Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)

Лабораторная работа №2

Тема: Изучение перегрузки стандартных операций в языке Си++

Студент, гр. ИУ8-24 Кривицкий В. М.

Преподаватель Барыкин Д. В.

Москва

2024

Цель работы: овладение навыками разработки программ на языке Си++, использующих перегрузку стандартных операций.

Условие задачи: Дан класс (например, с именем Vector), задающий вектор размерности n. Поля класса: указатель на массив, задающий вектор (тип элемента int или double в зависимости от варианта), 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 = (присваивание с копированием), операцию = (присваивание с перемещением), а также операцию вставки (<<) объекта в поток cout или в файл (объект класса ostream) и операцию извлечения (>>) объекта из потока cin или из файла (объект класса istream). Также продемонстрировать разницу между конструктором копирования и конструктором перемещения и между операциями присваивания с копированием и перемещением. Исходные коды класса разместить в двух файлах: в заголовочном файле класса и файле реализации класса.

При выполнении работы все входные данные читаются из текстового файла input.txt (создать этот файл любым текстовым редактором), результаты выводятся в файл output.txt. В отчете представить содержимое этих файлов.

Описание перегруженной функции - скалярное произведение векторов (одинаковой размерности), на выходе значение этого произведения

Тип элемента vector - double

Типы операндов и результата для перегруженной операции: первый операнд - vector, второй операнд - vector, результат - double.

Исходный код консольного представлен ниже.

Листинг программы с комментариями:

#include <iostream>

#include <fstream> // Для работы с файлами

class Vector {

private:

double\* elements; // Указатель на массив элементов вектора

int size; // Размер вектора

public:

// Конструктор без параметров

Vector() : elements(nullptr), size(0) {}

// Конструктор для создания вектора из массива

Vector(double\* arr, int n) : size(n) {

elements = new double[size]; // Выделение памяти под массив элементов вектора

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

elements[i] = arr[i]; // Копирование элементов из переданного массива в вектор

}

}

// Конструктор копирования

Vector(const Vector& other) : size(other.size) {

elements = new double[size]; // Выделение памяти под новый массив элементов вектора

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

elements[i] = other.elements[i]; // Копирование элементов из другого вектора в текущий

}

}

// Конструктор перемещения

Vector(Vector&& other) noexcept : elements(other.elements), size(other.size) {

other.elements = nullptr; // Перемещение указателя на массив элементов из другого вектора

other.size = 0; // Обнуление размера другого вектора

}

// Деструктор

~Vector() {

delete[] elements; // Освобождение памяти, выделенной под массив элементов вектора

}

// Перегрузка оператора присваивания с копированием

Vector& operator=(const Vector& other) {

if (this != &other) {

delete[] elements; // Освобождение памяти, выделенной под текущий массив элементов вектора

size = other.size; // Присваивание нового размера

elements = new double[size]; // Выделение памяти под новый массив элементов

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

elements[i] = other.elements[i]; // Копирование элементов из другого вектора

}

}

return \*this;

}

// Перегрузка оператора присваивания с перемещением

Vector& operator=(Vector&& other) noexcept {

if (this != &other) {

delete[] elements; // Освобождение памяти, выделенной под текущий массив элементов вектора

elements = other.elements; // Перемещение указателя на массив элементов

size = other.size; // Присваивание нового размера

other.elements = nullptr; // Обнуление указателя в другом векторе

other.size = 0; // Обнуление размера другого вектора

}

return \*this;

}

// Перегрузка оператора доступа к элементу вектора по индексу

double& operator[](int index) {

if (index < 0 || index >= size) {

throw std::out\_of\_range("Index out of range"); // Проверка выхода за границы вектора

}

return elements[index]; // Возвращение ссылки на элемент вектора по указанному индексу

}

// Перегрузка оператора вставки в поток вывода

friend std::ostream& operator<<(std::ostream& out, const Vector& vec) {

out << "(";

for (int i = 0; i < vec.size; ++i) {

out << vec.elements[i]; // Вывод элементов вектора через запятую

if (i < vec.size - 1) {

out << ", ";

}

}

out << ")";

return out;

}

// Перегрузка оператора извлечения из потока ввода

friend std::istream& operator>>(std::istream& in, Vector& vec) {

for (int i = 0; i < vec.size; ++i) {

in >> vec.elements[i]; // Ввод элементов вектора из потока

}

return in;

}

// Скалярное произведение векторов

friend double operator\*(const Vector& vec1, const Vector& vec2) {

if (vec1.size != vec2.size) {

throw std::invalid\_argument("Vectors must be of the same size"); // Проверка размеров векторов

}

double result = 0;

for (int i = 0; i < vec1.size; ++i) {

result += vec1.elements[i] \* vec2.elements[i]; // Вычисление скалярного произведения

}

return result;

}

};

int main() {

std::ifstream input("input.txt"); // Открыть файл для чтения

std::ofstream output("output.txt"); // Открыть файл для записи

double arr1[3], arr2[3];

for (int i = 0; i < 3; ++i) {

input >> arr1[i]; // Чтение данных для первого вектора из файла

}

for (int i = 0; i < 3; ++i) {

input >> arr2[i]; // Чтение данных для второго вектора из файла

}

Vector vec1(arr1, 3); // Создание первого вектора

Vector vec2(arr2, 3); // Создание второго вектора

output << "Vector 1: " << vec1 << std::endl; // Вывод первого вектора в файл

output << "Vector 2: " << vec2 << std::endl; // Вывод второго вектора в файл

output << "Scalar product: " << (vec1 \* vec2) << std::endl; // Вывод скалярного произведения векторов

return 0;

}

Данные, поступающие на вход программы:  
1 2 3

4 5 6

Данные, выходящие из программы:

Vector 1: (1, 2, 3)

Vector 2: (4, 5, 6)

Scalar product: 32

Вывод: овладел навыками разработки программ на языке Си++, использующих перегрузку стандартных операций.