Министерство образования и науки Российской Федерации Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌ высшего‌ ‌образования‌

**«Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №2**

Дисциплина: «Дискретная математика и математическая логика»

Тема: Определение свойств отношений

Выполнила работу

студентка группы ИВТ-22-2б

Ищенко Д. О.

Проверила

Старший преподаватель кафедры ИТАС

Рустамкханова Г. И.

Пермь, 2023

**Постановка задачи**

Необходимо создать «Определение свойств отношений», который имеет следующие возможности:

1. Считывание тестовой матрицы отношения из файла
2. Определение следующих свойств данного отношения:
   1. Рефлексивность
   2. Нерефлексивность
   3. Симметричность
   4. Асимметричность
   5. Антисимметричность
   6. Транзитивность
   7. Связность

**Анализ задачи**

Для реализации отношения создан класс Relation, имеющий следующие характеристики:

1. Поля:
   1. int[,] matrix : матрица отношения двух множеств, где «1» - это наличие отношения, «0» - отсутствие отношения
   2. string properties : текстовое сообщение, содержащее все характеристики отношения.
2. Методы:
   1. Relation() : конструктор объекта класса, который автоматически задаёт отношению свойства
   2. DetermineProperties() : заполняет properties соответствующими свойствам сообщениями
   3. IsReflexive() : определяет рефлексивность отношения
   4. IsAntiReflexive() : определяет антирефлексивность отношения
   5. IsSymmetric() : определяет симметричность отношения
   6. IsAntiSymmetric() : определяет антисимметричность отношения
   7. IsAsymmetric() : определяет асимметричность отношения
   8. IsTransitive() : определяет транзитивность отношения
   9. IsConnected() : определяет связность отношения
3. Вспомогательные внешние функции:
   1. InputIntNumber() : для корректного считывания целых чисел с консоли
   2. ReadMatrix() : для считывания матрицы отношения из тестовых файлов
   3. Main() : основная функция программы

Свойства отношения определяются путём обхода по матрице отношений, где «1» обозначает наличие отношения, а «0» - отсутствие отношения.

Рефлексивность определяется обходом по главной диагонали матрицы. Если она является единичной, то отношение рефлексивно. Если же главная диагональ содержит только нули, то отношение антирефлексивно. Если на диагонали содержатся и нули, и единицы, то отношение свойствами рефлексивности не обладает. Определены функции IsReflexive(), IsAntiReflexive().

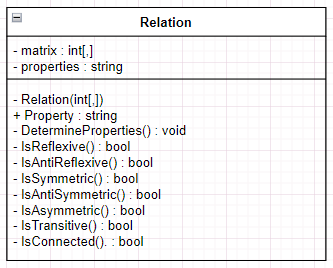
Отношения симметричности определяются путём сравнения значений, симметричных относительно главной диагонали. Если , то отношение симметрично. Если равенство не выполняется, то происходит проверка главной диагонали. Если отношение несимметрично и рефлексивно, то оно асимметрично; если отношение несимметрично и нерефлексивно, то оно антисимметрично. Иначе отношение не обладает никакими свойствами симметричности. Определены функции IsSymmetric(), IsAntiSymmetric(), IsAsymmetric().

Транзитивность определяется путём обхода всех единичных значений в строке. Отношение транзитивно, если *.* Если эти равенства не выполняются, то отношение антитранзитивно. Определена функция IsTransitive().

Связность определяется наличием хотя бы одной единицы для *,* не включая главную диагональ. Если же хотя бы в одной паре отсутствует единица, то отношение несвязно. Определена функция IsConnected().

Проверки на корректность ввода и корректность данных реализованы с помощью вспомогательных функций (например, функция InputIntNumber() проверяет, правильно ли введено целочисленное число).

**UML-диаграмма класса**



**Тесты**

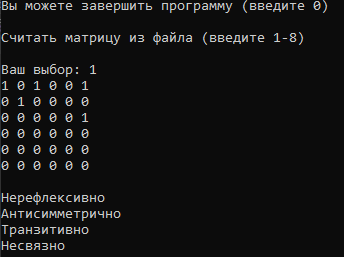
**\**

Рисунок 1 - Демонстрация работы программы на 1 тестовом файле

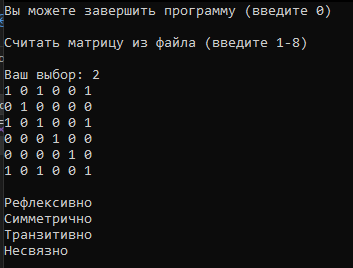


Рисунок 2 - Демонстрация работы программы на 2 тестовом файле

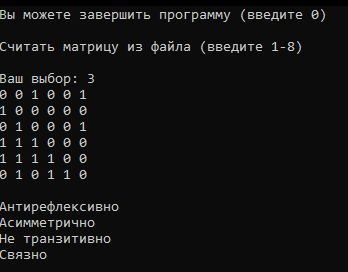


Рисунок 3 - Демонстрация работы программы на 3 тестовом файле

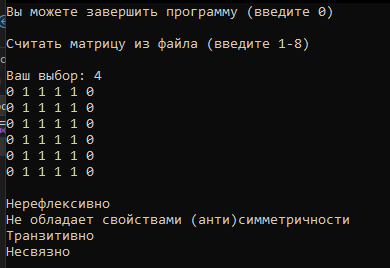


Рисунок 4 - Демонстрация работы программы на 4 тестовом файле

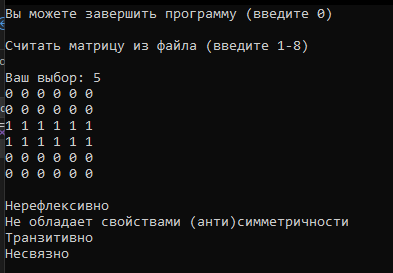


Рисунок 5 - Демонстрация работы программы на 5 тестовом файле

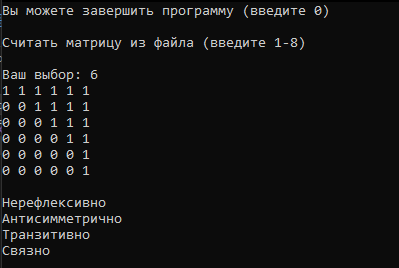


Рисунок 6 - Демонстрация работы программы на 6 тестовом файле

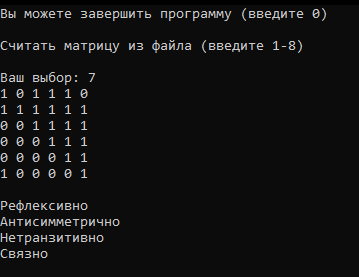


Рисунок 7 - Демонстрация работы программы на 7 тестовом файле

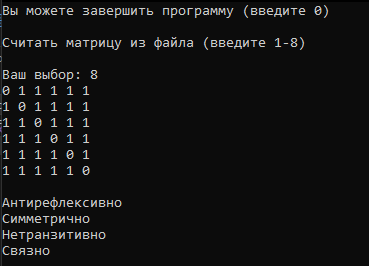


Рисунок 8 - Демонстрация работы программы на 8 тестовом файле

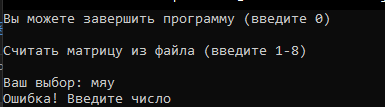


Рисунок 9 - Введение пользователем строки вместо целого числа

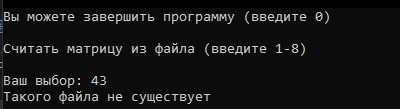


Рисунок 10 - Введение пользователем непредусмотренного целого числа

**Код программы**

Язык программирования С#

namespace lab6

{

internal class Program

{

static void Main(string[] args)

{

while (true)

{

Console.Clear();

int[,] matrix = { };

int choice = InputIntNumber("Вы можете завершить программу (введите 0)\n\n" +

"Считать матрицу из файла (введите 1-8)\n\n" +

"Ваш выбор: ");

switch (choice)

{

case 0:

{

Environment.Exit(0);

break;

}

case 1:

{

matrix = ReadMatrix("../../../files/m1.txt");

break;

}

case 2:

{

matrix = ReadMatrix("../../../files/m2.txt");

break;

}

case 3:

{

matrix = ReadMatrix("../../../files/m3.txt");

break;

}

case 4:

{

matrix = ReadMatrix("../../../files/m4.txt");

break;

}

case 5:

{

matrix = ReadMatrix("../../../files/m5.txt");

break;

}

case 6:

{

matrix = ReadMatrix("../../../files/m6.txt");

break;

}

case 7:

{

matrix = ReadMatrix("../../../files/m7.txt");

break;

}

case 8:

{

matrix = ReadMatrix("../../../files/m8.txt");

break;

}

default:

{

Console.WriteLine("Такого файла не существует");

break;

}

}

if (choice < 9 && choice > 0)

{

Relation set = new Relation(matrix);

Console.WriteLine(set.Property);

}

Console.ReadKey(true);

}

}

static int[,] ReadMatrix(string path)

{

string[] st;

string[] row = File.ReadAllLines(path);

int[,] matrix = new int[6, 6];

for (int i = 0; i < 6; ++i)

{

Console.WriteLine(row[i]);

st = row[i].Split(" ");

for (int j = 0; j < 6; ++j)

{

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

}

}

return matrix;

}

class Relation

{

private int[,] matrix;

private string properties;

public Relation(int[,] mat)

{

matrix = mat;

properties = "";

DetermineProperties();

}

public string Property

{

get => properties;

}

private void DetermineProperties()

{

if (IsReflexive())

properties = String.Concat(properties, "\nРефлексивно");

else if (IsAntiReflexive())

properties = String.Concat(properties, "\nАнтирефлексивно");

else

properties = String.Concat(properties, "\nНе рефлексивно и не антирефлексивно");

if (IsSymmetric())

properties = String.Concat(properties, "\nСимметрично");

else if (IsAntiSymmetric())

{

if (IsAsymmetric())

properties = String.Concat(properties, "\nАсимметрично");

else

properties = String.Concat(properties, "\nАнтисимметрично");

}

else

properties = String.Concat(properties, "\nНе обладает свойствами (анти)симметричности");

if (IsTransitive())

properties = String.Concat(properties, "\nТранзитивно");

else

properties = String.Concat(properties, "\nНе транзитивно");

if (IsConnected())

properties = String.Concat(properties, "\nСвязно");

else

properties = String.Concat(properties, "\nНесвязно");

}

private bool IsReflexive()

{

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

{

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

return false;

}

return true;

}

private bool IsAntiReflexive()

{

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

{

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

return false;

}

return true;

}

private bool IsSymmetric()

{

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

{

for (int j = 0; j < matrix.GetLength(1); ++j)

{

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

return false;

}

}

return true;

}

private bool IsAntiSymmetric()

{

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

{

for (int j = 0; j < matrix.GetLength(1); ++j)

{

if ((matrix[i, j] == 1) && (matrix[j, i] != 0) && (i != j))

return false;

}

}

return true;

}

private bool IsAsymmetric()

{

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

{

for (int j = 0; j < matrix.GetLength(1); ++j)

{

if (i == j)

{

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

return false;

}

else if ((matrix[i, j] == 1) && (matrix[j, i] != 0) && (i != j))

return false;

}

}

return true;

}

private bool IsTransitive()

{

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

{

for (int j = 0; j < matrix.GetLength(1); ++j)

{

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

{

for (int k = 0; k < matrix.GetLength(1); ++k)

{

if ((matrix[i, k] == 0) && (matrix[j, k] == 1))

return false;

}

}

}

}

return true;

}

private bool IsConnected()

{

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

{

for (int j = 0; j < matrix.GetLength(1); ++j)

{

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

return false;

}

}

return true;

}

}

static int InputIntNumber(string msg = "")

{

bool isNumber;

string inputLine;

int res;

Console.Write(msg);

do

{

inputLine = Console.ReadLine();

isNumber = int.TryParse(inputLine, out res);

if (!isNumber)

Console.Write("Ошибка! Введите число\n" + msg);

} while (!isNumber);

return res;

}

}