



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 и системы управления» (ИУ)  
КАФЕДРА «Информационная безопасность» (ИУ8)

Лабораторная работа № 2  
ПО КУРСУ  
«Алгоритмические языки»  
на тему «Изучение перегрузки стандартных  
операций в языке Си++»

Студент

ИУ8-25  
(Группа)

В.В.Гоза  
(И. О. Фамилия)

Преподаватель:

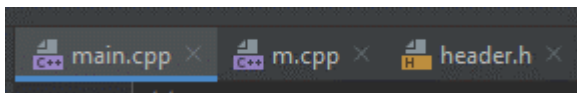
В. В. Соборова  
(И.О. Фамилия)

2022 г.

## Условие:

Описание операции перегруженной операции	Тип элемента вектора (массива)	Типы операндов и результата для перегруженной операции			№ варианта
		Первый операнд	Второй операнд	Результат	
+ сложение векторов одинаковой размерности, на выходе вектор такой же размерности элемент которого равен сумме соответствующих элементов двух векторов	double	Vector	Vector	Vector	1
		Vector	double *	Vector	2
		double *	Vector	Vector	3

## Программа:



### main.cpp:

```
//Вариант 3
#include "header.h"

int main() {
    std::ifstream fin("input.txt");
    std::ofstream fout("output.txt");
    std::string str;
    int n = 0;
    fin >> n;
    double *mas = new double[n];
    fin.ignore();
    std::getline(fin, str);
    fout << str << ":" << std::endl;
    for (int i = 0; i < n; ++i) {
        fin >> mas[i];
        fout << mas[i] << " ";
    }
    fout << std::endl;
    double *inpt = new double[n];
    fin.ignore();
    str = "";
    getline(fin, str);
    fout << str << ":" << std::endl;
    for (int i = 0; i < n; ++i) {
        fin >> inpt[i];
        fout << inpt[i] << " ";
    }
    fout << std::endl;
    Vector vec_1(mas, n);
    Vector sum = inpt + vec_1;
    fout << "Vector 3: " << std::endl;
    sum.print(fout);
    return 0;
}
```

## m.cpp:

```
//  
// Created by goza- on 08.03.2022.  
//  
#include <iostream>  
#include <vector>  
#include <fstream>  
#include <string>  
#include <regex>  
#include "header.h"  
  
Vector::Vector() : p(nullptr), n(0) {}  
  
Vector::Vector(int n) : n(n) {  
    p = new double[n];  
}  
  
Vector::Vector(double *p, int n) {  
    this->n = n;  
    this->p = new double[n];  
    for (int i = 0; i < n; ++i) {  
        this->p[i] = p[i];  
    }  
}  
  
Vector::Vector(const Vector &V) {  
    n = V.n;  
    p = new double[n];  
    for (int i = 0; i < n; ++i) {  
        p[i] = V.p[i];  
    }  
}  
  
Vector::Vector (Vector&& V) {  
    std::swap (p, V.p);  
    std::swap (n, V.n);  
}  
  
void Vector::print(std::ostream &out) {  
    for (int i; i < n; ++i) {  
        out << p[i] << " ";  
    }  
}  
  
double &Vector::operator[](int index) {  
    return p[index];  
}  
  
Vector &Vector::operator=(Vector & inpt) {  
    if (this != &inpt) {  
        n = inpt.n;  
        if (p != nullptr) {  
            delete[] p;  
        }  
        p = new double[n];  
        for (int i = 0; i < n; i++) p[i] = inpt.p[i];  
    }  
    return *this;  
}  
  
Vector &Vector::operator=(Vector && inpt) {  
    if (this != &inpt) {
```

```

        std::swap(p, inpt.p);
        std::swap(n, inpt.n);
    }
    return *this;
}

Vector::~Vector() {
    if (p != nullptr) {
        delete[] p;
    }
}

std::ifstream &operator>>(std::ifstream &fin, Vector &ob) {
    fin >> ob.n;
    for (size_t i = 0; i < ob.n; ++i) {
        fin >> ob.p[i];
    }
    return fin;
}

std::ostream &operator<<(std::ostream &out, const Vector &ob) {
    for (size_t i = 0; i < ob.n; ++i) {
        out << ob.p[i] << " ";
    }
    out << std::endl;
    return out;
}

Vector operator+(double *inpt, Vector &V) {
    Vector res(V.n);
    for (size_t i = 0; i < V.n; ++i) {
        res.p[i] = inpt[i] + V.p[i];
    }
    return res;
}

```

## header.hpp:

```

//
// Created by goza- on 05.03.2022.
//

#ifndef LABA2_HEADER_H
#define LABA2_HEADER_H

#pragma once
#include <iostream>
#include <fstream>
#include <string>
#include <vector>

using namespace std;

class Vector {
    double *p = nullptr;
    int n = 0;
public:
    Vector ();
    Vector (int n);
    Vector (double* p, int n);

```

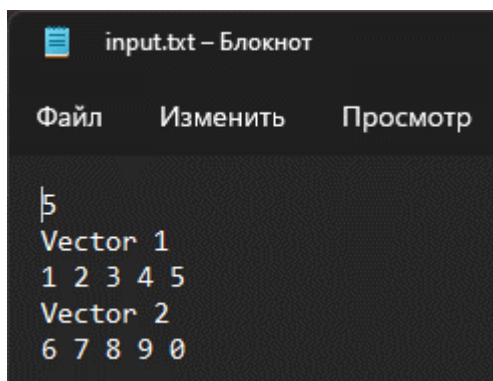
```

Vector (const Vector& V);
Vector (Vector&& V);
void print (std::ostream& out);
double& operator[] (int index);
Vector& operator = (Vector& inpt);
Vector& operator = (Vector&& inpt);
~Vector();
friend Vector operator+(double *inpt, Vector& V);
friend std::ostream& operator << (std::ostream& out, const Vector& ob);
friend std::ifstream& operator >> (std::ifstream& fin, Vector& ob);
};

#endif //LABA2_HEADER_H

```

**Файл input.txt, из которого читаются данные:**



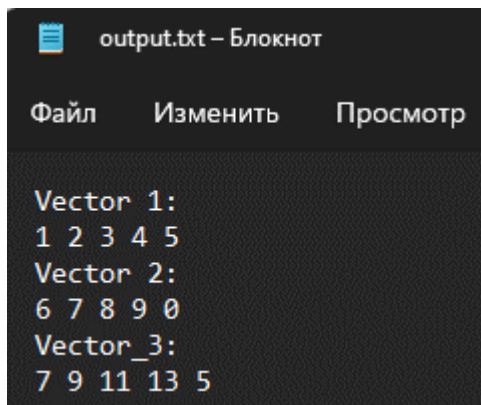
```

input.txt - Блокнот
Файл  Изменить  Просмотр

Vector 1
1 2 3 4 5
Vector 2
6 7 8 9 0

```

**Вывод программы в файл output.txt:**



```

output.txt - Блокнот
Файл  Изменить  Просмотр

Vector 1:
1 2 3 4 5
Vector 2:
6 7 8 9 0
Vector_3:
7 9 11 13 5

```