Министерство образования и науки РФ

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Дискретная математика

Лабораторная работа № 2

Тема: «Свойства матриц»

Выполнил: студентка группы ИВТ-23-1б

Хорошилова К. П.

Проверил: ст. преподаватель

Рустамханова Г. И.

г. Пермь – 2024**Цель работы**

Реализовать программу, которая будет считывать матрицу бинарных отношений и определять её свойства.

**Задачи работы**

1. Реализовать считывание матрицы бинарных отношений из файла и консоли.
2. Проверить матрицу на свойство рефлексивности или антирефлексивности.
3. Проверить матрицу на свойство симметричности, антисимметричности и асимметричности.
4. Проверить матрицу на свойство транзитивности.
5. Проверить матрицу на свойство связности.
6. Реализовать вывод матрицы и свойств.
7. Написать тесты для проверки корректности работы программы.
8. Оформить отчет.

# Код программы

using System.IO;

using System.Text.RegularExpressions;

namespace Дискретная\_математика\_\_\_Свойства\_отношений

{

internal class Program

{

int ChekInput()

{

bool isCorectInput = false;

string str;

int number;

do

{

isCorectInput = Int32.TryParse(str = Console.ReadLine(), out number);

} while (!isCorectInput);

return number;

}

void ProcessingString(string str, ref bool[,] matrix, int i)

{

Regex regex = new Regex("[0-1]");

MatchCollection matches = regex.Matches(str);

int j = 0;

if (matches.Count > 0)

{

foreach (Match match in matches)

{

switch (match.Value)

{

case "0":

{

matrix[i, j] = false;

break;

}

case "1":

{

matrix[i, j] = true;

break;

}

}

j++;

}

}

else

{

Console.WriteLine("Совпадений не найдено");

}

}

void WritingDataFile(ref bool[,] matrix)

{

StreamReader file = new StreamReader("C:\\Users\\Asus\\source\\repos\\lab\_disc\_2\\lab\_disc\_2\\matrix.txt");

string str;

int count = 0;

while (!file.EndOfStream)

{

str = file.ReadLine();

ProcessingString(str, ref matrix, count);

count++;

}

file.Close();

}

void InputMatrix(ref bool[,] matrix)

{

string strin;

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

{

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

{

strin = Console.ReadLine();

if (strin == "0" || strin == "1")

{

if (strin == "0")

{

matrix[str, col] = false;

}

else

{

matrix[str, col] = true;

}

}

else

{

Console.WriteLine("Введите другое значение");

col--;

}

}

}

}

void PrintMatrix(ref bool[,] matrix)

{

if (matrix.Length != 0)

{

Console.WriteLine("Матрица отношений: ");

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

{

Console.Write('|');

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

{

if (matrix[i, j])

{

Console.Write(1);

}

else

{

Console.Write(0);

}

Console.Write('\t');

}

Console.WriteLine('|');

}

}

else

{

Console.WriteLine("Матрица отношений пуста");

}

}

void AnalysisMatrix(ref bool[,] matrix)

{

bool isChekReflex = true;

CheckReflex(ref matrix, ref isChekReflex);

ChekSymmetry(ref matrix, ref isChekReflex);

ChekTransitivity(ref matrix);

ChekConnectivity(ref matrix);

}

void CheckReflex(ref bool[,] matrix, ref bool isChekReflex)

{

int count = 0;

bool isChange = true;

bool isTemp = true;

while (count < 6 && isChange)

{

if (matrix[count, count])

{

isChekReflex = true;

}

else

{

isChekReflex = false;

}

if (count != 0)

{

if (!(isTemp == isChekReflex))

{

isChange = false;

}

}

else

{

isTemp = isChekReflex;

}

count++;

}

if (isChange)

{

if (isChekReflex)

{

Console.WriteLine("1) Матрица рефлексивная. На главной диагонали 1");

}

else

{

Console.WriteLine("1) Матрица антирефлексивная. На главной диагонали 0");

}

}

else

{

Console.WriteLine("1) Не выполняется");

isChekReflex = true;

}

}

void ChekSymmetry(ref bool[,] matrix, ref bool isChekReflex) // При 7 или 6 не прочитывает смену

{

bool isChekSymmetry = true;

bool isChange = true;

bool isTemp = true;

int str = 0, col = 0;

while (str < 6 && isChange)

{

while (col < 6 && isChange)

{

if (str != col)

{

if (matrix[str, col] == matrix[col, str])

{

isChekSymmetry = true;

}

else

{

isChekSymmetry = false;

}

if (str == 0 && col == 1)

{

isTemp = isChekSymmetry;

}

else

{

if (isTemp != isChekSymmetry)

{

isChange = false;

}

}

}

col++;

}

str++;

col = str;

}

if (!isChange)

{

Console.WriteLine("2) Не выполняется");

}

else

{

if (isChekSymmetry && isChekReflex)

{

Console.WriteLine("2) Матрица симметричная");

}

else

{

if (isChekReflex)

{

Console.WriteLine("2) Матрица антисимметричная");

}

else

{

Console.WriteLine("2) Матрица асимметриная");

}

}

}

}

void ChekTransitivity(ref bool[,] matrix)

{

int x = 0, y = 0, z = 0;

bool isChekTransit = true;

while (x < 6 && isChekTransit)

{

while (y < 6 && isChekTransit)

{

if (x != y)

{

while (z < 6 && isChekTransit)

{

if (y != z)

{

if (matrix[x, y])

{

if (matrix[y, z])

{

isChekTransit = matrix[x, z] == matrix[y, z];

}

}

}

z++;

}

}

y++;

z = 0;

}

x++;

y = 0;

}

if (isChekTransit)

{

Console.WriteLine("3) Матрица транзитивная");

}

else

{

Console.WriteLine("3) Матрица не транзитивная");

}

}

void ChekConnectivity(ref bool[,] matrix)

{

bool isChekConnectivety = true;

int str = 0, col = 0;

while (str < 6 && isChekConnectivety)

{

while (col < 6 && isChekConnectivety)

{

if (str != col)

{

if (!matrix[str, col])

{

if (!matrix[col, str])

{

isChekConnectivety = false;

}

}

}

col++;

}

str++;

col = str;

}

if (isChekConnectivety)

{

Console.WriteLine("4) Матрица связная");

}

else

{

Console.WriteLine("4) Матрица несвязная");

}

}

void Menu()

{

bool isEnd = true;

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

do

{

Console.WriteLine('\t' + "Меню");

Console.WriteLine("1) Извлечение данных из файла");

Console.WriteLine("2) Заполнение матрицы вручную");

Console.WriteLine("3) Отображение системы отношений");

Console.WriteLine("4) Анализ свойств отношений");

Console.WriteLine("5) Законьчить работу");

Console.WriteLine("Введите номер действия");

switch (ChekInput())

{

case 1:

{

WritingDataFile(ref matrix);

break;

}

case 2:

{

InputMatrix(ref matrix);

break;

}

case 3:

{

PrintMatrix(ref matrix);

break;

}

case 4:

{

AnalysisMatrix(ref matrix);

break;

}

case 5:

{

isEnd = false;

break;

}

default:

{

Console.WriteLine("Введите номер действия из списка.");

break;

}

};

} while (isEnd);

}

static void Main(string[] args)

{

Program main = new Program();

main.Menu();

}

}

}

**Анализ кода**

Этот код представляет собой консольное приложение на языке C#, которое выполняет операции с матрицами отношений, анализируя их свойства в рамках дискретной математики. Программа позволяет вводить матрицу вручную или считывать ее из файла, выводить матрицу на экран и проверять такие свойства, как рефлексивность, симметричность, транзитивность и связность.

Программа начинается с метода Main, который создает экземпляр класса Program и вызывает метод Menu. Menu представляет собой главное меню приложения, где пользователю предлагается выбрать одно из действий. Работа с матрицами осуществляется через булевый двумерный массив размером 6x6.

Метод ChekInput отвечает за корректный ввод данных. Пользователь должен ввести целое число, и метод проверяет, действительно ли это число. Если ввод некорректен, пользователь должен попробовать снова.

Метод ProcessingString обрабатывает строку, содержащую значения 0 и 1. С помощью регулярного выражения извлекаются все символы, соответствующие этим значениям, и преобразуются в булевый формат (false для 0 и true для 1). Результаты записываются в заданную строку матрицы.

Метод WritingDataFile загружает матрицу из файла matrix.txt. Программа построчно считывает данные из файла, используя метод ProcessingString для обработки каждой строки и записи значений в матрицу.

Метод InputMatrix реализует ручной ввод матрицы. Пользователь заполняет элементы матрицы, вводя значения 0 или 1 для каждого элемента. Если ввод некорректен (например, введено значение, отличное от 0 или 1), программа просит повторить ввод для соответствующего элемента.

Метод PrintMatrix выводит текущую матрицу на экран. Каждый элемент матрицы отображается в виде 0 или 1, а строки и столбцы разделяются пробелами.

Метод AnalysisMatrix анализирует свойства матрицы. Он последовательно вызывает другие методы для проверки рефлексивности, симметричности, транзитивности и связности. Результаты каждой проверки выводятся на экран.

Метод CheckReflex проверяет рефлексивность матрицы. Рефлексивной считается матрица, если на главной диагонали находятся только единицы. Если вместо единиц на диагонали расположены только нули, матрица антирефлексивная. Если же условие рефлексивности нарушено, программа выводит, что матрица не является ни рефлексивной, ни антирефлексивной.

Метод ChekSymmetry проверяет симметричность матрицы. Матрица считается симметричной, если для любого элемента матрицы matrix[i, j] равно matrix[j, i]. Если это условие выполняется для всех элементов, программа сообщает о симметричности матрицы. Если же условие нарушено, матрица может быть антисимметричной (когда элемент не равен своему зеркальному отображению) или асимметричной (если антисимметричность нарушена из-за рефлексивных элементов).

Метод ChekTransitivity проверяет транзитивность матрицы. Транзитивной считается матрица, если для любых элементов matrix[x, y] и matrix[y, z], где оба равны true, выполняется условие, что matrix[x, z] также равно true. Если хотя бы одно нарушение транзитивности найдено, программа сообщает, что матрица не транзитивная.

Метод ChekConnectivity проверяет связность матрицы. Связной считается матрица, в которой для каждой пары элементов либо matrix[i, j] равно true, либо matrix[j, i] равно true. Если это условие выполняется для всех пар элементов, матрица связная, иначе – несвязная.

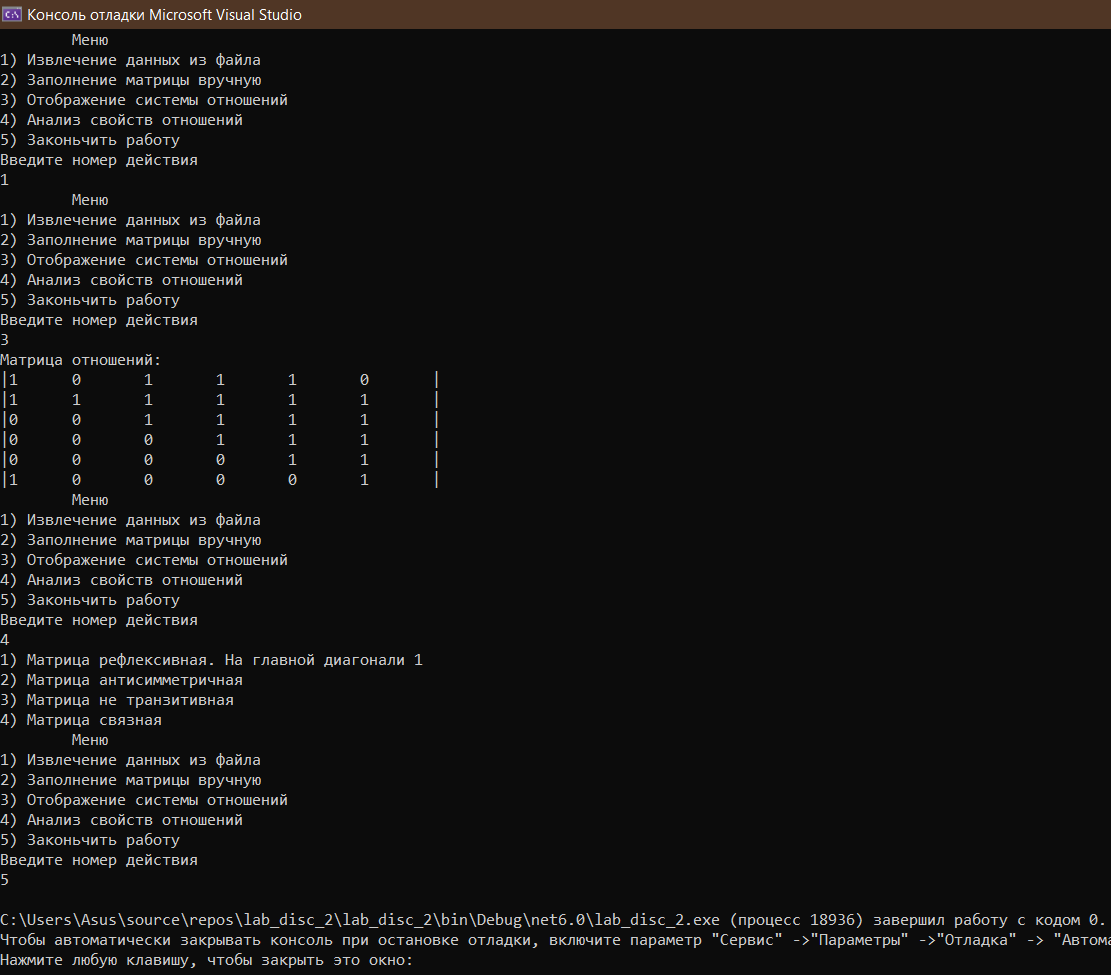
Метод Menu управляет основным циклом работы программы. Пользователю предлагается меню с пятью вариантами действий:

1. Загрузка данных из файла.
2. Ручной ввод матрицы.
3. Отображение текущей матрицы.
4. Анализ свойств матрицы.
5. Завершение работы программы.

Каждое действие вызывается соответствующим методом, описанным выше. Программа продолжает работу, пока пользователь не выберет завершение.

Ссылка на Github: <https://github.com/ksinph/disc_math>

**Тесты**



# Заключение

# В ходе лабораторной работы я реализовала код, который будет считывать матрицу бинарных отношений и определять её свойства.