МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
"Омский государственный технический университет"

Кафедра «Автоматизированные системы обработки информации и управления»

ПРАКТИЧЕСКАЯ РАБОТА №3

по дисциплине

«Измерительные средства аналитики программных систем и технологий»

по теме

«Метрика Чапина.»

Выполнил: Шмидт А.В.
гр. ИВТ - 244
Проверил: Зубарев А.А.
к.т.н., доцент

Омск 2025

ЗАДАНИЕ

1. Выбрать программу, в которой ОБЯЗАТЕЛЬНО должны быть ВЕТВЛЕНИЯ, или ЦИКЛЫ (или то и другое).
2. Выполнить фрагментацию программы.
3. Провести классификацию «ролей переменных» по методике Чапина в исследуемой программе. Результат классификации представить табличном виде.
4. Провести расчёты значения меры сложности Q исследуемой программы по методу Чапина.

ПРИМЕР ОПРЕДЕЛЕНИЯ МЕТРИК ПО ЧАПИНУ

Программа на C++ генерирует две случайные целочисленные матрицы 7×7 (значения от −50 до 50). Для каждой матрицы выводит полностью матрицу, главную и побочную диагонали, а также их суммы. Дополнительно вычисляет и выводит разность: сумма главной диагонали первой матрицы минус сумма побочной диагонали второй матрицы. Ниже представлен листинг программы.

```
#include <iostream>
#include <iomanip>
#include <random>
#include <string>
using namespace std;

const int N = 7;

// Print main (true) or anti-diagonal (false)
void printDiagonal(const int matrix[N][N], bool isMain)
{
    for (int i = 0; i < N; ++i)
    {
        if (isMain)
            cout << setw(4) << matrix[i][i];
        else
            cout << setw(4) << matrix[i][N - 1 - i];
    }
    cout << "\n";
}

// Calculate sum of main (true) or anti-diagonal (false)
long long sumDiagonal(const int matrix[N][N], bool isMain)
{
    long long sum = 0;
    for (int i = 0; i < N; ++i)
    {
        sum += isMain ? matrix[i][i] : matrix[i][N - 1 - i];
    }
    return sum;
}

// Fill matrix with random integers in range [min, max]
void fillRandom(int matrix[N][N], int minVal = -50, int maxVal = 50)
{
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> dis(minVal, maxVal);

    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            matrix[i][j] = dis(gen);
}
```

```
// Print full matrix
void printMatrix(const int matrix[N][N], const string& name)
{
    cout << "Matrix " << name << ":\n";
    for (int i = 0; i < N; ++i)
    {
        for (int j = 0; j < N; ++j)
            cout << setw(4) << matrix[i][j];
        cout << "\n";
    }
    cout << "\n";
}

// Process and display information for one matrix
void processMatrix(const int matrix[N][N], const string& name)
{
    cout << "Matrix " << name << ":\n";

    cout << "  Main diagonal:    ";
    printDiagonal(matrix, true);

    cout << "  Anti-diagonal:    ";
    printDiagonal(matrix, false);

    long long sumMain = sumDiagonal(matrix, true);
    long long sumAnti = sumDiagonal(matrix, false);

    cout << "  Sum of main diagonal:     " << sumMain << "\n";
    cout << "  Sum of anti-diagonal:     " << sumAnti << "\n\n";
}

int main()
{
    int A[N][N], B[N][N];

    // Seed is different each run thanks to random_device
    fillRandom(A);
    fillRandom(B);

    // Optional: print full matrices
    printMatrix(A, "A");
    printMatrix(B, "B");

    processMatrix(A, "A");
    processMatrix(B, "B");

    long long difference = sumDiagonal(A, true) - sumDiagonal(B, false);
    cout << "Difference (main diagonal of A - anti-diagonal of B) = " << difference << "\n";

    cin.get();
    return 0;
}
```

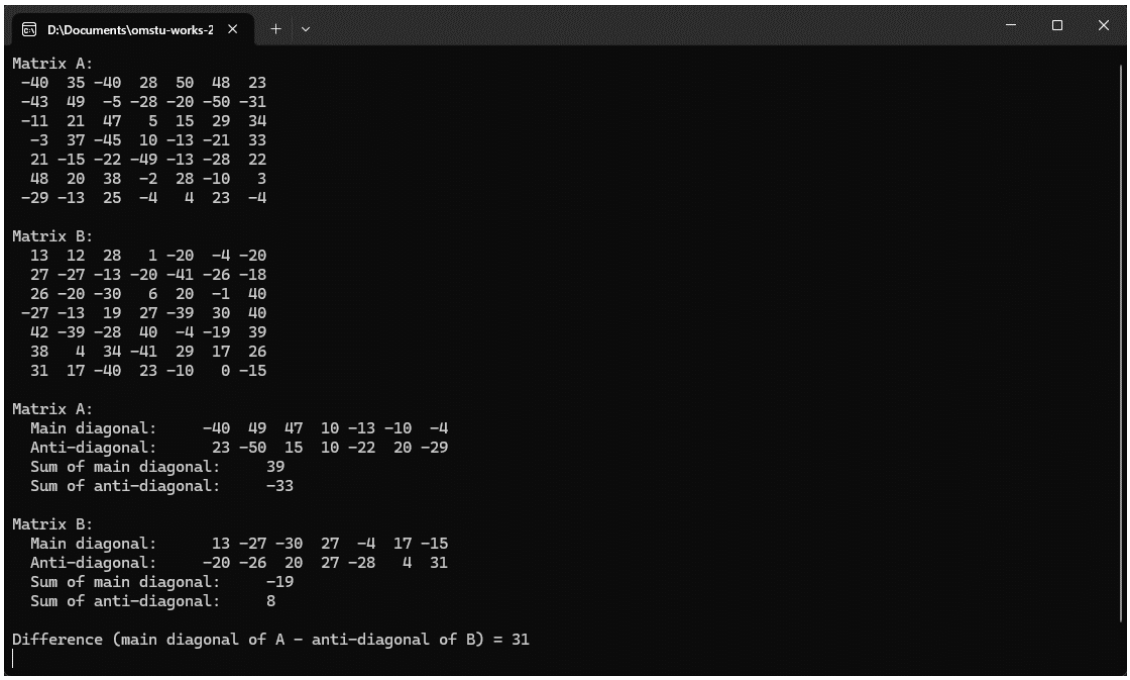

Рисунок 1 – Результат работы программы.

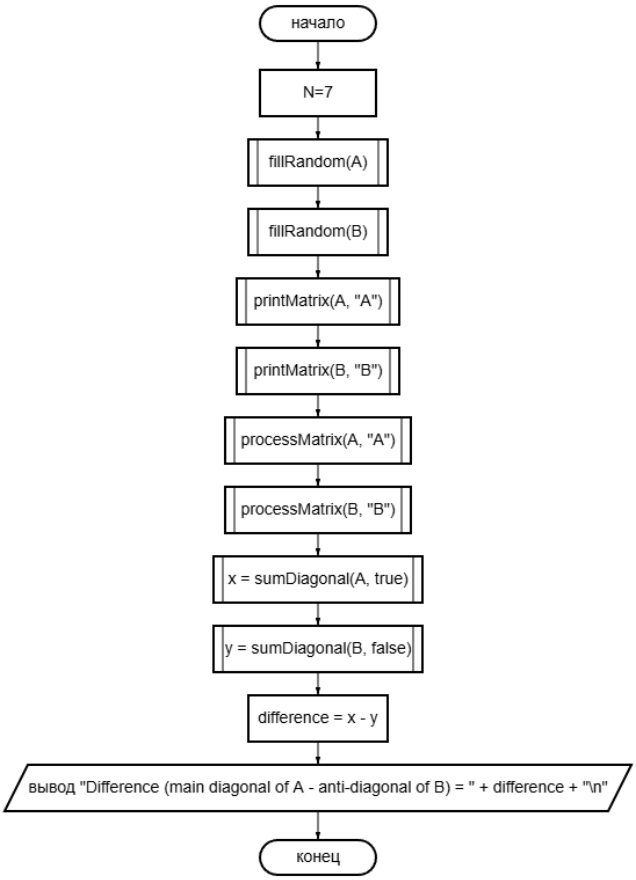

Рисунок 2 – Схема алгоритма главной функции.

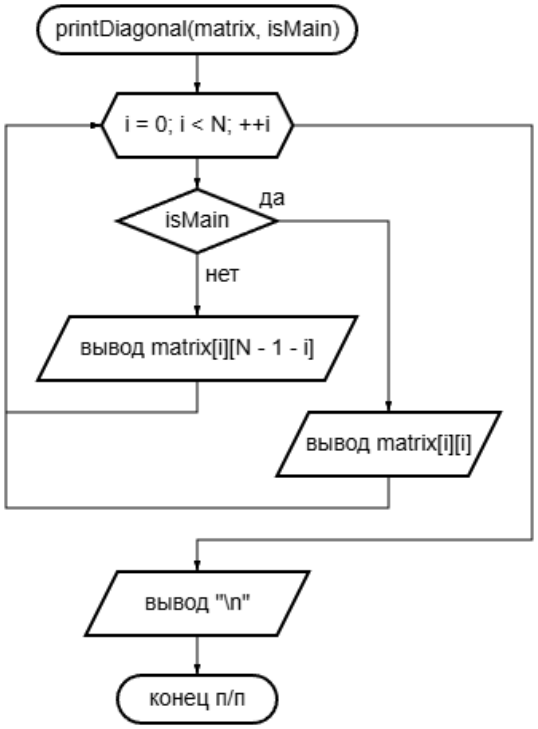

Рисунок 3 – Схема алгоритма подпрограммы для вывода элементов диагоналей матриц.

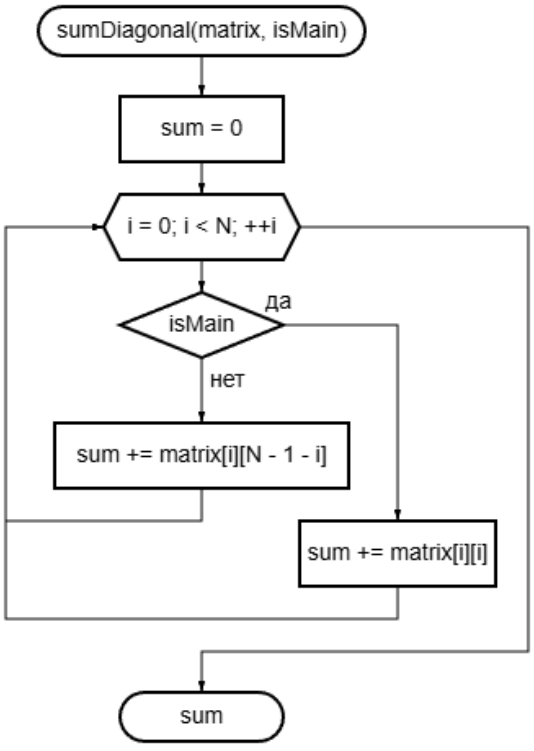

Рисунок 4 – Схема алгоритма подпрограммы, возвращающей сумму элементов диагонали матрицы.

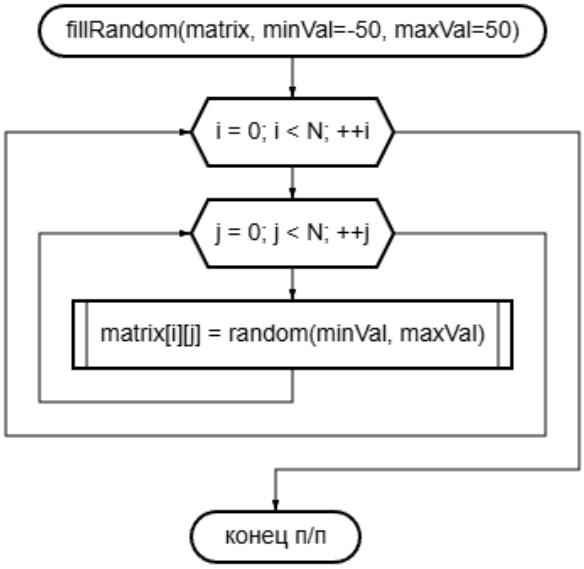

Рисунок 5 – Схема алгоритма подпрограммы для заполнения матрицы случайными числами.

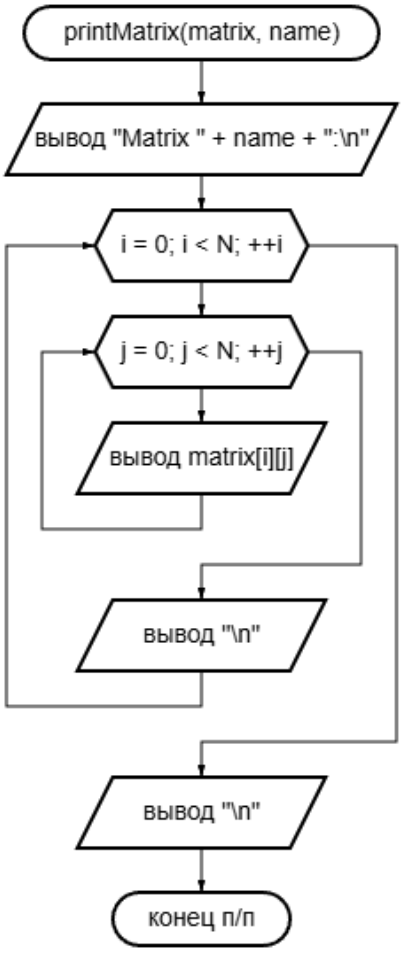

Рисунок 6 – Схема алгоритма подпрограммы для полного вывода матрицы.

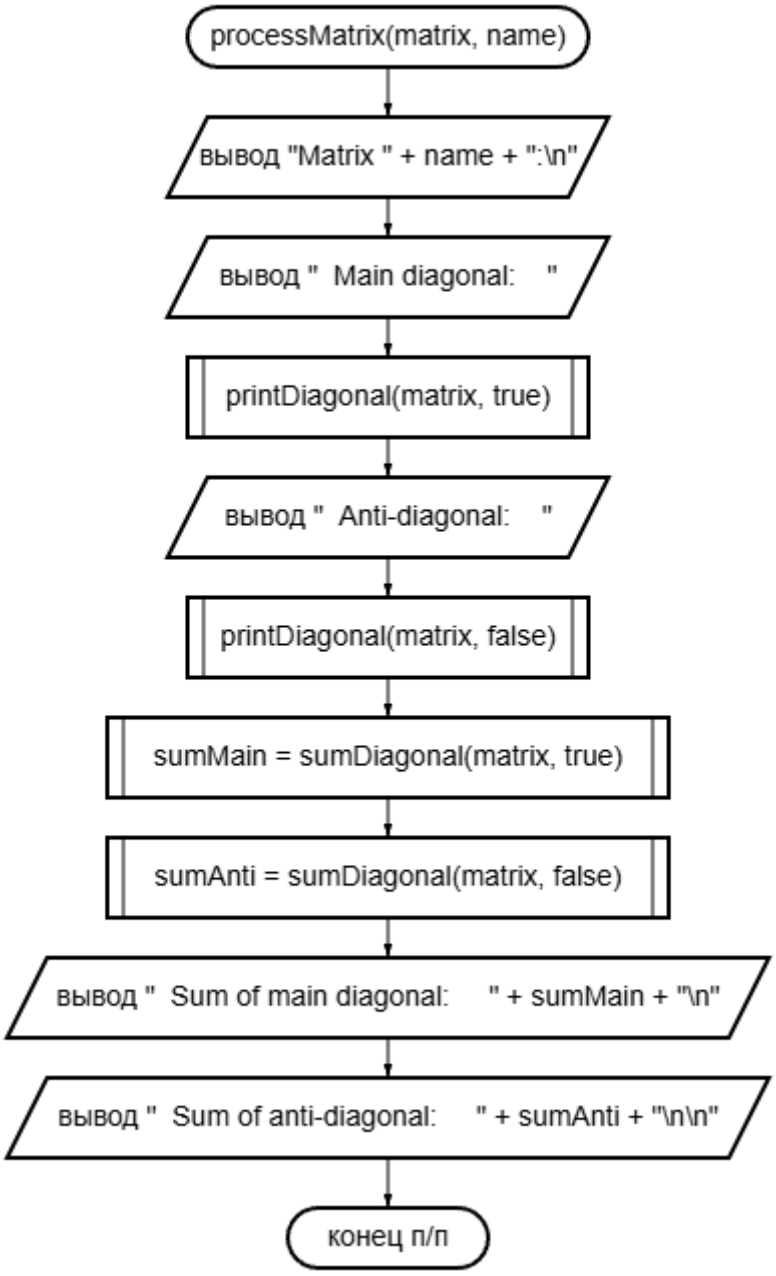

Рисунок 7 – Схема алгоритма подпрограммы для централизованной обработки и вывода матрицы.

Таблица 1 – Классификация ролей переменных.

Модуль / Переменная / Директива	Роль на входе	Роль на выходе	Комментарий (роль и назначение)
Глобальные и заголовки			
#include <iostream>, <iomanip>, <random>, <string>	T	—	Обеспечивают ввод-вывод и генерацию случайных чисел
using namespace std;	T	—	Обеспечивает доступ к std
const int N = 7;	P	—	Константа-размер, используется как входной параметр
fillRandom			
matrix[N][N] (параметр)	M	M	Заполняется случайными значениями → выходные данные
minVal = -50, maxVal = 50	P	—	Параметры по умолчанию, входные данные
rd, gen, dis (локальные)	T	—	Обеспечивающие объекты для генерации
i, j (счётчики циклов)	C	—	Управляют циклами (условие i < N, j < N)
printDiagonal			
matrix[N][N], isMain	P	—	Входные данные (матрица и флаг)
i (счётчик цикла)	C	—	Управляет циклом
sumDiagonal			
matrix[N][N], isMain	P	—	Входные данные
sum (локальная)	M	M	Накапливает и возвращает сумму
i (счётчик цикла)	C	—	Управляет циклом
printMatrix			
matrix[N][N], name	P	—	Входные данные
i, j (счётчики)	C	—	Управляют вложенными циклами
processMatrix			

matrix[N][N], name	P	—	Входные данные

Продолжение таблицы 2 – Классификация ролей переменных.

sumMain, sumAnti (локальные)	M	M	Вычисляются и выводятся
i (счётчик)	C	—	Управляет циклом
main			
A[N][N], B[N][N]	M	M	Заполняются и используются
difference (локальная)	M	M	Вычисляется и выводится
(вызовы fillRandom, printMatrix, processMatrix и т.д.)	—	—	Передача управления

Таблица 3 – Подсчёт ролей для каждого модуля.

№	Модуль	T	P	M	C	Wi = 0.5T + P + 2M + 3C	Ei	Ri = 1 + (Ei/3)²	Qi = √(Ri·Wi)
1	fillRandom	3	2	1	2	0.5·3 + 2 + 2·1 + 3·2 = 11.5	0	1.00	√11.5 ≈ 3.39
2	printDiagonal	0	2	0	1	0 + 2 + 0 + 3·1 = 5	0	1.00	√5 ≈ 2.24
3	sumDiagonal	0	2	1	1	0 + 2 + 2·1 + 3·1 = 7	0	1.00	√7 ≈ 2.65
4	printMatrix	0	2	0	2	0 + 2 + 0 + 3·2 = 8	0	1.00	√8 ≈ 2.83
5	processMatrix	0	2	2	1	0 + 2 + 2·2 + 3·1 = 9	0	1.00	√9 = 3.00
6	main	0	1	3	0	0 + 1 + 2·3 + 0 = 7	0	1.00	√7 ≈ 2.65

В итоге подсчитываем меру сложности программы:

$\Sigma Qi = 3.39 + 2.24 + 2.65 + 2.83 + 3.00 + 2.65 \approx 16.76$

$n = 6$

$Q = \Sigma Qi / n \approx 16.76 / 6 \approx 2.79 \approx 2.8$

ВЫВОД

В ходе выполнения практической работы исследована программа на языке C++, выполненная в императивной парадигме и содержащая циклы, что соответствует требованиям задания.

Программа разбита на 6 логических модулей-функций. Проведена классификация всех переменных и обеспечивающих сущностей по ролям Чапина (P, M, C, T). Вычислена мера сложности потока данных по методу Чапина:

$Q \approx 2,8$

Полученное значение Q = 2,8 характеризует программу как простую, хорошо структурированную и легко понимаемую. Все циклы локальны, условия выхода формируются внутри самих модулей (Ei = 0, Ri = 1 во всех модулях), внешних управляющих переменных C нет. Сложность равномерно распределена по модулям (Qi от 2,24 до 3,39), основной вклад вносят модули с модификацией данных (роль M) и управлением циклами (роль C).

Метрика Чапина подтвердила высокое качество кода: низкое сцепление, высокая связность, хорошая поддерживаемость. Программа имеет рациональную структуру и полностью удовлетворяет критериям структурного программирования.