МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

#### Факультет информационных технологий и компьютерной безопасности

(факультет)

### Кафедра Систем автоматизированного проектирования и информационных систем

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

по дисциплине Объектно-ориентированное программирование

Тема: «Работа с массивами»

Выполнили студенты группы ИСТ-222 Е.Г. Гладнева

Подпись, дата Инициалы, фамилия

Е.В. Журавлёва

Подпись, дата Инициалы, фамилия

## Принял Б.Н. Тишуков

Подпись, дата Инициалы, фамилия

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

2023

Лабораторная работа № 2

«Работа с массивами»

1 Цель работы

Целью работы является изучение и умение работать с массивами.

2 Задание на лабораторную работу

* Написать программу в соответствии с вариантом задания для обработки

одномерного массива (6 вариант).

Дан одномерный массив, состоящий из N целочисленных элементов.

Заполнить массив случайными числами.

Найти максимальный положительный элемент.

Вычислить сумму элементов массива.

Вывести ненулевые элементы на экран в обратном порядке.

* Написать программу в соответствии с вариантом задания для обработки

двумерного массива. Дан двумерный массив размерностью 5 × 6, заполненный целыми числами с клавиатуры. Сформировать одномерный массив, каждый элемент которого равен наибольшему по модулю элементу соответствующего столбца.

* Выполнить пирамидальную сортировку элементов одномерного массива.

3 Ход выполнения

Для начала создадим переменную n и дадим пользователю возможность ввести значение для этой переменной (кол - во элементов массива), далее с помощью рандомного заполнения заполним массив. Ищем максимальный положительный элемент и сумму элементов, идя по массиву и складывая все. Следующим этапом проходимся по массиву и находим ненулевые элементы и идём теперь с последнего элемента до первого. Для примера возьмём n = 3. Результат проделанной работы представлен на рисунке 1.

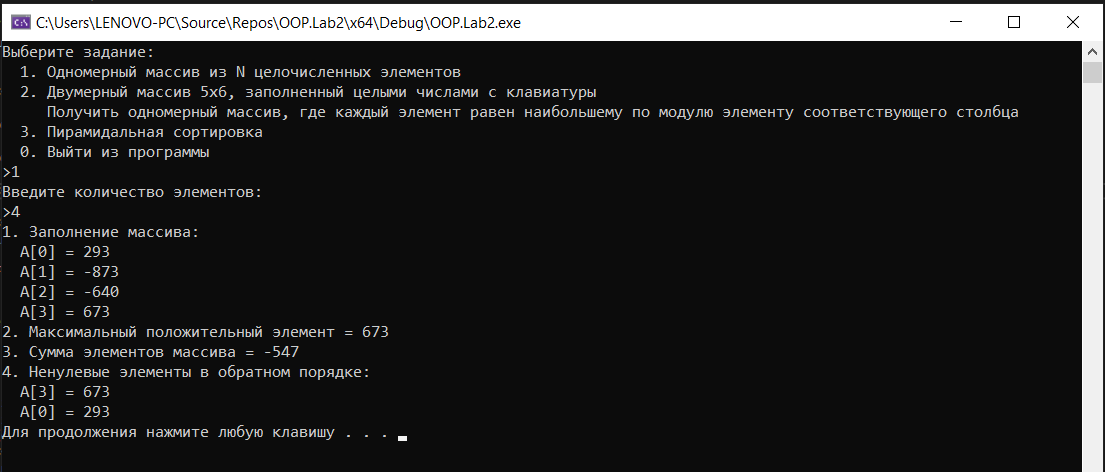


Рисунок 1 — Одномерный массив, состоящий из N целочисленных элементов

Создадим переменные с = 5, m = 6, дадим пользователю ввести матрицу размером 5 × 6, считаем введённую матрицу. После чего проходимся по ней и начнём с первого элемента в первом столбце, присвоив его B[i], далее идём по матрице и сравниваем значение числа по модулю каждого из чисел с текущим, найдя максимальный присваиваем его B[i]. Результат представлен на рисунке 2.

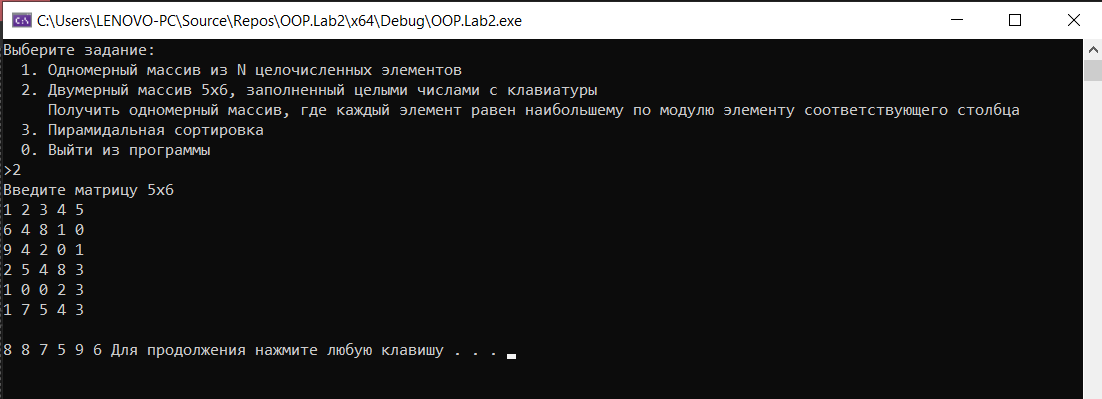


Рисунок 2 — Сформированный новый одномерный массив

Создаём новую переменную n – количество элементов массива и даём пользователю ввести значение для неё. Заполняем массив рандомно и сортируем его, для этого инициализируем наибольший элемент как корень, далее в переменную largest (наибольший) запихиваем левый дочерний элемент, если он больше корня, либо правый, если он самый большой элемент на данный момент, если же самый большой элемент не корень, тогда рекурсивно преобразуем в двоичную кучу затронутое поддерево. Возьмём за n число 5. Результат представлен на рисунке 3.

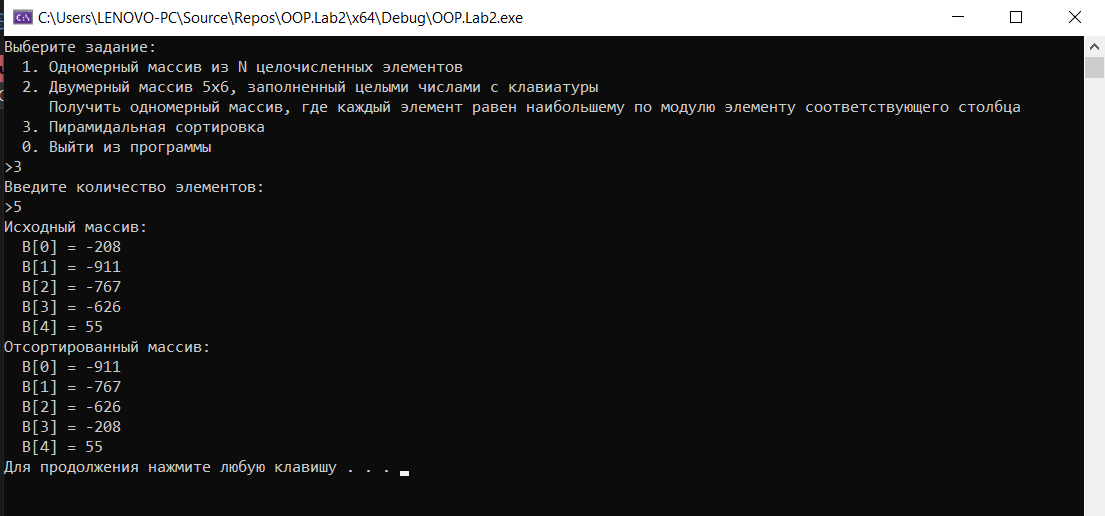


Рисунок 3 — Отсортированный пирамидально массив

Вывод

В ходе выполнения лабораторной работы мы изучили и научились работать с массивами.

Приложение А

Листинг программного кода языка C++

#include <iostream>

#include <cmath>

/\*Дан двумерный массив размерностью 5 6, заполненный целыми числами с клавиатуры.

Сформировать одномерный массив, каждый элемент которого равен наибольшему по модулю

элементу соответствующего столбца.\*/

#include <time.h>

#define MIN -1000

#define MAX 1000

using namespace std;

int\* heap\_sort(int\* arr, int n); // пирамидальная сортировка

int main()

{

setlocale(LC\_ALL, "RUS");

srand((unsigned int)time(NULL));

int choice = -1; // выбранный пункт меню

int n;

int x, sum = 0;

int\* arr;

while (choice != 0)

{

cout << "Выберите задание:" << endl

<< " 1. Одномерный массив из N целочисленных элементов" << endl

<< " 2. Двумерный массив 5х6, заполненный целыми числами с клавиатуры" << endl

<< " Получить одномерный массив, где каждый элемент равен наибольшему по модулю элементу соответствующего столбца" << endl

<< " 3. Пирамидальная сортировка" << endl

<< " 0. Выйти из программы" << endl

<< ">";

cin >> choice;

switch (choice)

{

case 1:

{

cout << "Введите количество элементов:\n>";

cin >> n;

arr = new int[n];

cout << "1. Заполнение массива:" << endl;

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

{

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

cout << " A[" << i << "] = " << arr[i] << endl;

// поиск макс. + элемента

x = -1;

if (arr[i] > 0 && arr[i] > x)

x = arr[i];

// сумма элементов

sum += arr[i];

}

cout << "2. Максимальный положительный элемент = ";

if (x != -1)

cout << x << endl;

else

cout << "отсутствует" << endl;

cout << "3. Сумма элементов массива = " << sum << endl;

cout << "4. Ненулевые элементы в обратном порядке:" << endl;

for (int i = n - 1; i >= 0; i--)

{

if (arr[i] > 0)

cout << " A[" << i << "] = " << arr[i] << endl;

}

delete[] arr;

break;

}

case 2:

{

const int с = 5;

const int m = 6;

int A[с][m], B[m];

cout << "Введите матрицу 5x6" << endl;

for (int i = 0; i < с; i++)

for (int j = 0; j < m; j++) // считываем матрицу

cin >> A[i][j];

for (int i = 0; i < m; i++)

{

B[i] = abs(A[0][i]);

for (int j = 1; j < с; j++)

if (abs(A[j][i]) > B[i]) // сравниваем со значением по модулю

B[i] = A[j][i]; // присваиваем B[i] значение этого максимального элемента

}

cout << endl;

for (int i = 0; i < m; i++)

cout << B[i] << ' ';

break;

}

case 3:

{

cout << "Введите количество элементов:\n>";

cin >> n;

arr = new int[n];

cout << "Исходный массив:" << endl;

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

{

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

cout << " B[" << i << "] = " << arr[i] << endl;

}

arr = heap\_sort(arr, n);

cout << "Отсортированный массив:" << endl;

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

{

cout << " B[" << i << "] = " << arr[i] << endl;

}

delete[] arr;

break;

}

case 0:

exit(0);

default:

{

cout << "Нет такого варианта" << endl;

break;

}

}

system("pause");

system("cls");

}

return 0;

}

int\* heapify(int\* arr, int n, int i)

{

int largest = i;

// Инициализируем наибольший элемент как корень

int left = 2 \* i + 1;

int right = 2 \* i + 2;

// Если левый дочерний элемент больше корня

if (left < n && arr[left] > arr[largest])

largest = left;

// Если правый дочерний элемент больше, чем самый большой элемент на данный момент

if (right < n && arr[right] > arr[largest])

largest = right;

// Если самый большой элемент не корень

if (largest != i)

{

swap(arr[i], arr[largest]);

// Рекурсивно преобразуем в двоичную кучу затронутое поддерево

arr = heapify(arr, n, largest);

}

return arr;

}

// Основная функция, выполняющая пирамидальную сортировку

int\* heap\_sort(int\* arr, int n)

{

// Построение кучи (перегруппируем массив)

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

// Один за другим извлекаем элементы из кучи

for (int i = n - 1; i >= 0; i--)

{

// Перемещаем текущий корень в конец

swap(arr[0], arr[i]);

// вызываем процедуру heapify на уменьшенной куче

arr = heapify(arr, i, 0);

}

return arr;

}