ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ПРАКТИЧЕСКОЙ РАБОТЕ № 4.3

По дисциплине «Языки программирования»

Выполнил: ст. гр. ТКИ – 141

Негодаева Т. А.

Проверил: к.т.н., доц.

Васильева М. А.

**Содержание**

* Формулировка задачи
* Блок схема алгоритма
* Решение задачи на языке программирования C++
* Решение тестовых примеров
* Зачет задания в GitHub

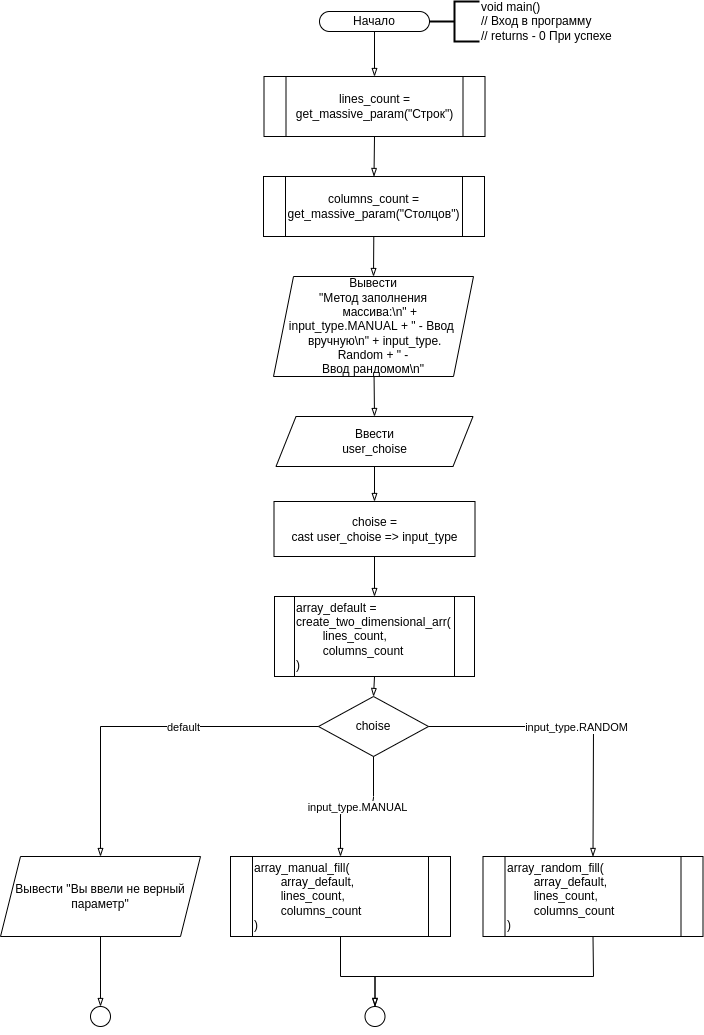
**Задание 4.3**

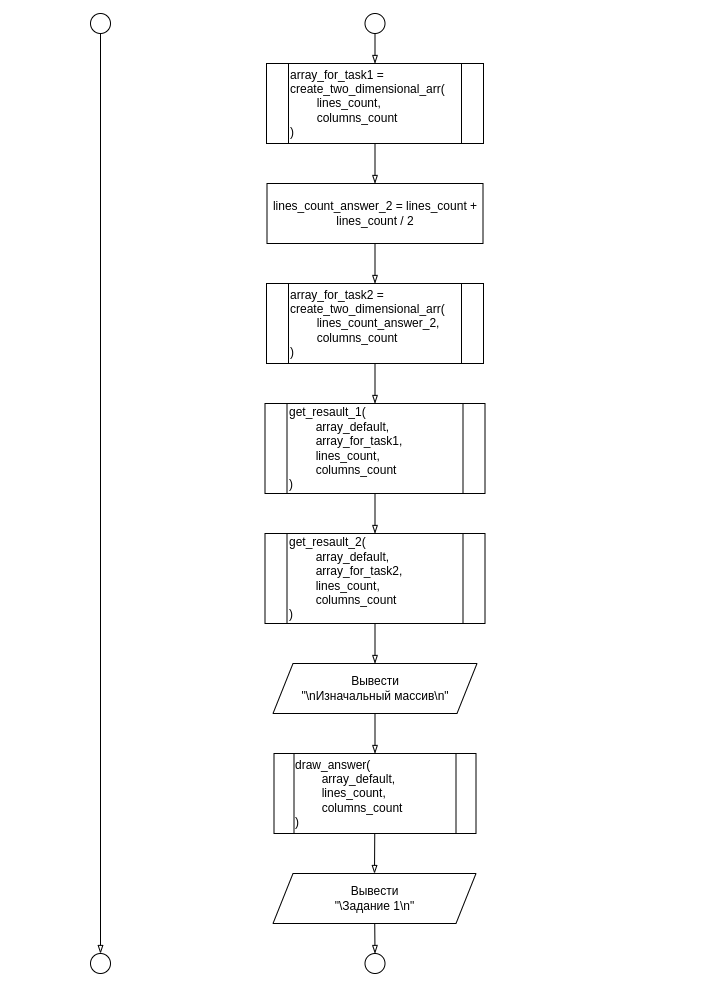
1. **Формулировка задачи**

|  |  |  |
| --- | --- | --- |
| **№ варианта** | **Задачи** | **Константы** |
| 1 | 1. Заменить максимальный по модулю элемент каждой строки на противоположный по знаку.  2. Вставить после каждой четной строки первую строку. |  |

1. **Блок схема алгоритма**

Блок-схемы алгоритмов функций представлены на рисунках (Рисунок 1,Рисунок 2, Рисунок 3, Рисунок 4, Рисунок 5, Рисунок 6, Рисунок 7, Рисунок 8, Рисунок 9, Рисунок 10)





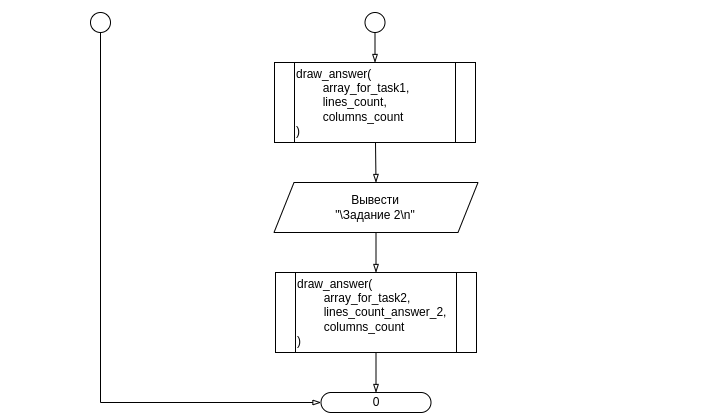


Рисунок 1 - Блок-схема функции main

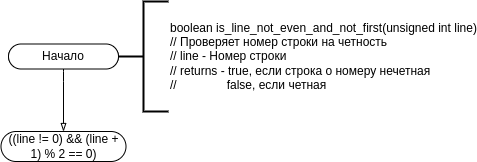


Рисунок 2 - Блок-схема функции is\_line\_not\_even\_and\_not\_first

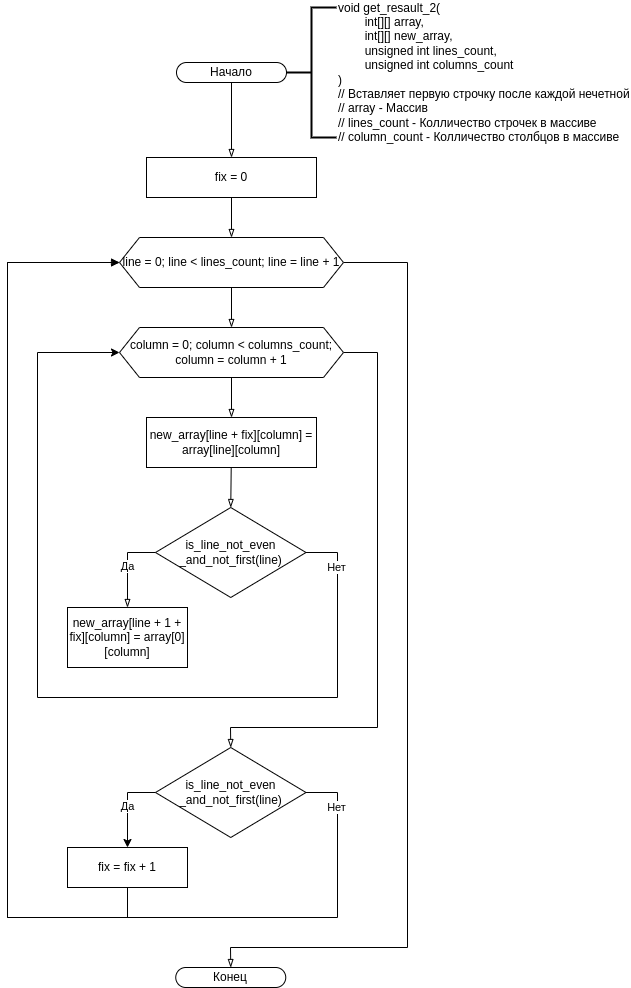


Рисунок 3 - Блок-схема функции get\_resault\_2()

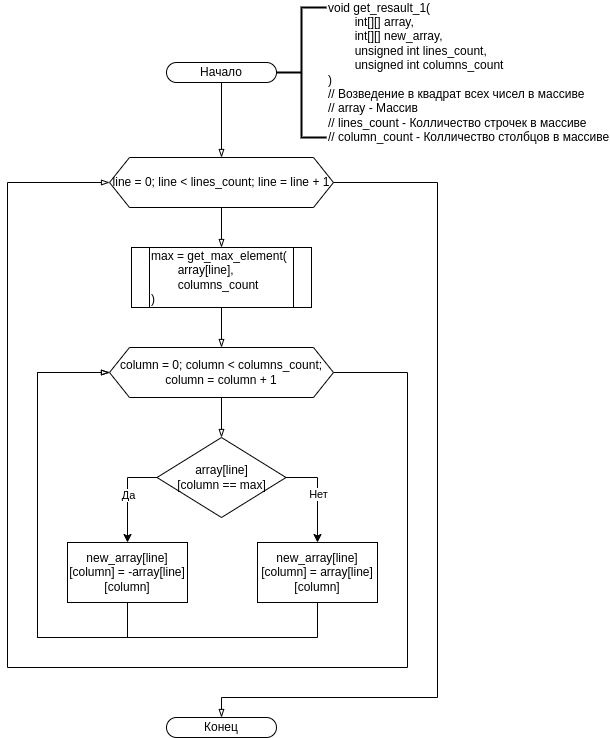


Рисунок 4 - Блок-схема функции get\_resault\_1()

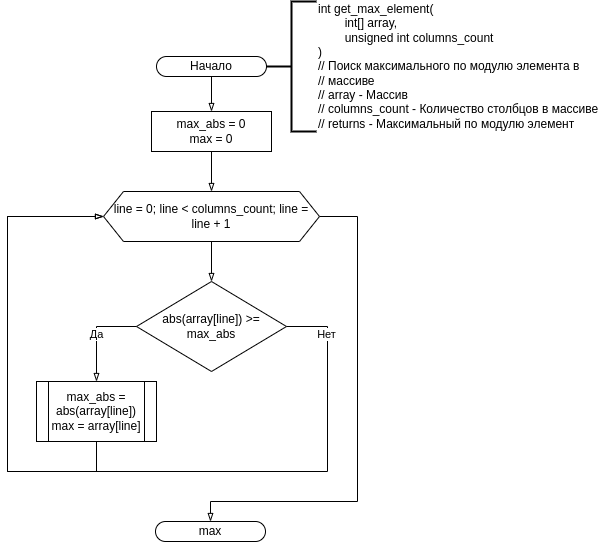


Рисунок 5 - Блок-схема функции get\_max\_element

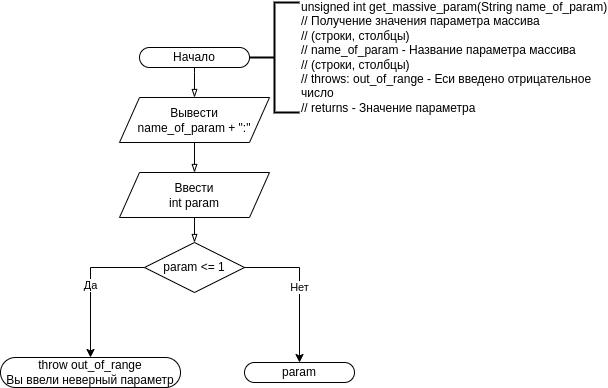


Рисунок 6 - Блок-схема функции get\_massive\_param

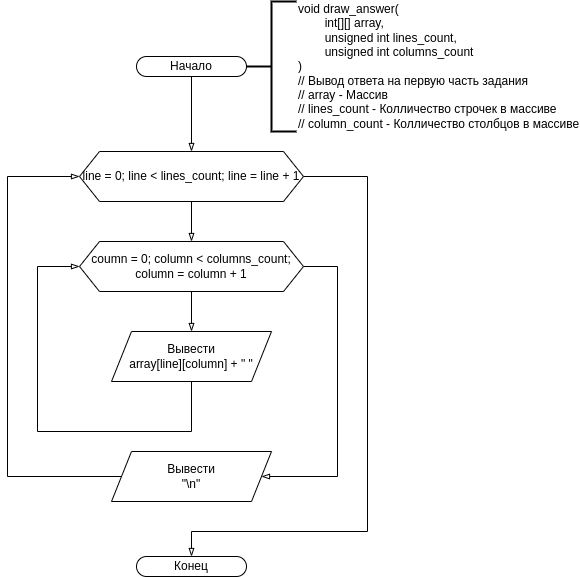


Рисунок 7 - Блок-схема функции draw\_answer

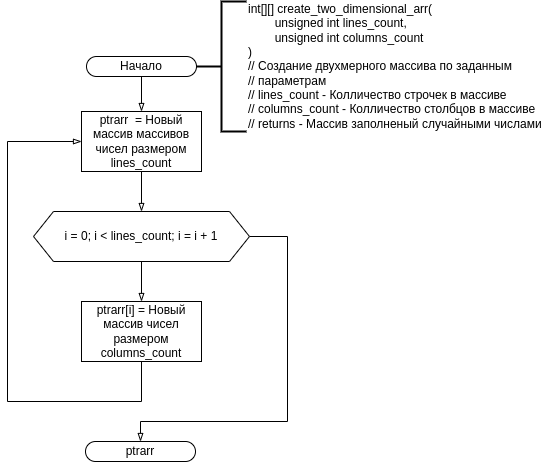


Рисунок 8 - Блок-схема функции create\_two\_dimensional\_arr

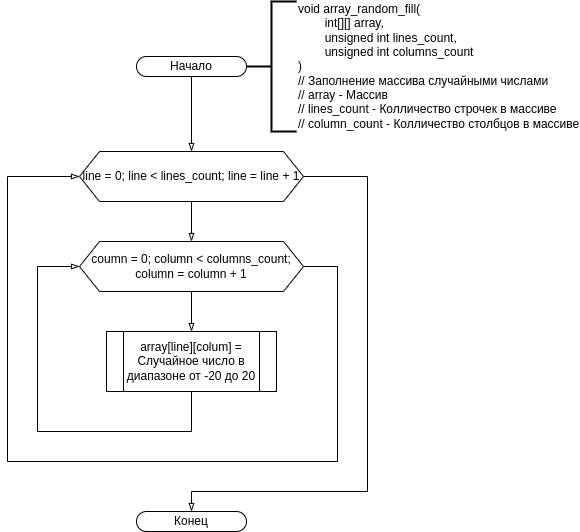


Рисунок 9 - Блок-схема функции array\_random\_fill

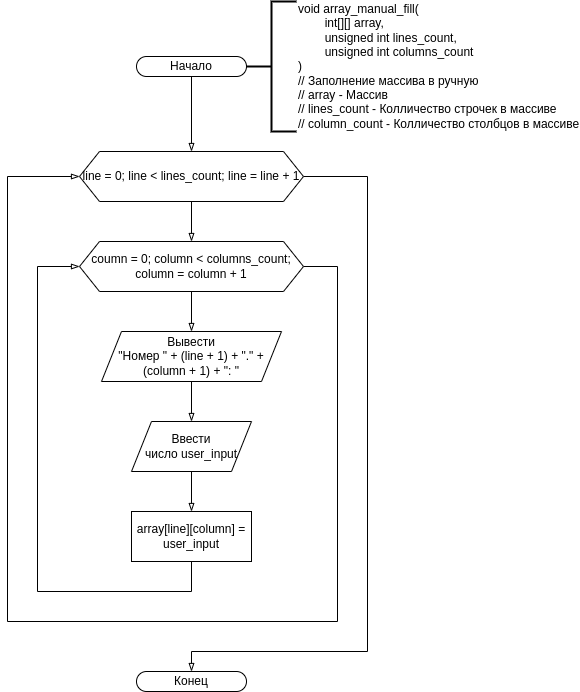


Рисунок 10 - Блок-схема функции array\_manual\_fill

1. **Решение задачи на языке программирования C++**

#include <exception>

#include <random>

#include <string>

#include <iostream>

using namespace std;

/\*\*

\*\brief Пользовательский ввод выбора набора массива

\*\param MANUAL Ручной ввод данных

\*\param RANDOM Случайный набор данных

\*/

enum class input\_type

{

MANUAL = 1,

RANDOM

};

/\*\*

\*\brief Получение значения параметра массива(строки, столбцы)

\*\param name\_of\_param Название параметра массива(строка, столбец)

\*\throw Если введено отрицательное число

\*\return Значение параметра

\*/

size\_t get\_massive\_param(string name\_of\_param);

/\*\*

\*\brief Проверяет номер строки на четность

\*\param line Номер строки

\*\return true если строка по номеру нечетная, и false, если четная

\*/

bool is\_line\_not\_even\_and\_not\_first(const size\_t line);

/\*\*

\*\brief Создание двумерного массива по заданным параметрам

\*\param lines\_count Колличество строчек в массиве

\*\param columns\_count Колличество столбцов в массиве

\*\return Массив заполненый случайными числами

\*/

int\*\* create\_two\_dimensional\_arr(const size\_t lines\_count, const size\_t columns\_count);

/\*\*

\*\brief Заполение массива случайными числами

\*\param array Массив

\*\param lines\_count Колличество строчек в массиве

\*\param column\_count Колличество столбцов в массиве

\*/

void array\_random\_fill(int\*\* array, const size\_t lines\_count, const size\_t columns\_count);

/\*\*

\*\brief Заполение массива в ручную

\*\param array Массив

\*\param lines\_count Колличество строчек в массиве

\*\param columns\_count Колличество столбцов в массиве

\*/

void array\_manual\_fill(int\*\* array, const size\_t lines\_count, const size\_t columns\_count);

/\*\*

\*\brief Поиск максимального по модолю элемента в массиве

\*\param array Массив

\*\param columns\_count Колличество столбцов в массиве

\*\return Максимальный по модулю элемент

\*/

int get\_max\_element(int\* array, const size\_t columns\_count);

/\*\*

\*\brief Возведение в квадрат всех чисел в массиве

\*\param array Массив

\*\param lines\_count Колличество строчек в массиве

\*\param columns\_count Колличество столбцов в массиве

\*/

void get\_resault\_1(int\*\* array, int\*\* new\_array, const size\_t lines\_count, const size\_t columns\_count);

/\*\*

\*\brief Вставляет первую строчку после каждей нечетной

\*\param array Массив

\*\param lines\_count Колличество строчек в массиве

\*\param columns\_count Колличество столбцов в массиве

\*/

void get\_resault\_2(int\*\* array, int\*\* new\_array, const size\_t lines\_count, const size\_t columns\_count);

/\*\*

\* \brief Вывод ответа на первую часть задания

\*\param lines\_count Колличество строчек в массиве

\*\param columns\_count Колличество столбцов в массиве

\*/

void draw\_answer(int\*\* array, const size\_t lines\_count, const size\_t columns\_count);

/\*\*

\*\brief Удаление двухмерного массива

\*\param lines\_count Колличество строчек (массивов)

\*/

void delete\_array(int\*\*& array, const size\_t lines\_count);

/\*\*

\* \brief вход в программу

\* \return 0 При успехе

\*/

int main()

{

setlocale(LC\_ALL, "Russian");

try

{

size\_t lines\_count = get\_massive\_param("Строк");

size\_t columns\_count = get\_massive\_param("Столбцов");

cout << "Метод заполнения массива:\n" << static\_cast<int>(input\_type::MANUAL) << " - Ввод вручную\n"

<< static\_cast<int>(input\_type::RANDOM) << " - Ввод рандомом\n";

int user\_choise = 0;

cin >> user\_choise;

const auto choise = static\_cast<input\_type>(user\_choise);

int\*\* array\_default = create\_two\_dimensional\_arr(lines\_count, columns\_count);

switch (choise)

{

case input\_type::MANUAL:

{

array\_manual\_fill(array\_default, lines\_count, columns\_count);

break;

}

case input\_type::RANDOM:

{

array\_random\_fill(array\_default, lines\_count, columns\_count);

break;

}

cout << "!!!";

}

int\*\* array\_for\_task1 = create\_two\_dimensional\_arr(lines\_count, columns\_count);

size\_t lines\_count\_answer\_2 = lines\_count + lines\_count / 2;

int\*\* array\_for\_task2 = create\_two\_dimensional\_arr(lines\_count\_answer\_2, columns\_count);

get\_resault\_1(array\_default, array\_for\_task1, lines\_count, columns\_count);

get\_resault\_2(array\_default, array\_for\_task2, lines\_count, columns\_count);

cout << "\n------------Изначальный массив----------" << endl;

draw\_answer(array\_default, lines\_count, columns\_count);

cout << "\n------------Задание 1------------" << endl;

draw\_answer(array\_for\_task1, lines\_count, columns\_count);

cout << "\n------------Задание 2-----------" << endl;

draw\_answer(array\_for\_task2, lines\_count\_answer\_2, columns\_count);

delete\_array(array\_for\_task1, lines\_count);

delete\_array(array\_for\_task2, lines\_count\_answer\_2);

delete\_array(array\_default, lines\_count);

}

catch (exception& err)

{

cout << err.what();

}

return 0;

}

bool is\_line\_not\_even\_and\_not\_first(const size\_t line)

{

return ((line != 0) && (line + 1) % 2 == 0);

}

size\_t get\_massive\_param(string name\_of\_param)

{

cout << name\_of\_param << ":";

int param;

cin >> param;

if (param <= 1)

{

throw out\_of\_range("Вы ввели неверный параметр");

}

return param;

}

void array\_random\_fill(int\*\* array, const size\_t lines\_count, const size\_t columns\_count)

{

random\_device rd;

mt19937 gen(rd());

uniform\_int\_distribution<> uniformIntDistribution(-20, 20);

for (size\_t line = 0; line < lines\_count; line++) {

for (size\_t column = 0; column < columns\_count; column++) {

array[line][column] = uniformIntDistribution(gen);

}

}

}

void array\_manual\_fill(int\*\* array, const size\_t lines\_count, const size\_t columns\_count)

{

for (size\_t line = 0; line < lines\_count; line++) {

for (size\_t column = 0; column < columns\_count; column++) {

cout << "Номер " << line + 1 << "." << column + 1 << ": ";

int user\_input = 0;

cin >> user\_input;

array[line][column] = user\_input;

}

}

}

int\*\* create\_two\_dimensional\_arr(const size\_t lines\_count, const size\_t columns\_count)

{

int\*\* ptrarr = new int\* [lines\_count];

for (size\_t i = 0; i < lines\_count; i++) {

ptrarr[i] = new int[columns\_count];

}

return ptrarr;

}

int get\_max\_element(int\* array, const size\_t columns\_count)

{

int max\_abs = 0;

int max = 0;

for (size\_t line = 0; line < columns\_count; line++)

{

if (abs(array[line]) >= max\_abs)

{

max\_abs = abs(array[line]);

max = array[line];

}

}

return max;

}

void get\_resault\_1(int\*\* array, int\*\* new\_array, const size\_t lines\_count, const size\_t columns\_count)

{

for (size\_t line = 0; line < lines\_count; line++) {

int max = get\_max\_element(array[line], columns\_count);

for (size\_t column = 0; column < columns\_count; column++) {

if (array[line][column] == max)

{

new\_array[line][column] = -array[line][column];

}

else

{

new\_array[line][column] = array[line][column];

}

}

}

}

void get\_resault\_2(int\*\* array, int\*\* new\_array, const size\_t lines\_count, const size\_t columns\_count)

{

int fix = 0;

for (size\_t line = 0; line < lines\_count; line++)

{

for (size\_t column = 0; column < columns\_count; column++)

{

new\_array[line + fix][column] = array[line][column];

if (is\_line\_not\_even\_and\_not\_first(line))

{

new\_array[line + 1 + fix][column] = array[0][column];

}

}

if (is\_line\_not\_even\_and\_not\_first(line))

{

fix++;

}

}

}

void draw\_answer(int\*\* array, const size\_t lines\_count, const size\_t columns\_count)

{

for (size\_t line = 0; line < lines\_count; line++)

{

for (size\_t column = 0; column < columns\_count; column++)

{

cout << array[line][column] << " ";

}

cout << endl;

}

}

void delete\_array(int\*\*& array, const size\_t lines\_count)

{

for (size\_t line = 0; line < lines\_count; line++)

{

delete[] array[line];

}

delete[] array;

array = nullptr;

}

1. **Решение тестовых примеров**

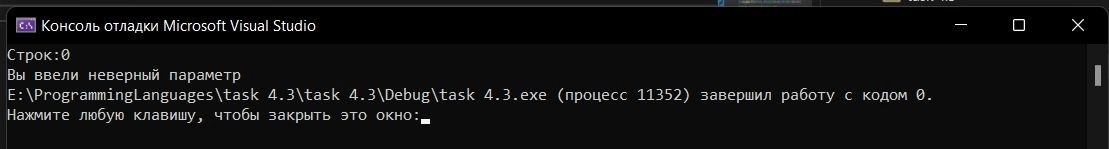


Рисунок 11 - Ошибка при не правильном количестве строк

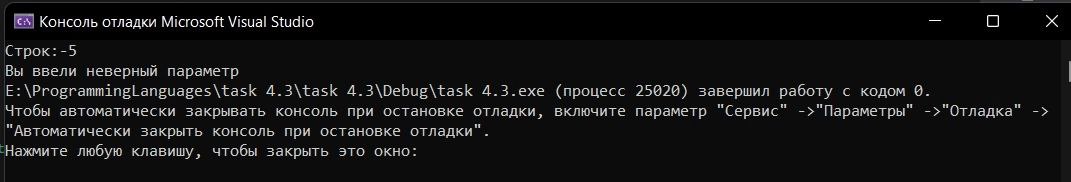


Рисунок 12 - Ошибка при отрицательном вводе

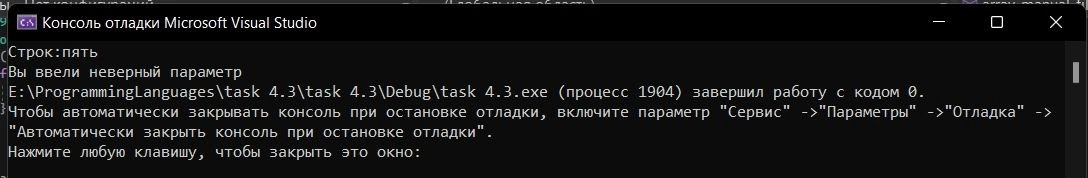


Рисунок 13 - Ошибка при буквенном вводе

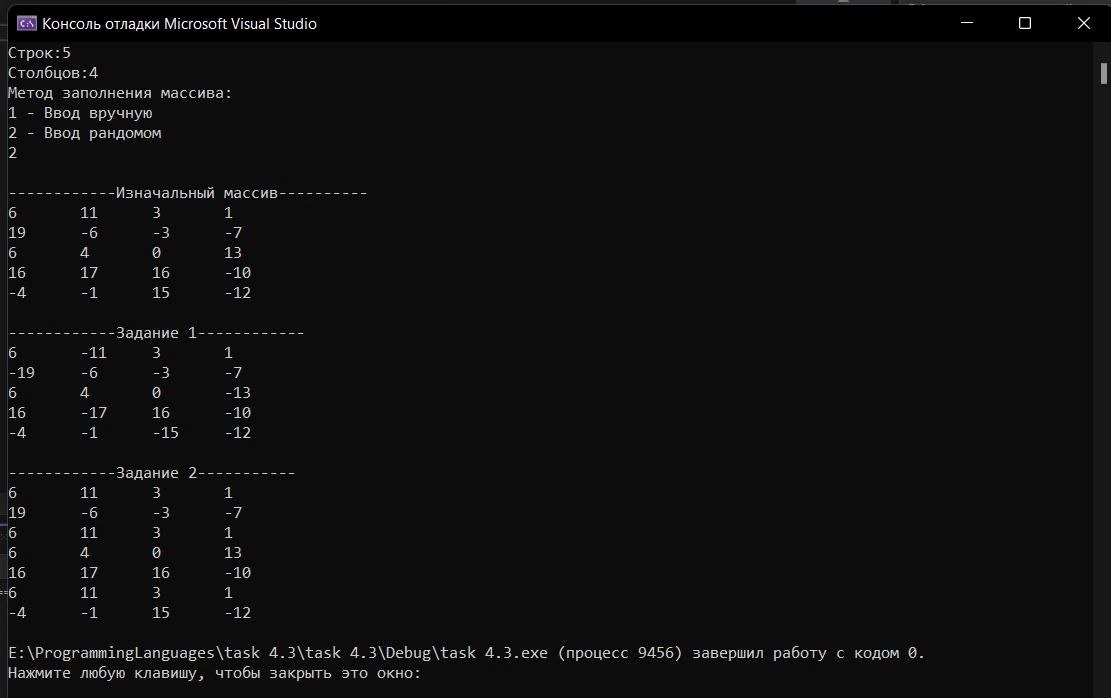


Рисунок 14 - Результат при правильном вводе (Массив 5x4)

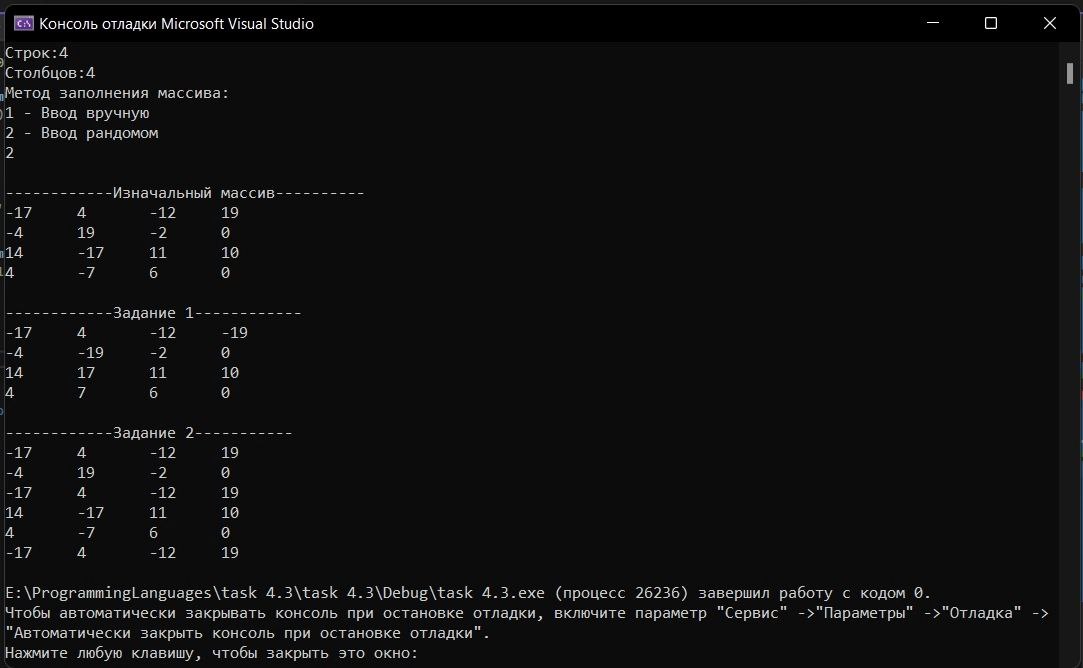


Рисунок 15 - Результат при правильном вводе (Массив 4x4)

1. **Зачет задания в GitHub**

