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**

Дискретная математика

Семестр 3

Лабораторная работа 2

Выполнил  
студент группы РИС-22-2б  
Баяндин К.С.

Проверила  
старший преподаватель кафедры ИТАС  
Рустамханова Г.И.

Пермь, 2023

Постановка задачи

Создать консольное приложение, выводящее свойства бинарных отношений

Для этого разработать:

1. Бинарную матрицу, записанную в файле
2. Считывание информации из файла
3. Анализ матрицы на отношения и вывод свойства на экран

Алгоритмы работы программы

Для работы с файлом был разработан следующий алгоритм: в переменную *path* указывается путь к текстовому файлу. Если файла не существует, то он создается. Таким образом, можно создать бинарную матрицу в файле, не выходя из программы:

if (!File.Exists(path))

{

// Создание файла

using (StreamWriter sw = File.CreateText(path))

{

Console.WriteLine("Введите количество столбцов: ");

int numRow = Int32.Parse(Console.ReadLine());

Console.WriteLine("Введите элементы матрицы: ");

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

sw.WriteLine(Console.ReadLine());

}

}

После создания файла файл открывается и происходит анализ бинарной матрицы внутри файла. Первым делом, матрица выводится на экран. Если файл был создан до нас, то вывод матрицы на экран позволит нам узнать, что находится внутри файла. Более того, вывод матрицы позволяет нам с помощью самостоятельного анализа проверять работу программы.

Console.WriteLine("Ваша матрица: ");

while ((s = sr.ReadLine()) != null)

{

Console.WriteLine(s);

k++;

matrix += s + " ";

}

После прочтения файла, матрциа записывается в строку, а далее в массив. Далее вызываются функции, в которые передается полученный массив. Вся работа с файлом происходит в главной функции *Main().*

int[,] arr = new int[rows, columns];

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

{

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

{

arr[i, j] = (int)Char.GetNumericValue(matrix[a]);

a += 2;

}

}

//вывод свойств матрицы

IsReflex(matrix, columns, b);

IsSymmetric(arr, columns, b);

if (IsTransitive(arr) == true)

Console.WriteLine("Транзитивность");

if (IsComplete(arr) == true)

Console.WriteLine("Полнота");

Всего в программе реализовано 4 функции для вывода свойств:

1. Рефлексивность и Антирефлексивность – рефлексивны ли элементы сами себе

static void IsReflex(string matrix, int columns, int b)

{

int a = 0;

for (int i = 0; i < matrix.Length; i += 2 \* columns + 2)

{

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

a++;

else if (matrix[i] == '0')

b++;

}

if (a == columns)

Console.WriteLine("Рефлексивность");

if (b == columns)

Console.WriteLine("Антирефлексивность");

}

1. Симметричность, антисимметричность и асимметричность – сравнивает элементы на симметрию относительно главной диагонали

static void IsSymmetric(int[,] matrix, int columns, int b)

{

int a = 0;

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

{

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

{

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

{

a++;

}

}

if (a == 1)

break;

}

if (a == 0)

if (b == columns)

Console.WriteLine("Асимметричность");

else

Console.WriteLine("Симметричность");

if (a >= columns \* columns - columns - 1)

Console.WriteLine("Антисимметричность");

}

1. Транзитивность – если 1 элемент равен 2, а 2 элемент равен 3, то 1 элемент равен 3. Возвращает *true*, если это условие выполняется, *false*, если не удовлетворяет условию транзитивности

static bool IsTransitive(int[,] matrix)

{

int n = matrix.GetLength(0);

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;

}

1. Полнота – если все элементы имеют связи с другими элементами.

static bool IsComplete(int[,] matrix)

{

int n = matrix.GetLength(0);

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

{

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

{

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

return false;

}

}

return true;

}

**Результаты работы программы**

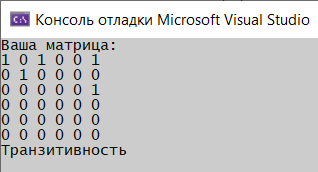


Рисунок 1 – результат работы с 1 матрицей

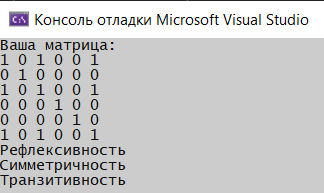


Рисунок 2 – результат работы с 2 матрицей

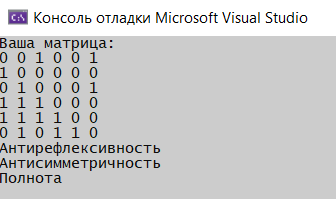


Рисунок 3 – результат работы с 3 матрицей

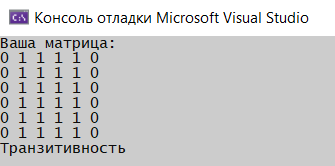


Рисунок 4 – результат работы с 4 матрицей

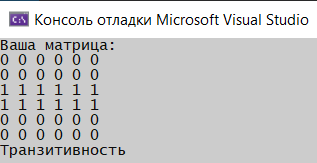


Рисунок 5 – результат работы с 5 матрицей

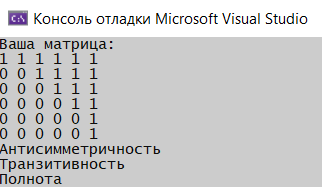


Рисунок 6 – результат работы с 6 матрицей

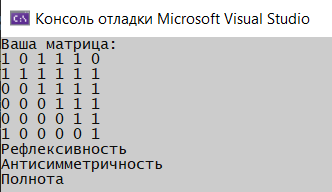


Рисунок 7 – результат работы с 7 матрицей

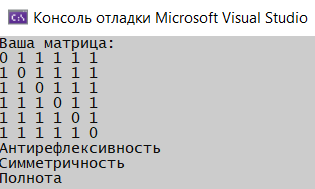


Рисунок 8 – результат работы с 8 матрицей

Листинг

using System;

using System.IO;

using System.Text;

namespace matrix\_properties

{

class Program

{

// Функция для проверки рефлексивности

static void IsReflex(string matrix, int columns, int b)

{

int a = 0;

for (int i = 0; i < matrix.Length; i += 2 \* columns + 2)

{

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

a++;

else if (matrix[i] == '0')

b++;

}

if (a == columns)

Console.WriteLine("Рефлексивность");

if (b == columns)

Console.WriteLine("Антирефлексивность");

}

// Функция для проверки симметрии

static void IsSymmetric(int[,] matrix, int columns, int b)

{

int a = 0;

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

{

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

{

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

{

a++;

}

}

if (a == 1)

break;

}

if (a == 0)

if (b == columns)

Console.WriteLine("Асимметричность");

else

Console.WriteLine("Симметричность");

if (a >= columns \* columns - columns - 1)

Console.WriteLine("Антисимметричность");

}

//функция для проверки транзитивности

static bool IsTransitive(int[,] matrix)

{

int n = matrix.GetLength(0);

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;

}

// Функция для проверки полноты

static bool IsComplete(int[,] matrix)

{

int n = matrix.GetLength(0);

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

{

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

{

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

return false;

}

}

return true;

}

static void Main(string[] args)

{

string path = @"C:\Users\trajt\source\repos\Discrete-mathematics-2-course\matrix\_properties\My\_text.txt";

if (!File.Exists(path))

{

// Создание файла

using (StreamWriter sw = File.CreateText(path))

{

Console.WriteLine("Введите количество столбцов: ");

int numRow = Int32.Parse(Console.ReadLine());

Console.WriteLine("Введите элементы матрицы: ");

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

sw.WriteLine(Console.ReadLine());

}

}

// Открытие файла для чтения

using (StreamReader sr = File.OpenText(path))

{

string s;

string matrix = "";

int k = 0;

//вывод матрицы + создание строки матрицы

Console.WriteLine("Ваша матрица: ");

while ((s = sr.ReadLine()) != null)

{

Console.WriteLine(s);

k++;

matrix += s + " ";

}

int rows = k;

int columns = rows;

int a = 0;

int b = 0;

//создание матриц

int[,] arr = new int[rows, columns];

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

{

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

{

arr[i, j] = (int)Char.GetNumericValue(matrix[a]);

a += 2;

}

}

//вывод свойств матрицы

IsReflex(matrix, columns, b);

IsSymmetric(arr, columns, b);

if (IsTransitive(arr) == true)

Console.WriteLine("Транзитивность");

if (IsComplete(arr) == true)

Console.WriteLine("Полнота");

}

}

}

}