

1. Цель работы

Целью работы является изучение структуры данных одномерных массивов, научиться совершать операции над одномерным массивом.

2. Задание

Согласно варианту №3 требуется написать программу, которая в одномерном массиве, состоящем из n вещественных элементов вычислит сумму отрицательных элементов и произведение элементов массива, расположенных между минимальным и максимальным элементом массива:

- Реализовать ввод с клавиатуры пользователем значения (n);
- Реализовать проверку вводимых данных;
- Реализовать сортировку по убыванию.

3. Описание созданных функций

Для реализации задания были использованы следующие функции:

Имя: Mas().

Назначение: предназначена для вывода массива.

Входные данные:

- *arr – динамический массив;
- n – переменная, хранящая в себе значение размера массива

Выходные данные:

- cout – вывод сообщений в консоль.

Побочный эффект: отсутствует.

Тестовые данные:

n	cout
5	Arr{ 54,57,-38,50,32 }
10	Arr{ -20,95,95,-94,34,-78,38,68,35,53 }
8	Arr{ -68,-41,32,96,-76,41,43,-19 }

Прототип: void Mas(int *arr,int n, int nmax, int nmin);

Алгоритм:

- псевдокод

Вывод сообщения в консоль о том, что выводится массив данных;

Цикл, выводящий рандомно генерируемый массив заданного размера;

Вывод перехода на новую строку.

Имя: negmult().

Назначение: предназначена для нахождения и суммирования отрицательных элементов массива;

Входные данные:

- *arr – динамический массив;
- n – переменная, хранящая в себе значение размера массива;

Выходные данные:

- negmult – переменная, хранящая в себе сумму отрицательных элементов;

Побочный эффект: отсутствует.

Тестовые данные:

arr	n	negmult
Arr{54,57,-38,50,32}	5	-38
Arr{-20,95,95,-94,34,-78,38,68,35,53}	10	-192
Arr{-68,-41,32,96,-76,41,43,-19}	8	-204

Прототип: void negmult(int *arr, int n, int negnum);

Алгоритм:

- псевдокод

Цикл, перебирающий и сравнивающий с 0 каждый элемент массива;

Вывод результата на экран.

Имя: Multip().

Назначение: предназначена для нахождения минимально и максимального элементов массива, а так же для перемножения элементов стоящих между минимальным и максимальным элементом массива;

Входные данные:

- *arr – динамический массив;
- n – переменная, хранящая в себе значение размера массива;

Выходные данные:

- mult – переменная, хранящая в себе произведение элементов между минимальным и максимальным значением массива;

Побочный эффект: потеря точности при использовании типа double.

Тестовые данные:

arr	n	mult
Arr{79,59,-100,-69,-13}	5	59

Arr{46,2,0,-56,-44,-74,56,42,-14,91}	10	-32928
Arr{93,93,-81,-21,45,28,-10,-90}	8	-1.99323e ¹⁰

Прототип: void Multip(int *arr, int n, int nmax, int nmin, int ii, int jj, double mult);

Алгоритм:

- псевдокод

Цикл, перебирающий значения и сравнивающий его с 0, если значение меньше, то записываем его в переменную nmin;

Вывод значения минимального элемента;

Цикл, перебирающий значения и сравнивающий его с 0, если значение больше, то записываем его в переменную nmax;

Вывод значения минимального элемента;

Условие если индекс минимального элемента меньше чем индекс максимального элемента, то выполняем вложенный цикл по которому перемножаем значения между минимальным и максимальным элементом массива;

Условие если индекс минимального элемента больше чем индекс максимального элемента, то выполняем вложенный цикл по которому перемножаем значения между минимальным и максимальным элементом массива;

Вывод Mult в консоль.

Имя: sort().

Назначение: предназначена для сортировки массива по возрастанию;

Входные данные:

- *arr – динамический массив;
- n – переменная, хранящая в себе значение размера массива;

Выходные данные:

- arr – переменная, хранящая в себе сумму отрицательных элементов;

Побочный эффект: отсутствует.

Тестовые данные:

arr	n	arr
Arr{5,-62,95,-57,-53}	5	Arr{-62,-57,-53,5,95}
Arr{39,54,-79,18,-13,-6,57,57,82,87}	10	Arr{-79,-13,-6,18,39,54,57,57,82,87}
Arr{-37,-12,-23,15,-46,54,-92,30}	8	Arr{-92,-46,-37,-23,-12,15,30,54}

Прототип: void sort(int *arr, int n, int *Swap);

Алгоритм:

- псевдокод

Цикл с вложенным циклом, который перебирает значения массива начиная с последнего элемента сравнивая его с предыдущим и если он больше, то меняет их местами;

Цикл для вывода отсортированного массива;

Освобождение памяти занятой динамическими массивами.

Имя: sort().

Назначение: предназначена для сортировки массива по возрастанию;

Входные данные:

Выходные данные:

- n – переменная, хранящая в себе размер массива;

Побочный эффект: отсутствует.

Тестовые данные:

n
5
10
8

Прототип: int EnterN(void);

Алгоритм:

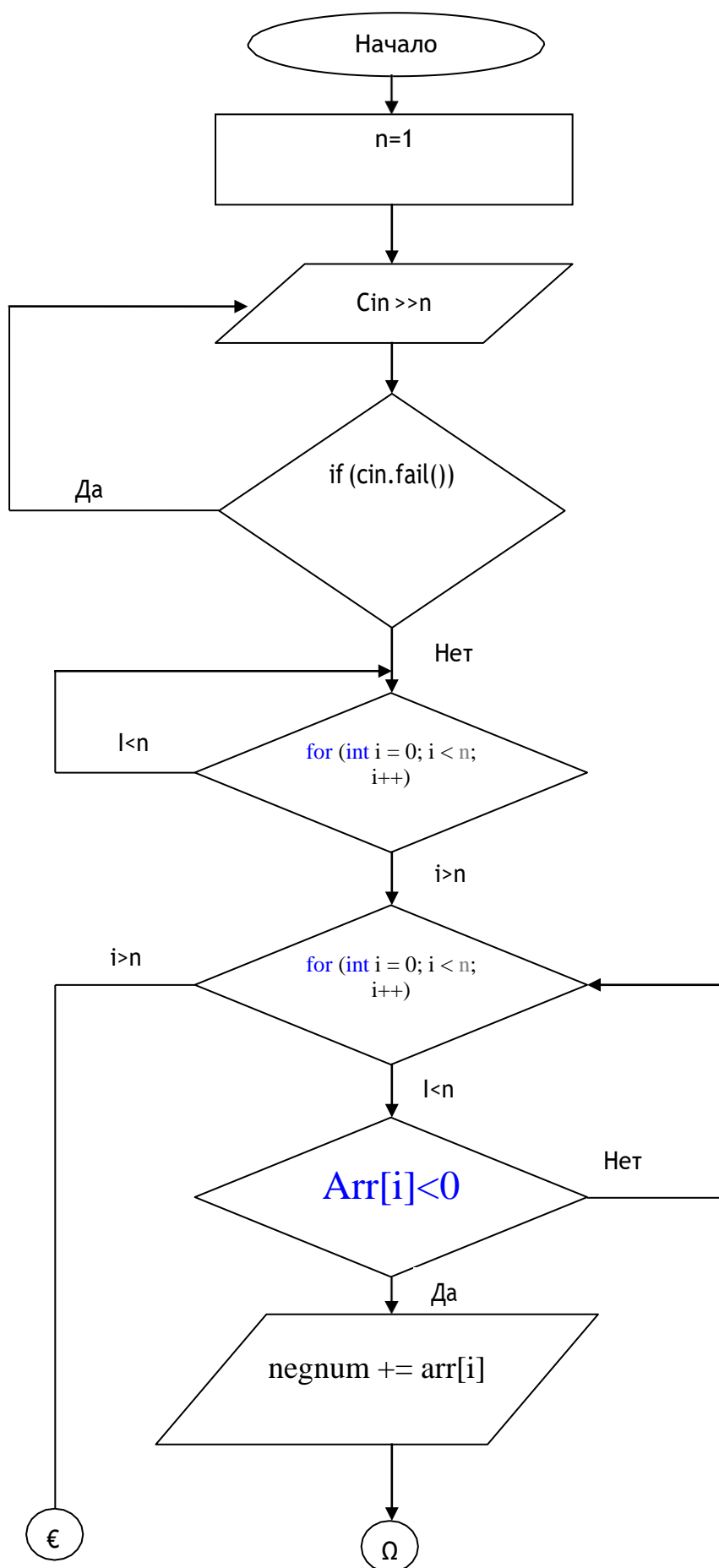
- псевдокод

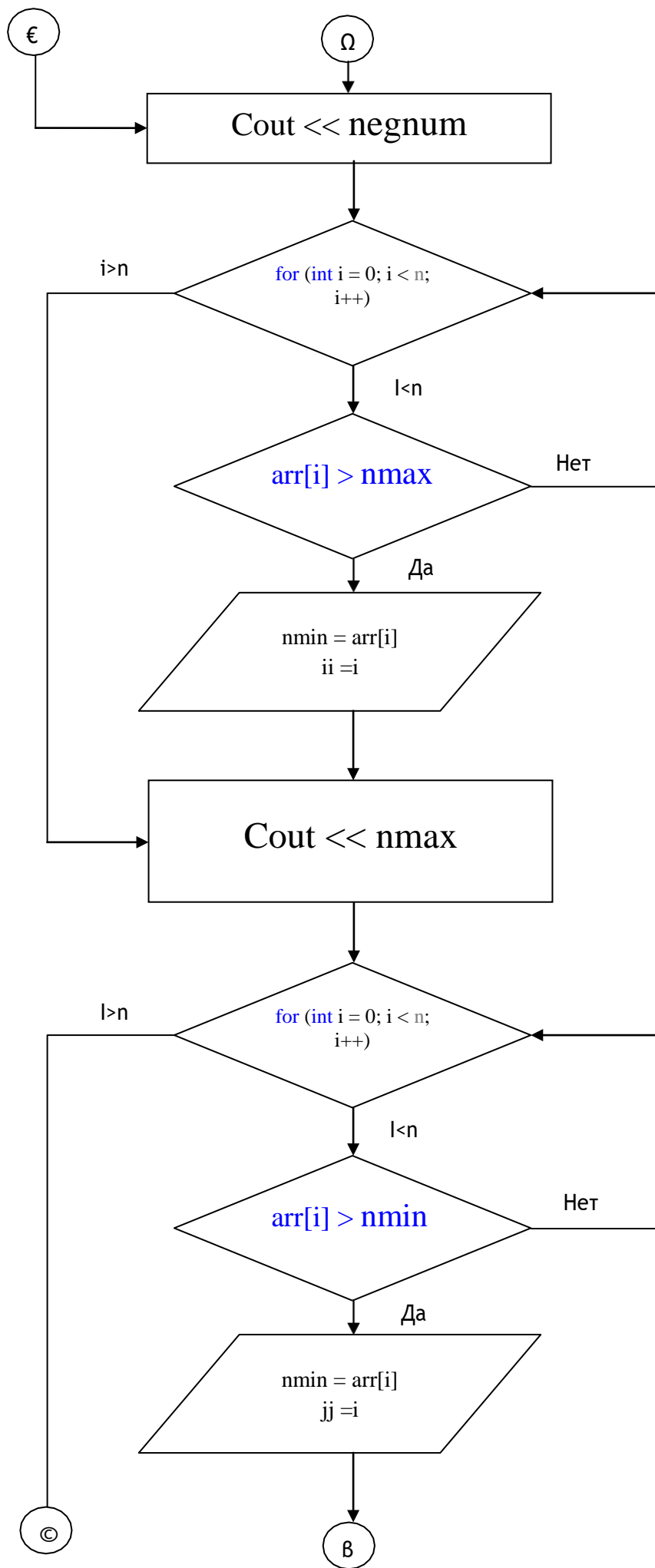
Ввод значения с клавиатуры;

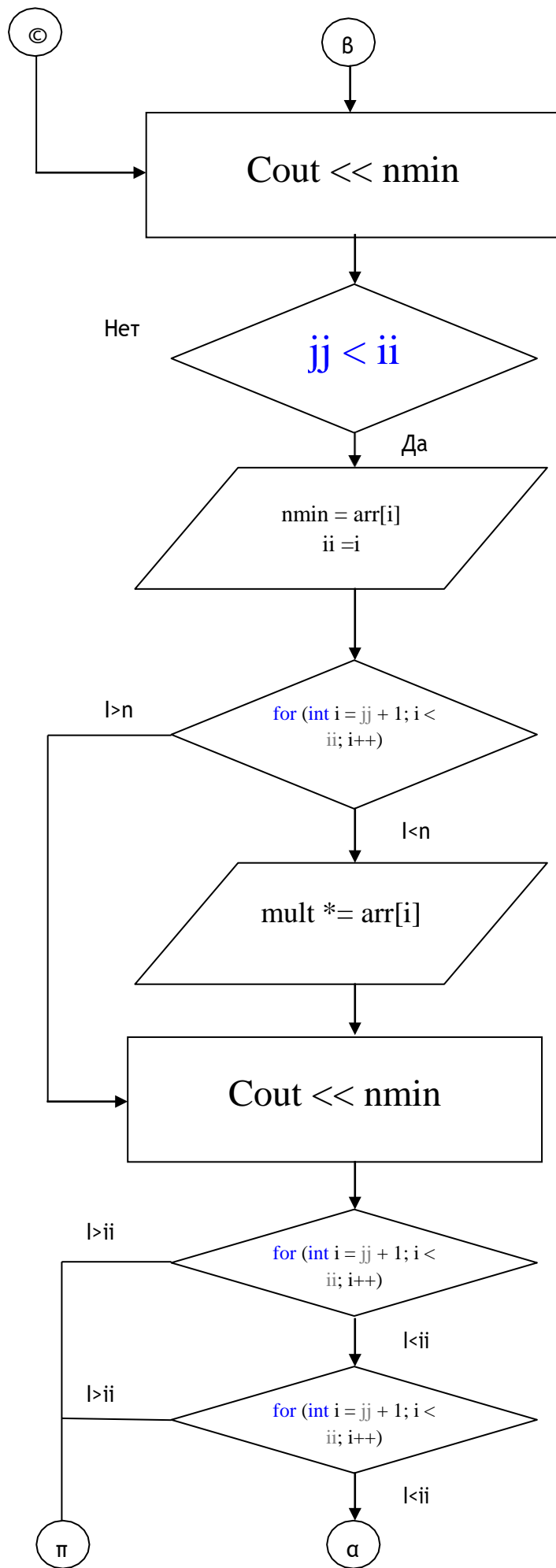
Проверка введённого значения на принадлежность к числовому значению;

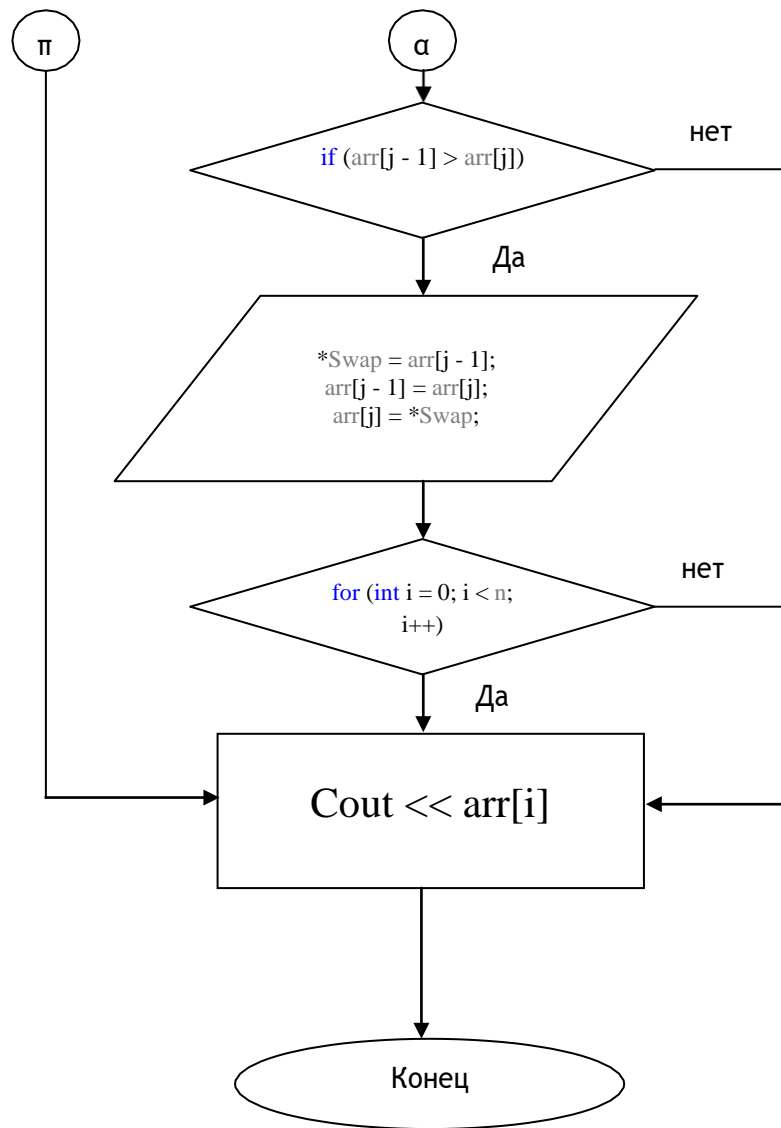
Если ввод удачный, то возвращаем переменную

Блок-схема









4. Листинг программы

```
5. #include <iostream>
6. #include <time.h>
7. #include <stdlib.h>
8. #include <conio.h>
9. #include <crtdbg.h>
10.     #define _CRTDBG_MAP_ALLOC
11.     #define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
12.     #define newDBG_NEW

13.     using namespace std; // подключение пространства имён std

14.     // функция по выводу массива
15.     void Mas(int *arr, int n, int nmax, int nmin)
16.     {
17.         srand(time(NULL));

18.         // вывод массива
19.         cout << "Массив:\n";
20.         for (int i = 0; i < n; i++)
21.         {
22.             arr[i] = rand () % 201 + (-100);
23.             cout << "m[" << i << "] = " << arr[i] << " " << endl;
24.         }
25.         cout << "\n";
26.     }

27.     //функция по нахождению произведения отрицательных элементов
28.     void negmult(int *arr, int n, int negnum )
29.     {
30.         //находим произведение отрицательных элементов
31.         for (int i = 0; i < n; i++)
32.         {
33.             if (arr[i] < 0)
34.                 negnum += arr[i];
35.         }
36.         cout << "Сумма отрицательных элементов массива равна:" << negnum << endl;
37.     }

38.     // функция по нахождению произведения элементов между мин. и макс.
    значением массива
39.     void Multip(int *arr, int n, int nmax, int nmin, int ii, int jj, double
    mult)
40.     {
41.         // находим минимальный элемент массива
42.         for (int i = 0; i < n; i++)
43.         {
```

```

44.     if (arr[i] < nmin)
45.     {
46.         nmin = arr[i];
47.         jj = i; // индекс минимального
48.     }
49. }

50.     cout << "Минимальный элемент m[" << jj << "] = " << nmin << endl;

51.     // находим максимальный элемент массива
52.     for (int i = 0; i < n; i++)
53.     {
54.         if (arr[i] > nmax)
55.         {
56.             nmax = arr[i];
57.             ii = i; // индекс максимального
58.         }
59.     }

60.     cout << "Максимальный элемент m[" << ii << "] = " << nmax << endl;

61.     // произведение элементов между минимальным и максимальным элементом
62.     if (jj < ii)
63.     {
64.         for (int i = jj + 1; i < ii; i++)
65.             mult *= arr[i];
66.     }
67.     else
68.     {
69.         for (int i = ii + 1; i < jj; i++)
70.             mult *= arr[i];
71.     }

72.     cout << "Произведение элементов: " << mult << endl;
73. }

74.     //функция сортировки массива по возрастанию (пузырьковая сортировка)
75.     void sort(int *arr, int n, int *Swap)
76.     {
77.         for (int i = 0; i < n; i++)
78.         {
79.             for (int j = n - 1; j >= i; j--)
80.             {
81.                 if (arr[j - 1] > arr[j])
82.                 {
83.                     *Swap = arr[j - 1];
84.                     arr[j - 1] = arr[j];
85.                     arr[j] = *Swap;
86.                 }

```

```

87.     }
88.     }
89.     cout << "\n";
90.     cout << "Отсортированный массив:\n";

91.     //вывод отсортированного массива
92.     for (int i = 0; i < n; i++)
93.     {
94.         cout << "m[" << i << "] = " << arr[i] << " " << endl;
95.     }
96.     delete[] arr; // очистка памяти
97.     delete[] Swap; // очистка памяти
98.     }

99.     //функция для ввода размера массива
100.    int EnterN(void)
101.    {
102.        int n;
103.        while (true)
104.        {
105.            cout << "Введите размер массива:";
106.            cin >> n;

107.            if (cin.fail())
108.            {
109.                cin.clear();
110.                cout << "Введено неверное значение.Введите значение заново!";
111.            }
112.            else
113.                break;
114.        }
115.        return n;
116.    }

117.    int main()
118.    {
119.        setlocale(LC_ALL, "Russian");

120.        int nmax = 0 , nmin = 0;
121.        int n = EnterN();
122.        int ii = 0, jj = 0; // переменные для хранения индексов значений
123.        double mult = 1; // переменная для хранения произведения
124.        int negnum = 0; //переменная для хранения суммы отрицательных элементов
125.        int *Swap = new int[n];
126.        int *arr = new int [n]; // основной динамический массив

127.        Mas(arr,n,nmax,nmin);
128.        negmult(arr, n, negnum);
129.        Multip(arr, n, nmin, nmax, ii, jj, mult);

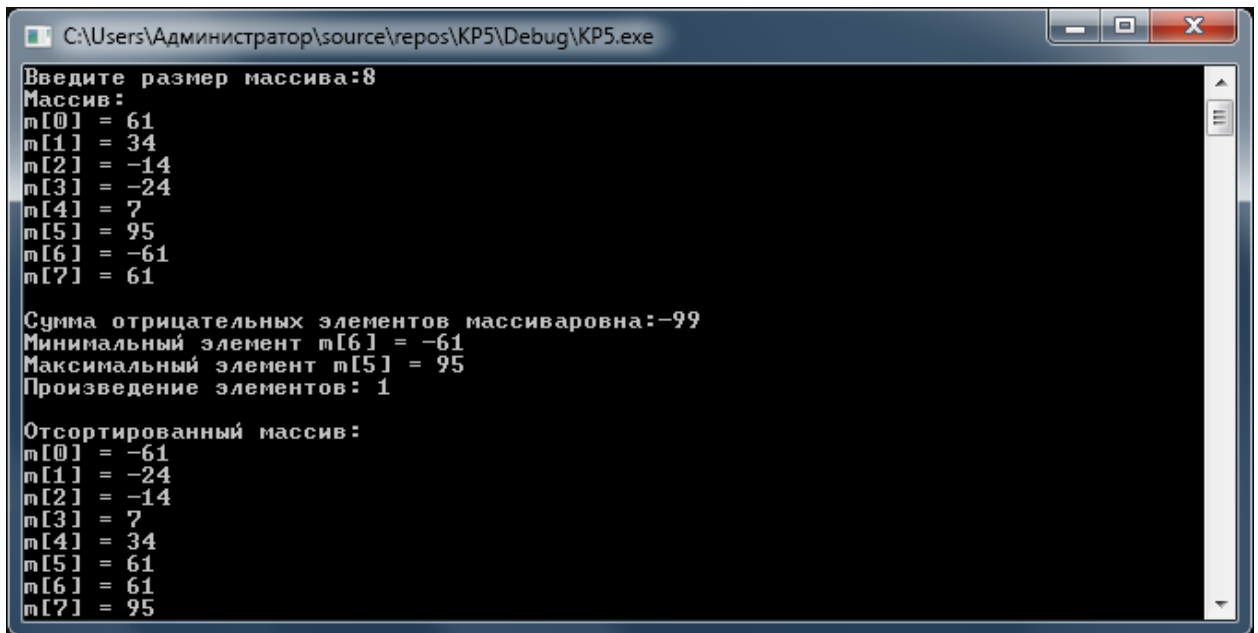
```

```
130.     sort(arr, n,Swap);

131.     //Для обнаружения утечек памяти
132.     _CrtSetReportMode( _CRT_WARN, _CRTDBG_MODE_FILE );
133.     _CrtSetReportFile( _CRT_WARN, _CRTDBG_FILE_STDOUT );
134.     _CrtSetReportMode( _CRT_ERROR, _CRTDBG_MODE_FILE );
135.     _CrtSetReportFile( _CRT_ERROR, _CRTDBG_FILE_STDOUT );
136.     _CrtSetReportMode( _CRT_ASSERT, _CRTDBG_MODE_FILE );
137.     _CrtSetReportFile( _CRT_ASSERT, _CRTDBG_FILE_STDOUT );
138.     _CrtDumpMemoryLeaks();
139.     system("pause");
140.     return 0;
141. }
```

6. Пример выполнения программы

Ниже приведён пример выполнения программы



```
C:\Users\Администратор\source\repos\KP5\Debug\KP5.exe
Введите размер массива:8
Массив:
m[0] = 61
m[1] = 34
m[2] = -14
m[3] = -24
m[4] = 7
m[5] = 95
m[6] = -61
m[7] = 61

Сумма отрицательных элементов массива равна: -99
Минимальный элемент m[6] = -61
Максимальный элемент m[5] = 95
Произведение элементов: 1

Отсортированный массив:
m[0] = -61
m[1] = -24
m[2] = -14
m[3] = 7
m[4] = 34
m[5] = 61
m[6] = 61
m[7] = 95
```

Рис 1. Пример выполнения программы

6. Анализ результатов и выводы

Из достоинств можно выделить следующее:

- Программа выполняет поставленную задачу и работает без ошибок
- Программа выполняет проверку вводимого значения на соответствие
- Использовано динамическое выделение памяти для массива

Из недостатков можно выделить следующее:

- Отсутствует реализация меню