МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №9

по дисциплине: Основы программирования
тема: «Использование функций
при решении задач на одномерные массивы»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
Притчин Иван Сергеевич
Черников Сергей Викторович

Код-ревьер: ст. группы ПВ-223
Голуцкий Георгий Юрьевич

Белгород 2022 г.

Лабораторная работа № 9

Вариант №1

Содержание отчёта:

- Тема лабораторной работы.
- Номер варианта.
- Цель лабораторной работы.
- Решения задач.
- Вывод по работе.

Тема лабораторной работы: Использование функций при решении задач на одномерные массивы

Цель лабораторной работы: получение навыков решении задач на одномерные массивы.

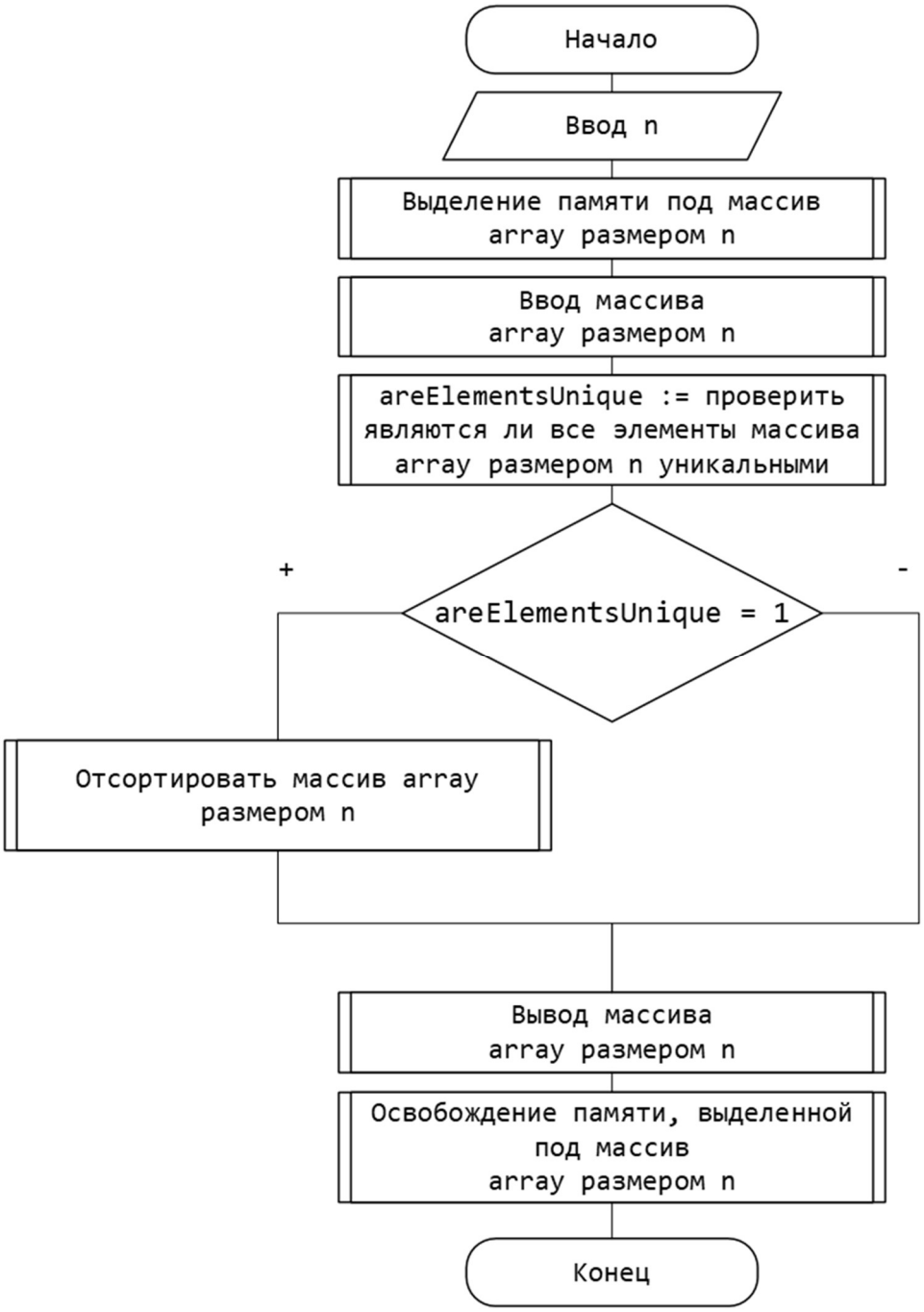
Решения задач:

1. **Если возможно, то упорядочить данный массив размера n по убыванию, иначе массив оставить без изменения.**

Входные данные	Выходные данные
1 1 2 4	4 2 1
2 4 2 4	4 2 4
3 1 3 1 4	1 3 1 4
4 4 2 3 1	4 3 2 1

Подзадачи:

1. Ввод массива.
2. Проверка элементов массива на наличие повторяющихся.
    a. Поиск позиции элемента в массиве.
        i. Обмен значений по адресам
3. Сортировка элементов массива.
4. Вывод массива.


Начало
Ввод n
Выделение памяти под массив array размером n
Ввод массива array размером n
areElementsUnique := проверить являются ли все элементы массива array размером n уникальными
areElementsUnique = 1
+
-
Отсортировать массив array размером n
Вывод массива array размером n
Освобождение памяти, выделенной под массив array размером n
Конец

main.c

```c
#include "../../libs/alg/alg.h"

int main() {
    size_t n;
    scanf("%zu", &n);

    int *array = (int *) malloc(n * sizeof(int));

    inputArray(array, n);

    if (areElementsUnique(array, n))
        sortInsertion(array, n);

    outputArray(array, n);

    free(array);

    return 0;
}
```

1func.c

```c
#include "../alg.h"

// вводит в массив array с консоли size элементов
void inputArray(int * const array, size_t size) {
    for (size_t i = 0; i < size; i++) {
        scanf("%d", &array[i]);
    }
}

// выводит массив array размером size в консоль
void outputArray(const int * const array, size_t size) {
    for (size_t i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }

    printf("\n");
}

// возвращает индекс первого с левого конца элемента массива array размером size,
// равный searchElement
size_t linearSearch(const int * const array, size_t size, int searchElement) {
    size_t index = 0;

    while (index < size && array[index] != searchElement)
        index++;

    return index;
}

// возвращает "истина" если все элементы массива array размера size уникальны
bool areElementsUnique(const int * const array, size_t size) {
    for (size_t i = 0; i < size - 1; i++)
```

```
        if (linearSearch(array + i + 1, size - i - 1, array[i]) != (size - i - 1))
            return false;

    return true;
}

// сортирует элементы массива array размером size по невозрастанию
void sortInsertion(int * const array, size_t size) {
    for (size_t i = 1; i < size; i++) {
        int currentElement = array[i];
        size_t j = i;

        while (j > 0 && array[j - 1] < currentElement) {
            intSwap(array + j, array + j - 1);
            j--;
        }

        array[j] = currentElement;
    }
}

// обменивает значения по адресам a и b
void intSwap(int *a, int *b) {
    int t = *a;
    *a = *b;
    *b = t;
}
```

```
3
1 2 4
4 2 1

Process finished with exit code 0
```

```
4
1 3 1 4
1 3 1 4

Process finished with exit code 0
```

```
3
4 2 4
4 2 4

Process finished with exit code 0
```

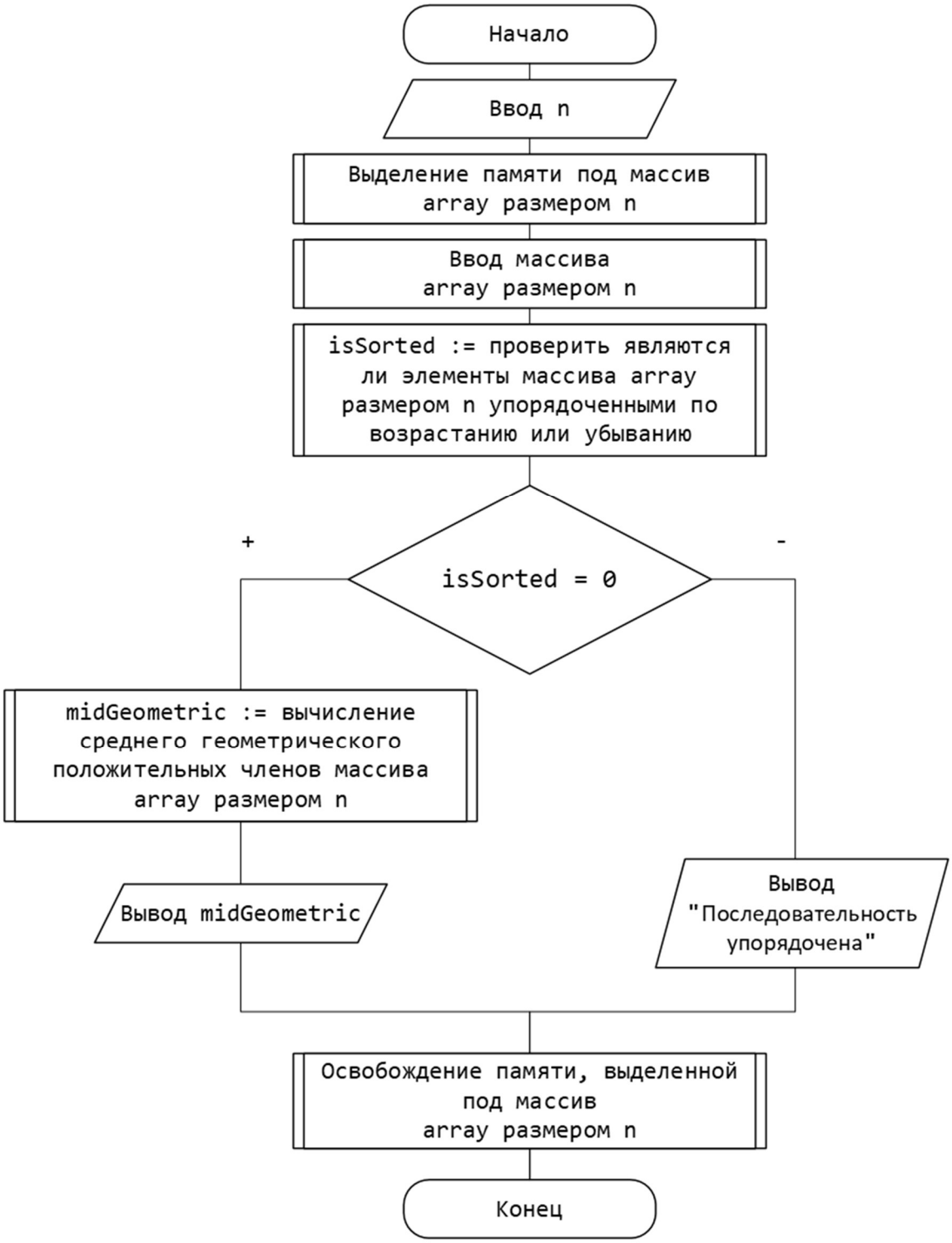
```
4
4 2 3 1
4 3 2 1

Process finished with exit code 0
```

2. Если данная целочисленная последовательность не упорядочена ни по неубыванию, ни по невозрастанию, найти среднее геометрическое положительных членов.

Входные данные	Выходные данные
1 4 1 2	2
2 9 1 3 3 0	3
3 2 -1 -1 0	2
4 -1 -2 -1	0
5 2 4 3	2.884499
6 -1 -1 -1	“Последовательность упорядочена”
7 1 2 4	“Последовательность упорядочена”
8 4 2 2	“Последовательность упорядочена”

Подзадачи:

1. Ввод массива.
2. Проверка являются ли элементы последовательности не упорядочены по возрастанию или по убыванию.
    a. Проверка являются ли элементы последовательности упорядочены по возрастанию.
    b. Проверка являются ли элементы последовательности упорядочены по убыванию
3. Нахождение среднего геометрического.


Начало
Ввод n
Выделение памяти под массив array размером n
Ввод массива array размером n
isSorted := проверить являются ли элементы массива array размером n упорядоченными по возрастанию или убыванию
isSorted = 0
+
-
midGeometric := вычисление среднего геометрического положительных членов массива array размером n
Вывод midGeometric
Вывод "Последовательность упорядочена"
Освобождение памяти, выделенной под массив array размером n
Конец

main.c

```c
#include "../../libs/alg/alg.h"

int main() {
    size_t n;
    scanf("%zu", &n);

    int *array = (int *) malloc(n * sizeof(int));

    inputArray(array, n);

    if (!isSorted(array, n)) {
        double midGeometricNum = midGeometric(array, n);

        printf("%.6lf", midGeometricNum);
    } else
        printf("Последовательность упорядочена");

    free(array);

    return 0;
}
```

3func.c

```c
#include "../alg.h"

// вводит в массив array с консоли size элементов
void inputArray(int * const array, size_t size) {
    for (size_t i = 0; i < size; i++) {
        scanf("%d", &array[i]);
    }
}

// возвращает "истина" если массив упорядочен по невозрастанию или неубыванию,
// иначе - "ложь"
bool isSorted(const int * const array, size_t arraySize) {
    return isSortedNotDec(array, arraySize) || isSortedNotInc(array, arraySize);
}

// возвращает "истина" если массив упорядочен по неубыванию, иначе - "ложь"
bool isSortedNotDec(const int * const array, size_t arraySize) {
    for (size_t i = 0; i < arraySize - 1; i++)
        if (array[i] > array[i + 1])
            return false;

    return true;
}

// возвращает "истина" если массив упорядочен по невозрастанию, иначе - "ложь"
bool isSortedNotInc(const int * const array, size_t arraySize) {
    for (size_t i = 0; i < arraySize - 1; i++)
        if (array[i] < array[i + 1])
            return false;

    return true;
}
```

```cpp
// возвращает среднее геометрическое положительных элементов массива
// array размером arraySize
double midGeometric(const int * const array, size_t arraySize) {
    size_t elementsAmount = 0;
    double mult = 1;
    for (size_t i = 0; i < arraySize; i++)
        if (array[i] > 0) {
            mult *= array[i];
            elementsAmount++;
        }

    return elementsAmount > 0 ? pow(mult, 1.0 / elementsAmount) : 0;
}
```

```
3
4 1 2
2.000000
Process finished with exit code 0
```

```
4
2 -1 -1 0
2.000000
Process finished with exit code 0
```

```
3
2 4 3
2.884499
Process finished with exit code 0
```

```
3
1 2 4
Последовательность упорядочена
Process finished with exit code 0
```

```
5
9 1 3 3 0
3.000000
Process finished with exit code 0
```

```
3
-1 -2 -1
0.000000
Process finished with exit code 0
```

```
3
-1 -1 -1
Последовательность упорядочена
Process finished with exit code 0
```

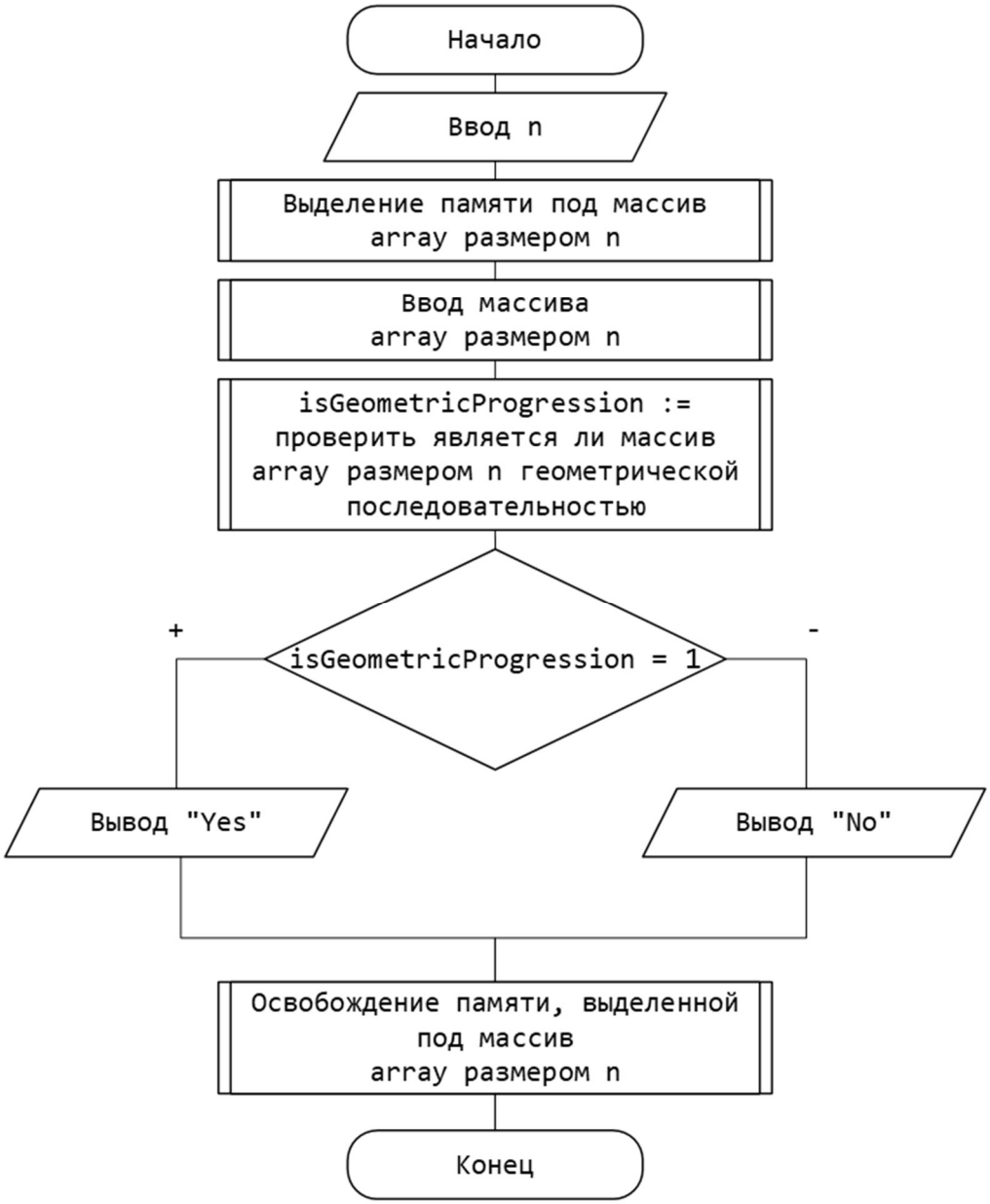
```
3
4 2 2
Последовательность упорядочена
Process finished with exit code 0
```

3. Определить, можно ли, переставив члены данной целочисленной последовательности длины n ($n > 1$), получить геометрическую прогрессию с знаменателем q ($|q| \neq 1$). Разрешимое допущение: знаменатель прогрессии – целое число

Входные данные	Выходные данные
1 4 1 2	“Yes”
2 -1 -4 -16 2 8	“Yes”
3 1 2 5	“No”
4 1 1	“No”
5 0 1	“No”
6 1 3 0	“No”
7 1 2 -4 -8 -16	“No”
8 1 1 1 1 -1	“No”
9 0 0 0	“No”
10 1 2 4 4 4 4 8	“No”
11 1 -1 1	“No”

Подзадачи:

1. Ввод массива.
2. Определение, является ли последовательность упорядоченной.
    a. Нахождение числа в последовательности
    b. Отсортировать массив в соответствии с изменяемым условием.
    c. Сравнение модулей чисел.



main.c

```c
#include "../../libs/alg/alg.h"

int main() {
    size_t n;
    scanf("%zu", &n);

    int *array = (int *) malloc(n * sizeof(int));

    inputArray(array, n);

    if(isGeometricProgression(array, n))
        printf("YES");
    else
        printf("NO");

    free(array);

    return 0;
}
```

9func.c

```c
#include "../alg.h"

#define EPS 0.0000001

// возвращает "истину" если a равен b с точностью EPS = 0.0000001
int fcompare(double a, double b) {
    return fabs(a - b) < EPS;
}

// вводит в массив array с консоли size элементов
void inputArray(int * const array, size_t size) {
    for (size_t i = 0; i < size; i++) {
        scanf("%d", &array[i]);
    }
}

// возвращает "истина" если модуль числа a больше модуля числа b, иначе - "ложь"
bool compareByModulus(int a, int b) {
    return abs(a) < abs(b);
}

// сортирует массив array размером arraySize в соответствии с условием comparator
void insertionSortByComparator(int *const array, size_t arraySize,
                               bool (*comparator)(int, int)) {
    for (size_t i = 1; i < arraySize; i++) {
        size_t j = i;

        while (j > 0 && comparator(array[j], array[j - 1])) {
            intSwap(array + j, array + j - 1);
            j--;
        }
    }
}

// обменивает значения по адресам a и b
void intSwap(int *a, int *b) {
    int t = *a;
    *a = *b;
    *b = t;
}

// возвращает индекс первого с левого конца элемента массива array размером size,
// равный searchElement
size_t linearSearch(const int * const array, size_t size, int searchElement) {
    size_t index = 0;

    while (index < size && array[index] != searchElement)
        index++;

    return index;
}

// возвращает "истина" если последовательность array размером arraySize > 1
// является геометрической прогрессией, иначе - "ложь"
bool isGeometricProgression(const int * const array, size_t arraySize) {
    if (linearSearch(array, arraySize, 0) == arraySize - 1)
```

```
        return false;

    assert(arraySize > 1);

    int *sortedArray = malloc(sizeof(array[0]) * arraySize);

    memcpy(sortedArray, array, arraySize * sizeof(array[0]));
    insertionSortByComparator(sortedArray, arraySize, compareByModulus);

    double q = (1.0 * sortedArray[1]) / sortedArray[0];

    if (fcompare(fabs(q), 1))
        return false;

    for (size_t i = 1; i < arraySize - 1; i++)
        if (!fcompare(q, (1.0 * sortedArray[i + 1]) / sortedArray[i])) {
            free(sortedArray);
            return false;
        }

    free(sortedArray);
    return true;
}
```

```
3
4 1 2
YES
Process finished with exit code 0
3
1 2 5
NO
Process finished with exit code 0
2
0 1
YES
Process finished with exit code 0
5
1 2 -4 -8 -16
NO
Process finished with exit code 0
3
0 0 0
NO
Process finished with exit code 0
3
1 -1 1
NO
Process finished with exit code 0
```

```
5
-1 -4 -16 2 8
YES
Process finished with exit code 0
2
1 1
NO
Process finished with exit code 0
3
1 3 0
NO
Process finished with exit code 0
5
1 1 1 1 -1
NO
Process finished with exit code 0
7
1 2 4 4 4 4 8
NO
Process finished with exit code 0
1
42
Assertion failed: arraySize > 1, f
func.c, line 51

Process finished with exit code 3
```

Вывод: в ходе выполнения лабораторной работы получены навыки решения задач на одномерные массивы.