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»

Тема: Свойства отношений.

Выполнил:

Студент группы ИВТ-20-2б

Карелов Вадим Андреевич

Проверила:

Старший преподаватель кафедры ИТАС

Рустамханова Г.И.

Пермь, 2021

**Постановка задачи**

Ввести отношения через матрицу и вывести свойства:

1. Рефлексивные / антирефлексивные / не рефлексивные
2. Симметричные / асимметричные / антисимметричные / не симметричные
3. Связные / не связные
4. Транзитивные / не транзитивные

**Анализ задачи**

1. Вводимая матрица хранится в отдельной статической переменной в виде двумерного целочисленного массива.
2. Программа предусматривает 2 способа ввода: вручную и из файла.

static void MatrixInput()

{

string vvod;

bool ok = false;

while (!ok)

{

vvod = "";

while (vvod != "1" && vvod != "2")

{

Console.WriteLine("Выберете способ ввода матрицы: 1 - вручную, 2 - из файла");

vvod = Console.ReadLine();

}

switch (vvod)

{

case "1": ok = ManualMatrixInput(); break;

case "2": ok = MatrixFromFile(); break;

}

}

}

1. После ввода матрица выводится на экран пользователя.

static void Print()

{

Console.WriteLine("Вывод матрицы");

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

{

string str = "";

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

{

str += mat[i, j].ToString() + " ";

}

Console.WriteLine(str);

}

}

1. Проверка всех свойств производится из отдельного метода, в котором поочередно и при необходимости вызываются методы для проверки отдельных свойств.
2. Рефлексивность и антирефлексивность отличаются только проверяемым элементом на главной диагонали.

static bool IsReflective()

{

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

{

if (mat[i, i] == 0) return false;

}

return true;

}

static bool IsAntireflective()

{

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

{

if (mat[i, i] == 1) return false;

}

return true;

}

static void CheckProperties()

{

Console.WriteLine("\nСвойства отношений:");

if (IsReflective())

Console.WriteLine("Рефлексивные");

else if (IsAntireflective())

Console.WriteLine("Антирефлексивные");

if (IsSymmetric())

Console.WriteLine("Симметричные");

else if (IsAsymmetric())

Console.WriteLine("Ассимитричные");

else if (IsAntisemetric())

Console.WriteLine("Антисимметричные");

if (IsConnective())

Console.WriteLine("Связные");

if (IsTransitive())

Console.WriteLine("Транзитивные");

}

1. Симметричность.

static bool IsSymmetric()

{

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

{

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

{

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

}

}

return true;

}

static bool IsAntisemetric()

{

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

{

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

{

if (i != j && mat[i, j] \* mat[j, i] == 1) return false;

}

}

return true;

}

static bool IsAsymmetric()

{

return IsAntisemetric() && IsAntireflective();

}

1. Связность.

static bool IsConnective()

{

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

{

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

{

if (mat[i, j] + mat[j, i] == 0) return false;

}

}

return true;

}

1. Транзитивность.

static bool IsTransitive()

{

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

{

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

{

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

{

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

{

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

{

return false;

}

}

}

}

}

return true;

}

**Код программы**

using System;

using System.IO;

using System.Linq;

namespace RelationshipProperties

{

class Program

{

static void Main(string[] args)

{

MatrixInput();

Print();

CheckProperties();

}

static int[,] mat;

static void MatrixInput()

{

string vvod;

bool ok = false;

while (!ok)

{

vvod = "";

while (vvod != "1" && vvod != "2")

{

Console.WriteLine("Выберете способ ввода матрицы: 1 - вручную, 2 - из файла");

vvod = Console.ReadLine();

}

switch (vvod)

{

case "1": ok = ManualMatrixInput(); break;

case "2": ok = MatrixFromFile(); break;

}

}

}

static bool ManualMatrixInput()

{

string vvod = "";

int n;

while (!int.TryParse(vvod, out n) || n < 3 || n > 5)

{

Console.WriteLine("Введите размер квадратной матрицы (3 - 5)");

vvod = Console.ReadLine();

}

mat = new int[n, n];

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

{

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

{

string str = "";

while (!int.TryParse(str, out mat[i, j]))

{

Console.WriteLine($"Введите элемент {i + 1}.{j + 1}:");

str = Console.ReadLine();

}

}

}

return true;

}

static bool MatrixFromFile()

{

Console.WriteLine("Файл input.txt");

Console.WriteLine("Формат ввода:");

Console.WriteLine("Размерность матрицы");

Console.WriteLine("Все элементы в одной строке");

Console.WriteLine("\nДля продолжения нажмите любую клавишу");

Console.ReadKey();

try

{

StreamReader strR = new StreamReader("input.txt", System.Text.Encoding.Default);

int n = int.Parse(strR.ReadLine());

int[] t = strR.ReadLine().Split(' ').Select(int.Parse).ToArray();

strR.Close();

int[,] tmat = new int[n, n];

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

{

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

{

tmat[i, j] = t[i \* n + j];

}

}

mat = tmat;

return true;

}

catch

{

Console.WriteLine("Проверьте корректность файла\n");

return false;

}

}

static void Print()

{

Console.WriteLine("Вывод матрицы");

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

{

string str = "";

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

{

str += mat[i, j].ToString() + " ";

}

Console.WriteLine(str);

}

}

static void CheckProperties()

{

Console.WriteLine("\nСвойства отношений:");

if (IsReflective())

Console.WriteLine("Рефлексивные");

else if (IsAntireflective())

Console.WriteLine("Антирефлексивные");

if (IsSymmetric())

Console.WriteLine("Симметричные");

else if (IsAsymmetric())

Console.WriteLine("Ассимитричные");

else if (IsAntisemetric())

Console.WriteLine("Антисимметричные");

if (IsConnective())

Console.WriteLine("Связные");

if (IsTransitive())

Console.WriteLine("Транзитивные");

}

// ==== Properties ====

static bool IsReflective()

{

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

{

if (mat[i, i] == 0) return false;

}

return true;

}

static bool IsAntireflective()

{

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

{

if (mat[i, i] == 1) return false;

}

return true;

}

static bool IsSymmetric()

{

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

{

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

{

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

}

}

return true;

}

static bool IsAntisemetric()

{

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

{

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

{

if (i != j && mat[i, j] \* mat[j, i] == 1) return false;

}

}

return true;

}

static bool IsAsymmetric()

{

return IsAntisemetric() && IsAntireflective();

}

static bool IsConnective()

{

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

{

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

{

if (mat[i, j] + mat[j, i] == 0) return false;

}

}

return true;

}

static bool IsTransitive()

{

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

{

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

{

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

{

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

{

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

{

return false;

}

}

}

}

}

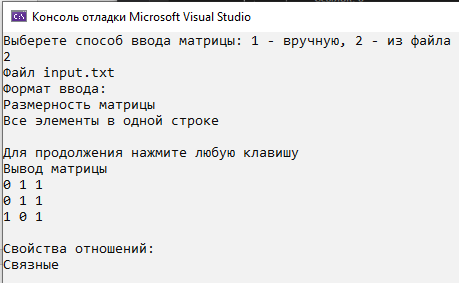
return true;

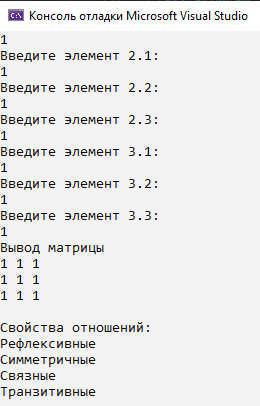
}

}

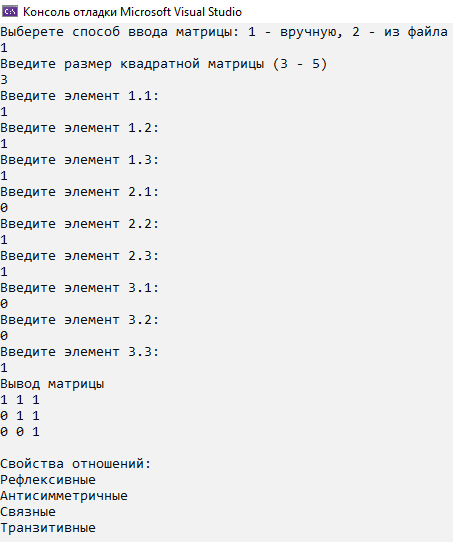
}

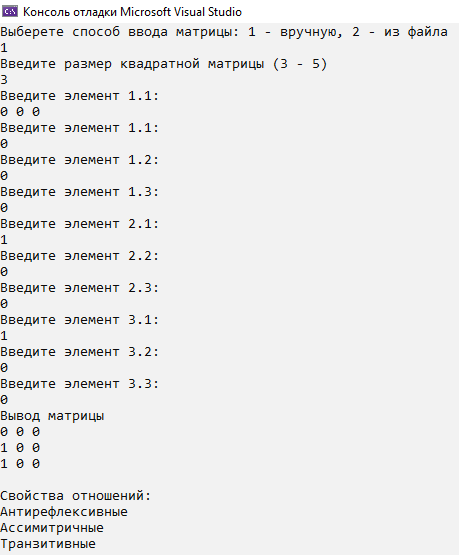
**Пример работы программы**











**Ссылки**

GitHub: https://github.com/VadimKarelov/RelationshipProperties