МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Факультет автоматизации и информатики

Кафедра Автоматизированные системы управления

ЛАБОРАТОРНАЯ РАБОТА 2

на тему: «Обработка динамических двумерных массивов»

Студент ПМ-23-2 (группа) _____ (подпись, дата) Первушин Е.А. (фамилия, инициалы)

Руководитель

доцент (ученая степень, ученое звание) _____ (подпись, дата) Мирошников А.И. (фамилия, инициалы)

Липецк 2023

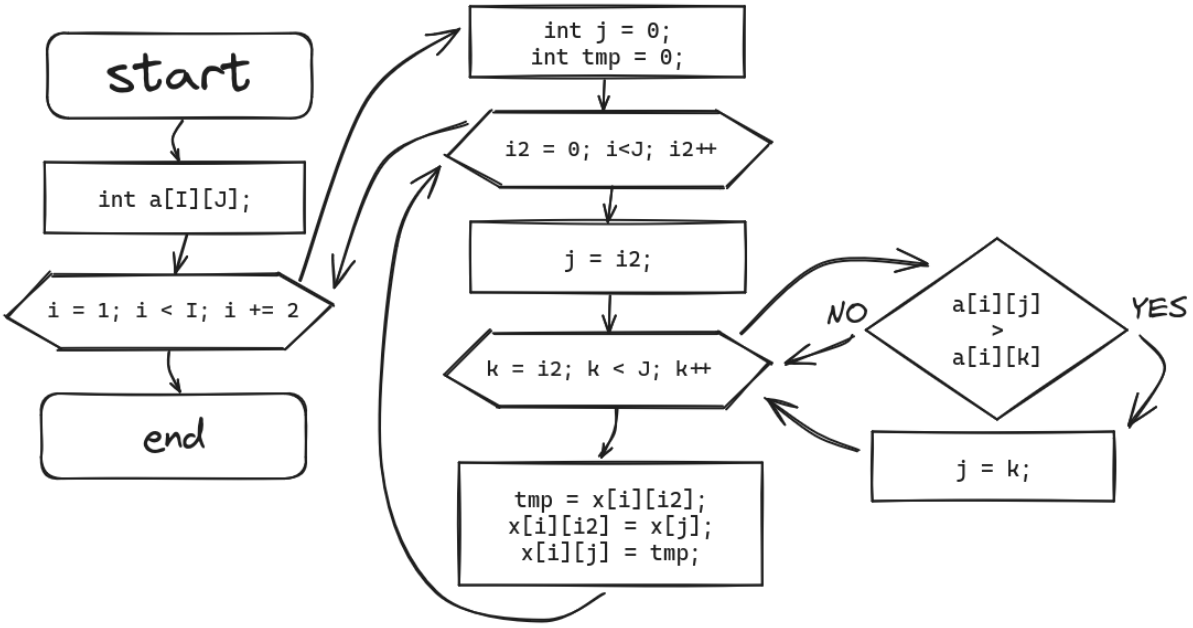
ЦЕЛЬ РАБОТЫ: Освоить работу с динамическими массивами и функциями на языке C

ЗАДАНИЕ: Разработать программу, осуществляющую ввод, вывод и матричные операции над двумерными массивами, в соответствии с вариантом по номеру в журнале. Каждая из операций должна быть реализована в виде отдельной функции. Предусмотреть обработку ошибок и удобный интерфейс. Память под массивы должны выделяться динамически. Предусмотреть возможность заполнения матриц случайными числами в заданном диапазоне.

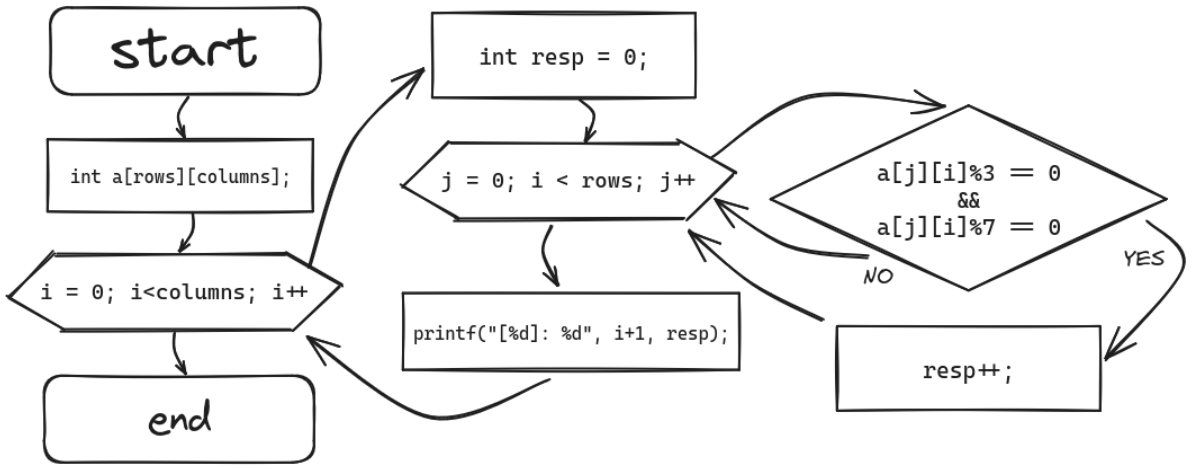
ВАРИАНТ 9

задание 9 Даны две вещественные матрицы произвольной размерности (размерность вводится пользователем). Выполнить следующие операции:

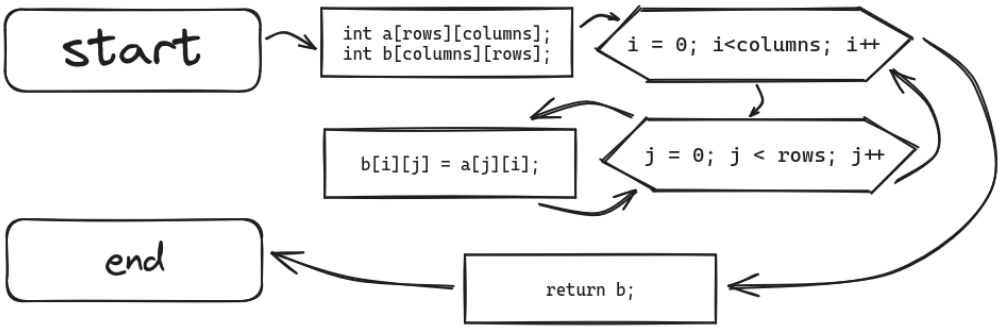
— отсортировать четные строки обеих матриц в порядке возрастания их минимальных элементов



— для каждого столбца подсчитайте и выведите на экран количество элементов, кратных 3 и кратных 7

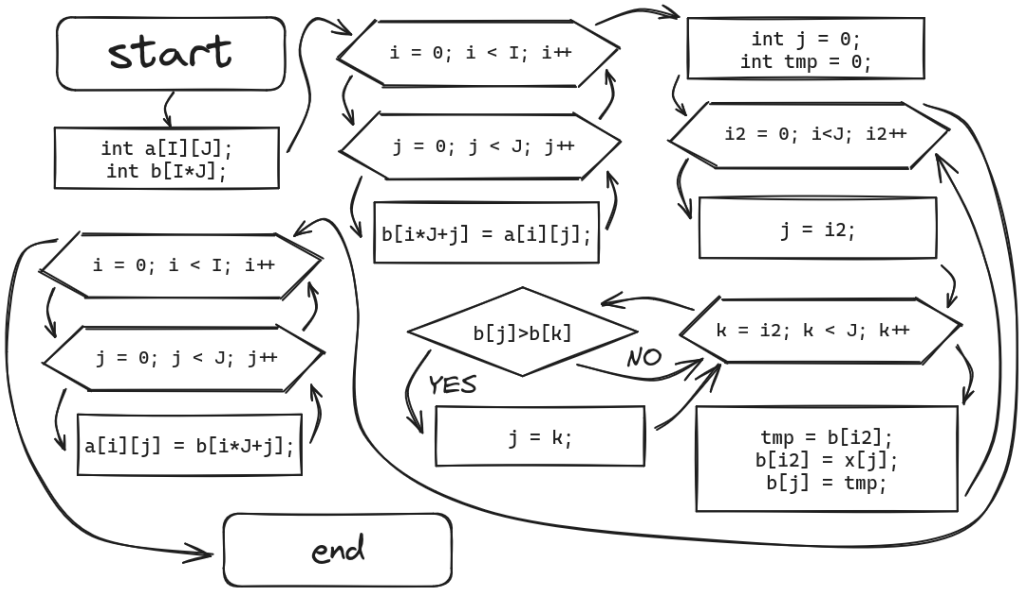


— вычислить две транспонированные матрицы на основе имеющихся

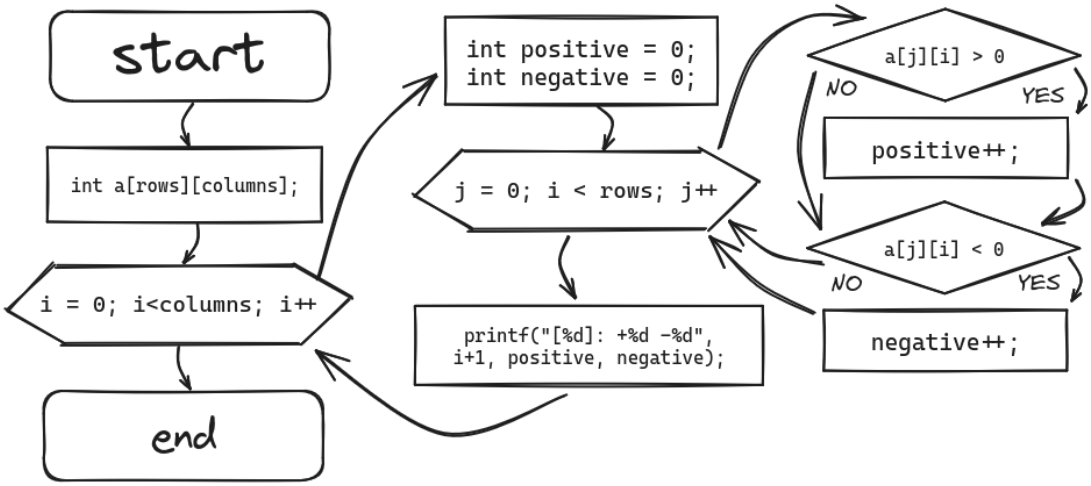


задание 19 Даны три вещественные квадратные матрицы (размер вводится пользователем). Выполнить следующие операции:

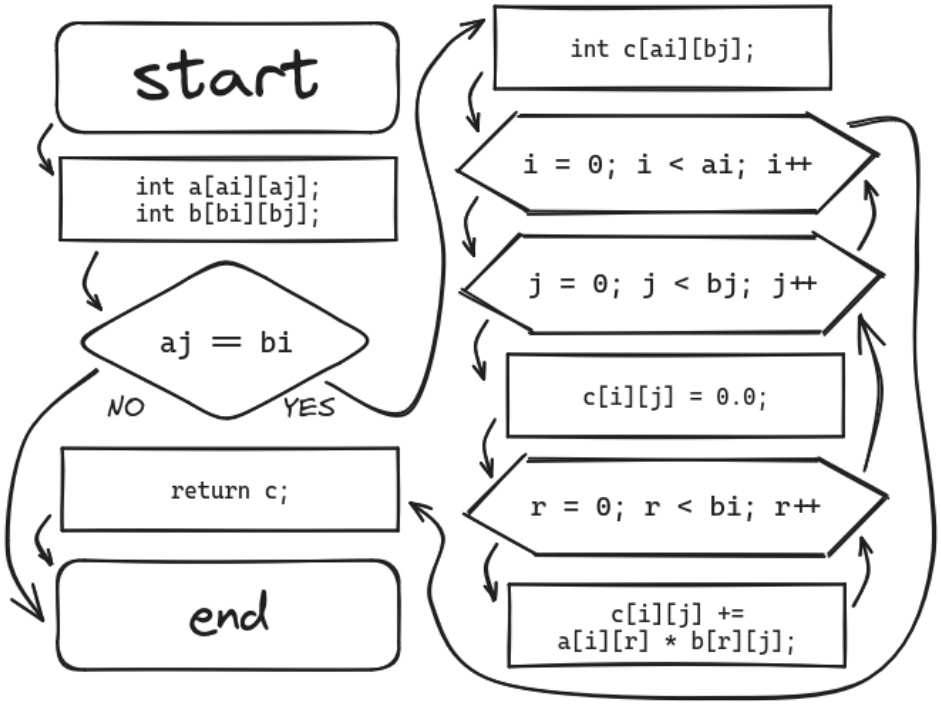
— отсортировать все элементы матриц в порядке возрастания слева направо и сверху вниз



— для каждой из матриц найдите количество положительных и отрицательных элементов в каждом столбце



— вычислить все возможные произведения матриц



ТЕКСТ ПРОГРАММЫ

```
#include <stdio.h>
#include <string>
#include <cmath>
using namespace std;


/* COLORS TEXT IN TERMINAL */
const char WHITE[]   = "\033[38;5;255m";
const char GRAY[]    = "\033[38;5;246m";
const char BLUE[]    = "\033[38;5;27m";
const char RED[]     = "\033[38;5;196m";
const char GREEN[]   = "\033[38;5;47m";
const char PURPLE[]  = "\033[38;5;200m";
const char NOCOLOR[] = "\033[0m";

void clear() { if (system("clear") != 0) { system("cls"); } }


void lidt_sorted(double *x, int size) {
    int j = 0;
    int tmp = 0;
    for (int i = 0; i < size; i++){
        j = i;
        for (int k = i; k < size; k++) {
            if (x[j] > x[k]) { j = k; }
        }
        tmp = x[i];
        x[i] = x[j];
        x[j] = tmp;
    }
}


class Matrix {
public:
    int rows;
    int columns;
    double **value;

    Matrix() {}
```

```
    Matrix(int rows, int columns) : rows(rows), columns(columns) { this->init(); }
    Matrix(int rows, int columns, double** value) : rows(rows), columns(columns),
value(value) {}

    void init() {
        value = new double*[rows];
        for (int i = 0; i < rows; i++) {
            value[i] = new double[columns];
            for (int j = 0; j < columns; j++) { value[i][j] = 0.0; }
        }
    }

    void print() {
        for (int i = 0; i < rows; i++) {
            printf(" %s|%s ", GRAY, NOCOLOR);
            for (int j = 0; j < columns; j++) { printf("%10.2lf ", value[i][j]); }
            printf("%s|%s\n", GRAY, NOCOLOR);
        }
    }

    void input() {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                scanf("%lf", &value[i][j]);
            }
        }
    }

    void input_random() {
        double A, B;

        printf("\n %sминимальное  %sзначение: %s", RED, GREEN, NOCOLOR); scanf("%lf", &A);
        printf(" %sмаксимальное %sзначение: %s", RED, GREEN, NOCOLOR); scanf("%lf", &B);

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                value[i][j] = A + (double) rand() / RAND_MAX * (B - A);
            }
        }
    }

    void sorted_row(int row) {
        lidt_sorted(value[row], columns);
    }

    void sorted() {
        double* temp = new double[rows*columns];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                temp[i*columns + j] = value[i][j];
            }
        }

        lidt_sorted(temp, rows*columns);

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                value[i][j] = temp[i*columns + j];
            }
        }

        delete[] temp;
    }

    void print_counts_columns_multiples(int a, int b = 1) {
        for (int i = 0; i < columns; i++) {
            int resp = 0;
            for (int j = 0; j < rows; j++) {
                if (fmod(value[j][i], a) == 0 && fmod(value[j][i], b) == 0) { resp++; }
            }
            printf(" %s[%d]: %s%d\n", GREEN, i+1, NOCOLOR, resp);
        }
    }
```

```
    void print_counts_columns_positive_and_negative() {
        for (int i = 0; i < columns; i++) {
            int positive = 0;
            int negative = 0;
            for (int j = 0; j < rows; j++) {
                if (value[j][i] > 0) { positive++; }
                else if (value[j][i] < 0) { negative++; }positive
            }
            printf(" %s[%d]: %s+ %s%d %s| %s- %s%d\n", GREEN, i+1, RED, NOCOLOR, positive,
GRAY, RED, NOCOLOR, negative);
        }
    }

    void print_transposed_matrix() {
        for (int i = 0; i < columns; i++) {
            printf(" %s|%s ", GRAY, NOCOLOR);
            for (int j = 0; j < rows; j++) { printf("%10.2lf ", value[j][i]); }
            printf("%s|%s\n", GRAY, NOCOLOR);
        }
    }

    double* operator[](int i) { return this->value[i]; }

    ~Matrix() {
        for (int i = 0; i < rows; i++) { delete[] value[i]; }
        delete[] value;
    }
};


Matrix* product_of_matrices(Matrix *a, Matrix *b) {
    Matrix *resp = new Matrix(a->rows, b->columns);

    for (int i = 0; i < resp->rows; i++) {
        for (int j = 0; j < resp->columns; j++) {
            resp->value[i][j] = 0.0;
            for (int r = 0; r < b->rows; r++) {
                resp->value[i][j] += a->value[i][r] * b->value[r][j];
            }
        }
    }

    return resp;
}



void task_9() {
    clear();
    Matrix *matrix_1 = new Matrix;
    Matrix *matrix_2 = new Matrix;

    try {
        printf("\n %sкол-во строк    %smatrix-1: %s", GREEN, RED, NOCOLOR); scanf("%d",
&matrix_1->rows);
        printf(" %sкол-во столбцов %smatrix-1: %s", GREEN, RED, NOCOLOR); scanf("%d",
&matrix_1->columns);
        matrix_1->init();

        printf("\n %sкол-во строк    %smatrix-2: %s", GREEN, RED, NOCOLOR); scanf("%d",
&matrix_2->rows);
        printf(" %sкол-во столбцов %smatrix-2: %s", GREEN, RED, NOCOLOR); scanf("%d",
&matrix_2->columns);
        matrix_2->init();

        while (true) {
            begin:
            clear();
            printf("\n %smatrix-1:%s\n", RED, NOCOLOR); matrix_1->print();
            printf("\n %smatrix-2:%s\n", RED, NOCOLOR); matrix_2->print();

            printf("\n %s1 %s-> %sввести %smatrix-1%s\n", RED, GRAY, GREEN, RED, NOCOLOR);
            printf(" %s2 %s-> %sввести %smatrix-2%s\n", RED, GRAY, GREEN, RED, NOCOLOR);
```

```
            printf("\n %s3 %s-> %sзаполнить %smatrix-1 %sслучайными числами%s\n", RED, GRAY,
GREEN, RED, GREEN, NOCOLOR);
            printf(" %s4 %s-> %sзаполнить %smatrix-2 %sслучайными числами%s\n", RED, GRAY,
GREEN, RED, GREEN, NOCOLOR);

            printf("\n %s5 %s-> %sотсортировать четные строки в порядке возрастания%s\n",
RED, GRAY, GREEN, NOCOLOR);
            printf(" %s6 %s-> %sкол-во элементов в каждом столбце, кратных 3 и 7%s\n", RED,
GRAY, GREEN, NOCOLOR);
            printf(" %s7 %s-> %sвычислить транспонированные матрицы%s\n", RED, GRAY, GREEN,
NOCOLOR);

            printf("\n %s0 %s-> %sвернуться в главное меню %s\n", RED, GRAY, GREEN, NOCOLOR);

            while (true) {
                int point = -1;
                printf("\n %sвведи номер пункта:%s ", PURPLE, NOCOLOR);
                scanf("%d", &point);

                switch (point) {
                    case 0: { goto end; }
                    case 1: {
                        clear(); printf("\n %sввод %smatrix-1 %s(%dx%d)%s\n\n", GREEN, RED,
GRAY, matrix_1->rows, matrix_1->columns, NOCOLOR);
                        matrix_1->input(); goto begin;
                    } case 2: {
                        clear(); printf("\n %sввод %smatrix-2 %s(%dx%d)%s\n\n", GREEN, RED,
GRAY, matrix_2->rows, matrix_2->columns, NOCOLOR);
                        matrix_2->input(); goto begin;
                    }
                    case 3: { clear(); matrix_1->input_random(); goto begin; }
                    case 4: { clear(); matrix_2->input_random(); goto begin; }
                    case 5: {
                        clear();
                        for (int i = 1; i < matrix_1->rows; i += 2) { matrix_1-
>sorted_row(i); }
                        for (int i = 1; i < matrix_2->rows; i += 2) { matrix_2-
>sorted_row(i); }
                        goto begin;
                    } case 6: {
                        printf("\n %smatrix-1%s\n", RED, NOCOLOR); matrix_1-
>print_counts_columns_multiples(3, 7);
                        printf("\n %smatrix-2%s\n", RED, NOCOLOR); matrix_2-
>print_counts_columns_multiples(3, 7);
                        break;
                    } case 7: {
                        printf("\n %sтранспонированная %smatrix-1:%s\n", GREEN, RED,
NOCOLOR); matrix_1->print_transposed_matrix();
                        printf("\n %sтранспонированная %smatrix-2:%s\n", GREEN, RED,
NOCOLOR); matrix_2->print_transposed_matrix();
                        break;
                    } default: {
                        printf("\n %sОШИБКА: %sданные не подходят\n", RED, NOCOLOR);
                        scanf("%*[^\r\n]");
                        continue;
                    }
                }
            }
            break;
        }
    } catch (...) {}

    end:
    delete matrix_1;
    delete matrix_2;
}



void task_19() {
    clear();
    Matrix *matrix_1 = new Matrix;
    Matrix *matrix_2 = new Matrix;
    Matrix *matrix_3 = new Matrix;
```

```
    try {
        printf("\n %sразмер %smatrix-1: %s", GREEN, RED, NOCOLOR);
        scanf("%d", &matrix_1->columns);
        matrix_1->rows = matrix_1->columns;
        matrix_1->init();

        printf(" %sразмер %smatrix-2: %s", GREEN, RED, NOCOLOR);
        scanf("%d", &matrix_2->columns);
        matrix_2->rows = matrix_2->columns;
        matrix_2->init();

        printf(" %sразмер %smatrix-3: %s", GREEN, RED, NOCOLOR);
        scanf("%d", &matrix_3->columns);
        matrix_3->rows = matrix_3->columns;
        matrix_3->init();

        while (true) {
            begin:
            clear();
            printf("\n %smatrix-1:%s\n", RED, NOCOLOR); matrix_1->print();
            printf("\n %smatrix-2:%s\n", RED, NOCOLOR); matrix_2->print();
            printf("\n %smatrix-3:%s\n", RED, NOCOLOR); matrix_3->print();

            printf("\n %s1 %s-> %sввести %smatrix-1%s\n", RED, GRAY, GREEN, RED, NOCOLOR);
            printf(" %s2 %s-> %sввести %smatrix-2%s\n", RED, GRAY, GREEN, RED, NOCOLOR);
            printf(" %s3 %s-> %sввести %smatrix-3%s\n", RED, GRAY, GREEN, RED, NOCOLOR);

            printf("\n %s4 %s-> %sзаполнить %smatrix-1 %sслучайными числами%s\n", RED, GRAY,
GREEN, RED, GREEN, NOCOLOR);
            printf(" %s5 %s-> %sзаполнить %smatrix-2 %sслучайными числами%s\n", RED, GRAY,
GREEN, RED, GREEN, NOCOLOR);
            printf(" %s6 %s-> %sзаполнить %smatrix-3 %sслучайными числами%s\n", RED, GRAY,
GREEN, RED, GREEN, NOCOLOR);

            printf("\n %s7 %s-> %sотсортировать все элементы матриц%s\n", RED, GRAY, GREEN,
NOCOLOR);
            printf(" %s8 %s-> %sкол-во %s+/- %sэлементов в каждом столбце%s\n", RED, GRAY,
GREEN, RED, GREEN, NOCOLOR);
            printf(" %s9 %s-> %sвсе возможные произведения матриц%s\n", RED, GRAY, GREEN,
NOCOLOR);

            printf("\n %s0 %s-> %sвернуться в главное меню %s\n", RED, GRAY, GREEN, NOCOLOR);

            while (true) {
                int point = -1;
                printf("\n %sвведи номер пункта:%s ", PURPLE, NOCOLOR);
                scanf("%d", &point);

                switch (point) {
                    case 0: { goto end; }
                    case 1: {
                        clear(); printf("\n %sввод %smatrix-1 %s(%dx%d)%s\n\n", GREEN, RED,
GRAY, matrix_1->rows, matrix_1->columns, NOCOLOR);
                        matrix_1->input(); goto begin;
                    } case 2: {
                        clear(); printf("\n %sввод %smatrix-2 %s(%dx%d)%s\n\n", GREEN, RED,
GRAY, matrix_2->rows, matrix_2->columns, NOCOLOR);
                        matrix_2->input(); goto begin;
                    } case 3: {
                        clear(); printf("\n %sввод %smatrix-3 %s(%dx%d)%s\n\n", GREEN, RED,
GRAY, matrix_2->rows, matrix_2->columns, NOCOLOR);
                        matrix_3->input(); goto begin;
                    }
                    case 4: { clear(); matrix_1->input_random(); goto begin; }
                    case 5: { clear(); matrix_2->input_random(); goto begin; }
                    case 6: { clear(); matrix_3->input_random(); goto begin; }
                    case 7: {
                        clear();
                        matrix_1->sorted();
                        matrix_2->sorted();
                        matrix_3->sorted();
                        goto begin;
                    } case 8: {
                        printf("\n %smatrix-1%s\n", RED, NOCOLOR); matrix_1-
>print_counts_columns_positive_and_negative();
```

```
                        printf("\n %smatrix-2%s\n", RED, NOCOLOR); matrix_2-
>print_counts_columns_positive_and_negative();
                        printf("\n %smatrix-3%s\n", RED, NOCOLOR); matrix_3-
>print_counts_columns_positive_and_negative();
                        break;
                    } case 9: {
                        Matrix *temp;

                        temp = product_of_matrices(matrix_1, matrix_1);
                        printf("\n %smatrix-1 * matrix-1%s\n", RED, NOCOLOR); temp->print();
                        delete temp;

                        if (matrix_1->columns == matrix_2->rows) {
                            temp = product_of_matrices(matrix_1, matrix_2);
                            printf("\n %smatrix-1 * matrix-2%s\n", RED, NOCOLOR); temp-
>print();
                            delete temp;
                        }
                        if (matrix_1->columns == matrix_3->rows) {
                            temp = product_of_matrices(matrix_1, matrix_3);
                            printf("\n %smatrix-1 * matrix-3%s\n", RED, NOCOLOR); temp-
>print();
                            delete temp;
                        }
                        if (matrix_2->columns == matrix_1->rows) {
                            temp = product_of_matrices(matrix_2, matrix_1);
                            printf("\n %smatrix-2 * matrix-1%s\n", RED, NOCOLOR); temp-
>print();
                            delete temp;
                        }

                        temp = product_of_matrices(matrix_2, matrix_2);
                        printf("\n %smatrix-2 * matrix-2%s\n", RED, NOCOLOR); temp->print();
                        delete temp;

                        if (matrix_2->columns == matrix_3->rows) {
                            temp = product_of_matrices(matrix_2, matrix_3);
                            printf("\n %smatrix-2 * matrix-3%s\n", RED, NOCOLOR); temp-
>print();
                            delete temp;
                        }
                        if (matrix_3->columns == matrix_1->rows) {
                            temp = product_of_matrices(matrix_3, matrix_1);
                            printf("\n %smatrix-3 * matrix-1%s\n", RED, NOCOLOR); temp-
>print();
                            delete temp;
                        }
                        if (matrix_3->columns == matrix_2->rows) {
                            temp = product_of_matrices(matrix_3, matrix_2);
                            printf("\n %smatrix-3 * matrix-2%s\n", RED, NOCOLOR); temp-
>print();
                            delete temp;
                        }

                        temp = product_of_matrices(matrix_3, matrix_2);
                        printf("\n %smatrix-3 * matrix-3%s\n", RED, NOCOLOR); temp->print();
                        delete temp;

                        break;
                    } default: {
                        printf("\n %sОШИБКА: %sданные не подходят\n", RED, NOCOLOR);
                        scanf("%*[^\r\n]");
                        continue;
                    }
                }
            }
            break;
        }
    } catch (...) {}

    end:
    delete matrix_1;
    delete matrix_2;
}
```

```
int main() {
    srand(time(0));

    while (true) {
        clear();
        printf("\n    %sОБРАБОТКА ДИНАМИЧЕСКИХ ДВУМЕРНЫХ МАССИВОВ%s\n\n", BLUE, NOCOLOR);


        printf(" %s1 %s-> %sоперации с двумя матрицами %s(задание  9)%s\n", RED, GRAY, GREEN,
GRAY, NOCOLOR);
        printf(" %s2 %s-> %sоперации с тремя матрицами %s(задание 19)%s\n", RED, GRAY, GREEN,
GRAY, NOCOLOR);

        printf("\n %s0 %s-> %sзавершить программу %s\n", RED, GRAY, GREEN, NOCOLOR);

        while (true) {
            int point = -1;
            printf("\n %sвведи номер пункта:%s ", PURPLE, NOCOLOR);
            scanf("%d", &point);

            switch (point) {
                case 0: { return 0; }
                case 1: { task_9(); break; }
                case 2: { task_19(); break; }
                default: {
                    printf("\n %sОШИБКА: %sданные не подходят\n", RED, NOCOLOR);
                    scanf("%*[^\r\n]");
                    continue;
                }
            }

            break;
        }
    }
}
```

ПРИМЕР ВЫПОЛНЕНИЯ

```
→ lab-2 git:(main) ✗ g++ -o main main.cpp
→ lab-2 git:(main) ✗ ./main

/* clear */

   ОБРАБОТКА ДИНАМИЧЕСКИХ ДВУМЕРНЫХ МАССИВОВ

1 -> операции с двумя матрицами (задание  9)
2 -> операции с тремя матрицами (задание 19)

0 -> завершить программу

введи номер пункта: 2

/* clear */

размер matrix-1: 1
размер matrix-2: 2
размер matrix-3: 3

/* clear */

matrix-1:
|       0.00 |

matrix-2:
|       0.00       0.00 |
|       0.00       0.00 |

matrix-3:
|       0.00       0.00       0.00 |
|       0.00       0.00       0.00 |
|       0.00       0.00       0.00 |
```

```
1 -> ввести matrix-1
2 -> ввести matrix-2
3 -> ввести matrix-3

4 -> заполнить matrix-1 случайными числами
5 -> заполнить matrix-2 случайными числами
6 -> заполнить matrix-3 случайными числами

7 -> отсортировать все элементы матриц
8 -> кол-во +/- элементов в каждом столбце
9 -> все возможные произведения матриц

0 -> вернуться в главное меню

введи номер пункта: 9

matrix-1 * matrix-1
|       0.00 |

matrix-2 * matrix-2
|       0.00       0.00 |
|       0.00       0.00 |

matrix-3 * matrix-3
|       0.00       0.00 |
|       0.00       0.00 |
|       0.00       0.00 |

введи номер пункта: 0

/* clear */

   ОБРАБОТКА ДИНАМИЧЕСКИХ ДВУМЕРНЫХ МАССИВОВ

1 -> операции с двумя матрицами (задание  9)
2 -> операции с тремя матрицами (задание 19)

0 -> завершить программу

введи номер пункта: 0
→ lab-2 git:(main) ✗
```