|  |  |
| --- | --- |
|  | **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 № 1

ПО КУРСУ

«Алгоритмические языки»

на тему «**Изучение перегрузки стандартных операций в языке Си++**»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ8-23 |  |  |  | Д.С. Афанасьев |
|  | (Группа) |  |  |  | (И. О. Фамилия) |
|  |  |  |  |  |  |
| Преподаватель: |  |  |  |  | М. В. Малахов |
|  |  |  |  |  | (И.О. Фамилия) |

2024

Введение

Цели и задачи работы

Цель работы состоит в овладении навыками разработки программ на языке Си++, использующих перегрузку стандартных операций. Для достижения цели необходимо выполнить следующие задачи:

- изучить необходимые учебные материалы, посвященные перегрузке стандартных операций в языке Си++ ;

- разработать программу на языке Си++ для решения заданного варианта задания;

- отладить программы;

- выполнить решение контрольного примера с помощью программы и ручной расчет контрольного примера;

- подготовить отчет по лабораторной работе.

Условия для 4 варианта

Студент в работе создает класс и необходимые оператор- функции для перегрузки заданных в своем варианте операций. Особое внимание обратить на способы перегрузки унарных и бинарных операций и параметры оператор- функций для этих операций, когда оператор функция является членом класса и когда не является. Знать те случаи, когда оператор функция должна быть обязательно членом класса и случаи, когда оператор- функция обязательно не принадлежит классу. При защите работы необходимо обосновать выбор способа определения оператор- функции – внутри класса или вне его. При необходимости студент выполняет ручной расчет для проверки работы программы для задачи небольшой размерности. Результаты работы программы, ручного расчета представляются в отчете.

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.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Описание операции перегруженной операции | Тип элемента вектора (массива) | Типы операндов и результата для перегруженной операции | | | № варианта |
| Первый операнд | Второй операнд | Результат |  |
| + сложение векторов, на выходе вектор, длина которого сумме длин векторов вначале идут элементы первого вектора, затем второго, если один из векторов задан обычным массивом, то считать, что его длина равна длине вектора, заданным объектом класса | double | Vector | Vector | Vector | 4 |
|
|

Основная часть

Исходный текст программы:

Vector.h

#include <iostream>

#ifndef VECTOR\_H

#define VECTOR\_H

class Vector {

private:

  double\* elems;

  int sz;

public:

  Vector();

  Vector(double\*, int);

  Vector(const Vector&);

  Vector(Vector&&);

  ~Vector();

  int size();

  bool empty();

  void clear();

  double operator [] (int);

  Vector& operator = (const Vector&);

  Vector& operator = (Vector&&);\

  Vector& operator ++ ();

  Vector operator ++ (int);

};

std::istream& operator >> (std::istream&, Vector&);

std::ostream& operator << (std::ostream&, Vector&);

Vector operator + (Vector&, Vector&);

#endif

Vector.cpp

#include “Vector.h”

#include <iostream>

Vector::Vector() {

  elems = nullptr;

  sz = 0;

  // std::cout << “Empty vector” << std::endl;

}

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

  sz = size;

  elems = new double[sz];

  for (int i = 0; i < sz; ++i) elems[i] = arr[i];

  // std::cout << “vector whith array” << std::endl;

}

Vector::Vector(const Vector& other\_vec) {

  sz = other\_vec.sz;

  elems = new double[sz];

  for (int i = 0; i < sz; ++i) elems[i] = other\_vec.elems[i];

  // std::cout << “vector whith another one copy “ << std::endl;

}

Vector::Vector(Vector&& other\_vec) {

  //if (elems) delete[] elems;

  elems = nullptr;

  sz = 0;

  std::swap(sz, other\_vec.sz);

  std::swap(elems, other\_vec.elems);

  // std::cout << “vector whith another one replace” << std::endl;

}

Vector::~Vector() {

  if (elems) {

    delete[] elems;

    sz = 0;

    // std::cout << “Vector is cleared.” << std::endl;

  }

  elems = nullptr;

}

int Vector::size() {

  return sz;

}

bool Vector::empty() {

  return sz == 0;

}

void Vector::clear() {

  if (sz == 0) return;

  delete[] elems;

  elems = nullptr;

  sz = 0;

  // std::cout << “Vector is cleared” << std::endl;

}

double Vector::operator [] (int idx) {

  if (idx >= sz || idx < 0) {

    std::cerr << “Index out of range” << std::endl;

    return 1e9 + 1;

  }

  return elems[idx];

}

Vector& Vector::operator = (const Vector& other\_vec) {

  if (elems == other\_vec.elems) return \*this;

  if (elems) delete[] elems;

  sz = other\_vec.sz;

  elems = new double[sz];

  for (int i = 0; i < sz; ++i) elems[i] = other\_vec.elems[i];

  std::cout << “copy value” << std::endl;

  return \*this;

}

Vector& Vector::operator = (Vector&& other\_vec) {

  if (elems) delete[] elems;

  sz = 0;

  elems = nullptr;

  std::swap(sz, other\_vec.sz);

  std::swap(elems, other\_vec.elems);

  std::cout << “replace object” << std::endl;

  return \*this;

}

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

  int sz;

  in >> sz;

  double\* arr = new double[sz];

  for (int i = 0; i < sz; ++i) {

    in >> arr[i];

  }

  vec = Vector(arr,sz);

  if (arr) delete[] arr;

  return in;

}

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

  out << vec.size() << ‘\n’;

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

    out << vec[i] << ‘ ‘;

  }

  out << std::endl;

  return out;

}

Vector operator + (Vector& vec1, Vector& vec2) {

  int sz = vec1.size() + vec2.size();

  double\* arr = new double[sz];

  for (int i = 0; i < vec1.size(); ++i) arr[i] = vec1[i];

  for (int i = 0; i < vec1.size(); ++i) arr[vec1.size() + i] = vec2[i];

  Vector res\_vec(arr, sz);

  if (arr) delete[] arr;

  return res\_vec;

}

Vector& Vector::operator ++ () {

  for (int i = 0; i < sz; ++i) ++elems[i];

  return \*this;

}

Vector Vector::operator ++ (int) {

  Vector vec = \*this;

  for (int i = 0; i < sz; ++i) elems[i]++;

  return vec;

}

Lab2.cpp

#include "Vector.h"

#include <fstream>

#include <iostream>

#include <string>

int main() {

  double arr[] = {1.0, 2.0, 3.0, 4.0, 5.0};

  std::ifstream in;

  Vector vec1;

  Vector vec2;

  Vector vec3(arr, 5);

  in.open("./input/input\_lab2.txt");

  if (!in.is\_open()) {

    std::cerr << "Error: Could not open input file\n";

    return 1;

  }

  in >> vec2;

  in.close();

  vec1 = vec2;

  vec1 = vec1 + vec2;

  std::cout << vec1[0] << ' ' << vec1[1] << ' ' << vec1[2] << std::endl;

  vec1 = vec1;

  std::cout << vec1[0] << ' ' << vec2[0] << std::endl;

  vec2 = vec1++;

  std::cout << vec1[0] << ' ' << vec2[0] << std::endl;

  vec2 = ++vec1;

  std::cout << vec1[0] << ' ' << vec2[0] << std::endl;

  std::cout << vec1[0] << ' ' << vec1[1] << ' ' << vec1[2] << std::endl;

  std::ofstream out;

  out.open("./output/output\_lab2.txt");

  if (!out.is\_open()) {

    std::cerr << "Error: Could not open output file\n";

    return 1;

  }

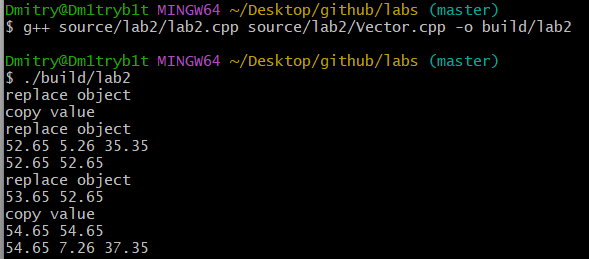
  out << vec1 << vec2 << vec3;

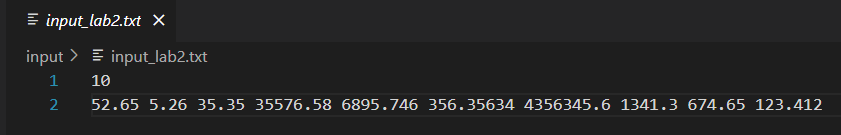
  out.close();

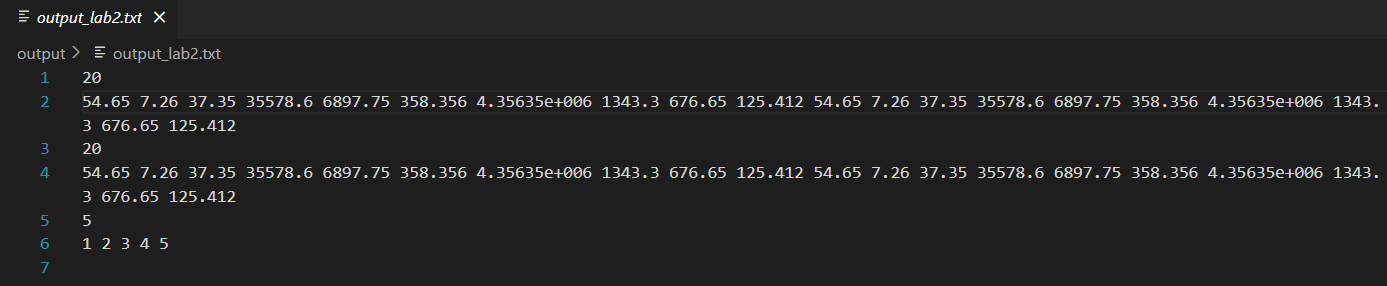
  return 0;

}

Снимки выполнения работы программы







Заключение

Задачи лабораторной работы были решены, результаты проверены. На практике Приобретены навыки разработки программ на языке Си++, использующих перегрузку стандартных операций.