Задание для Лабораторной работы 2

Написать компьютерную программу, содержащую

- Описание классов vect и matr, содержащие поля int dim, double*b (double*a), int num, static int count;

- Конструкторы и деструктор, содержащие вывод сообщений о выполненном действии и номерах участвующих объектов;

- Набор оператор-функций (компонентных и внешних) для операций векторной алгебры, содержащих вывод сообщений о выполненных действиях и номерах участвующих объектов:

  v+v, v-v, -v, v*v, k*v, v=v,

  m+m, m-m, -m, m*m, k*m, m=m,

  m*v.

- Функцию main, содержащую сценарий работы с векторами и матрицами.

Представить результаты в виде двух файлов:

- Компьютерная программа на C++;

- Отчет о выполнении лабораторной работы с описанием алгоритма и структуры программы.

```
#include <iostream>
using namespace std;

// Класс Вектор
class vect {
private:
   int dim;       // Размерность
   double* b;     // Указатель на данные
   int num;       // Уникальный номер объекта
   static int count; // Счётчик объектов

public:
   // Конструктор с размерностью
   vect(int d) : dim(d), num(++count) {
      b = new double[dim]{};
      cout << "Vect #" << num << " created (dim=" << dim << ")" << endl;
   }

   // Конструктор копирования
   vect(const vect& other) : dim(other.dim), num(++count) {
      b = new double[dim];
      for (int i = 0; i < dim; ++i) b[i] = other.b[i];
```

```
    cout << "Vect #" << num << " copied from #" << other.num << endl;
}

// Деструктор
~vect() {
    delete[] b;
    cout << "Vect #" << num << " destroyed" << endl;
}

// Оператор присваивания
vect& operator=(const vect& other) {
    if (this != &other) {
        delete[] b;
        dim = other.dim;
        b = new double[dim];
        for (int i = 0; i < dim; ++i) b[i] = other.b[i];
    }
    cout << "Vect #" << num << " = Vect #" << other.num << endl;
    return *this;
}

// Операторы для векторов
vect operator+(const vect& other) const {
    cout << "Vect #" << num << " + Vect #" << other.num << endl;
    vect res(dim);
    for (int i = 0; i < dim; ++i) res.b[i] = b[i] + other.b[i];
    return res;
}

vect operator-(const vect& other) const {
    cout << "Vect #" << num << " - Vect #" << other.num << endl;
    vect res(dim);
    for (int i = 0; i < dim; ++i) res.b[i] = b[i] - other.b[i];
    return res;
}

vect operator-() const {
    cout << "-Vect #" << num << endl;
    vect res(dim);
    for (int i = 0; i < dim; ++i) res.b[i] = -b[i];
    return res;
}

double operator*(const vect& other) const { // Скалярное произведение
    cout << "Vect #" << num << " * Vect #" << other.num << endl;
    double sum = 0;
    for (int i = 0; i < dim; ++i) sum += b[i] * other.b[i];
    return sum;
}

friend vect operator*(double k, const vect& v); // Умножение на скаляр
```

```
};

// Умножение скаляра на вектор (внешняя функция)
vect operator*(double k, const vect& v) {
   cout << k << " * Vect #" << v.num << endl;
   vect res(v.dim);
   for (int i = 0; i < v.dim; ++i) res.b[i] = k * v.b[i];
   return res;
}

// Инициализация статической переменной
int vect::count = 0;

// Класс Матрица (аналогично вектору, упрощённая версия)
class matr {
private:
   int dim;
   double* a;
   int num;
   static int count;

public:
   matr(int d) : dim(d), num(++count) {
      a = new double[dim*dim]{};
      cout << "Matr #" << num << " created (dim=" << dim << ")" << endl;
   }

   ~matr() {
      delete[] a;
      cout << "Matr #" << num << " destroyed" << endl;
   }

   // Другие операторы и методы аналогично вектору...
};

int matr::count = 0;

// Пример main
int main() {
   vect v1(3); // Создаём вектор размерности 3
   vect v2(3);
   vect v3 = v1 + v2; // Сложение векторов
   double dot = v1 * v2; // Скалярное произведение
   vect v4 = 2.5 * v1; // Умножение на скаляр

   matr m1(2); // Создаём матрицу 2x2
   return 0;
}
```