Лабораторная работа № 8

«Разработка, отладка и испытание циклических алгоритмов и программ обработки одномерных динамических массивов»

Выполнить 5 заданий с № 0 по № 4 включительно

ЗАДАНИЕ 0

Отладка ошибок в программе

Задача:

знать, что такое и как проводить отладку в программе;

знать, что такое и как расставлять точки останова в коде программы;

уметь **пошагово** выполнять код программы и показать изменение значений переменных и массивов в окне «Локальные», изменение значений переменных, элементов массивов в ходе работы цикла в окне «Локальные».

Поскольку отладку нужно изучать, имея код программы в качестве материала для практики, то будем работать с кодом решения нижеследующей задачи.

Заменить в двумерном массиве все отрицательные элементы на единицы, положительные – на восьмерки, а нули оставить без изменений на своих местах. Посчитать и вывести количество замен значений элементов на единицы и количество замен значений элементов на восьмерки. Пример кода решения данной задачи:

```
#include <iostream>
#include <Windows.h>
using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    const int mMax = 100, nMax = 150;
    float X[mMax][nMax];
    int m, n;
    cout << "Введите размер массива. Сколько строк: ";
    cin >> m;
    cout << "Введите размер массива. Сколько столбцов: ";
    cin >> n;//Дальше работаем с m, n (а не с mMax, nMax) элементами массива, например, вводим их
    for (int i = 0; i < m; i++)//циклы для инициализации двумерного массива пользователем
    {//с клавиатуры
        for (int j = 0; j < n; j++)
        {
            cout << "X[" << i << "][" << j << "]: ";
            cin >> X[i][j];
        }
    }
    cout << "\nМасссив:\n";
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cout << X[i][j] << '\t';//во внутреннем цикле печатаем элементы одной строки массива
        }
        cout << endl;//во внешнем цикле после печати строки значений переводим курсор на новую строку
    }//чтобы у нас получилась на консоли таблица, а не элементы в одну строку с пробелами
    int countNegativ = 0, countPositiv = 0;
    cout << "\nИзмененный массив:\n";
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
```

```
        {
            if (X[i][j] < 0)
            {
                X[i][j] = 1;
                countNegativ++;
            }
            else//чтобы не сделать отрицательные значения единицей, а потом ее сделать восьмеркой
            {
                if (X[i][j] > 0)
                {
                    X[i][j] = 8;
                    countPositiv++;
                }
            }
            cout << X[i][j] << '\t';
        }
        cout << endl;
    }
    cout << "Заменено значений меньше нуля: " << countNegativ << "\nЗаменено значений больше нуля: " << countPositiv << endl;
    system("pause");
    return 0;
}
```

```
D:\2019\Labs\x64\Debug\Lab3_7.exe
Введите размер массива. Сколько строк: 3
Введите размер массива. Сколько столбцов: 4
X[0][0]: 0
X[0][1]: -9
X[0][2]: -8
X[0][3]: 1
X[1][0]: -2
X[1][1]: 4
X[1][2]: 0
X[1][3]: 5
X[2][0]: 7
X[2][1]: -8
X[2][2]: 6
X[2][3]: 0

Масссив:
0       -9      -8      1
-2      4       0       5
7       -8      6       0

Измененный массив:
0       1       1       8
1       8       0       8
8       1       8       0
Заменено значений меньше нуля: 4
Заменено значений больше нуля: 5
Для продолжения нажмите любую клавишу . . .
```

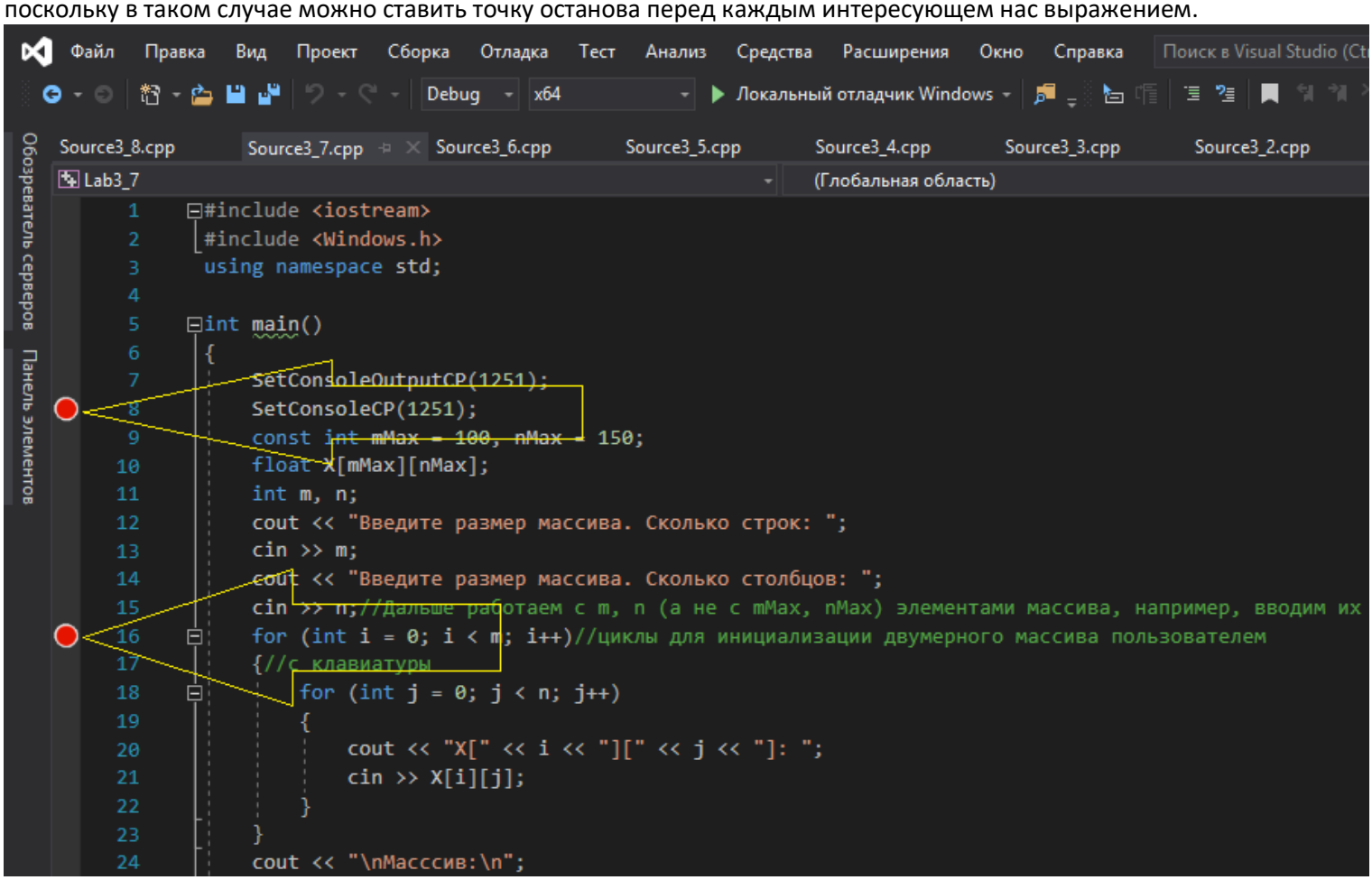
При написании кода программы компилятор может подчеркивать какой-либо код, а окно Список ошибок при этом показывает число ошибок, большее нуля. Это означает, что есть синтаксические ошибки (ошибки в написании программы), которые нужно исправить.

После исправления синтаксических ошибок, которые обычно надо исправлять с самой ранней (она находится в самой верхней строке, из всех строк с ошибками) программа может запускаться и работать, но не выдавать нужный результат, выводить ошибочный результат – это логические ошибки, то есть код синтаксически написан с точки зрения компилятора формально правильно, но он дает компьютеру команды, которые не приводят к нужному для программиста (заказчика) результату. Логические ошибки исправляются путем, например, вывода промежуточных значений на консоль, по которым мы ориентируемся и находим место, где получено первое неправильное значение, разбираемся с выражением кода, которое дает этот промежуточный первый неправильный результат, исправляем выражение и запускаем программу на выполнение, чтобы протестировать успешность исправлений. Если все стало правильно, то можно закомментировать строки с печатью промежуточных результатов, поскольку они уже не нужны.

Еще может потребоваться написать в коде дополнительные выражения, которые будут для нас что-то дополнительно считать, проверять, сравнивать и печатать. По этим дополнительным данным также можно быстрее «выследить» место в коде, где начинается ошибочный результат (это значит локализовать ошибку) и исправить ее.

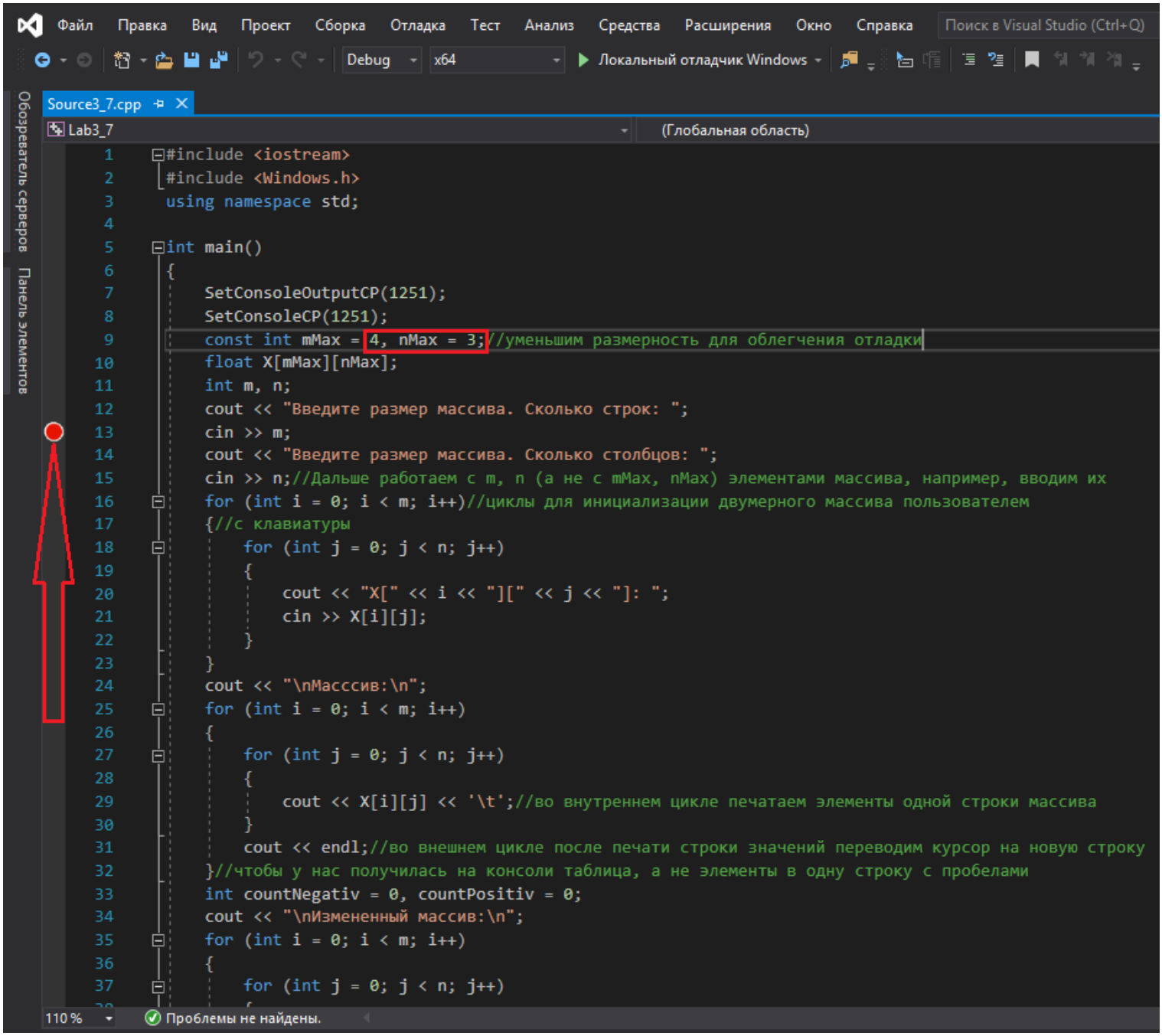
Но есть еще возможность просматривать внутреннее содержимое программы, хранимые в ней переменные, массивы, данные или отсутствие таковых на интересующий нас момент работы программы. Для этого в коде программы нужно поставить **точку останова** перед выражением, перед которым мы хотим поставить выполнение программы на паузу и посмотреть, что к тому моменту в программе есть и какие там значения хранятся (переменные могут быть продекларированы, но еще не проинициализированы значениями, а значит при обращении к таким

переменным будет ошибка или значение в них будет ошибочным – не тем, какое «должно быть»). Точек останова можно поставить одну или несколько, для этого надо дважды нажать правой кнопкой мыши по полю в начале интересующей строки с кодом и увидеть отображение красного кружка. Снять точку останова можно двойным нажатием правой кнопкой мыши по нему до момента, пока кружок не исчезнет. В некоторых участках программы поставить точку останова нельзя, но ее можно обычно поставить перед или после этих участков. Поскольку точка останова ставится в начале строки, то это еще один аргумент, чтобы писать одно выражение кода в одной строке, поскольку в таком случае можно ставить точку останова перед каждым интересующем нас выражением.



После расстановки точек останова в коде программы можно запустить программу на отладку, то есть она будет выполняться до момента срабатывания точки останова (срабатывает пауза, временная блокировка работы программы), далее можно пошагово (по одному выражению) выполнять программу. Помните, что если в очередном выражении ожидаются действия пользователя с клавиатуры, то перейти к следующему за ними шагу нельзя, пока в данном нужном моменте работы (отладки) программы не ввести с клавиатуры нужные данные. Окно программы (консольное окно) доступно и запущено, хотя может быть «свернуто» внизу строки состояния на рабочем столе на мониторе компьютера.

Возьмем в качестве программы для отладки последнюю вышеописанную программу. Поскольку нам будет отображаться дамп памяти локальной области программы («фотография» хранимых в локальной области нашей программы в оперативной памяти имен переменных, массивов и хранимых в них данных (при их наличии)), то уменьшим размеры массива до 3 строк и 4 столбцов (остальной код прежний). Точку останова поставим одну в начале строки кода № 13.

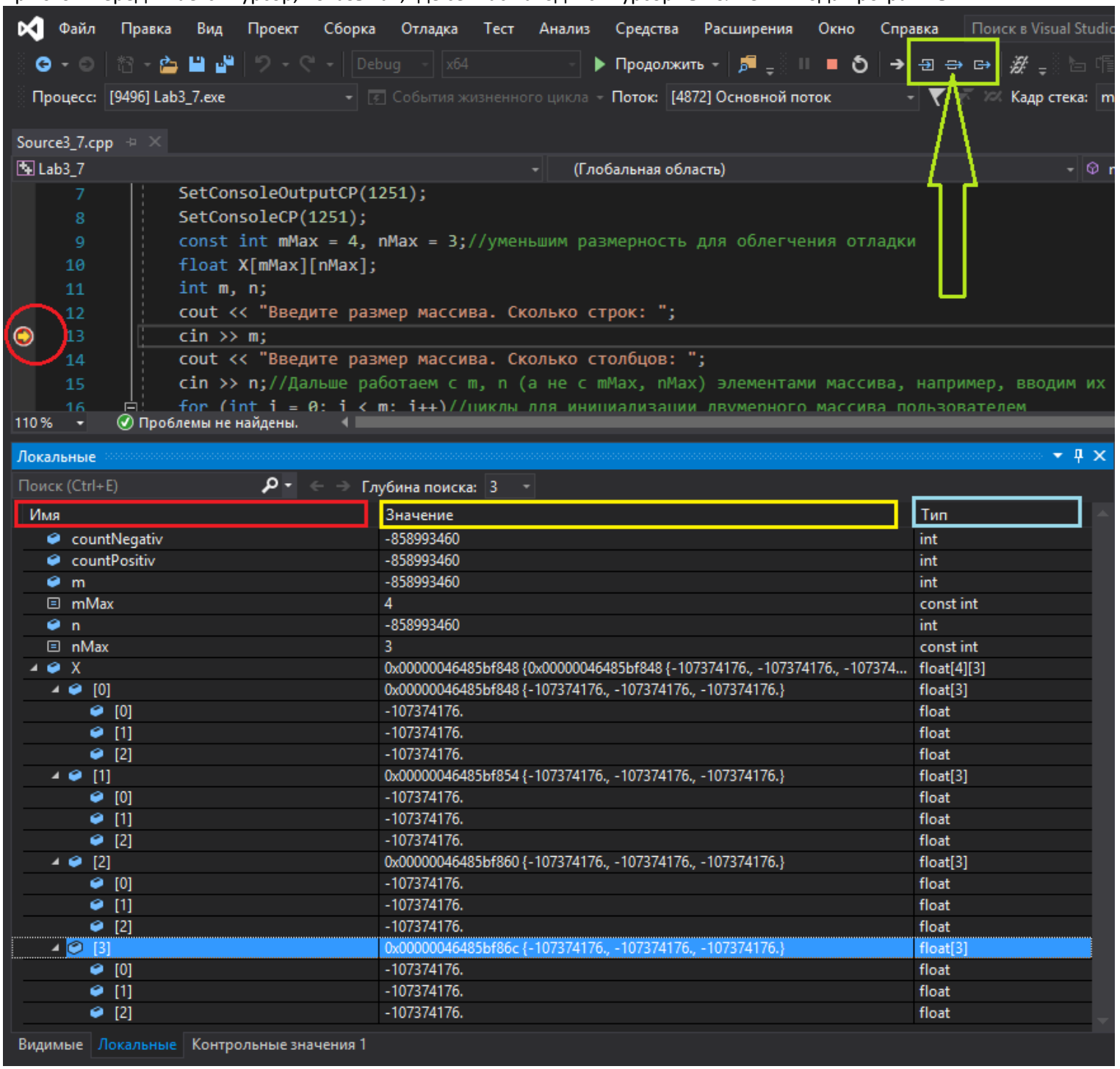


Итак, запускаем программу на отладку как обычно, она выполняется в моем случае до строки кода № 13 и останавливается (в красном кружке желтая стрелка – в этом месте остановилось выполнение программы).

Вверху окна появились дополнительные кнопки, которых раньше не было. Это кнопка «Шаг с заходом» (F11), «Шаг с обходом» (F10) и «Шаг с выходом» (Shift+F11). «Шагом с заходом» (F11) называется шаг в программе, когда нам будет показываться внутреннее содержимое вызываемых функций. Это может быть большое количество кода, который вызывает другой код и так далее, поэтому обычно пользуются «Шагом с обходом» (F10), когда мы по коду идем «большими» цельными шагами, равными отдельным выражениям нашего кода. «Шаг с выходом» (Shift+F11) используется для того, чтобы прекратить пошаговое прохождение программы и запустить ее в обычном режиме начиная с момента нажатия на кнопку «Шаг с выходом» (Shift+F11).

Внизу должно отобразиться окно «Локальные» с дампом памяти. В нем мы видим таблицу с текущими данными программы в режиме реального времени. Есть столбец «Имя» с именами переменных, констант, массивов. Они имеют пиктограммы, обозначающие их тип и «треугольник», по которому можно нажать мышью и раскрыть внутреннее содержимое, например, массивов, структур struct, объектов классов. На скриншоте массив уже развернут. Есть столбец «Тип», в котором указаны типы переменных, типы и размеры массивов. Столбец «Значение» содержит значения, хранимые в созданных переменных, массивах, указателях (имя массива хранит адрес начала данного массива, а указатель хранит адрес переменной, массива, объекта, на которую он ссылается). Обратите внимание, какие значения хранят непроинициализированные переменные разных типов. Обратите внимание, что переменные countPositiv и countNegativ, которые декларируются и инициализируются только в строке кода № 33 уже созданы специальным контроллером, но не проинициализированы. Желтая стрелка на поле слева показывает, где мы сейчас находимся, какой код выполняется в программе. Итак, нажимаем на F10, нажимаем мышью по соответствующей кнопке и передвигаемся по коду, а когда кнопки перехода к следующему выражению не доступны, то в консольном окне программы вводим требуемые значения с клавиатуры. Если при этом смотреть в окно «Локальные», то можно

видеть, как изменяются значения переменных, как постепенно заполняется массив, как нарастают счетчики. По коду при этом передвигается курсор, показывая, где сейчас находится курсор выполнения кода программы.



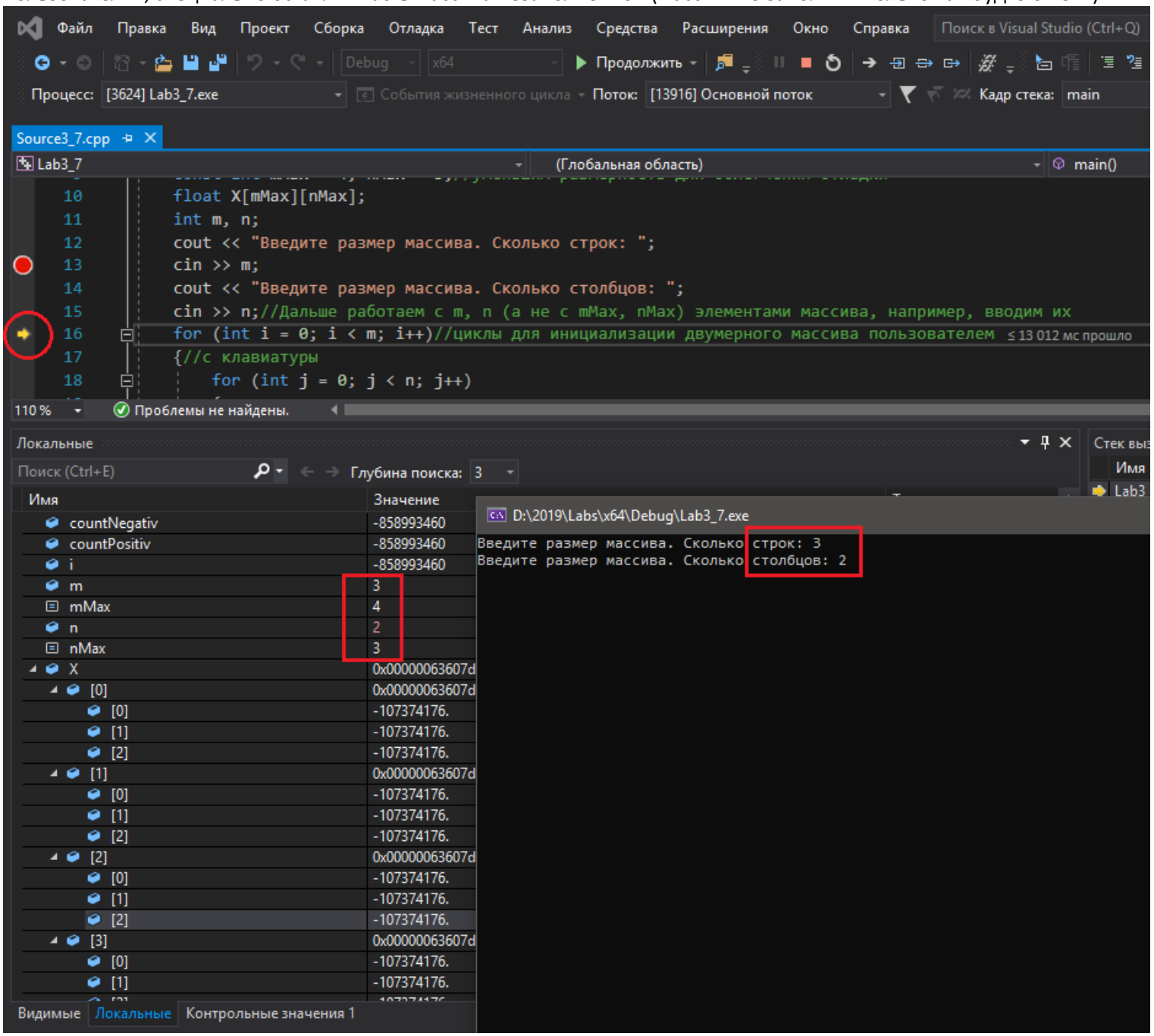
Локальные

Поиск (Ctrl+E) Глубина поиска: 3

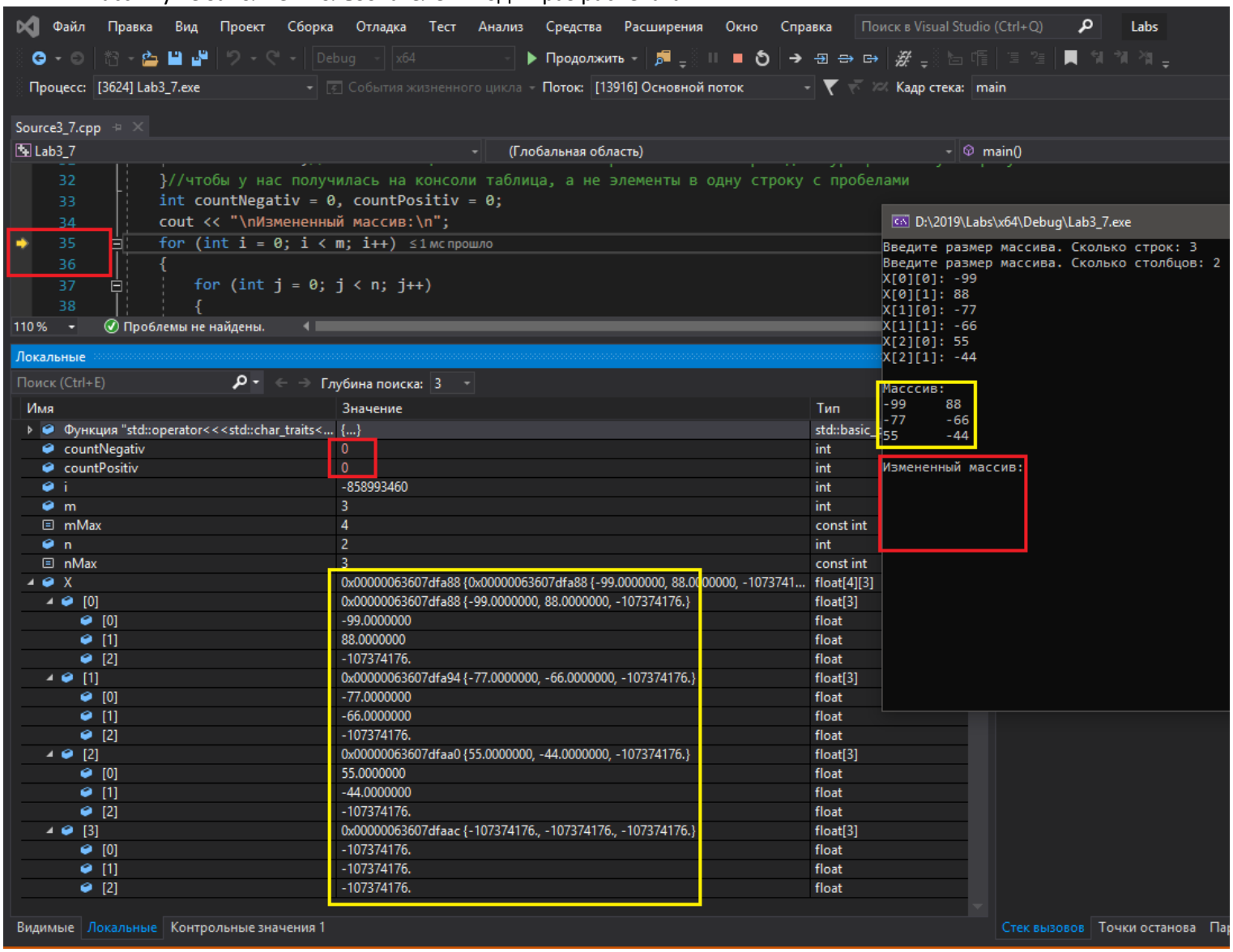
Имя	Значение	Тип
countNegativ	-858993460	int
countPositiv	-858993460	int
m	-858993460	int
mMax	4	const int
n	-858993460	int
nMax	3	const int
X	0x00000046485bf848 {0x00000046485bf848 {-107374176., -107374176., -107374...	float[4][3]
[0]	0x00000046485bf848 {-107374176., -107374176., -107374176.}	float[3]
[0]	-107374176.	float
[1]	-107374176.	float
[2]	-107374176.	float
[1]	0x00000046485bf854 {-107374176., -107374176., -107374176.}	float[3]
[0]	-107374176.	float
[1]	-107374176.	float
[2]	-107374176.	float
[2]	0x00000046485bf860 {-107374176., -107374176., -107374176.}	float[3]
[0]	-107374176.	float
[1]	-107374176.	float
[2]	-107374176.	float
[3]	0x00000046485bf86c {-107374176., -107374176., -107374176.}	float[3]
[0]	-107374176.	float
[1]	-107374176.	float
[2]	-107374176.	float

Видимые Локальные Контрольные значения 1

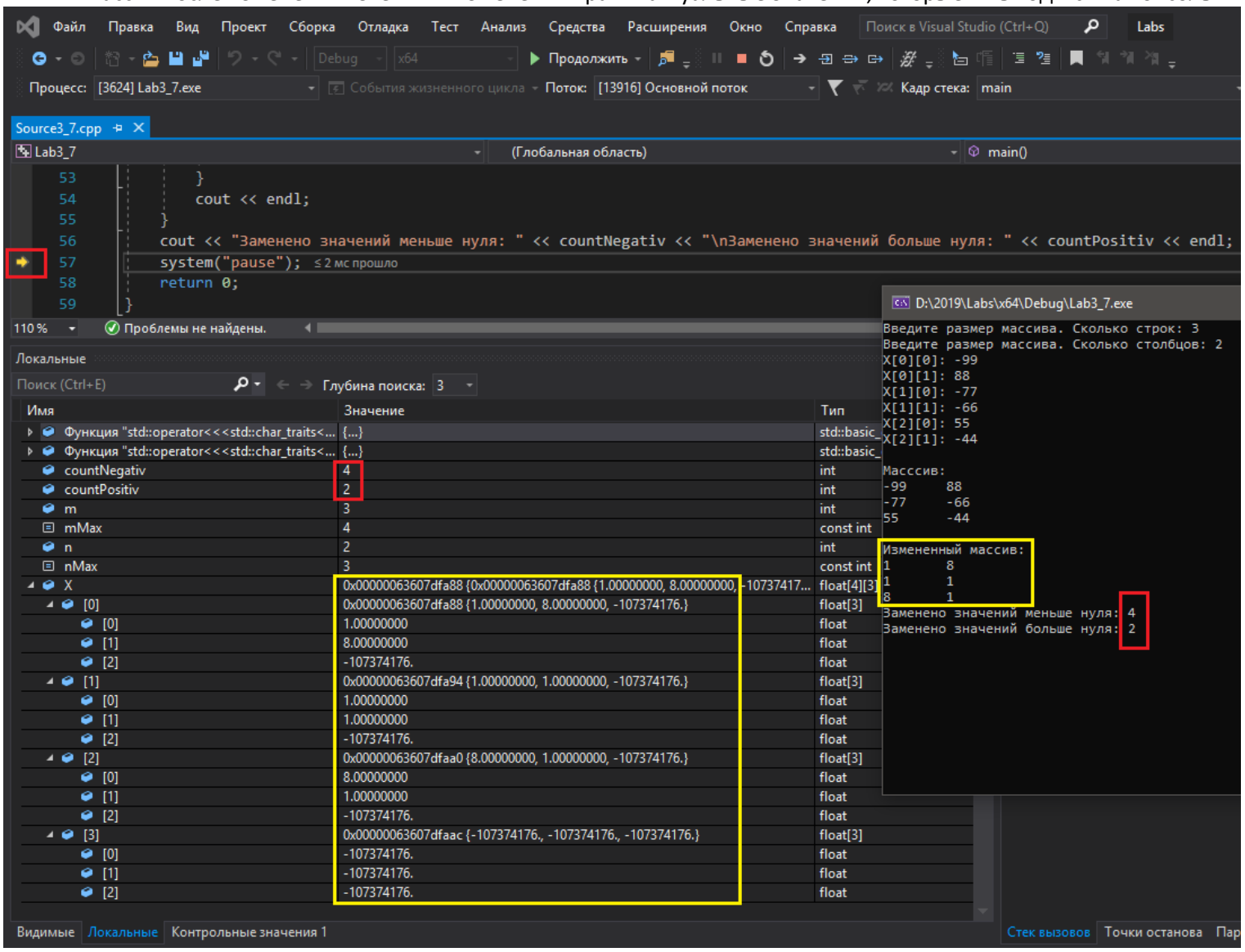
Сделаем некоторое количество шагов по коду программы и введем оба значения размера массива «для пользователя», специально оставляя часть массива незаполненной (массив 4x3 заполним только как будто он 3x2).



Массив уже заполнен пользователем и один раз распечатан.



Массив после изменений. Счетчики изменений хранят актуальные значения, которые и выводятся на консоль.



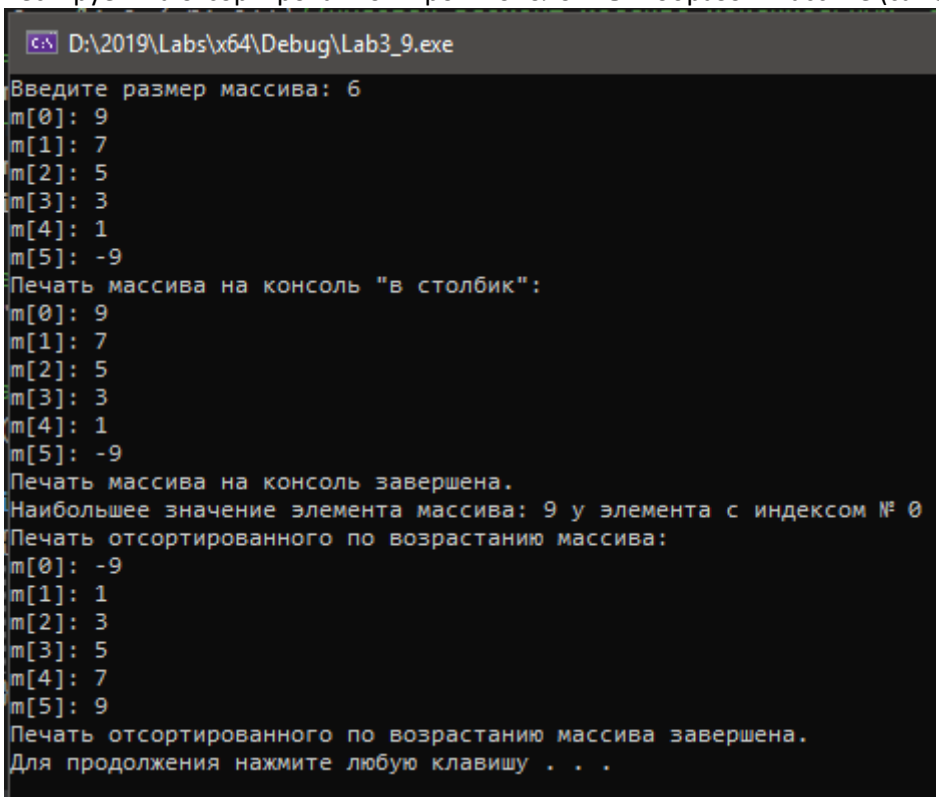
Таким образом содержимое переменных и массивов можно смотреть «на лету» при отладке, даже не печатая на консоль.

Если в работе программы есть проблема, то точку останова нужно ставить **перед** тем участком кода, который «подозреваем» больше остальных и пошагово его проходить, следя за изменением хранимых значений в окне «Локальные». Обратите внимание, что в окне «Локальные» массив показан реального размера 4x3, а его периферийные незаполненные элементы имеют соответствующее значение, которое означает, что они не проинициализированы.

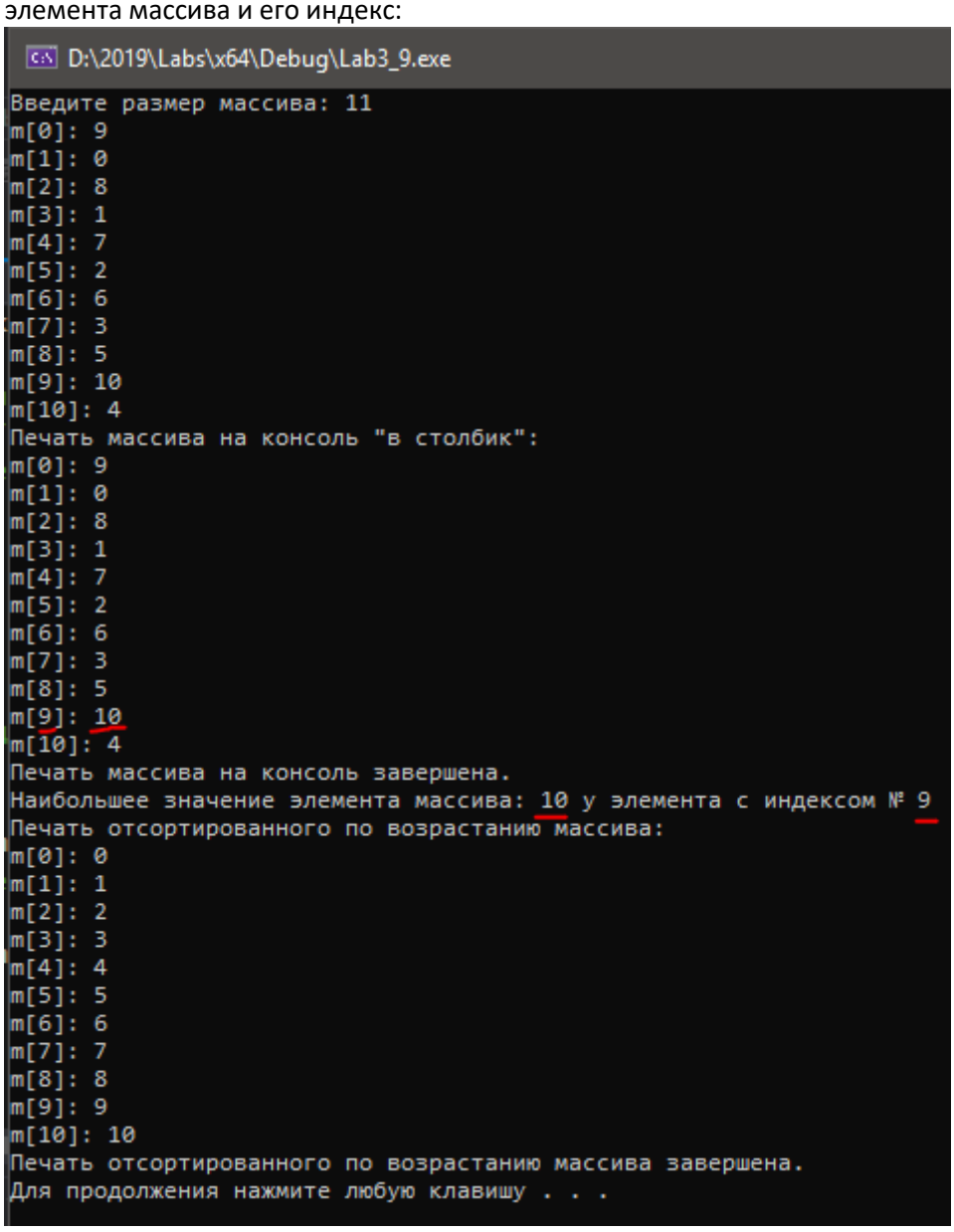
Пример программы по созданию, печати, сортировке, поиску максимального элемента и его индекса в динамическом одномерном массиве.

```cpp
#include <iostream>
#include <Windows.h>
using namespace std;
//ДООП - динамическая область оперативной памяти - управляемая куча в Microsoft Visual C++ - Managed heap.
int main()//Динамические объекты можно разместить только в ДООП
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    int n;//переменная для хранения размера массива (количества штук элементов, которые можно в него поместить)
    do//цикл с постусловием заменяет код с goto и это грамотно
    {
        cout << "Введите размер массива: ";
        cin >> n;
    }
    while (n <= 0);//помним про обязательную тут "точку с запятой"
    int* m = new int[n];//создаем динамический массив для хранения n штук целочисленных элементов
    if (m == NULL)//если под массив не удалось найти места в ДООП
    {
        cout << "Не удалось выделить память под массив.\n";//сообщаем пользователю об этом
        system("pause");
        return 0;//и завершаем программу
    }
    for (int i = 0; i < n; i++)//если программа не завершилась, значит память под массив выделена успешно
    {//и можно заполнять элементы массива значениями, вводимыми пользователем с клавиатуры
        cout << "m[" << i << "]: ";
        cin >> m[i];
    }
    cout << "Печать массива на консоль \"в столбик\":\n";//проверяем, действительно ли в массиве находятся
    for (int i = 0; i < n; i++)//все введенные пользователем элементы в порядке их ввода пользователем
    {
        cout << "m[" << i << "]: " << m[i] << endl;
    }
    cout << "Печать массива на консоль завершена.\n";
    int max = m[0], iMax = 0;//найдем максимальное значение элемента в массиве и его индекс. Для этого просто объявим
    for (int i = 1; i < n; i++)//нулевой элемент массива максимальным, а его индекс № 0 - индексом максимального элемента массива
    {//теперь проходимся с первого элемента массива (поскольку нулевой элемент мы уже прошли перед циклом) до последнего элемента
        if (m[i] > max)//массива и, если значение очередного элемента будет больше "максимального" значения, то этот очередной
        {//элемент делаем максимальным и его индекс делаем индексом максимального элемента
            max = m[i];
            iMax = i;
        }
    }//переменные max и iMax созданы перед циклом, а потому они существуют и после завершения цикла и хранят значения, помещенные
    cout << "Наибольшее значение элемента массива: " << max << " у элемента с индексом № " << iMax << endl;//в них последними
    for (int i = 0; i < n-1; i++)//СОРТИРОВКА массива по ВОЗРАСТАНИЮ. Для сортировки методом пузырька одномерного массива нужно
    {//написать цикл в цикле. Пишите код, который не выйдет за границы массива: индексы крайних элементов № 0 и № (n-1)
        for (int j = 0; j < n-i-1; j++)//здесь есть оптимизация
        {
            if (m[j] > m[j + 1])//если значение предшествующего элемента больше значения следующего элемента, то их расположение
            {//противоречит принципу "по ВОЗРАСТАНИЮ" и нужно обменять хранимые в них значения местами с использованием
                int t = m[j];//промежуточной переменной t
                m[j] = m[j + 1];
                m[j + 1] = t;
            }
        }
    }
    cout << "Печать отсортированного по возрастанию массива:\n";//проверяем, удалось ли отсортировать массив по ВОЗРАСТАНИЮ
    for (int i = 0; i < n; i++)//самая "неудобная" программе проверка - изначально ввести значения массива,
    {//отсортированные прямо противоположным образом (то есть по убыванию)
        cout << "m[" << i << "]: " << m[i] << endl;
    }
    cout << "Печать отсортированного по возрастанию массива завершена.\n";
    delete[] m;//обязательно удаляем массив из ДООП, поскольку он нам уже не нужен. Система теперь пометит ранее занятую область ДООП
    m = NULL;//как свободную, и чтобы наш указатель на нее ошибочно не указывал, "зануляем" его - теперь он будет указывать на "пустоту"
    system("pause");
    return 0;
}
```

Тестируем на отсортированном противоположным образом массиве (самый неудобный программе случай):



Работает корректно. Теперь введем «разбросанные» данные, чтобы проверить поиск максимального значения элемента массива и его индекс:



ЗАДАНИЕ 1:

Массивы A и B динамические; выделение динамической области оперативной памяти (ДООП) выполнить посредством **new**, а после работы с динамическим массивом обязательно освободить от него динамическую область оперативной памяти посредством **delete**.

Предусмотреть реакцию программы на ситуации, при которых задача не имеет решения.

ВАРИАНТЫ:

1. Заданы два массива A(N) и B(M). Первым на печать вывести массив, сумма значений которого окажется наименьшей.
2. Заданы два массива A(N) и B(M). Первым на печать вывести массив, произведение значений которого окажется наименьшим.
3. Заданы два массива A(M) и B(M). В каждом из массивов найти наименьшее значение и прибавить его ко всем элементам массивов. На печать вывести исходные и преобразованные массивы.
4. Заданы два массива A(M) и B(M). В каждом из массивов найти наибольшее значение и вычесть его из всех элементов массивов. На печать вывести исходные и преобразованные массивы.
5. Заданы два массива A(M) и B(M). В каждом из массивов найти среднее арифметическое всех элементов массивов. На печать вывести исходные массивы и найденные значения.
6. Заданы два массива A(N) и B(M). Первым на печать вывести массив, содержащий наименьшее значение. Напечатать также это значение и его порядковый номер.
7. Заданы два массива A(N) и B(M). Первым на печать вывести массив, содержащий наибольшее значение. Напечатать также это значение и его порядковый номер.
8. Заданы два массива A(M) и B(M). Подсчитать в них количество отрицательных элементов и первым на печать вывести массив, имеющий наименьшее их количество.
9. Заданы два массива A(M) и B(M). Подсчитать в них количество положительных элементов и первым на печать вывести массив, имеющий наименьшее их количество.
10. Заданы два массива A(M) и B(M). Подсчитать в них количество отрицательных элементов и первым на печать вывести массив, имеющий наибольшее их количество.
11. Заданы два массива A(M) и B(M). Подсчитать в них количество положительных элементов и первым на печать вывести массив, имеющий наибольшее их количество.
12. Заданы два массива A(M) и B(M). Подсчитать в них количество элементов, больших значения **t** (t вводится пользователем с клавиатуры) и первым на печать вывести массив, имеющий наименьшее их количество.
13. Заданы два массива A(N) и B(N). Подсчитать в них количество элементов, меньших значения **t** (t вводится пользователем с клавиатуры) и первым на печать вывести массив, имеющий наименьшее их количество.
14. Заданы два массива A(M) и B(M). Подсчитать в них количество элементов, больших значения **t** (t вводится пользователем с клавиатуры) и первым на печать вывести массив, имеющий наибольшее их количество.
15. Заданы два массива A(M) и B(M). В каждом из массивов найти наименьшее значение и умножить на него все элементы массивов. На печать вывести исходные и преобразованные массивы.
16. Заданы два массива A(M) и B(M). В каждом из массивов найти наибольшее значение и умножить на него все элементы массивов. На печать вывести исходные и преобразованные массивы.

ЗАДАНИЕ 2:

Массив создается в динамической области памяти функцией **new** и удаляется функцией **delete**. Предусмотреть реакцию программы на ситуации, при которых задача не имеет решения.

В одномерном массиве, состоящем из n вводимых с клавиатуры элементов, вычислить:

ВАРИАНТЫ:

1. Произведение элементов массива, расположенных между максимальным и минимальным элементами;
2. Сумму элементов массива, расположенных между первым и последним нулевыми элементами;
3. Сумму элементов массива, расположенных до последнего положительного элемента;
4. Сумму элементов массива, расположенных между первым и последним положительными элементами;
5. Произведение элементов массива, расположенных между первым и вторым нулевыми элементами;
6. Сумму элементов массива, расположенных между первым и вторым отрицательными элементами;
7. Сумму элементов массива, расположенных до минимального элемента;
8. Сумму элементов массива, расположенных после последнего отрицательного элемента;
9. Сумму элементов массива, расположенных после последнего элемента, равного нулю;
10. Сумму модулей элементов массива, расположенных после минимального по модулю элемента;
11. Сумму элементов массива, расположенных после минимального элемента;
12. Сумму элементов массива, расположенных после первого положительного элемента;
13. Сумму модулей элементов массива, расположенных после первого отрицательного элемента;
14. Сумму модулей элементов массива, расположенных после первого элемента, равного нулю;
15. Сумму положительных элементов массива, расположенных до максимального элемента;

16. Сумму элементов массива, расположенных после последнего элемента, значение которого выше среднего арифметического значения всех элементов массива;

ЗАДАНИЕ 3:

Номер варианта	Задание
1	Дан массив размера N. Вывести его элементы в обратном порядке.
2	Дан массив размера N. Вывести вначале его элементы с четными индексами, а затем — с нечетными.
3	Дан целочисленный массив A размера 10. Вывести номер первого из тех его элементов A[i], которые удовлетворяют двойному неравенству: A[1] < A[i] < A[10]. Если таких элементов нет, то вывести 0.
4	Дан целочисленный массив размера N. Преобразовать его, прибавив к четным числам первый элемент. Первый и последний элементы массива не изменять.
5	Дан целочисленный массив размера N. Вывести вначале все его четные элементы, а затем — нечетные.
6	Поменять местами минимальный и максимальный элементы массива размера 10.
7	Заменить все положительные элементы целочисленного массива размера 10 на значение минимального.
8	Дан массив размера 10. Переставить в обратном порядке элементы массива, расположенные между его минимальным и максимальным элементами.
9	Дан массив размера N. Осуществить циклический сдвиг элементов массива влево на одну позицию.
10	Дан массив размера N и число k (0 < k < 5, k < N). Осуществить циклический сдвиг элементов массива влево на k позиций.
11	Проверить, образуют ли элементы целочисленного массива размера N арифметическую прогрессию. Если да, то вывести разность прогрессии, если нет — вывести 0.
12	Дан массив ненулевых целых чисел размера N. Проверить, чередуются ли в нем четные и нечетные числа. Если чередуются, то вывести 0, если нет, то вывести номер первого элемента, нарушающего закономерность.
13	Дан массив размера N. Найти количество его минимумов.
14	Дан массив размера N. Найти максимальный из его локальных минимумов.
15	Дан массив размера N. Определить количество участков, на которых его элементы возрастают.
16	Дан массив размера N. Определить количество его промежутков монотонности (то есть участков, на которых его элементы возрастают или убывают).
17	Дано вещественное число R и массив размера N. Найти элемент массива, который наиболее близок к данному числу.
18	Дано вещественное число R и массив размера N. Найти два элемента массива, сумма которых наиболее близка к данному числу.
19	Дан массив размера N. Найти номера двух ближайших чисел из этого массива.
20	Дан целочисленный массив размера N. Определить максимальное количество его одинаковых элементов.
21	Дан целочисленный массив размера N. Удалить из массива все элементы, встречающиеся менее двух раз.
22	Дан целочисленный массив размера N. Если он является перестановкой, то есть содержит все числа от 1 до N, то вывести 0, в противном случае вывести номер первого недопустимого элемента.
23	Дан массив размера N. Преобразовать его, вставив после каждого положительного элемента нулевой элемент.
24	Дан целочисленный массив размера N. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии — количество этих элементов (длина серии может быть равна 1). Вывести массив, содержащий длины всех серий исходного массива.
25	Дан целочисленный массив размера N. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии — количество этих элементов (длина серии может быть равна 1). Преобразовать массив, увеличив каждую его серию на один элемент.
26	Дан целочисленный массив размера N. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии — количество этих элементов (длина серии может быть равна 1). Преобразовать массив, увеличив серию наибольшей длины на один элемент.
27	Даны два массива A и B размера 5, элементы которых упорядочены по возрастанию. Объединить эти массивы так, чтобы результирующий массив остался упорядоченным.
28	Заменить все отрицательные элементы целочисленного массива размера 10 на значение максимального.
29	Дан массив размера N. Осуществить циклический сдвиг элементов массива вправо на одну позицию.
30	Проверить, образуют ли элементы целочисленного массива размера N геометрическую прогрессию. Если да, то вывести знаменатель прогрессии, если нет — вывести 0.

ЗАДАНИЕ 4:

Вариант 1:
В одномерном массиве {3.24, -7.16, 2.28, -0.16, -3.22, 7.14, 2.88, -3.20, 0.99, -4.15} вычислить:
1) сумму отрицательных элементов массива;
2) произведение элементов массива, расположенных между максимальным и минимальным элементами;
3) упорядочить элементы массива по возрастанию.

Вариант 2:
В одномерном массиве {-4.23, 5.66, -2.70, 10.16, 1.12, -7.14, -5.16, 4.26, 0.99, 3.15} вычислить:
1) сумму положительных элементов массива;
2) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами;
3) упорядочить элементы массива по убыванию.

Вариант 3:
В одномерном массиве {97, 0, -12, 26, 62, -83, -94, 0, -66, 0} вычислить:
1) произведение элементов массива с четными номерами (0 - четное);
2) сумму элементов массива, расположенных между первым и последним нулевыми элементами;
3) преобразовать массив таким образом, чтобы сначала располагались все неотрицательные, а потом - все отрицательные элементы.

Вариант 4:
В одномерном массиве {3.24, -0.16, 2.28, -0.16, -3.22, 7.14, 0.88, -3.20, 0.99, 4.15} вычислить:
1) сумму элементов массива с нечетными номерами (0 - четное);
2) сумму элементов массива, расположенных между первым и последним отрицательными элементами;
3) сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившееся в конце массива место заполнить нулями.

Вариант 5:
В одномерном массиве {3.24, -7.16, 2.18, -0.16, -3.22, 7.14, 2.88, -3.20, -0.99, -4.15} вычислить:
1) максимальный элемент массива;
2) сумму элементов массива, расположенных до последнего положительного элемента;
3) сжать массив, удалив из него все элементы, модуль которых находится в интервале [2.2,5.5]. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 6:
В одномерном массиве {44, 0, 28, -16, -22, 0, 88, -20, -99, -15} вычислить:
1) минимальный элемент массива;
2) сумму элементов массива, расположенных между первым и последним положительными элементами;
3) преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом - все остальные.

Вариант 7:
В одномерном массиве {33, 0, 28, -16, -32, 74, 0, -30, 99, 0} вычислить:
1) номер максимального элемента массива;
2) произведение элементов массива, расположенных между первым и вторым нулевыми элементами;
3) преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине - элементы, стоявшие в четных позициях (0 - четное).

Вариант 8:
В одномерном массиве {3.24, -0.16, 2.28, 0.16, 3.22, 7.14, 2.88, -3.20, 0.99, -4.15} вычислить:
1) номер минимального элемента массива;
2) сумму элементов массива, расположенных между первым и вторым отрицательными элементами;
3) преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом - все остальные.

Вариант 9:
В одномерном массиве {-3.24, 0, 2.28, -0.16, 0, -7.14, 2.88, -3.20, 0.99, 0} вычислить:
1) максимальный по модулю элемент массива;
2) сумму элементов массива, расположенных между первым и вторым положительными элементами (0 - не

положительное);
3) преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

Вариант 10:
В одномерном массиве {24, -16, 0, -96, -32, 714, 0, -320, 0, -415} вычислить:
1) максимальный по модулю элемент массива;
2) сумму модулей элементов массива, расположенных после первого элемента, равного нулю;
3) преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине - элементы, стоявшие в нечетных позициях (0 - четное).

Вариант 11:
В одномерном массиве {3.24, 7.16, 2.28, -0.16, -3.22, 7.14, 2.88, -3.20, 0.99, -4.15} вычислить:
1) номер минимального по модулю элемента массива;
2) сумму модулей элементов массива, расположенных после первого отрицательного элемента;
3) сжать массив, удалив из него все элементы, величина которых находится в интервале [-5.5, 5.5]. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 12:
В одномерном массиве {-3.24, -7.16, -2.28, -0.16, -3.22, 7.14, 2.88, -3.20, 0.99, -4.15} вычислить:
1) номер максимального по модулю элемента массива;
2) сумму элементов массива, расположенных после первого положительного элемента;
3) преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале [-3,3], а потом - все остальные.

Вариант 13:
В одномерном массиве {3.24, -7.16, 2.28, -0.16, -3.22, 7.14, 2.88, -3.20, 0.99, -4.15} вычислить:
1) количество элементов массива, лежащих в диапазоне от -4 до 4;
2) сумму элементов массива, расположенных после максимального элемента;
3) упорядочить элементы массива по убыванию модулей элементов.

Вариант 14:
В одномерном массиве {3.24, 0, 2.28, 0, -3.22, 0, 2.88, -3.20, 0, 0} вычислить:
1) количество элементов массива, равных 0;
2) сумму элементов массива, расположенных после минимального элемента;
3) упорядочить элементы массива по возрастанию модулей элементов.

Вариант 15:
В одномерном массиве {3.24, -7.16, 2.28, -4.16, -3.22, 7.14, 0.1, -3.20, 3.99, -4.15} вычислить:
1) количество элементов массива, больших 3;
2) произведение элементов массива, расположенных после максимального по модулю элемента;
3) преобразовать массив таким образом, чтобы сначала располагались все отрицательные элементы, а потом - все положительные (0 - положительное).

Вариант 16:
В одномерном массиве {3.24, -7.16, 2.28, -0.16, -3.22, 7.14, 2.88, -3.20, 0.99, -4.15} вычислить:
1) количество отрицательных элементов массива;
2) сумму модулей элементов массива, расположенных после минимального по модулю элемента;
3) заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию

Вариант 17:
В одномерном массиве {3.24, 0, 2.28, 0, -3.22, 7.14, 2.88, -3.20, 0.99, -4.15} вычислить:
1) количество положительных элементов массива (0 - не положительное);
2) сумму элементов массива, расположенных после последнего элемента, равного 0;
3) преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает 1, а потом - все остальные.

Вариант 18:
В одномерном массиве {3.24, -7.16, 2.28, -0.16, -3.22, 7.14, 2.88, 3.20, 0.99, 4.15} вычислить:
1) количество элементов массива, меньших 2;
2) сумму целых частей элементов массива, расположенных после последнего отрицательного элемента;

3) преобразовать массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более чем на 20%, а потом - все остальные.

Вариант 19:
В одномерном массиве {3.24, -7.16, 2.28, -0.16, -3.22, 7.14, 2.88, -3.20, 0.99, -4.15} вычислить:
1) произведение отрицательных элементов массива;
2) сумму положительных элементов массива, расположенных до максимального элемента;
3) изменить порядок следования элементов массива на обратный.

Вариант 20:
В одномерном массиве {3.24, 7.16, 2.28, -0.16, -3.22, 7.14, 2.88, -3.20, 0.99, -4.15} вычислить:
1) произведение положительных элементов массива;
2) сумму элементов массива, расположенных до минимального элемента;
3) упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах

Контрольные вопросы

1) В каких ситуациях в программировании целесообразно использовать динамические массивы?
2) Что будет возвращено при попытке объявить динамический массив недопустимо большого размера?
3) Как размещаются в памяти элементы динамического массива?
4) Назовите порядок освобождения памяти, выделенной под двумерный динамический массив.
5) Какая область динамической памяти, выделенной под двумерный динамический массив, будет освобождена, если только применить операцию: delete[] mass; ?
6) Что такое одномерный массив? Для чего используются одномерные массивы? Как они описываются?
7) Как называется номер элемента одномерного массива?
8) Как в программе использовать значение конкретного элемента одномерного массива?
9) Как можно заполнить одномерный массив?
10) Почему в программе на C++ необходимо, чтобы был известен размер массива?
11) Можно ли выполнить прямое присваивание массивов объявленных так: int x[10], y[10];?
12) Когда, с какой целью и почему возможно объявление безразмерных массивов?
13) Какие ограничения распространяются на тип массива?
14) Для чего в программах используются двумерные массивы? Как они описываются?
15) Сколько индексов характеризуют конкретный элемент двумерного массива?
16) Как в программе использовать значение конкретного элемента двумерного массива?
17) Как можно заполнить двумерный массив?
18) Какую структуру данных описывает двумерный массив?
19) Какой индекс двумерного массива изменяется быстрее при последовательном размещении элементов массива в оперативной памяти?

Домашнее задание

Прочитать страницы 259 – 367 книги Дейтел, Х. Как программировать на C++ / Х.Дейтел, П.Дейтел (путь на сервере: s1 / Предметы / ОАиП_Шаляпин / Дейтел Харви - Как программировать на C++.pdf).