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

**«Пермский национальный исследовательский
политехнический университет»**

Электротехнический факультет
Кафедра «Информационные технологии и автоматизированные системы»
направление подготовки: 09.03.01– «Информатика и вычислительная техника»

Лабораторная работа
по дисциплине
«Дискретная математика и математическая логика»
на тему
«Свойства отношений»

Выполнил студент гр. ИВТ-23-1б
Попонин Михаил Александрович

Проверил:

_____

_____

_____ _____
(оценка) (подпись)

_____
(дата)

г. Пермь, 2024

Цель работы и задачи работы

Целью данной работы является разработка программы для анализа бинарной матрицы отношений. Программа проверяет свойства матрицы, такие как рефлексивность, анти-рефлексивность, симметричность, асимметричность, антисимметричность, транзитивность и связность, а затем выводит результаты проверки.

Код программы

Листинг 1 – весь код программы

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
void menu();
void request(string filename)
{
	int n = 0;
	bool arr[255][255];
	ifstream file(filename);
	if (!file)
	{
		cout << "\n# Error with file";

		cout << "\n\n-   -   -   -   -   -   -   -   -   -\n\n";

		menu();
		return;
	}
	string line;
	getline(file, line);
	for (int i = 0; i < line.length(); i++)
	{
		if (line[i] == '0' || line[i] == '1')
		{
			n++;
		}
	}
	cout << "# Matrix size - " << n << "x" << n << "\n\n";
	file.clear();
	file.seekg(0);
	for (int i = 0; i < n; i++)
	{
		for (int j = 0; j < n; j++)
		{
			file >> arr[i][j];
		}
	}
	file.close();

	cout << "# Matrix\n";
	for (int i = 0; i < n; i++)
	{
		cout << "\n    ";
		for (int j = 0; j < n; j++)
		{
			cout << arr[i][j] << " ";
		}
	}
	cout << endl;

	cout << "\n# Reflexivity          ";
	bool is_reflexive = true;
	for (int i = 0; i < n; i++)
	{
		if (arr[i][i] == 0)
		{
			is_reflexive = false;
			break;
		}
	}
```

```
	cout << (is_reflexive ? "[ + ]" : "[ - ]");

	cout << "\n# Anti_reflexivity     ";
	bool is_anti_reflexive = true;
	for (int i = 0; i < n; i++)
	{
		if (arr[i][i] == 1)
		{
			is_anti_reflexive = false;
			break;
		}
	}
	cout << (is_anti_reflexive ? "[ + ]" : "[ - ]");

	cout << "\n# Symmetry             ";
	bool is_symmetric = true;
	for (int i = 0; i < n; i++)
	{
		for (int j = i + 1; j < n; j++)
		{
			if (arr[i][j] != arr[j][i])
			{
				is_symmetric = false;
				break;
			}
		}
		if (!is_symmetric) break;
	}
	cout << (is_symmetric ? "[ + ]" : "[ - ]");

	cout << "\n# Asymmetry            ";
	bool is_asymmetric = true;
	for (int i = 0; i < n; i++)
	{
		for (int j = i + 1; j < n; j++)
		{
			if (arr[i][j] == arr[j][i] || arr[i][i] != 0)
			{
				is_asymmetric = false;
				break;
			}
		}
		if (!is_asymmetric) break;
	}
	cout << (is_asymmetric ? "[ + ]" : "[ - ]");

	cout << "\n# Anti_symmetry        ";
	bool is_anti_symmetric = true;
	for (int i = 0; i < n; i++)
	{
		for (int j = i + 1; j < n; j++)
		{
			if (arr[i][j] == arr[j][i] && arr[i][j] == 1)
			{
				is_anti_symmetric = false;
				break;
			}
		}
		if (!is_anti_symmetric) break;
	}
	cout << (is_anti_symmetric ? "[ + ]" : "[ - ]");

	cout << "\n# Transitivity         ";
	bool is_transitive = true;
	for (int i = 0; i < n; i++)
```

```
	{
		for (int j = 0; j < n; j++)
		{
			if (arr[i][j]) {
				for (int k = 0; k < n; k++)
				{
					if (arr[j][k] && !arr[i][k])
					{
						is_transitive = false;
						break;
					}
				}
				if (!is_transitive) break;
			}
		}
		if (!is_transitive) break;
	}
	cout << (is_transitive ? "[ + ]" : "[ - ]");

	cout << "\n# Connectedness        ";
	bool is_connected = true;
	for (int i = 0; i < n; i++)
	{
		for (int j = 0; j < n; j++)
		{
			if (i != j and arr[i][j] != 1 and arr[j][i] != 1)
			{
				is_connected = false;
				break;
			}
		}
		if (!is_connected) break;
	}
	cout << (is_connected ? "[ + ]" : "[ - ]") << endl << endl;


	menu();
}
void menu()
{
	cout << "# Enter filename or exit :  ";
	string filename = "";
	cin >> filename;
	if (filename != "exit")
	{
		cout << "# filename - " << filename << endl;
		request(filename);
	}
}
int main()
{
	menu();
	return 0;
}
```

Объяснение функций

Программа выполняет следующие задачи:

1. Открытие и чтение файла, содержащего бинарную матрицу.
2. Автоматическое определение размерности матрицы на основе входных данных.
3. Проверка следующих свойств бинарной матрицы:
    - Рефлексивность: наличие единиц на диагонали матрицы.
    - Анти-рефлексивность: отсутствие единиц на диагонали.
    - Симметричность: равенство элементов arr[i][j] и arr[j][i] для всех индексов.
    - Асимметричность: отсутствие парных элементов arr[i][j] и arr[j][i], равных 1, а также наличие нулей на диагонали.
    - Антисимметричность: если arr[i][j] == 1 и arr[j][i] == 1, то это свойство не выполняется.
    - Транзитивность: если arr[i][j] == 1 и arr[j][k] == 1, то требуется, чтобы arr[i][k] == 1.
    - Связность: проверка существования хотя бы одного направления между любыми двумя элементами матрицы.
4. Отображение самой матрицы и результатов проверки свойств на экране.
5. Организация интерфейса пользователя, позволяющего вводить имя файла с матрицей или завершить работу программы.

Функция menu() организует взаимодействие с пользователем. Она запрашивает имя файла, передает его для анализа в функцию request() или завершает выполнение программы, если пользователь вводит "exit".

Функция request(string filename) выполняет всю основную работу по обработке файла, чтению матрицы, проверке ее свойств и выводу результатов. Если файл не найден, функция сообщает об ошибке и возвращает пользователя в главное меню.

Алгоритмы проверки каждого свойства реализованы в виде последовательных циклов с использованием условий. Программа предоставляет наглядные результаты в формате [+], если свойство выполняется, и [-], если не выполняется.

Пример работы программы

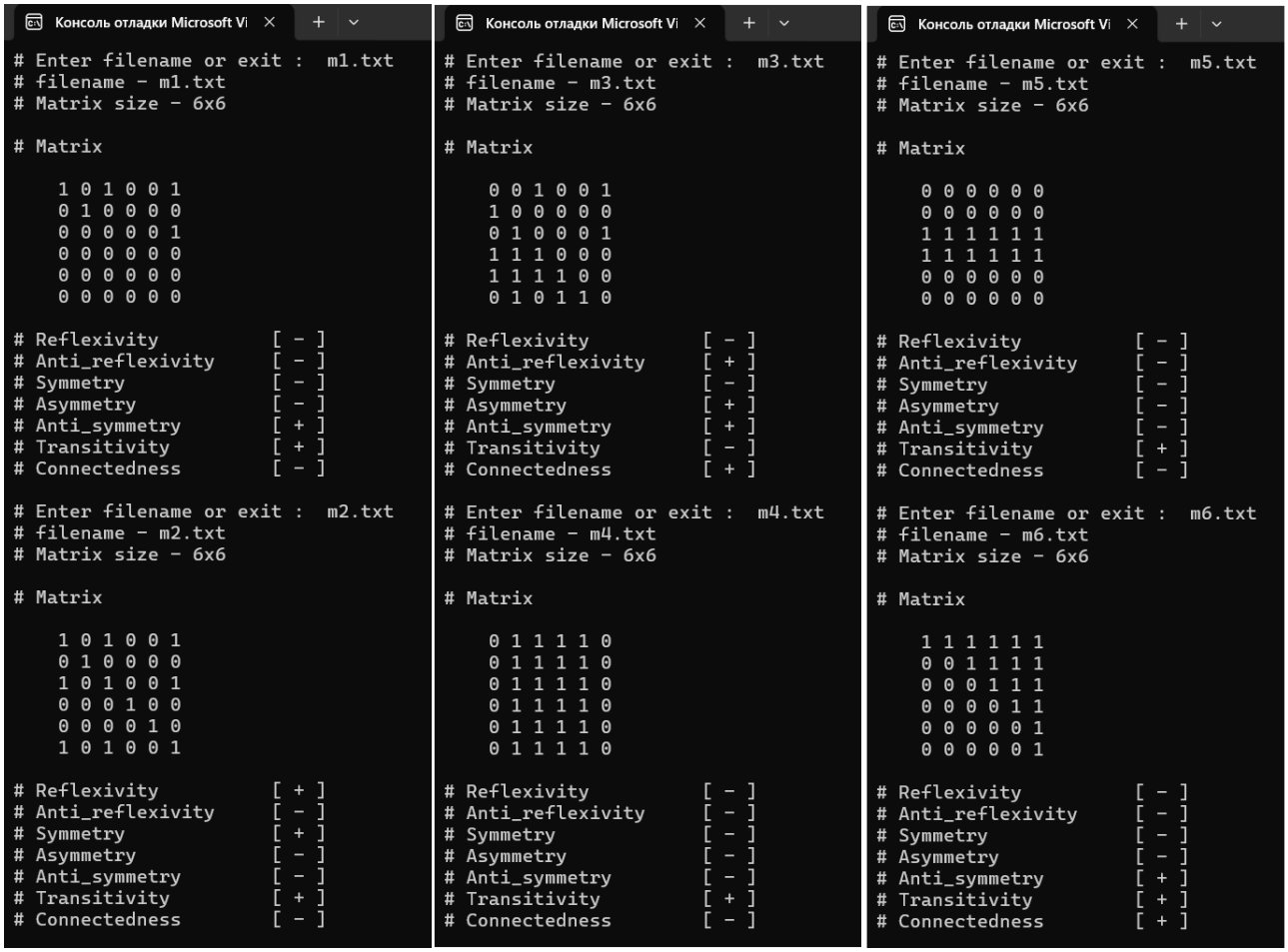

Рисунок 1 – Примеры работы программы