ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ассистент		М. А. Мурашова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Лабораторная работа 5. Обработка числовых последовательностей

по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4134К		Иванов И.В.
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

Цель работы:
Целью работы является изучение структуры данных одномерный массив.
Задание на лабораторную работу:
Задания на лабораторную работу приводятся в каждом варианте. При написании программ можно использовать как динамические, так и нединамические массивы. Размерность последних задаѐтся именованной константой.

Вариант 1

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

1. сумму отрицательных элементов массива;
2. произведение элементов массива, расположенных между максимальным и минимальным элементами.

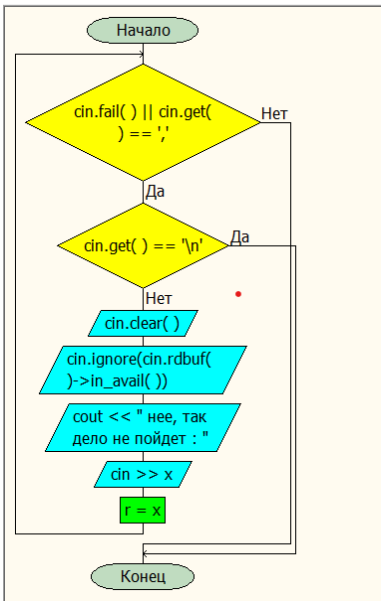
Упорядочить элементы массива по возрастанию.

Ход выполнения:

Использованные функции:

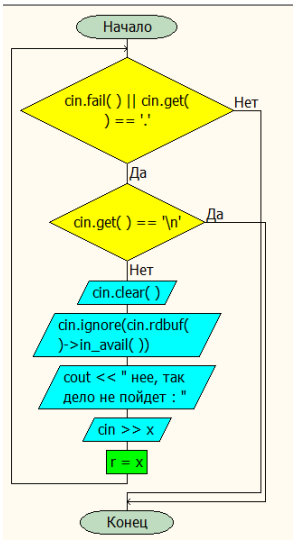
Имя: check
Назначение: проверка на запятую и на символы
Побочные эффекты: Отсутствует

Блок схема:

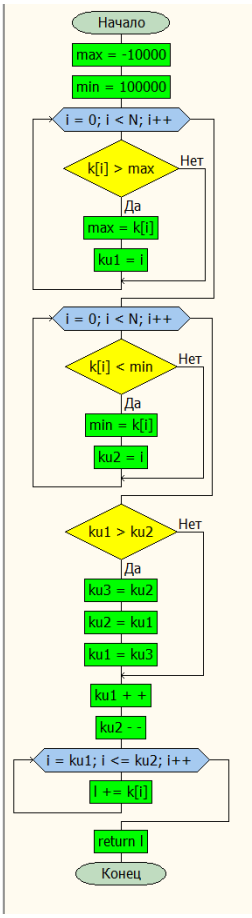


Имя: checksize
Назначение: проверка на точку и на символы
Побочные эффекты: Отсутствует

Блок-схема:



Имя: da
Назначение: подсчёт произведения элементов массива, расположенных между максимальным и минимальным элементами
Входные данные: *k
Выходные данные: l
Побочные эффекты: отсутствуют
Блок-схема:

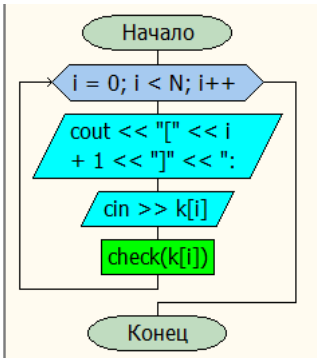


Входные данные	Выходные данные
2 3 8 6 9	**144**
1 3 4 2 6	**24**

Псевдокод:

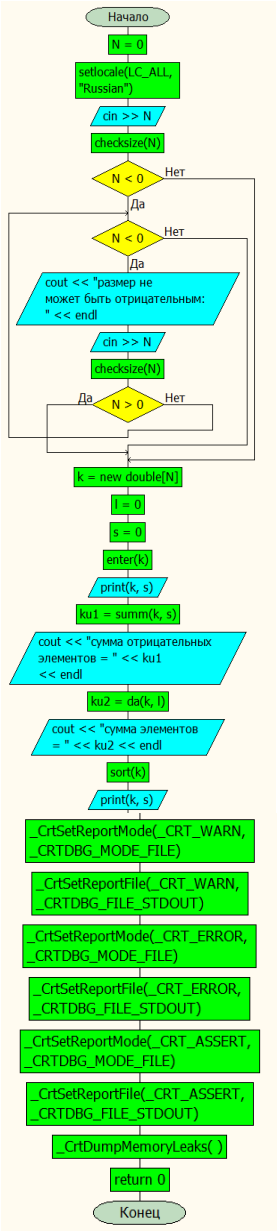
Пока i меньше N то выполняется условие
    Если k[i] больше max
    То max равно k[i]
    И ku1 равно i
Пока i меньше N то выполняется условие
    Если k[i] меньше min
    То min равно k[i]
    И ku2 равно и
Если ku1больше ku2
  То ku2 становится на место ku1
Пока i меньше или равно ku2 выполняется условие
    Суммирования k[i]

Имя: enter
Назначение: ввод элементов массива в консоли
Побочные эффекты: отсутствуют
Блок-схема:



Псевдокод:
Ввод элементов массива

Имя: main
Назначение: Создание массива и его элементов, вызов сопутствующих функций
Входные данные: N(количество элементов массива), k(массив)
Выходные данные: s(сумма отрицательных элементов массива),l(произведения элементов массива, расположенных между максимальным и минимальным элементами)
Побочные эффекты: нет
Блок-схема:

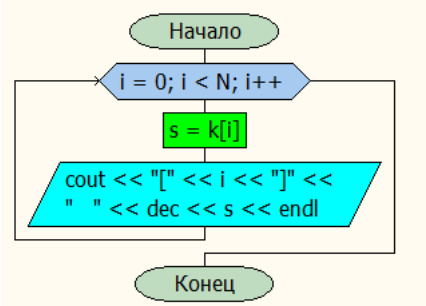


Входные данные	**Выходные данные**
5 2 5 6	**144**
1 3 6 8 3	**45**

Псевдокод:
Ввод элементов массива
Проверка выведенного значения
Вычисление произведения
Присвоение его переменной
Вывод полученного значения
Вычисление суммы
Присвоение его переменной
Вывод полученного значения
Проверка на утечку данных

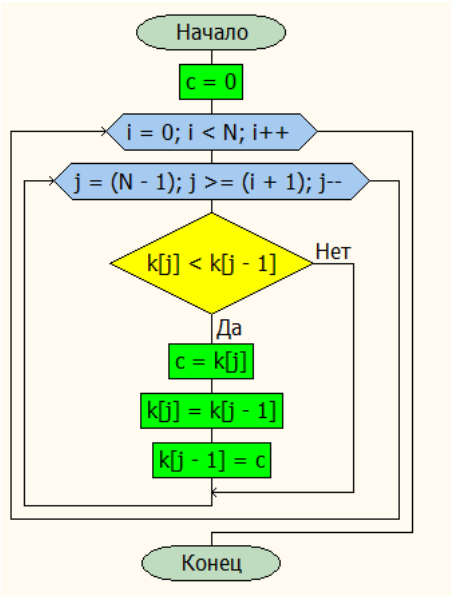
Имя: print
Назначение: вывод элементов массива в консоль
Побочные эффекты: отсутствуют

Блок-схема:



Псевдокод:
Вывод элементов массива

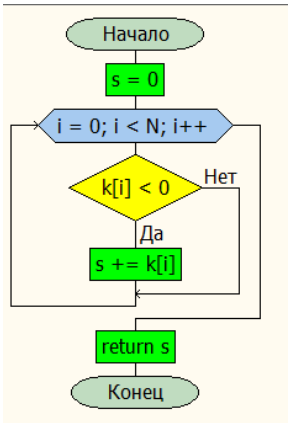
Имя: sort
Назначение: сортировка элементов по возрастанию
Входные данные: k[i]
Выходные данные: k[i]
Побочные эффекты: отсутствуют
Блок-схема:



Входные данные	**Выходные данные**
9 4 2 8 6	**2 4 6 8 9**
-4 3 8 -1 5	**-4 -1 3 5 8**

Псевдокод:
Пока i меньше N
Пока j больше i+1 выполняется условие
Если k[j] меньше k[j-1]
То происходит по парная сортировка
Иначе возвращаемся обратно в цикл for

Имя: summ
Назначение: подсчёт суммы отрицательных элементов
Входные данные: k[i]
Выходные данные: s
Побочные эффекты: отсутствуют
Блок-схема:



Входные данные	Выходные данные
9 4 2 8 6	**0**
-4 3 8 -1 5	**-5**

Псевдокод:

Пока I меньше N выполняется условие
    Если k[i] меньше 0
    То мы прибавляем к s k[i]
    Иначе заходим обратно в цикл for.

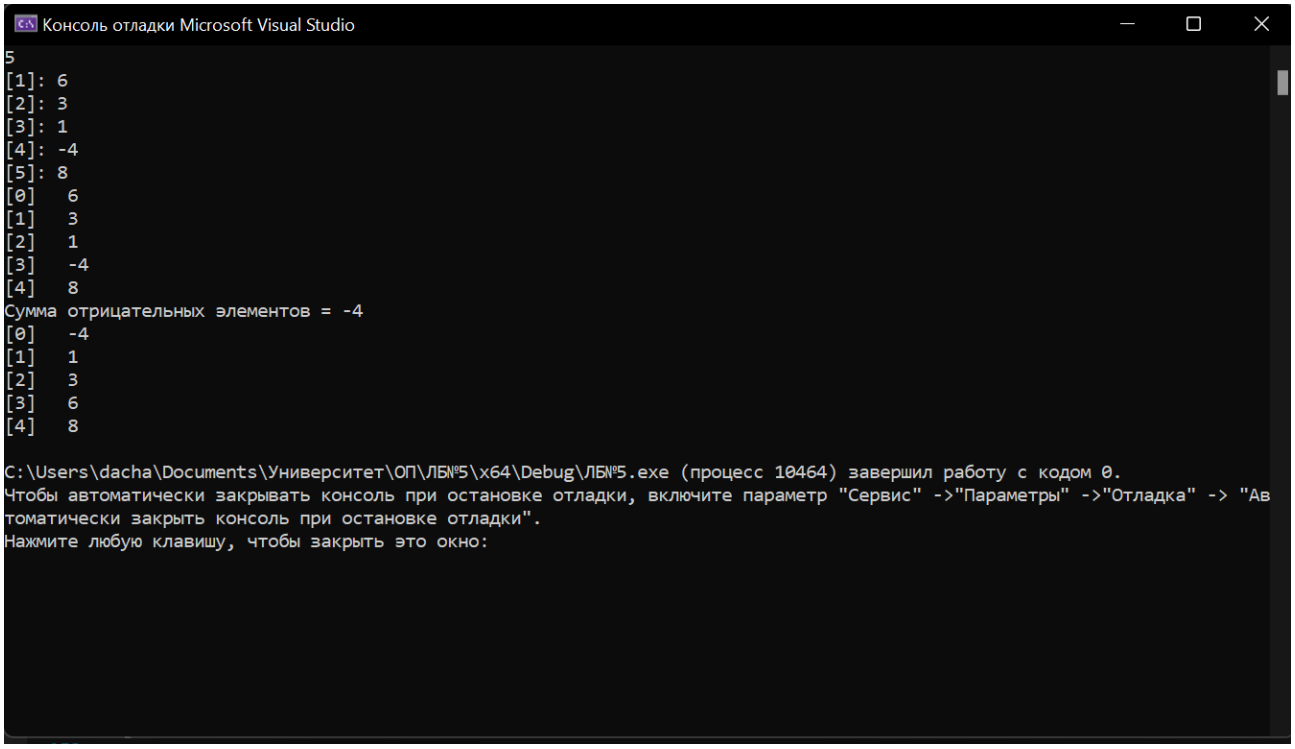
Листинг всей программы:

```
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>
#ifdef _DEBUG
#ifndef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define newDBG_NEW
#endif
#endif
#include<cmath>
#include <iostream>
#include<bitset>
using namespace std;
int N;
void checksize(int& r) {


   while (cin.fail() || cin.get() == '.')
   {


         if (cin.get() == '\n') {
            break;
         }
         double x;
         cin.clear();
         cin.ignore(cin.rdbuf()->in_avail());
         cout << " нее, так дело не пойдет : ";
         cin >> x;
         r = x;

   }
}
```

```
void check(double& r) {
    while (cin.fail() || cin.get() == ',')
    {
        if (cin.get() == '\n') {
            break;
        }
        double x;
        cin.clear();
        cin.ignore(cin.rdbuf()->in_avail());
        cout << " нее, так дело не пойдет : ";
        cin >> x;
        r = x;
    }
}
void enter(double* k) {
    for (int i = 0; i < N; i++) {
        cout << "[" << i + 1 << "]" << ": ";
        cin >> k[i]; check(k[i]);
    }
}
void print(double* k,double s) {

    for (int i = 0; i < N; i++) {
        s = k[i];
        cout << "[" << i << "]" << "    " << dec << s<< endl;

    }
}
void sort(double *k)
{

    int c = 0;
    for (int i = 0; i < N; i++) {
        for (int j = (N - 1); j >= (i + 1); j--) {
            if (k[j] < k[j - 1]) {
                c = k[j];
                k[j] = k[j - 1];
                k[j - 1] = c;
            }
        }
    }
}
int summ(double *k, int s)
{
    s = 0;
    for (int i = 0; i < N; i++)
        if (k[i] < 0) {
            s += k[i];
        }

    return s;
}
double da(double *k, int l) {
    int max, min, ku1, ku2, ku3;

    max = -10000;
    min = 100000;
    for (int i = 0; i < N; i++)
        if (k[i] > max) {
            max = k[i];
            ku1 = i;
        }


    for (int i = 0; i < N; i++)
        if (k[i] < min) {
            min = k[i];
            ku2 = i;
```

```
        }

    if (ku1 > ku2) {
        ku3 = ku2;
        ku2 = ku1;
        ku1 = ku3;
    }
    ku1++;
   ku2--;
    for (int i = ku1; i <= ku2; i++) {

            l += k[i];

    }
    return l;

}
int main() {
    int N = 0;
    setlocale(LC_ALL, "Russian");
    cin >> N; checksize(N);
    if (N < 0) {
        while (N < 0) {
            cout << "размер не может быть отрицательным: " << endl;
            cin>>N;
            checksize(N);
            if (N > 0) {
                break;
            }
            else {
                continue;
            }
        }
    }
        double* k = new double[N];
        int  ku1, ku2, ku3, l;
        l = 0;
        int s;
        s = 0;
        enter(k);
        print(k, s);
        ku1 = summ(k, s);
        cout << "сумма отрицательных элементов = " << ku1 << endl;
         ku2=da(k, l);
         cout << "сумма элементов = " << ku2 << endl;
        sort(k);
        print(k, s);
        delete[]k;
        _CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_FILE);
        _CrtSetReportFile(_CRT_WARN, _CRTDBG_FILE_STDOUT);
        _CrtSetReportMode(_CRT_ERROR, _CRTDBG_MODE_FILE);
        _CrtSetReportFile(_CRT_ERROR, _CRTDBG_FILE_STDOUT);
        _CrtSetReportMode(_CRT_ASSERT, _CRTDBG_MODE_FILE);
        _CrtSetReportFile(_CRT_ASSERT, _CRTDBG_FILE_STDOUT);
        _CrtDumpMemoryLeaks();
        return 0;

}
```

Результат работы программы:



Вывод:
Я изучил структуру данных одномерного массива.

Достоинства программы:
Скорость работы.
Проверка значений.
Наличие пользовательского интерфейса.

Недостатки программы:
Недостатков нет.