1. Лабораторная работа № 4

«Обработка двумерных массивов с помощью функций»

* 1. Цель работы

Изучить основные принципы работы с массивами в языках С/С++. Исследовать способы передачи параметров в функции.

## Постановка задачи

Получить навыки работы с матрицами; создания функций, способных работать с матрицами.

* 1. Ход выполнения работы

Вариант выполнения задания – 18.

* + 1. Описание алгоритма решения задачи
       1. Определение входных и выходных данных

В главном меню программы пользователь вводит целое число, тем самым пользователь выбирает, что должна будет программа делать дальше. При вводе другого числа программа выведет сообщение об ошибке и попросить пользователя ввести число ещё раз.

При создании матрицы программа сначала попросит ввести два натуральных числа – количество строк и столбцов в матрице соответственно. Затем программа попросит пользователя ввести n\*m элементов матрицы. Каждый элемент принадлежит области целых чисел.

* + - 1. Структурная схема алгоритма

Структурная схема алгоритма представлена на Рисунках 4.1, 4.2 и 4.3.

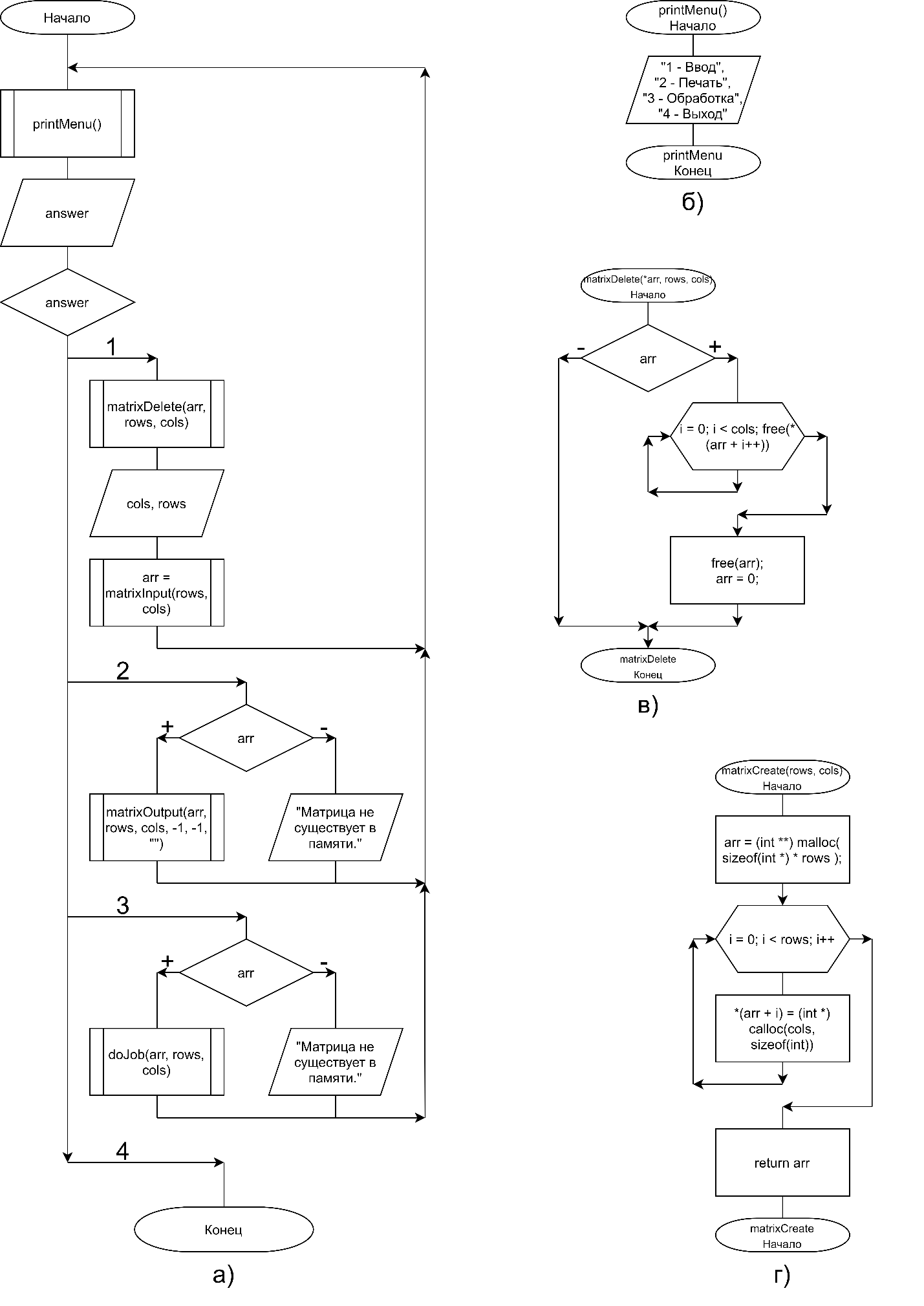


Рисунок 4.1 – Структурные схемы: а) основной программы;

б) функции printMenu;

в) функции matrixDelete; г) функции matrixCreate

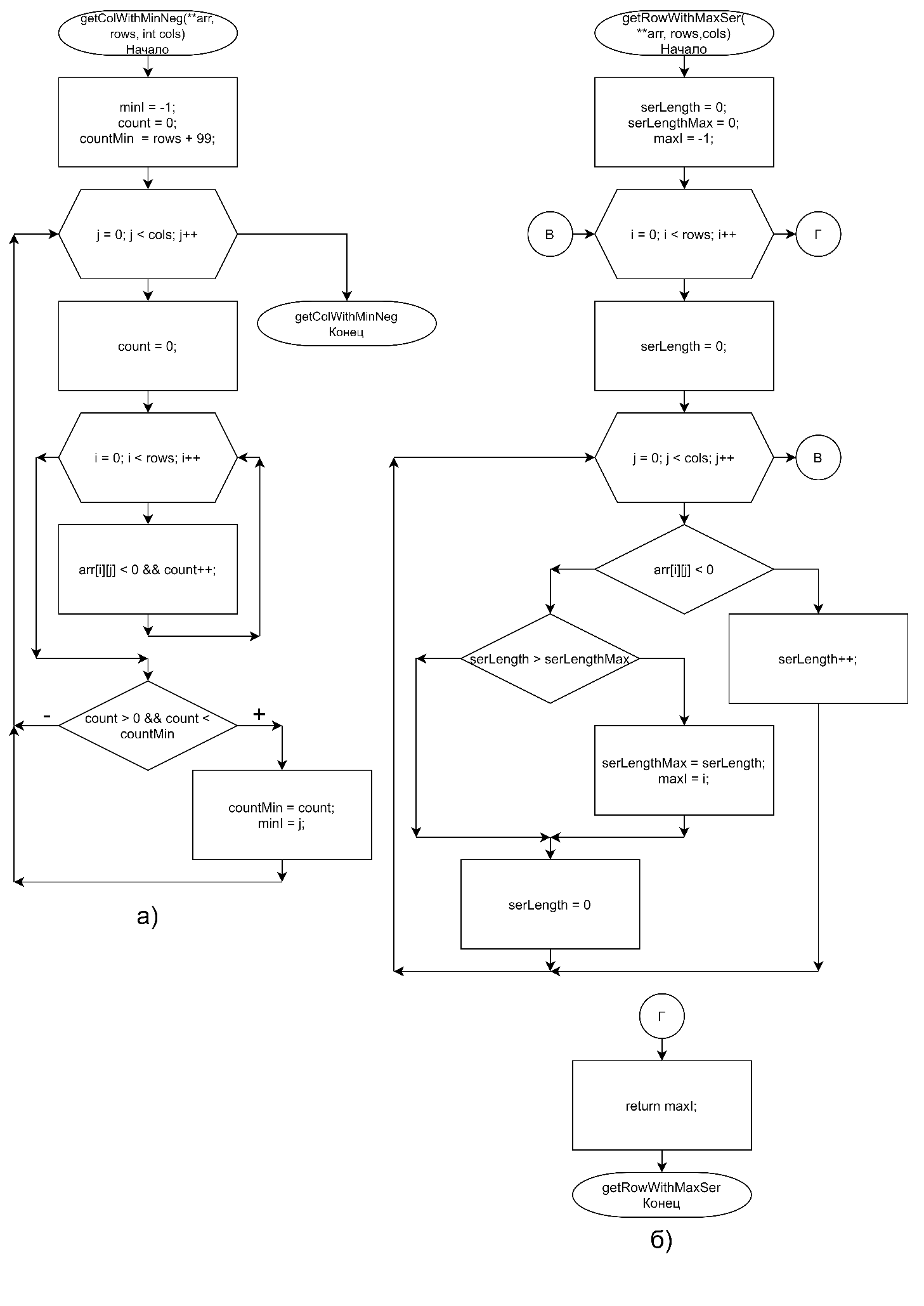


Рисунок 4.2 – Структурные схемы функций: а) getColWithMinNeg;

б) getRowWithMaxSer

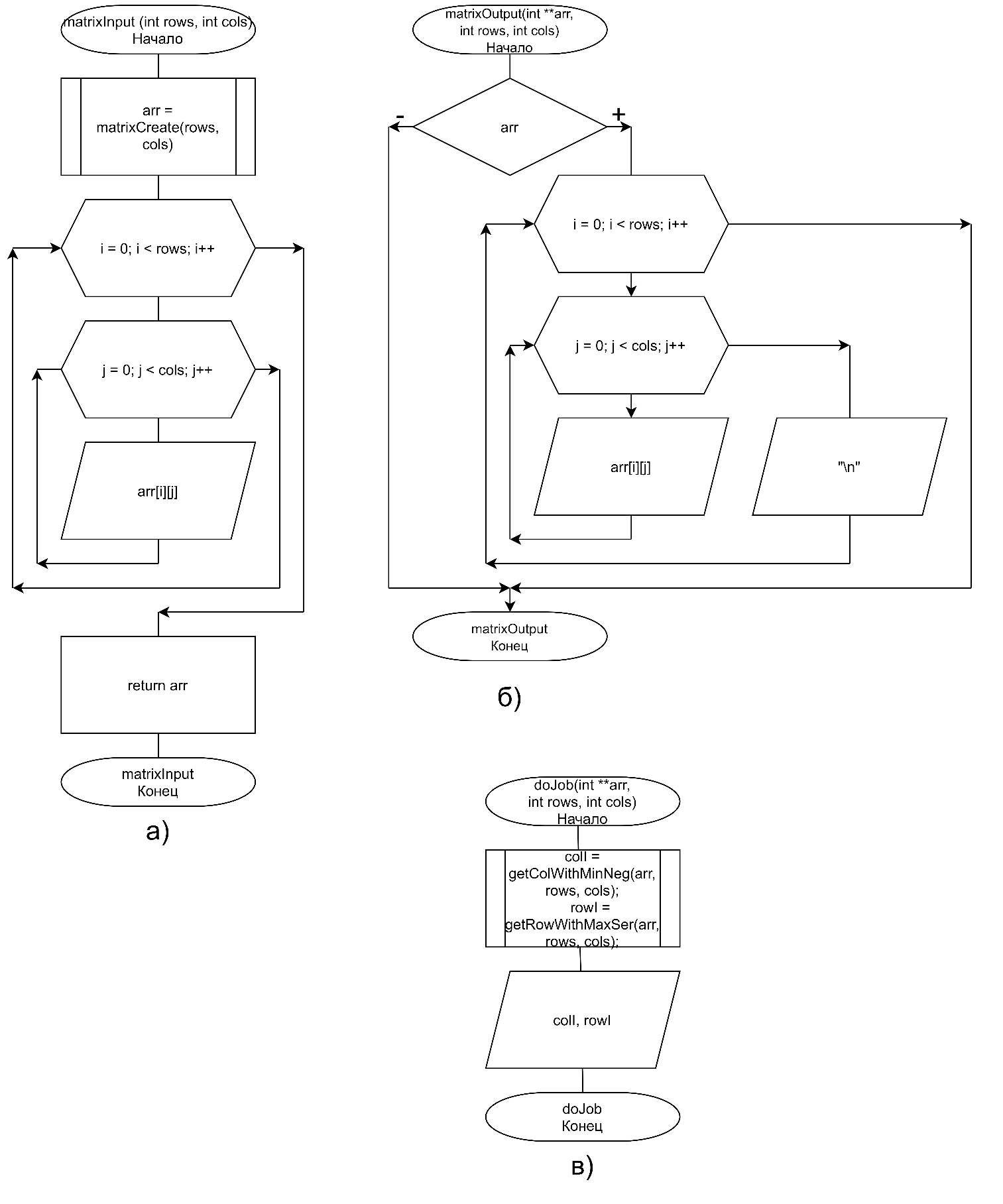


Рисунок 4.3 – Структурные схемы функций: а) matrixInput;

б) matrixOutput; в) doJob

* + 1. Текст программы (ЯП C, операционная система GNU/Linux Ubuntu)

#include <stdio.h>

#include <stdlib.h>

#define CLRSCR() fflush(NULL); system("clear")

#define pause(a) fflush(NULL); printf("Для продолжения нажмите %c ...\n", a);\

while (getchar() != a);

#define printTitle(title) printf("Запущена функция: %s ...\n\n", title)

// Раздел описания цифр для выбора действия

#define PROGRAM\_EXIT 4

#define PROGRAM\_DO\_JOB 3

#define PROGRAM\_INPUT\_ARR 1

#define PROGRAM\_PRINT\_ARR 2

// Раздел описания соответствий названий цветов их последовательностям

#define FONT\_DEFAULT "\e[0m"

#define FONT\_RED "\e[0;91m"

#define FONT\_YELLOW "\e[0;93m"

#define FONT\_GREEN "\e[0;92m"

#define FONT\_BLUE "\e[0;94m"

// Раздел объявления прототипов функций

void printMenu (); // Печать меню

int \*\*matrixInput (int rows, int cols); // Функция создания матрицы и ввода значений элементов

void matrixOutput(int \*\*arr, int rows, int cols, int HLCol, int HLRow, char HLColor[]); // Функция печати матрицы

int \*\*matrixCreate(int rows, int cols);// Функция выделения памяти под матрицу

void matrixDelete(int \*\*arr, int rows, int cols); // Функция удаления матрицы

void doJob(int \*\*arr, int rows, int cols);// Функция запуска выполнения задания по

// варианту

int getColWithMinNeg (int \*\*arr, int rows, int cols); // Функция нахождения

// столбца с минимальным количеством отрицательных элементов

int getRowWithMaxSer (int \*\*arr, int rows, int cols); // Функция нахождения

// строки с максимальной серией

//////////////////////////////////////////////////////////////////////////////////

int main ()

{

int answer, rows = 0, cols = 0;

int \*\*arr = 0;

do

{

printMenu();

printf("> "); scanf("%d", &answer);

switch (answer)

{

case PROGRAM\_EXIT:

printTitle("Выход из программы");

matrixDelete(arr, rows, cols);

break;

case PROGRAM\_INPUT\_ARR:

printTitle("Ввод матрицы");

matrixDelete(arr, cols, rows);

printf("Для начала введите количество строк в матрице: "); scanf("%d", &rows);

printf("А теперь введите количество столбцов в матрице: "); scanf("%d", &cols);

arr = matrixInput(rows, cols);

break;

case PROGRAM\_PRINT\_ARR:

printTitle("Вывод матрицы на экран");

if (arr)

matrixOutput(arr, rows, cols, -1, -1, "");

else

printf("Матрица не существует в памяти. Для начала создайте её.");

printf("\n");

break;

case PROGRAM\_DO\_JOB:

printTitle("Выполнение задания по варианту");

if (arr)

doJob(arr, rows, cols);

else

printf("Над несуществующей матрицей невозможно выполнить работу.");

printf("\n");

break;

default:

printf("Введена неверная цифра: %d. Повторите попытку...", answer);

break;

}

pause('=');

} while (answer != PROGRAM\_EXIT);

}

//////////////////////////////////////////////////////////////////////////////////

void printMenu()

{

CLRSCR();

printf("=====Меню программы=====\n");

printf("%2d) %s.\n", PROGRAM\_INPUT\_ARR, "Ввод матрицы" );

printf("%2d) %s.\n", PROGRAM\_PRINT\_ARR, "Вывести матрицу на экран" );

printf("%2d) %s.\n", PROGRAM\_DO\_JOB, "\*my\* Выполнить задание по варианту");

printf("%2d) %s.\n", PROGRAM\_EXIT, "Выйти из программы" );

return;

}

//////////////////////////////////////////////////////////////////////////////////

void matrixDelete(int \*\*arr, int rows, int cols)

{

if (arr)

{

for (int i = 0; i < cols; free(\*(arr + i++)));

free(arr); arr = 0;

}

}

//////////////////////////////////////////////////////////////////////////////////

int \*\*matrixCreate(int rows, int cols)

{

int \*\*arr = 0;

arr = (int \*\*) malloc( sizeof(int \*) \* rows );

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

\*(arr + i) = (int \*) calloc(cols, sizeof(int));

return arr;

}

//////////////////////////////////////////////////////////////////////////////////

int \*\*matrixInput (int rows, int cols)

{

int \*\*arr = 0;

arr = matrixCreate(rows, cols);

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

{

for (int j = 0; j < cols; j++)

{

CLRSCR();

matrixOutput(arr, rows, cols, i, j, FONT\_RED);

printf("Введите элемент arr[%d][%d]: ", i + 1, j + 1); scanf("%d", \*(arr + i) + j);

}

}

CLRSCR();

printf("Отлично! Новая матрица была создана!\n");

return arr;

}

//////////////////////////////////////////////////////////////////////////////////

void matrixOutput(int \*\*arr, int rows, int cols, int HLCol, int HLRow, char HLColor[])

{

if (arr)

{

printf(" ");

for (int i = 0; i++ < cols; printf("%3d ", i));

printf("\n");

printf(" \_");

for (int i = 0; i++ < cols; printf("\_\_\_\_"));

printf("\_\_\n");

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

{

printf("%3d| ", i + 1);

for (int j = 0; j < cols; j++)

if ( (HLCol == i && HLRow == j) || ( HLRow < 0 && HLCol == i ) || ( HLCol < 0 && HLRow == j ) )

printf("%s%3d \e[0m", HLColor, arr[i][j]);

else

printf("%3d ", arr[i][j]);

printf("\n");

}

printf(" |\n");

}

}

//////////////////////////////////////////////////////////////////////////////////

void doJob(int \*\*arr, int rows, int cols)

{

int colI= 0, // Номер столбца, содержащий минимальное количество

// отрицательных элементов

rowI = 0; // Номер строки, содержащей самую длинную серию отрицательных

// элементов

colI = getColWithMinNeg(arr, rows, cols);

rowI = getRowWithMaxSer(arr, rows, cols);

printf("Действие 1: поиск столбца, содержащего минимальное количество отрицательных элементов.\n");

printf("Результат:\n \* ");

if (colI >= 0)

{

printf("Был найден столбец под номером %d. Для удобства он выделен ЗЕЛЁНЫМ цветом.\n", colI + 1);

matrixOutput(arr, rows, cols, -1, colI, FONT\_GREEN);

}

else

printf("Не был найден ни один столбец, содержащий хотя бы один отрицательный элемент.\n");

printf("\n================================================================\n\n");

printf("Действие 2: поиск строки, содержащей максимальную серию из отрицательных элементов.\n");

printf("Результат:\n \* ");

if (rowI >= 0)

{

printf("Была найдена строка под номером %d. Для удобства она выделена СИНИМ цветом.\n", rowI + 1);

matrixOutput(arr, rows, cols, rowI, -1, FONT\_BLUE);

}

else

printf("Не была найдена ни одина строка, содержащая хотя бы один отрицательный элемент.\n");

printf("\n================================================================\n");

}

//////////////////////////////////////////////////////////////////////////////////

int getColWithMinNeg(int \*\*arr, int rows, int cols)

{

int i = 0, j = 0, // Переменные для циклов

minI = -1, // Содержит номер столбца (если таковой будет

// найден)

countMin = rows + 99, // Минимальное количество отрицательных элементов в

// найденном столбце

count = 0; // Количество отрицательных элементов в

// рассматриваемом столбце

for (j = 0; j < cols; j++) // Номер каждого столбца

{

count = 0;

for (i = 0; i < rows; i++) // Номер каждой строчки

if ( arr[i][j] < 0 )

count++;

if (count > 0 && count < countMin)

{

countMin = count;

minI = j;

}

}

return minI;

}

//////////////////////////////////////////////////////////////////////////////////

int getRowWithMaxSer(int \*\*arr, int rows, int cols)

{

int serLength = 0, // Длина текущей серии

serLengthMax = 0, // Максимальная длина серии

maxI = -1, // Номер строки с максимально длинной серией (если

// таковая будет найдена)

i, j; // Переменные для циклов

for (i = 0; i < rows; i++) // Каждая строчка

{

serLength = 0;

for (j = 0; j < cols; j++) // Каждый столбец

{

if (arr[i][j] < 0)

serLength++;

else

{

if (serLength > serLengthMax)

{ serLengthMax = serLength; maxI = i; }

serLength = 0;

}

}

if (serLength > serLengthMax)

{ serLengthMax = serLength; maxI = i; }

}

return maxI;

}

* + 1. Описание тестовых примеров

На Рисунке 4.4 продемонстрировано меню программы.

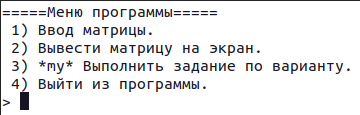


Рисунок 4.4 – Меню программы

На Рисунке 4.5 продемонстрирован ввод элемента в матрицу 4х5.

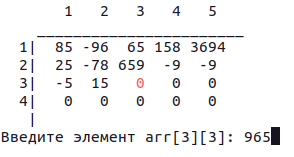


Рисунок 4.5 – Ввод элемента 3 3 в матрицу

На Рисунке 4.6 продемонстрирован вывод матрицы на экран.

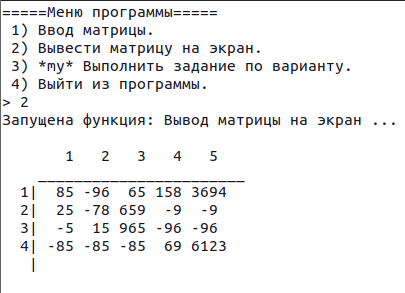


Рисунок 4.6 – Печать матрицы на экран

На Рисунках 4.7, 4.8 и 4.9 продемонстрированы обработки трёх разных матриц.

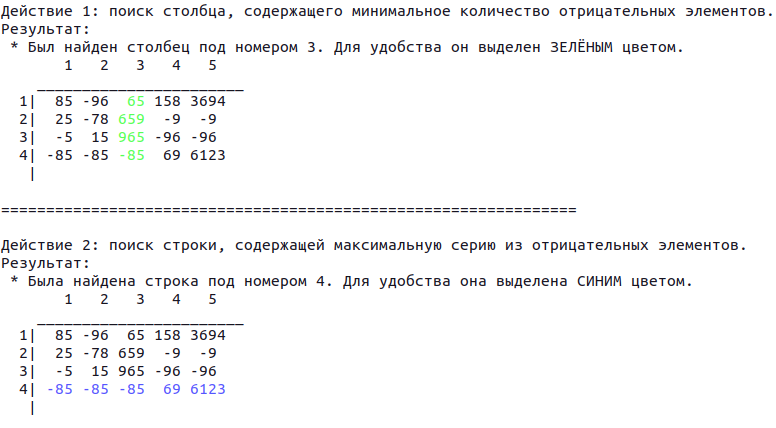


Рисунок 4.7 – Обработка матрицы № 1

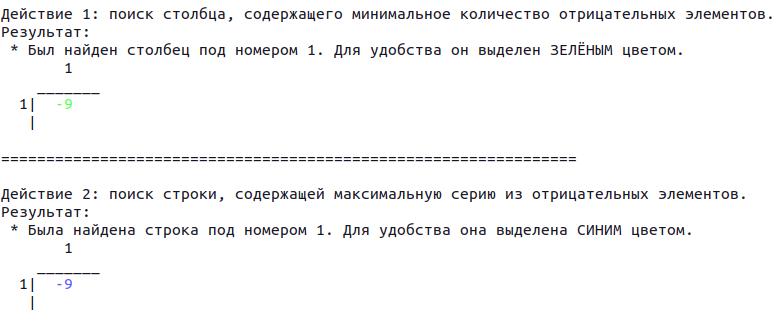


Рисунок 4.8 – Обработка матрицы № 2

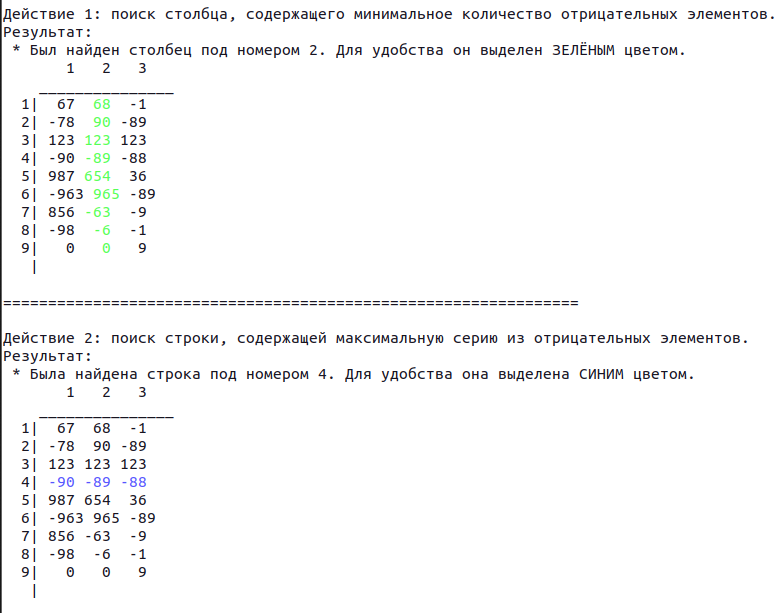


Рисунок 4.9 – Обработка матрицы № 3

Если пользователь попытается обработать матрицу, которая не была создана, то программа выдаст ошибку (Рисунок 4.10).

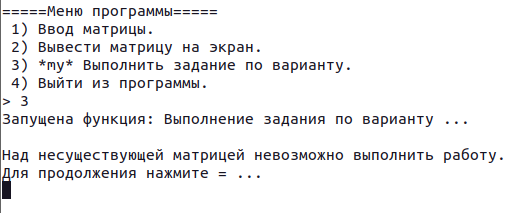


Рисунок 4.10 – Обработка несуществующей матрицы

Из программы можно выйти через меню, введя цифру 4 (Рисунок 4.11).

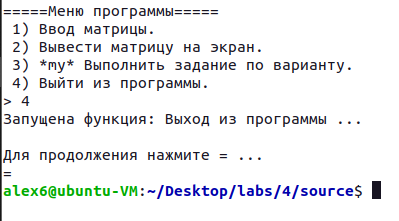


Рисунок 4.11 – Выход из программы

## Вывод

В ходе выполнения данной лабораторной работы были получены навыки работы с функциями, с двумерными массивами, ссылками и указателями, указывающими на другой указатель. Выявлено, что при работе с матрицами удобно использовать двумерные массивы; в больших программах более рационально и эстетично использовать функции. Так же были повторно закреплены навыки работы с одномерными массивами, с обычными указателями, с циклами for, while и do..while, с встроенными функциями malloc, realloc и free.