МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных

технологий (МОСИТ)

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 6**

**по дисциплине «***Структуры и алгоритмы обработки данных***»**

**тема «**Создание класса. Перегрузка операций**»**

Выполнил: студент группы ИВБО-02-15 Прохоров А.В.

Приняла: старший преподаватель кафедры Скворцова Л.А.

**1. Условие задания**

**Цель:** получить навыки по обработке с классами и перегрузкой операций.

**Задание 1**

Разработайте АТД согласно варианту задания.

**Задание 2**

Реализуйте АТД варианта, используя пользовательский тип – класс.

Включить в класс методы:

* + конструктор по умолчанию;
  + конструктор с параметрами;
  + копирующий конструктор и деструктор, если используется динамическая память для представления данных.

Перегрузить операции:

* + операцию>>для вывода экземпляров класса в поток ostream.
  + выполните перегрузку дополнительных операций, указанных в задании варианта.

**Задание 3**

Разработайте программу, демонстрирующую применение класса.

|  |  |  |
| --- | --- | --- |
| 5 | Массив одномерный динамический вещественных чисел | - Удаление из одного массива элементов другого  = = Равенство двух массивов |

**2. Определение структуры представления данных и операций над данными**

Массив—структура данных, представленная в виде группы ячеек памяти одного типа, объединенных одним именем. Имя массива является указателем на его первый элемент. Отдельная ячейка массива—элемент массива.

Перегрузка функций—определение нескольких функций или операторов с одинаковыми именами, но различными параметрами.

**3. Разработка программы**

**3.1. Декомпозиция**

Список подзадач:

1) Создание массива

2) Заполнение массива

3) Вывод массива

4) Определение размера массива

5) Оператор >>

6) Удаление элементов второго массива из первого

**3.2. Определение функций**

int Array::getSize() const — возвращает размер массива

bool Array::operator==(const Array &right) const — перегруженный оператор сравнения

istream &operator>> (istream & input, Array &obj) — перегруженный оператор ввода

void Array::setArray() — заполнение массива

void Array::getArray() — вывод массива

void Array::compare() — удаление из первого элементов второго

void menu() — меню программы

**3.3. Код разработанного приложения**

#include <iostream>

#include <cstdlib>

#include <ostream>

#include <iomanip>

using namespace std;

class Array

{

friend istream &operator>> (istream &, Array &); // перегруженный оператор ввода

private:

float \*ptr; // указатель на массив

int size; // размер массива

public:

Array(); // конструктор по умолчанию

Array(int); // конструктор с параметрами

Array(const Array &); // конструктор копии

~Array(); // десструктор

int getSize() const;

bool operator== (const Array &) const; // оператор сравнения

Array &operator- (const int &); // оператор удаления элемента по ключу

float &operator[] (int);

void setArray(); // заполнение массива

void getArray(); // вывод массива

void compare(const Array &) const; //удаление из первого элементов второго

};

Array::Array() // конструктор по умолчанию, без параметров

{

size = 10; // по умолчанию размер массива = 10 элементов

ptr = new float[size]; // выделить место в памяти для массива

for (int ix = 0; ix < size; ix++) // обнуляем массив

ptr[ix] = 0;

}

Array::Array(int arraySize) // конструктор с параметрами

{

// если значение параметра больше 0, присвоить size значение arraySize, иначе - 10

size = (arraySize > 0 ? arraySize : 10);

ptr = new float[size]; // выделить место в памяти для массива

for (int ix = 0; ix < size; ix++) // обнуляем массив

ptr[ix] = 0;

}

Array::Array(const Array &arrayToCopy) // конструктор копии

:size(arrayToCopy.size) // инициализатор размера массива

{

ptr = new float[size]; // выделить место в памяти для массива

for (int ix = 0; ix < size; ix++)

ptr[ix] = arrayToCopy.ptr[ix]; // заполняем массив значениями массива arrayToCopy

}

Array::~Array() // десструктор класса Array

{

delete[] ptr; // освободить память, удалив массив

}

int Array::getSize() const // возвратить количество элементов массива

{

return size;

}

bool Array::operator==(const Array &right) const// оператор сравнения

{

if (size != right.size)

return false; // массивы с разным количеством элементов

for (int ix = 0; ix < size; ix++)

if (ptr[ix] != right.ptr[ix])

return false; // массивы не равны

return true; // массивы равны

}

float &Array::operator[] (int subscript)

{

if (subscript < 0 || subscript >= size)

{

std::cerr << "\n Ошибка индекса: " << subscript << std::endl;

exit(0); // завершить работу программы, неправильный индекс

}

return ptr[subscript]; // возврат ссылки на элемент массива

}

Array &Array::operator- (const int & key) // оператор удаления элемента по ключу

{

int counterKey = 0; // счетчик найденных ключевых элементов

// определяем количество элементов, которые необходимо удалить

for (int ix = 0; ix < size; ix++)

{

if (ptr[ix] == key)

counterKey++;

}

Array temp(size);

temp = \*this; // сохраняем текущий массив во временный объект

delete[] ptr; // jосвобождае память

size = size - counterKey; // переопределяем размер

ptr = new float[size];

int counter = 0, ix = 0;

while (counter < size)

{

if (temp[counter] != key)

{

ptr[ix] = temp.ptr[counter]; // скопировать массив

ix++;

}

counter++;

}

return \*this;

}

// перегруженный оператор ввода, для ввода значений массива с клавиатуры

istream &operator>> (istream & input, Array &obj)

{

for (int ix = 0; ix < obj.size; ix++)

input >> obj.ptr[ix]; // заполняем массив объекта obj

return input; // позволяет множественный ввод, типа cin >> x >> y >> z >> ...

}

void Array::setArray() // заполнение массива

{

cout << "Введите размер массива: ";

cin >> size;

cout << "Введите массив: " << endl;

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

cout << "Элемент " << ix << ": ";

cin >> ptr[ix]; // ввод элемента массива с клавиатуры

}

}

void Array::getArray() // вывод массива

{

cout << "Массив: " << endl;

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

cout << setw(5) << ptr[ix]; // вывод элементов массива на экран

}

cout << endl; // новая строка

}

void Array::compare(const Array &right) const

{

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

for (int ix = 0; ix < right.size; ix++) {

if (ptr[ix] != right.ptr[ix] && ix>1) {

ptr[ix-1] = ptr[ix];

}

}

}

}

void menu() {

cout << "1. Создать 2 массива:" << endl;

cout << "2. Вывести их на экран:" << endl;

cout << "3. Сравнить массивы:" << endl;

cout << "4. Удалить из первого элементы второго:" << endl;

cout << "5. Выйти:" << endl;

}

int main()

{

setlocale(LC\_ALL, "Russian");

Array myArray1;

Array myArray2;

Array myArray3(myArray1); // создаем объект класса Array, используя конструктор копирования

myArray3 = myArray2; // присваиваем массиву myArray3 массив myArray2

char c;

do

{

menu();

cin >> c;

switch (c)

{

case '1':

myArray1.setArray();

myArray2.setArray();

break;

case '2':

myArray1.getArray();

myArray2.getArray();

break;

case '3':

if (myArray1 == myArray2)

cout << "Массивы myArray1 и myArray2 равны!" << endl;

else

cout << "Массивы myArray1 и myArray2 не равны!" << endl;

break;

case '4':

myArray1.compare(myArray2);

myArray1.getArray();

break;

case '5':

exit(0);

default: cout << "Неверный пункт меню!" << endl;

}

system("Pause");

system("cls");

} while (c != '5');

system("Pause");

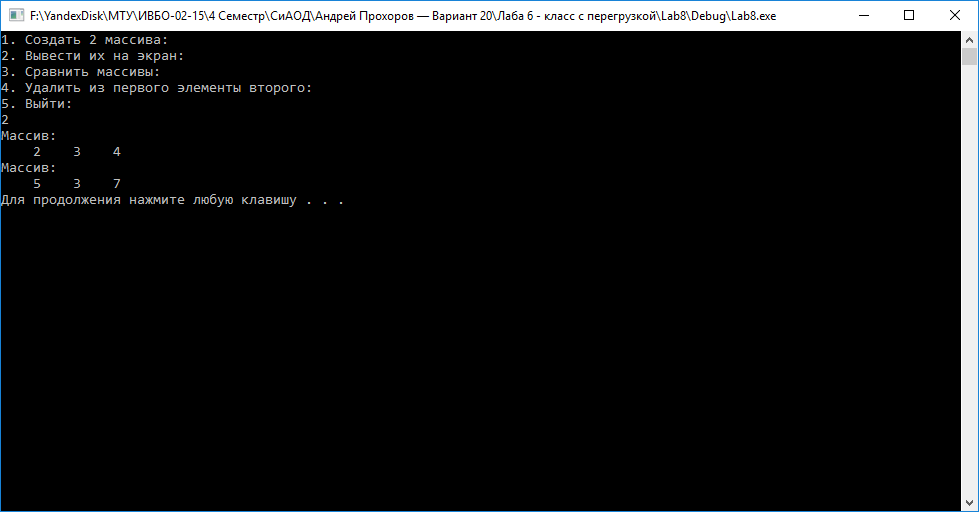
return 0;

}

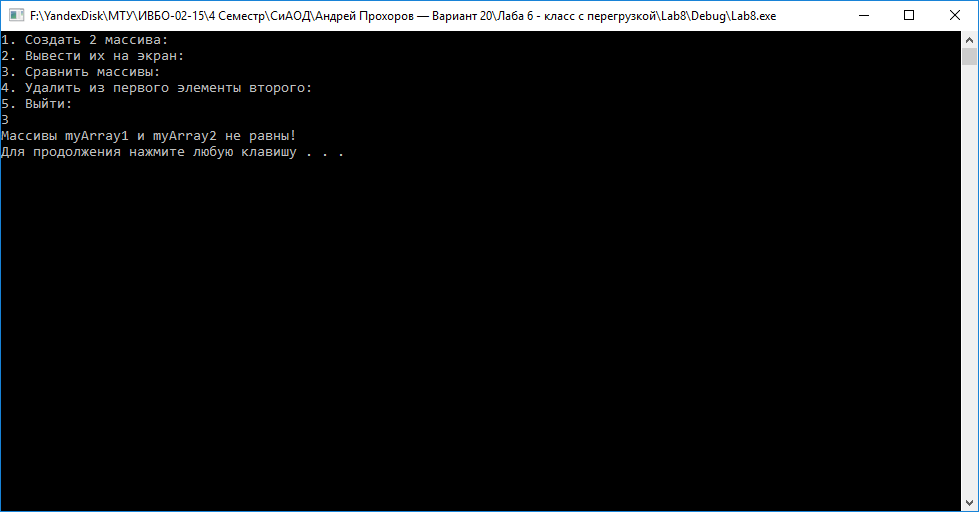
**4. Тестирование приложения**

**4.1. Контрольные тесты программы**

Тест №1—создание и вывод 2 массивов



Тест №2—сравнение массивов



Тест №3—удаление элементов второго массива из первого

