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 №1**

Дисциплина: «Дискретная математика и математическая

логика»

Тема: Калькулятор множеств

Выполнил работу

студент группы ИВТ-22-2б

Казанцев А.В.

Проверила

Старший преподаватель кафедры ИТАС

Рустамханова Г.И.

Пермь 2023

**Цель работы**

1. Создание консольного приложения «Анализатор матриц отношений»

2. Изучение теории дискретной математики об свойствах отношений

3. Изучение и отработка навыков определения свойств матриц отношений

**Постановка задачи**

1. Программа должен иметь возможность задавать матрицу отношений из файла
2. Программа должна определять свойства отношении:
   1. Рефлективность
   2. Симметричность
   3. Транзитивность
   4. Эквивалентность
   5. Связность(полнота)
   6. Порядок
3. Выводить экран полученную матрицу NxN и полученные свойства

**Анализ задачи**

1. Определение типа заданных выражений.

|  |  |  |  |
| --- | --- | --- | --- |
| Пункт | Входные данные | Классы входных данных | Выходные данные |
| 1 | Файл содержащий N строк содержащих N значений через пробел. | N – целое число  N>0 | Матрица NxN |
| Количество файлов – целое число | matrixCount>=1 | Проверка нескольких матриц последовательно |
| 2 | Файл содержит не равное количество строк и значений | N!=M | Ошибка |
| elementCount- целое число | elementCount<0 | Ошибка |

**Ход работы**

Программа имеет возможность открыть файл с подходящий имеем.  
программа считывает файл в массив строк. Из массива строк считываются значения в массив целых значений NxN. Файл закрывается.

После проходит проверка свойств матрицы отношений. Ключевыми являются рефлективность симметричность транзитивность и связность. Свойства порядка и эквивалентности основываются на некоторых свойствах, указанных выше.  
Методы, используемые в программе:

ReadMatrix(string path) – метод открывает и считывает файл в массив строк. Из массива строк данные извлекаются в матрицу отношений matrix.

Matrix – имеет модификатор и не требуется вызывать ее в каждом методе. Все действия в последствии выполняются над этой матрицей.

WriteMatrix() – метод отображает матрицу в консоль.

CheckReflexivity()- возвращает строку Рефлексивное/Не рефлексивное/Частично рефлексивное. Программа проходит по главное диагонали квадратной матрицы и проверяет если все ли значения равны 0 или 1.

CheckSymmetry()- возвращает строку Симметричное/ Антисимметричное/ Асимметричное. Программа проверяет значения выше главной диагонали и проверяет его через умножение на обратное значение. Если выполняется условие что при умножении на обратное значения значение не меняется, то эта пара значении подходит под условие симметричности.

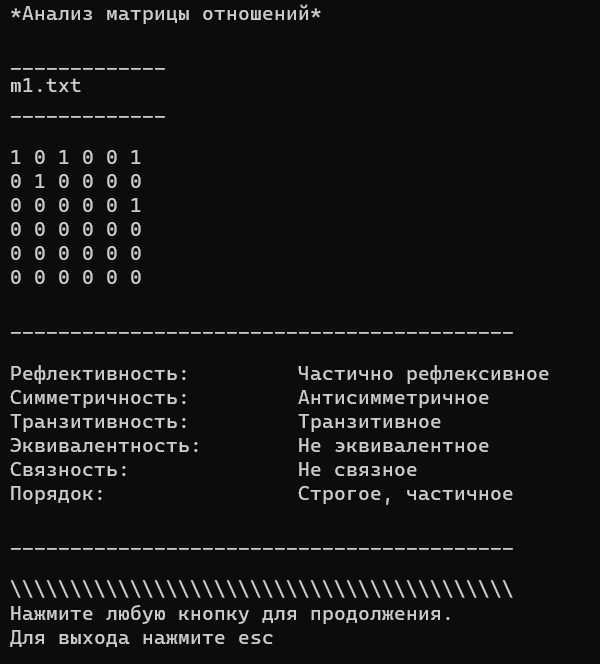
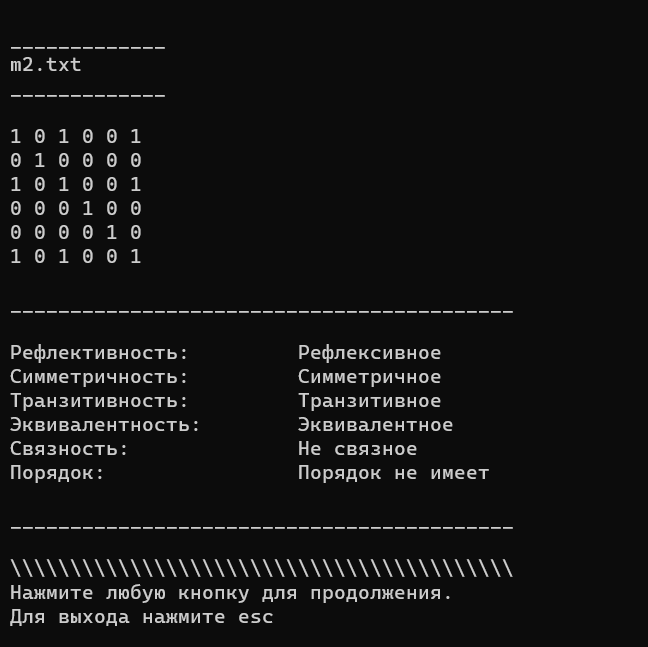
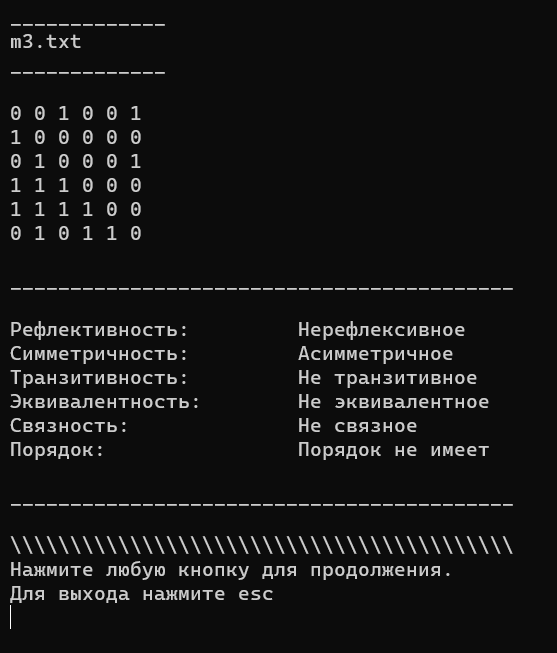
CheckTransitive() – возвращает строку Транзитивное / Не транзитивное. Три цикла проверяют (x;y),(y;z) и (x;z). Если (x;y)=(y;z)=1, а (x;z) это «Не транзитивное» отношение.

CheckEquivalence(string refl, string sym, string tran)- получает результаты трех функций выше и если матрица имеет свойство «Рефлексивное», «Симметричное» и «Транзитивное» матрица обладает свойством «Эквивалентное», Иначе «Не эквивалентное».

CheckConnectivity() – возвращает строку «Связное»/«Не связное». Программа проверяет значения выше главной диагонали и сравнивает их с обратными значениями. Если они не равны на всей матрице то матрица «Связное», иначе «Не связное».

CheckOrder(string refl, string sym, string tran, string conn)- на основе результатов полученных выше получает порядок матрицы. Первое условие проверяет на «Антисимметричное» и «Транзитивное» .  
второе условие проверяет «Нестрогое» через «Рефлективное» если «Не Рефлективное» порядок «Строгое». Вторым условие определяется полнота Если «Связное» это «полное», иначе «не полное».

CheckAll(string path) – получает строку с адресов матрицы. Выводит свойства Матрицы на экран.

Тесты  

**Код программы**

using System.IO;

namespace MatrixProperties

{

public class Program

{

static int[,] matrix;

static string path = "m1.txt";

static void Main(string[] args)

{

Console.WriteLine("\*Анализ матрицы отношений\*");

for (int i = 1; i <= 3; i++)

{

path = "m" + i.ToString() + ".txt";

CheckAll(path);

Console.WriteLine("Нажмите любую кнопку для продолжения.\nДля выхода нажмите esc");

if (Console.ReadKey(true).Key == ConsoleKey.Escape)

Environment.Exit(0);

}

}

public static void CheckAll(string path)

{

ReadMatrix(path);

Console.WriteLine("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\n"+path+"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

WriteMatrix();

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

string refl = CheckReflexivity();

Console.WriteLine("Рефлективность: \t"+refl);

string sym = CheckSymmetry();

Console.WriteLine("Симметричность: \t"+sym);

string tran = CheckTransitive();

Console.WriteLine("Транзитивность: \t"+tran);

string equi = CheckEquivalence(refl,sym,tran);

Console.WriteLine("Эквивалентность:\t"+equi);

string conn = CheckConnectivity();

Console.WriteLine("Связность:\t\t" + conn);

string ord = CheckOrder(refl, sym, tran, conn);

Console.WriteLine("Порядок:\t\t"+ord);

Console.WriteLine("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

Console.WriteLine(@"\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\");

}

public static string CheckOrder(string refl, string sym, string tran, string conn)

{

if (sym == "Антисимметричное"&& tran=="Транзитивное")

{

if(refl== "Рефлексивное")

{

if (conn == "Связное") return "Нестрогое, полное";

else return "Нестрогое, частичное";

}

else

{

if (conn == "Связное") return "Строгое, полное";

else return "Строгое, частичное";

}

}

return "Порядок не имеет";

}

public static string CheckConnectivity()

{

bool isConnectivity=true;

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

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

{

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

isConnectivity = false;

}

if (isConnectivity) return "Связное";

else return "Не связное";

}

public static string CheckEquivalence(string refl, string sym, string tran)

{

if (refl == "Рефлексивное" && sym == "Симметричное" && tran == "Транзитивное")

return "Эквивалентное";

else return "Не эквивалентное";

}

public static string CheckTransitive()

{

bool isTransitive = true;

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

{

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

{

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

{

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

{

isTransitive = false;

return "Не транзитивное";

}

}

}

}

if (isTransitive) { return "Транзитивное"; }

return "";

}

public static void ReadMatrix(string path)

{

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

matrix = new int[lines.GetLength(0), (lines[0].Length + 1) / 2];

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

{

string[] buf = lines[i].Split(' ');

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

{

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

}

}

}

public static void WriteMatrix()

{

if (matrix == null) { Console.WriteLine("Матрица пуста. "); return; }

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

{

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

{

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

}

Console.WriteLine();

}

Console.WriteLine();

}

public static string CheckReflexivity()

{

bool isReflexivity = true;

bool isNotReflexivity = true;

for (int i = 0, j = 0; i < matrix.GetLength(0) && (isReflexivity || isNotReflexivity); i++, j++)

{

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

if (!(matrix[i, j] == 0 && isNotReflexivity)) isNotReflexivity = false;

}

if (isReflexivity) { return "Рефлексивное"; }

else if (isNotReflexivity) { return "Нерефлексивное"; }

else { return "Частично рефлексивное"; }

}

public static string CheckSymmetry()

{

bool isSymmetry = true;

bool isAntiSymmetry = true;

for (int i = 0; i < matrix.GetLength(0) && (isSymmetry || isAntiSymmetry); i++)

{

for (int j = i + 1; j < matrix.GetLength(1)&& (isSymmetry || isAntiSymmetry); j++)

{

if (!(matrix[i, j] \* matrix[j, i] == matrix[i, j])) isSymmetry = false;

if (!(matrix[i, j] \* matrix[j, i] == 0)) isAntiSymmetry = false;

}

}

string isReflexivity = CheckReflexivity();

if (isSymmetry) { return "Симметричное"; }

else if (isAntiSymmetry && isReflexivity != "Нерефлексивное")

{ return "Антисимметричное"; }

else if (isAntiSymmetry && isReflexivity == "Нерефлексивное")

{ return "Асимметричное"; }

else return "не соответствует";

}

}

}