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.

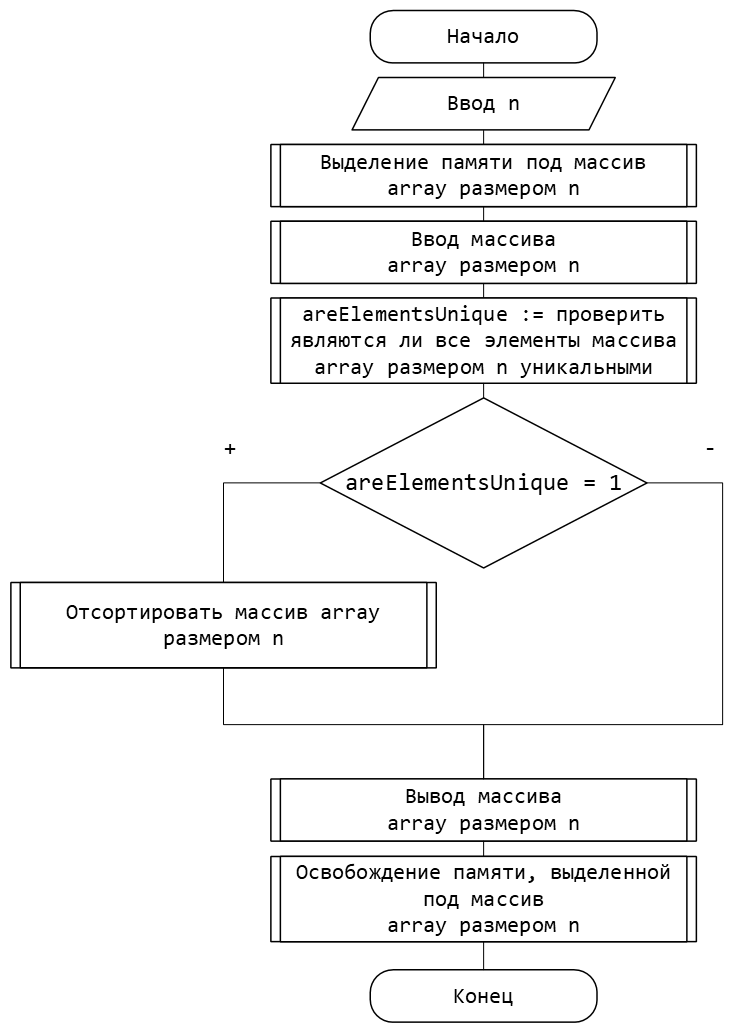
**Решения задач:**

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.
   1. Поиск позиции элемента в массиве.
      1. Обмен значений по адресам
3. Сортировка элементов массива.
4. Вывод массива.



main.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

#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;  
}

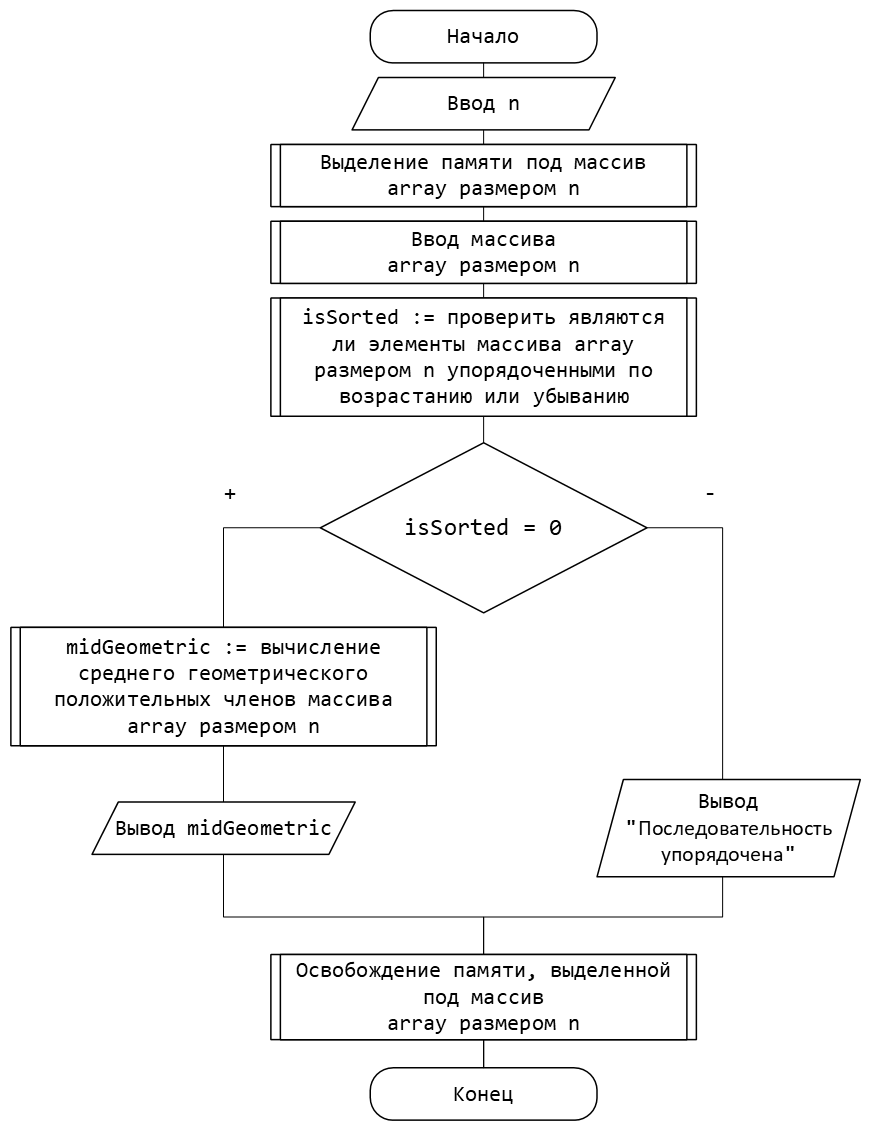
|  |  |
| --- | --- |
|  |  |
|  |  |

**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.
   1. Проверка являются ли элементы последовательности упорядочены по возрастанию.
   2. Проверка являются ли элементы последовательности упорядочены по убыванию
3. Нахождение среднего геометрического.



main.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

#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**;  
}  
  
*// 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;  
}

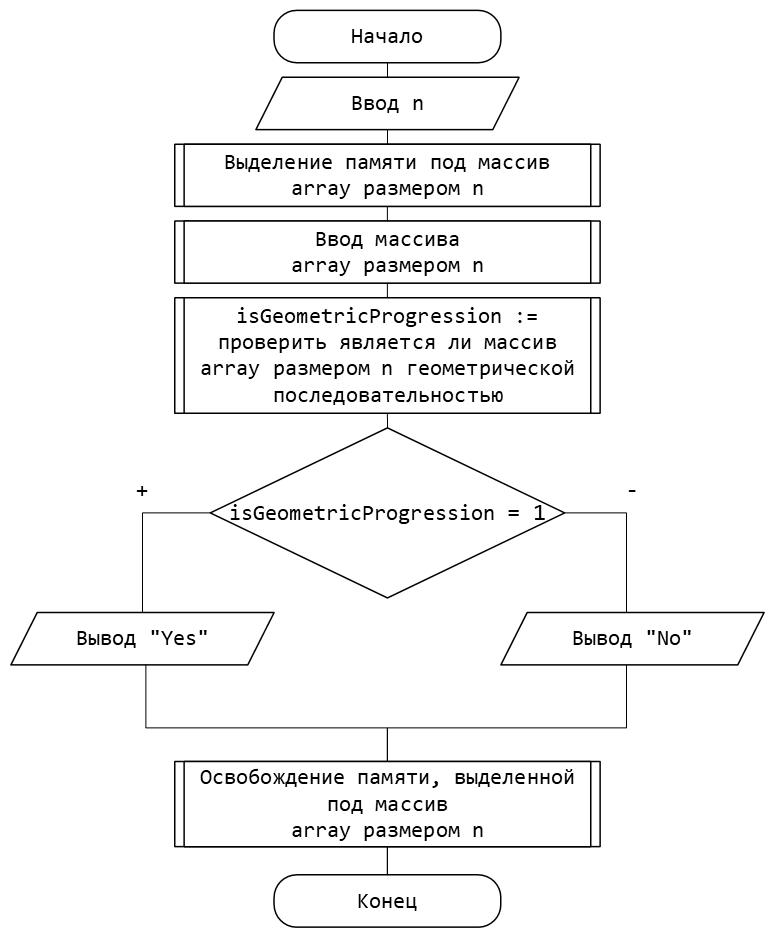
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

**3. Определить, можно ли, переставив члены данной целочисленной последовательности длины 𝑛 (𝑛 > 1), получить геометрическую прогрессию c знаменателем 𝑞 (|𝑞| ̸= 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.
   1. Нахождение числа в последовательности
   2. Отсортировать массив в соответствии с изменяемым условием.
   3. Сравнение модулей чисел.



main.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

#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**;  
}

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

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