Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Дискретная математика и математическая логика»

Тема: «Свойства матриц»

Семестр 2

Выполнил работу

Студент группы ИВТ-23-2б

Меновщиков Глеб Николаевич

Проверил

старший преподаватель кафедры ИТАС

Рустамханова Гульшат Ильдаровна

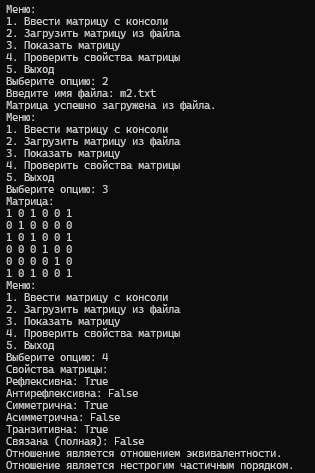
**Цель программы**

Цель данной программы — предоставить пользователю инструменты для работы с матрицами размером 6x6, состоящими из элементов 0 и 1. Программа позволяет вводить матрицу с консоли, загружать её из файла, отображать содержимое матрицы и проверять различные свойства матрицы, такие как рефлексивность, симметричность, транзитивность и другие. На основе этих свойств программа определяет, является ли заданное отношение отношением эквивалентности или порядком.

**Описание работы программы**

1. **Меню пользователя**: Программа запускается с текстового меню, где пользователь может выбрать одну из следующих опций:
   * Ввод матрицы с консоли.
   * Загрузка матрицы из файла.
   * Отображение текущей матрицы.
   * Проверка свойств матрицы.
   * Выход из программы.
2. **Ввод матрицы с консоли**: При выборе этой опции программа запрашивает у пользователя ввод элементов матрицы поэлементно. Пользователь должен вводить только 0 или 1.
3. **Загрузка матрицы из файла**: Пользователь может загрузить матрицу из текстового файла. Программа ожидает, что файл содержит 6 строк, каждая из которых содержит 6 чисел, разделённых пробелами. Если файл не может быть прочитан, программа выводит сообщение об ошибке.
4. **Отображение матрицы**: Программа выводит текущую матрицу в консоль в удобочитаемом формате.
5. **Проверка свойств матрицы**: Программа анализирует матрицу и проверяет следующие свойства:
   * Рефлексивность: все элементы на главной диагонали равны 1.
   * Антирефлексивность: все элементы на главной диагонали равны 0.
   * Симметричность: для всех пар (i, j) элементы matrix[i, j] и matrix[j, i] равны.
   * Асимметричность: если matrix[i, j] равно 1, то matrix[j, i] должно быть 0.
   * Транзитивность: если matrix[i, j] и matrix[j, k] равны 1, то matrix[i, k] также должно быть 1.
   * Связность: для любых двух различных элементов i и j, хотя бы один из элементов matrix[i, j] или matrix[j, i] должен быть равен 1.
6. **Определение типа отношения**: На основе проверенных свойств программа определяет, является ли отношение:
   * Отношением эквивалентности (рефлексивное, симметричное и транзитивное).
   * Нестрогим полным порядком (рефлексивное, антисимметричное, транзитивное и связное).
   * Строгим полным порядком (антисимметричное, транзитивное и связное, но не рефлексивное).
   * Нестрогим частичным порядком (рефлексивное, антисимметричное и транзитивное, но не связное).
   * Строгим частичным порядком (антисимметричное, транзитивное и не связное, но не рефлексивное).
   * Не соответствует ни одному из известных типов порядка.
7. **Выход из программы**: Пользователь может выйти из программы, выбрав соответствующую опцию в меню.

**Скриншоты работы:**



**Программа:**

<https://github.com/Gpushak/CsharpUnProj/tree/7d357b4e79d29a5e293faec5bbd90852e5c73cee/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%B0%D1%8F%20%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/MatrDiskMat>

using System;

using System.IO;

class Program

{

const int size = 6;

static int[,] matrix = new int[size, size];

static void Main()

{

while (true)

{

Console.WriteLine("Меню:");

Console.WriteLine("1. Ввести матрицу с консоли");

Console.WriteLine("2. Загрузить матрицу из файла");

Console.WriteLine("3. Показать матрицу");

Console.WriteLine("4. Проверить свойства матрицы");

Console.WriteLine("5. Выход");

Console.Write("Выберите опцию: ");

int choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

InputMatrixFromConsole();

break;

case 2:

LoadMatrixFromFile();

break;

case 3:

DisplayMatrix();

break;

case 4:

CheckMatrixProperties();

break;

case 5:

return;

default:

Console.WriteLine("Неверный выбор, попробуйте снова.");

break;

}

}

}

static void InputMatrixFromConsole()

{

Console.WriteLine("Введите элементы матрицы 6x6 (только 0 или 1):");

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

{

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

{

Console.Write($"Элемент [{i},{j}]: ");

matrix[i, j] = int.Parse(Console.ReadLine());

}

}

}

static void LoadMatrixFromFile()

{

Console.Write("Введите имя файла: ");

string fileName = Console.ReadLine();

try

{

using (StreamReader sr = new StreamReader(fileName))

{

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

{

string[] row = sr.ReadLine().Split();

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

{

matrix[i, j] = int.Parse(row[j]);

}

}

}

Console.WriteLine("Матрица успешно загружена из файла.");

}

catch (Exception e)

{

Console.WriteLine("Ошибка чтения файла: " + e.Message);

}

}

static void DisplayMatrix()

{

Console.WriteLine("Матрица:");

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

{

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

{

Console.Write(matrix[i, j] + " ");

}

Console.WriteLine();

}

}

static void CheckEquivalence(bool reflexive, bool symmetric, bool transitive)

{

if (reflexive && symmetric && transitive)

{

Console.WriteLine("Отношение является отношением эквивалентности.");

}

else

{

Console.WriteLine("Отношение не является отношением эквивалентности.");

}

}

static void CheckOrder(bool reflexive, bool antiSymmetric, bool transitive, bool connected)

{

if (reflexive && antiSymmetric && transitive && connected)

{

Console.WriteLine("Отношение является нестрогим полным порядком.");

}

else if (!reflexive && antiSymmetric && transitive && connected)

{

Console.WriteLine("Отношение является строгим полным порядком.");

}

else if (reflexive && antiSymmetric && transitive && !connected)

{

Console.WriteLine("Отношение является нестрогим частичным порядком.");

}

else if (!reflexive && antiSymmetric && transitive && !connected)

{

Console.WriteLine("Отношение является строгим частичным порядком.");

}

else

{

Console.WriteLine("Отношение не соответствует ни одному из известных типов порядка.");

}

}

static void CheckMatrixProperties()

{

bool reflexive = true, antiReflexive = true;

bool symmetric = true, antiSymmetric = true, asymmetric = true;

bool transitive = true, connected = true;

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

{

if (matrix[i, i] != 1)

reflexive = false;

if (matrix[i, i] != 0)

antiReflexive = false;

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

{

if (i != j)

{

if (matrix[i, j] != matrix[j, i])

symmetric = false;

if (matrix[i, j] == 1 && matrix[j, i] == 1)

asymmetric = false;

if (matrix[i, j] == 0 && matrix[j, i] == 0)

connected = false;

}

for (int k = 0; k < size; k++)

{

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

transitive = false;

}

}

}

Console.WriteLine("Свойства матрицы:");

Console.WriteLine($"Рефлексивна: {reflexive}");

Console.WriteLine($"Антирефлексивна: {antiReflexive}");

Console.WriteLine($"Симметрична: {symmetric}");

Console.WriteLine($"Асимметрична: {asymmetric}");

Console.WriteLine($"Транзитивна: {transitive}");

Console.WriteLine($"Связана (полная): {connected}");

CheckEquivalence(reflexive, symmetric, transitive);

CheckOrder(reflexive, antiSymmetric, transitive, connected);

}

}