

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 ЛАБОРАТОРНОЙ РАБОТЕ №4.3  
По дисциплине «Языки программирования»  
**ВАРИАНТ 10**

Выполнил: ст. гр. ТКИ-141  
Фамилия Кручинин Игорь  
Проверил: к.т.н., доц. Васильева М. А.

Москва 2023

# 1 ФОРМУЛИРОВКА ЗАДАНИЯ

Создать консольное приложение, вычисляющее значения переменных по представленным в таблице формулам (Таблица 1). Расчёт примера осуществить по заданным константам. Вывести на экран значения исходных данных, а также результат вычислений. Дополнить свой отчёт блок-схемой алгоритма.

Таблица 1 – Исходные данные

| Вариант | Формулы   |
|---------|---|
| 10      | <ol style="list-style-type: none"><li>1. Заменить нулевой элемент каждого столбца максимальным по модулю элементом массива.</li><li>2. Вставить после каждого столбца, содержащего максимальный по модулю элемент, строку из нулей.</li></ol> |

## 2 БЛОК-СХЕМА АЛГОРИТМА

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций расчета значений **a** и **b** представлены ниже (Ошибка: источник перекрёстной ссылки не найден).

Рисунок 1 Блок-схема основного алгоритма

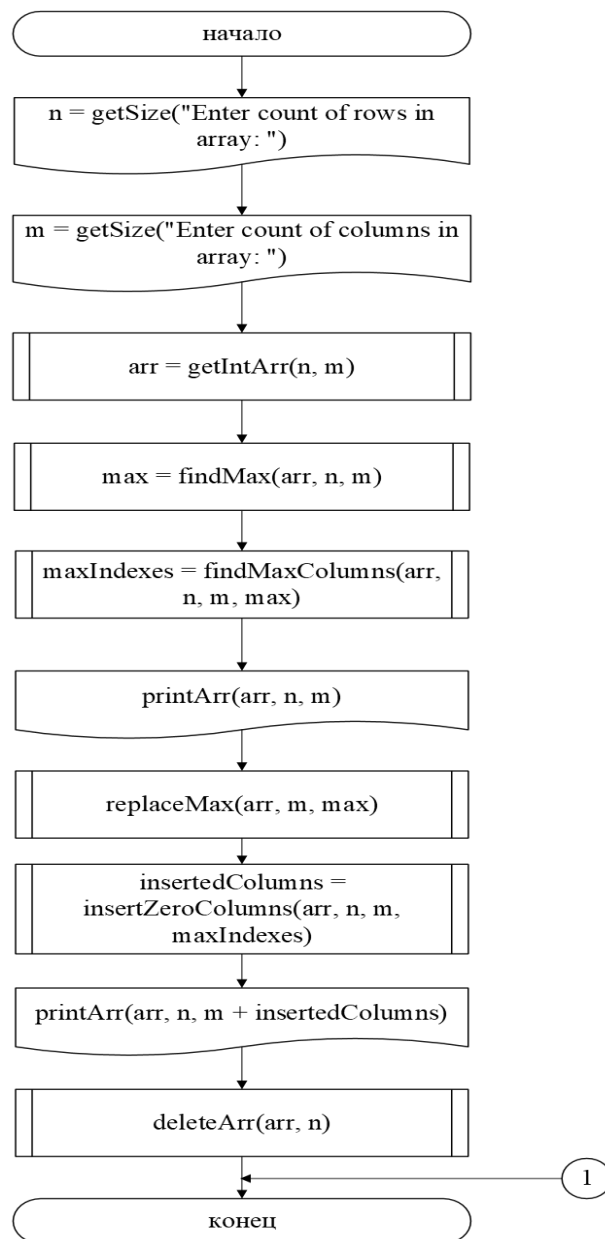
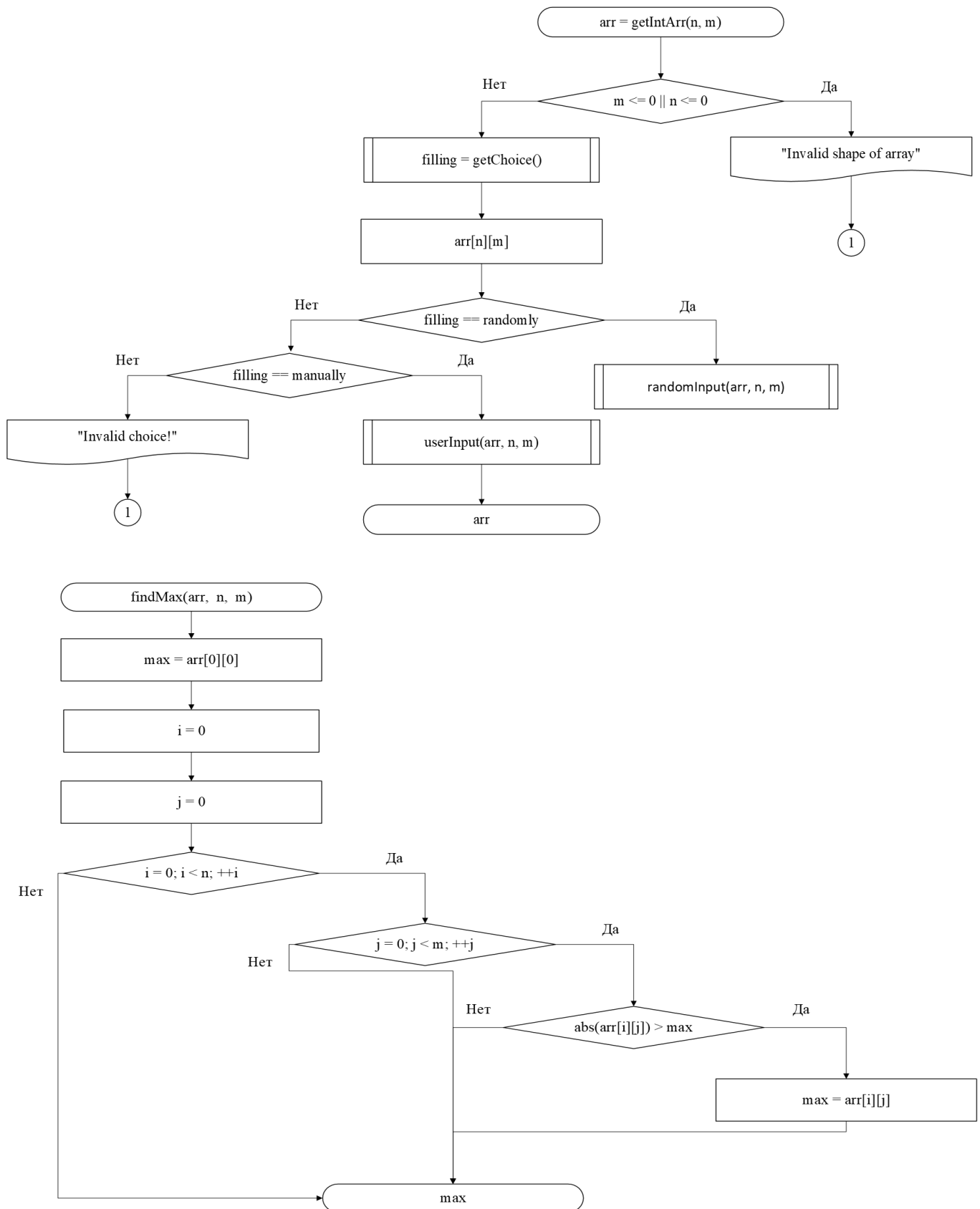
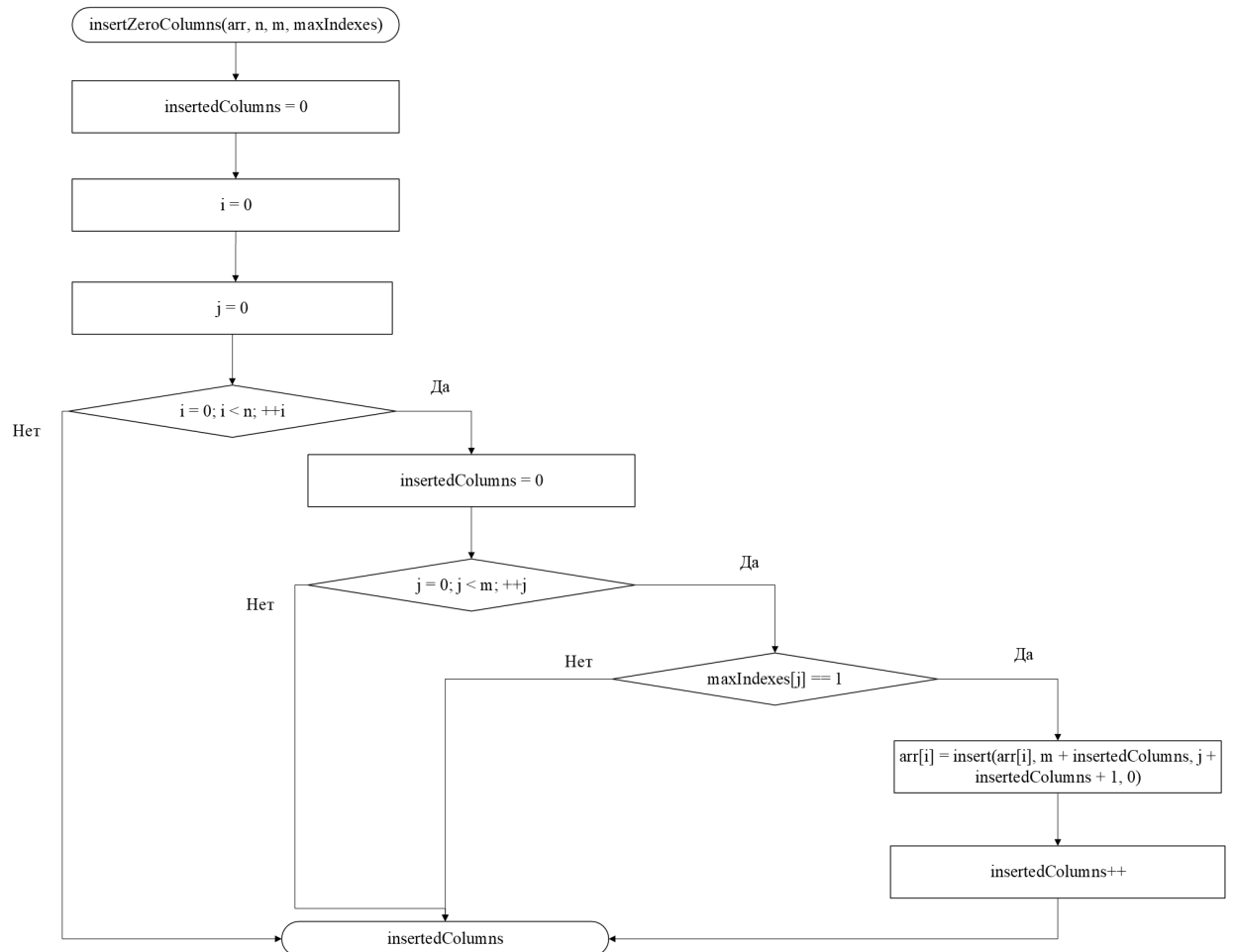
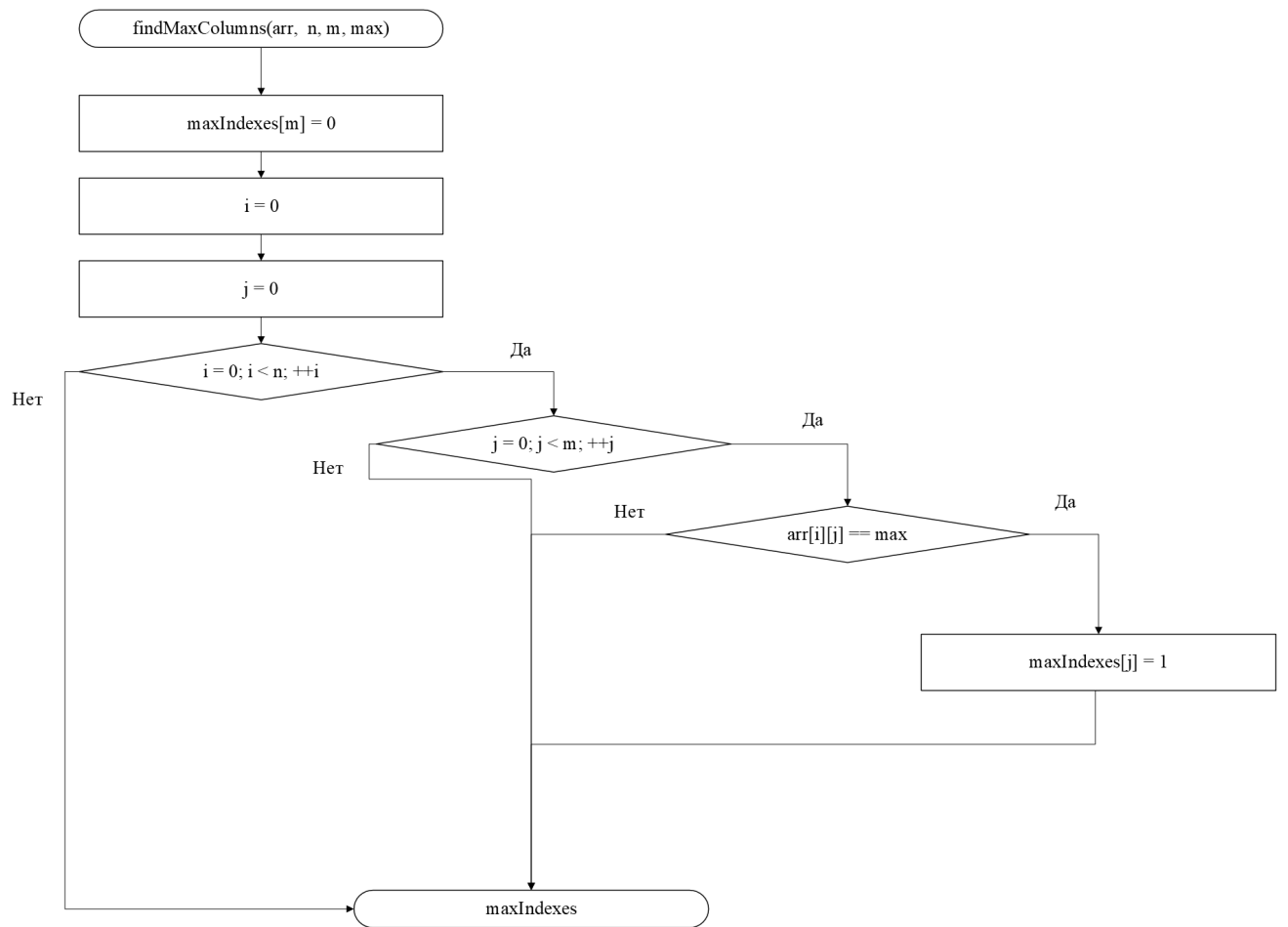
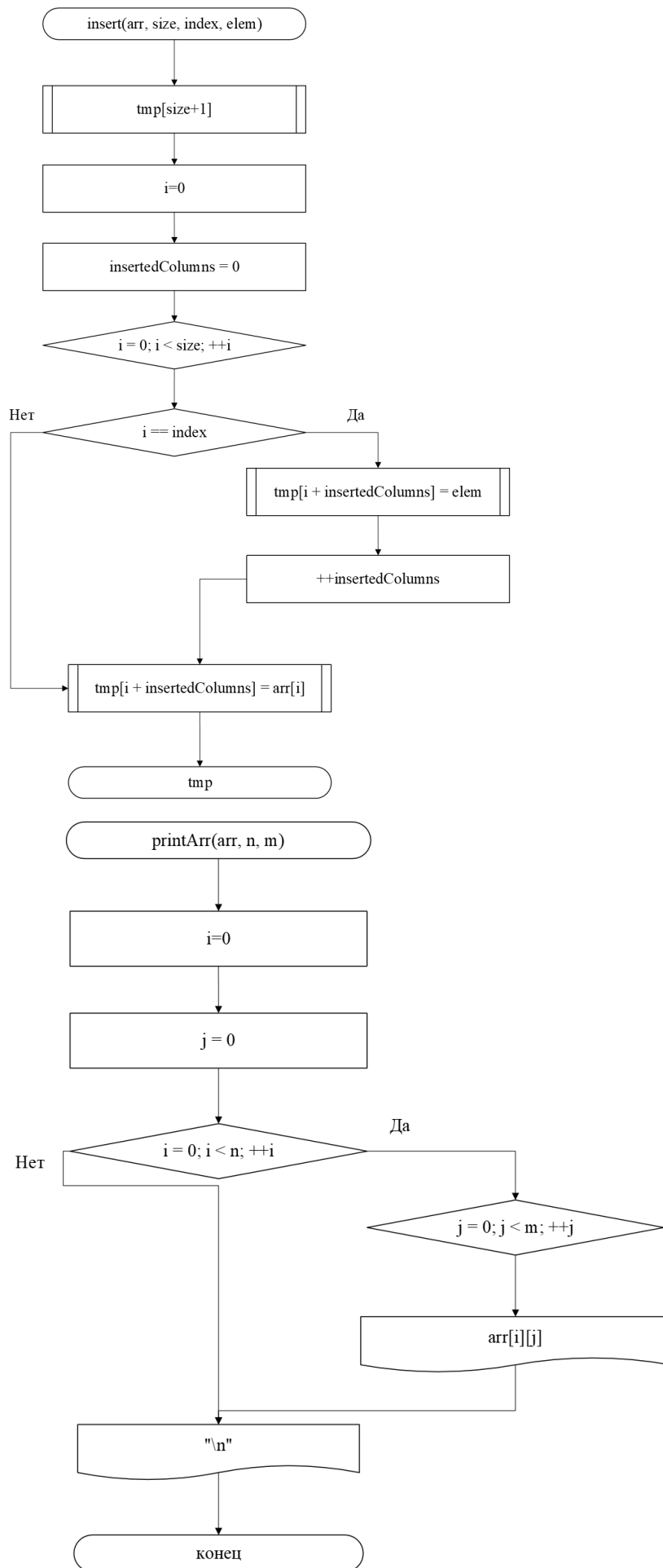
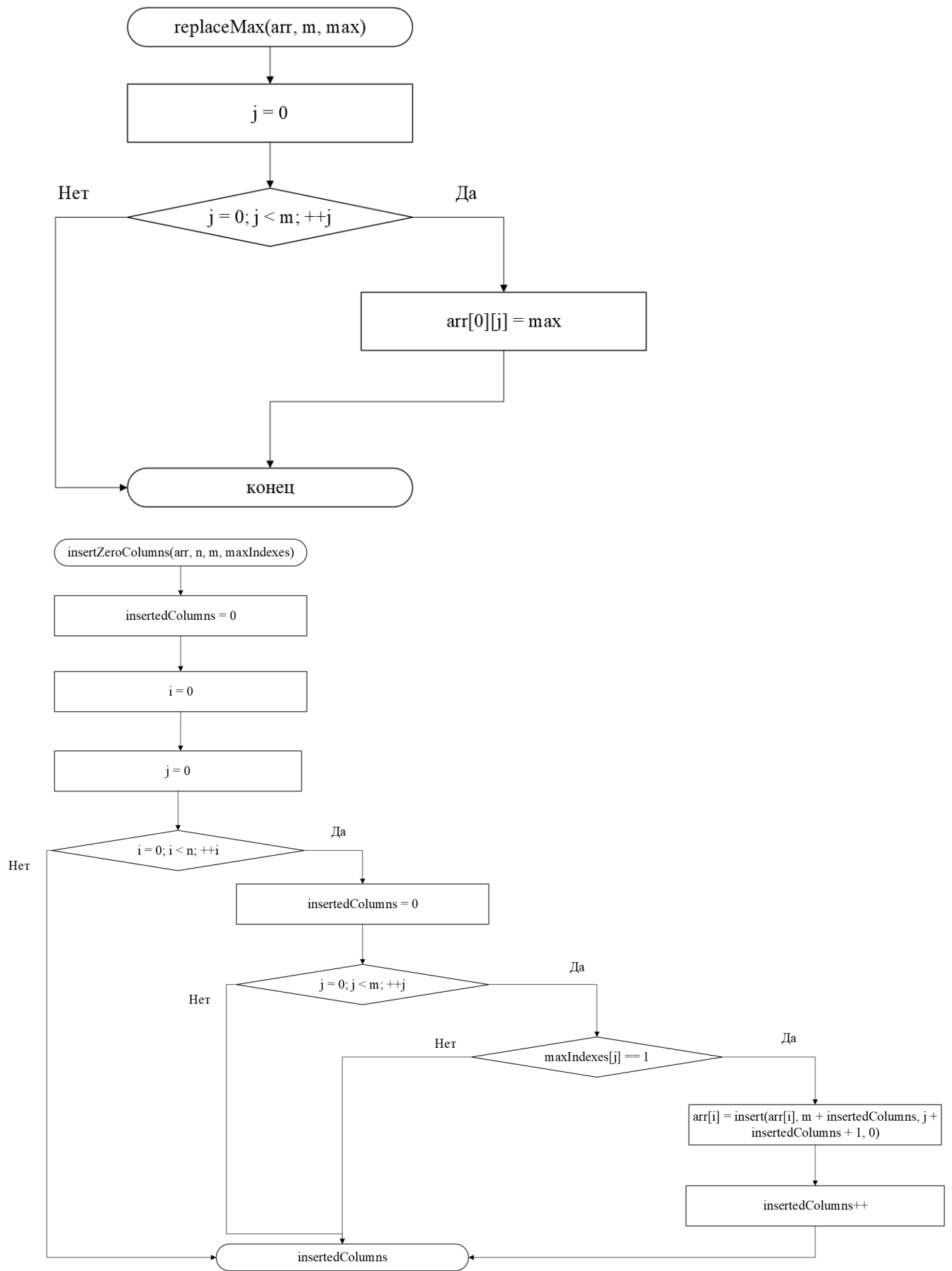


Рисунок 2 – Блок-схема используемых функций









### 3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C

```
#include "stdio.h"
#include "stdlib.h"
#include "time.h"
#include "stdbool.h"
#include "string.h"
#include "math.h"

enum Choice {manually, randomly};

/**
 * @brief печатает массив на экран
 * @param **arr исходный массив, n и m - количество строк и
 столбцов соответственно
 */
void printArr(int **arr, size_t n, size_t m);

/**
 * @brief Освобождает массив по указателю
 * @param **arr исходный массив, n - количество строк
 */
void deleteArr(int **arr, size_t n);

/**
 * @brief считывает строку и проверяет, что в ней только числа
 * @param *arr строка матрицы (одномерный массив), size размер
 массива, index - индекс, после которого нужно вставить, elem -
 элемент, который нужно вставить
 * @return изменённый массив
 */
int *insert(int *arr, size_t size, size_t index, int elem);

/**
 * @brief заполняет массив целыми числами
 * @param ***arr двумерный массив, n и m - размеры массива
 */
void randomInput(int ***arr, size_t n, size_t m);

/**
 * @brief заполняет массив пользовательским вводом
 * @param ***arr двумерный массив, n и m - размеры массива
 */
void userInput(int ***arr, size_t n, size_t m);

/**
 * @brief ищет максимальный элемент массива
 * @param **arr массив, n и m размер массива
 * @return максимальный элемент
 */
int findMax(int **arr, size_t n, size_t m);
```



```

/**
 * @brief ищет максимальный элемент массива
 * @param ***arr массив, n и m размер массива, max - максимальный
элемент массива
 * @return массив bool, где на каждом индексе j стоит или не
стоит флаг, есть ли в столбце максимальный элемент
 */
int *findMaxColumns(int **arr, size_t n, size_t m, int max);

/**
 * @brief заменяет 0-е элементы столбца максимальным элементом в
массиве
 * @param ***arr массив, m количество столбцов в массиве, max -
максимальный элемент массива
 */
void replaceMax(int ***arr, size_t m, int max);

/**
 * @brief вставляет нули после столбца, содержащего максимальный
элемент в массиве
 * @param ***arr массив, n и m размер массива, maxIndexes -
массив bool, где на каждом индексе j стоит или не стоит флаг,
есть ли в столбце максимальный элемент
 * @return изменённый массив
 */
int insertZeroColumns(int ***arr, size_t n, size_t m, int
*maxIndexes);

/**
 * @brief выделяет память для массива
 * @param n и m размер массива
 * @return созданный массив
 */
int **initArray(size_t n, size_t m);

/**
 * @brief считывает целое число из stdin
 * @param message сообщение, выводимое пользователю перед вводом
 * @return введённое число
 */
size_t getSize(const char *message);

/**
 * @brief считывает массив целых чисел из stdin или заполняет его
случайными числами (выбор предоставляется пользователю при
вводе)
 * @param n, m Количество строк и столбцов в массиве
соответственно
 * @return полученный массив
 */
int **getIntArr(size_t n, size_t m);

/**

```

```

* @brief предлагают пользователю выбор, заполнять массив
случайными числами или числами из пользовательского ввода
(stdin)
* @return выбор пользователя
*/
enum Choice getChoice();

/**
* @brief целое число из stdin и возвращает результат
* @param message сообщение, выводимое пользователю перед вводом
* @return считанное целое число
*/
int getInt(const char *message);

/**
* @brief проверяет размер на корректность, если размер
некорректен, выходит из программы
* @param n, m размер
*/
void checkSize(size_t n);

int main() {
    size_t n = getSize("Enter count of rows in array: ");
    size_t m = getSize("Enter count of columns in array: ");
    int **arr = getIntArr(n, m);
    int max = findMax(arr, n, m);
    int *maxIndexes = findMaxColumns(arr, n, m, max);
    printArr(arr, n, m);
    // Заменяем нулевой элемент максимальным по модулю элементом
массива
    replaceMax(&arr, m, max);
    int insertedColumns = insertZeroColumns(&arr, n, m,
maxIndexes);
    puts("Output array:\n");
    printArr(arr, n, m + insertedColumns);
    deleteArr(arr, n);
    return 0;
}

size_t getSize(const char *message) {
    int value = getInt(message);
    checkSize(value);
    return (size_t)value;
}

int **getIntArr(size_t n, size_t m) {
    if (m <= 0 || n <= 0) {
        puts("Invalid shape of array");
        abort();
    }
    enum Choice filling = getChoice();
    int **arr = initArray(n, m);
    if (filling == randomly) {

```

```

        randomInput(&arr, n, m);
    } else if (filling == manually) {
        userInput(&arr, n, m);
    } else {
        puts("Invalid choice!");
        abort();
    }
    return arr;
}

int *insert(int *arr, size_t size, size_t index, int elem) {
    int *tmp = (int*)calloc(size + 1, sizeof(size_t));
    memset(tmp, 0, size + 1);
    size_t i = 0;
    int insertedColumns = 0;
    for (i = 0; i < size; ++i) {
        if (i == index) {
            tmp[i + insertedColumns] = elem;
            ++insertedColumns;
        }
        tmp[i + insertedColumns] = arr[i];
    }
    free(arr);
    arr = tmp;
    return tmp;
}

void printArr(int **arr, size_t n, size_t m) {
    puts("Array:\n");
    size_t i = 0;
    size_t j = 0;
    for (i = 0; i < n; ++i) {
        for (j = 0; j < m; ++j) {
            printf("%d\t", arr[i][j]);
        }
        puts("\n");
    }
}

void randomInput(int ***arr, size_t n, size_t m) {
    srand(time(NULL));
    size_t i = 0, j = 0;
    for (i = 0; i < n; ++i) {
        for (j = 0; j < m; ++j) {
            (*arr)[i][j] = rand() % 1000;
        }
    }
}

void userInput(int ***arr, size_t n, size_t m) {
    puts("Enter array elements:\n");

```

```

        int i = 0, j = 0;
        for (i = 0; i < n; ++i) {
            for (j = 0; j < m; ++j) {
                (*arr)[i][j] = getInt(NULL);
            }
            puts("\n");
        }
    }

    int findMax(int **arr, size_t n, size_t m) {
        int max = arr[0][0];
        int i = 0;
        int j = 0;
        for (i = 0; i < n; ++i) {
            for (j = 0; j < m; ++j) {
                if (abs(arr[i][j]) > max) {
                    max = arr[i][j];
                }
            }
        }
        return max;
    }

    int *findMaxColumns(int **arr, size_t n, size_t m, int max) {
        int *maxIndexes = calloc(m, sizeof(int));
        memset(maxIndexes, 0, m);
        int i = 0, j = 0;
        for (i = 0; i < n; ++i) {
            for (j = 0; j < m; ++j)
            {
                if (arr[i][j] == max) {
                    // если хотя одна строка содержит на индексе
j максимальное число, значит столбец j содержит максимальное
число
                    maxIndexes[j] = 1;
                }
            }
        }
        return maxIndexes;
    }

    void replaceMax(int ***arr, size_t m, int max) {
        int j = 0;
        for (j = 0; j < m; ++j) {
            (*arr)[0][j] = max;
        }
    }

    int insertZeroColumns(int ***arr, size_t n, size_t m, int
*maxIndexes) {
        int insertedColumns = 0;
        int i = 0, j = 0;
        for (i = 0; i < n; ++i) {

```

```

        insertedColumns = 0;
        for (j = 0; j < m; ++j) {
            if (maxIndexes[j]) {
                (*arr)[i] = insert((*arr)[i], m +
insertedColumns, j + insertedColumns + 1, 0);
                insertedColumns++;
            }
        }
    }
    return insertedColumns;
}

int **initArray(size_t n, size_t m) {
    int **arr = (int**)calloc(n, sizeof(int*));
    size_t i = 0;
    for (i = 0; i < n; ++i) {
        arr[i] = (int*)calloc(m, sizeof(int));
    }
    return arr;
}

enum Choice getChoice() {
    printf("\nDo you want to enter the array or fill it random
digits? (%d - for enter, %d - for random): ", (int)manually,
(int)randomly);
    int temp = getInt(NULL);
    return (enum Choice)temp;
}

void deleteArr(int **arr, size_t n) {
    int i = 0;
    for (i = 0; i < n; ++i) {
        free(arr[i]);
    }
    free(arr);
}

int getInt(const char *message) {
    if (message) {
        printf("%s", message);
    }
    int value = 0;
    if (scanf("%d", &value) != 1) {
        puts("Error!\n");
        abort();
    }
    // По условию нужны целые числа, т.е. знаковые, поэтому знак
не проверяем
    return value;
}

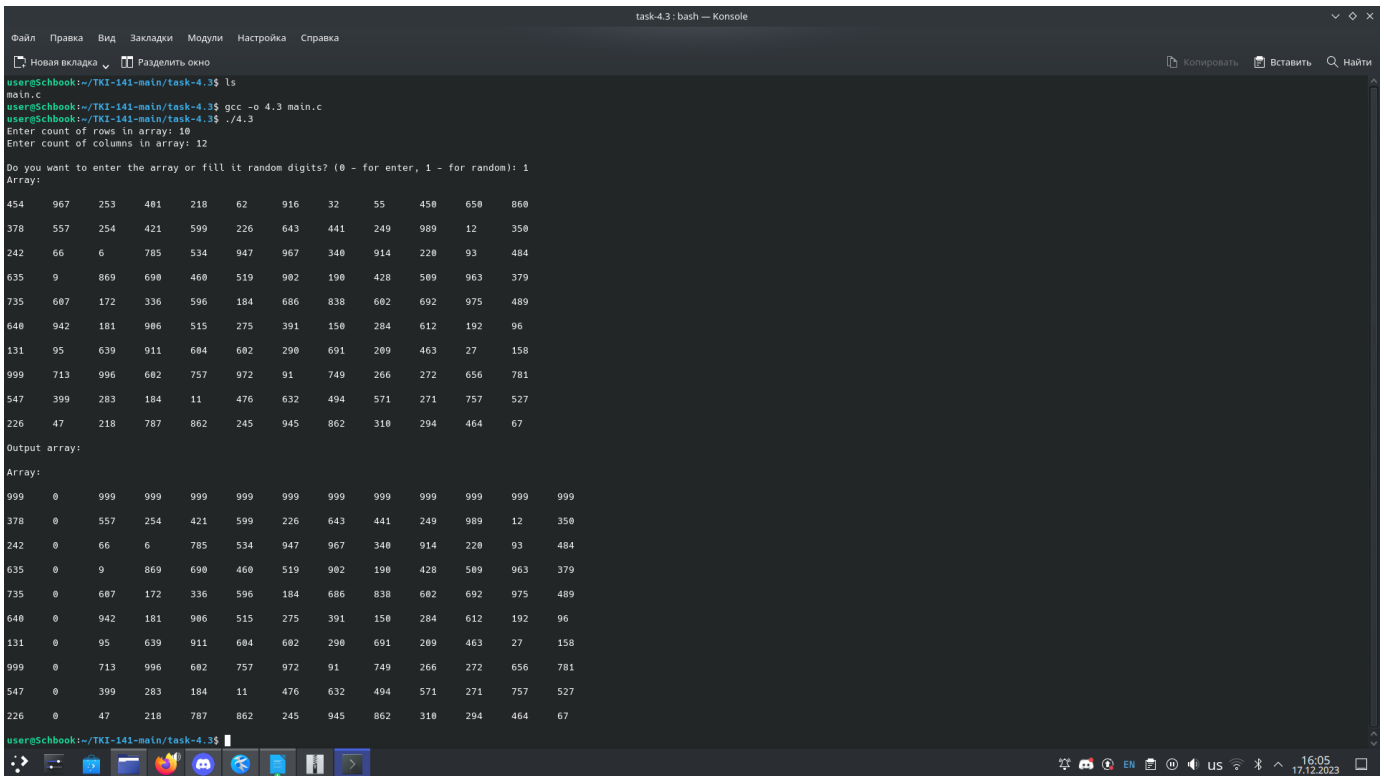
void checkSize(size_t n) {
    if (n <= 0) {

```

```
        puts("Error!\n");  
        abort();  
    }  
}
```

## 4 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Результаты выполнения программы представлены ниже (Рисунок 3).



```
task-4.3: bash — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
[+] Новая вкладка  [x] Разделить окно
Копировать  Вставить  Найти
user@Schbook:~/TKI-141-main/task-4.3$ ls
main.c
user@Schbook:~/TKI-141-main/task-4.3$ gcc -o 4.3 main.c
user@Schbook:~/TKI-141-main/task-4.3$ ./4.3
Enter count of rows in array: 10
Enter count of columns in array: 12
Do you want to enter the array or fill it random digits? (0 - for enter, 1 - for random): 1
Array:
454  967  253  481  218  62  916  32  55  458  658  868
378  557  254  421  599  226  643  441  249  989  12  358
242  66  6  785  534  947  967  348  914  228  93  484
635  9  869  698  468  519  982  198  428  589  963  379
735  687  172  336  596  184  686  838  682  692  975  489
648  942  181  986  515  275  391  158  284  612  192  96
131  95  639  911  684  682  298  691  289  463  27  158
999  713  996  682  757  972  91  749  266  272  656  781
547  399  283  184  11  476  632  494  571  271  757  527
226  47  218  787  862  245  945  862  318  294  464  67
Output array:
Array:
999  0  999  999  999  999  999  999  999  999  999  999
378  0  557  254  421  599  226  643  441  249  989  12  358
242  0  66  6  785  534  947  967  348  914  228  93  484
635  0  9  869  698  468  519  982  198  428  589  963  379
735  0  687  172  336  596  184  686  838  682  692  975  489
648  0  942  181  986  515  275  391  158  284  612  192  96
131  0  95  639  911  684  682  298  691  289  463  27  158
999  0  713  996  682  757  972  91  749  266  272  656  781
547  0  399  283  184  11  476  632  494  571  271  757  527
226  0  47  218  787  862  245  945  862  318  294  464  67
user@Schbook:~/TKI-141-main/task-4.3$
```

Рисунок 3 – Результаты выполнения программы

## 5 ОТМЕТКА О ВЫПОЛНЕНИИ ЗАДАНИЯ В ВЕБ-ХОСТИНГЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ

