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: студент группы ПВ-223
Мелехов Артём Дмитриевич

Проверили:
ст. преп. Притчин Иван Сергеевич
асс. Черников Сергей Викторович

Белгород 2022 г.

Лабораторная работа №9 «Использование функций при решении задач на одномерные массивы»

Цель работы: получение навыков решения задач на одномерные массивы.

Содержание отчета:

Тема лабораторной работы

Цель лабораторной работы

Решения задач. Для каждой задачи указаны:

– Условие задачи.

– Тестовые данные.

–Выделение подзадач.

– Исходный код с комментариями (спецификацией) к функциям.

Вывод.

Задача №1.

Условие:

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

Тестовые данные:

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

Выделение подзадач:

1. Ввод *n*. *n* – размер массива;
2. Ввод элементов массива размера *n*;
3. Проверка массива на уникальность его элементов;
4. Упорядочивание массива при его неуникальности;
5. Освобождение памяти выделенной под массив.

Код программы:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>
#include <stdbool.h>
```

```c
//Позволяет ввести массив a размера n
void inputArray(int* a, size_t n)
{
    for (size_t i = 0; i < n; i++)
        scanf("%d", &a[i]);
}

//Проверка массива a размера n на уникальность
bool uniquenessOfTheArray(int* a, size_t n)
{
    for (size_t i = 0; i < n - 1; i++)
        for (size_t j = i + 1; j < n; j++)
            if (a[i] == a[j])
                return false;

    return true;
}

//Обмен значениями двух чисел
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

//Упорядочивание массива a размера n по убыванию,
//если это возможно
void orderingAnArray(int *a, size_t n)
{
    if (uniquenessOfTheArray(a, n))
        for (size_t i = 0; i < n - 1; i++)
            for (size_t j = i + 1; j < n; j++)
                if (a[i] < a[j])
                    swap(&a[i], &a[j]);
}

//Выводит массив a размера n
void outputArray(int *a, size_t n)
{
    for (size_t i = 0; i < n; i++)
        printf("%d ", a[i]);

    printf("\n");
}

int main(void) {

    int n;
    scanf("%d", &n);

    int *a = (int *) malloc(sizeof(int) * n);
    inputArray(a, n);

    orderingAnArray(a, n);

    outputArray(a, n);

    free(a);

    return 0;
}
```

Задача №2.

Условие:

Дана целочисленная последовательность. Упорядочить по неубыванию часть последовательности, заключённую между первым вхождением максимального значения и последним вхождением минимального.

Тестовые данные:

Входные данные	Выходные данные
10 3 2 1 0	10 1 2 3 0
0 3 2 1 10	0 1 2 3 10
10 5 4 4 7 8 10	10 4 5 4 7 8 10 10

Выделение подзадач:

1. Ввод *n*. *n* – размер массива;
2. Ввод элементов массива размера *n*;
3. Поиск первого максимального элемента в массиве;
4. Поиск последнего минимального элемента в массиве;
5. Обратная сортировка выбором элементов массива между первым максимумом и последним минимумом;
6. Освобождение памяти выделенной под массив.

Код программы:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>
#include <limits.h>

//Позволяет ввести массив a размера n
void inputArray(int* a, size_t n)
{
    for (size_t i = 0; i < n; i++)
        scanf("%d", &a[i]);
}

//Обменивает 2 числа значениями
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}
```

```
//Обратная сортировка выбором массива a размера n
void reverseSortingByChoice(int* a, const size_t initialIndex,
                                    const size_t finalIndex)
{
    for (size_t i = initialIndex; i < finalIndex - 1; i++)
    {
        int min = INT_MAX;
        int index = 0;

        for (size_t j = i; j < finalIndex; j++)
            if (a[j] < min)
            {
                min = a[j];
                index = j;
            }

        if (i != index)
            swap(&a[i], &a[index]);
    }
}

//Возвращает индекс последнего минимального числа
//из массива a размера n
size_t searchForTheLastMinIndex(int* a, size_t n)
{
    size_t minIndex;
    long long minNumber = LLONG_MAX;

    for (size_t i = 0; i < n; i++)
    {
        if (a[i] <= minNumber)
        {
            minIndex = i;
            minNumber = a[i];
        }
    }

    return minIndex;
}

//Возвращает индекс первого максимального числа
//из массива a размера n
size_t searchForTheFirstMaxIndex(int* a, size_t n)
{
    size_t maxIndex;
    long long maxNumber = LLONG_MIN;

    for (size_t i = 0; i < n; i++)
    {
        if (a[i] > maxNumber)
        {
            maxIndex = i;
            maxNumber = a[i];
        }
    }

    return maxIndex;
}

//Поиск минимума из двух числовых значений индексов
size_t min2(size_t a, size_t b)
{
    return a < b ? a : b;
}
```

```
//Поиск максимума из двух числовых значений индексов
size_t max2(size_t a, size_t b)
{
    return a > b ? a : b;
}

//Сортировка элементов массива a размера n

//между первым максимумом и последним минимумом
void sortingBySelectionBetweenFirstMaxAndLastMin
                                (int* a, size_t n)
{
    size_t firstMax = searchForTheFirstMaxIndex(a, n);
    size_t lastMin = searchForTheLastMinIndex(a, n);

    size_t initialIndex = min2(firstMax, lastMin);
    size_t finalIndex = max2(firstMax, lastMin);

    reverseSortingByChoice(a, ++initialIndex,
                                  finalIndex);
}

//Выводит массив a размера n
void outputArray(int* a, size_t n)
{
    for (size_t i = 0; i < n; i++)
        printf("%d ", a[i]);

    printf("\n");
}

int main(void)
{
    int n;
    scanf("%d", &n);

    int* a = (int*) malloc(sizeof(int) * n);
    inputArray(a, n);

    sortingBySelectionBetweenFirstMaxAndLastMin(a, n);

    outputArray(a, n);

    free(a);

    return 0;
}
```

Задача №3.

Условие:

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

Тестовые данные:

Входные данные	Выходные данные
4 1 2	2
9 1 3 3 0	3
2 -1 -1 0	2
-1 -2 -1	0
2 4 3	2.884499
-1 -1 -1	Последовательность упорядочена
1 2 4	Последовательность упорядочена
4 2 2	Последовательность упорядочена

Выделение подзадач:

1. Ввод *n*. *n* – размер массива;
2. Ввод элементов массива размера *n*;
3. Проверка массива на уникальность;
4. Подсчёт среднего геометрического элементов массива при его неупорядоченности;
5. Освобождение памяти выделенной под массив.

Код программы:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <malloc.h>

//Позволяет ввести массив a размера n
void inputArray(int* a, size_t n)
{
    for (size_t i = 0; i < n; i++)
        scanf("%d", &a[i]);
}

//Проверка массива a размера n на невозрастание
bool nonIncreasingSequence(int* a, const size_t n)
{
    for (int i = 0; i < n - 1; i++)
        if (a[i] < a[i + 1])
            return false;

    return true;
}

//Проверка массива a размера n на неубывание
bool nonDecreasingSequence(int* a, const size_t n)
{
    for (int i = 0; i < n - 1; i++)
        if (a[i] > a[i + 1])
            return false;

    return true;
}

//Проверка массива a размера n на упорядоченность
bool orderingOfTheArray(int* a, size_t n)
{
    return nonIncreasingSequence(a, n)
        || nonDecreasingSequence(a, n);
}
```

```c
//Подсчёт среднего геометрического
//всех положительных
//элементов неупорядоченного массива a размера n
double theGeometricMeanOfThePositiveElementsOfTheArray
(int* a, size_t n)
{
    if (!orderingOfTheArray(a, n))
    {
        double multiply = 1.0;
        int power = 0;

        for (int i = 0; i < n; i++)
            if (a[i] > 0)
            {
                multiply *= a[i];
                power++;
            }

        if ((multiply - 1.0) < 0.000001)
            return 0;

        return pow(multiply, 1.0 / power);
    }

    return -1;
}

int main(void)
{
    int n;
    scanf("%d", &n);

    int* a = (int*)malloc(sizeof(int) * n);
    inputArray(a, n);

    double answer =
        theGeometricMeanOfThePositiveElementsOfTheArray
        (a, n);

    printf(answer == -1 ? "The sequence is ordered"
        : "%lf", answer);

    free(a);

    return 0;
}
```

Задача №4.

Условие:

Если число x встречается в данной целочисленной последовательности, то упорядочить по неубыванию часть последовательности после первого вхождения x.

Тестовые данные:

Входные данные	Выходные данные
16 8 4 2 1 x = 4	16 8 4 1 2
16 8 4 2 1 x = 16	16 1 2 4 8
16 8 4 2 1 x = 1	16 8 4 2 1
16 8 4 2 1 x = 3	16 8 4 2 1

Выделение подзадач:

1. Ввод n. n – размер массива;
2. Ввод элемента после которого будет произведена сортировка (x);
3. Ввод элементов массива размера n;
4. Поиск первого вхождения элемента x;
5. Реверсивная сортировка выбором всех элементов после x;
6. Освобождение памяти выделенной под массив.

Код программы:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>

//Позволяет ввести массив a размера n
void inputArray(int* a, size_t n)
{
    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]);
}

//Возвращает индекс первого вхождения элемента x
//в массиве a размера n
size_t searchingForTheIndexOfAnElement(int* a,
                  const size_t n, const int x)
{
    for (int i = 0; i < n; i++)
        if (a[i] == x)
            return i;

    return -1;
}

//Обменивает два значения
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}
```

```c
//Обратная сортировка выбором массива a размера n

//начиная с элемента x
void reverseSortingByChoiceFromX(int* a,
       const size_t n, const int indexX)
{
    for (int i = indexX + 1; i < n; i++)
    {
        int min = a[i];
        int index = i;

        for (int j = i + 1; j < n; j++)
            if (a[j] < min)
            {
                min = a[j];
                index = j;
            }

        if (i != index)
            swap(&a[i], &a[index]);
    }
}

//Сортировка элементов массива a размера n
//после первого вхождения числа x
void sortingItemsAfterX(int* a, const size_t n,
                                    const int x)
{
    size_t indexX = searchingForTheIndexOfAnElement
                                          (a, n, x);

    if (indexX != -1)
        reverseSortingByChoiceFromX(a, n, indexX);
}

//Выводит массив a размера n
void outputArray(int* a, size_t n)
{
    for (size_t i = 0; i < n; i++)
        printf("%d ", a[i]);

    printf("\n");
}
```

```
int main(void)

{
    printf("Input n: ");
    int n;
    scanf("%d", &n);

    printf("Input x: ");
    int x;
    scanf("%d", &x);


    printf("Input array: ");
    int* a = (int*) malloc(sizeof(int) * n);
    inputArray(a, n);

    sortingItemsAfterX(a, n, x);

    printf("Result: ");
    outputArray(a, n);

    free(a);

    return 0;
}
```

Задача №5.

Условие:

Даны две последовательности. Получить упорядоченную по невозрастанию последовательность состоящую из тех членов первой последовательности, которых нет во второй.

Тестовые данные:

Входные данные	Выходные данные
a = {1, 2, 4} b = {4}	c = {2, 1}
a = {1, 2, 2, 4} b = {1, 4}	c = {2, 2}
a = {1, 2, 2, 4} b = {3}	c = {4, 2, 2, 1}
a = {1, 3, 3, 4, 6, 8, 8, 9, 10, 12, 12, 13, 15, 15} b = {0, 3, 5, 5, 8, 9, 9, 11, 14, 14}	c = {1, 4, 6, 10, 12, 12, 13, 15, 15}

Выделение подзадач:

1. Ввод *sizeA*. *sizeA* – размер массива;
2. Ввод элементов массива размера *sizeA*;
3. Ввод *sizeA*. *sizeA* – размер массива;
4. Ввод элементов массива размера *sizeA*;
5. Реверсивная сортировка выбором двух массивов;
6. Сохранение элементов из первого массива, которых нет во втором в новый массив;
7. Освобождение памяти выделенной под массивы.

Код программы:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>
#include <stdbool.h>

//Позволяет ввести массив a размера n
void inputArray(int* a, size_t n)
{
    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]);
}

//Обменивает два значения
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

//Обратная сортировка выбором массива a размера n
void reverseSortingByChoice(int* a, const size_t n)
{
    for (int i = 0; i < n; i++)
    {
        int min = a[i];
        int index = i;

        for (int j = i + 1; j < n; j++)
            if (a[j] < min)
            {
                min = a[j];
                index = j;
            }

        if (i != index)
            swap(&a[i], &a[index]);
    }
}
```

```c
//Замолняет массив с уникальными элементами
//из массивов a и b размерами sizeA и sizeB
//соответсвенно
void uniqueElementsOf2Arrays(int* a, size_t sizeA,
    int* b, size_t sizeB, int** c, size_t* sizeC)
{
    int maxSize = sizeA + sizeB;
    int sizeCount = 0;
    *c = (int*) malloc(sizeof(int) * maxSize);

    for (int i = 0; i < sizeA; i++)
    {
        bool isUnique = false;

        for (int j = 0; j < sizeB; j++)
            if (a[i] == b[j])
            {
                isUnique = true;
                break;
            }

        if (!isUnique)
            (*c)[sizeCount++] = a[i];

    }

    *c = (int*) realloc(*c, sizeof(int) * sizeCount);
    *sizeC = sizeCount;
}

//Выводит массив a размера n
void outputArray(int* a, size_t n)
{
    for (size_t i = 0; i < n; i++)
        printf("%d ", a[i]);

    printf("\n");
}
```

```c
int main(void)
{
    printf("Enter the number of elements in the 1st array: ");
    int sizeA;
    scanf("%d", &sizeA);

    printf("Enter the 1st array: ");
    int* a = (int*)malloc(sizeof(int) * sizeA);
    inputArray(a, sizeA);

    printf("Enter the number of elements in the 2nd array: ");
    int sizeB;
    scanf("%d", &sizeB);

    printf("Enter the 2nd array: ");
    int* b = (int*)malloc(sizeof(int) * sizeB);
    inputArray(b, sizeB);

    reverseSortingByChoice(a, sizeA);
    reverseSortingByChoice(b, sizeB);

    size_t sizeC;
    int* c;
    uniqueElementsOf2Arrays(a, sizeA, b, sizeB,
        &c, &sizeC);

    printf("Result: ");
    outputArray(c, sizeC);

    free(a);
    free(b);
    free(c);

    return 0;
}
```

Задача №6.

Условие:

Дана целочисленная последовательность, содержащая как положительные, так и отрицательные числа. Упорядочить последовательность следующим образом: сначала идут отрицательные числа, упорядоченные по невозрастанию, потом положительные, упорядоченные по неубыванию.

Тестовые данные:

Входные данные	Выходные данные
3 2 1 1 -4 -5 -6	-4 -5 -6 1 1 2 3
-3 -2 -1 0 1 2 3 4	-1 -2 -3 0 1 2 3 4
1 2 3 4 5	1 2 3 4 5

Выделение подзадач:

1. Ввод *sizeA*. *sizeA* – размер массива;
2. Ввод элементов массива размера *sizeA*;
3. Сортировка массива выбором;
4. Получение обратного порядка элементов массива;
5. Получение обратного порядка отрицательных элементов массива;
6. Освобождение памяти выделенной под массивы.

Код программы:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>

//Позволяет ввести массив a размера n
void inputArray(int* a, size_t n)
{
    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]);
}

//Обменивает два значения
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

//Сортировка массива a размера n выбором
void sortingByChoice(int* a, const size_t n)
{
    for (int i = 0; i < n; i++)
    {
        int max = a[i];
        int index = i;

        for (int j = i + 1; j < n; j++)
            if (a[j] > max)
            {
                max = a[j];
                index = j;
            }

        if (i != index)
            swap(&a[i], &a[index]);
    }
}

//Возвращает индекс первого неотрицательного
//элемента массива a размера n
int getIndexFirsrtNonNegativeNumber(int* a,
                                    size_t n)
{
    int firstNonNegativeIndex = 0;

    while (a[firstNonNegativeIndex] < 0)
        firstNonNegativeIndex++;

    return firstNonNegativeIndex;
}
```

```c
//Изменение порядка элементов массива a размера n
//на обратный
void reverseArray(int* a, const size_t n)
{
    for (size_t i = 0, j = n - 1; i < j; i++, j--)
        swap(&a[i], &a[j]);
}

//Изменяет порядок отрицательных чисел
//отсортированного массива a размера n на обратный
void reverseNegativeValuesInArray(int* a, size_t n)
{
    if (a[0] < 0)
    {
        int firstNonNegativeNumber =
            getIndexFirsrtNonNegativeNumber(a, n);

        reverseArray(a, firstNonNegativeNumber);
    }
}

//Выводит массив a размера n
void outputArray(int* a, size_t n)
{
    for (size_t i = 0; i < n; i++)
        printf("%d ", a[i]);

    printf("\n");
}

int main(void)
{
    printf("Enter the number of elements in array: ");
    int sizeA;
    scanf("%d", &sizeA);

    printf("Enter array: ");
    int* a = (int*) malloc(sizeof(int) * sizeA);
    inputArray(a, sizeA);

    sortingByChoice(a, sizeA);
    reverseArray(a, sizeA);

    reverseNegativeValuesInArray(a, sizeA);

    printf("Result: ");
    outputArray(a, sizeA);

    free(a);

    return 0;
}
```

Задача №7.

Условие:

Дана целочисленная последовательность и целое число x. Определить, есть ли x среди членов последовательности, и если нет, то найти члены последовательности, ближайшие к x снизу и сверху.

Тестовые данные:

Входные данные	Выходные данные
1 3 6 2 5 x = 6	*x* – элемент последовательности
1 3 6 2 5 x = 4	3 5
1 3 6 2 5 x = 0	-∞ 1
1 3 6 2 5 x = 7	6 ∞
1 1 5 5 5 x = 3	1 5

Выделение подзадач:

1. Ввод *sizeA*. *sizeA* – размер массива;
2. Ввод элементов массива размера *sizeA*;
3. Ввод элемента *x*. *x* – целое число относительно которого будет происходить поиск ближайших чисел;
4. Вывод максимально близких по значению к *x* чисел;
5. Освобождение памяти выделенной под массив.

Код программы:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>
#include <limits.h>

//Структура описывающая 2 максимальноблизких к x
//по значению элемента массива
typedef struct
{
    int maxLeft;
    int minRight;
} minToX;

//Позволяет ввести массив a размера n
void inputArray(int* a, size_t n)
{
    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]);
}

//Возвращает максимальное из двух значений
int max2(int a, int b)
{
    return a > b ? a : b;
}

//Возвращает минимальное из двух значений
int min2(int a, int b)
{
    return a < b ? a : b;
}


//Выводит максимально близкие к значению x
//элементы из массива a размера n
minToX theNumbersClosestsToXInTheArray(int* a,
                  const size_t n, const int x)
```

```
{
    minToX numbers = { INT_MIN, INT_MAX };

    for (int i = 0; i < n; i++)
    {
        if (a[i] == x)
            return (minToX) { x, x };
        else if (a[i] < x)
            numbers.maxLeft = max2(numbers.maxLeft, a[i]);
        else
            numbers.minRight = min2(numbers.minRight, a[i]);
    }

    return numbers;
}

int main(void)
{
    printf("Enter the number of elements in array: ");
    int sizeA;
    scanf("%d", &sizeA);

    printf("Enter array: ");
    int* a = (int*)malloc(sizeof(int) * sizeA);
    inputArray(a, sizeA);

    printf("Enter x: ");
    int x;
    scanf("%d", &x);

    minToX numbers = theNumbersClosestsToXInTheArray
                                        (a, sizeA, x);

    if (numbers.maxLeft == numbers.minRight)
        printf("x is the element of the sequence\n");
    else
    {
        printf(numbers.maxLeft == INT_MIN ?
            "- infinity : " : "%d : ", numbers.maxLeft);

        printf(numbers.minRight == INT_MAX ?
            "infinity\n" : "%d\n", numbers.minRight);
    }

    free(a);

    return 0;
}
```

Задача №8.

Условие:

Дана целочисленная последовательность. Получить массив из уникальных элементов последовательности.

Тестовые данные:

Входные данные	Выходные данные
1 2 4 1 2	4
1 2 3 4 5	1 2 3 4 5
1 1 1 1 1	Последовательность пуста

Выделение подзадач:

1. Ввод *sizeA*. *sizeA* – размер массива;
2. Ввод элементов массива размера *sizeA*;
3. Сортировка массива выбором;
4. Заполнение новго массива уникальными элементами первого;
5. Освобождение памяти выделенной под массивы.

Код программы:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>

//Позволяет ввести массив a размера n
void inputArray(int* a, size_t n)
{
    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]);
}

//Обменивает два значения
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

//Сортировка массива a размера n выбором
void sortingByChoice(int* a, const size_t n)
{
    for (int i = 0; i < n; i++)
    {
        int max = a[i];
        int index = i;

        for (int j = i + 1; j < n; j++)
            if (a[j] > max)
            {
                max = a[j];
                index = j;
            }

        if (i != index)
            swap(&a[i], &a[index]);
    }
}


//Заполняет массив b уникальными элементами из
//массива a размера n
void uniqueArrayElements(int* a, const size_t sizeA,
    int** b, size_t* sizeB)
{
    int maxSize = 1;
    int sizeCount = 0;
    *b = (int*)malloc(sizeof(int) * maxSize);

    int blackList = a[sizeA - 1];
    for (int i = 0; i < sizeA; i++)
    {
        if (a[i] != blackList && a[i] != a[i + 1])
        {
            if (sizeCount == maxSize)
            {
```

```
                maxSize *= 2;
                *b = (int*) realloc(*b, sizeof(int)
                    * maxSize);
            }

            (*b)[sizeCount++] = a[i];
        }

        blackList = a[i];
    }

    *b = (int*) realloc(*b, sizeof(int) * sizeCount);
    *sizeB = sizeCount;
}

//Выводит массив a размера n
void outputArray(int* a, size_t n)
{
    if (n != 0)
    {
        for (size_t i = 0; i < n; i++)
            printf("%d ", a[i]);
    }
    else
        printf("The sequence is empty");

    printf("\n");
}


int main(void)
{
    printf("Enter the number of elements in array: ");
    int sizeA;
    scanf("%d", &sizeA);

    printf("Enter array: ");
    int* a = (int*)malloc(sizeof(int) * sizeA);
    inputArray(a, sizeA);

    sortingByChoice(a, sizeA);

    size_t sizeB;
    int* b;
    uniqueArrayElements(a, sizeA, &b, &sizeB);

    printf("Result: ");
    outputArray(b, sizeB);

    free(a);
    free(b);

    return 0;
}
```

Задача №9.

Условие:

Определить, можно ли, переставив члены данной целочисленной последовательности длины n, получить геометрическую прогрессию с знаменателем q. Разрешимое допущение: знаменатель прогрессии – целое число.

Тестовые данные:

Входные данные	Выходные данные
4 1 2	Yes
-1 -4 -16 2 8	Yes
1 2 5	No
1 1	No
0 1	No
1 3 0	No
1 2 -4 -8 -16	No
1 1 1 1 -1	No
0 0 0	No
1 2 4 4 4 4 8	No
1 -1 1	No

Выделение подзадач:

1. Ввод *sizeA*. *sizeA* – размер массива;
2. Ввод элементов массива размера *sizeA*;
3. Сортировка массива выбором по модулю чисел;
4. Получение обратного порядка элементо массива;
5. Проверка массива на то, являются ли его элементы геометрической прогрессией;
6. Освобождение памяти выделенной под массивы.

Код программы:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>
#include <math.h>
#include <stdbool.h>

//Позволяет ввести массив a размера n
void inputArray(int* a, size_t n)
{
    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]);
}

//Обменивает два значения
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

//Сортировка массива a размера n выбором
void sortingByChoiceByAbs(int* a, const size_t n)
{
    for (int i = 0; i < n; i++)
    {
        int max = abs(a[i]);
        int index = i;

        for (int j = i + 1; j < n; j++)
            if (abs(a[j]) > max)
            {
                max = abs(a[j]);
                index = j;
```

```
            }

        if (i != index)
            swap(&a[i], &a[index]);
    }
}

//Изменение порядка элементов
//массива a размера n на обратный
void reverseArray(int* a, const size_t n)
{
    for (size_t i = 0, j = n - 1; i < j; i++, j--)
        swap(&a[i], &a[j]);
}
```

```
//Возвращает 'истину', если элементы

//массива a размера n можно переставить
//в геометрическую прогрессию
bool isGeometricProgression(int* a, const size_t n)
{
    if (a[0] == 0)
        return false;

    int q = a[1] / a[0];

    if (abs(q) == 1)
        return false;

    for (int i = 1; i < n - 1; i++)
    {
        int newQ = a[i + 1] / a[i];

        if (abs(newQ - q) > 0.000001
            || a[i + 1] % a[i])
            return false;
    }

    return true;
}

int main(void)
{
    printf("Enter the number of elements in array: ");
    int sizeA;
    scanf("%d", &sizeA);

    printf("Enter array: ");
    int* a = (int*)malloc(sizeof(int) * sizeA);
    inputArray(a, sizeA);

    sortingByChoiceByAbs(a, sizeA);
    reverseArray(a, sizeA);

    printf("Result: ");
    printf(isGeometricProgression(a, sizeA) ? "Yes"
        : "No");

    free(a);

    return 0;
}
```

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