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 – 111

Богомолов Никита Игоревич

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2022

Оглавление

[Задание 4-3 3](#_Toc117970572)

[Формулировка задачи 3](#_Toc117970573)

[Блок-схема алгоритма 3](#_Toc117970574)

[Решение задачи на языке программирования C++ 5](#_Toc117970575)

[Решение тестовых примеров 14](#_Toc117970576)

[Зачет задания в GitHub 19](#_Toc117970577)

# Задание 4-3

## Формулировка задачи

|  |  |
| --- | --- |
| **№ варианта** | **Задача** |
| 4 | 1. Заменить минимальный по модулю элемент каждого столбца на противоположный. 2. Удалить все строки, содержащие максимальные элементы. |

## Блок-схема алгоритма

Блок-схемы алгоритмов функций представлены на блок-схемах:

[Блок-схема 1 - Функция main 4](#_Toc118383868)

[Блок-схема 2 - Функция getSize 5](#_Toc118383869)

[Блок-схема 3 - Функция creatingArray 5](#_Toc118383870)

[Блок-схема 4 - Функция manualFillingArray 6](#_Toc118383871)

[Блок-схема 5 - Функция autoFillingArray 7](#_Toc118383872)

[Блок-схема 6 - Функция actOne 9](#_Toc118383873)

[Блок-схема 7 - Функция findAbsMaxElements 9](#_Toc118383874)

[Блок-схема 8 - Функция findAbsMinElements 10](#_Toc118383875)

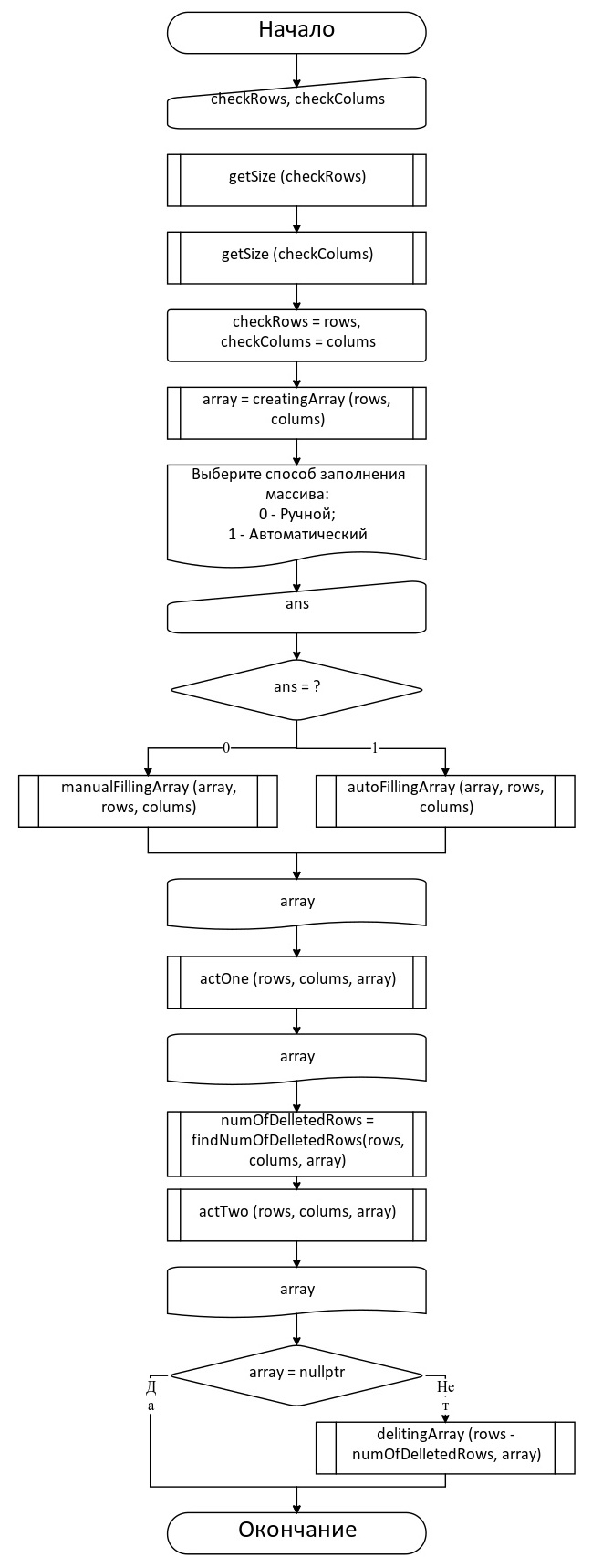
[Блок-схема 9 - Функция delitingArray 11](#_Toc118383876)

[Блок-схема 10 - Функция actTwo 13](#_Toc118383877)

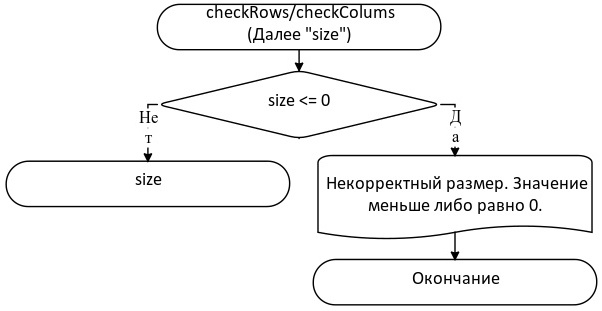
[Блок-схема 11 - Функция findRowsToDell 13](#_Toc118383878)

[Блок-схема 12 - Функция findMaxElements 14](#_Toc118383879)

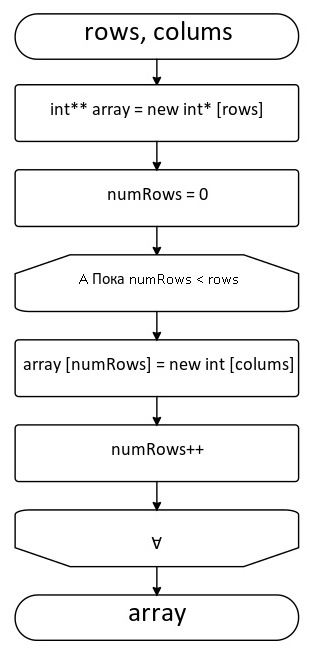
[Блок-схема 13 - Функция findNumOfDelletedRows 15](#_Toc118383880)



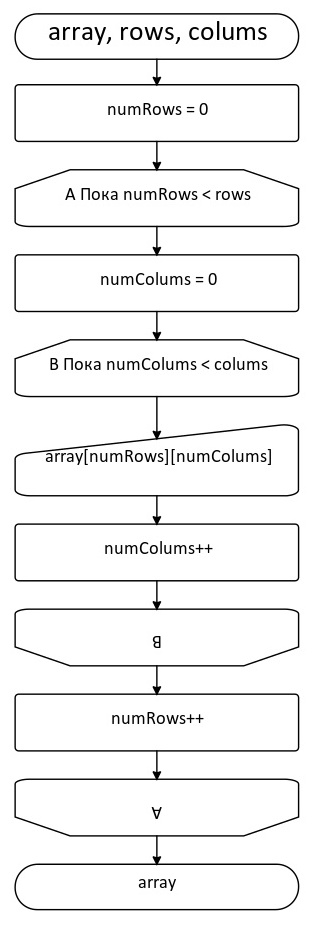
Блок-схема 1 - Функция main



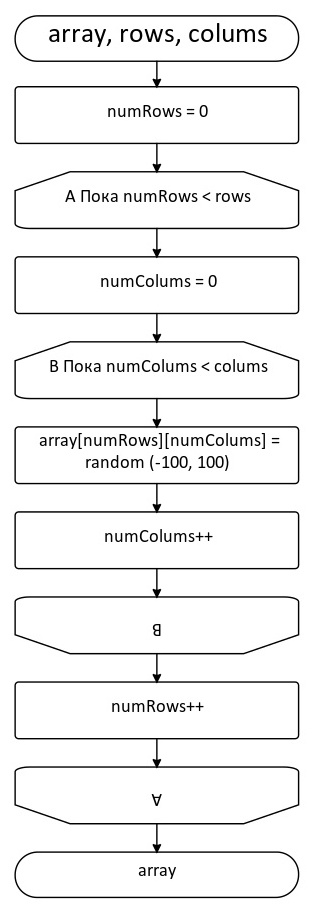
Блок-схема 2 - Функция getSize



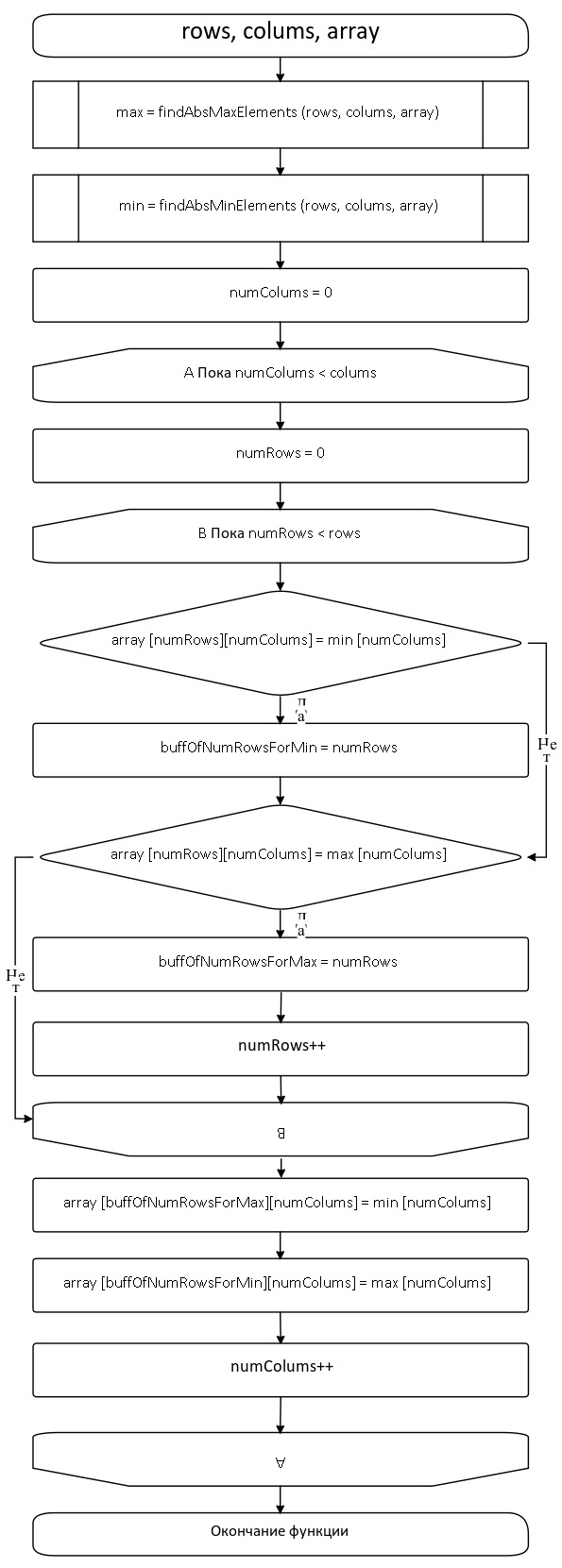
Блок-схема 3 - Функция creatingArray



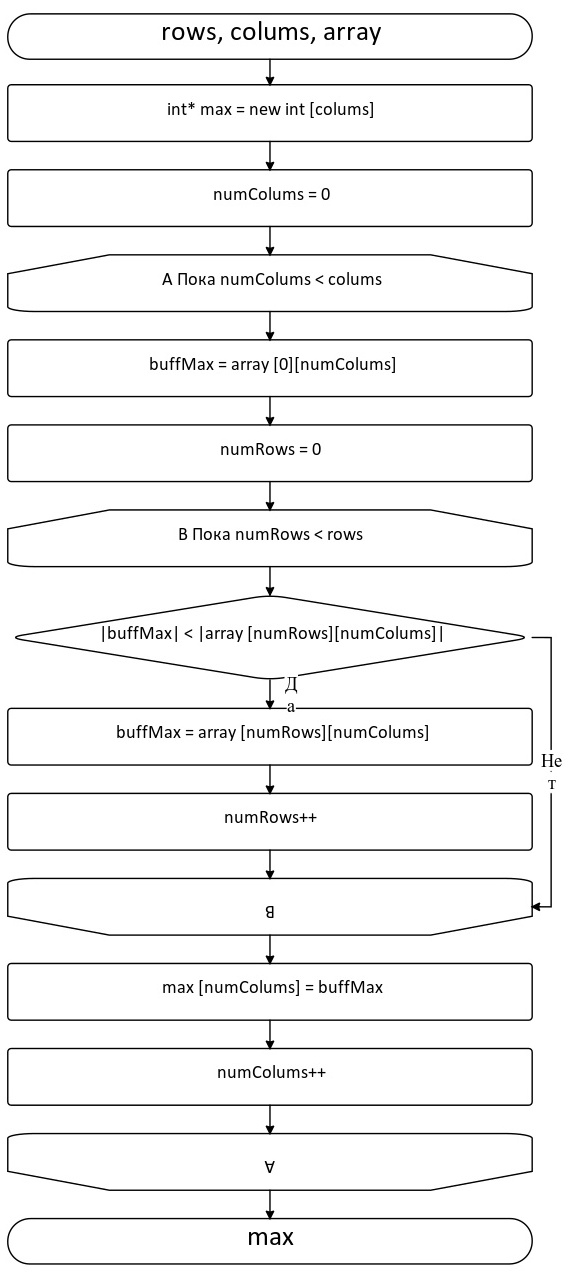
Блок-схема 4 - Функция manualFillingArray



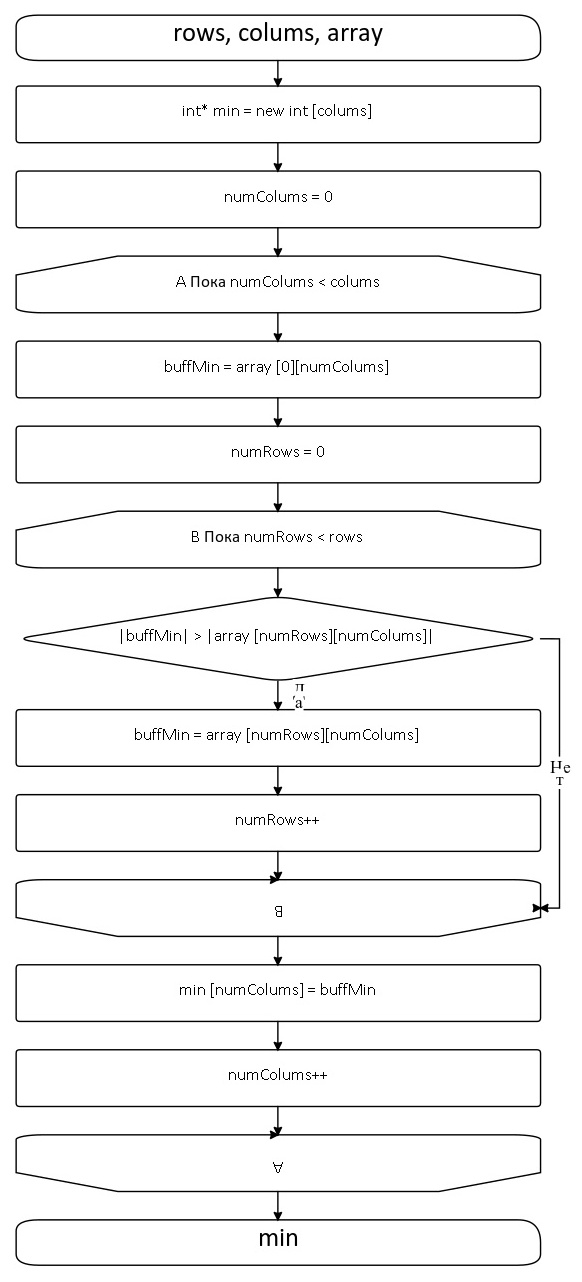
Блок-схема 5 - Функция autoFillingArray



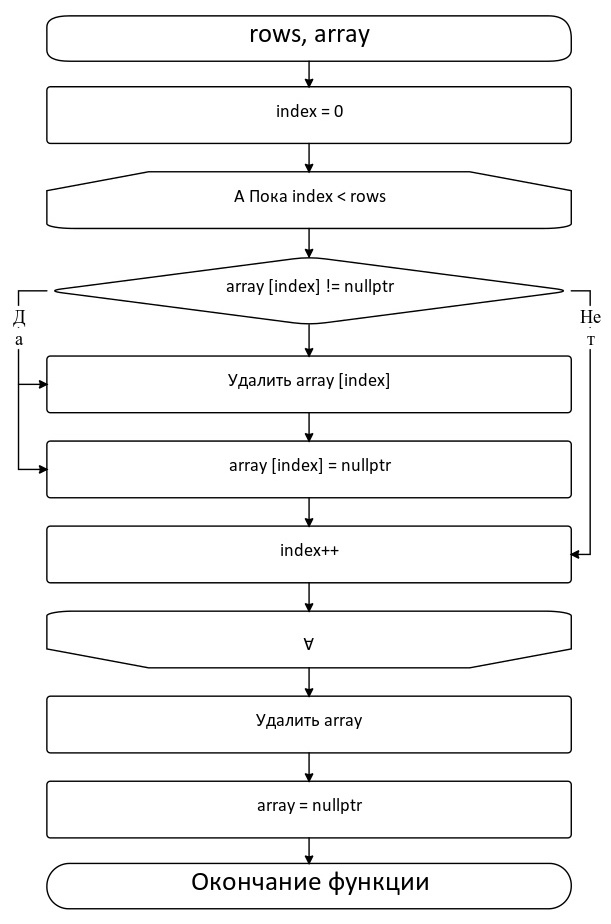
Блок-схема 6 - Функция actOne



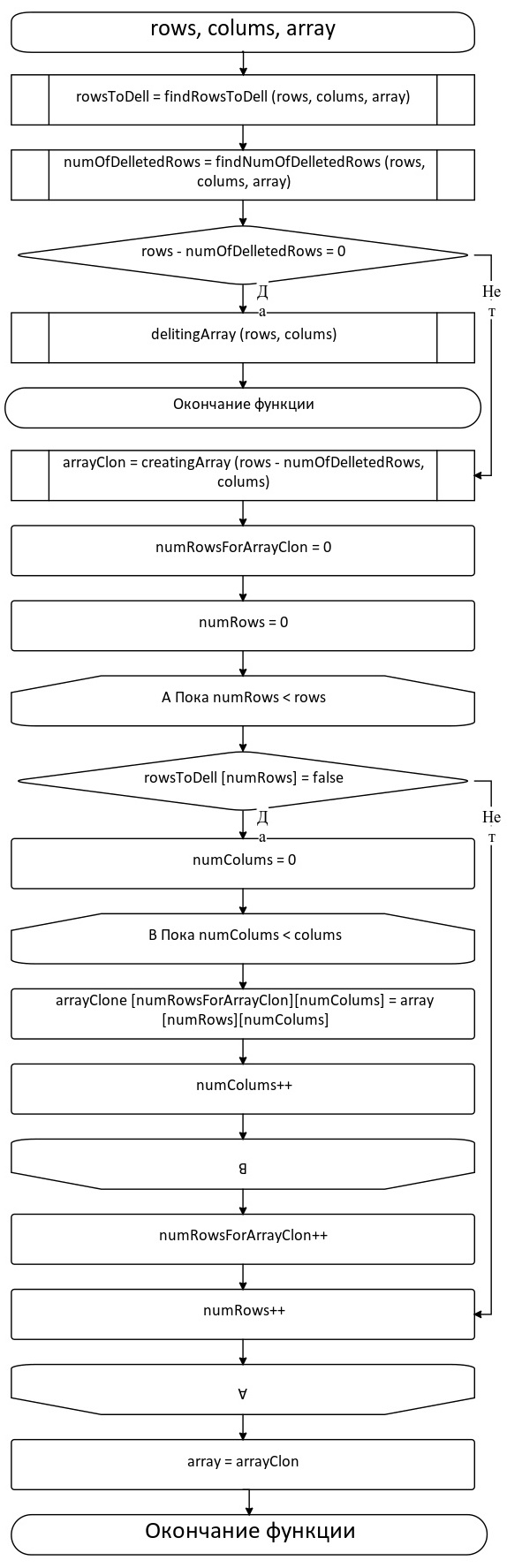
Блок-схема 7 - Функция findAbsMaxElements



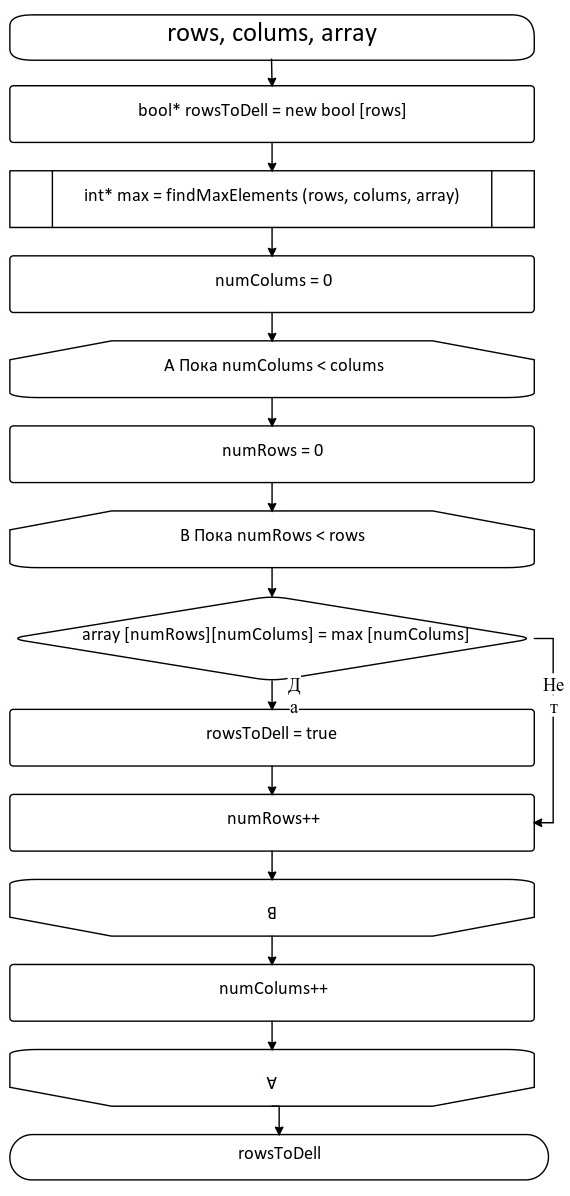
Блок-схема 8 - Функция findAbsMinElements



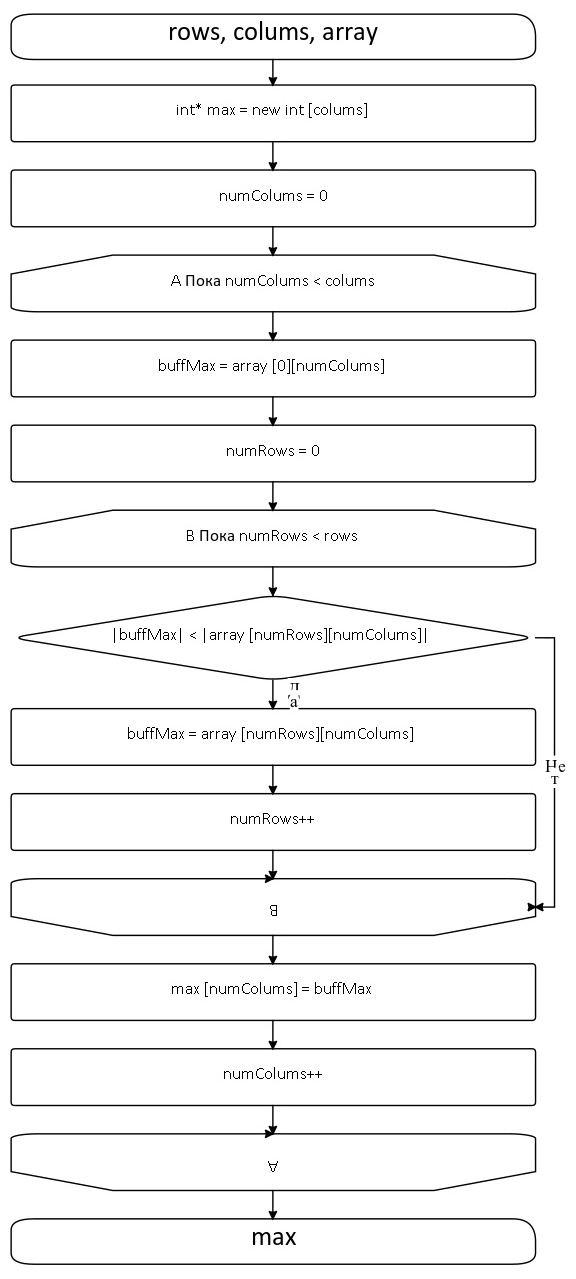
Блок-схема 9 - Функция delitingArray



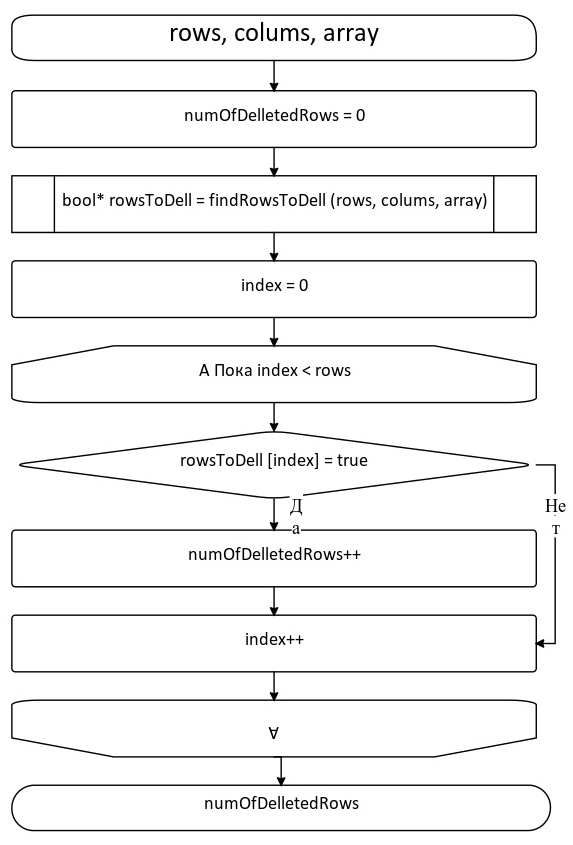
Блок-схема 10 - Функция actTwo



Блок-схема 11 - Функция findRowsToDell



Блок-схема 12 - Функция findMaxElements



Блок-схема 13 - Функция findNumOfDelletedRows

## Решение задачи на языке программирования C++

#include <iostream>

#include <cmath>

#include <sstream>

#include <random>

/\*\*

\* \brief Возможные варианты ответа пользователя

\* \yes Если пользователь ввёл "1"

\* \no Если пользователь ввёл "2"

\*/

enum class answer

{

yes,

no

};

/\*\*

\* \brief Проверка корректности ввода размерности массива

\* \param size Размер строки/столбца для проверки на корректность

\* \return false Если ввод некорректен

\*/

size\_t getSize(const int size);

/\*\*

\* \brief Создание массива array

\* \param rows Количество строк

\* \param colums Количество столбцов

\*/

int\*\* creatingArray(const size\_t rows, const size\_t colums);

/\*\*

\* \brief Ручное заполнение массива

\* \param rows Количество строк

\* \param colums Количество столбцов

\*/

void manualFillingArray(int\*\* array, const size\_t rows, const size\_t colums);

/\*\*

\* \brief Автоматическое заполнение массива

\* \param rows Количество строк

\* \param colums Количество столбцов

\*/

void autoFillingArray(int\*\* array, const size\_t rows, const size\_t colums);

/\*\*

\* \brief Функция преобразующая array в string для удобства вывода

\* \param rows Количество строк

\* \param colums Количество столбцов

\* \param array Массив

\* \return buffer.str() Возвращение string с массивом

\* \return deletedArray Возвращает сообщение о том, что масств удалён

\*/

std::string toString(const size\_t rows, const size\_t colums, int\*\* array);

/\*\*

\* \brief Функция выполняет задание 1 (поменять местами минимальный и максимальный по модулю элементы столбцов)

\* \param rows Количество строк

\* \param colums Количество столбцов

\* \param array Массив

\*/

void actOne(const size\_t rows, const size\_t colums, int\*\* array); // !

/\*\*

\* \brief Функция ищет максимальный по модулю элемент каждого столбца

\* \param rows Количество строк

\* \param colums Количество столбцов

\* \param array Массив

\* \return max Возвращает список максимальных элементов

\*/

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

/\*\*

\* \brief Функция ищет минимальный по модулю элемент каждого столбца

\* \param rows Количество строк

\* \param colums Количество столбцов

\* \param array Массив

\* \return min Возвращает список минимальных элементов

\*/

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

/\*\*

\* \brief Функция удаляет массив

\* \param rows Количество строк

\* \param arr Удаляемый массив

\*/

void deletingArray(const size\_t rows, int\*\*& arr);

/\*\*

\* \brief Функция выполняет задание 2 (удалить строки содержащие максимальные элементы столбцов)

\* \param rows Количество строк

\* \param colums Количество столбцов

\* \param array Массив

\*/

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

/\*\*

\* \brief Функция ищет строки для удаления

\* \param rows Количество строк

\* \param colums Количество столбцов

\* \param array Массив

\* \return rowsToDell Возвращает список строк для удаления

\*/

bool\* findRowsToDell(const size\_t rows, const size\_t colums, int\*\* array);

/\*\*

\* \brief Функция ищет максимальный элемент каждого столбца

\* \param rows Количество строк

\* \param colums Количество столбцов

\* \param array Массив

\* \return max Возвращает список максимальных элементов

\*/

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

/\*\*

\* \brief Функция ищет количество удаляемых строк

\* \param rows Количество строк

\* \param colums Количество столбцов

\* \param array Массив

\* \return max Возвращает список максимальных элементов

\*/

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

/\*\*

\* \brief Точка входа в программу

\* \return 0 Возвращает ноль в случае успеха

\* \return 1 Возвращает один если некорректный ввод размерности массива

\*/

int main()

{

std::cout << "Введите размерность массива:\n Количество строк: ";

int checkRows;

std::cin >> checkRows;

try

{

getSize(checkRows);

}

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

{

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

return 1;

}

const size\_t rows = getSize(checkRows);

std::cout << " Количество столбцов: ";

int checkColums;

std::cin >> checkColums;

try

{

getSize(checkColums);

}

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

{

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

return 1;

}

const size\_t colums = getSize(checkColums);

std::cout << "Выберите способ заполнения массива:\n "

<< static\_cast<int>(answer::yes) << " - Ручное\n "

<< static\_cast<int>(answer::no) << " - Автоматическое\n ";

int ans;

std::cin >> ans;

int\*\* array = creatingArray(rows, colums);

try

{

const auto chosen = static\_cast<answer>(ans);

switch (chosen)

{

case answer::no:

{

autoFillingArray(array, rows, colums);

break;

}

case answer::yes:

{

manualFillingArray(array, rows, colums);

break;

}

default :

std::cout << "Вы ввели неправильный ответ!\n";

return 1;

}

}

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

{

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

return 1;

}

std::cout << "Массив:\n" << toString(rows, colums, array) << "\n";

actOne(rows, colums, array);

std::cout << "Массив по итогу первого задания:\n" << toString(rows, colums, array) << std::endl;

size\_t numOfDelletedRows = findNumOfDelletedRows(rows, colums, array);

actTwo(rows, colums, array);

std::cout << "Массив по итогу второго задания:\n" << toString(rows - numOfDelletedRows, colums, array) << std::endl;

if(array != nullptr)

{

deletingArray(rows - numOfDelletedRows, array);

}

return 0;

}

size\_t getSize(const int size)

{

if (size <= 0)

{

throw std::out\_of\_range("Некорректный размер. Значение меньше либо равно 0.");

}

return size;

}

int\*\* creatingArray(const size\_t rows, const size\_t colums)

{

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

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

array [numRows] = new int [colums];

return array;

}

void manualFillingArray(int\*\* array, const size\_t rows, const size\_t colums)

{

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

{

for(size\_t numColums = 0; numColums < colums; numColums++)

{

std::cout << "Элемент " << numRows + 1 << "." << numColums + 1 << ": ";

std::cin >> array[numRows][numColums];

}

}

}

void autoFillingArray(int\*\* array, const size\_t rows, const size\_t colums)

{

std::random\_device rd;

std::mt19937\_64 mt\_rand(rd());

std::uniform\_int\_distribution<int> dist(-100, 100);

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

{

for(size\_t numColums = 0; numColums < colums; numColums++)

{

array[numRows][numColums] = dist(mt\_rand);

}

}

}

std::string toString(const size\_t rows, const size\_t colums, int\*\* array)

{

std::stringstream buffer;

std::string deletedArray = "Массив удалён. \n";

if (array != nullptr)

{

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

{

for (size\_t numColums = 0; numColums < colums; numColums++)

{

buffer << array [numRows][numColums] << "; ";

}

buffer << "\n";

}

return buffer.str();

}

else

return deletedArray;

}

void actOne(const size\_t rows, const size\_t colums, int\*\* array)

{

int\* max = findAbsMaxElements(rows, colums, array);

int\* min = findAbsMinElements(rows, colums, array);

size\_t buffOfNumRowsForMin;

size\_t buffOfNumRowsForMax;

for(size\_t numColums = 0; numColums < colums; numColums++)

{

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

{

if(array[numRows][numColums] == min[numColums])

{

buffOfNumRowsForMin = numRows;

}

if(array[numRows][numColums] == max[numColums])

{

buffOfNumRowsForMax = numRows;

}

}

array[buffOfNumRowsForMax][numColums] = min[numColums];

array[buffOfNumRowsForMin][numColums] = max[numColums];

}

}

int\* findAbsMaxElements(const size\_t rows, const size\_t colums, int\*\* array)

{

int\* max = new int [colums];

for(size\_t numColums = 0; numColums < colums; numColums++)

{

int buffMax = array[0][numColums];

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

{

if(abs(buffMax) < abs(array[numRows][numColums]))

{

buffMax = array[numRows][numColums];

}

}

max[numColums] = buffMax;

}

return max;

}

int\* findAbsMinElements(const size\_t rows, const size\_t colums, int\*\* array)

{

int\* min = new int [colums];

for(size\_t numColums = 0; numColums < colums; numColums++)

{

int buffMin = array[0][numColums];

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

{

if(abs(buffMin) > abs(array[numRows][numColums]))

{

buffMin = array[numRows][numColums];

}

}

min[numColums] = buffMin;

}

return min;

}

void deletingArray(const size\_t rows, int\*\*& arr)

{

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

{

if (arr[index] != nullptr)

{

delete[] arr[index];

arr[index] = nullptr;

}

}

delete[] arr;

arr = nullptr;

}

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

{

bool\* rowsToDell = findRowsToDell(rows, colums, array);

size\_t numOfDelletedRows = findNumOfDelletedRows(rows, colums, array);

if(rows - numOfDelletedRows == 0)

{

return deletingArray(rows, array);

}

int\*\* arrayClon = creatingArray(rows - numOfDelletedRows, colums);

size\_t numRowsForArrayClon = 0;

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

{

if(!rowsToDell[numRows])

{

for(size\_t numColums = 0; numColums < colums; numColums++)

{

arrayClon[numRowsForArrayClon][numColums] = array[numRows][numColums];

}

numRowsForArrayClon++;

}

}

array = arrayClon;

}

bool\* findRowsToDell(const size\_t rows, const size\_t colums, int\*\* array)

{

bool\* rowsToDell = new bool [rows];

int\* max = findMaxElements(rows, colums, array);

for(size\_t numColums = 0; numColums < colums; numColums++)

{

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

{

if(array[numRows][numColums] == max[numColums])

{

rowsToDell[numRows] = true;

}

}

}

return rowsToDell;

}

int\* findMaxElements(const size\_t rows, const size\_t colums, int\*\* array)

{

int\* max = new int [colums];

for(size\_t numColums = 0; numColums < colums; numColums++)

{

int buffMax = array[0][numColums];

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

{

if(buffMax < array[numRows][numColums])

{

buffMax = array[numRows][numColums];

}

}

max[numColums] = buffMax;

}

return max;

}

size\_t findNumOfDelletedRows(const size\_t rows, const size\_t colums, int\*\* array)

{

size\_t numOfDelletedRows = 0;

bool\* rowsToDell = findRowsToDell(rows, colums, array);

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

{

if(rowsToDell[index])

{

numOfDelletedRows++;

}

}

return numOfDelletedRows;

}

## Решение тестовых примеров

[Тестовый пример 1 - Ручное заполнение массива 3\*3 14](#_Toc117969052)

[Тестовый пример 2 - Автоматическое заполнение 3\*3 15](#_Toc117969053)

[Тестовый пример 3 - Автоматическое заполнение массива 7\*4 16](#_Toc117969054)

[Тестовый пример 4 - Пример удаления массива по итогу второго задания 17](#_Toc117969055)

[Тестовый пример 5 - Пример ввода отрицательного количества строк 17](#_Toc117969056)

[Тестовый пример 6 - Пример ввода нулевого количества строк 18](#_Toc117969057)

[Тестовый пример 7 - Пример ввода отрицательного количества столбцов 18](#_Toc117969058)

[Тестовый пример 8 - Пример ввода нулевого количества столбцов 18](#_Toc117969059)

[Тестовый пример 9 - Пример ввода некорректного выбора 18](#_Toc117969060)

Изображение выглядит как текст

Автоматически созданное описание

Тестовый пример 1 - Ручное заполнение массива 3\*3

Изображение выглядит как текст

Автоматически созданное описание

Тестовый пример 2 - Автоматическое заполнение 3\*3

Изображение выглядит как текст

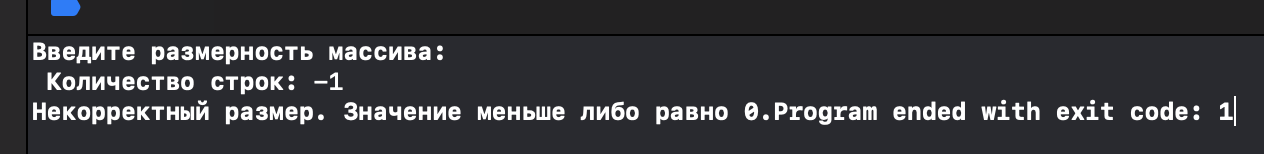
Автоматически созданное описание

Тестовый пример 3 - Автоматическое заполнение массива 7\*4

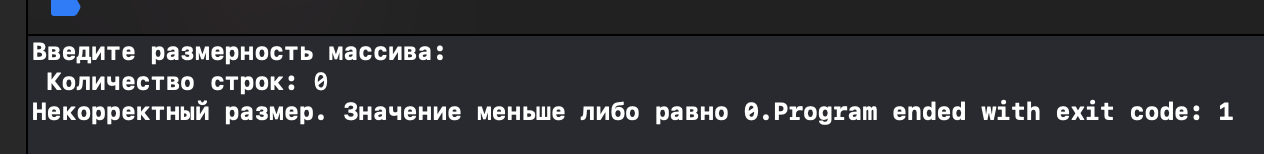
Изображение выглядит как текст

Автоматически созданное описание

Тестовый пример 4 - Пример удаления массива по итогу второго задания



Тестовый пример 5 - Пример ввода отрицательного количества строк



Тестовый пример 6 - Пример ввода нулевого количества строк

Изображение выглядит как текст

Автоматически созданное описание

Тестовый пример 7 - Пример ввода отрицательного количества столбцов

Изображение выглядит как текст

Автоматически созданное описание

Тестовый пример 8 - Пример ввода нулевого количества столбцов

Изображение выглядит как текст

Автоматически созданное описание

Тестовый пример 9 - Пример ввода некорректного выбора

## Зачет задания в GitHub

Изображение выглядит как текст

Автоматически созданное описание