|  |  |
| --- | --- |
|  | **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 ИУ «Информатика и системы управления»

**КАФЕДРА ИУ8 «Компьютерная безопасность*»***

**ЛАБОРАТОРНАЯ РАБОТА №2**

**Название лабораторной работы: Перегрузка операторов класса.**

**по курсу: Алгоритмические языки.**

**Группа ИУ8 – 24**

**Студент: Полонский Сергей**

**Преподаватель: Барыкин Дмитрий**

**Дата: 06.07.2024**

ОТЧЕТ:

1) Цель работы : научится применять перегрузку операторов класса на практике.

2) УСЛОВИЕ ЗАДАЧИ: Дан класс(например, с именем Vector), задающий вектор размерности n.Поля класса : указатель на массив, задающий вектор(тип элемента double), массив должен создаваться динамически; число элементов(размерность) вектора(тип int).Класс включает : конструктор без параметров, задающий пустой вектор(число элементов равно 0); конструктор, создающий объект вектор на основе обычного одномерного массива размерности n; конструктор копирования, конструктор перемещения, деструктор.

Необходимо перегрузить операции и продемонстрировать их работу.Перегрузить операцию[](обращение к элементу вектора по индексу), операцию = (копирование вектора или создание копии вектора), операцию\* (умножение числа на вектора), на выходе вектор такой же размерности, каждый элемент которого равен произведению соответствующего элемента исходного вектора на число.

3) Текст программы с коментариями:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Vector {

friend void operator <<(ostream& m, const Vector& f) {

for (int i = 0; i < f.n; ++i) m << f.vec[i] << " ";

} // перегрузка оператора <<, который выполняет вывод элементов вектора в поток

friend void operator >>(istream& m, Vector& f) {

for (int i = 0; i < f.n; ++i) m >> f.vec[i];

} // перегрузка оператора >>, который выполняет ввод элементов вектора из потока

friend Vector& operator ^(bool\* a, const Vector& b) {

bool\* mas = new bool[b.n];

Vector\* c = new Vector(mas, b.n);

for (int i = 0; i < b.n; ++i) c->vec[i] = bool(a[i] \* b.vec[i]);

return \*c;

} // ^ логическая операция (исключающая ИЛИ) с двумя векторами одинаковой размерности, на выходе вектор такой же размерности элемент, которого равен логическая операции ^ соответствующих элементов двух векторов

public:

Vector(); // конструктор

Vector(bool\* mas, int k); // конструктор создающий вектор из заданного массива

Vector(const Vector& a); // конструктор копирования

Vector(Vector&& V); // конструктор перемещения, меняет местами 2 вектора

~Vector(); // деструктор

bool operator[](int a); // оператор получения элемента вектора по заданному номеру

Vector& operator=(const Vector& b); // оператор копирования одного вектора в другой

Vector& operator=(Vector&& b); // оператор перемещение из одного вектора в другой

//Vector& operator\*(int k); // оператор умножение вектора на число

Vector(const string& a);

private:

bool\* vec = nullptr;

int n = 0;

};

Vector::Vector() {

vec = new bool [0];

n = 0;

}

Vector::Vector(bool\* mas, int k) {

this->vec = new bool[k];

for (int i = 0; i < k; ++i) this->vec[i] = mas[i];

this->n = k;

}

Vector::Vector(const Vector& a) {

this->vec = new bool[a.n];

this->n = a.n;

for (int i = 0; i < a.n; ++i) this->vec[i] = a.vec[i];

}

Vector::Vector(Vector&& a) { // правая ссылка для отличия от предыдущего конструктора

swap(vec, a.vec);

swap(n, a.n);

}

Vector::~Vector() {

delete vec;

}

bool Vector::operator[](int a) {

if (a < n) { return vec[a]; }

else cout << "Error";

}

Vector& Vector::operator=(const Vector& b) {

n = b.n;

delete vec;

vec = new bool[b.n];

for (int i = 0; i < b.n; ++i) vec[i] = b.vec[i];

return \*this;

}

Vector& Vector::operator=(Vector&& b) {

n = b.n;

if(vec != nullptr) delete vec;

swap(vec, b.vec);

return \*this;

}

/\*Vector& Vector::operator\*(int k) {

for (int i = 0; i < n; ++i) vec[i] = vec[i] \* k;

return \*this;

}\*/

Vector::Vector(const string& a) {

int n = int(a[0])-48;

this->n = n;

this->vec = new bool[n];

for (int i = 1; i <= n; ++i) {

this->vec[i-1] = bool(int(a[2 \* i])-48);

}

};

int main()

{

ifstream file("input.txt");

string l;

int num;

getline(file, l);

num = atoi(l.c\_str());

Vector\* masukaz = new Vector[num];

for (int i = 0; i < num; ++i) {

getline(file, l);

Vector a(l);

masukaz[i] = a;

}

cout << masukaz[0];

cout << endl;

cout << masukaz[1];

cout << endl;

bool m[5] = {1,0,0,1,1};

cout<<(m^masukaz[1]);

file.close();

//bool j[5] = {1,1,1,1,0};

//Vector vec1(m, 5);

//Vector vec3(j, 5);

//Vector vec2;

//vec2 = vec1;

//cout<< vec1[4] << endl;

//cout << vec2[4] << endl;

//cin >> vec2;

//cout<<(j ^ vec1);

}

4) Вывод: мы научились использовать перегрузку операторов класса.

ВАЖНО: Текстовый файл состоит:

1 строчка - количество векторов

2 строчка – первая цифра – количество элементов в векторе, пробел, элементы вектора через пробел.

Пример input.txt :

2

5 1 1 1 0 0

5 1 0 1 0 1