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 – 111

Байло А.В.

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2022

# Задание 4-3

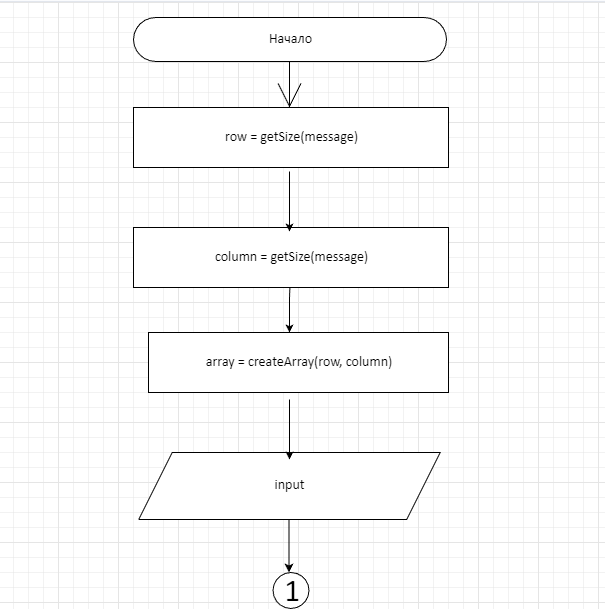
1. Формулировка задачи

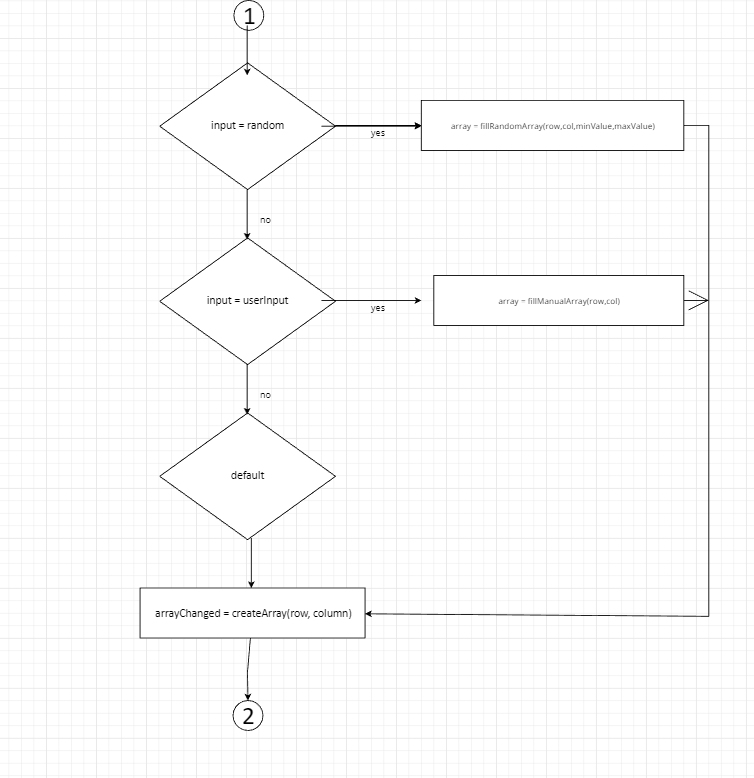
Изображение выглядит как текст

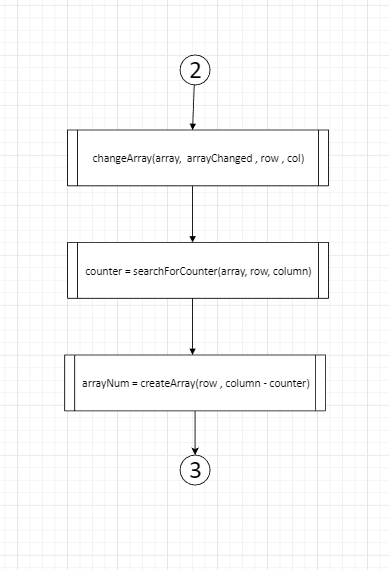
Автоматически созданное описание

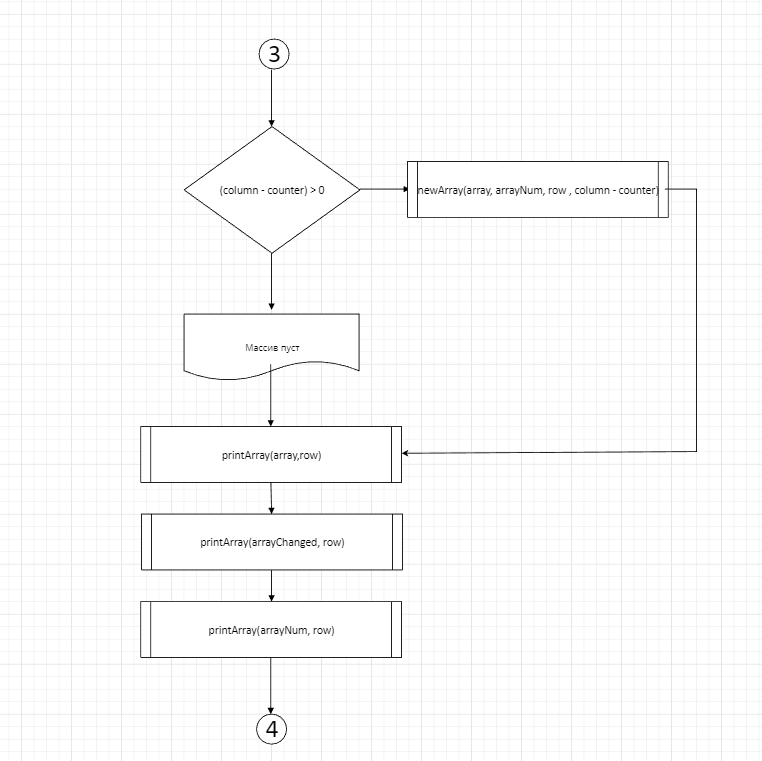
1. Блок-схемы алгоритмов функций представлены на рисунках (Рисунок 1 - 12 )

Рисунок 1 – Блок-схема функции main()









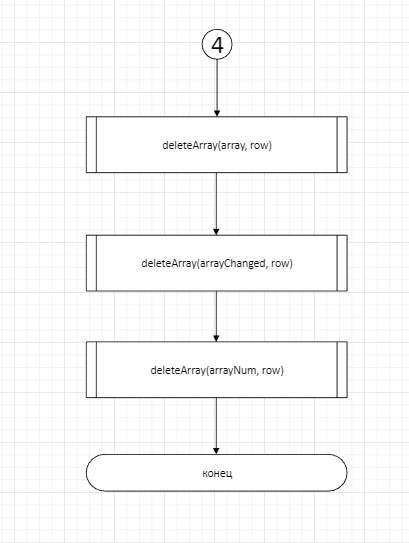


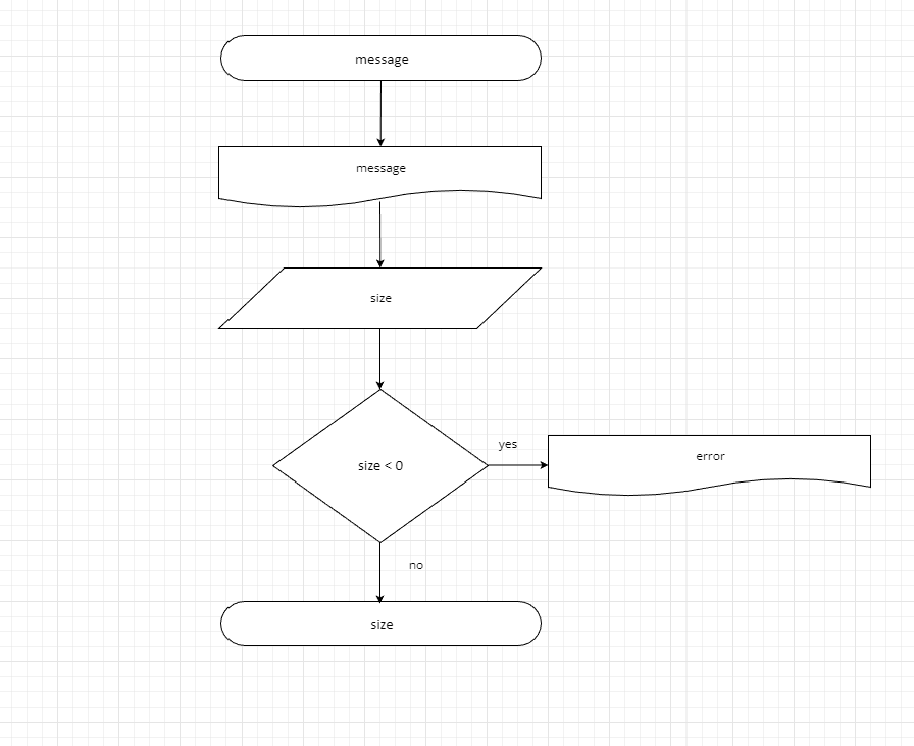
Рисунок 2 – Блок-схема функции getSizeйц

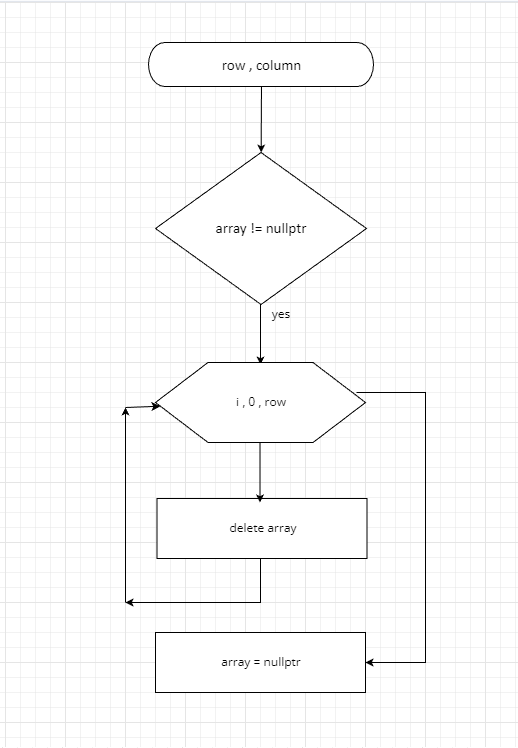
Рисунок 3 – Блок-схема функции deleteArray()

Рисунок 4 – Блок-схема функции createArray

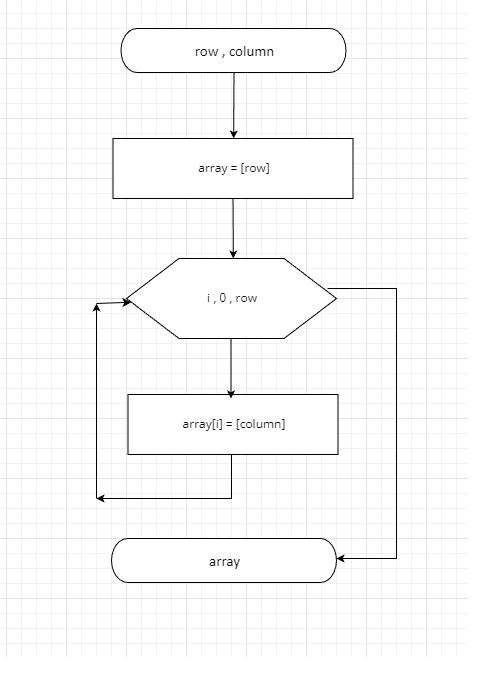


Рисунок 5 – Блок-схема функции printArray

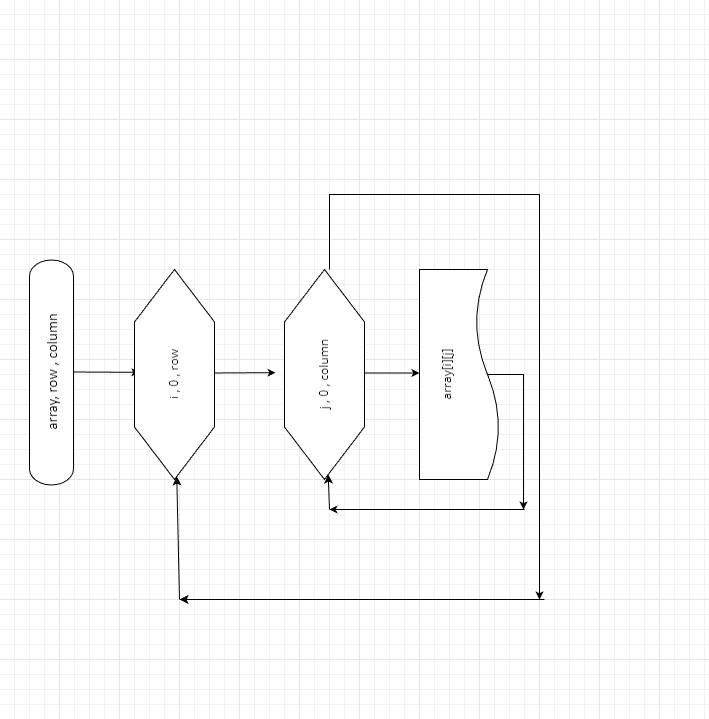


Рисунок 6 – Блок-схема функции searchForCouter

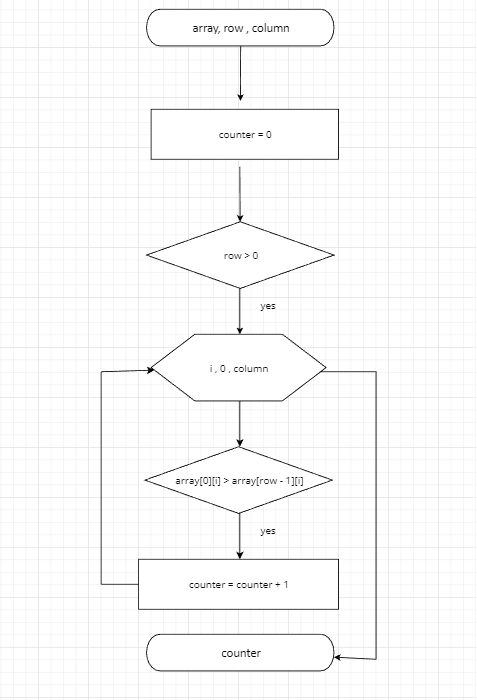


Рисунок 7 – Блок-схема функции fillRandomArray

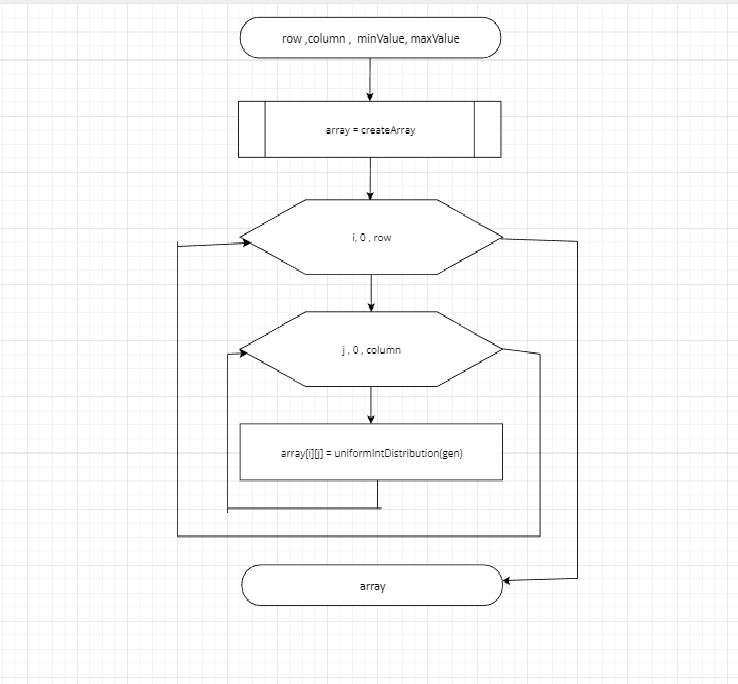


Рисунок 8 – Блок-схема функции fillManualArray

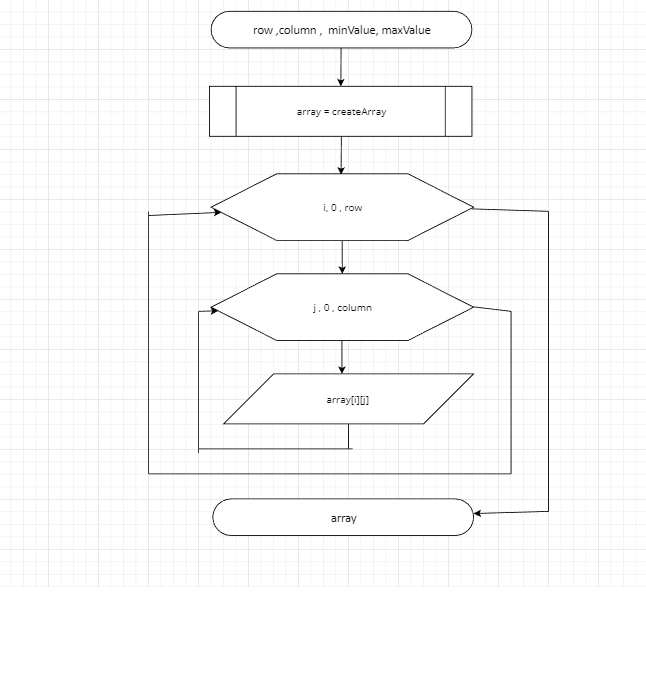


Рисунок 9 – Блок-схема функции columnNumbers

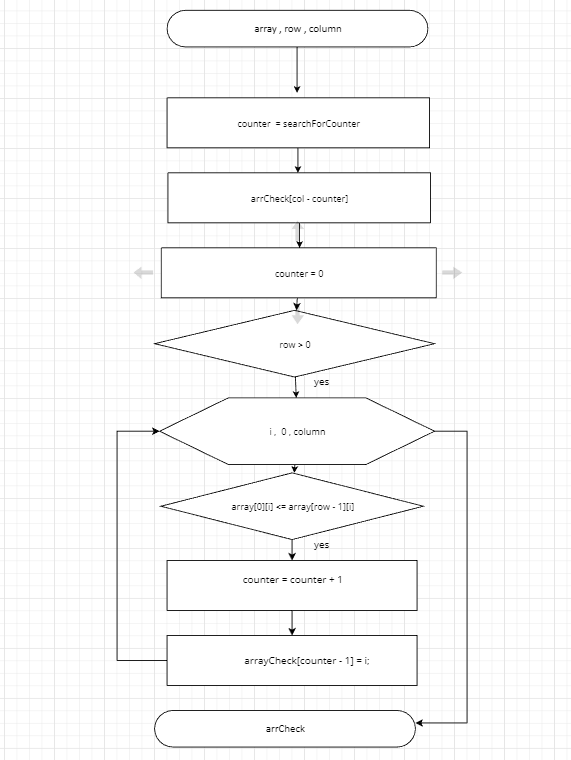


Рисунок 10 – Блок-схема функции searchForMini

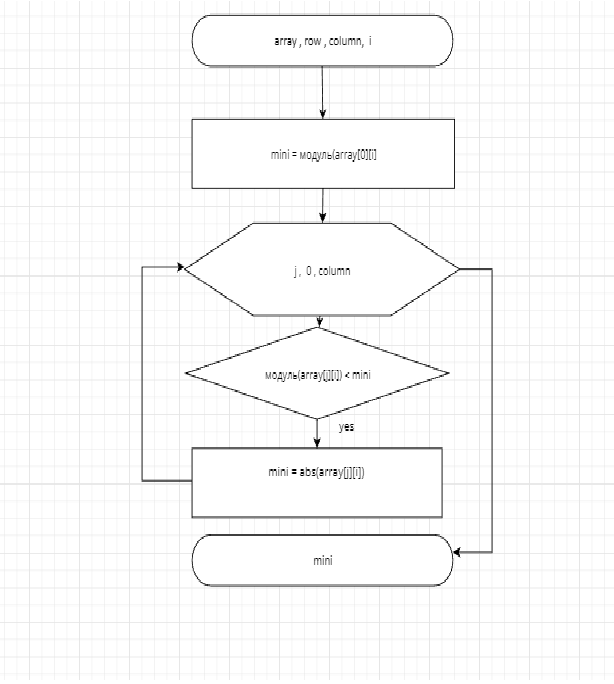


Рисунок 11 – Блок-схема функции changeArray

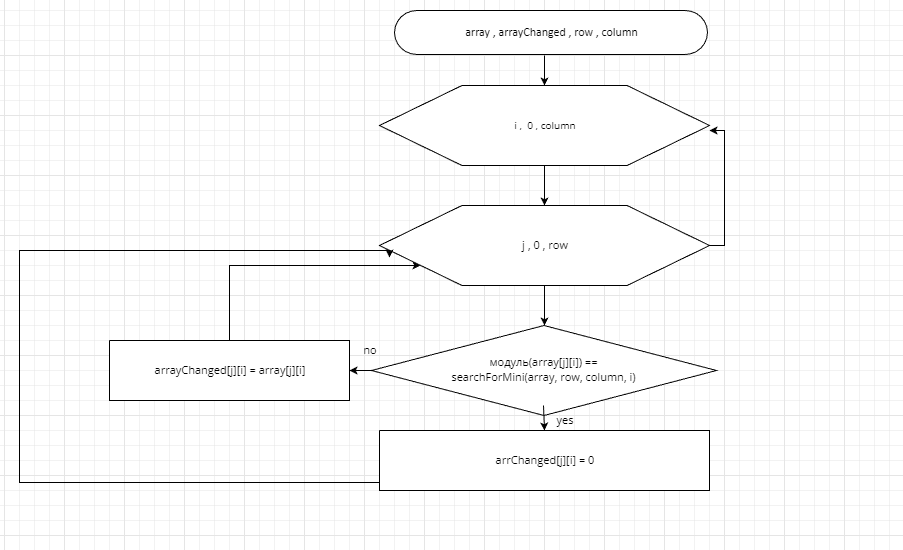
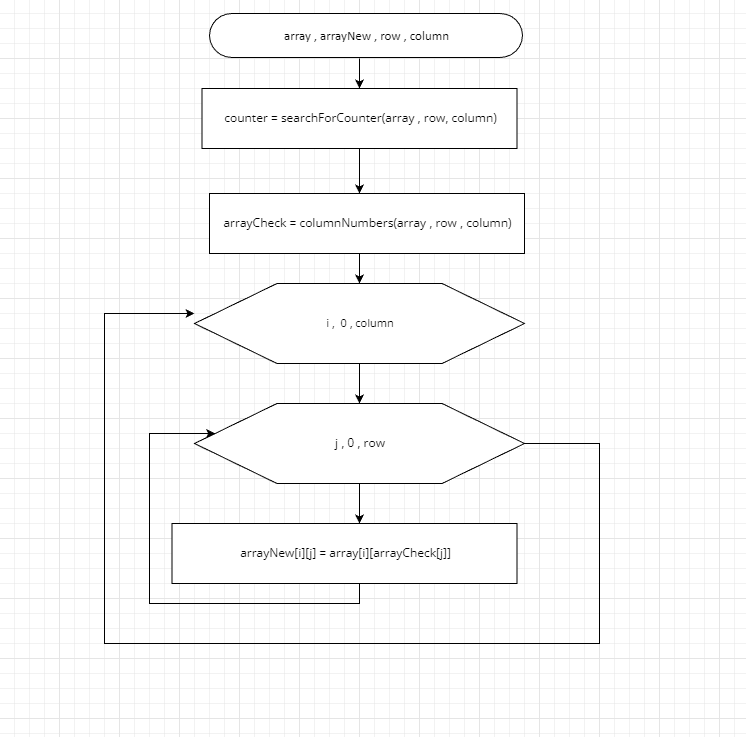


Рисунок 12 – Блок-схема функции newArray



Решение задачи на языке программирования C++

#include <iostream>

#include <ctime>

#include <random>

using namespace std;

/\*

\*\brief Функция считывающая размеры массива

\*\param message Побуждающее сообщение (подсказка для пользователя)

\*\return Возвращает размеры массива

\*/

size\_t getSize(const string& message);

/\*

\*\brief Выбор типа заполнения массива

\*\param random - Случайное заполнение

\*\param userInput - Пользовательский ввод

\*/

enum class fillarray

{

random,

userInput

};

/\*

\*\brief Фукция удаляющая массив

\*\param array - указатель на массив

\*\param row - кол-во строк в массиве

\*/

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

/\*

\*\brief Фукция создающая двухмерный массив

\*\param row - кол-во строк в массиве

\*\return Возвращает указатель на массив

\*/

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

/\*

\*\brief Фукция выводящая на экран массив

\*\param array - указатель на массив

\*\param row - кол-во строк в массиве

\*\param column - кол-во столбцов в массиве

\*/

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

/\*

\*\brief Фукция которая ищет кол-во столбцов , в которых первый элемент больше последнего

\*\param array - указатель на массив

\*\param row - кол-во строк в массиве

\*\param column - кол-во столбцов в массиве

\*\return Возвращает кол-во столбцов , в которых первый элемент больше последнего

\*/

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

/\*

\*\brief Фукция которая заполняет массив случайными элементами

\*\param row - кол-во строк в массиве

\*\param column - кол-во столбцов в массиве

\*\param minValue нижняя граница рандома

\*\param maxValue верхняя граница рандома

\*\return Возвращает указатель на заполненный массив

\*/

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

/\*

\*\brief Функция создающая массив с номерами столбцов, в которых первый элемент меньше последнего

\*\param \*\*array указатель на массив , столбцы которого необходимо анализировать

\*\param row - кол-во строк в массиве

\*\param column - кол-во столбцов в массиве

\*\return Возвращает указатель на массив с номерами столбцов, в которых первый элемент меньше последнего

\*/

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

/\*

\*\brief Фукция заполняющая массив пользовательским вводом

\*\param row - кол-во строк в массиве

\*\param column - кол-во столбцов в массиве

\*\return Возвращает указатель на заполненный массив

\*/

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

/\*

\*\brief Фукция которая ищет минимальный по модулю элемент столбца

\*\param row - кол-во строк в массиве

\*\param column - кол-во столбцов в массиве

\*\param i - номер строки, в котоой следует искать минимум

\*\return Возвращает минимальный по модулю элемент столбца

\*/

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

/\*

\*\brief Фукция ,заменяющая минимальный по модулю элемент столбца массива нулем

\*\param array - указатель на массив

\*\param arrayChanged указатель на массив, минимальные элементы которого будут заменены

\*\param row - кол-во строк в массиве

\*\param column - кол-во столбцов в массиве

\*/

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

/\*

\*\brief Фукция удаляющая в массиве столбцы, первый элемент которых больше последнего

\*\param array - указатель на массив

\*\param arrayNew указатель на массив с удаленными столбцами

\*\param row - кол-во строк в массиве

\*\param column - кол-во столбцов в массиве

\*/

void newArray(int\*\* array, int\*\* arrayNew, const size\_t row, const size\_t column);//doc

/\*

\*\brief Точка входа в программу

\*\return возвращает 0 в случае успеха

\*/

int main()

{

setlocale(LC\_ALL, "Rus");

const int minValue = -100;

const int maxValue = 100;

int row = getSize("Bведите кол-во строк\n");

int column = getSize ("Введите кол-во рядов массива\n");

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

cout << "Как следует заполнить массив?" << "\n" << static\_cast<int>(fillarray::random) << '\t' << "Случайно" << "\n" << static\_cast <int>(fillarray::userInput) << '\t' << "Пользовательский ввод" << endl;

int input = 0;

cin >> input;

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

switch (choice)

{

case(fillarray::random):

{

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

break;

}

case(fillarray::userInput):

{

array = fillManualArray(row, column);

break;

}

default:

{

cout << "Нет такой операции заполнения" << endl;

}

break;

}

//Пункт 1

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

changeArray(array, arrayChanged, row, column);

//Пункт 2

int counter = searchForCounter(array, row, column);

int\*\* arrayNum = createArray(row, column - counter);

if (column - counter > 0)

{

newArray(array, arrayNum, row, column);

}

else

{

cout << "Все столбца были удалены, новый массив пуст!" << endl;

}

//Вывод

cout << "Начальный массив" << endl;

printArray(array, row, column);

cout << "Массив с измененными минимальными элементами" << endl;

printArray(arrayChanged, row, column);

cout << "Массив с удаленными столбцами" << endl;

printArray(arrayNum, row, column-counter);

//Удаление

deleteArray(array, row);

deleteArray(arrayChanged, row);

deleteArray(arrayNum, row);

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;

}

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;

}

}

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;

}

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

{

int counter = 0;

if (row > 0)

{

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

{

if (array[0][i] > array[row - 1][i])

{

counter += 1;

}

}

return counter;

}

}

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;

}

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

{

int counter = searchForCounter(array, row, column);

int\* arrayCheck = new int[(column - counter)];

counter = 0;

if (row > 0)

{

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

{

if (array[0][i] <= array[row - 1][i])

{

counter += 1;

arrayCheck[counter - 1] = i;

}

}

}

return arrayCheck;

}

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

{

int mini = std::abs(array[0][i]);

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

{

if (std::abs(array[j][i]) < mini)

{

mini = std::abs(array[j][i]);

}

}

return mini;

}

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

{

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

{

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

{

if (std::abs(array[j][i]) == searchForMini(array, row, column, i))

{

arrayChanged[j][i] = 0;

}

else

{

arrayChanged[j][i] = array[j][i];

}

}

}

}

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

{

int counter = searchForCounter(array, row, column);

int\* arrayCheck = columnNumbers(array, row, column);

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

{

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

{

arrayNew[i][j] = array[i][arrayCheck[j]];

}

}

if (arrayCheck != nullptr)

{

delete[] arrayCheck;

arrayCheck = nullptr;

}

}

1. Решение тестовых примеров

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 13– Решение тестового примера

1. Зачет задания в GitHub



Рисунок 14 – Зачет задания