ФАКУЛЬТЕТ «Информатика и системы управления»  
КАФЕДРА «Информационная безопасность»

ОТЧЕТ  
по лабораторной работе № 2  
по учебной дисциплине «Алгоритмические языки»  
на тему: «Изучение перегрузки стандартных операций в языке Си++»

Вариант 18

Выполнил:   
Студент 1 курса, гр. ИУ8-24  
Ожогин Михаил

**Цель работы:**

Овладение навыками разработки программ на языке C++, использующих перегрузку стандартных операций.

**Задачи работы:**

* Изучить необходимые учебные материалы, посвященные перегрузке стандартных операций в языке Си++
* Разработать программу на языке Си++ для решения заданного варианта задания
* Отладить программу
* Выполнить решение контрольного примера с помощью программы и ручной расчет контрольного примера
* Подготовить отчет по лабораторной работе

**Условие задачи:**

1. **Описание класса CustomVector:**
   * Класс CustomVector задает вектор размерности n.
   * Поля класса:
     1. Указатель на массив типа int, задающий вектор.
     2. Число элементов (размерность) вектора.
2. **Конструкторы класса CustomVector:**
   * Конструктор без параметров, задающий пустой вектор (число элементов равно 0).
   * Конструктор, создающий объект вектора на основе обычного одномерного массива размерности n.
   * Конструктор копирования.
   * Конструктор перемещения.
   * Деструктор.
3. **Необходимые операции для перегрузки:**
   * Операция []: обращение к элементу вектора по индексу.
   * Операция = (присваивание с копированием).
   * Операция = (присваивание с перемещением).
   * Операция вставки (<<) объекта в поток (объект класса ostream).
   * Операция извлечения (>>) объекта из потока (объект класса istream).
4. **Демонстрация разницы между конструктором копирования и конструктором перемещения**
5. **Исходные коды разместить в файлах**
   * Заголовочный файл класса
   * Файл реализации класса
6. **Обработка входных и выходных данных:**
   * Все входные данные читаются из текстового файла input.txt.
   * Результаты выводятся в файл output.txt.

**Выполнение работы:**

Заголовочный файл класса CustomVector – **CustomVector.h**:

#ifndef CUSTOMVECTOR\_H

#define CUSTOMVECTOR\_H

#include <cmath>

#include <fstream>

#include <iostream>

class CustomVector {

private:

int\* arr;

size\_t size;

size\_t capacity;

size\_t closestPowerOfTwo(size\_t num) const;

void resize(size\_t newCapacity);

public:

CustomVector();

CustomVector(int\* int\_arr, size\_t& vec\_size);

CustomVector(const CustomVector& copied\_vec);

CustomVector(CustomVector&& moved\_vec) noexcept;

~CustomVector();

int operator[](size\_t index) const;

CustomVector& operator--();

friend std::ostream& operator<<(std::ostream& os, const CustomVector& vec);

friend std::istream& operator>>(std::istream& is, CustomVector& vec);

void push\_back(const int& value);

};

CustomVector::CustomVector() : arr(new int[1]), size(0), capacity(1) {}

CustomVector::CustomVector(int\* int\_arr, size\_t& vec\_size) : arr(int\_arr), size(vec\_size), capacity(closestPowerOfTwo(vec\_size)) {}

CustomVector::CustomVector(const CustomVector& other) : arr(new int[other.capacity]), size(other.size), capacity(other.capacity) {

for (size\_t i = 0; i < other.size; ++i) {

arr[i] = other.arr[i];

}

}

CustomVector::CustomVector(CustomVector&& moved\_vec) noexcept : arr(moved\_vec.arr), size(moved\_vec.size), capacity(moved\_vec.capacity) {

moved\_vec.arr = nullptr;

moved\_vec.size = 0;

moved\_vec.capacity = 0;

}

CustomVector::~CustomVector() {

delete[] arr;

}

int CustomVector::operator[](size\_t index) const {

return arr[index];

}

CustomVector& CustomVector::operator--() {

for (size\_t i = 0; i < size; ++i) {

--arr[i];

}

return \*this;

}

std::ostream& operator<<(std::ostream& os, const CustomVector& vec) {

os << "[";

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

os << vec.arr[i] << " ";

}

os << "]";

return os;

}

std::istream& operator>>(std::istream& is, CustomVector& vec) {

int value;

is >> value;

vec.push\_back(value);

return is;

}

void CustomVector::push\_back(const int& value) {

if (size == capacity) {

resize(capacity \* 2);

}

arr[size++] = value;

}

void CustomVector::resize(size\_t newCapacity) {

int\* new\_arr = new int[newCapacity];

for (size\_t i = 0; i < size; ++i) {

new\_arr[i] = arr[i];

}

delete[] arr;

arr = new\_arr;

capacity = newCapacity;

}

size\_t CustomVector::closestPowerOfTwo(size\_t num) const {

if (num == 0 || num == 1) {

return 1;

}

size\_t power = std::ceil(std::log2(num));

return std::pow(2, power);

}

void customPrint(const std::string& str) {

std::ofstream os("output.txt",std::ios::app);

os << str;

std::cout << str;

os.close();

}

void customPrint(const CustomVector &vec) {

std::ofstream os("output.txt",std::ios::app);

os << vec;

std::cout << vec;

os.close();

}

#endif // CUSTOMVECTOR\_H

Файл реализации – **main.cpp**:

#include "../include/CustomVector.h"

int main() {

// Clearing output.txt file

std::ofstream os("output.txt");

os.close();

// Reading input from input.txt file

std::ifstream is("input.txt");

if (!is.is\_open()) {

std::cerr << "Failed to open input.txt" << std::endl;

return 1;

}

// Creating a vector object using the default constructor

CustomVector vec1;

// Creating a vector object based on data from input.txt

size\_t size;

is >> size;

int\* arr = new int[size];

for (size\_t i = 0; i < size; ++i) {

is >> arr[i];

}

CustomVector vec2(arr, size);

customPrint("Outputting elements of the vector:\n");

customPrint("Vector 1 (created using the default constructor): ");

customPrint(vec1);

customPrint("\n");

customPrint("Vector 2 (created using data from input.txt): ");

customPrint(vec2);

customPrint("\n\n");

customPrint("Filling vector 1 with data from the input.txt but data coming in Vector 1 directly:\n");

is >> size;

for (size\_t i = 0; i < size; ++i) {

is >> vec1;

}

is.close();

customPrint(vec1);

customPrint("\n\n");

customPrint("Example of the prefix-decrement operation on Vector 2:\n");

customPrint(--vec2);

customPrint("\n\n");

customPrint("Example of assignment operation with copying:\n");

CustomVector vec3 = vec1;

customPrint("Vector 3 (copied from Vector 1): ");

customPrint(vec3);

customPrint("\n\n");

customPrint("Example of assignment operation with moving:\n");

CustomVector vec4 = std::move(vec2);

customPrint("Vector 4 (moved from Vector 2): ");

customPrint(vec4);

customPrint("\n");

customPrint("Vector 2 after move: ");

customPrint(vec2); // Vector 2 is now empty

customPrint("\n");

return 0;

}

Файл со входными данными – **input.txt**

5

126923 9524 5 0 -156

5

-194 1247 0 5 12950

Файл с выходными данными – **output.txt**

Outputting elements of the vector:

Vector 1 (created using the default constructor): []

Vector 2 (created using data from input.txt): [126923 9524 5 0 -156 ]

Filling vector 1 with data from the input.txt but data coming in Vector 1 directly:

[-194 1247 0 5 12950 ]

Example of the prefix-decrement operation on Vector 2:

[126922 9523 4 -1 -157 ]

Example of assignment operation with copying:

Vector 3 (copied from Vector 1): [-194 1247 0 5 12950 ]

Example of assignment operation with moving:

Vector 4 (moved from Vector 2): [126922 9523 4 -1 -157 ]

Vector 2 after move: []

**Пример и контрольный расчет примера:**

Пусть задан вектор длины 3 с элементами 10 0 -15. Проведя операцию префиксного декремента, мы получим вектор с элементами 9 -1 -16, то есть каждый элемент вектора уменьшился на единицу. Обратимся к работе, где задан вектор длины 5 с элементами 126923 9524 5 0 -156. Проведя операцию префиксного декремента, мы получаем вектор, 126922 9523 4 -1 -157. Отсюда видно, что каждый элемент вектора уменьшился ровно на единицу, а следовательно оператор реализован верно.

**Вывод:**

В ходе выполнения лабораторной работы были изучены основы перегрузки стандартных операций в языке C++. Были реализованы операции работы с векторами, такие как обращение к элементу по индексу, копирование и перемещение векторов, операция префиксного декремента. В процессе работы были изучены способы перегрузки операций в классах C++, в том числе перегрузка как членами класса, так и дружественными функциями.