

ГУАП
КАФЕДРА №43

ОТЧЁТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

К.Т.Н., доц. _____
должность, уч. степень, звание

подпись, дата

А.В. Туманова _____
инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

ОБРАБОТКА ЧИСЛОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. Z7431

 26.11.2018
подпись, дата

 М.Д.Семочкин
инициалы, фамилия

Санкт-Петербург
2018

1. Цель работы

Целью работы является изучение структуры данных одномерный массив.

2. Задание

Согласно варианту №14,

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. количество элементов массива, равных 0;
2. сумму элементов массива, расположенных после минимального элемента.

Упорядочить элементы массива по возрастанию модулей элементов.

При написании программ можно использовать как динамические, так и нединамические массивы. Размерность последних задаётся именованной константой.

Примечание: массивы, созданные с помощью классов (таких, как например, `array` или `vector` из библиотеки STL) использовать запрещается.

Примечание: если использовались операторы динамического выделения памяти, то следует вставить дополнительный код, обнаруживающий утечки памяти. См. раздел «Как обнаружить утечки памяти?». Все утечки памяти должны быть устранены.

3. Описание созданных функций

Для реализации задания нам потребуются следующие функции:

Имя: `countZeroElements`

Назначение: Найти количество элементов массива, равных 0

Входные данные:

- `*arr` – указатель на массив
- `n` – количество элементов в массиве

Выходные данные:

- `counter` – количество элементов массива, равных 0

Побочный эффект: отсутствует.

Тестовые данные:

массив	результат
[30, 20, 5, -1, 35, 0, 5, 0]	2

Прототип: `int countZeroElements(float *arr, int n)`

Алгоритм:

- псевдокод

счетчик = 0

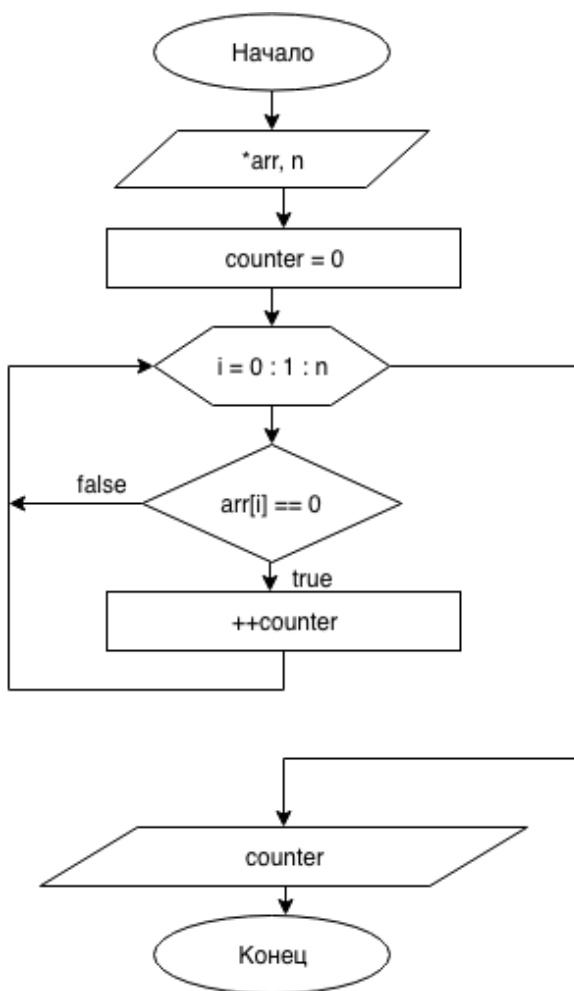
для каждого элемента в массиве

если он равен 0

увеличить счетчик на 1

вернуть счетчик

- блок-схема



Имя: getSumAfterMin

Назначение: Вернуть сумму элементов массива, находящихся после его минимального элемента

Входные данные:

- *arr – указатель на массив
- n – количество элементов в массиве

Выходные данные:

- sum – сумма элементов, находящихся после его минимального элемента

Побочный эффект: отсутствует.

Тестовые данные:

массив	результат
[30, 20, 5, -1, 35, 0, 5, 0]	40

Прототип: float getSumAfterMin(float *arr, int n)

Алгоритм:

- псевдокод

индекс минимального элемента = 0

сумма = 0

для каждого элемента в массиве

 если он меньше чем элемент массива с индексом минимального элемента

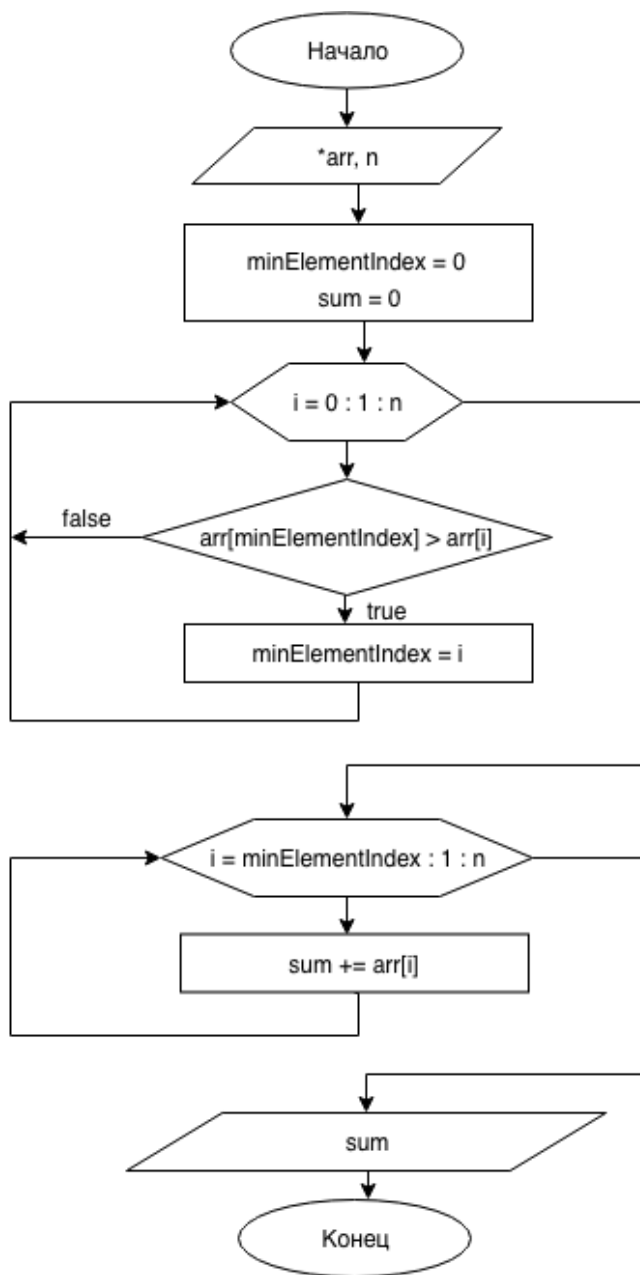
 индекс минимального элемента = индекс текущего элемента

для всех элементов массива от элемента с индексом минимального элемента до
последнего элемента массива

 прибавить к сумме значение элемента

вернуть сумму

- блок-схема



Имя: sortArray

Назначение: Отсортировать массив по возрастанию модулей его элементов

Входные данные:

- *arr – указатель на массив
- n – количество элементов в массиве

Выходные данные: нет

Побочный эффект: исходный массив перезаписывается отсортированным массивом и не сохраняется

Тестовые данные:

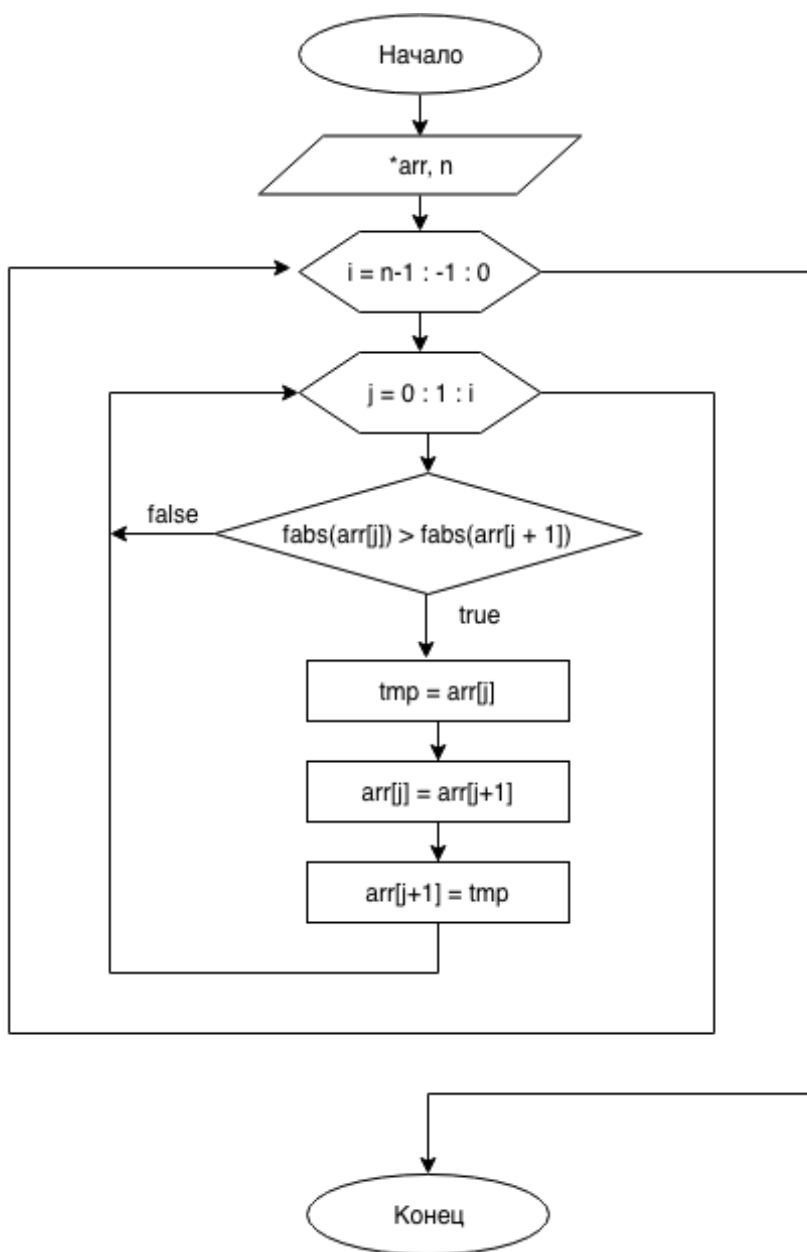
массив	результат
[30, 20, 5, -1, 35, 0, 5, 0]	[0, 0, -1, 5, 5, 20, 30, 35]

Прототип: void sortArray(float *arr, int n)

Алгоритм:

- псевдокод
 - для каждого элемента в массиве от последнего до первого
 - для каждого элемента в массиве от первого до текущего
 - если элемент больше следующего по модулю
 - поменять его местами со следующим

- блок-схема



4. Листинг программы

```
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <string>
using namespace std;

int countZeroElements(float *arr, int n) {
    // Считает нулевые элементы в массиве
    int counter = 0;
    for(int i = 0; i < n; ++i)
        if(arr[i] == 0)
            ++counter;
    return counter;
}

float getSumAfterMin(float *arr, int n) {
    // Возвращает сумму элементов массива, находящихся
    // после его минимального элемента
    int minElementIndex = 0;
    float sum = 0;

    // Найти индекс минимального элемента
    for(int i = 0; i < n; ++i)
        if(arr[minElementIndex] > arr[i])
            minElementIndex = i;

    // Найти сумму
    for(int i = minElementIndex + 1; i < n; ++i)
        sum += arr[i];

    return sum;
}

void sortArray(float *arr, int n) {
    for(int i = (n - 1); i >= 1; --i) {
        for(int j = 0; j < i; ++j) {
            if(fabs(arr[j]) > fabs(arr[j + 1])) {
                // Поменять элементы местами с помощью дополнительной
                // переменной
                float tmp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = tmp;
            }
        }
    }
}

void printArray(float *arr, int n) {
    for(int i = 0; i < n; ++i)
        cout << arr[i] << endl;
}

int main() {
    int n;
    float *arr, sumAfterMin;
    setlocale(LC_ALL, "russian");

    cout << "Введите количество элементов в массиве: ";
    cin >> n;
```



```
if (n <= 0) {
    cout << "Введена некорректная длина массива" << endl;
    return 0;
}

arr = new float[n];
for(int i = 0; i < n; ++i) {
    cout << "Введите " << i << "-й элемент: ";
    cin >> arr[i];
}

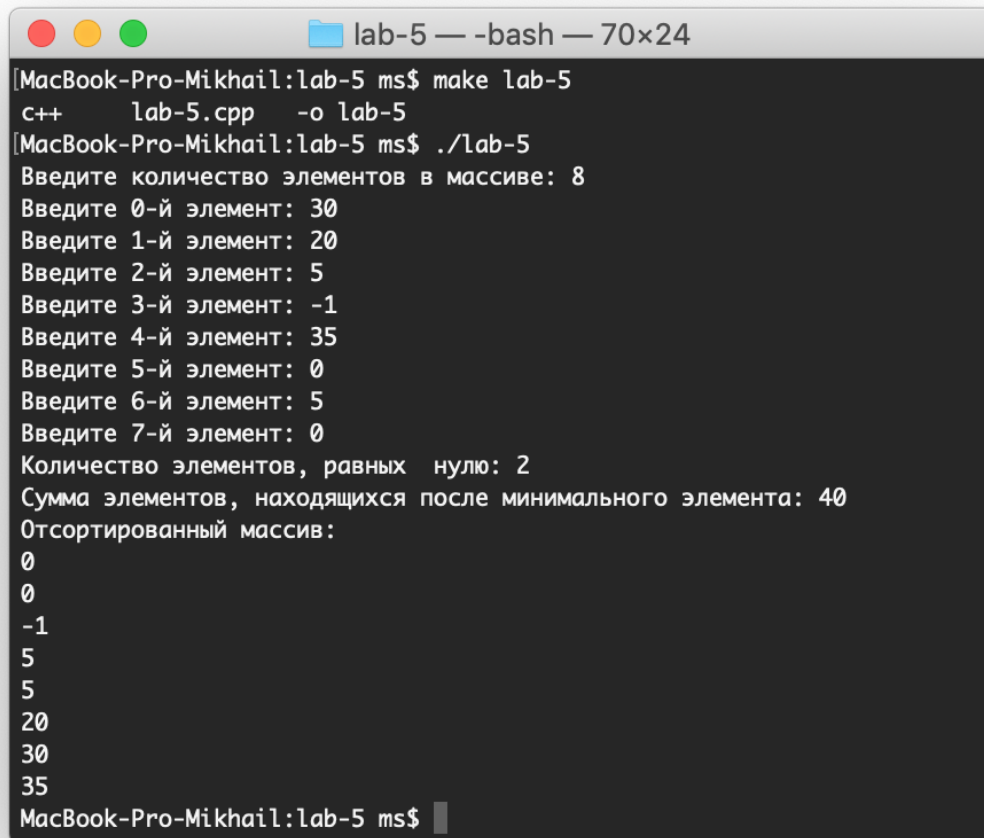
cout << "Количество элементов, равных нулю: " << countZeroElements(arr, n)
<< endl;
cout << "Сумма элементов, находящихся после минимального элемента: ";
cout << getSumAfterMin(arr, n) << endl;

sortArray(arr, n);
cout << "Отсортированный массив: " << endl;
printArray(arr, n);

return 0;
}
```

5. Пример выполнения программы

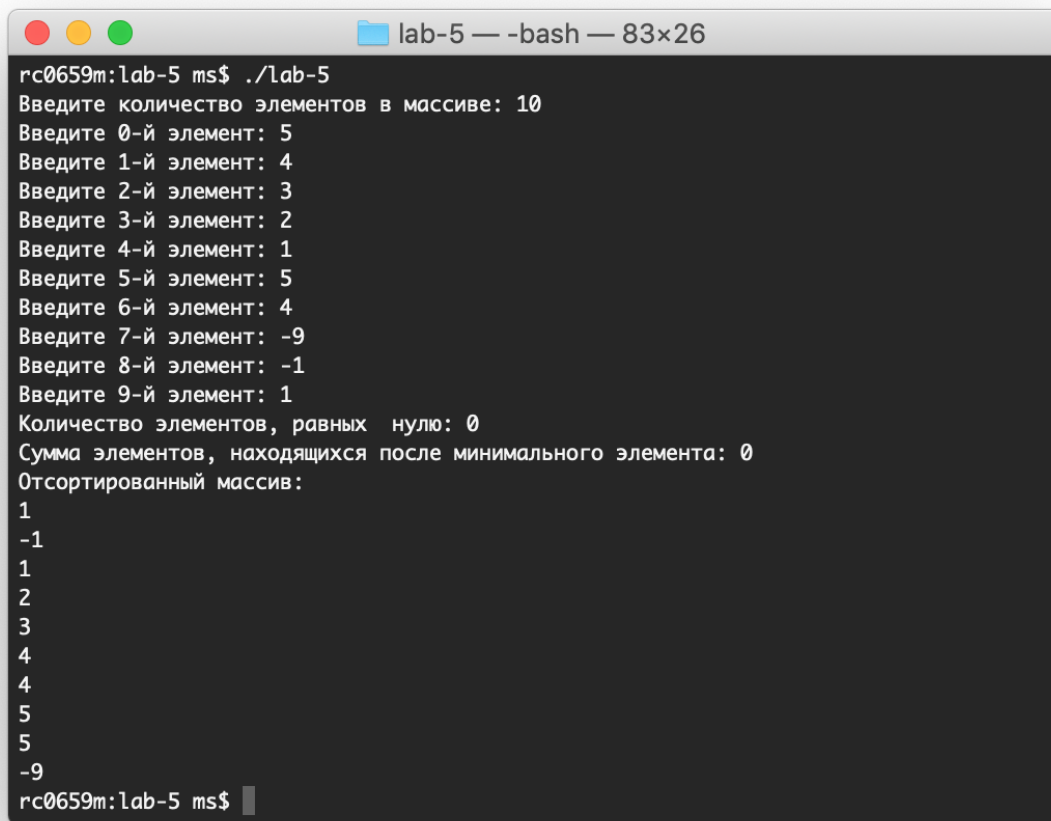
Ниже показан пример выполнения программы.



```
MacBook-Pro-Mikhail:lab-5 ms$ make lab-5
c++ lab-5.cpp -o lab-5
MacBook-Pro-Mikhail:lab-5 ms$ ./lab-5
Введите количество элементов в массиве: 8
Введите 0-й элемент: 30
Введите 1-й элемент: 20
Введите 2-й элемент: 5
Введите 3-й элемент: -1
Введите 4-й элемент: 35
Введите 5-й элемент: 0
Введите 6-й элемент: 5
Введите 7-й элемент: 0
Количество элементов, равных нулю: 2
Сумма элементов, находящихся после минимального элемента: 40
Отсортированный массив:
0
0
-1
5
5
20
30
35
MacBook-Pro-Mikhail:lab-5 ms$
```

Видно, что результаты расчётов совпадают с тестовыми данными.

Для тестовых данных 5 4 3 2 1 5 4 -9 -1 1:



```
rc0659m:lab-5 ms$ ./lab-5
Введите количество элементов в массиве: 10
Введите 0-й элемент: 5
Введите 1-й элемент: 4
Введите 2-й элемент: 3
Введите 3-й элемент: 2
Введите 4-й элемент: 1
Введите 5-й элемент: 5
Введите 6-й элемент: 4
Введите 7-й элемент: -9
Введите 8-й элемент: -1
Введите 9-й элемент: 1
Количество элементов, равных нулю: 0
Сумма элементов, находящихся после минимального элемента: 0
Отсортированный массив:
1
-1
1
2
3
4
4
5
5
-9
rc0659m:lab-5 ms$
```

Для этих тестовых данных,

- в массиве нет элементов, равных 0
- сумма элементов, находящихся после минимального элемента (-9) – элементов -1, 1 равна 0
- элементы массива должны быть отсортированы по модулям их значений

Как видно из результата работы программы, для данных тестовых данных программа работает правильно.

6. Анализ результатов и выводы

К достоинствам программы можно отнести:

- Программа выполняет поставленную задачу и работает без ошибок (для корректных тестовых данных).
- Задание реализовано в виде отдельных функций.

Из недостатков можно отметить:

- Не производится проверка входных данных.
- Алгоритм пузырьковой сортировки не является самым оптимальным из существующих алгоритмов сортировки