Федеральное государственное автономное образовательное учреждение высшего образования

«ОМСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Информатика и вычислительная техника.

Лабораторная работа №9

по дисциплине «Программирование» на тему:

«Программирование с возвратом из функций нескольких значений.»

Выполнил: студент группы ИВТ-244 Шмидт Антон Владиславович

Проверил: ассистент кафедры ИВТ Горшенин Алексей Юрьевич

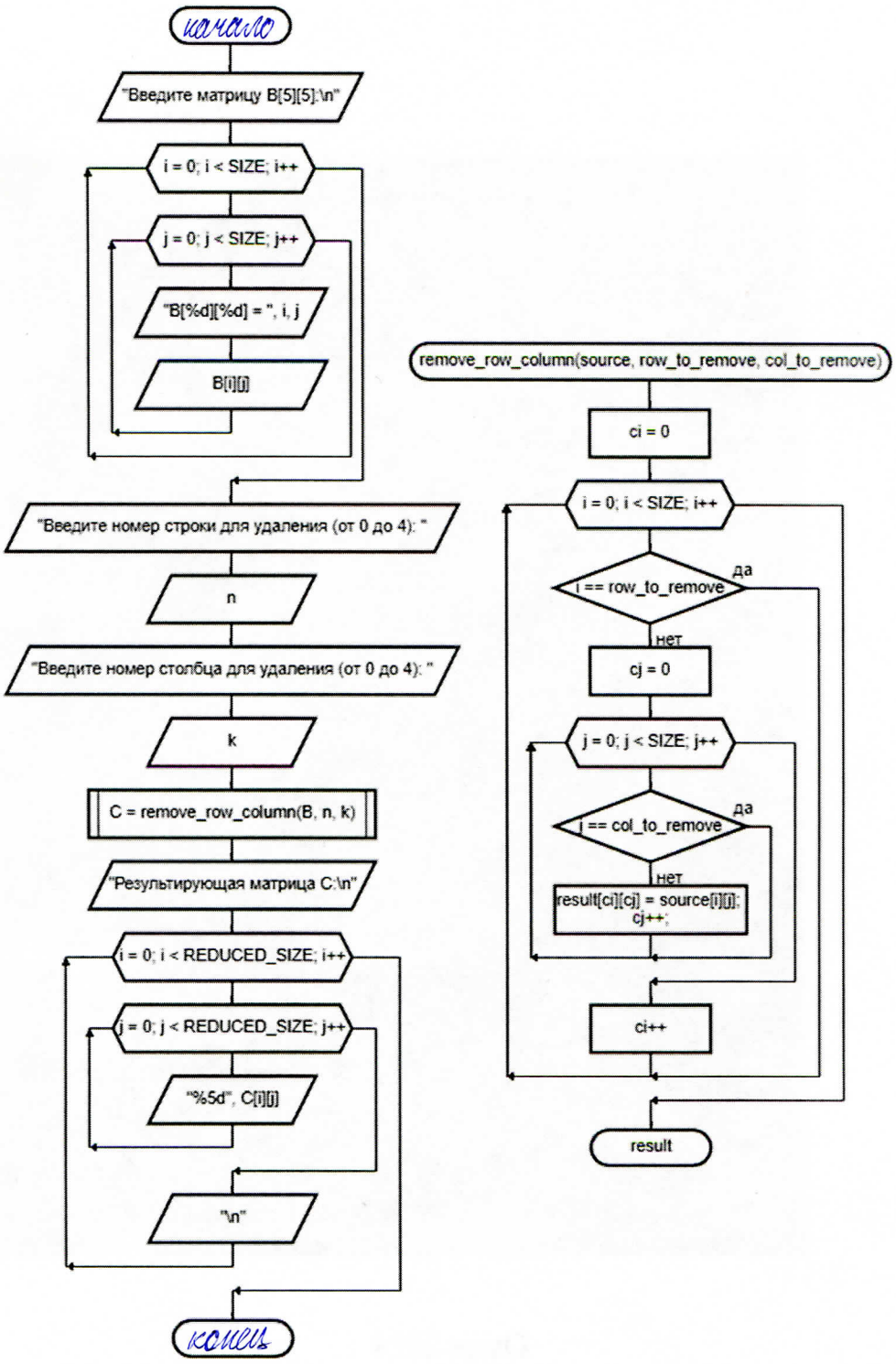
15.12.24

Омск 2024

Задача 1 (программа 9_1)

Задача: написать и отладить программу для задачи из лабораторной работы 7 (программа7_1) с выделением алгоритма обработки или формирования массива. Организовать вызов процедуры с параметрами-массивами для двух наборов исходных данных. Обработку массив выполнить в подпрограмме, а ввод исходных данных и вывод результатов в главной функции. (Для ускорения отладки массив можно задать константой).

Схема алгоритма:



Решение кодом:

```
#include <locale.h>
#include <stdio.h>

#define SIZE 5
#define REDUCED_SIZE 4

void remove_row_column(int source[SIZE][SIZE], int
result[REDUCED_SIZE][REDUCED_SIZE], int row_to_remove, int
col_to_remove) {
    int ci = 0;
    for (int i = 0; i < SIZE; i++) {
        if (i == row_to_remove) continue;
        int cj = 0;
        for (int j = 0; j < SIZE; j++) {
            if (j == col_to_remove) continue;
            result[ci][cj] = source[i][j];
            cj++;
        }
        ci++;
    }
}

void main9_1() {
    setlocale(LC_ALL, "ru_RU");

    int B[SIZE][SIZE], C[REDUCED_SIZE][REDUCED_SIZE];
    int n, k;

    printf("Введите матрицу B[5][5]:\n");
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf("B[%d][%d] = ", i, j);
            scanf_s("%d", &B[i][j]);
        }
    }

    printf("Введите номер строки для удаления (от 0 до 4): ");
    scanf_s("%d", &n);

    printf("Введите номер столбца для удаления (от 0 до 4): ");
    scanf_s("%d", &k);

    remove_row_column(B, C, n, k);

    printf("Результирующая матрица C:\n");
    for (int i = 0; i < REDUCED_SIZE; i++) {
        for (int j = 0; j < REDUCED_SIZE; j++) {
            printf("%5d", C[i][j]);
        }
        printf("\n");
    }
}
```

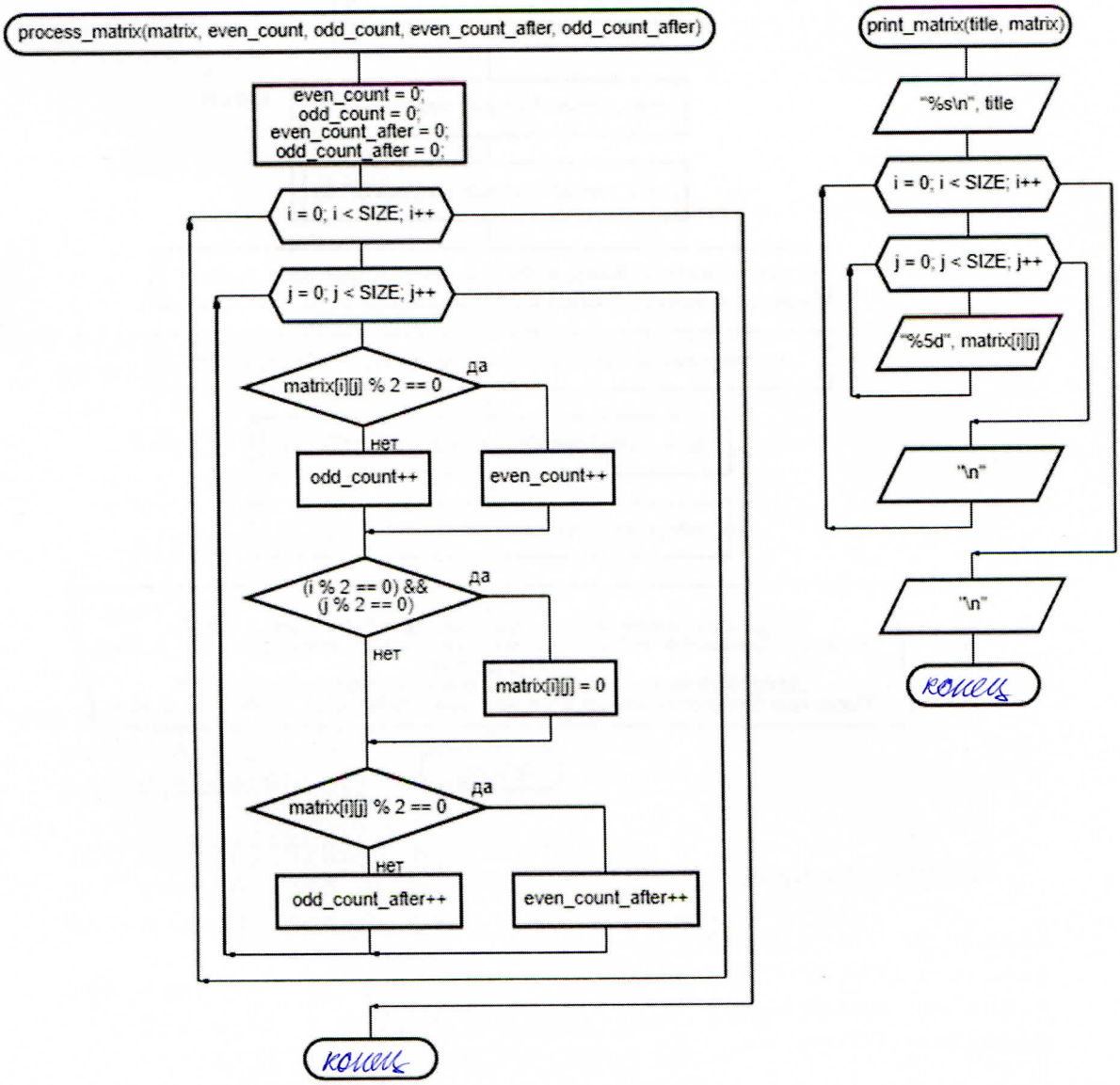
Результат работы:

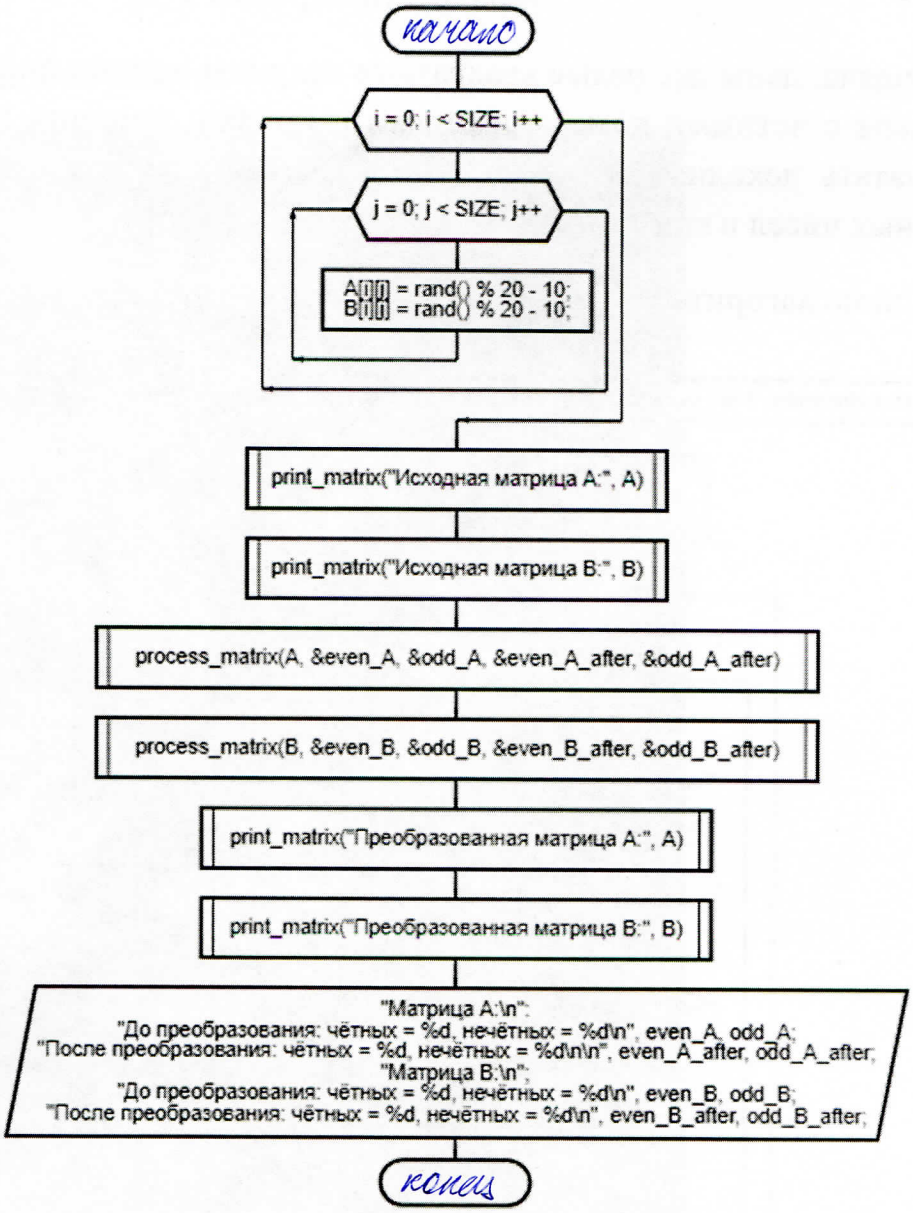
```
Введите матрицу B[5][5]:
B[0][0] = 1
B[0][1] = 2
B[0][2] = 3
B[0][3] = 4
B[0][4] = 5
B[1][0] = 6
B[1][1] = 7
B[1][2] = 8
B[1][3] = 9
B[1][4] = 10
B[2][0] = 11
B[2][1] = 12
B[2][2] = 13
B[2][3] = 14
B[2][4] = 15
B[3][0] = 16
B[3][1] = 17
B[3][2] = 18
B[3][3] = 19
B[3][4] = 20
B[4][0] = 21
B[4][1] = 22
B[4][2] = 23
B[4][3] = 24
B[4][4] = 25
Введите номер строки для удаления (от 0 до 4): 4
Введите номер столбца для удаления (от 0 до 4): 4
Результирующая матрица C:
    1    2    3    4
    6    7    8    9
   11   12   13   14
   16   17   18   19
```

Задача 2 (программа 9_2)

Задача: даны две целые квадратные матрицы четного порядка. Элементы массивов с четными номерами строки и столбца заменить нулем (стереть). Напечатать исходные и полученные массивы, количество четных и число нечетных чисел в каждом.

Схема алгоритма:





Решение кодом:

```
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define SIZE 4

void process_matrix(int matrix[SIZE][SIZE], int* even_count,
int* odd_count, int* even_count_after, int* odd_count_after) {
    *even_count = 0;
    *odd_count = 0;
    *even_count_after = 0;
    *odd_count_after = 0;

    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (matrix[i][j] % 2 == 0) {
```

```
                (*even_count)++;
            }
            else {
                (*odd_count)++;
            }

            if ((i % 2 == 0) && (j % 2 == 0)) {
                matrix[i][j] = 0;
            }

            if (matrix[i][j] % 2 == 0) {
                (*even_count_after)++;
            }
            else {
                (*odd_count_after)++;
            }
        }
    }
}

void print_matrix(const char* title, int matrix[SIZE][SIZE]) {
    printf("%s\n", title);
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf("%5d", matrix[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

void main9_2() {
    setlocale(LC_ALL, "ru_RU");
    srand(time(NULL));

    int A[SIZE][SIZE], B[SIZE][SIZE];
    int even_A, odd_A, even_A_after, odd_A_after;
    int even_B, odd_B, even_B_after, odd_B_after;

    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            A[i][j] = rand() % 20 - 10;
            B[i][j] = rand() % 20 - 10;
        }
    }

    print_matrix("Исходная матрица A:", A);
    print_matrix("Исходная матрица B:", B);

    process_matrix(A, &even_A, &odd_A, &even_A_after,
&odd_A_after);
    process_matrix(B, &even_B, &odd_B, &even_B_after,
&odd_B_after);

    print_matrix("Преобразованная матрица A:", A);
```

```
    print_matrix("Преобразованная матрица B:", B);

    printf("Матрица A:\n");
    printf("До преобразования: чётных = %d, нечётных = %d\n",
even_A, odd_A);
    printf("После преобразования: чётных = %d, нечётных =
%d\n\n", even_A_after, odd_A_after);

    printf("Матрица B:\n");
    printf("До преобразования: чётных = %d, нечётных = %d\n",
even_B, odd_B);
    printf("После преобразования: чётных = %d, нечётных = %d\n",
even_B_after, odd_B_after);
}
```

Результат работы:

```
Исходная матрица A:
    4   -6    5   -4
    9    1    5   -3
    0   -8   -5    2
    8    4   -9   -6

Исходная матрица B:
  -10  -10    1    4
   -6  -10    3    8
    1   -6   -6    8
    6   -3    3  -10

Преобразованная матрица A:
    0   -6    0   -4
    9    1    5   -3
    0   -8    0    2
    8    4   -9   -6

Преобразованная матрица B:
    0  -10    0    4
   -6  -10    3    8
    0   -6    0    8
    6   -3    3  -10

Матрица A:
До преобразования: чётных = 9, нечётных = 7
После преобразования: чётных = 11, нечётных = 5

Матрица B:
До преобразования: чётных = 11, нечётных = 5
После преобразования: чётных = 13, нечётных = 3
```