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

Борцов Ю. Д.

Проверил: к.т.н., доц.

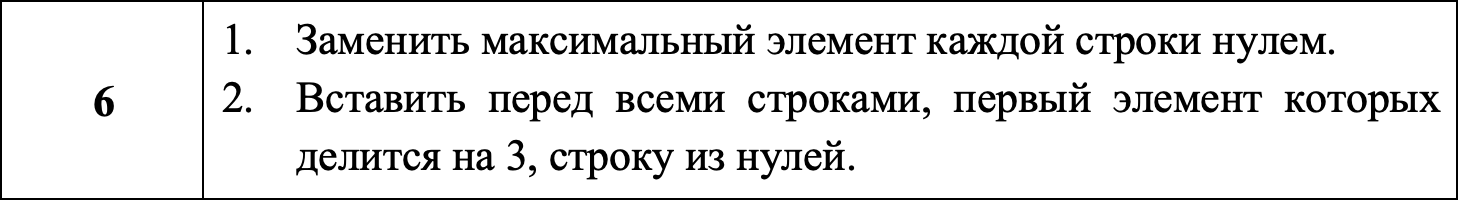
Васильева М. А.

Москва 2022

**Задание 4-3**

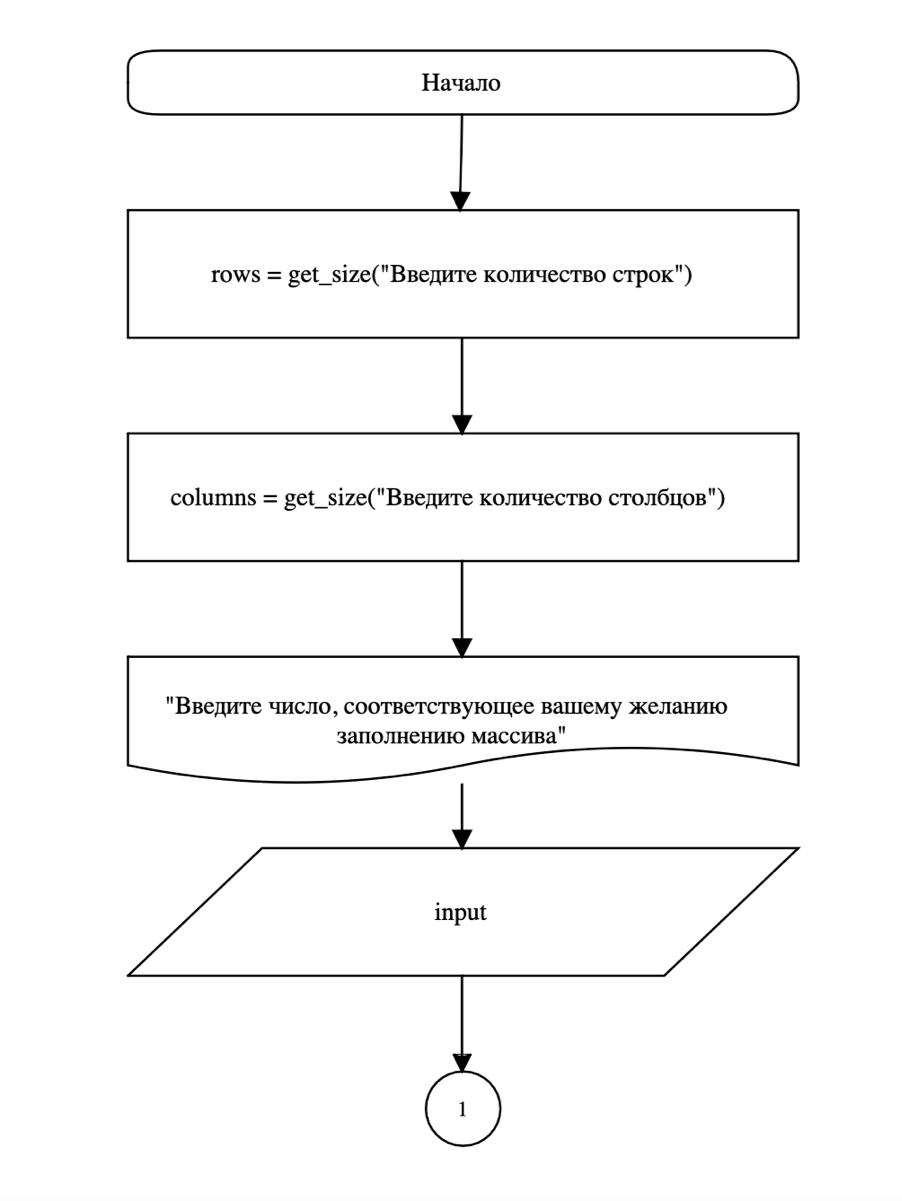
**1. Формулировка задачи**

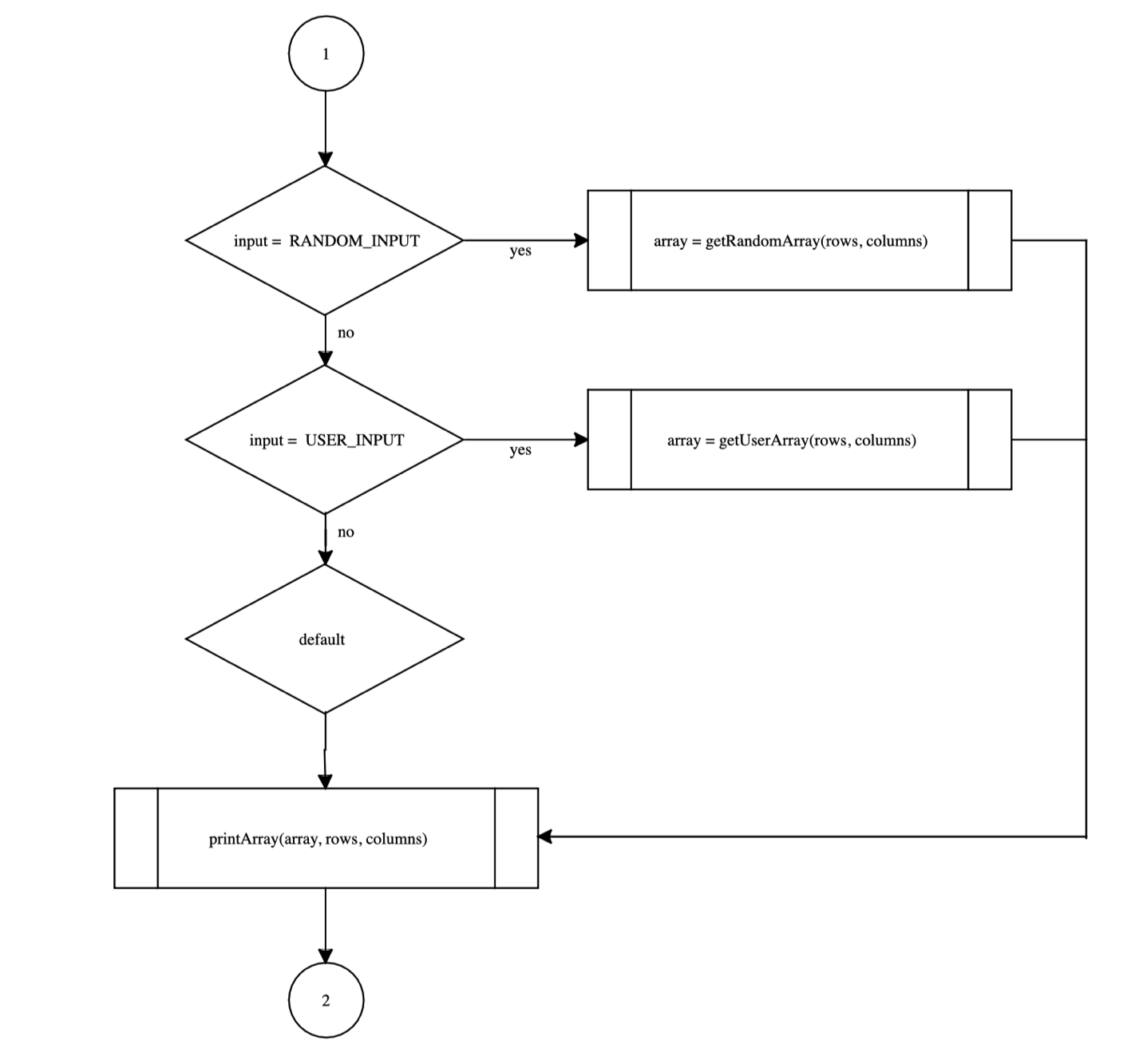
Создать многомерный массив nˣm из n целых чисел. Реализовать возможность заполнения массива, как случайными числами, так и с помощью клавиатуры по желанию пользователя. Пункты задания организовать в виде функций (методов). Вывести массив на экран.

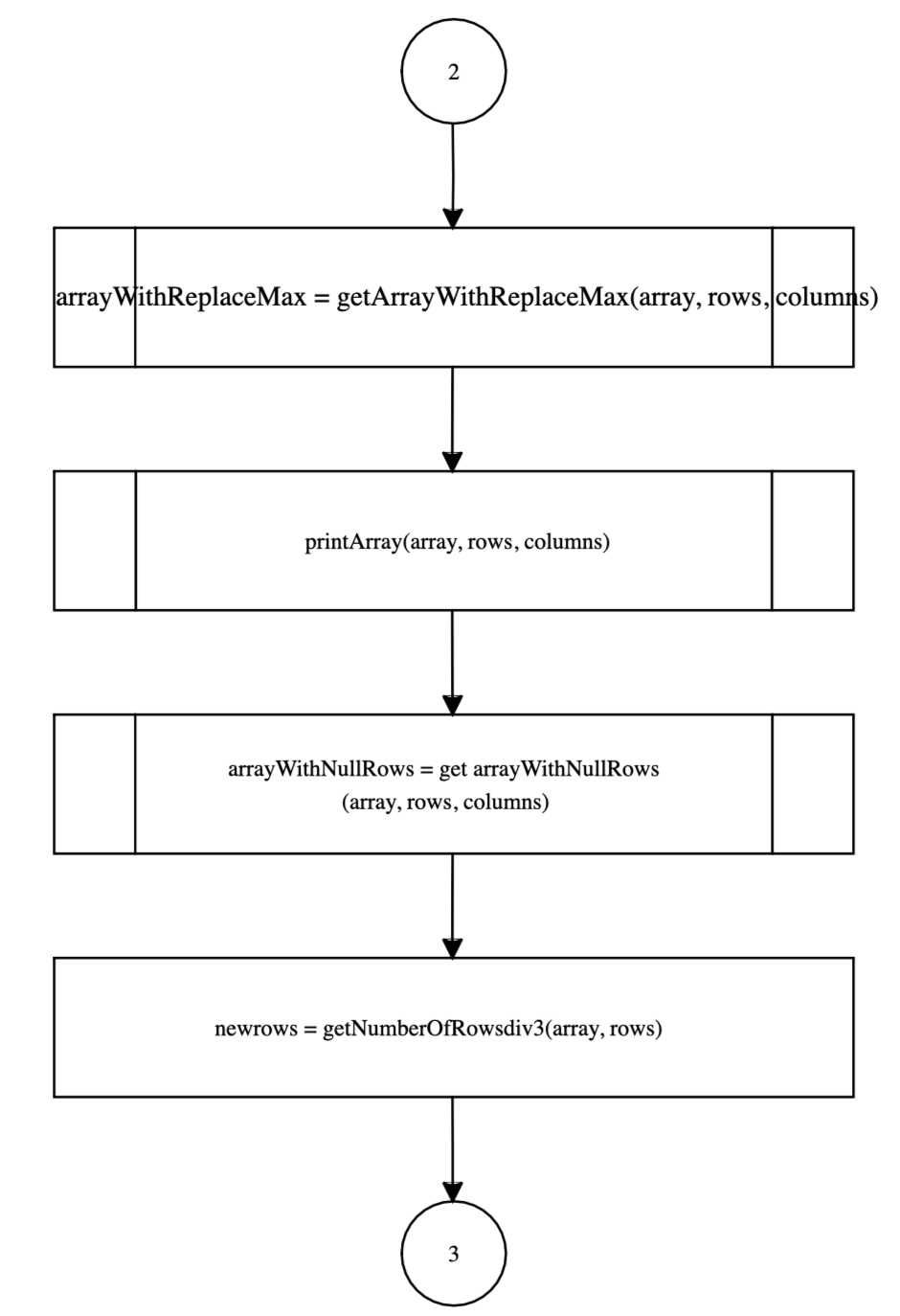


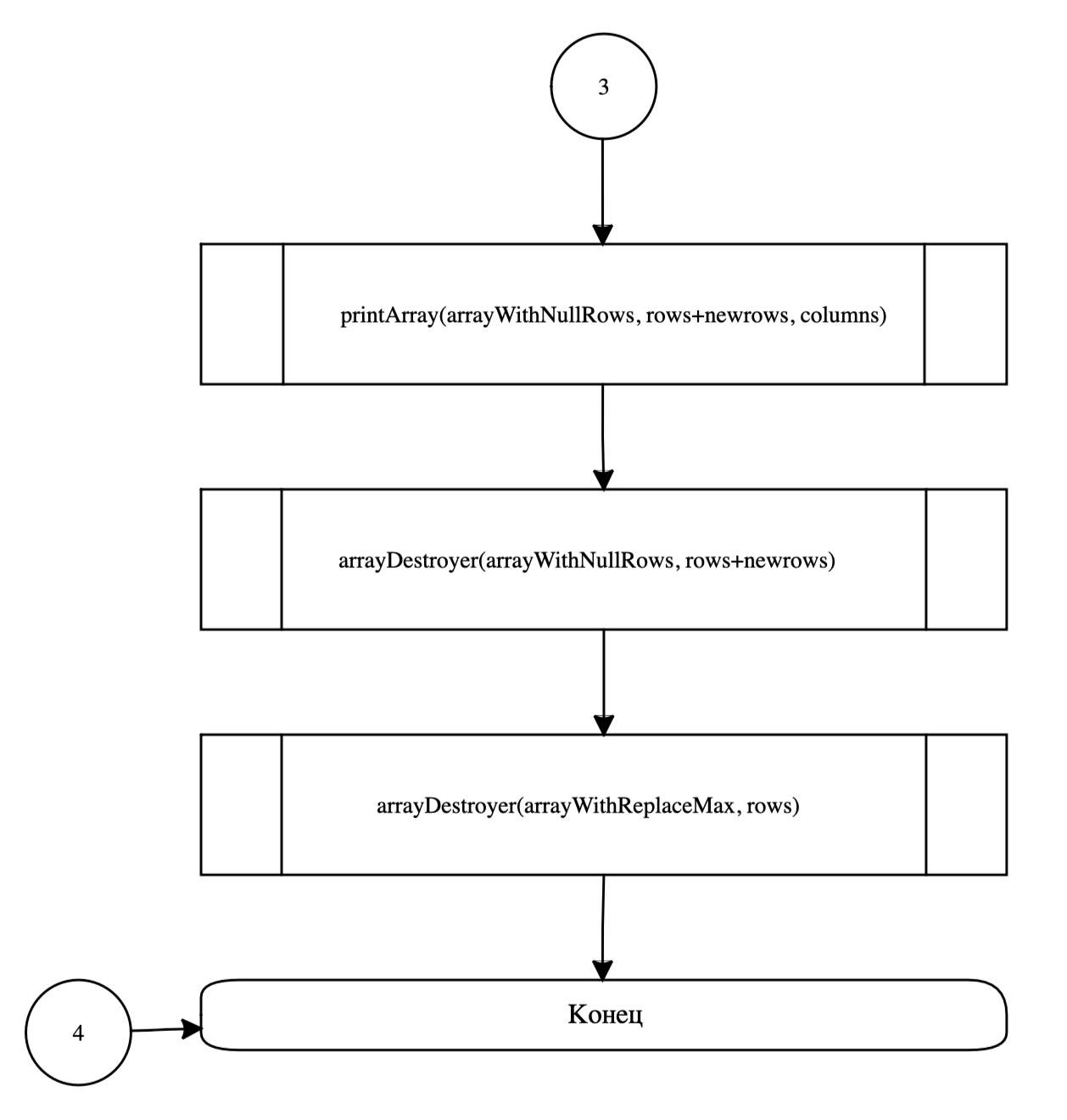
**2. Блок-схемы алгоритмов функций представлены на рисунках**

**(Рисунок 1-13)**

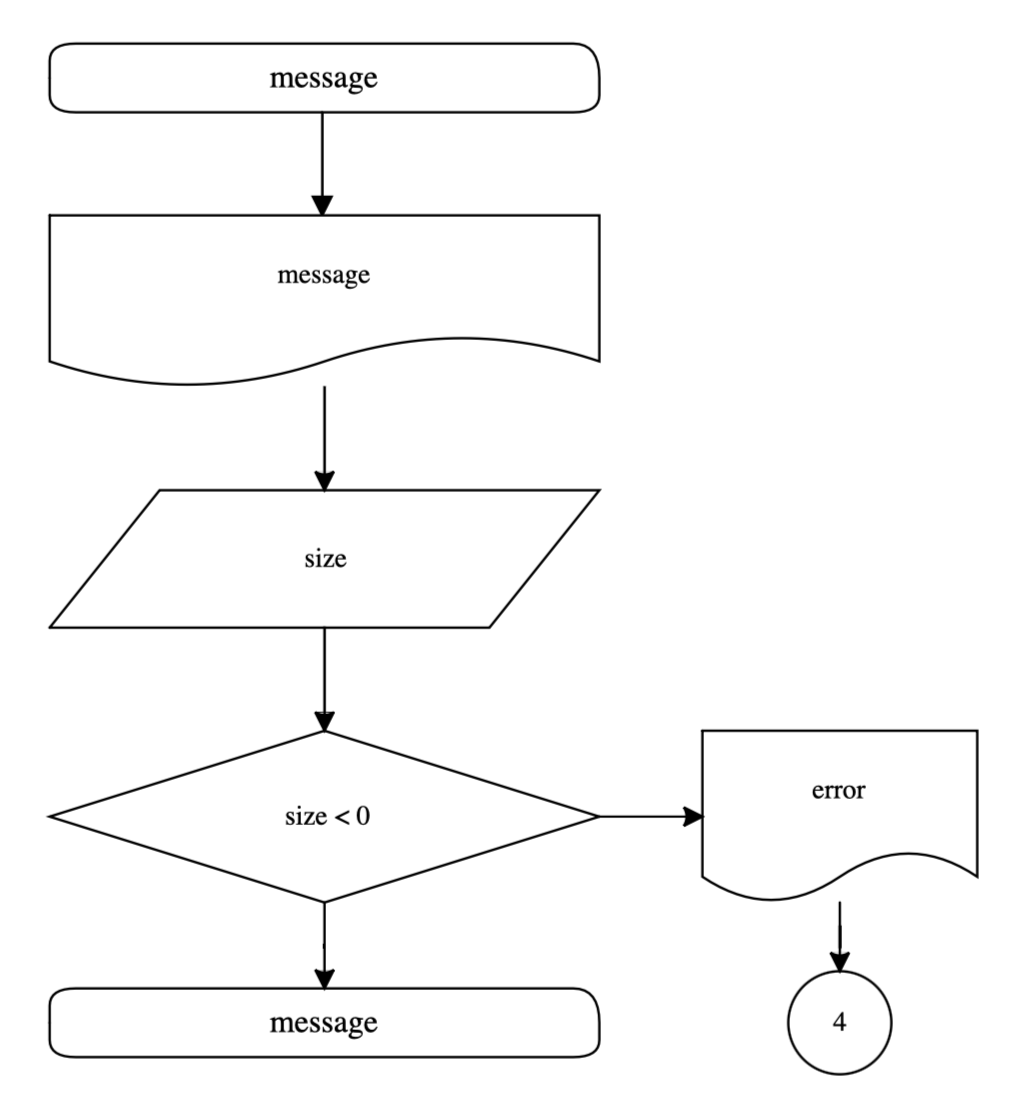




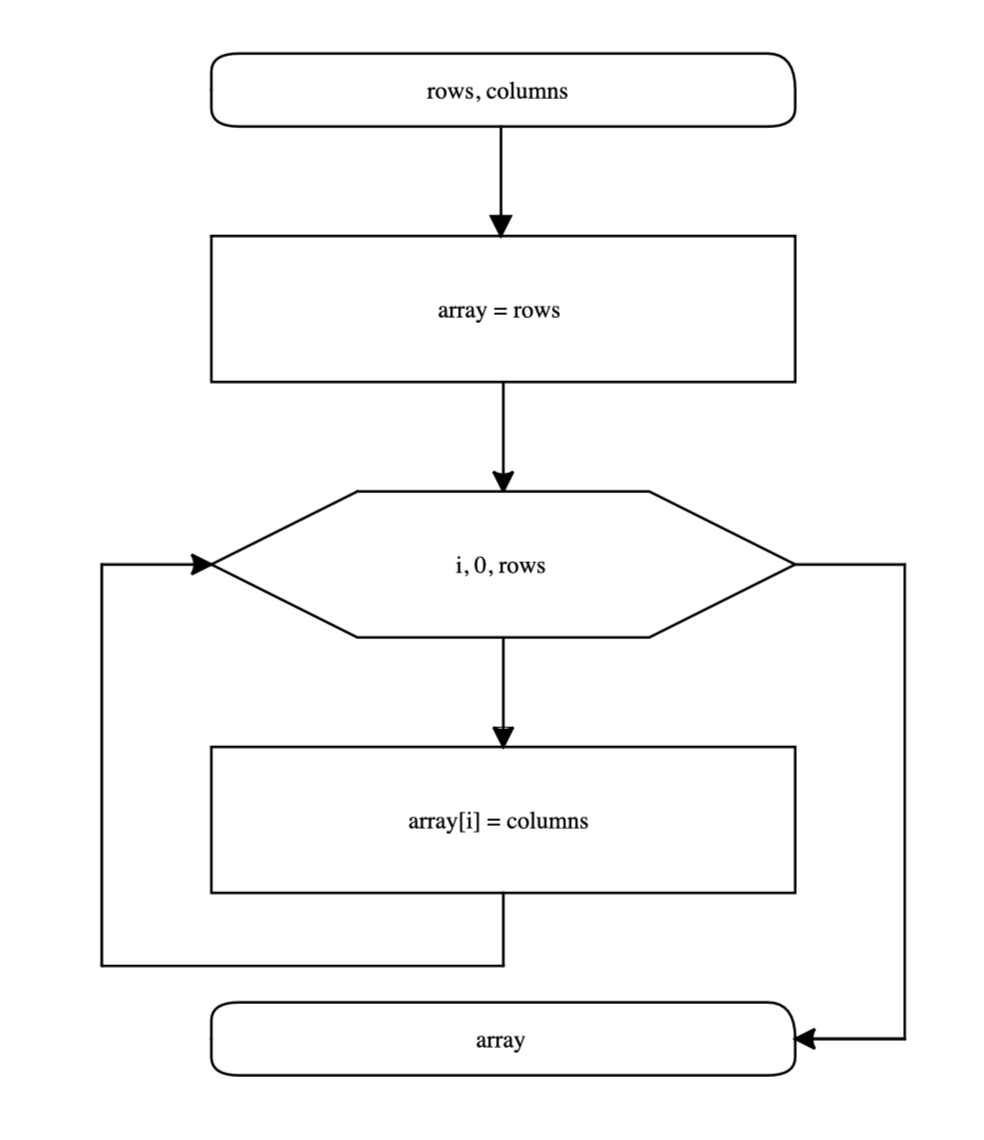




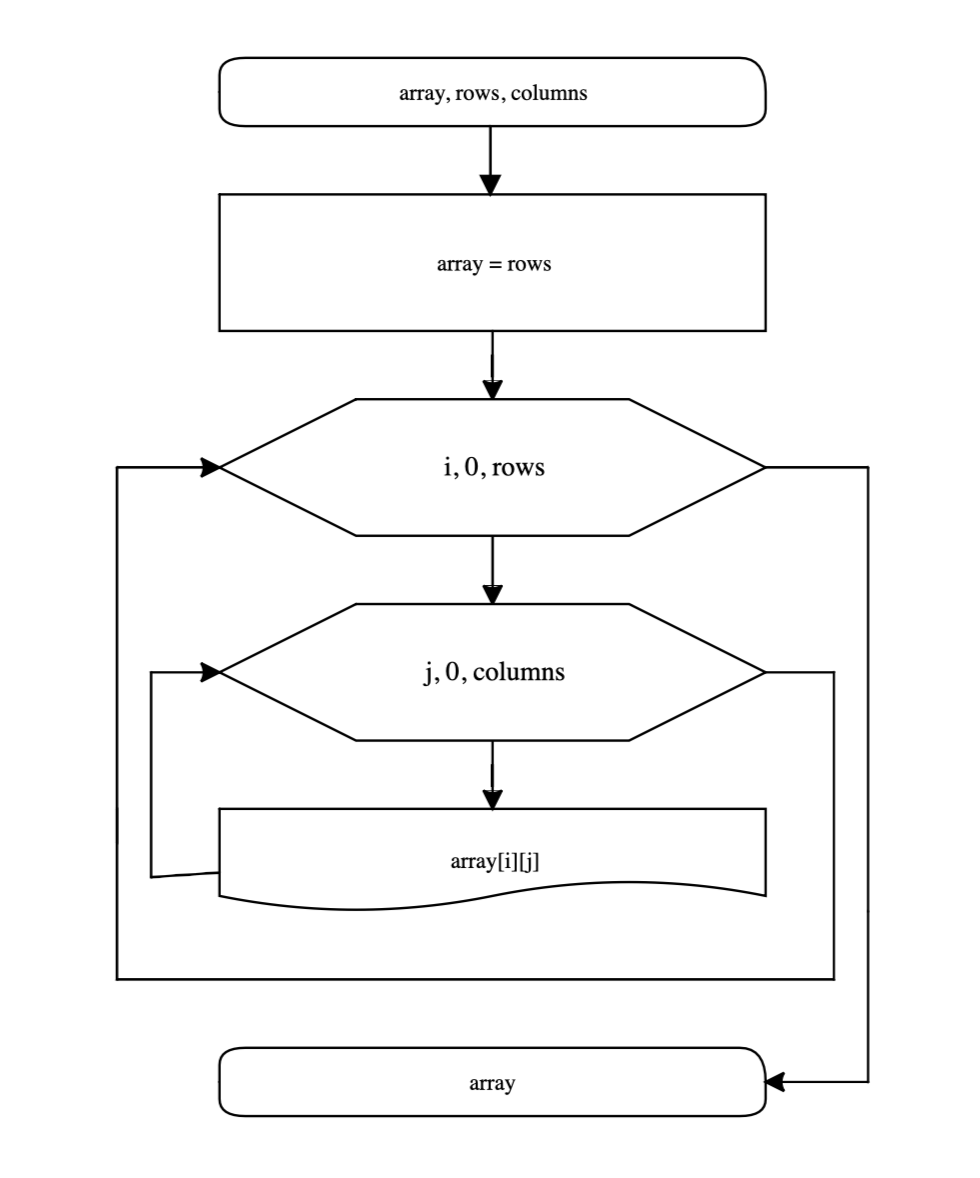
**Рисунок 1 – Блок-схема функции main**



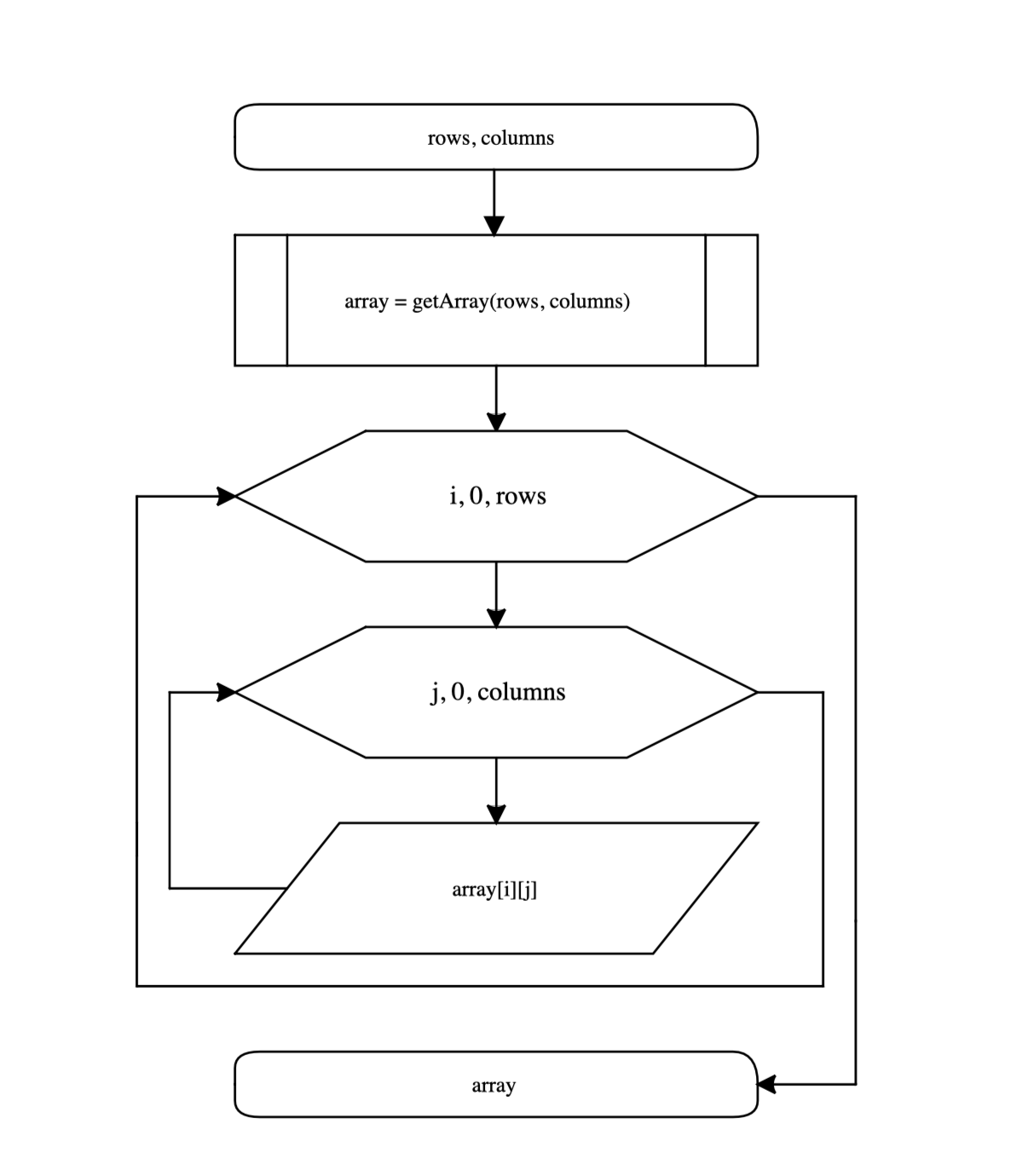
**Рисунок 2 – Блок-схема функции get\_size**



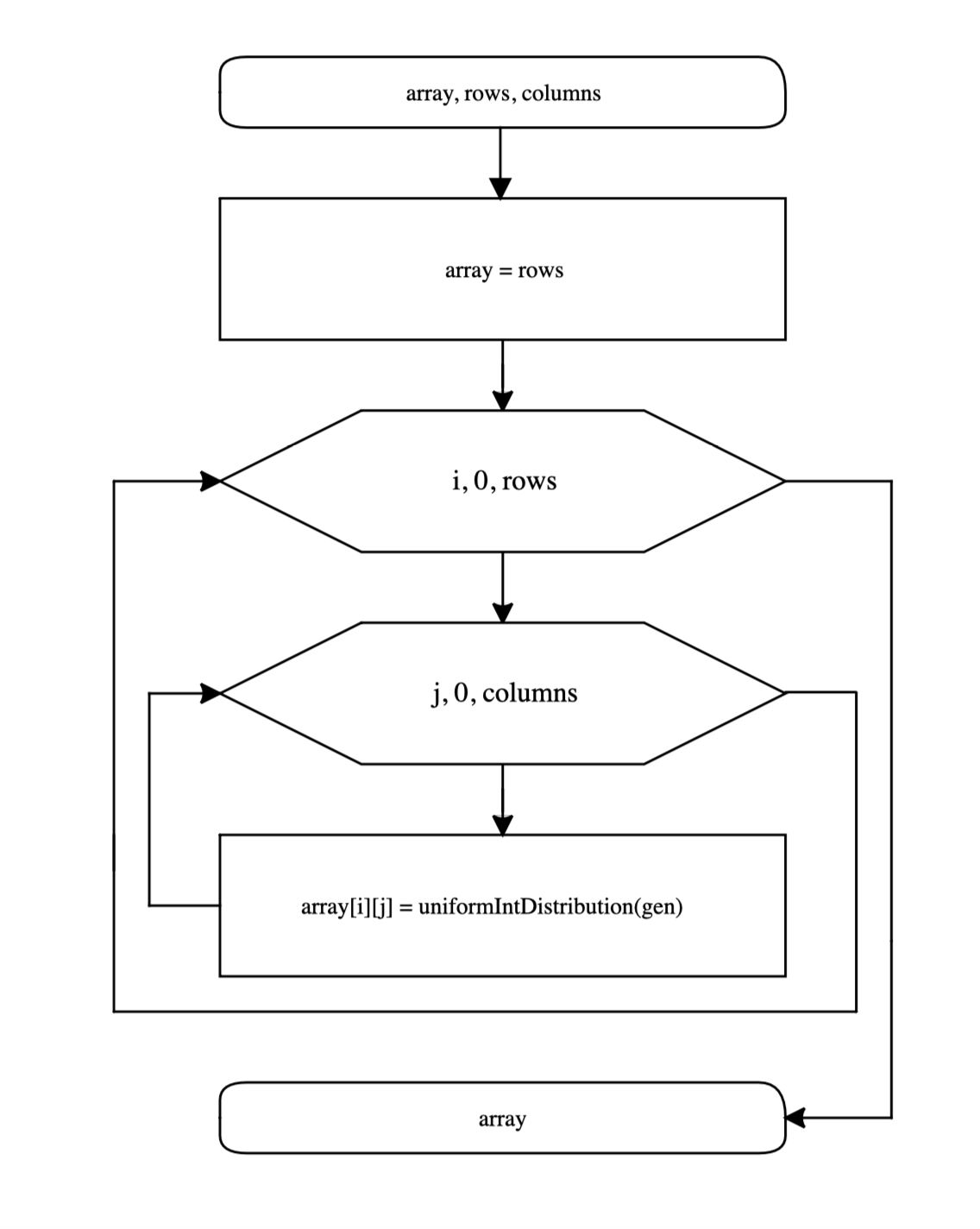
**Рисунок 3 – Блок-схема функции getArray**



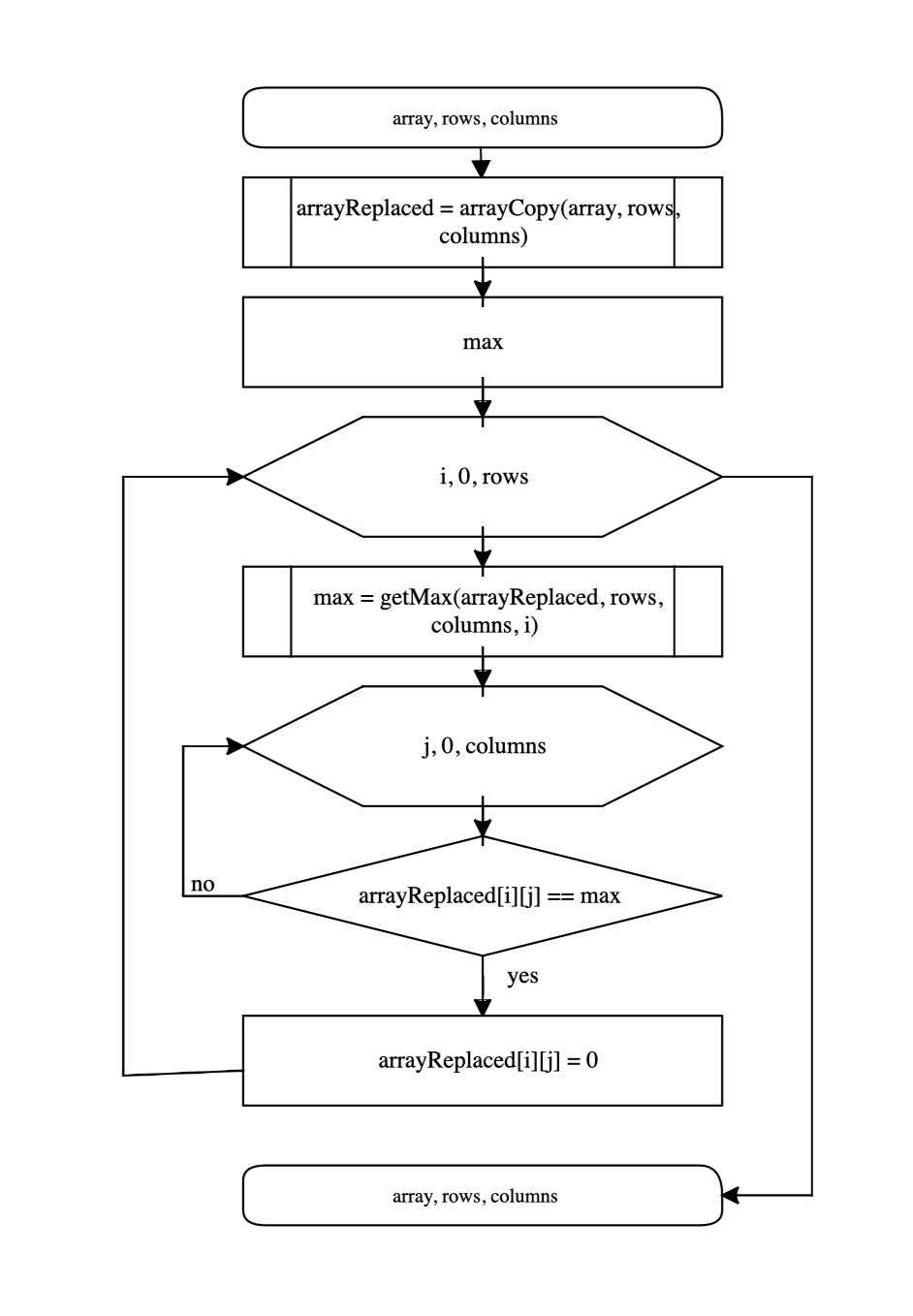
**Рисунок 4 – Блок-схема функции printArray**



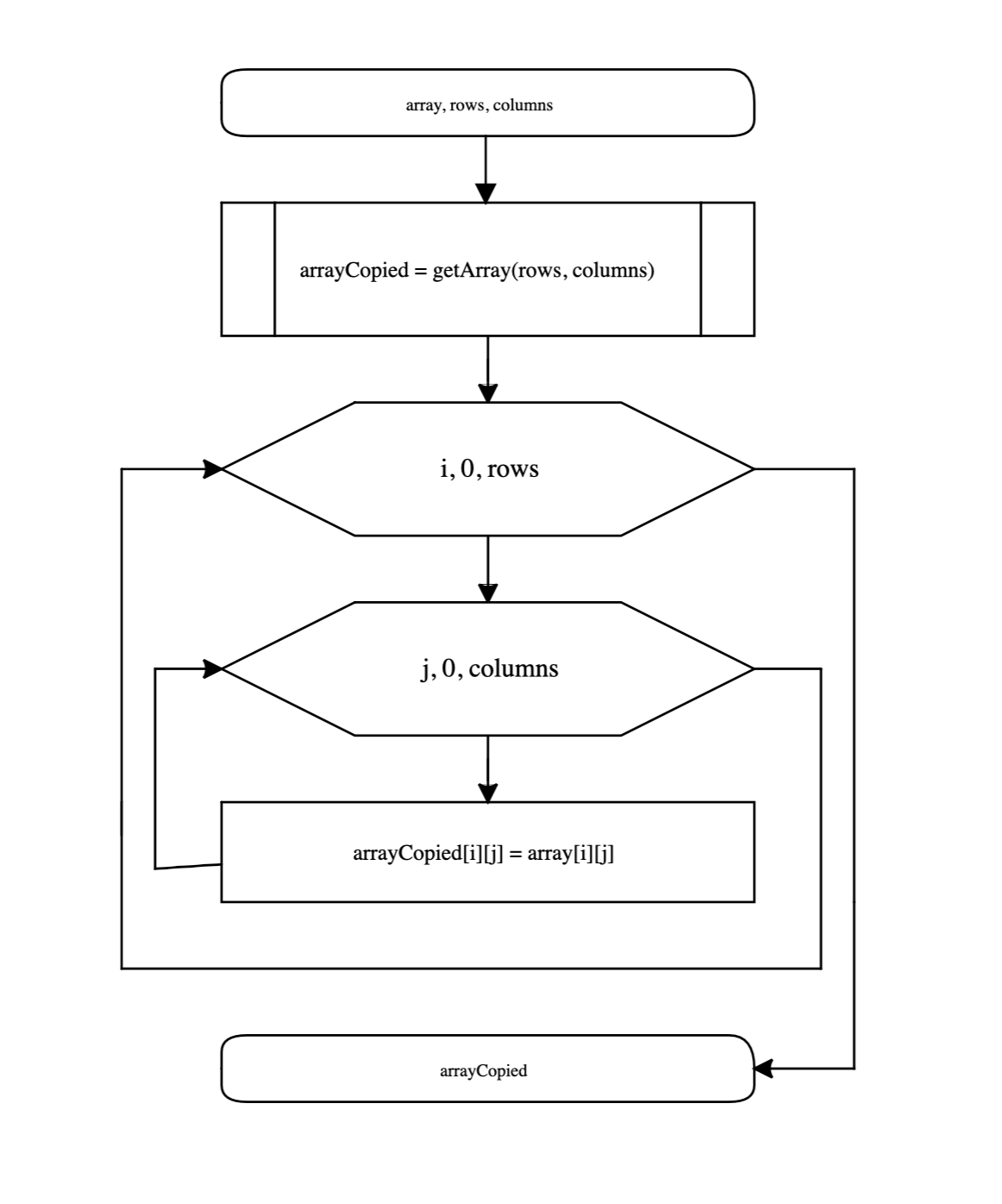
**Рисунок 5 – Блок-схема функции getUserArray**



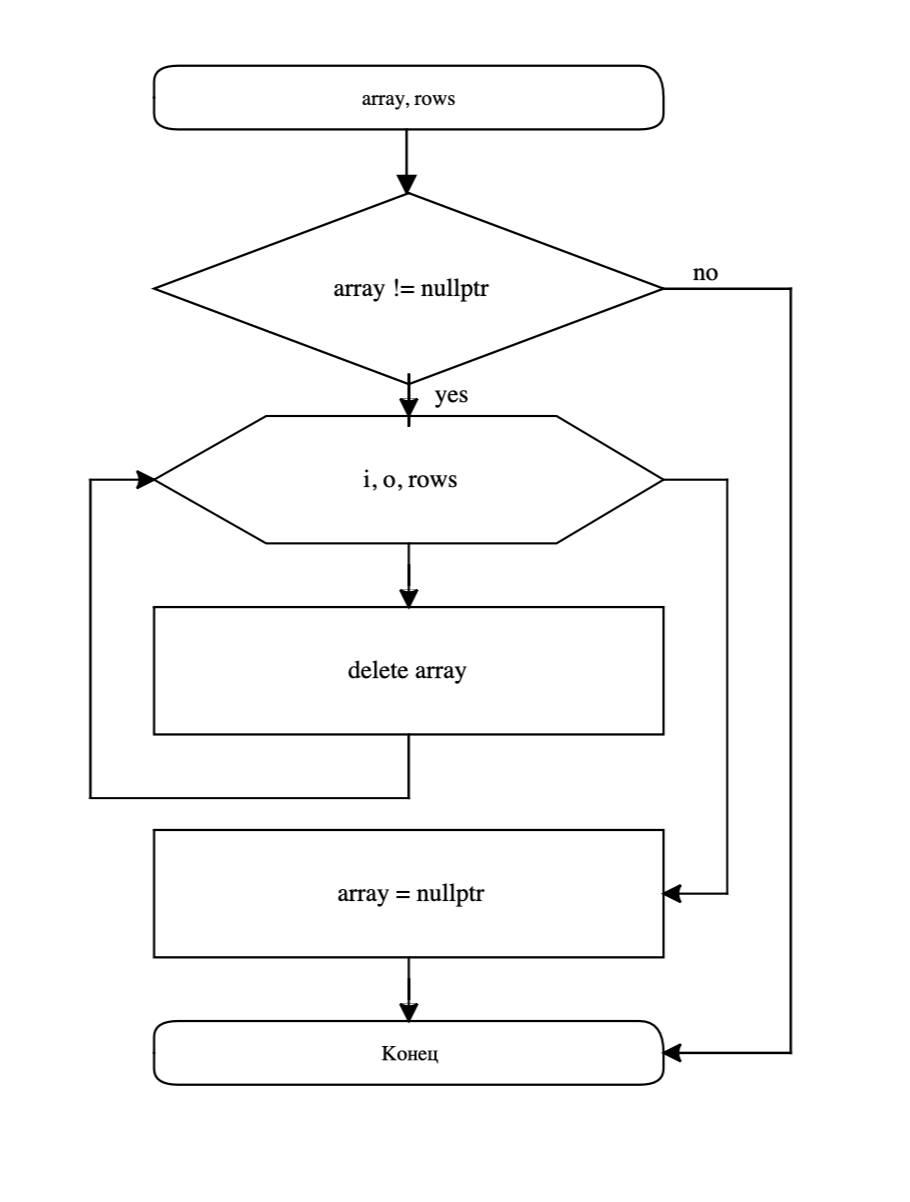
**Рисунок 6 – Блок-схема функции getRandonArray**



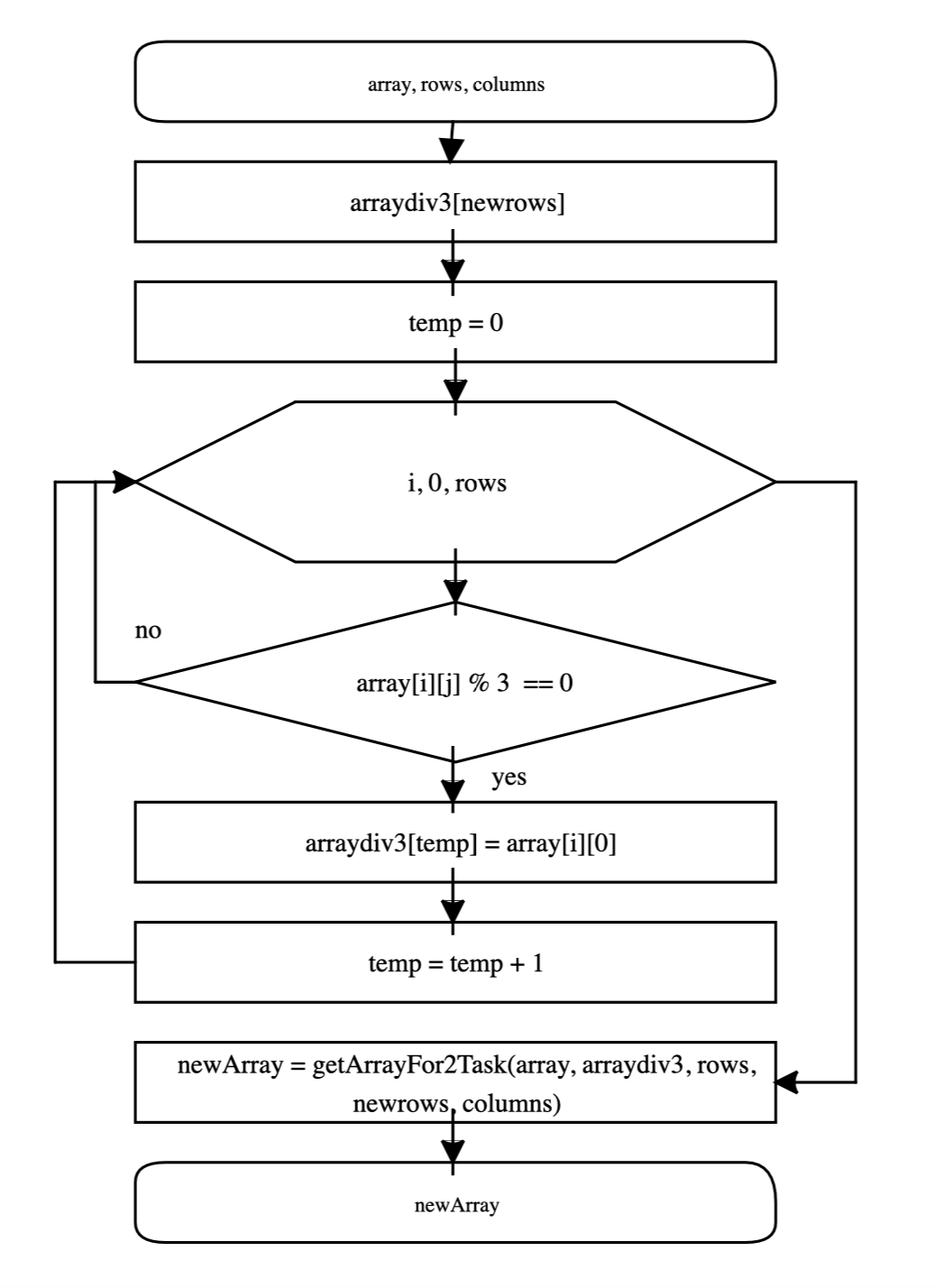
**Рисунок 7 – Блок-схема функции getArrayWithReplacedMax**

****

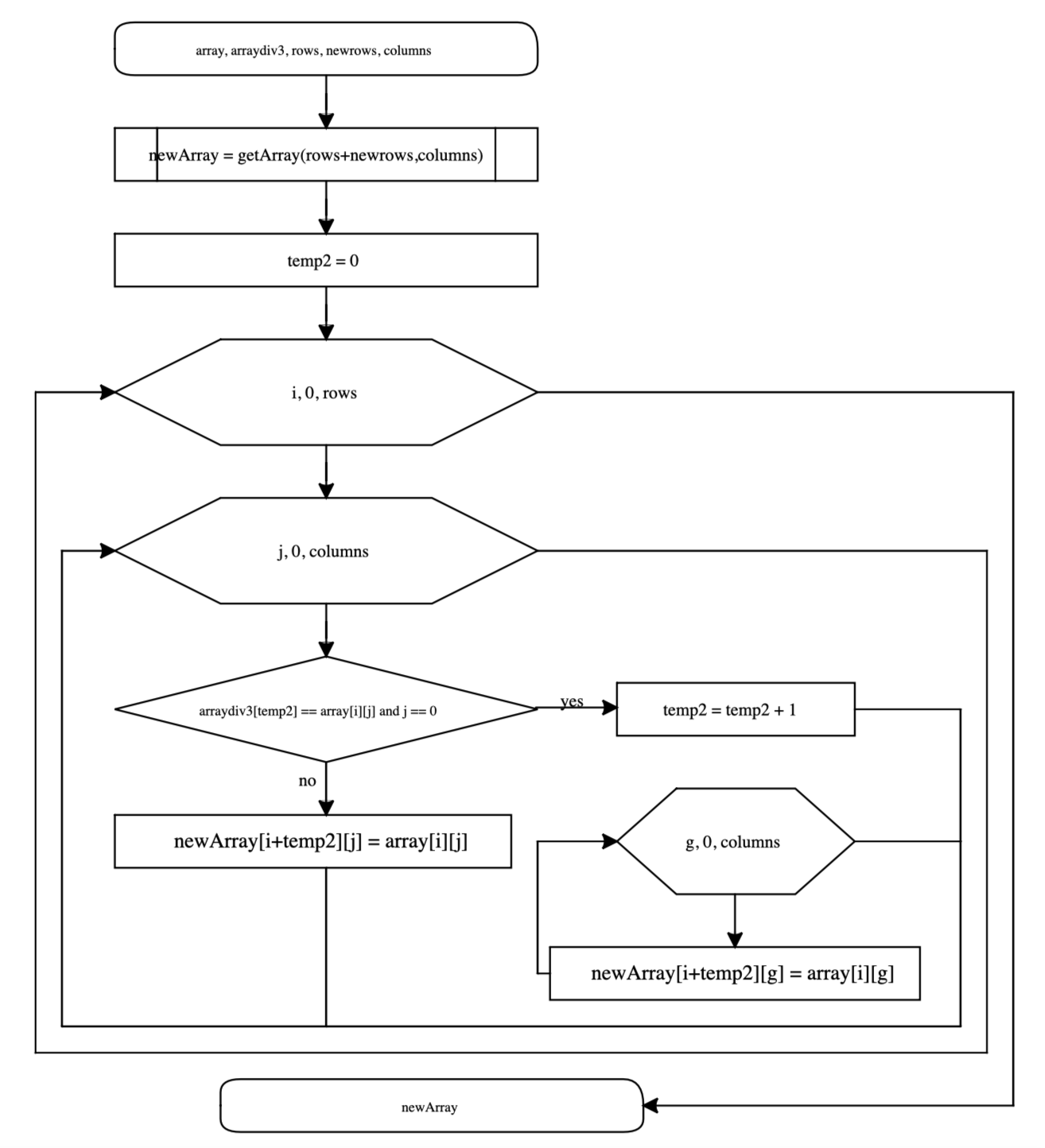
**Рисунок 8 – Блок-схема функции arrayCopy**



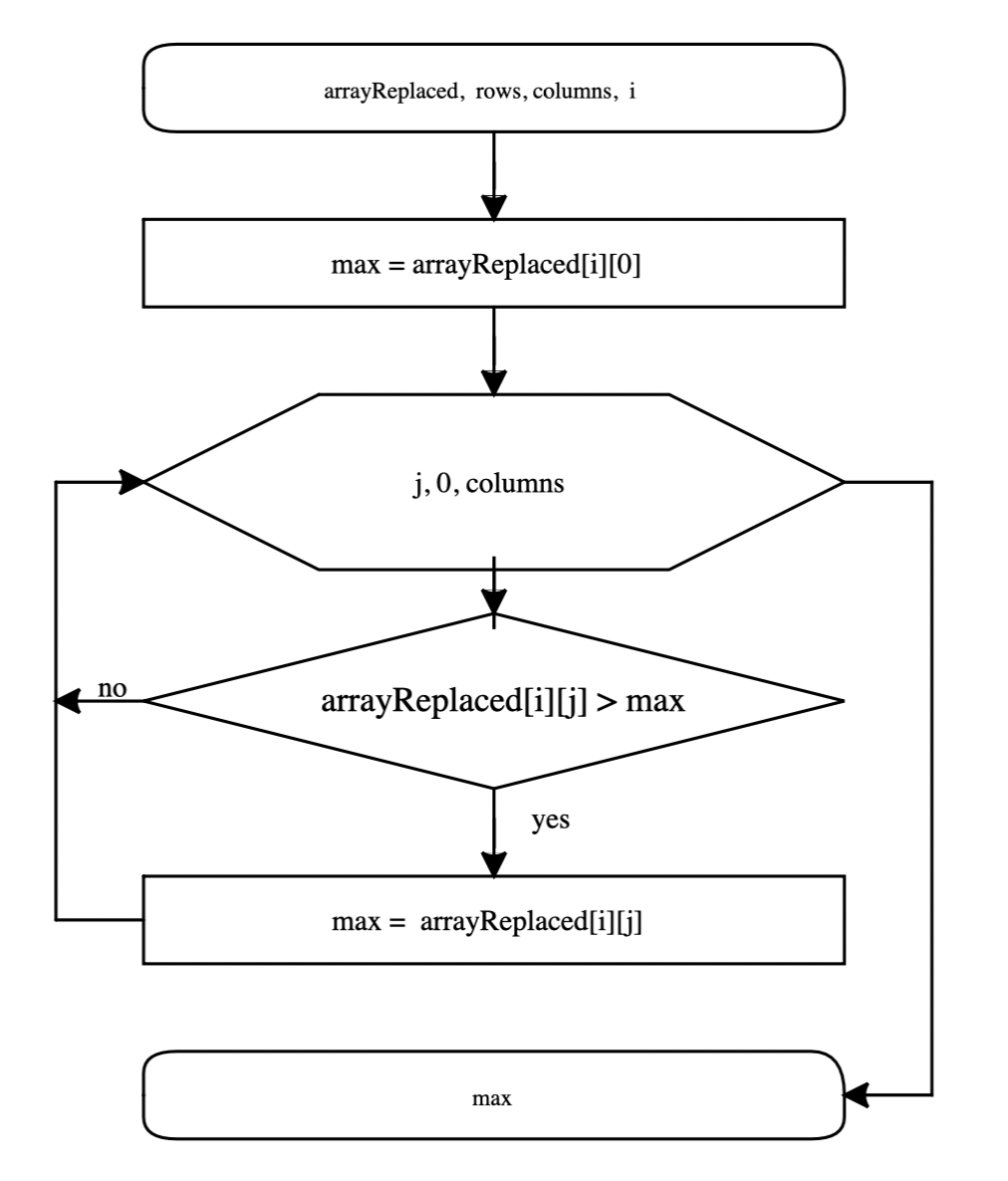
**Рисунок 9 – Блок-схема функции arrayDestroyer**



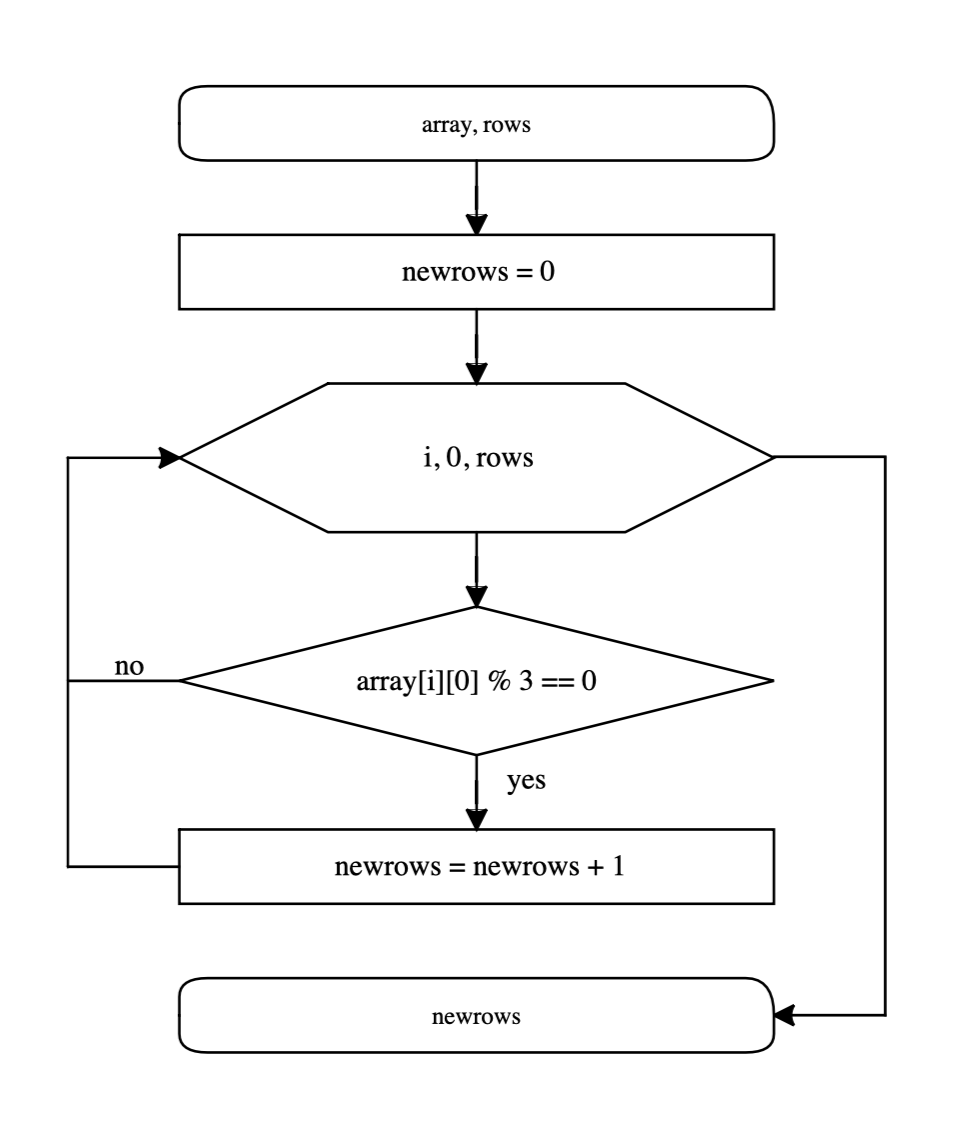
**Рисунок 10 – Блок-схема функции getArrayWithNullRows**



**Рисунок 11 – Блок-схема функции getArrayFor2Task**



**Рисунок 12 – Блок-схема функции getMax**



**Рисунок 13 – Блок-схема функции getNumberOfRowsdiv3**

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

#include <iostream>

#include <ctime>

#include <random>

using namespace std;

/\*\*

\* \brief Считывает значение числа из консоли.

\* \param message Побуждающее сообщение для пользователя.

\* \return Число.

\*/

size\_t get\_size(const string& message);

/\*\*

\* \brief Создаёт двумерный массив.

\* \param rows Количество строк в массиве.

\* \param columns Количество столбцов массива.

\* \return Возвращает указатель на массив.

\*/

int\*\* getArray(const size\_t rows, const size\_t columns);

/\*\*

\* \brief Создает массив чисел, которые вводит пользователь.

\* \param rows Число строк массива.

\* \param columns Число столбцов массива.

\* \return Возвращает указатель на массив, заполненный пользовательскими числами.

\*/

int\*\* getUserArray(const size\_t rows, const size\_t columns);

/\*\*

\* \brief Заполняет массив случайными числами от -100 до 100.

\* \param rows Количество строк в массиве.

\* \param columns Количество столбцов в массиве.

\* \return Возвращает указатель на массив, заполненный случайными числами.

\*/

int\*\* getRandomArray(const size\_t rows, const size\_t columns);

/\*\*

\* \brief Выводит массив в консоли.

\* \param array Указатель на массив.

\* \param rows Количество строк в массиве.

\* \param columns Количество столбцов в массиве.

\*/

void printArray(int\*\* array, const size\_t rows, const size\_t columns);

/\*\*

\* \brief Выбор варианта заполнения массива.

\* \param USER\_INPUT Пользователь заполняет массив.

\* \param RANDOM\_INPUT Массив заполняется случайными числами.

\*/

enum class userInput

{

USER\_INPUT = 1,

RANDOM\_INPUT

};

/\*\*

\* \brief Передает указатель на массив, в котором самое большое число строки заменяется на ноль.

\* \param array Указатель на массив.

\* \param rows Количество строк в массиве.

\* \param columns Количество столбцов в массиве.

\* \return Возвращает указатель на массив, с замененными числами.

\*/

int\* getIndexMaxInRow(int\*\* array, const size\_t rows, const size\_t columns);

/\*\*

\* \brief Копирует массив.

\* \param array Указатель на массив.

\* \param rows Количество строк в массиве.

\* \param columns Количество столбцов в массиве.

\* \return Возвращает указатель на скопированный массив.

\*/

int\*\* arrayCopy(int\*\* array, const size\_t rows, const size\_t columns);

/\*\*

\* \brief Создает массив, в котором перед строками, первый элемент которых делится на 3, вставлена строка нулей.

\* \param array Указатель на массив.

\* \param rows Количество строк в массиве.

\* \param columns Количество столбцов в массиве.

\* \return Указатель на новый массив.

\*/

int\*\* getArrayWithNullRows(int\*\* array, const size\_t rows, const size\_t columns);

/\*\*

\* \brief Создает массив, в котором каждый максимальный элемент строки заменен нулем.

\* \param array Указатель на массив.

\* \param rows Количество строк массива.

\* \param columns Количество столбцов массива.

\* \return Указатель на массив, в котором каждый максимальный элемент строки заменен нулем.

\*/

int\*\* getArrayWithReplaceMax(int\*\* array, const size\_t rows, const size\_t columns);

/\*\*

\* \brief Определяет максимальный элемент в строке массива.

\* \param arrayReplaced Указатель на двумерный массив.

\* \param rows Количество строк массива.

\* \param columns Количество столбцов массива.

\* \param i Номер строки массива, в которой нужно найти максимальный элемент.

\* \return Вовзращает значение максимального элемента строки i.

\*/

int getMax(int\*\* arrayReplaced, const size\_t rows, const size\_t columns, const size\_t i);

/\*\*

\* \brief Определяет количество строк массива, в которых первый элемент кратен 3.

\* \param array Указатель на двумерный массив.

\* \param rows Количество строк массива.

\* \return Возвращает число строк, первый элемент которых кратен 3.

\*/

int getNumberOfRowsdiv3(int\*\* array, const size\_t rows);

/\*\*

\* \brief Создает конечный массив для задания 2.

\* \param aray Указатель на изначальный массив.

\* \param arraydiv3 Указатель на список из первых элементов строк массива, которые делятся на 3.

\* \param rows Количество строк изначального массива

\* \param newrows Количество строк, которые будут содержать нули.

\* \param columns Количество столбцов массива.

\*/

int\*\* getArrayFor2Task(int\*\* array, const int\* arraydiv3, const size\_t rows, const size\_t newrows, const size\_t columns);

/\*\*

\* \brief Удаляет созданный двумерный массив.

\* \param array Указатель на двумерный массив.

\* \param rows Количество строк в массиве.

\*/

void arrayDestroyer(int\*\*& array, const size\_t rows);

/\*\*

\* \brief Точка входа в программу.

\* \return Возвращает 0 в случае успеха.

\*/

int main()

{

size\_t rows = 0;

size\_t columns = 0;

try

{

rows = get\_size("Введите количество строк массива: ");

columns = get\_size("Введите количество столбцов массива: ");

}

catch (std::out\_of\_range& e)

{

std::cerr << e.what();

return 1;

}

cout << "\n";

cout << "Введите число, соответствующее вашему желанию заполнения массива:\n" << "\n"

<< static\_cast<int>(userInput::USER\_INPUT) << " - вы заполняете массив вручную.\n"

<< static\_cast<int>(userInput::RANDOM\_INPUT) << " - заполняет массив случайными числами.\n\n";

int input = 0;

cout << "Ваше число выбора варианта заполнения массива: ";

cin >> input;

const auto choice = static\_cast<userInput>(input);

cout << "\n";

int\*\* array = nullptr;

switch(choice)

{

case userInput::USER\_INPUT:

{

array = getUserArray(rows, columns);

break;

}

case userInput::RANDOM\_INPUT:

{

array = getRandomArray(rows, columns);

break;

}

default:

{

cerr << "Введено неверное значение\n";

}

}

// Вывод начального массива

cout << "Начальный массив:" << "\n" << "\n";

printArray(array, rows, columns);

cout << "\n";

// Вывод массива, в котором каждый максимальный элемент строки заменен нулем.

cout << "Массив, в котором каждый максимальный элемент строки заменен нулем: " << "\n" << "\n";

int\*\* arrayWithReplaceMax = getArrayWithReplaceMax(array, rows, columns);

printArray(arrayWithReplaceMax, rows, columns);

cout << "\n";

// Вывод массива, в котором перед строками, первый элемент которых делится на 3, вставлена строка нулей.

cout << "Массив, в котором перед строками, первый элемент которых делится на 3, вставлена строка нулей: " << "\n" << "\n";

int\*\* arrayWithNullRows = getArrayWithNullRows(array, rows, columns);

int newrows = getNumberOfRowsdiv3(array, rows);

printArray(arrayWithNullRows, rows+newrows, columns);

arrayDestroyer(arrayWithNullRows, rows+newrows);

arrayDestroyer(arrayWithReplaceMax, rows);

return 0;

}

size\_t get\_size(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 static\_cast<size\_t>(size);

}

int\*\* getArray(const size\_t rows, const size\_t columns)

{

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

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

{

array[i] = new int[columns];

}

return array;

}

void printArray(int\*\* array, const size\_t rows, const size\_t columns)

{

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

{

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

{

cout.width(3);

cout << array[i][j] << "\t";

}

cout << "\n" << "\n";

}

}

int\*\* getUserArray(const size\_t rows, const size\_t columns)

{

int\*\* array = getArray(rows, columns);

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

{

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

{

cin >> array[i][j];

}

cout << "\n";

}

return array;

}

int\*\* getRandomArray(const size\_t rows, const size\_t columns)

{

const int minValue = -100;

const int maxValue = 100;

int\*\* array = getArray(rows, columns);

random\_device rd;

mt19937 gen(rd());

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

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

{

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

{

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

}

}

return array;

}

int\*\* getArrayWithReplaceMax(int\*\* array, const size\_t rows, const size\_t columns)

{

int\*\* arrayReplaced = arrayCopy(array, rows, columns);

int max;

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

{

max = getMax(arrayReplaced, rows, columns, i);

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

{

if (arrayReplaced[i][j] == max)

{

arrayReplaced[i][j] = 0;

}

}

}

return arrayReplaced;

}

int\*\* arrayCopy(int\*\* array, const size\_t rows, const size\_t columns)

{

int\*\* arrayCopied = getArray(rows, columns);

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

{

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

{

arrayCopied[i][j] = array[i][j];

}

}

return arrayCopied;

}

void arrayDestroyer(int\*\*& array, const size\_t rows)

{

for (int i = 0; i < rows; i++)

{

delete[] array[i];

}

if (array != nullptr)

{

delete[] array;

array = nullptr;

}

}

int\*\* getArrayWithNullRows(int\*\* array, const size\_t rows, const size\_t columns)

{

int newrows = getNumberOfRowsdiv3(array, rows);

int arraydiv3[newrows];

size\_t temp = 0;

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

{

if ((array[i][0] % 3) == 0)

{

arraydiv3[temp] = array[i][0];

temp++;

}

}

int\*\* newArray = getArrayFor2Task(array, arraydiv3, rows, newrows, columns);

return newArray;

}

int\*\* getArrayFor2Task(int\*\* array, const int\* arraydiv3, const size\_t rows, const size\_t newrows, const size\_t columns)

{

int\*\* newArray = getArray(rows+newrows,columns);

size\_t temp2 = 0;

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

{

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

{

if

{

temp2++;

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

{

newArray[i+temp2][g] = array[i][g];

}

}

else

{

newArray[i+temp2][j] = array[i][j];

}

}

}

return newArray;

}

int getMax(int\*\* arrayReplaced, const size\_t rows, const size\_t columns, const size\_t i)

{

int max = arrayReplaced[i][0];

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

{

if (arrayReplaced[i][j] > max)

{

max = arrayReplaced[i][j];

}

}

return max;

}

int getNumberOfRowsdiv3(int\*\* array, const size\_t rows)

{

size\_t newrows = 0;

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

{

if ((array[i][0] % 3) == 0)

{

newrows++;

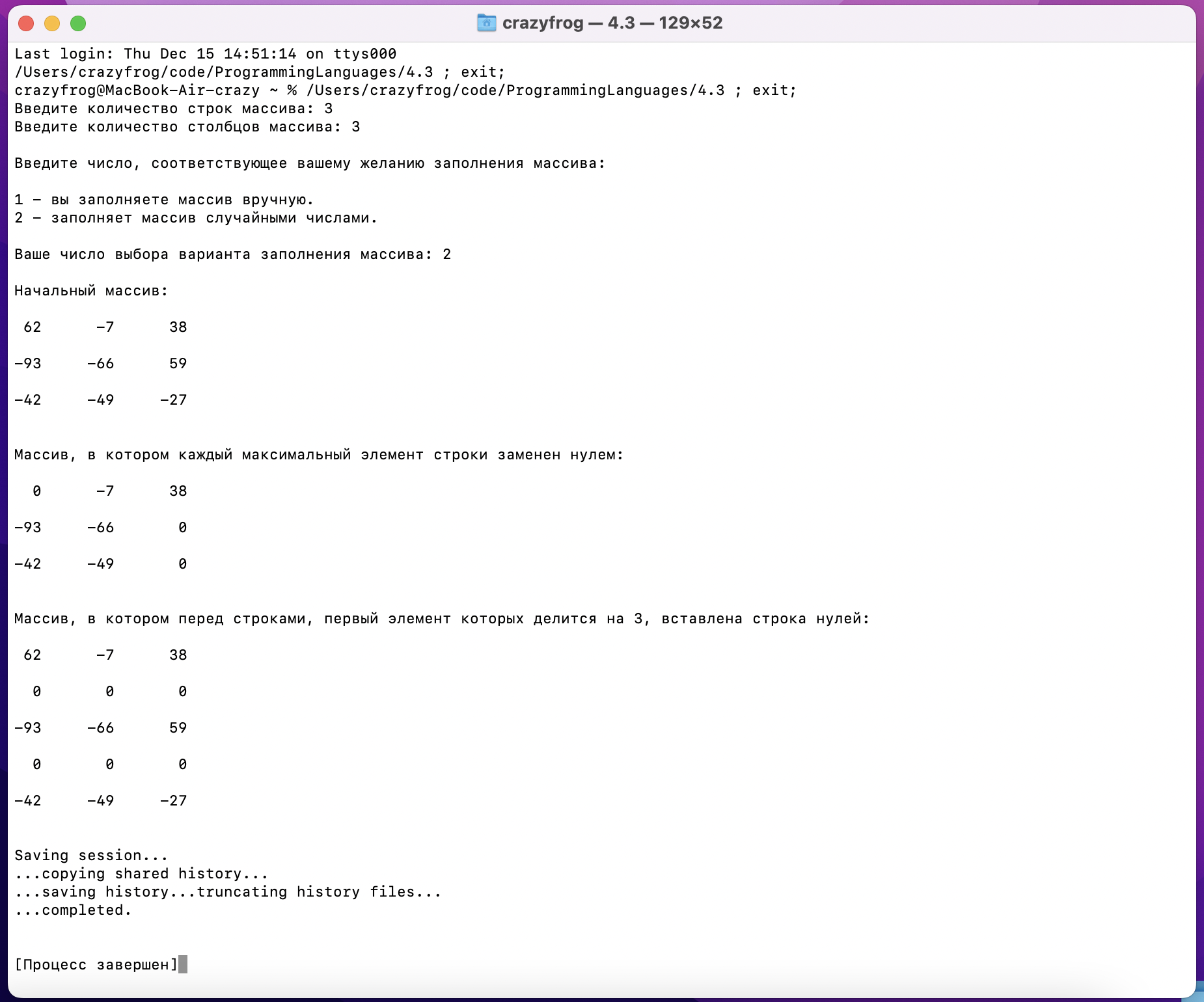
}

}

return newrows;

}

**4. Решение тестовых примеров**

****

**Рисунок 14 – Решение тестового примера**

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 15 – Решение тестового примера с отрицательным размером массива**

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 16 – Заполнение массива максимальным и кратным 3 элементом**

**Изображение выглядит как стол

Автоматически созданное описание**

**Рисунок 17 – Ручное заполнение массива**

**5. Approve задания на сайте GitHub.com**

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 18 – Скриншот одобренного задания 4.3**