МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

старший преподаватель С.А.Рогачёв должность, уч. степень звание подпись, дата инициалы, фамилия

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №5

«АЛГОРИТМЫ СОРТИРОВКИ»

по дисциплине: СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. Z7431 22.01.2018 М.Д.Семочкин

подпись, дата инициалы, фамилия

Санкт-Петербург

2018

**1. Цель работы**

Целью работы является изучение алгоритмов внутренней сортировки и получение практических навыков их использования, и анализа их сложности.

**2. Задание на лабораторную работу**

Использовать неупорядоченный массив A, содержащий n целочисленных элементов. Величина n определяется по согласованию с преподавателем. Дополнительно в программе должны быть реализованы следующие функции:

1)  Поиск элемента либо по его порядковой позиции, либо по его содержимому;

2)  Добавление/удаление элемента с последующей пересортировкой последовательности;

3)  В программе должен быть реализован подсчет количества сравнений и перестановок, при осуществлении сортировки.

Согласно варианту 26,

Задание: Найти k-ое по порядку число среди элементов массива

Алгоритм сортировки: Четно – нечетная

**3. Листинг программы, реализующей алгоритм**

#include <iostream>

#include <limits>

#include <time.h>

#include <stdlib.h>

**using** **namespace** std;

**const** **int** MIN\_RAND\_VALUE = -99;

**const** **int** MAX\_RAND\_VALUE = 99;

**void** sortArray(**int** \*arr, **int** n) {

*// Четно-нечетная сортировка*

*// Счетчики сравнений и перестановок*

**int** swapCounter = 0;

**int** comparsionCounter = 0;

**for** (**int** i = 0; i < n; i++) {

*// (i % 2) ? 0 : 1 используется для определения четности/нечетности индекса*

**for** (**int** j = (i % 2) ? 0 : 1; j < n - 1; j += 2) {

comparsionCounter++;

**if** (arr[j] > arr[j + 1]) {

*// Поменять элементы местами с помощью дополнительной переменной*

**int** tmp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = tmp;

swapCounter++;

}

}

}

cout << "Количество сравнений: " << comparsionCounter << endl;

cout << "Количество перестановок: " << swapCounter << endl;

}

**void** fillArrWithRandomNumbers(**int** \*arr, **int** n) {

*// Заполняет массив случайными числами*

srand(time(**NULL**));

**for** (**int** i = 0; i < n; i++) {

arr[i] = MIN\_RAND\_VALUE + (rand() % (MAX\_RAND\_VALUE - MIN\_RAND\_VALUE + 1));

}

}

**void** addElemToArray(**int** \*&arr, **int** n, **int** value) {

*// Добавляет элемент в массив*

*// n - длина массива до добавления элемента!*

*// Копируем массив в новый массив длины n+1*

**int** \*newArr = **new** **int**[n + 1];

**for** (**int** i = 0; i < n; i++) {

newArr[i] = arr[i];

}

*// Удаляем старый массив*

**delete**[] arr;

*// В последний элемент нового массива записываем value*

newArr[n] = value;

*// Записываем в arr ссылку на новый массив*

arr = newArr;

}

**void** deleteElemFromArray(**int** \*&arr, **int** n, **int** index) {

*// Удаляет элемент из массива*

*// n - длина массива до удаления элемента!*

*// Копируем массив в новый массив длины n-1 до index*

**int** \*newArr = **new** **int**[n - 1];

**for** (**int** i = 0; i < index; i++) {

newArr[i] = arr[i];

}

*// Копируем остальные элементы, пропустив index*

**for** (**int** i = (index + 1); i < n; i++) {

newArr[i - 1] = arr[i];

}

*// Удаляем старый массив*

**delete**[] arr;

*// Записываем в arr ссылку на новый массив*

arr = newArr;

}

**int** findElemByValue(**int** \*arr, **int** n, **int** value) {

*// Возвращает индекс элемента, если он найден, и -1 если не найден*

**for** (**int** i = 0; i < n; i++) {

**if** (arr[i] == value) {

**return** i;

}

}

**return** -1;

}

**void** printArray(**int** \*arr, **int** n) {

cout << "=========== МАССИВ ==========" << endl;

**for**(**int** i = 0; i < n; ++i) {

cout << arr[i];

**if** (i < n - 1) {

cout << "\t ";

}

}

cout << endl << "=============================" << endl;

}

**int** main() {

setlocale(LC\_ALL, "russian");

**int** n; *// Длина массива*

**int** \*arr; *// Указатель на массив*

*// Ввод длины массива*

cout << "Введите количество элементов в массиве: ";

cin >> n;

**while** (cin.fail() || n <= 0) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Введена некорректная длина массива!" << endl;

cout << "Повторите ввод: ";

cin >> n;

}

arr = **new** **int**[n];

fillArrWithRandomNumbers(arr, n);

cout << "Массив создан и заполнен случайными числами от "

<< MIN\_RAND\_VALUE << " до " << MAX\_RAND\_VALUE << endl;

*// Меню пользователя*

**int** menuInput = -1;

**int** userInput = -1;

**while** (menuInput != 0) {

cout << endl <<

"====== ДОСТУПНЫЕ КОМАНДЫ =====" << endl <<

"1 - Вывести массив на экран" << endl <<

"2 - Отсортировать массив" << endl <<

"3 - Поиск элемента по индексу" << endl <<

"4 - Поиск элемента по значению" << endl <<

"5 - Добавить элемент в массив и отсортировать заново" << endl <<

"6 - Удалить элемент из массива по индексу и "

<< "отсортировать заново" << endl <<

"7 - Удалить элемент из массива по значению и "

<< "отсортировать заново" << endl <<

"0 - Выход" << endl <<

"==============================" << endl <<

"Выберите действие: ";

cin >> menuInput;

cout << endl;

**while** (cin.fail() || (menuInput < 0) || (menuInput > 7)) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка ввода, выберите действие: ";

cin >> menuInput;

}

**switch** (menuInput) {

*// Вывести массив на экран*

**case** 1: {

printArray(arr, n);

**break**;

*// Отсортировать массив*

} **case** 2: {

sortArray(arr, n);

cout << "Массив отсортирован!" << endl;

**break**;

*// Поиск элемента по индексу*

} **case** 3: {

cout << "Введите индекс: ";

cin >> userInput;

**while** (cin.fail() || userInput < 0 || userInput >= n) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Индекс должен быть меньше длины массива!"

<< endl << "Повторите ввод: ";

cin >> userInput;

}

cout << "Элемент с индексом " << userInput << " имеет значение "

<< arr[userInput] << endl;

**break**;

*// Поиск элемента по значению*

} **case** 4: {

cout << "Введите значение: ";

cin >> userInput;

**while** (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Вы должны ввести целое число!"

<< endl << "Повторите ввод: ";

cin >> userInput;

}

**int** foundIndex = findElemByValue(arr, n, userInput);

**if** (foundIndex == -1) {

cout << "Элемент с данным значением не найден" << endl;

} **else** {

cout << "Элемент со значением " << userInput <<

" найден по индексу " << foundIndex << endl;

}

**break**;

*// Добавить элемент в массив и отсортировать заново*

} **case** 5: {

cout << "Введите новый элемент массива: ";

cin >> userInput;

**while** (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Вы ввели недопустимое значение, попробуйте еще: ";

cin >> userInput;

}

addElemToArray(arr, n, userInput);

n++;

cout << "Элемент добавлен!" << endl;

sortArray(arr, n);

cout << "Массив отсортирован!" << endl;

**break**;

*// Удалить элемент из массива по индексу и*

*// отсортировать заново*

} **case** 6: {

cout << "Введите индекс элемента, который хотите удалить: ";

cin >> userInput;

**while** (cin.fail() || userInput < 0 || userInput >= n) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Индекс должен быть меньше длины массива!"

<< endl << "Повторите ввод: ";

cin >> userInput;

}

deleteElemFromArray(arr, n, userInput);

n--;

cout << "Элемент удален!" << endl;

sortArray(arr, n);

cout << "Массив отсортирован!" << endl;

**break**;

*// Удалить элемент из массива по значению и*

*// отсортировать заново*

} **case** 7: {

cout << "Введите значение элемента, который хотите удалить: ";

cin >> userInput;

**while** (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Вы ввели недопустимое значение, попробуйте еще: ";

cin >> userInput;

}

**int** foundIndex = findElemByValue(arr, n, userInput);

**if** (foundIndex == -1) {

cout << "Элемент с данным значением не найден" << endl;

} **else** {

deleteElemFromArray(arr, n, foundIndex);

n--;

cout << "Элемент со значением " << userInput <<

" найден по индексу " << foundIndex << " и удален." << endl;

sortArray(arr, n);

cout << "Массив отсортирован!" << endl;

}

**break**;

}

}

}

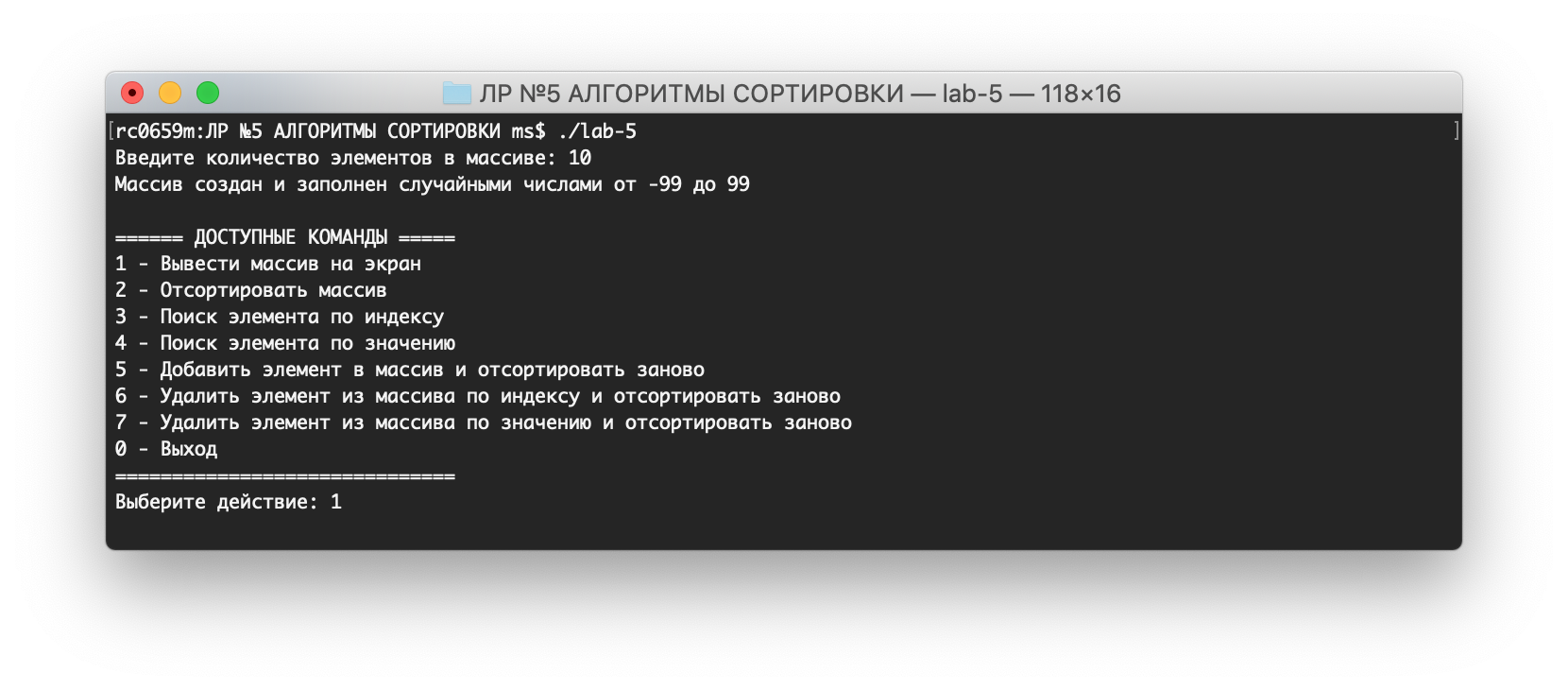
**return** 0;

}

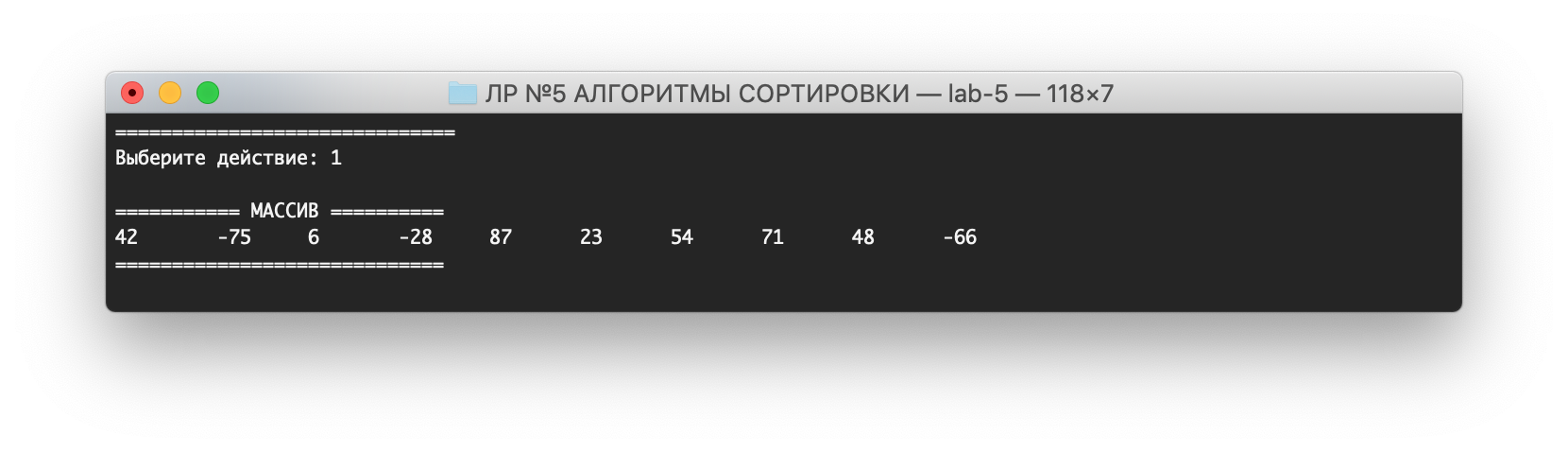
**4. Пример работы программы**

1) Программа имеет меню пользователя. Пользователь должен ввести нужную цифру, чтобы выполнить действие, или 0 чтобы выйти из программы.

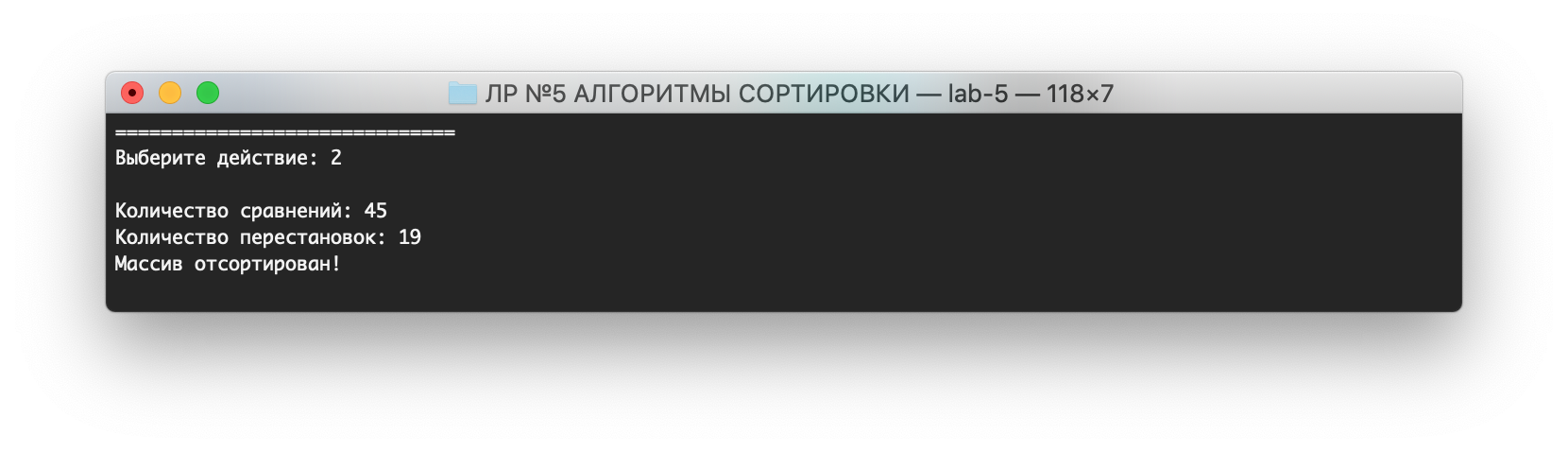
При запуске программы создается массив введенной пользователем длины и заполняется случайными числами. Границы значений случайных чисел определяются константами.



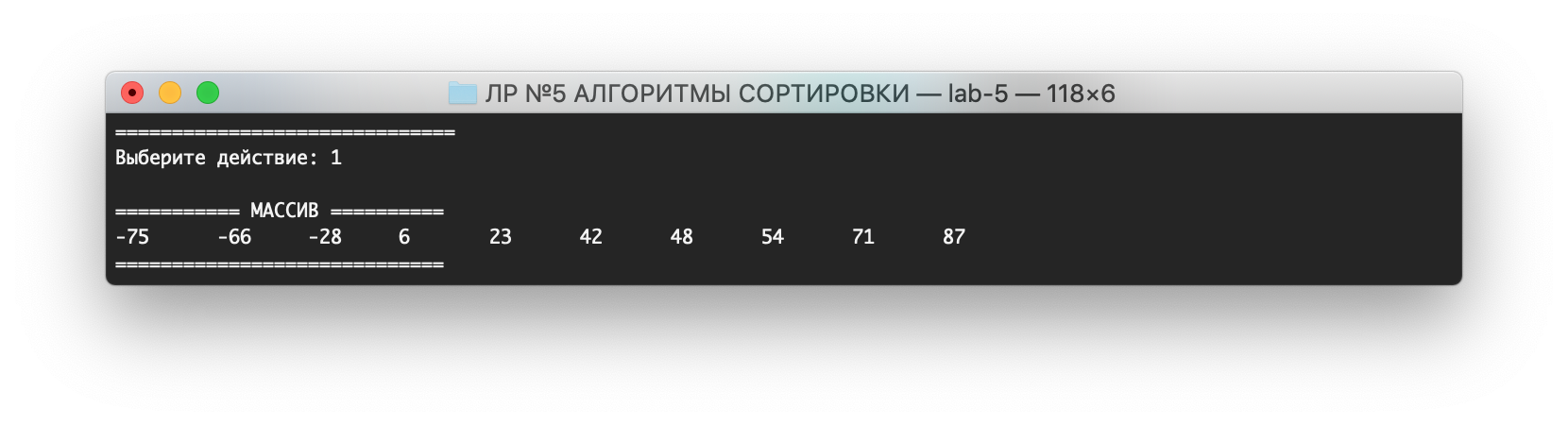
2) 1 – команда вывода массива на экран



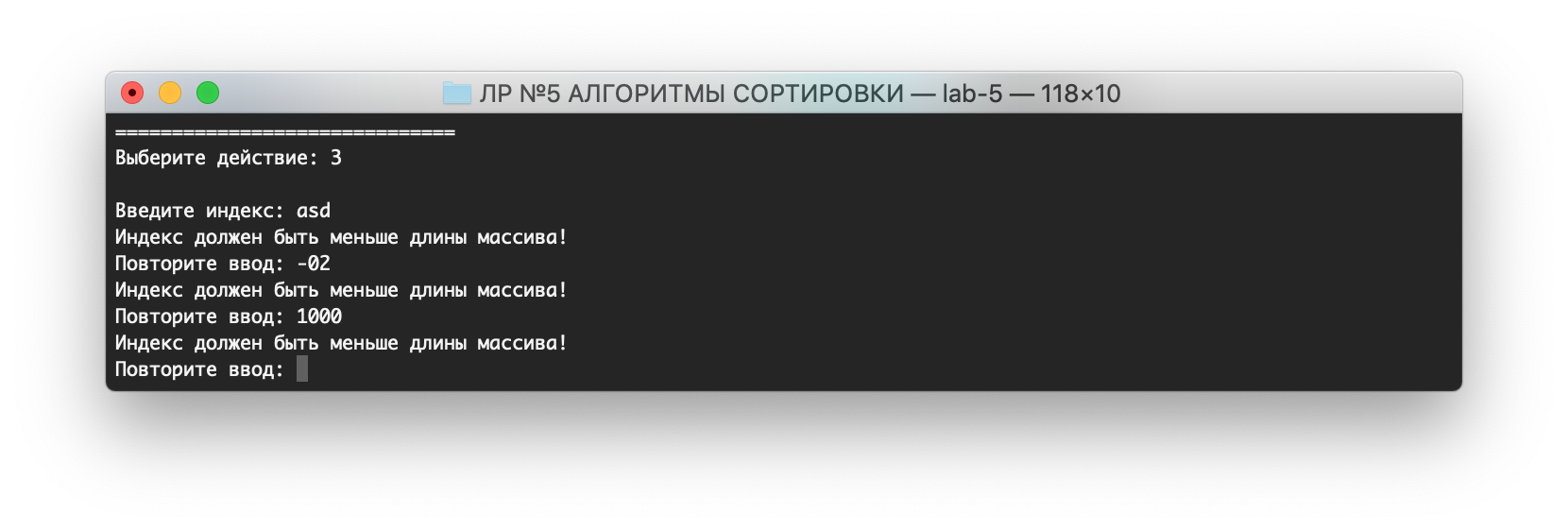
3) 2 – команда сортирует массив методом четно-нечетной сортировки, а также подсчитывает количество сравнений и перестановок во время выполнения сортировки

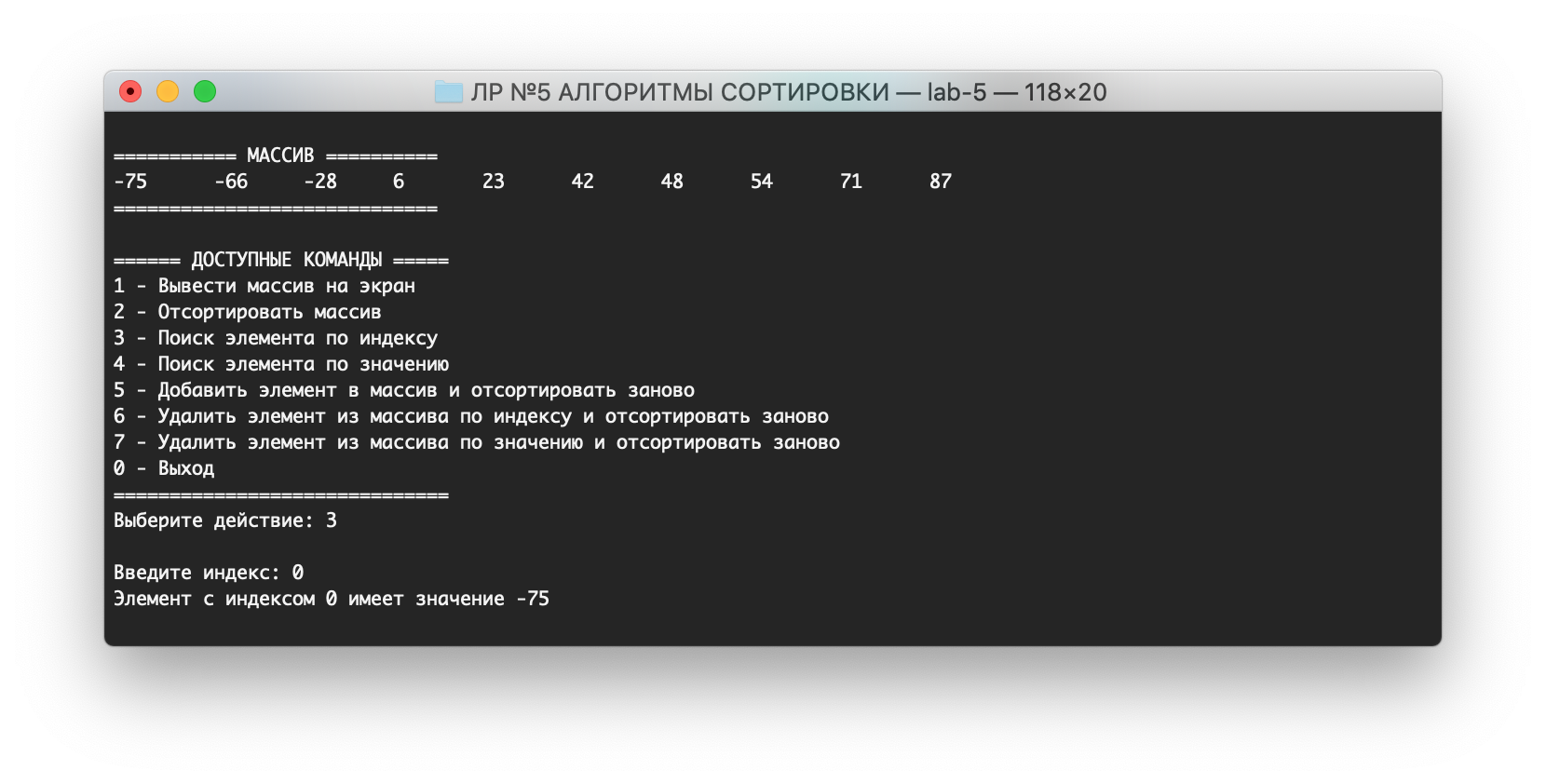


После чего можно снова вывести массив на экран командой 1:

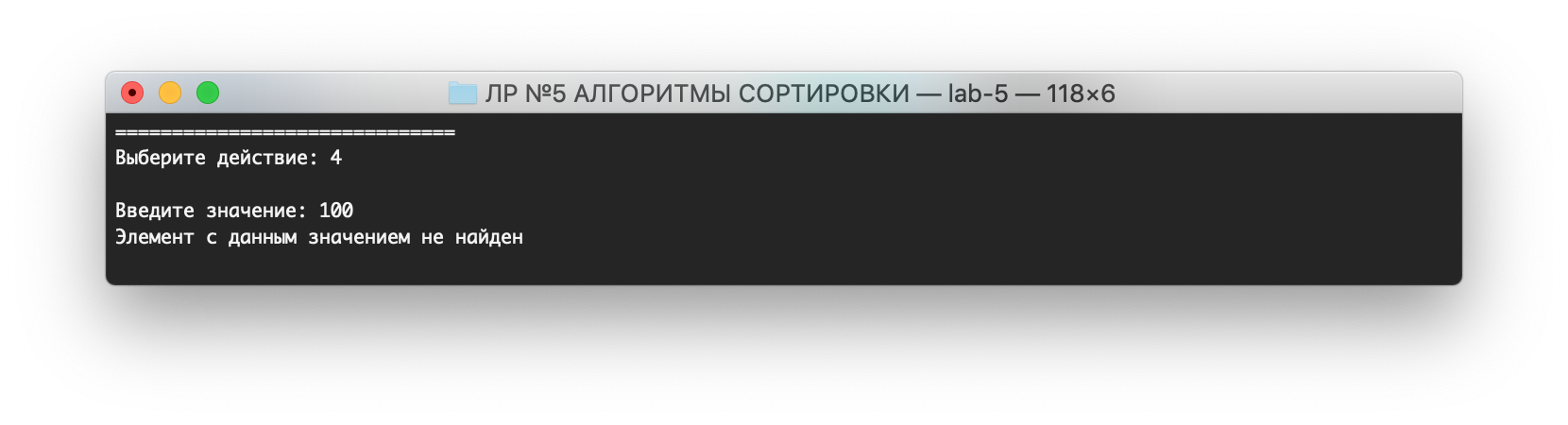


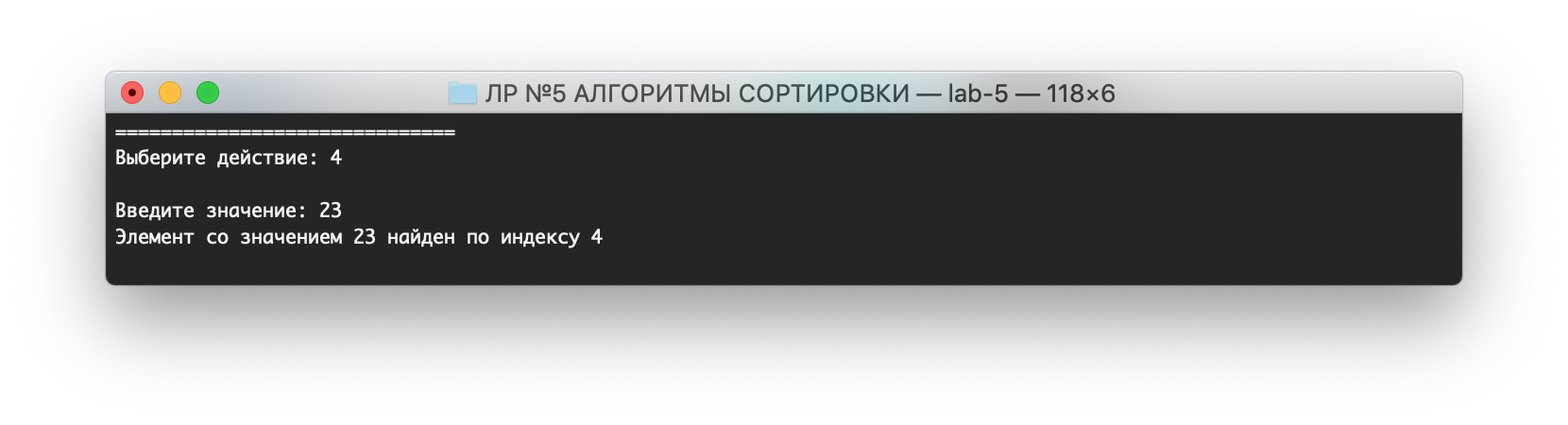
4) 3 – команда ищет и выводит элемент в массиве по индексу



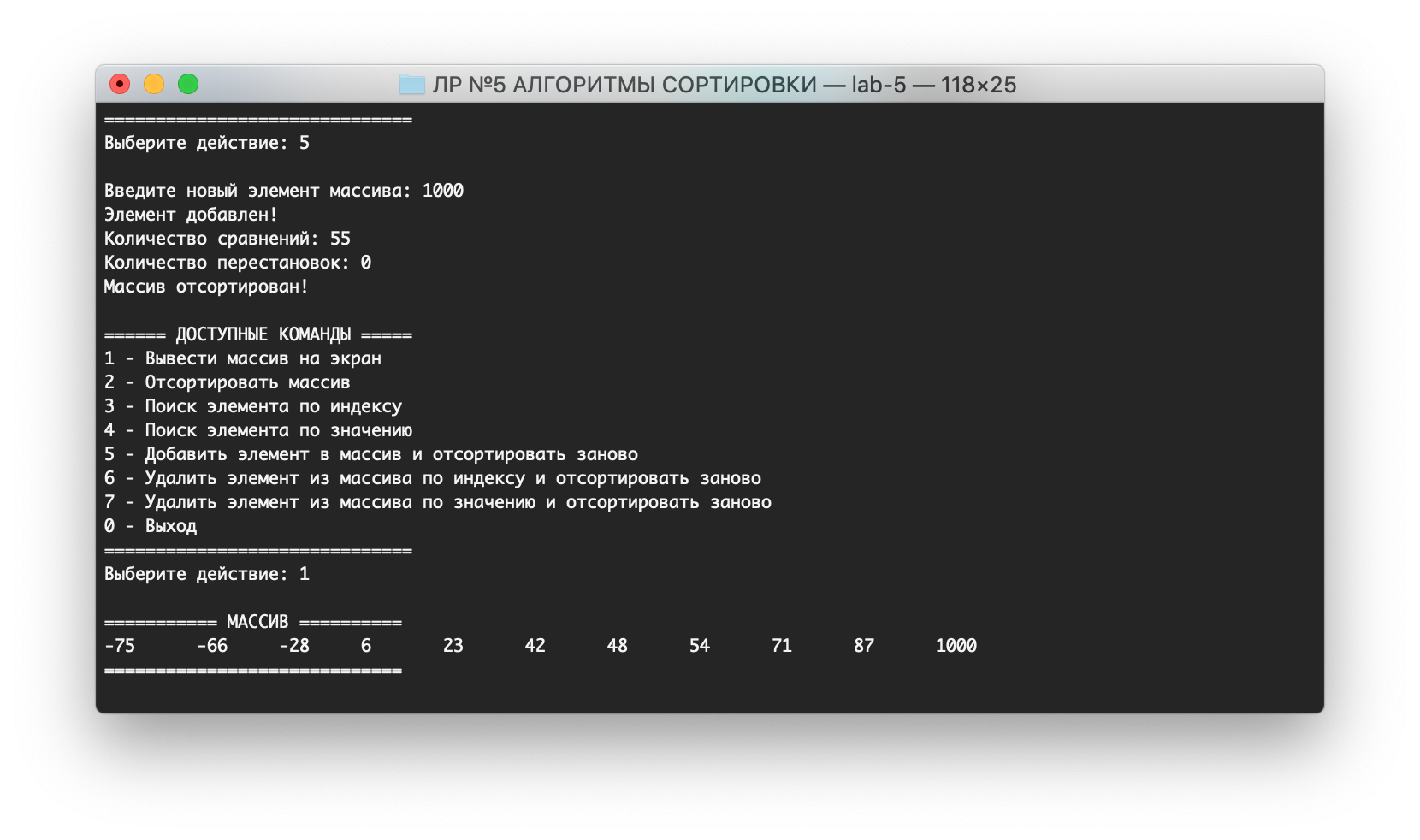


5) 4 – команда ищет и выводит элемент в массиве по его значению

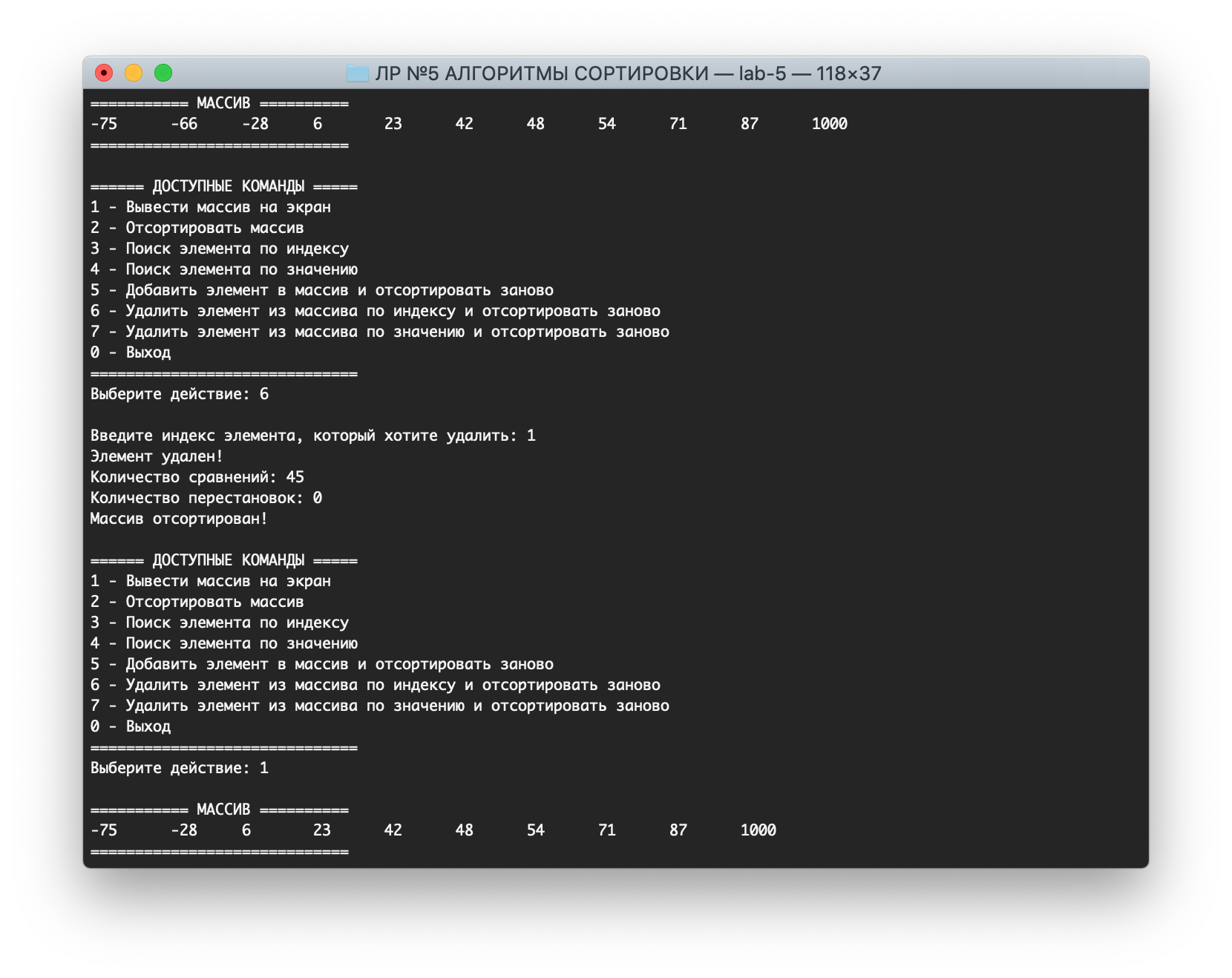




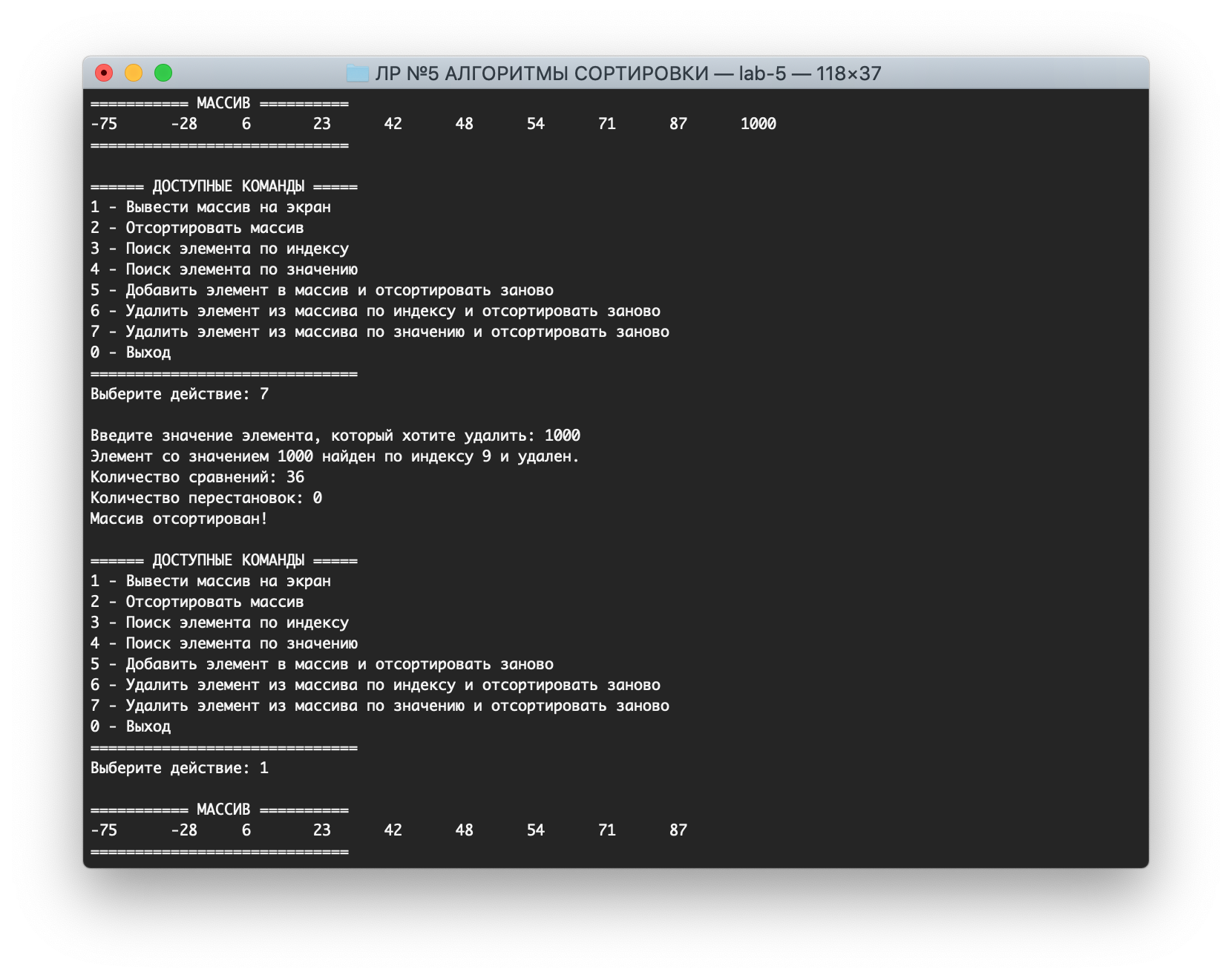
6) 5 – команда добавляет элемент в массив и сортирует его заново



7) 6 – команда удаляет элемент из массива по индексу и сортирует его заново



8) 7 – команда находит элемент, значение которого введено пользователем, удаляет его и заново сортирует массив



5) Для выполнения задания “Найти k-ое по порядку число среди элементов массива” нужно запустить программу и выполнить команду 2 для сортировки массива, и 3 для вывода k-ого числа.

**5. Временная и пространственная сложность алгоритма**

Рассмотрим функцию sortArray, реализующую алгоритм четно-нечетной сортировки:

Разработанный алгоритм использует следующие данные:  
−один массив размерностью n = sizeof(arr)/sizeof(int);  
−три переменные целого типа. Значит, пространственная сложность алгоритма определяется следующим образом:

v = + n\*Cint + 3\*Cint

где Cint – константа, характеризующая объем памяти, отводимый под переменную целого типа.

Теоретическая пространственная сложность алгоритма составляет:

V(n) = O(v) = O(max(O(n\*Cint), O(3\*Cintt))) = O(max(O(n), O(1), O(1))) = O(n)

Теоретическую временную сложность алгоритма определяем на основе анализа текста программы, реализующей данный алгоритм. Для начала рассчитаем теоретическую временную алгоритма сортировки:

TSort = O(max(O(K1), O(n2 \*K2))) = O(max(O(1), O(n2))) = O(n2),

Где K1 – операции сравнения, присваивания, используемые в алгоритме и имеющие временную сложность «1», K2 – цикличные операции, занимающие в наихудшем случае шагов.

**6. Вывод**

Была создана и протестирована программа с пользовательским интерфейсом для работы с массивом. Были изучены некоторые виды сортировки и основы работы с массивами в C++. Программа работает без ошибок и позволяет выполнить задачу, данную в задании.