих действий. Реализованный класс должен содержать необходимые конструкторы, деструктор и перегруженные операторы.

Вариант 11:   
Класс: матрица целых чисел. Умножение матрицы на вещественное число и присвоение результата третьей переменной – матрице осуществляется одной строкой C=A\*b, где b – целое число.

3. Цель работы.

Изучить виды конструкторов, принципы использования конструкторов, назначение деструктора, способы перегрузки операторов.

4. Код программы.

#include <iostream>

using namespace std;

class Matrix

{

private:

size\_t rows\_, cols\_;

int\*\* arr;

void copy(Matrix& M)

{

size\_t t = rows\_; rows\_ = M.rows\_; M.rows\_ = t;

t = cols\_; cols\_ = M.cols\_; M.cols\_ = t;

int\*\* p = arr; arr = M.arr; M.arr = p;

}

public:

Matrix(const size\_t rows = 0, const size\_t cols = 0)

{

rows\_ = rows; cols\_ = cols; arr = nullptr;

if (rows\_ && cols\_)

{

arr = new int\* [rows\_];

for (size\_t i = 0; i < rows\_; i++)

arr[i] = new int[cols\_];

}

}

Matrix(const Matrix& M)

{

rows\_ = M.rows\_; cols\_ = M.cols\_; arr = nullptr;

if (rows\_ && cols\_)

{

arr = new int\* [rows\_];

for (size\_t i = 0; i < rows\_; i++)

{

arr[i] = new int[cols\_];

for (size\_t j = 0; j < cols\_; ++j)

arr[i][j] = M.arr[i][j];

}

}

}

Matrix& operator = (const Matrix& M)

{

Matrix tmp(M);

copy(tmp);

return \*this;

}

~Matrix()

{

if (arr != nullptr)

{

for (size\_t i = 0; i < rows\_; i++)

delete arr[i];

delete[] arr;

}

}

size\_t rows() const

{

return rows\_;

};

size\_t cols() const

{

return cols\_;

};

int\* operator[](size\_t i) { return arr[i]; }

const int\* operator[](size\_t i) const { return arr[i]; }

};

void Print(Matrix& M) {

for (size\_t i = 0; i < M.rows(); i++)

{

for (size\_t j = 0; j < M.cols(); j++)

cout << M[i][j] << ' ';

cout << endl;

}

}

void Input(Matrix& M) {

for (size\_t i = 0; i < M.rows(); i++)

{

for (size\_t j = 0; j < M.cols(); j++)

cin >> M[i][j];

}

}

Matrix operator \* (const Matrix& M, int m)

{

Matrix temp(M.rows(), M.cols());

for (size\_t i = 0; i < M.rows(); i++)

{

for (size\_t j = 0; j < M.cols(); j++)

{

temp[i][j] = M[i][j] \* m;

}

}

return temp;

}

int main()

{

int m = 2;

Matrix A(3, 3), B;

Input(A); Input(B);

cout << "A = " << endl;

Print(A);

cout << "A\*m = B" << endl;

B = A \* m;

Print(B);

cout << endl;

cout << "A = " << endl;

Print(A);

return 0;

}

5. Вывод.

Изучил виды конструкторов, принципы использования конструкторов, назначение деструктора, способы перегрузки операторов.