**3 ЛАБОРАТОРНАЯ РАБОТА №3**

**«ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ОБРАБОТКИ ОДНОМЕРНЫХ МАССИВОВ»**

**3.1 Цель работы**

Изучение особенностей представления и обработки одномерных массивов в языках С/С++ с учетом связи указателей и массивов. Получение практических навыков реализации алгоритмов обработки одномерных массивов средствами языков С/С++. Исследование особенностей обработки одномерных динамических массивов.

**3.2 Вариант задания – 20**

Требуется оформить программу в виде функций, выполняющих:

- выделение памяти и заполнения массива (с клавиатуры);

- перемещение элементов последовательности со значением 0 в ее начало;

- упорядочивание элементов последовательности ap, ap+1, …, aq по возрастанию, используя алгоритм сортировки методом прямого обмена;

- вывод массива на экран;

- освобождение памяти.

Вызов функций осуществлять из интерактивного меню. Последовательность состоит из вещественных чисел.

**3.3 Порядок выполнения работы**

3.3.1 Разработать алгоритм решения задачи.

3.3.2 Разработать структурную схему алгоритма решения задачи.

3.3.3 Разработать программу на языке Си.

3.3.4 Разработать тестовые примеры. Выполнить тестирование используя разработанные тестовые примеры. Выполнить отладку

3.3.6 Сделать выводы по проделанной работе.

**3.4 Ход работы**

3.4.1 Для задания был разработан алгоритм решения задачи:

Для программs былf подключены библиотекf <stdio.h> для ввода и вывода информации. Далее выполняется основная программа которая начинается с фразы «main()» и в блоке этой функции между фигурными скобками текст основной программы. В функции сперва объявляются переменные, счётчики, массив. Затем выполняется бесконечный цикл в котором можно выбирать пункты меню с помощью switch. Затем осуществляется ввод количества элементов массива и непосредсвенная инициализация массива в первом пункте меню. Во втором пункте все нулевые значения массива перемещаются в начало массива. В третьем пункте меню осуществляется сортировка массива по возрастанию методом пузырька. Четвёртый пункт выводит массив на экран. Пятый пункт делает очистку памяти. Шестой пункт осуществляет выход из программы с помощью return.

3.4.2 Разработана структурная схема алгоритма решения задачи и представлена на рисунке 3.1.

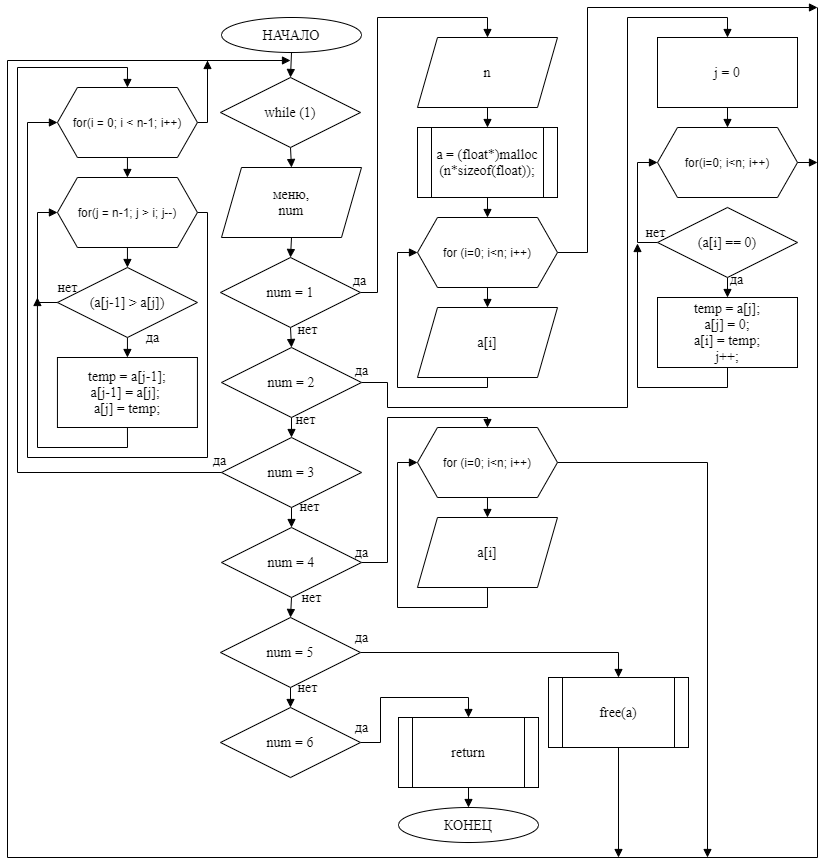


Рисунок 3.1 – Структурная схема программы

3.4.4 Написана программы на Си согласно вышеописанного алгоритма.

#include <stdio.h>

//---------------------основная программа---------------------

int main()

{

//объявление переменных

float \*a; //указатель на массив

unsigned int num; //для "меню"

int n; //кол-во элементов массива

int i,j,m; //счётчики

float temp; //переменная

//меню

printf(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \n");

printf("/ \n");

printf(" The program works with an array of real numbers \n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \n\n");

while (1)

{

printf("Choose the next action\n");

printf("1 - creating an array\n");

printf("2 - move array elements with the value '0' to the beginning of the array\n");

printf("3 - sort the array in ascending order\n");

printf("4 - show the array\n");

printf("5 - delete array\n");

printf("6 - exit the program\n");

printf(" -> "); scanf("%d",&num); printf("\n");

switch (num)

{

case 1:

{

printf("Input the number of array elements: "); scanf("%i",&n);

//выделение памяти

a = (float\*)malloc(n\*sizeof(float));

printf("Input %i array elements:\n", n);

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

scanf("%f", &a[i]);

printf("\n");

break;

}

case 2:

{

j=0;

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

if (a[i] == 0)

{

temp = a[j];

a[j] = 0;

a[i] = temp;

j++;

}

printf("All zeros are sent to the beginning of the array");

printf("\n\n");

break;

}

case 3:

{

//сортировка методом прямого обмена

for(i = 0; i < n-1; i++)

for(j = n-1; j > i; j--)

if(a[j-1] > a[j])

{

temp = a[j-1];

a[j-1] = a[j];

a[j] = temp;

}

printf("Sorting was successful");

printf("\n\n");

break;

}

case 4:

{

printf("Current array:\n");

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

printf("%f ", a[i]);

printf("\n");

break;

}

case 5:

{

free(a);

printf("The array was deleted\n");

printf("\n");

break;

}

case 6: return;

}

}

}

2.4.5 Выполнена отладка программы.

Результаты тестирования отображены на рисунках 3.2 и 3.3.

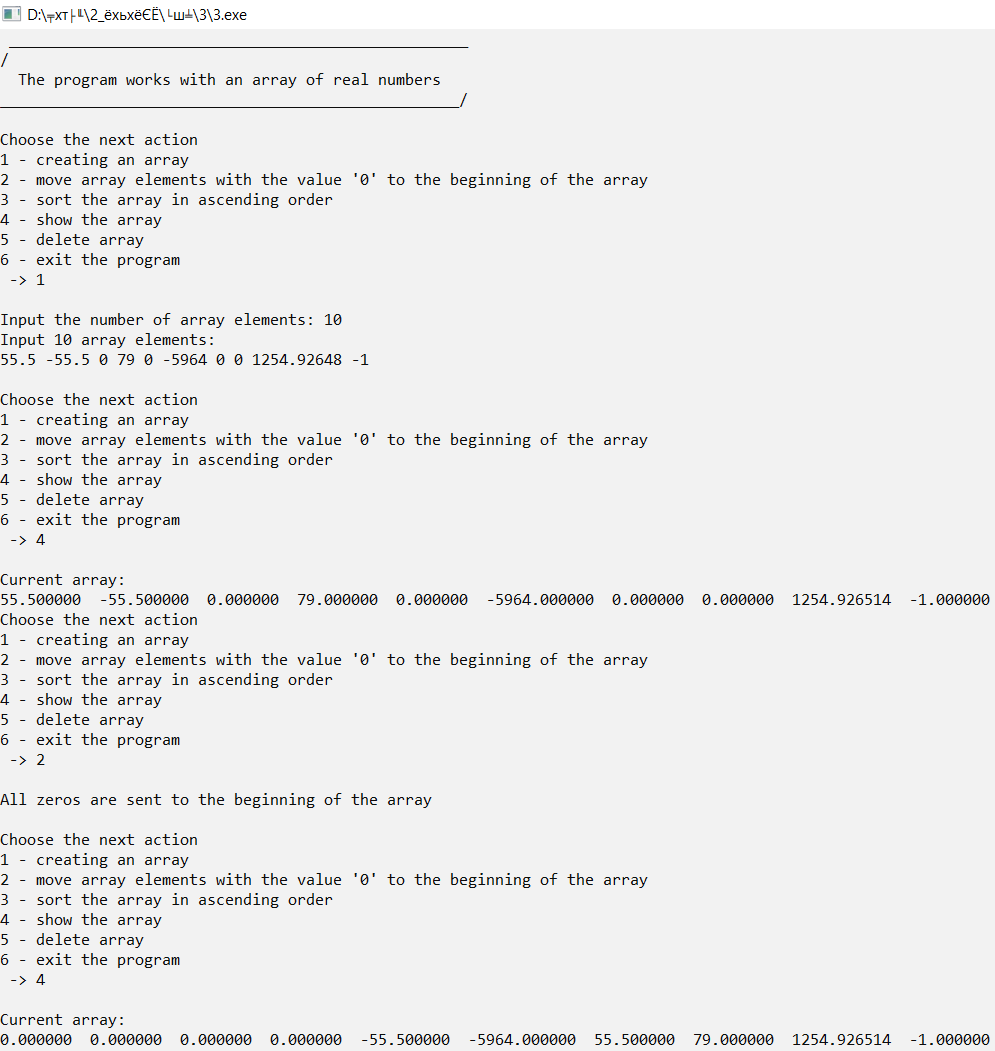


Рисунок 3.2 – Тест 1 часть

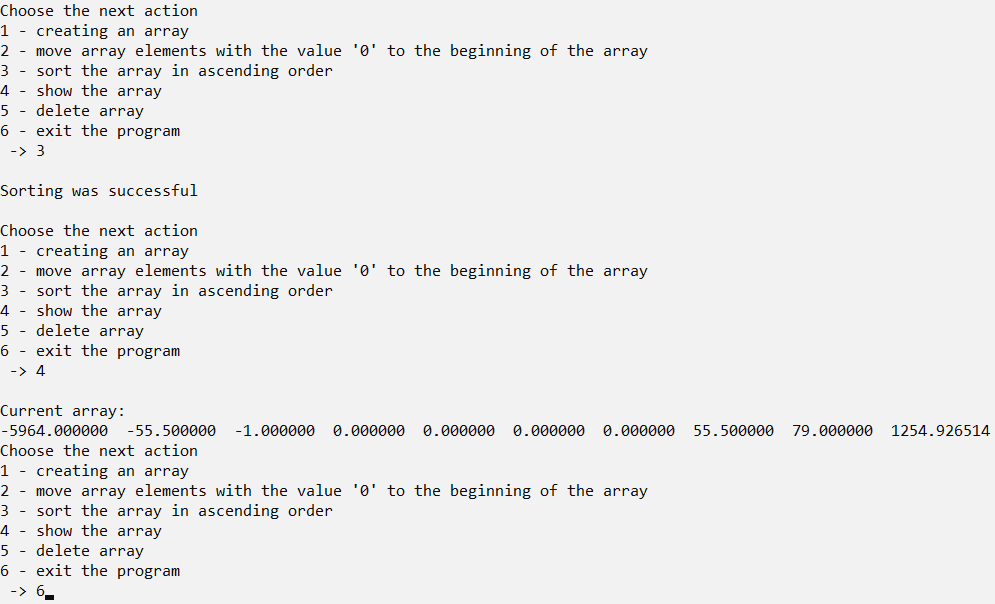


Рисунок 3.3 – Тест 2 часть

Результаты тестирования полностью соответствуют ожиданиям.

**Выводы**

В ходе выполнения данной лабораторной работы были получены навыки программирования алгоритмов обработки одномерных массивов. Был изучен новый тип данных «массив», который относится к группе структурных типов, а также способ задания этого типа данных. Были изучены различные алгоритмы при работе с массивами и способы их применения. Полученные во время разработки навыки помогут разрабатывать более сложные программы с массивами, более эффективные по времени выполнения алгоритмы.

float \*a;

a = (float\*)malloc(n\*sizeof(float));

free(a);

//сортировка методом прямого обмена

for(i = 0; i < n-1; i++)

for(j = n-1; j > i; j--)

if(a[j-1] > a[j])

{

temp = a[j-1];

a[j-1] = a[j];

a[j] = temp;

}

//сортировка методом прямого выбора

for (i = 0; i < n-1; i++)

{

min = i;

for (j = i+1; j < n; j++)

if (a[j] < a[min]) min = j;

temp = a[i]; a[i] = a[min]; a[min] = temp;

}

//сортировка методом вставки

