Министерство образования и науки РФ

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Дискретная математика и математическая логика

Лабораторная работа №2

Тема: «Определение свойств отношения по матрице»

Выполнил: студент группы ИВТ-23-1б

Адаев Даниил Дмитриевич

Проверил: Рустамханова Г. И.

Оглавление

**Цель работы 3**

**Функционал программы 4**

**Заключение 5**

**Тесты 6**

**Код 7**

**Цель работы**

Разработать программу, которая считывает матрицу из текстового файла и определяет свойства заданного отношения: рефлексивность, симметричность, транзитивность, полнота.

**Функционал программы**

Программа реализована на С++.

Программа считывает файл «matrix.txt» и помещает матрицу в двумерный вектор. Матрица должна быть квадратной, иначе выводится ошибка. Отображаются полученные значения вектора.

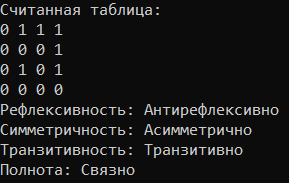
Выполняются три функции, определяющие свойства отношения:

void ReflexivitySymmetry(vector<vector<int>) выясняет рефлексивность и симметричность.  
Функция считывает число значений «1» на главной диагонали: если равно нулю – антирефлексивно, если равно числу рядов матрицы – рефлексивно, иначе – частично рефлексивно.  
Функция проверяет, симметричность значений в матрице относительно главной диагонали: если несоответствий нет – симметрично. Иначе проверяет на антисимметрию, если рефлексивно или асимметрию, если анти рефлексивно. Выводит соответствующий результат. Иначе выводит, что отношение не является симметричным, антисимметричным, асимметричным.

void Transitivity(vector<vector<int>>) выясняет транзитивность.  
Если функция находит в матрице единицу, выполняется проверка ряда, соответствующего второй координате: если в нем находится единица (выполняется отношение) в паре с элементом, с которым нет единицы (не выполняется отношение) в паре с первой координатой, определяет, как нетранзитивное. Если же таких случаев нет и выполнилось хотя бы одно условие транзитивности – определяет, как транзитивное. Иначе выводит, что транзитивность невозможно определить в данном множестве.

void Transitivity(vector<vector<int>>) выясняет полноту.  
Если функция находит в матрице пару значений, симметричных относительно главной диагонали, таких что оба равны нулю (не выполняются отношения), определяет, как несвязное, иначе – как связное.

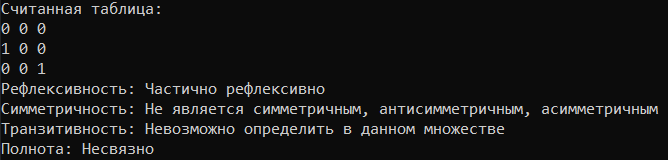
Пример программы:

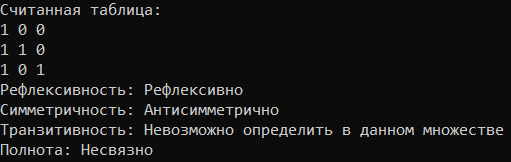


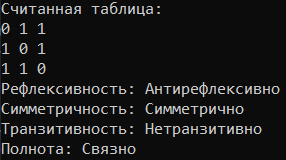
**Заключение**

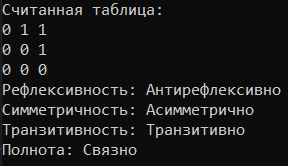
Программа создана успешно.

**Тесты**









**Код на C++**

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

using namespace std;

void ReflexivitySymmetry(vector<vector<int>> table)

{

cout << "Рефлексивность: ";

int number\_of\_elements = table.size();

int count = 0;

for (int i = 0; i < number\_of\_elements; i++)

{

if (table[i][i] == 1)

{

count++;

}

}

if (count == 0)

{

cout << "Антирефлексивно\n";

}

else if (count == number\_of\_elements)

{

cout << "Рефлексивно\n";

}

else

{

cout << "Частично рефлексивно\n";

}

cout << "Симметричность: ";

bool flag = true;

for (int y = 0; y < number\_of\_elements; y++)

{

for (int x = y + 1; x < number\_of\_elements; x++)

{

if (table[y][x] != table[x][y])

{

flag = false;

goto breakSymmetryCycle;

}

}

}

breakSymmetryCycle:

if (flag)

{

cout << "Симметрично\n";

return;

}

flag = true;

for (int y = 0; y < number\_of\_elements; y++)

{

for (int x = y + 1; x < number\_of\_elements; x++)

{

if (table[y][x] == 1 and table[x][y] == 1)

{

flag = false;

goto breakNoSymmetryCycle;

}

}

}

breakNoSymmetryCycle:

if (flag and count == 0)

{

cout << "Асимметрично\n";

}

else if (flag)

{

cout << "Антисимметрично\n";

}

else

{

cout << "Не является симметричным, антисимметричным, асимметричным\n";

}

}

void Transitivity(vector<vector<int>> table)

{

cout << "Транзитивность: ";

int number\_of\_elements = table.size();

bool true\_flag = true,

def\_flag = false;

for (int y = 0; y < number\_of\_elements; y++)

{

for (int x = 0; x < number\_of\_elements; x++)

{

if ((table[y][x] == 1) and (x != y))

{

for (int z = 0; z < number\_of\_elements; z++)

{

if ((def\_flag == false) and (x != z) and (table[x][z] == 1) and (table[y][z] == 1))

{

def\_flag = true;

}

if (table[x][z] == 1 and table[y][z] == 0)

{

true\_flag = false;

goto breakTransitivityCycle;

}

}

}

}

}

breakTransitivityCycle:

if (true\_flag)

{

if (def\_flag)

{

cout << "Транзитивно\n";

}

else

{

cout << "Невозможно определить в данном множестве\n";

}

}

else

{

cout << "Нетранзитивно\n";

}

}

void Completeness(vector<vector<int>> table)

{

cout << "Полнота: ";

int number\_of\_elements = table.size();

bool true\_flag = true;

for (int y = 0; y < number\_of\_elements; y++)

{

for (int x = y + 1; x < number\_of\_elements; x++)

{

if (table[y][x] == 0 and table[x][y] == 0)

{

true\_flag = false;

goto breakCompletenessCycle;

}

}

}

breakCompletenessCycle:

if (true\_flag)

{

cout << "Связно\n";

}

else

{

cout << "Несвязно\n";

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

ifstream ftable("matrix.txt");

if (!ftable)

{

cerr << "Ошибка открытия файла" << endl;

return 1;

}

vector<vector<int>> table;

string str;

int ymax = 0, xmax = 0, x = 0;

while (getline(ftable, str))

{

x = 0;

vector<int> tmpv;

for (char c : str)

{

if (c == '0')

{

tmpv.push\_back(0);

x++;

}

else if (c == '1')

{

tmpv.push\_back(1);

x++;

}

}

table.push\_back(tmpv);

ymax++;

xmax = max(x, xmax);

}

ftable.close();

if (ymax != xmax)

{

cerr << "Неверная таблица" << endl;

return 2;

}

cout << "Считанная таблица:" << endl;

for (const auto& row : table) {

for (const auto& elem : row) {

cout << elem << " ";

}

cout << endl;

}

ReflexivitySymmetry(table);

Transitivity(table);

Completeness(table);

}