

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.  
ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ  
компьютерной безопасности и  
криптографии

**Классификация бинарных отношений и системы замыканий**  
**ОТЧЁТ**  
**ПО ДИСЦИПЛИНЕ**  
**«ПРИКЛАДНАЯ УНИВЕРСАЛЬНАЯ АЛГЕБРА»**

студента 3 курса 331 группы  
специальности 10.05.01 Компьютерная безопасность  
факультета компьютерных наук и информационных технологий  
Яхина Шамиля Илдусовича

Преподаватель  
профессор, д.ф.-м.н.

\_\_\_\_\_ В. А. Молчанов  
подпись, дата

Саратов 2022

## СОДЕРЖАНИЕ

1	Бинарные отношения .....	4
1.1	Определение бинарного отношения .....	4
1.2	Классификация бинарных отношений .....	4
2	Системы замыканий .....	6
2.1	Определение системы замыканий .....	6
2.2	Лемма о системах замыканий бинарных отношений .....	6
3	Результаты работы .....	7
3.1	Описание алгоритма классификации бинарных отношений .....	7
3.1.1	Алгоритм проверки рефлексивности. Функция <i>bo_is_reflexive</i> .....	7
3.1.2	Алгоритм проверки антирефлексивности. Функция <i>bo_is_antireflexive</i> .....	7
3.1.3	Алгоритм проверки симметричности. Функция <i>bo_is_symmetric</i> .....	8
3.1.4	Алгоритм проверки антисимметричности. Функция <i>bo_is_antisymmetric</i> .....	8
3.1.5	Алгоритм проверки транзитивности. Функция <i>bo_is_transitive</i> .....	9
3.1.6	Алгоритм проверки отношения на эквивалентность, ква- зипорядок и порядок .....	10
3.2	Описание алгоритма построения основных замыканий бинарных отношений .....	10
3.2.1	Алгоритм построения замыкания относительно свойства рефлексивности .....	11
3.2.2	Алгоритм построения замыкания относительно свойства симметричности .....	11
3.2.3	Алгоритм построения замыкания относительно свойства транзитивности .....	12
3.2.4	Алгоритм построения замыкания относительно эквива- лентности .....	12
3.3	Код программы .....	12
3.4	Результаты тестирования программ .....	20
	ЗАКЛЮЧЕНИЕ .....	25

Цель работы: изучение основных свойств бинарных отношений и операций замыкания бинарных отношений.

# 1 Бинарные отношения

## 1.1 Определение бинарного отношения

Подмножества декартова произведения  $A \times B$  множеств  $A$  и  $B$  называются бинарными отношениями между элементами множеств  $A, B$  и обозначаются строчными греческими буквами:  $\rho, \sigma, \dots, \rho_1, \rho_2, \dots$

бинарными

Для бинарного отношения  $\rho \subset A \times B$  область определения  $D_\rho$  и множество значений  $E_\rho$  определяются как подмножества соответствующих множеств и по следующим формулам:

$$D_\rho = \{a : (a, b) \in \rho \text{ для некоторого } b \in B\},$$

$$E_\rho = \{b : (a, b) \in \rho \text{ для некоторого } a \in A\}.$$

## 1.2 Классификация бинарных отношений

Бинарное отношение  $\rho \subset A \cdot A$  называется:

1. *рефлексивным*, если  $(a, a) \in \rho$  для любого  $a \in A$ ;
2. *симметричным*, если  $(a, b) \in \rho \Rightarrow (b, a) \in \rho$ ;
3. *антисимметричным*, если  $(a, b) \in \rho$  и  $(b, a) \in \rho \Rightarrow a = b$ ;
4. *транзитивным*, если  $(a, b) \in \rho$  и  $(b, c) \in \rho \Rightarrow (a, c) \in \rho$

Основываясь на этом, можно выделить три типа отношений:

1. Отношение эквивалентности

Бинарное отношение  $\varepsilon$  на множестве  $A$  называется *отношением эквивалентности* (или просто *эквивалентностью*), если оно рефлексивно, симметрично и транзитивно,

2. Отношение квазипорядка

Бинарное отношение  $\omega$  на множестве  $A$  называется *отношением квазипорядка* (или просто *квазипорядком*), если оно рефлексивно и транзитивно,

3. Отношение порядка

Бинарное отношение  $\omega$  на множестве  $A$  называется *отношением порядка* (или просто *порядком*), если оно рефлексивно, антисимметрично и транзитивно,

Для того, чтобы реализовать алгоритм классификации бинарных отношений, удобно пользоваться матрицей бинарного отношения.

*Матрицей* бинарного отношения  $\rho$  между элементами множеств  $A =$

$\{a_1, \dots, a_m\}$  и  $B = \{b_1, \dots, b_n\}$  называется прямоугольная таблица  $M(\rho)$ , состоящая из  $m$  строк и  $n$  столбцов, в которой на пересечении  $i$ -ой строки и  $j$ -го столбца стоит элемент  $[M(\rho)]_{ij}$  из множества  $0,1$ , определяемый по правилу:

$$[M(\rho)]_{ij} = \begin{cases} 1 & , \text{ если } (a_i, b_j) \in \rho \\ 0 & , \text{ в противном случае} \end{cases}$$

## 2 Системы замыканий

### 2.1 Определение системы замыканий

Множество  $Z$  подмножеств множества  $A$  называется *системой замыканий*, если оно замкнуто относительно пересечений, т.е. выполняется

$$\cap B \in Z \text{ для любого подмножества } B \subset Z$$

### 2.2 Лемма о системах замыканий бинарных отношений

На множестве  $P(A^2)$  всех бинарных отношений между элементами множества  $A$  следующие множества являются системами замыканий:

1.  $Z_r$  - множество всех рефлексивных бинарных отношений между элементами множества  $A$ ,
2.  $Z_s$  - множество всех симметричных бинарных отношений между элементами множества  $A$ ,
3.  $Z_t$  - множество всех транзитивных бинарных отношений между элементами множества  $A$ ,
4.  $Z_{eq} = Eq(A)$  - множество всех отношений эквивалентности на множестве  $A$ .

Множество  $Z_{as}$  всех антисимметричных бинарных отношений между элементами множества  $A$  не является системой замыкания.

### 3 Результаты работы

#### 3.1 Описание алгоритма классификации бинарных отношений

Из пункта 1.2 следует, что в нашем алгоритме будут определяться 5 свойств бинарных отношений, а именно:

1. рефлексивность,
2. антирефлексивность,
3. симметричность,
4. антисимметричность,
5. транзитивность.

Как по матрице представления определить свойства бинарного отношения:

1. Для того, чтобы бинарное отношение было *рефлексивным*, на главной диагонали должны стоять только единицы,
2. Для того, чтобы бинарное отношение было *антирефлексивным*, на главной диагонали должны стоять только нули,
3. Для того, чтобы бинарное отношение было *симметричным*, матрица представления должна равняться транспонированной матрице,
4. Для того, чтобы бинарное отношение было *антисимметричным*, в матрице должны отсутствовать единицы, симметричные относительно главной диагонали,
5. Для того, чтобы бинарное отношение было *транзитивным*, матрица, полученная перемножением матрицы самой на себя, должна являться частью исходной матрицы бинарного отношения.

Распишем алгоритмы проверки этих свойств:

##### 3.1.1 Алгоритм проверки рефлексивности. Функция *bo\_is\_reflexive*

*Вход:* Размерность матрицы  $N$  и матрица представления бинарного отношения размерности  $N \times N$

*Выход:* Если данное отношение рефлексивно, то  $res = 1$ . Иначе  $res = 0$ .

Шаг 1. Создается элемент  $res$ , равный 0. Запускается цикл *for*  $i$  от 0 до  $N$ ;

Шаг 2. Если элемент главной диагонали  $a[i][i]$  равен единице, то  $res$  становится равным 1. Иначе  $res = 0$ ;

Шаг 3. Если  $res$  хоть раз стал равен 0, то возвращается значение  $res$  (т.е. 0). Если  $res$  всегда был равен 1, то его значение выведется после завершения

цикла.

Временная сложность алгоритма определения рефлексивности =  $O(n)$

3.1.2 Алгоритм проверки антирефлексивности. Функция *bo\_is\_antireflexive*

*Вход:* Размерность матрицы  $N$  и матрица представления бинарного отношения размерности  $N \times N$

*Выход:* Если данное отношение антирефлексивно, то  $res = 1$ . Иначе  $res = 0$ .

Шаг 1. Создается элемент  $res$ , равный 0. Запускается цикл *for*  $i$  от 0 до  $N$ ;

Шаг 2. Если элемент главной диагонали  $a[i][i]$  равен нулю, то  $res$  становится равным 1. Иначе  $res = 0$ ;

Шаг 3. Если  $res$  хоть раз стал равен 1, то возвращается значение  $res$  (т.е. 0). Если  $res$  всегда был равен 0, то его значение возвратится после завершения цикла.

Временная сложность алгоритма определения антирефлексивности =  $O(n)$

3.1.3 Алгоритм проверки симметричности. Функция *bo\_is\_symmetric*

*Вход:* Размерность матрицы  $N$  и матрица представления бинарного отношения размерности  $N \times N$

*Выход:* Если данное отношение симметрично, то  $res = 1$ . Иначе  $res = 0$ .

Шаг 1. Создается элемент  $res$ , равный 0. Запускается цикл *for*  $i$  от 0 до  $N$  и в нем еще один цикл от  $j = i+1$  до  $N$ ;

Шаг 2. Если элемент матрицы  $a[i][j]$  равен элементу, стоящему на том же месте в транспонированной матрице  $a[j][i]$ , то  $res$  становится равным 1. Иначе  $res = 0$ ;

Шаг 3. Если  $res$  хоть раз стал равен 0, то возвращается значение  $res$  (т.е. 0). Если  $res$  всегда был равен 1, то его значение выведется после завершения циклов.

Временная сложность алгоритма определения симметричности =  $O(n^2/2)$   
=  $O(n^2)$

3.1.4 Алгоритм проверки антисимметричности. Функция *bo\_is\_antisymmetric*

*Вход:* Размерность матрицы  $N$  и матрица представления бинарного отношения размерности  $N \times N$

*Выход:* Если данное отношение симметрично, то  $res = 1$ . Иначе  $res = 0$ .



Шаг 1. Создается элемент  $res$ , равный 0. Запускается цикл *for*  $i$  от 0 до  $N$  и в нем еще один цикл с  $j$  от  $i+1$  до  $N$ ;

Шаг 2. Если элемент матрицы  $a[i][j]$  равен единице и элемент, стоящий на том же месте в транспонированной матрице  $a[j][i]$  (т.е. элемент, который симметричен элементу  $a[i][j]$  относительно главной диагонали), то проверяется равенство  $i = j$  (вдруг это элемент главной диагонали) и если элементы равны, то  $res$  становится равным 1. Иначе  $res = 0$ . Если элементы не равны единице, то  $res = 1$ ;

Шаг 3. Если  $res$  хоть раз стал равен 0, то возвращается значение  $res$  (т.е. 0). Если  $res$  всегда был равен 1, то его значение выведется после завершения циклов.

Временная сложность алгоритма определения антисимметричности  $= O(n^2/2)$   
 $= O(n^2)$

### 3.1.5 Алгоритм проверки транзитивности. Функция *bo\_is\_transitive*

*Вход:* Размерность матрицы  $N$  и матрица представления бинарного отношения размерности  $N \times N$

*Выход:* Если данное отношение транзитивно, то  $res = 1$ . Иначе  $res = 0$ .

Шаг 1. Создается элемент  $res$ , равный 0. Запускается цикл *for*  $i$  от 0 до  $N$ , в нем цикл с  $j$  от 0 до  $N$  и в нем еще один цикл с  $k$  от 0 до  $N$ ;

Шаг 2. Если произведение элементов матрицы  $a[i][k]$  и  $a[k][j]$  меньше либо равно элементу  $a[i][j]$  (т.е. перемноженная матрица является частью исходной матрицы), то  $res$  становится равным 1. Иначе  $res = 0$ ;

Шаг 3. Если  $res$  хоть раз стал равен 0, то возвращается значение  $res$  (т.е. 0). Если  $res$  всегда был равен 1, то его значение выведется после завершения циклов.

После проверки бинарного отношения на эти 5 свойств, мы можем судить, к какому типу отношений относится данное бинарное отношение. В этом и заключается алгоритм классификации. Эти проверки проводятся в функции *bo\_result*.

Временная сложность алгоритма определения транзитивности  $= O(n^3)$

1. Бинарное отношение является отношением эквивалентности, если выполнялись следующие три свойства: рефлексивность, симметричность и транзитивность.

2. Если выполнены свойства рефлексивности и транзитивности, то это отношение является отношением квазипорядка,
3. Бинарное отношение является отношением порядка, если оно рефлексивно, антисимметрично и транзитивно.

3.1.6 Алгоритм проверки отношения на эквивалентность, квазипорядок и порядок

*Вход:* Размерность матрицы  $N$  и матрица представления бинарного отношения размерности  $N \times N$

*Выход:* «Данное отношение является отношением эквивалентности» или «Данное отношение является отношением квазипорядка» или «Данное отношение является отношением порядка».

Шаг 1. С помощью написанных выше алгоритмов проверяются свойства заданного отношения. Результаты вносятся в переменные  $res\_refl, res\_antirefl, res\_sim, res\_antisim, res\_tranz$  (соответственно: рефлексивность, антирефлексивность, симметричность, антисимметричность и транзитивность).

Шаг 2. Если выполняются свойства рефлексивности и транзитивности, т.е.  $res\_refl = 1$  и  $res\_tranz = 1$ , то выводится «Данное отношение является отношением квазипорядка» и проверяется свойство симметричности. Если оно выполняется, т.е.  $res\_sim = 1$ , то выводится «Данное отношение является отношением эквивалентности».

Шаг 3. Если выполняются свойства рефлексивности, антисимметричности и транзитивности, т.е.  $res\_refl = 1, res\_antisim = 1$  и  $res\_tranz = 1$ , то выводится «Данное отношение является отношением порядка».

Временная сложность алгоритма определения отношения эквивалентности  $= O(n^3 + n^2/2 + n) = O(n^3)$

Временная сложность алгоритма определения отношения квазипорядка  $= O(n^3 + n^2/2 + n) = O(n^3)$

Временная сложность алгоритма определения отношения порядка  $= O(n^3 + n) = O(n^3)$

## 3.2 Описание алгоритма построения основных замыканий бинарных отношений

1. Матрица *рефлексивного* замыкания равна  $R \cup E_n$ , т.е. необходимо все элементы главной диагонали заменить единицами,

2. Матрица *симметричного* замыкания равна  $R \cup R^T$ , т.е. если элемент матрицы равен единице, то симметричный ему элемент относительно главной диагонали тоже должен быть равен единице,
3. Стратегия построения матрицы *транзитивного* замыкания такова: если в отношении имеется две пары элементов (j, k) и (k, d), то необходимо добавить пару (j, d). После этого надо запустить цикл еще раз, т.к. после добавления новой пары этому условию могут удовлетворять еще две пары.

3.2.1 Алгоритм построения замыкания относительно свойства рефлексивности

*Вход:* Размерность матрицы N и матрица представления бинарного отношения размерности  $N \times N$

*Выход:* Матрица построенного замыкания относительно свойства рефлексивности размерности  $N \times N$ .

Шаг 1. Запускается цикл *for* i от 0 до N и в нем каждому элементу главной диагонали  $a[i][i]$  присваивается единица.

Временная сложность алгоритма определения построения замыкания рефлексивности =  $O(n)$

3.2.2 Алгоритм построения замыкания относительно свойства симметричности

*Вход:* Размерность матрицы N и матрица представления бинарного отношения размерности  $N \times N$

*Выход:* Матрица построенного замыкания относительно свойства симметричности размерности  $N \times N$ .

Шаг 1. Запускается цикл *for* i от 0 до N, в нем запускается еще один цикл с j от 0 до N и в нем все элементы  $a[i][j]$  проверяются на равенство единице. Если элемент равен единице, то симметричный ему элемент  $a[j][i]$  тоже становится равен единице.

Временная сложность алгоритма определения построения замыкания симметричности =  $O(n^2)$

### 3.2.3 Алгоритм построения замыкания относительно свойства транзитивности

*Вход:* Размерность матрицы  $N$  и матрица представления бинарного отношения размерности  $N \times N$

*Выход:* Матрица построенного замыкания относительно свойства транзитивности размерности  $N \times N$ .

Шаг 1. В данном алгоритме необходимо запустить цикл *for* 4 раза, т.к. после добавления новой пары в заданное отношение (в ходе построения замыкания), другие пары могут образовать условие для добавления еще одной пары. Перед последним циклом проверяется равенство элемента  $a[j][k]$  единице. Если равенство выполняется, то запускается цикл *for* с  $d$  от 0 до  $N$ .

Шаг 2. В этом цикле идет проверка на равенство единице элемента  $a[k][d]$ . Если условие выполняется, то по свойству транзитивности в отношение надо добавить пару  $(j, d)$ , т.е. в матрице размерности элементу  $a[j][d]$  присвоить единицу.

Временная сложность алгоритма определения построения замыкания транзитивности  $= O(n^4)$

### 3.2.4 Алгоритм построения замыкания относительно эквивалентности

*Вход:* Размерность матрицы  $N$  и матрица представления бинарного отношения размерности  $N \times N$

*Выход:* Матрица построенного замыкания относительно эквивалентности размерности  $N \times N$ .

Шаг 1. Построение замыкания относительно свойства рефлексивности.

Шаг 2. Построение замыкания относительно свойства симметричности.

Шаг 3. Построение замыкания относительно свойства транзитивности.

Временная сложность алгоритма определения построения замыкания эквивалентности  $= O(n^4 + n^2 + n) = O(n^4)$

## 3.3 Код программы

```
#include <iostream>

using namespace std;
int brk = 0;
```

```

bool bo_is_reflexive(int N, int** a)
{
    int res = 0;

    for (int i = 0; i < N; ++i)
    {
        if (a[i][i] == 1)
            res = 1;
        else res = 0;

        if (res == 0)
        {
            return res;
        }
    }

    return res;
}

bool bo_is_antireflexive(int N, int** a)
{
    int res = 0;

    for (int i = 0; i < N; ++i)
    {
        if (a[i][i] == 0)
            res = 1;
        else res = 0;

        if (res == 0)
        {
            return res;
        }
    }
}

```

```

return  res;
}

bool bo_is_symmetric(int N, int** a)
{
    int res = 0;

    for (int i = 0; i < N; ++i)
    {
        for (int j = i + 1; j < N; ++j)
        {
            if (a[i][j] == a[j][i])
                res = 1;
            else res = 0;

            if (res == 0)
            {
                return  res;
            }
        }
    }

    return  res;
}

bool bo_is_antisymmetric(int N, int** a)
{
    int res = 0;

    for (int i = 0; i < N; ++i)
    {
        for (int j = i + 1; j < N; ++j)
        {

```

```

if (a[i][j] == 1 && a[j][i] == 1) {
    if (i == j)
        res = 1;
    else res = 0;
}
else res = 1;

if (res == 0)
{
    return res;
}
}
}

return res;
}

bool bo_is_transitive(int N, int** a)
{
    int res = 0;

    for (int i = 0; i < N; ++i)
    {
        for (int j = 0; j < N; ++j)
        {
            for (int k = 0; k < N; ++k)
            {
                if (a[i][j] >= a[i][k] * a[k][j])
                    res = 1;
            }
        }
    }

    if (res == 0)
    {
        return res;
    }
}

```

```

}
}
}
}

return res;
}

//строим замыкания
void z_reflexive(int N, int** a)
{
for (int i = 0; i < N; i++)
{
a[i][i] = 1;
}
}

void z_sim(int N, int** a)
{
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
if (a[i][j] == 1)
a[j][i] = 1;
}
}
}

void z_tranz(int N, int** a)
{
for (int i = 0; i < N; i++)
for (int j = 0; j < N; j++)
for (int k = 0; k < N; k++)

```



```

if (a[j][k] == 1)
for (int d = 0; d < N; d++)
if (a[k][d] == 1)
a[j][d] = 1;

}

```

```

void z_build(int N, int** a, int vvod)
{
int** z_a;
z_a = new int* [N];
for (int i = 0; i < N; i++) {
z_a[i] = new int[N];
for (int j = 0; j < N; j++) {
z_a[i][j] = a[i][j];
}
}
switch (vvod)
{
case 1:
z_reflexive(N, z_a);
break;
case 2:
z_sim(N, z_a);
break;
case 3:
z_tranz(N, z_a);
break;
case 4:
z_reflexive(N, z_a);
z_sim(N, z_a);
z_tranz(N, z_a);
break;
}
}

```

```

case 5:
brk = 1;
break;
default:
cout << "Error" << endl;
break;
}
if (brk == 0) {
cout << "Построенное замыкание:" << endl;
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++)
cout << z_a[i][j] << ' ';
cout << endl;
}
}
}

void    bo_result(int N, int** a)
{
cout << "Введеная матрица:" << endl;
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++)
cout << a[i][j] << ' ';
cout << endl;
}

int res_refl = bo_is_reflexive(N, a);
int res_antirefl = bo_is_antireflexive(N, a);
int res_simm = bo_is_symmetric(N, a);
int res_antisimm = bo_is_antisymmetric(N, a);
int res_tranz = bo_is_transitive(N, a);
cout << "Результаты (1 - да, 0 - нет):" << endl;
cout << "Рефлексивность:" << res_refl << endl;
cout << "Антирефлексивность:" << res_antirefl << endl;
cout << "Симметричность:" << res_simm << endl;

```

```

cout << "Антисимметричность:" << res_antisimm << endl;
cout << "Транзитивность:" << res_tranz << endl;

if (res_refl == 1 && res_tranz == 1) {
cout << "Данное отношение является отношением квазипорядка" << endl;
if (res_simm == 1)
cout << "Данное отношение является отношением эквивалентности" << endl;
}
if (res_refl == 1 && res_antisimm == 1 && res_tranz == 1) {
cout << "Данное отношение является отношением порядка" << endl;
}

int vvod;
cout << "Введите, какое замыкание требуется построить:" << endl;
cout << "1 - рефлексивное" << endl << "2 - симметричное" << endl
<< "3 - транзитивное" << endl << "4 - эквивалентное"
<< endl << "5 - не строить никакое замыкание" << endl;
while (brk == 0) {
cout << "Введите номер:" << endl;
cin >> vvod;
z_build(N, a, vvod);
}
}

int main()
{
setlocale(LC_ALL, "Rus");

int sposob, i, j, N;
cout << "Введите способ ввода (1 - поэлементно, 2 - построчно): ";
cin >> sposob;
cout << "Введите размерность матрицы бинарного отношения: ";
cin >> N;
int** a;
a = new int* [N];

```

```

cout << "Введите матрицу A" << endl;
if (sposob == 1) {
for (i = 0; i < N; i++) {
a[i] = new int[N];
for (j = 0; j < N; j++) {
cout << "A["
<< i
<< "]["
<< j
<< "] = ";
cin >> a[i][j];
}
}
}
else
{
for (int i = 0; i < N; i++) {
a[i] = new int[N];
for (int j = 0; j < N; j++) {
cin >> a[i][j];
}
}
}

cout << endl;
bo_result(N, a);
cout << endl;
}

}

```

### 3.4 Результаты тестирования программ

Тестирование №1:

На вход поступает матрица:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Она обладает свойством симметричности.

Построим рефлексивное замыкание.

```

Консоль отладки Microsoft Visual Studio
Введите способ ввода (1 - поэлементно, 2 - построчно): 2
Введите размерность матрицы бинарного отношения: 5
Введите матрицу A
1 1 1 0 0
1 1 1 1 1
1 1 1 0 0
0 1 0 0 0
0 1 0 0 0

Введенная матрица:
1 1 1 0 0
1 1 1 1 1
1 1 1 0 0
0 1 0 0 0
0 1 0 0 0

Результаты (1 - да, 0 - нет):
Рефлексивность:0
Антирефлексивность:0
Симметричность:1
Антисимметричность:0
Транзитивность:0
Введите, какое замыкание требуется построить:
1 - рефлексивное
2 - симметричное
3 - транзитивное
4 - эквивалентное
5 - не строить никакое замыкание
Введите номер:
1
Построенное замыкание:
1 1 1 0 0
1 1 1 1 1
1 1 1 0 0
0 1 0 1 0
0 1 0 0 1
Введите номер:
5

```

Рисунок 1 – Тестирование №1

Тестирование №2:

На вход поступает матрица:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Она обладает свойством антирефлексивности, антисимметричности и транзитивности.

Построим все типы замыканий.

```
C:\Users\Shamil_\source\repos\test_lab1\Debug\test_lab1.exe
Введите способ ввода (1 - поэлементно, 2 - построчно): 2
Введите размерность матрицы бинарного отношения: 5
Введите матрицу A
0 0 0 0 0
1 0 0 0 0
1 1 0 0 0
1 1 0 0 0
1 1 1 0 0

Введенная матрица:
0 0 0 0 0
1 0 0 0 0
1 1 0 0 0
1 1 0 0 0
1 1 1 0 0

Результаты (1 - да, 0 - нет):
Рефлексивность:0
Антирефлексивность:1
Симметричность:0
Антисимметричность:1
Транзитивность:1
Введите, какое замыкание требуется построить:
1 - рефлексивное
2 - симметричное
3 - транзитивное
4 - эквивалентное
5 - не строить никакое замыкание
Введите номер:
1
Построенное замыкание:
1 0 0 0 0
1 1 0 0 0
1 1 1 0 0
1 1 0 1 0
1 1 1 0 1
Введите номер:
2
Построенное замыкание:
0 1 1 1 1
1 0 1 1 1
1 1 0 0 1
1 1 0 0 0
1 1 1 0 0
Введите номер:
3
Построенное замыкание:
0 0 0 0 0
1 0 0 0 0
1 1 0 0 0
1 1 0 0 0
1 1 1 0 0
Введите номер:
4
Построенное замыкание:
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

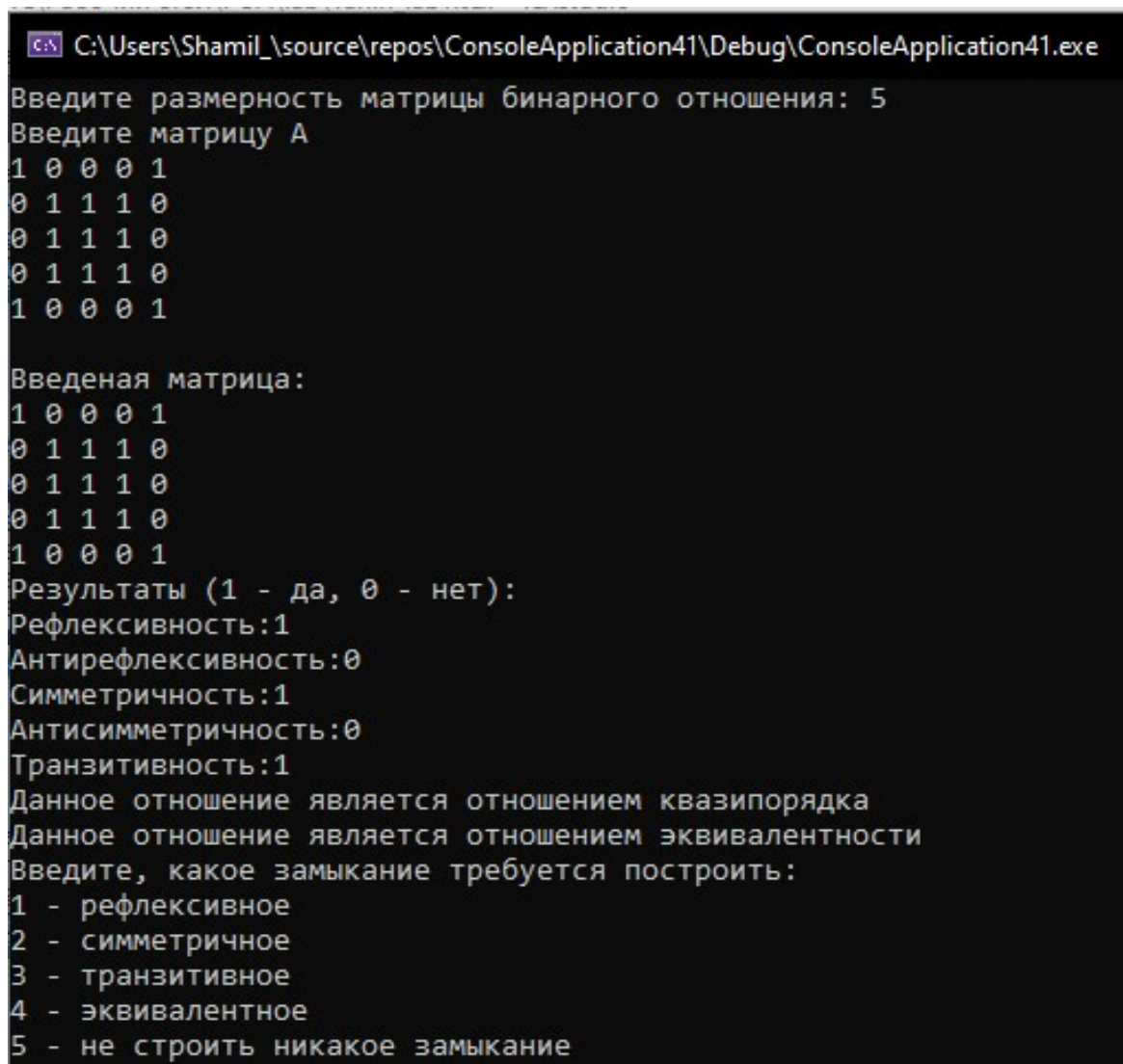
Рисунок 2 – Тестирование №2

### Тестирование №3:

На вход поступает матрица:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Она обладает свойством рефлексивности, симметричности и транзитивности, а значит является отношением квазипорядка и отношением эквивалентности.



```
C:\Users\Shamil_\source\repos\ConsoleApplication41\Debug\ConsoleApplication41.exe
Введите размерность матрицы бинарного отношения: 5
Введите матрицу A
1 0 0 0 1
0 1 1 1 0
0 1 1 1 0
0 1 1 1 0
1 0 0 0 1

Введенная матрица:
1 0 0 0 1
0 1 1 1 0
0 1 1 1 0
0 1 1 1 0
1 0 0 0 1
Результаты (1 - да, 0 - нет):
Рефлексивность:1
Антирефлексивность:0
Симметричность:1
Антисимметричность:0
Транзитивность:1
Данное отношение является отношением квазипорядка
Данное отношение является отношением эквивалентности
Введите, какое замыкание требуется построить:
1 - рефлексивное
2 - симметричное
3 - транзитивное
4 - эквивалентное
5 - не строить никакое замыкание
```

Рисунок 3 – Тестирование №3



## **ЗАКЛЮЧЕНИЕ**

В данной лабораторной работе были рассмотрены и изучены следующие темы: основные определения видов бинарных отношений, свойства бинарных отношений и основные системы замыкания на множестве бинарных отношений. В третьей части работы были реализованы алгоритмы классификации бинарных отношений и построения основных замыканий бинарных отношений.