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 № 1

По дисциплине «Языки программирования»

Выполнил: ст. гр. ТКИ – 141

Захаров Дмитрий Константинович

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2022

Содержание

[Задание 4-3 3](#_Toc115287553)

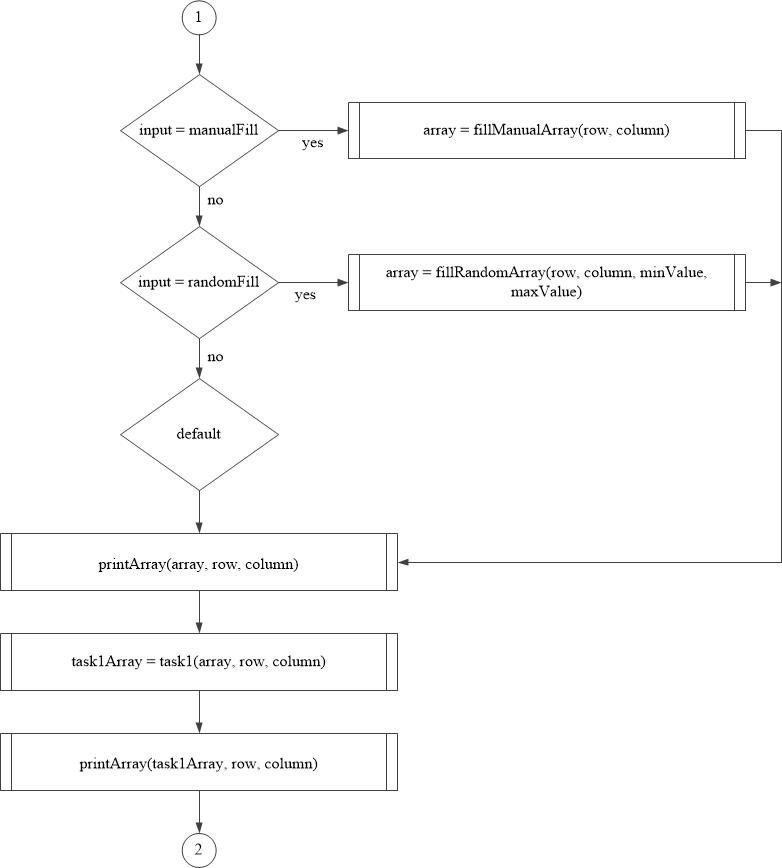
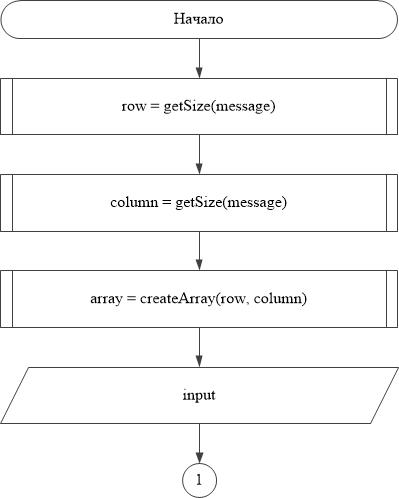
# Задание 4–3

1. Формулировка задачи

Создать многомерный массив n\*m из n целых чисел. Реализовать возможность заполнения массива, как случайными числами, так и с помощью клавиатуры по желанию пользователя. Пункты задания организовать в виде функций (методов). Вывести массив на экран.

|  |  |
| --- | --- |
| **№ варианта** | **Задачи** |
| 10 | 1. Заменить нулевой элемент каждого столбца максимальным по модулю элементом массива.  2. Вставить после каждого столбца, содержащего максимальный по модулю элемент, столбец из нулей. |

1. Блок-схемы алгоритмов функций представлены на рисунках (Рисунок 1 - 12)



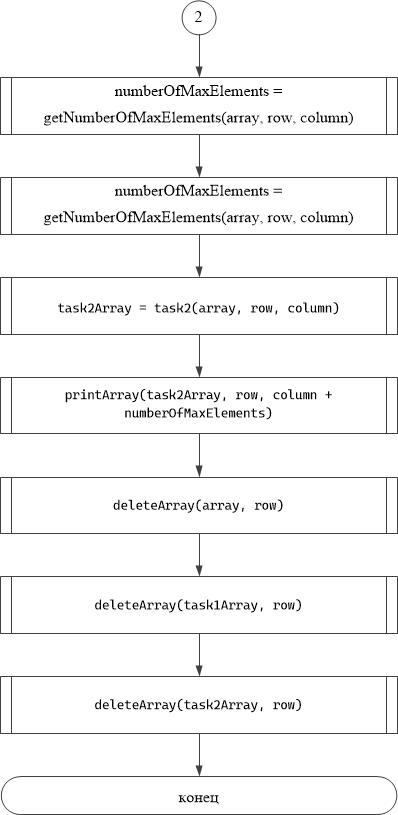


Рисунок 1 – Блок-схема функции main

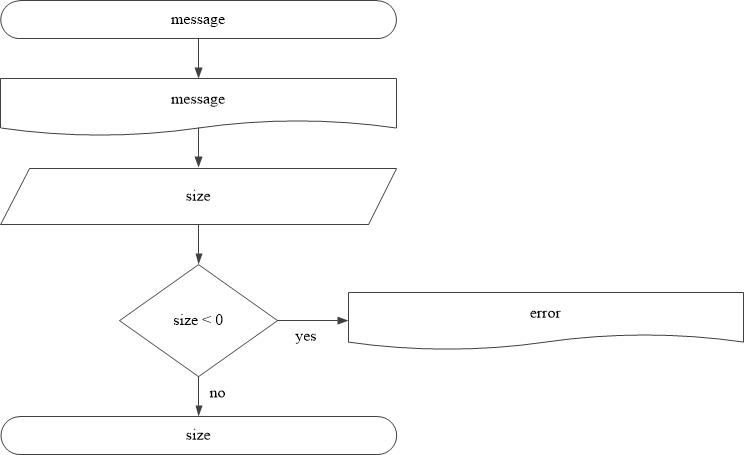


Рисунок 2 – Блок-схема функции getSize

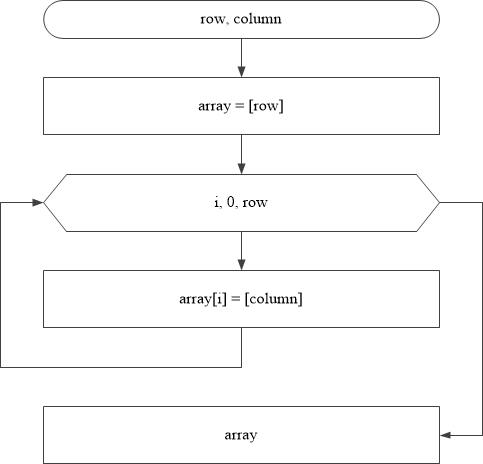


Рисунок 3 – Блок-схема функции createArray

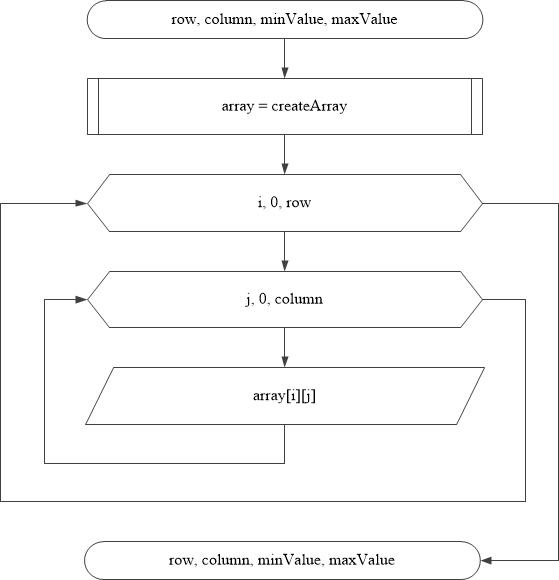


Рисунок 4 – Блок-схема функции fillManualArray

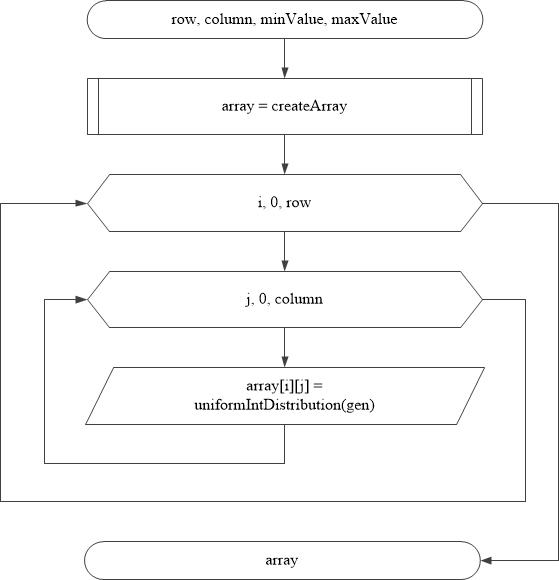


Рисунок 5 – Блок-схема функции fillRandomArray

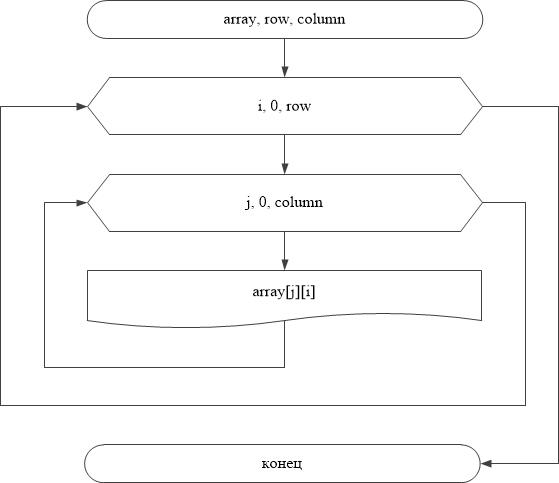


Рисунок 6 – Блок-схема функции printArray

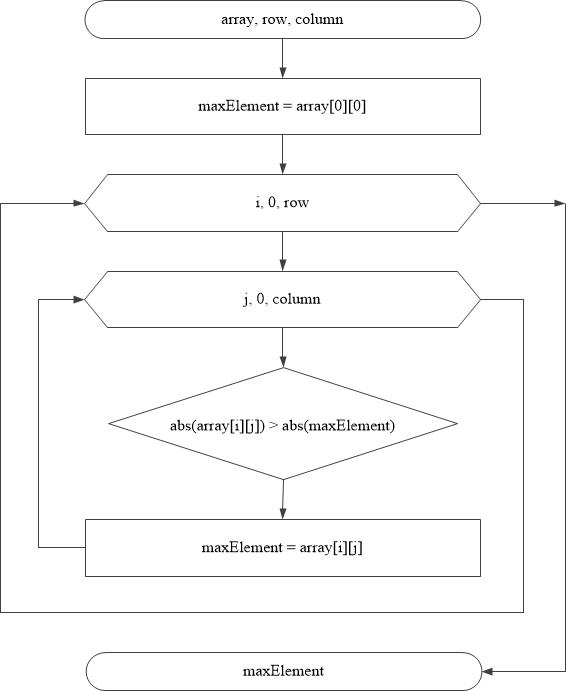


Рисунок 7 – Блок-схема функции getMaxElement

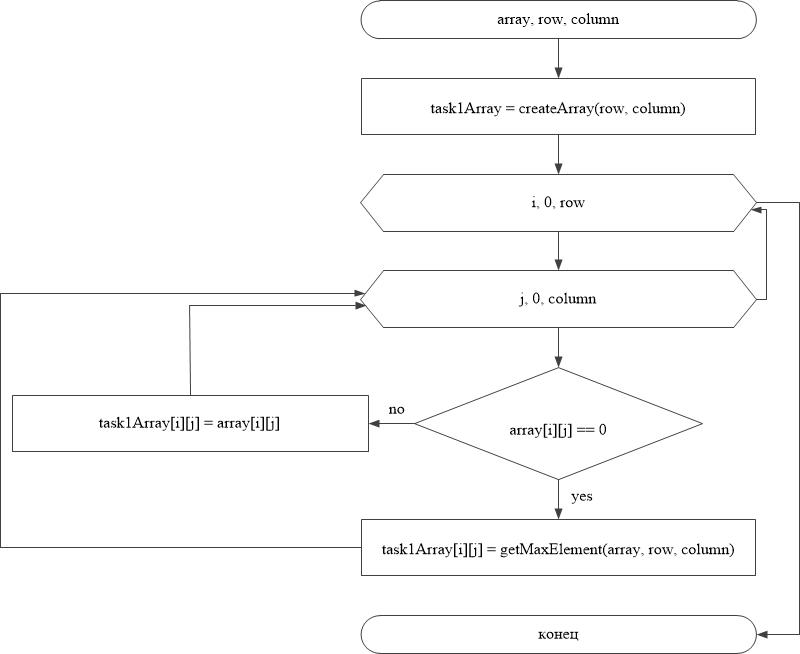


Рисунок 8 – Блок-схема функции task1

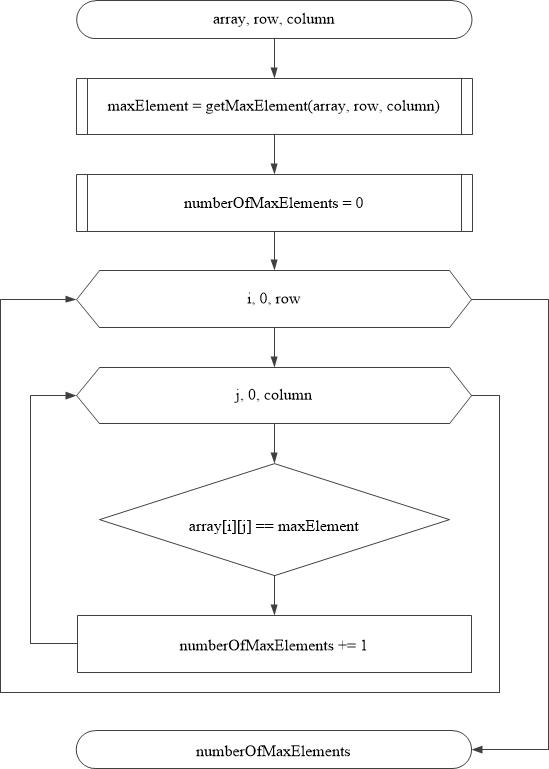


Рисунок 9 – Блок-схема функции getNumberOfMaxElements

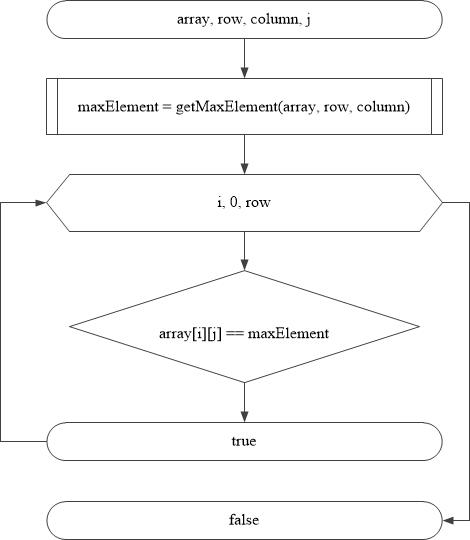


Рисунок 10 – Блок-схема функции isMaxHere

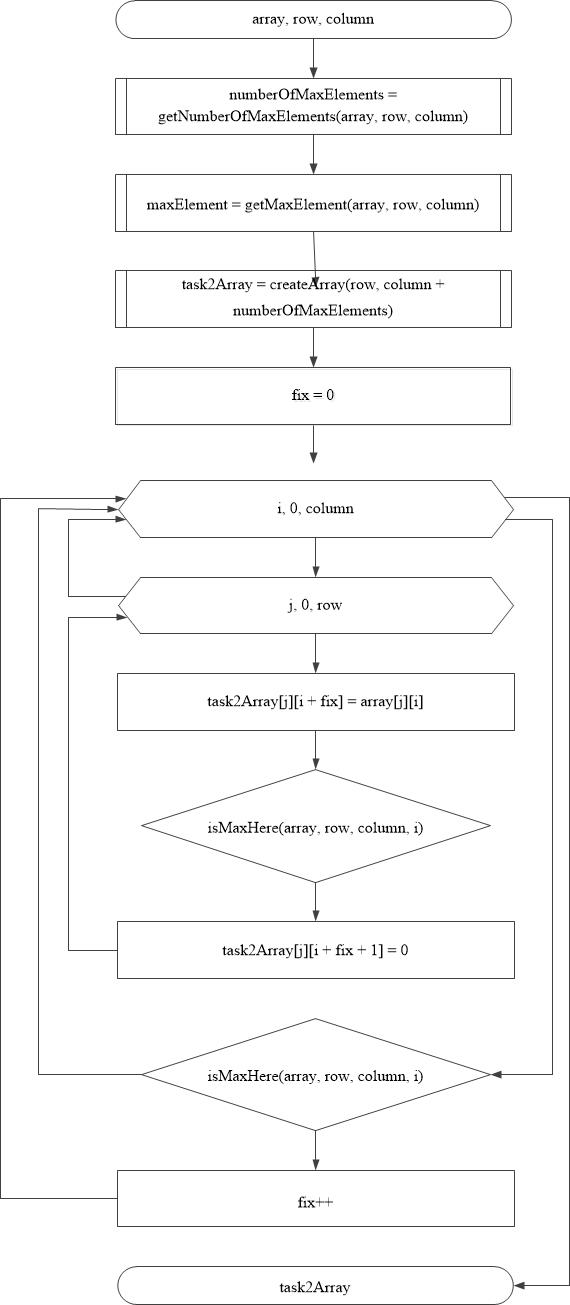


Рисунок 11 – Блок-схема функции task2

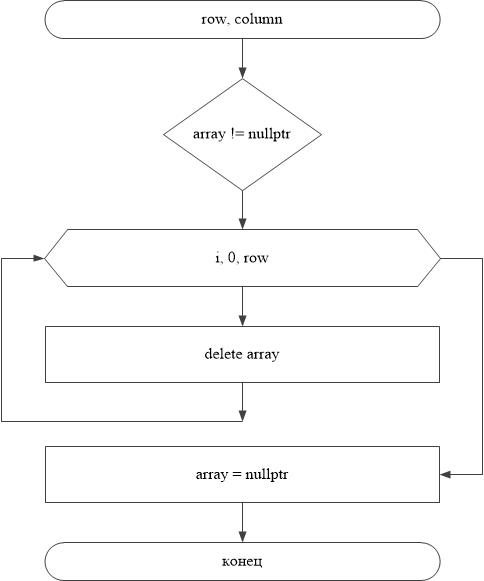


Рисунок 12 – Блок-схема функции deleteArray

Решение задачи на языке программирования C++

#include <iostream>

#include <random>

using namespace std;

/\*

\* \brief Метод заполнения массива

\* \param manualFill - ручной метод заполнения

\* \param randomFill - случайный метод заполнения

\*/

enum class fillArray

{

manualFill,

randomFill

};

/\*

\* \brief Функция, возвращающая размеры массива

\* \param message - подсказка для пользователя

\* \return Возврает размер массива

\*/

size\_t getSize(const string& message);

/\*

\* \brief Функция, определяющая максимального элемента в столбце

\* \param array - массив, в котором будут искать максимальный элемент

\* \param row - количество строк массива

\* \param column - количество столбцов массива

\* \param j - столбец, в котором ищется максимальный элемент

\* \return Возвращает 1, если в столбце есть максимальный элемент, возвращает 0, если нет максимального элемента

\*/

bool isMaxHere(int\*\* array, const size\_t row, const size\_t column, size\_t j);

/\*

\* \brief Функция, удаляющая массива

\* \param array - массива, который необходимо удалить

\* \param row - количество строк массива

\*/

void deleteArray(int\*\*& array, const size\_t row);

/\*

\* \brief Фунция, создающая массив

\* \param row - количество строк массива

\* \param column - количество столбцов массива

\* \return Возвращает указатель на созданный массив

\*/

int\*\* createArray(const size\_t row, const size\_t column);

/\*

\* \brief Функция, определяющая количество максимальных элементов в массиве

\* \param array - указатель на массив

\* \param row - количество строк массива

\* \param column - количество столбцов массива

\* \return Возвращает количество максимальных элементов

\*/

int getNumberOfMaxElements(int\*\* array, size\_t row, size\_t column);

/\*

\* \brief Функция, заменяющая 0 максимальным по модулю элементов массива

\* \param array - указатель на массив

\* \param row - количество строк массива

\* \param column - количество столбцов массива

\* \return Возвращает указатель на массив с замененными 0

\*/

int\*\* task1(int\*\* array, size\_t row, size\_t column);

/\*

\* \brief Функция, определяющая максимальный по модулю элемент

\* \param array - указатель на массив

\* \param row - количество строк массива

\* \param column - количество столбцов массива

\* \return Возвращает максимальный по модулю элемент

\*/

int getMaxElement(int\*\* array, const size\_t row, const size\_t column);

/\*

\* \brief Функция, заполняющая массив вручную

\* \param row - количество строк массива

\* \param column - количество столбцов массива

\* \return Возвращает указатель массива, заполненный вручную

\*/

int\*\* fillManualArray(const size\_t row, const size\_t column);

/\*

\* \brief Функция, заполняющая массив случайно

\* \param row - количество строк массива

\* \param column - количество столбцов массива

\* \param minValue - минимальный диапозон заполнения массива

\* \param maxValue - максимальный диапозон заполнения массива

\* \return Возвращает указатель массива, заполненный случайно

\*/

int\*\* fillRandomArray(const size\_t row, const size\_t column, const int minValue, const int maxValue);

/\*

\* \brief Функция, вставляющая столбец из 0 после столбца, в котором есть максимальный элемент

\* \param array - указатель на массив

\* \param row - количество строк в массиве

\* \param column - количество столбцов в массиве

\* \return Возвращает указатель на массив со вставленными столбцами из 0

\*/

int\*\* task2(int\*\* array, size\_t row, size\_t column);

/\*

\* \brief Функция, выводящая массив на экран

\* \param array - указатель на массив

\* \param row - количество строк в массиве

\* \param column - количество столбцов в массиве

\*/

void printArray(int\*\* array, const size\_t row, const size\_t column);

/\*

\* brief Точка входа в программу

\* return Возращает 0 в случае успеха

\*/

int main()

{

setlocale(LC\_ALL, "Rus");

static const int minValue = -150;

static const int maxValue = 150;

try

{

int row = getSize("Введите количество строк массива\n");

int column = getSize("Введите количество столбцов массива\n");

int\*\* array = createArray(row, column);

cout << "Введите номер операции" << '\n'

<< static\_cast <int>(fillArray::manualFill) << " " << "Заполнение вручную" << '\n'

<< static\_cast <int>(fillArray::randomFill) << " " << "Заполнение случайно" << '\n';

int input = 0;

cin >> input;

const auto choise = static\_cast <fillArray>(input);

switch (choise)

{

case (fillArray::manualFill):

{

array = fillManualArray(row, column);

break;

}

case (fillArray::randomFill):

{

array = fillRandomArray(row, column, minValue, maxValue);

break;

}

default:

{

cout << "Нет такого типа заполнения массива" << endl;

return 1;

break;

}

}

cout << "Начальный массив" << endl;

printArray(array, row, column);

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Массив с заменными нулями" << endl;

int\*\* task1Array = task1(array, row, column);

printArray(task1Array, row, column);

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

int numberOfMaxElements = getNumberOfMaxElements(array, row, column);

int\*\* task2Array = task2(array, row, column);

printArray(task2Array, row, column + numberOfMaxElements);

deleteArray(array, row);

deleteArray(task1Array, row);

deleteArray(task2Array, row);

}

catch (const std::exception& error)

{

cout << error.what();

}

return 0;

}

size\_t getSize(const string& message)

{

int size = -1;

cout << message;

cin >> size;

if (size < 0)

{

throw out\_of\_range("Incorrect size. Value has to be greater or equal zero.");

}

return size;

}

int\*\* createArray(const size\_t row, const size\_t column)

{

int\*\* array = new int\* [row];

for (size\_t i = 0; i < row; i++)

array[i] = new int[column];

return array;

}

int\*\* fillManualArray(const size\_t row, const size\_t column)

{

int\*\* array = createArray(row, column);

for (size\_t i = 0; i < row; i++)

{

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

{

int x;

cin >> x;

array[i][j] = x;

}

}

return array;

}

int\*\* fillRandomArray(const size\_t row, const size\_t column, const int minValue, const int maxValue)

{

int\*\* array = createArray(row, column);

random\_device rd;

mt19937 gen(rd());

uniform\_int\_distribution <> uniformIntDistribution(minValue, maxValue);

for (size\_t i = 0; i < row; i++)

{

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

{

array[i][j] = uniformIntDistribution(gen);

}

}

return array;

}

void printArray(int\*\* array, const size\_t row, const size\_t column)

{

for (size\_t i = 0; i < row; i++)

{

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

{

cout << array[i][j] << '\t';

}

cout << '\n';

}

}

int getMaxElement(int\*\* array, const size\_t row, const size\_t column)

{

int maxElement = array[0][0];

for (size\_t i = 0; i < row; i++)

{

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

{

if (std::abs(array[i][j]) > std::abs(maxElement))

{

maxElement = array[i][j];

}

}

}

return maxElement;

}

int\*\* task1(int\*\* array, size\_t row, size\_t column)

{

int\*\* task1Array = createArray(row, column);

for (size\_t i = 0; i < row; i++)

{

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

{

if (array[i][j] == 0)

{

task1Array[i][j] = getMaxElement(array, row, column);

}

else

{

task1Array[i][j] = array[i][j];

}

}

}

return task1Array;

}

int getNumberOfMaxElements(int\*\* array, size\_t row, size\_t column)

{

int maxElement = getMaxElement(array, row, column);

int numberOfMaxElements = 0;

for (size\_t i = 0; i < row; i++)

{

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

{

if (array[i][j] == maxElement)

{

numberOfMaxElements += 1;

}

}

}

return numberOfMaxElements;

}

bool isMaxHere(int\*\* array, const size\_t row, const size\_t column, size\_t j)

{

int maxElement = getMaxElement(array, row, column);

for (size\_t i = 0; i < row; i++)

{

if (j > column)

{

return false;

}

if (array[i][j] == maxElement)

{

return true;

}

}

return false;

}

int\*\* task2(int\*\* array, size\_t row, size\_t column)

{

int numberOfMaxElements = getNumberOfMaxElements(array, row, column);

int maxElement = getMaxElement(array, row, column);

int\*\* task2Array = createArray(row, column + numberOfMaxElements);

int fix = 0;

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

{

for (size\_t j = 0; j < row; j++)

{

task2Array[j][i + fix] = array[j][i];

if (isMaxHere(array, row, column, i))

{

task2Array[j][i + fix + 1] = 0;

}

}

if (isMaxHere(array, row, column, i))

{

fix++;

}

}

return task2Array;

}

void deleteArray(int\*\*& array, const size\_t row)

{

if (array != nullptr)

{

for (size\_t i = 0; i < row; i++)

{

delete[] array[i];

}

delete[] array;

array = nullptr;

}

}

1. Решение тестовых примеров

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13 – Решение тестового примера

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 14** – **Отрицательный размер массива**

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 15** – **Ввод массива нулевого размера**

1. Зачет задания в GitHub



Рисунок 16 – Зачет задания