

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №9
по курсу: «Языки и методы программирования»
«Перегрузка операций»

Выполнил:
Студент группы ИУ9-21Б

Проверил:
Посевин Д. П.

Москва, 2024

1. Цель

Данная работа предназначена для изучения возможностей языка C++, обеспечивающих применение знаков операций к объектам пользовательских типов.

2. Персональный вариант

Matrix<T, N> – антидиагональная матрица размера $N \times N$ с элементами типа T. (Все элементы антидиагональной матрицы, кроме лежащих на диагонали, идущей от нижнего левого угла до верхнего правого угла, равны нулю. Матрица должна быть представлена только числами, лежащими на диагонали) Операции:

1. «+» – сумма двух матриц;
2. «*» – произведение двух матриц;
3. «[]» – получение значения элемента, расположенного на i -той строке в j -том столбце.

3. Решение

3.1. Код

```
#include <cassert>
#include <iostream>
#include <type_traits>
#include <vector>

using namespace std;

template <typename T, int N, typename = typename
↳ std::enable_if_t<std::is_arithmetic_v<T>, T>>
class Matrix {
    private:
        vector<T> m;

    public:
        Matrix() : m(vector<T>(N * N)) {}

        Matrix(vector<T> diag) : m(vector<T>(N * N)) {
            assert(diag.size() == N);

            for (int i = 0; i < N; ++i)
                m[N * (N - 1 - i) + i] = diag[i];
        }
}
```

```

auto size() { return N; }

auto &operator()(int i, int j) {
    assert(0 <= i && i < N);
    assert(0 <= j && j < N);
    return m[i * N + j];
}

friend auto &operator<<(ostream &output, const Matrix<T, N>
↪ &m) {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j)
            output << m.m[i * N + j] << ' ';
        output << '\n';
    }
    return output;
}

friend auto operator*(Matrix<T, N> a, Matrix<T, N> b) {
    Matrix<T, N> result;

    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            for (int t = 0; t < N; ++t)
                result(i, j) += a(i, t) * b(t, j);

    return result;
}

friend auto operator+(Matrix<T, N> a, Matrix<T, N> b) {
    Matrix<T, N> result;

    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            result(i, j) += a(i, j) + b(i, j);

    return result;
}

auto operator+=(Matrix<T, N> &other) { return (*this =
↪ (*this) + other); }
auto operator*=(Matrix<T, N> &other) { return (*this =
↪ (*this) * other); }
};

```

```
int main() {  
    Matrix<int, 5> m({1, 2, 3, 4, 5});  
    Matrix<int, 5> m1({5, 4, 3, 2, 1});  
  
    cout << m << '\n';  
    cout << m1 << '\n';  
  
    auto res = (m1 * m) + (m * m1);  
  
    cout << res << '\n';  
}
```

Код 1: main.cc

3.2. Скриншоты

```
0 0 0 0 5
0 0 0 4 0
0 0 3 0 0
0 2 0 0 0
1 0 0 0 0

0 0 0 0 1
0 0 0 2 0
0 0 3 0 0
0 4 0 0 0
5 0 0 0 0

26 0 0 0 0
0 20 0 0 0
0 0 18 0 0
0 0 0 20 0
0 0 0 0 26
```

Рис. 1: Пример работы программы