Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Электротехнический факультет   
Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т  
по лабораторной работе**

**“Отношения между элементами”**

Дисциплина: «Дискретная математика и Математическая логика»

Выполнил   
студент группы ИВТ-23-2б  
Злыгостев Д.Н.

Проверил   
доцент кафедры ИТАС  
Рустамханова Г. И.

Пермь 2024

Оглавление

[1.Цель работы 3](#_Toc181313776)

[2.Задачи 3](#_Toc181313777)

[3.Теоретические сведения 3](#_Toc181313778)

[4.Описание программы 4](#_Toc181313779)

[5.Структура программы 5](#_Toc181313780)

[6.Пример работы программы 5](#_Toc181313781)

[Заключение 5](#_Toc181313782)

# 1.Цель работы

Целью лабораторной работы является разработка программы на языке C++, которая анализирует заданную бинарную матрицу размером 6x6 и выявляет ее основные свойства: рефлексивность, анти рефлексивность, симметричность, антисимметричность, асимметричность, транзитивность, полноту (связность), эквивалентность, а также определяет типы порядков (полный и частичный, строгий и нестрогий).

# 2.Задачи

1. Чтение матрицы из файла.
2. Проверка свойств бинарных отношений (рефлексивность, анти рефлексивность, симметричность, антисимметричность, асимметричность, транзитивность, полноту (связность), эквивалентность.
3. Проверка условий для строгих и нестрогих порядков.
4. Корректная обработка и вывод результатов.

# 3.Теоретические сведения

**Бинарные отношения** — это отношения между элементами двух множеств. На практике отношения часто представляются в виде матрицы, где строки и столбцы соответствуют элементам множества, а элементы матрицы (0 или 1) указывают на наличие или отсутствие отношений между элементами.

Основные свойства бинарных отношений:

1. **Рефлексивность** — если для всех iii элемент aii=1a\_{ii} = 1aii​=1 (каждый элемент связан сам с собой).
2. **Антирефлексивность** — если для всех iii элемент aii=0a\_{ii} = 0aii​=0.
3. **Симметричность** — если для всех iii и jjj, aij=ajia\_{ij} = a\_{ji}aij​=aji​.
4. **Антисимметричность** — если для всех i≠ji \neq ji=j, если aij=1a\_{ij} = 1aij​=1, то aji=0a\_{ji} = 0aji​=0.
5. **Асимметричность** — если для всех iii и jjj, если aij=1a\_{ij} = 1aij​=1, то aji=0a\_{ji} = 0aji​=0, и при этом на главной диагонали только нули.
6. **Транзитивность** — если для любых i,j,ki, j, ki,j,k, из aij=1a\_{ij} = 1aij​=1 и ajk=1a\_{jk} = 1ajk​=1 следует aik=1a\_{ik} = 1aik​=1.
7. **Полнота (связность)** — если для любых iii и jjj хотя бы один из aija\_{ij}aij​ или ajia\_{ji}aji​ равен 1.

Типы порядков:

* **Нестрогий полный порядок** — рефлексивность, антисимметричность, транзитивность, полнота.
* **Строгий полный порядок** — антирефлексивность, антисимметричность, транзитивность, полнота.
* **Нестрогий частичный порядок** — рефлексивность, антисимметричность, транзитивность, отсутствие полноты.
* **Строгий частичный порядок** — антирефлексивность, антисимметричность, транзитивность, отсутствие полноты.

# 4.Описание программы

Программа состоит из нескольких функций, каждая из которых проверяет отдельное свойство бинарного отношения:

* isReflexive() — проверка рефлексивности.
* isAntiReflexive() — проверка антирефлексивности.
* isSymmetric() — проверка симметричности.
* isAntiSymmetric() — проверка антисимметричности с изменением условия согласно задаче.
* isAsymmetric() — проверка асимметричности с изменением условия согласно задаче.
* isTransitive() — проверка транзитивности.
* isComplete() — проверка полноты.
* isEquivalent() — проверка эквивалентности.
* Функции для определения типов порядков.

Программа читает матрицу 6x6 из файла, после чего каждая из функций последовательно проверяет свойства матрицы и выводит результаты на экран.

# 5.Структура программы

1. **Чтение матрицы из файла**. Матрица считывается из файла с именем matrix.txt, который содержит бинарные данные (0 и 1). Файл должен быть в формате 6 строк по 6 элементов в каждой.
2. **Проверка свойств**. Для каждой функции программы вызывается соответствующий метод, который проверяет свойство матрицы и возвращает результат в виде «Да» или «Нет».
3. **Вывод результата**. Результаты проверки выводятся на экран для каждого свойства матрицы.

# 6. Код

#include <iostream>

#include <fstream>

#include <vector>

#include <sstream>

#include <functional>

using namespace std;

// Функции для проверки свойств

bool isReflexive(const vector<vector<int>>& matrix) {

for (int i = 0; i < matrix.size(); i++) {

if (matrix[i][i] != 1) return false;

}

return true;

}

bool isAntiReflexive(const vector<vector<int>>& matrix) {

for (int i = 0; i < matrix.size(); i++) {

if (matrix[i][i] != 0) return false;

}

return true;

}

bool isSymmetric(const vector<vector<int>>& matrix) {

for (int i = 0; i < matrix.size(); i++) {

for (int j = 0; j < matrix.size(); j++) {

if (matrix[i][j] != matrix[j][i]) return false;

}

}

return true;

}

// Изменённая функция антисимметричности

bool isAntiSymmetric(const vector<vector<int>>& matrix) {

bool diagonalHasOne = false; // Проверяем наличие хотя бы одной 1 на главной диагонали

for (int i = 0; i < matrix.size(); i++) {

if (matrix[i][i] == 1) {

diagonalHasOne = true;

}

}

if (!diagonalHasOne) return false;

for (int i = 0; i < matrix.size(); i++) {

for (int j = 0; j < matrix.size(); j++) {

if (i != j && matrix[i][j] == 1 && matrix[j][i] != 0) return false;

}

}

return true;

}

// Изменённая функция ассимметричности

bool isAsymmetric(const vector<vector<int>>& matrix) {

// Проверяем, что на главной диагонали все нули

for (int i = 0; i < matrix.size(); i++) {

if (matrix[i][i] != 0) return false;

}

for (int i = 0; i < matrix.size(); i++) {

for (int j = 0; j < matrix.size(); j++) {

if (matrix[i][j] == 1 && matrix[j][i] != 0) return false;

}

}

return true;

}

// Изменённая функция полноты (связности)

bool isComplete(const vector<vector<int>>& matrix) {

for (int i = 0; i < matrix.size(); i++) {

for (int j = 0; j < matrix.size(); j++) {

if (i != j && matrix[i][j] != 1 && matrix[j][i] != 1) {

return false; // Связь существует, если хотя бы один из элементов (i,j) или (j,i) равен 1

}

}

}

return true;

}

bool isTransitive(const vector<vector<int>>& matrix) {

int n = matrix.size();

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (matrix[i][j] == 1) {

for (int k = 0; k < n; k++) {

if (matrix[j][k] == 1 && matrix[i][k] != 1) return false;

}

}

}

}

return true;

}

bool isEquivalent(const vector<vector<int>>& matrix) {

return isReflexive(matrix) && isSymmetric(matrix) && isTransitive(matrix);

}

// Проверка свойств порядка

bool isStrictTotalOrder(const vector<vector<int>>& matrix) {

return isAntiReflexive(matrix) && isAntiSymmetric(matrix) && isTransitive(matrix) && isComplete(matrix);

}

bool isNonStrictTotalOrder(const vector<vector<int>>& matrix) {

return isReflexive(matrix) && isAntiSymmetric(matrix) && isTransitive(matrix) && isComplete(matrix);

}

bool isStrictPartialOrder(const vector<vector<int>>& matrix) {

return isAntiReflexive(matrix) && isAntiSymmetric(matrix) && isTransitive(matrix) && !isComplete(matrix);

}

bool isNonStrictPartialOrder(const vector<vector<int>>& matrix) {

return isReflexive(matrix) && isAntiSymmetric(matrix) && isTransitive(matrix) && !isComplete(matrix);

}

// Функция для чтения матрицы из файла

vector<vector<int>> readMatrixFromFile(const string& filename) {

ifstream file(filename);

vector<vector<int>> matrix;

if (file.is\_open()) {

string line;

while (getline(file, line)) {

vector<int> row;

for (char& ch : line) {

if (ch == '0' || ch == '1') {

row.push\_back(ch - '0'); // Преобразуем символ в число

}

}

if (!row.empty()) {

matrix.push\_back(row);

}

}

file.close();

}

else {

cout << "Ошибка: не удалось открыть файл!" << endl;

}

// Проверяем, что матрица имеет размер 6x6

if (matrix.size() != 6 || (matrix.size() > 0 && matrix[0].size() != 6)) {

cout << "Ошибка: матрица должна быть размером 6x6." << endl;

matrix.clear(); // Очищаем матрицу, если она неправильного размера

}

return matrix;

}

int main() {

setlocale(LC\_ALL, "Russian");

// Чтение матрицы из файла

string filename = "matrix.txt"; // Имя файла с матрицей

vector<vector<int>> matrix = readMatrixFromFile(filename);

if (matrix.empty()) {

cout << "Матрица не была прочитана из файла или имеет неверный размер." << endl;

return 1; // Завершаем программу, если матрица не загружена

}

// Вывод матрицы

cout << "Матрица 6x6:" << endl;

for (int i = 0; i < matrix.size(); i++) {

for (int j = 0; j < matrix[i].size(); j++) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

// Проверка свойств

cout << "\nПроверка свойств матрицы:" << endl;

cout << "Рефлексивность: " << (isReflexive(matrix) ? "Да" : "Нет") << endl;

cout << "Антирефлексивность: " << (isAntiReflexive(matrix) ? "Да" : "Нет") << endl;

cout << "Симметричность: " << (isSymmetric(matrix) ? "Да" : "Нет") << endl;

cout << "Антисимметричность: " << (isAntiSymmetric(matrix) ? "Да" : "Нет") << endl;

cout << "Ассимметричность: " << (isAsymmetric(matrix) ? "Да" : "Нет") << endl;

cout << "Транзитивность: " << (isTransitive(matrix) ? "Да" : "Нет") << endl;

cout << "Полнота (связность): " << (isComplete(matrix) ? "Да" : "Нет") << endl;

cout << "Эквивалентность: " << (isEquivalent(matrix) ? "Да" : "Нет") << endl;

// Проверка порядков

cout << "\nПроверка порядков:" << endl;

cout << "Нестрогий полный порядок: " << (isNonStrictTotalOrder(matrix) ? "Да" : "Нет") << endl;

cout << "Строгий полный порядок: " << (isStrictTotalOrder(matrix) ? "Да" : "Нет") << endl;

cout << "Нестрогий частичный порядок: " << (isNonStrictPartialOrder(matrix) ? "Да" : "Нет") << endl;

cout << "Строгий частичный порядок: " << (isStrictPartialOrder(matrix) ? "Да" : "Нет") << endl;

return 0;

}

# 7.Пример работы программы

Пример файла matrix.txt (рис.1):

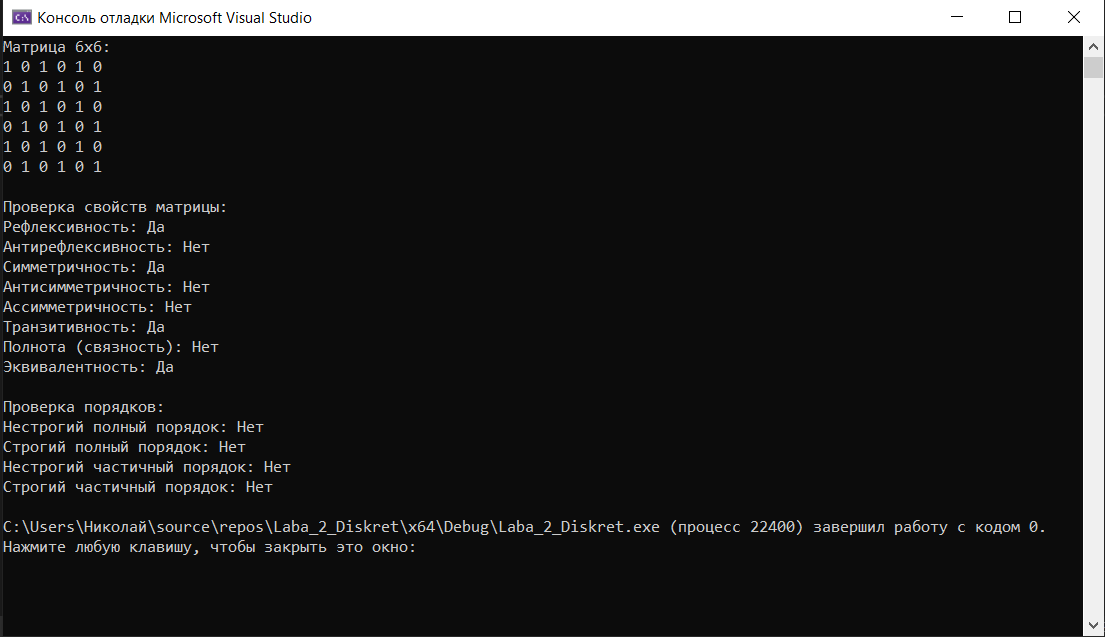


Рисунок.1 - Вывод

# Заключение

Разработанная программа успешно решает поставленную задачу: она корректно считывает бинарную матрицу, проверяет её на наличие свойств и выводит результаты. Данный код позволяет более точно анализировать матрицу и выявлять её свойства исключая ошибки.