Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления» (ИУ)
КАФЕДРА «Информационная безопасность» (ИУ8)

Домашнее задание №2

ПО КУРСУ

«Алгоритмические языки»

ТЕМА: «Объектно-ориентированные возможности языка Си++»

Студент ИУ8-21 (Группа) Г. А. Карев (И. О. Фамилия)

Преподаватель: В. В. Соборова (И.О. Фамилия)

Задание:

Общие требования

Каждый класс должен поддерживать возможность чтения данных объекта из файла и получение данных с помощью генераторов псевдослучайных чисел (можно использовать конструкторы или методы для инициализации полей).

Примечание 1. В случае большого размера исходных данных рекомендуется получать их с помощью генератора псевдослучайных чисел.

Примечание 2. В случае, когда требуется добавить элемент к массиву, матрице и т.п., например, при перегрузке операции ++, и значение нового элемента не заданы, для получения значения использовать генератор псевдослучайных чисел.

Вариант 12

Определить классы CVects для работы с массивом векторов на плоскости и CAngls для работы с массивом углов между векторами. Длина массива задается в конструкторе класса.

В классе должны быть определены необходимые конструкторы (в том числе конструктор копирования и перемещения), деструктор, операторы присваивания (копированием и перемещением), << (вставка в поток вывода), инкремент и декремент ++ и -- (справа и слева), увеличивающие и уменьшающие длину массива, операторы сложения (CVects и CAngls, возвращающий CVects, а также CAngls и CVects, возвращающий CVects), вычитания (CVects из CVects, возвращающий CAngls). При сложении CVects и CAngls или наоборот получаем новый CVects, путем поворота векторов на углы, положительное направление против часовой стрелки.

При сложении и вычитании длина результата - это минимум из длин исходных массивов, исходные массивы при этом не меняются.

Каждый класс должен быть описан в 2-х файлах (заголовочном и реализации). В отдельном файле должен быть написан тест на данный класс (функция main).

Код:

CVects.h

```
#pragma once
#include <iostream>
#include <cmath>

class CVects {

    struct Vector {
        double x;
        double y;
    };
    Vector* data;
    int size;

public:

    CVects(int n = 0);
    CVects(const CVects& other);
    CVects(CVects&& other) noexcept;

    ~CVects();

    CVects& operator=(const CVects& other); //Копирование
    CVects& operator=(CVects&& other) noexcept; //Перемещение

    CVects& operator++();
    CVects operator++(int);
    CVects& operator--();
    CVects operator--(int);

    friend std::ostream& operator<<(std::ostream& os, const CVects& v);

    friend CVects operator+(const CVects& v, const class CAngls& a);
    friend CVects operator+(const class CAngls& a, const CVects& v);
    friend class CAngls operator-(const CVects& v1, const CVects& v2);

    void readFromFile(std::istream& is);

    int getSize() const { return size; }
    const Vector* getData() const { return data; }
};
```

CVects.cpp

```cpp
#include "CVects.h"
#include "CAngls.h"
#include <cstdlib>
#include <ctime>
#include <algorithm>

CVects::CVects(int n) : size(n) {
    data = new Vector[n];
    for (int i = 0; i < n; ++i) {
        data[i] = { 0.0, 0.0 };
    }
}

CVects::CVects(const CVects& other) : size(other.size) {
    data = new Vector[size];
    for (int i = 0; i < size; ++i) {
        data[i] = other.data[i];
    }
}

CVects::CVects(CVects&& other) noexcept : data(other.data), size(other.size) {
    other.data = nullptr;
    other.size = 0;
}

CVects::~CVects() {
    delete[] data;
}

CVects& CVects::operator=(const CVects& other) {
    if (this != &other) {
        delete[] data;
        size = other.size;
        data = new Vector[size];
        for (int i = 0; i < size; ++i) {
            data[i] = other.data[i];
        }
    }
    return *this;
}

CVects& CVects::operator=(CVects&& other) noexcept {
    if (this != &other) {
        delete[] data;
        data = other.data;
        size = other.size;
        other.data = nullptr;
        other.size = 0;
    }
    return *this;
}

CVects& CVects::operator++() {
    Vector* newData = new Vector[size + 1];
    for (int i = 0; i < size; ++i) {
        newData[i] = data[i];
    }
    newData[size] = { static_cast<double>(rand() % 100), static_cast<double>(rand()
% 100) };
    delete[] data;
    data = newData;
    ++size;
    return *this;
```

```
}

CVects CVects::operator++(int) {
    CVects temp(*this);
    ++(*this);
    return temp;
}

CVects& CVects::operator--() {
    if (size > 0) {
        Vector* newData = new Vector[size - 1];
        for (int i = 0; i < size - 1; ++i) {
            newData[i] = data[i];
        }
        delete[] data;
        data = newData;
        --size;
    }
    return *this;
}

CVects CVects::operator--(int) {
    CVects temp(*this);
    --(*this);
    return temp;
}

std::ostream& operator<<(std::ostream& os, const CVects& v) {
    os << "Вектора (" << v.size << "):\n";
    for (int i = 0; i < v.size; ++i) {
        os << "  [" << i << "]: (" << v.data[i].x << ", " << v.data[i].y << ")\n";
    }
    return os;
}

void CVects::readFromFile(std::istream& is) {
    if (!is) {
        std::cerr << "Ошибка ввода строки" << std::endl;
        return;
    }

    for (int i = 0; i < size; ++i) {
        if (!(is >> data[i].x >> data[i].y)) {
            std::cerr << "Ошибка элемента " << i << std::endl;
            //Дописывает случайное значение в новое место
            data[i].x = static_cast<double>(rand() % 100);
            data[i].y = static_cast<double>(rand() % 100);
        }
    }
}
```

CAngls.h

```h
#pragma once
#include <iostream>
#include "CVects.h"

class CAngls {

    double* data;
    int size;

public:

    CAngls(int n = 0);
    CAngls(const CAngls& other);
    CAngls(CAngls&& other) noexcept;

    ~CAngls();

    CAngls& operator=(const CAngls& other); //Копирование
    CAngls& operator=(CAngls&& other) noexcept; //Премещение

    CAngls& operator++();
    CAngls operator++(int);
    CAngls& operator--();
    CAngls operator--(int);

    friend std::ostream& operator<<(std::ostream& os, const CAngls& a);

    friend class CVects operator+(const class CVects& v, const CAngls& a);
    friend class CVects operator+(const CAngls& a, const class CVects& v);
    friend CAngls operator-(const class CVects& v1, const class CVects& v2);

    void readFromFile(std::istream& is);

    int getSize() const { return size; }
    const double* getData() const { return data; }
};
```

CAngls.cpp

```cpp
#include "CAngls.h"
#include "CVects.h"
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <algorithm>

CAngls::CAngls(int n) : size(n) {
    data = new double[n];
    for (int i = 0; i < n; ++i) {
        data[i] = 0.0;
    }
}

CAngls::CAngls(const CAngls& other) : size(other.size) {
    data = new double[size];
    for (int i = 0; i < size; ++i) {
        data[i] = other.data[i];
    }
}

CAngls::CAngls(CAngls&& other) noexcept : data(other.data), size(other.size) {
    other.data = nullptr;
    other.size = 0;
}

CAngls::~CAngls() {
    delete[] data;
}

CAngls& CAngls::operator=(const CAngls& other) {
    if (this != &other) {
        delete[] data;
        size = other.size;
        data = new double[size];
        for (int i = 0; i < size; ++i) {
            data[i] = other.data[i];
        }
    }
    return *this;
}

CAngls& CAngls::operator=(CAngls&& other) noexcept {
    if (this != &other) {
        delete[] data;
        data = other.data;
        size = other.size;
        other.data = nullptr;
        other.size = 0;
    }
    return *this;
}

CAngls& CAngls::operator++() {
    double* newData = new double[size + 1];
    for (int i = 0; i < size; ++i) {
        newData[i] = data[i];
    }
    newData[size] = static_cast<double>(rand() % 360);
    delete[] data;
    data = newData;
    ++size;
    return *this;
```

```
}

CAngls CAngls::operator++(int) {
    CAngls temp(*this);
    ++(*this);
    return temp;
}

CAngls& CAngls::operator--() {
    if (size > 0) {
        double* newData = new double[size - 1];
        for (int i = 0; i < size - 1; ++i) {
            newData[i] = data[i];
        }
        delete[] data;
        data = newData;
        --size;
    }
    return *this;
}

CAngls CAngls::operator--(int) {
    CAngls temp(*this);
    --(*this);
    return temp;
}

std::ostream& operator<<(std::ostream& os, const CAngls& a) {
    os << "Углы (" << a.size << "):\n";
    for (int i = 0; i < a.size; ++i) {
        os << "  [" << i << "]: " << a.data[i] << " градусов\n";
    }
    return os;
}

CVects operator+(const CVects& v, const CAngls& a) {
    int minSize = std::min(v.getSize(), a.getSize());
    CVects result(minSize);

    const CVects::Vector* vData = v.getData();
    const double* aData = a.getData();

    for (int i = 0; i < minSize; ++i) {
        double angle = aData[i] * 3.14 / 180.0; //Переводим в радианы
        double cosA = cos(angle);
        double sinA = sin(angle);

        //Поворот векторов против часовой стрелик
        result.data[i].x = vData[i].x * cosA - vData[i].y * sinA;
        result.data[i].y = vData[i].x * sinA + vData[i].y * cosA;
    }

    return result;
}

CVects operator+(const CAngls& a, const CVects& v) {
    return v + a;
}


CAngls operator-(const CVects& v1, const CVects& v2) {
    int minSize = std::min(v1.getSize(), v2.getSize());
    CAngls result(minSize);

    const CVects::Vector* v1Data = v1.getData();
```

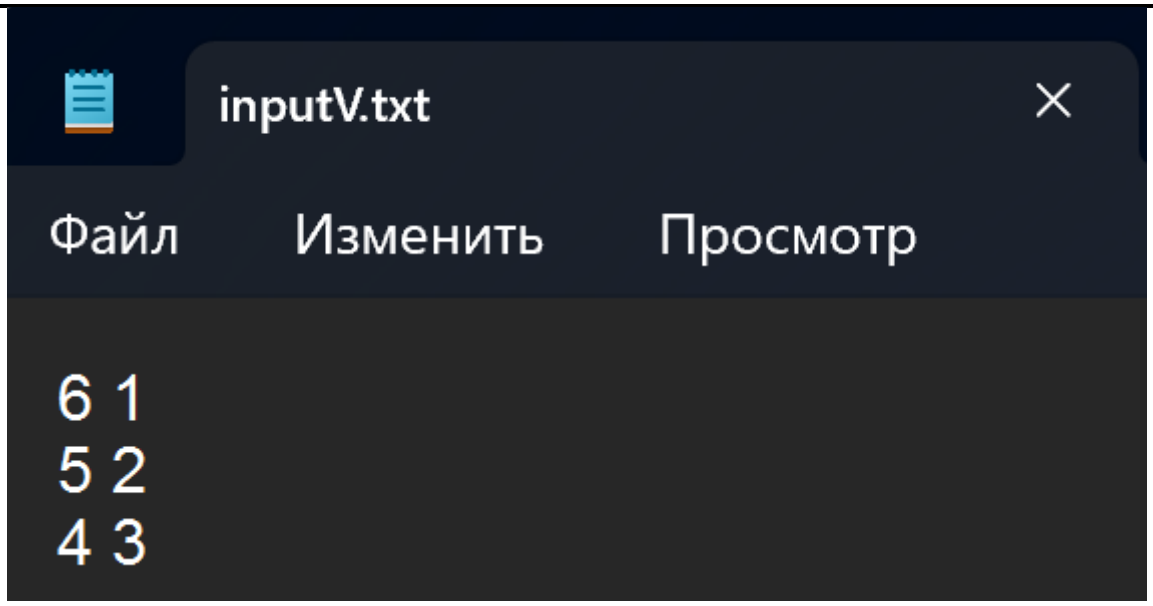
```
    const CVects::Vector* v2Data = v2.getData();

    for (int i = 0; i < minSize; ++i) {
        double dot = v1Data[i].x * v2Data[i].x + v1Data[i].y * v2Data[i].y;
        double det = v1Data[i].x * v2Data[i].y - v1Data[i].y * v2Data[i].x;
        result.data[i] = atan2(det, dot) * 180.0 / 3.14;
    }

    return result;
}

void CAngls::readFromFile(std::istream& is) {
    if (!is) {
        std::cerr << "Ошибка ввода" << std::endl;
        return;
    }

    for (int i = 0; i < size; ++i) {
        if (!(is >> data[i])) {
            std::cerr << "Ошибка чтения элемента " << i << std::endl;
            data[i] = static_cast<double>(rand() % 360);
        }
    }
}
```

dz.cpp

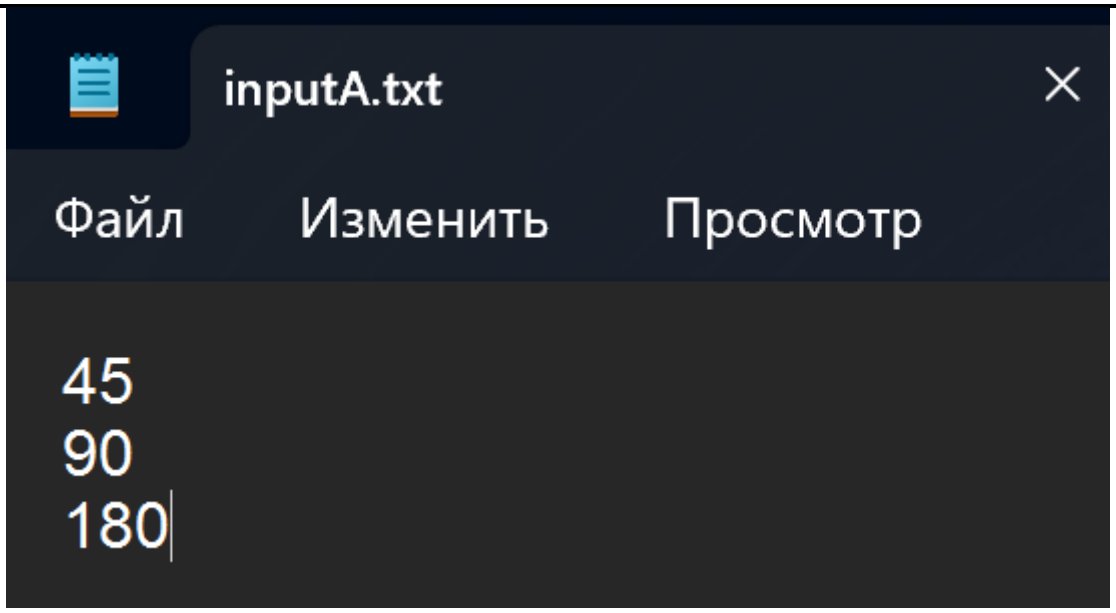
```cpp
#include <iostream>
#include <fstream>
#include "CVects.h"
#include "CAngls.h"

int main() {

    setlocale(LC_ALL, "RUS");
    srand(static_cast<unsigned>(time(nullptr)));

    std::ifstream vectsFile("inputV.txt");
    if (!vectsFile) {
        std::cerr << "Ошибка файла для векторов" << std::endl;
        return 1;
    }

    CVects v1(3);
    v1.readFromFile(vectsFile);
    std::cout << "Вектора v1 (из файла):\n" << v1 << std::endl;
    vectsFile.close();

    ++v1;
    std::cout << "Увеличение кол-ва векторов: \n" << v1 << std::endl;

    v1--;
    std::cout << "Уменьшение кол-ва векторов: \n" << v1 << std::endl;

    std::ifstream anglsFile("inputA.txt");
    if (!anglsFile) {
        std::cerr << "Ошибка файла для углов" << std::endl;
        return 1;
    }

     CAngls a1(3);
     a1.readFromFile(anglsFile);
     std::cout << "Углы v1 (из файла):\n" << a1 << std::endl;
     anglsFile.close();

     CVects vmoved = v1 + a1;
     std::cout << "Вращение векторов (CVects + CAngls): \n" << vmoved << std::endl;

     CVects vmoved1 = a1 + v1;
     std::cout << "Вращение векторов (CAngls + CVects): \n" << vmoved1 << std::endl;

     return 0;
}
```

Файлы для ввода данных:

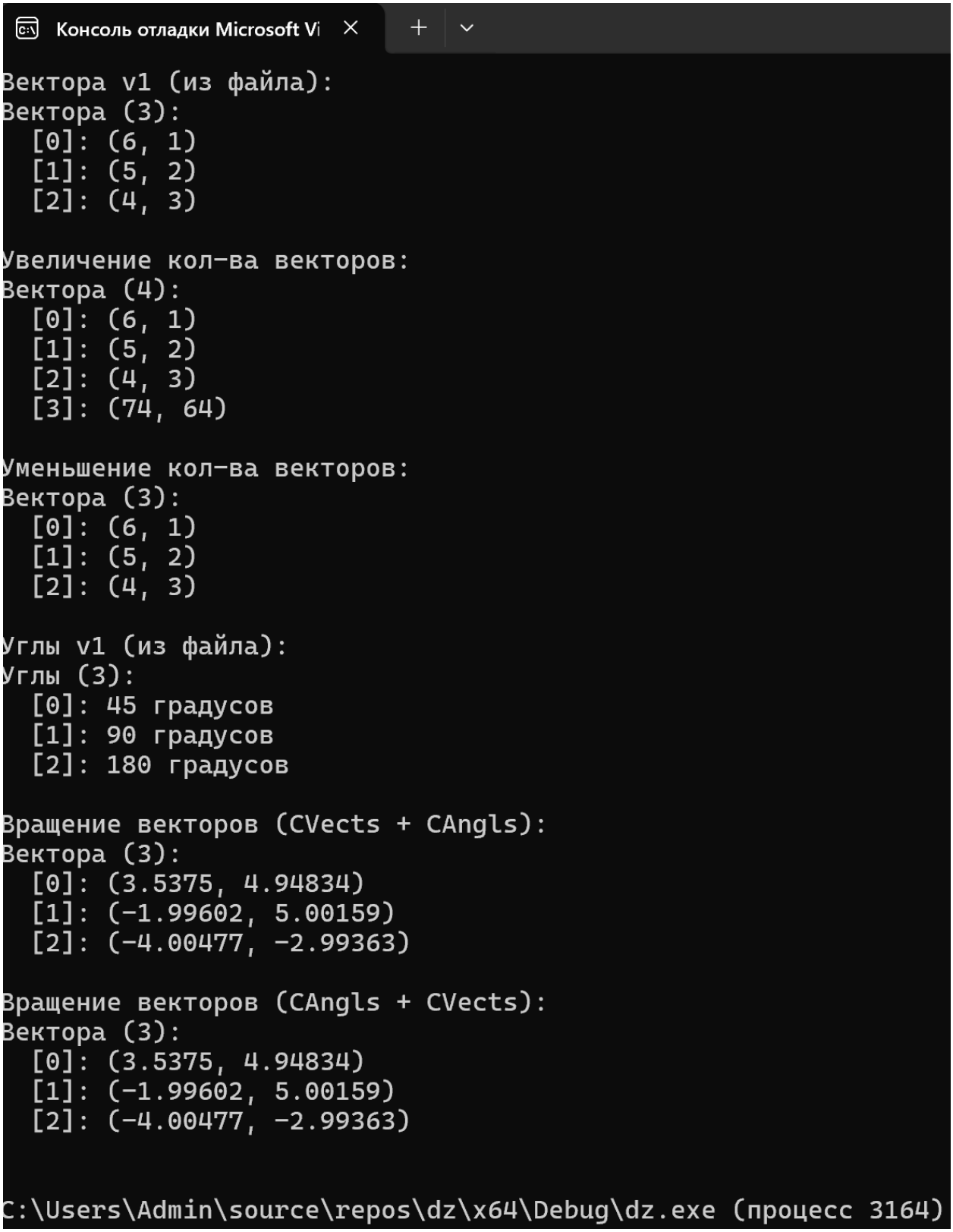
inputV.txt



inputA.txt



Вывод программы:



Вывод: Выполнил задание и написал работоспособную программу.