Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)

**Лабораторная работа № 2. Изучение перегрузки стандартных операций в языке Си++**

Работу выполнил:

Визгалов Ярослав Иванович

ИУ8-24

Вариант 3

**Цель:**

Цель работы состоит в овладении 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).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Описание операции перегруженной операции | Тип элемента вектора (массива) | Типы операндов и результата для перегруженной операции | | |
| Первый операнд | Второй операнд | Результат |
| + сложение векторов одинаковой размерности, на выходе вектор такой же размерности элемент которого равен сумме соответствующих элементов двух векторов | double | double \* | Vector | Vector |

**Текст программы**

#include <iostream>  
#include <fstream>  
using namespace std;  
  
class Vector {  
 double\* p = nullptr; // Указатель на массив, задающий вектор  
 int n = 0; // Размерность вектора (число элементов)  
 friend const Vector operator+(const double\* left, const Vector& right); // Объявляю оператор сложения дружественной функцией  
 friend ostream& operator<<(ostream& os, const Vector& v); // Объявляю оператор вывода дружественной функцией  
 friend istream& operator>>(istream& is, Vector& v); // Объявляю оператор ввода дружественной функцией  
public:  
 // Конструктор, создающий объект вектор на основе обычного одномерного массива размерности n  
 Vector(double\* p, int n) {  
 this->n = n;  
 this->p = new double[n]; // Выделение памяти под вектор  
 for (int i = 0; i < n; i++) this->p[i] = p[i]; // Копирование значений из переданного массива  
 }  
  
 // Конструктор, создающий пустой объект вектор заданной размерности  
 Vector(int n) : n(n) {  
 p = new double[n]; // Выделение памяти под вектор  
 }  
  
 // Конструктор копирования  
 Vector(const Vector& V) {  
 n = V.n;  
 p = new double[n];  
 for (int i = 0; i < n; i++)  
 p[i] = V.p[i];  
 }  
  
 // Конструктор перемещения  
 Vector(Vector&& V) {  
 std::swap(p, V.p); // Обмен указателями на массивы  
 std::swap(n, V.n); // Обмен значениями размерностей  
 }  
  
 // Функция печати вектора  
 void print() const {  
 for (int i = 0; i < n; i++)  
 cout << p[i] << " ";  
 cout << endl;  
 }  
  
 // Перегрузка оператора доступа к элементу вектора по индексу  
 double& operator[](int index) {  
 return p[index];  
 }  
  
 // Перегрузка оператора присваивания с копированием  
 Vector& operator =(const Vector& v2) {  
 if (this != &v2) {  
 n = v2.n;  
 if (p != nullptr) delete[] p; // Освобождаем память старого вектора  
 p = new double[n]; // Выделяем память для нового вектора  
 for (int i = 0; i < n; i++) p[i] = v2.p[i]; // Копируем значения из другого вектора  
 }  
 return \*this; // Возвращаем ссылку на текущий объект  
 }  
  
 // Перегрузка оператора присваивания с перемещением  
 Vector& operator =(Vector&& v2) {  
 if (this != &v2) {  
 std::swap(p, v2.p); // Обмен указателями на массивы  
 std::swap(n, v2.n); // Обмен значениями размерностей  
 }  
 return \*this; // Возвращаем ссылку на текущий объект  
 }  
 // Деструктор  
 ~Vector() {  
 if (p != nullptr) delete[] p; // Освобождаем память, если она была выделена  
 }  
};  
ostream& operator<<(ostream& os, const Vector& v) { // Перегруженный оператор вывода в поток  
 for (int i = 0; i < v.n; i++) // Вывод значений вектора через поток вывода  
 os << v.p[i] << " ";  
 return os;  
}  
  
istream& operator>>(istream& is, Vector& v) { // Перегруженный оператор ввода из потока  
 for (int i = 0; i < v.n; i++) // Считывание значений вектора из потока ввода  
 is >> v.p[i];  
 return is;  
}  
// Перегрузка оператора сложения  
const Vector operator+(const double\* left, const Vector& right) {  
 Vector result(right.n); // Создаем новый объект вектора  
 for (int i = 0; i < right.n; i++) result.p[i] = right.p[i]+left[i]; // Заполняем новый вектор суммами значений из двух объектов  
 return result; // Возвращаем новый вектор  
}  
  
int main() {  
 ifstream inputFile("input.txt");  
 int size;  
 inputFile >> size;  
 double\* values = new double[size];  
 for (int i = 0; i < size; ++i) {  
 inputFile >> values[i];  
 }  
 Vector V1(values, size);  
 double\* scalar = new double[size];  
 for (int i = 0; i < size; ++i) {  
 inputFile >> scalar[i];  
 cout << scalar[i];  
 }  
 cout << endl << V1 << endl;  
 Vector V2 = scalar + V1;  
 cout << V2;  
 return 0;  
}

Вывод программы: 4 5 5 5

Ручные рассчеты

Исходный вектор {2,2,3,4}

Массив {2,3,2,1}

2+2=4

2+3=5

3+2=5

4+1=5

Конечный вектор {4,5,5,5}

**Вывод**

В ходе выполнения лабораторной работы были изучены и применены основы перегрузки операций в языке C++. Был разработан класс Vector для работы с векторами и перегружены операции [], =, +, >>, <<. Эти операции позволяют удобно работать с векторами, выполняя над ними математические операции и копируя/перемещая их значения. Полученные результаты демонстрируют успешную реализацию перегруженных операций и их корректную работу.