МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. Шухова»
(БГТУ им. В. Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №3.1

по дисциплине: «Дискретная математика»

по теме: **Отношения и их свойства**

Выполнил/а: ст. группы ПВ-231

Чупахина София Александровна

Проверил: Рязанов Юрий Дмитриевич

Белгород, 2024

Содержание

Часть 1. Операции над отношениями

Вариант 6

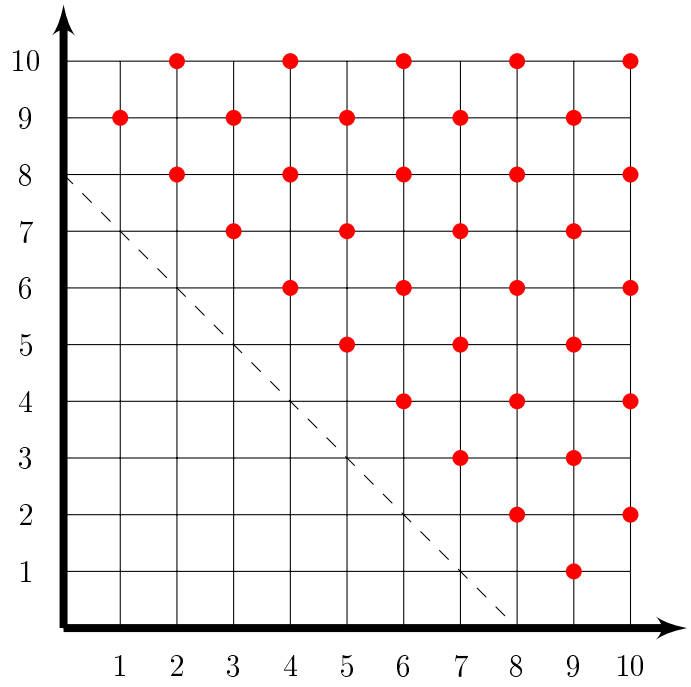
а) $A = \{(x, y) \mid x + y$ — четно и $x + y > 8\}$
$B = \{(x, y) \mid x + 2y > 20\}$
$C = \{(x, y) \mid (x, y) \in \{1, 2, 4, 8\} \times \{3, 5, 7, 10\}\}$

б) D $= A \cap B^{-1} \triangle AoBoC$

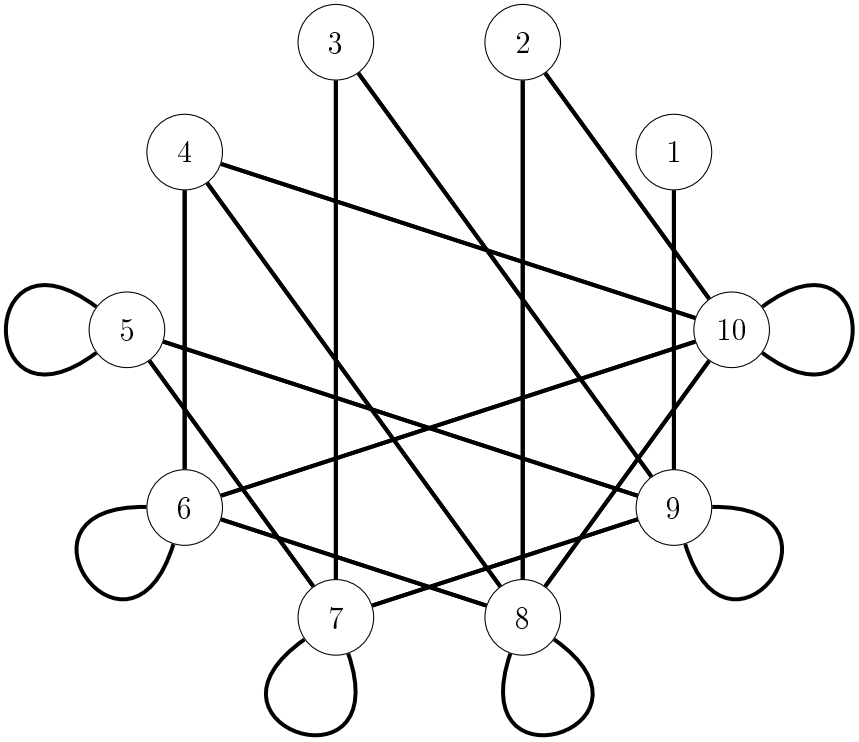
Задание 1.1

Текст задания: представить заданные на множестве $M = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ отношения A, B, C графиком, графом и матрицей.

Начнем с отношения A. На графике (первой четверти координатной плоскости с координатами, не превышающими 10) элементам отношения будут соответствовать точки, находящиеся строго выше линии $y = 8 - x$ (для удобства обозначим ее на графике пунктиром), сумма координат которых при этом четна.



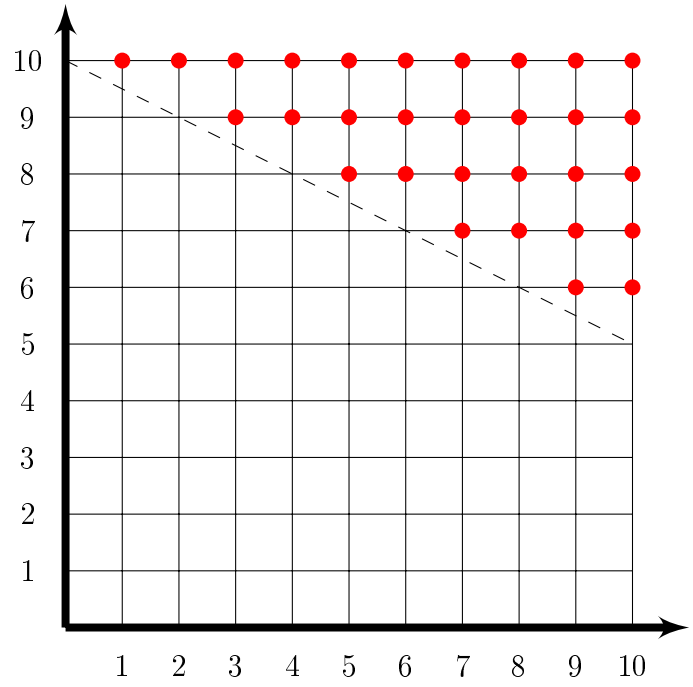
С помощью графа это отношение можно представить следующим образом. Отдельно можно заметить, что, поскольку вхождение или невхождение элемента в отношение определяется только суммой координат, при соответствующих условиям x и y элементом отношения будет являться как (x, y), так и (y, x), а потому граф состоит только из ребер.



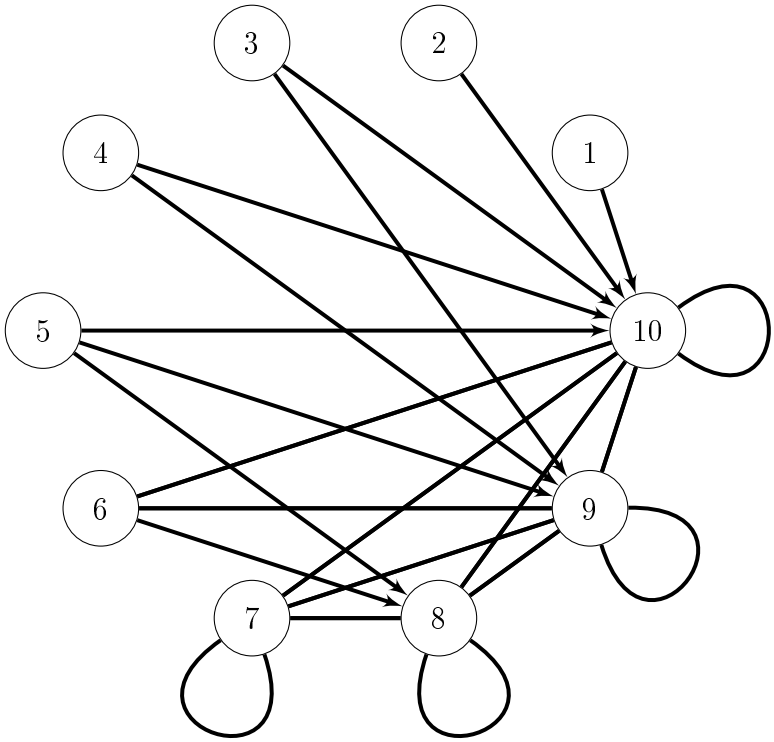
В виде матрицы же отношение A будет представлено следующим образом. Тут можно заметить, что единицы в матрице не совпадают с расположением точек графика: чтобы они совпали, график нужно мысленно повернуть на 90° по часовой стрелке:

0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	1	0	1
0	0	0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1	0	1
0	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1

Перейдем к отношению B. На графике элементам отношения будут соответствовать точки, находящиеся строго выше линии $y = 10 - 0.5x$ (для удобства также обозначим ее на графике пунктиром).



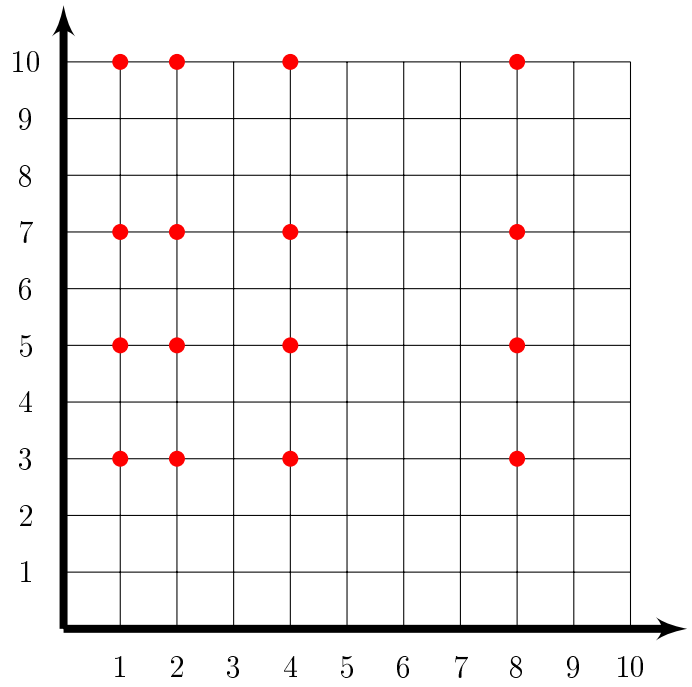
С помощью графа это отношение можно представить следующим образом. В отличие от предыдущего отношения, на этом графе присутствуют как ребра, так и дуги.



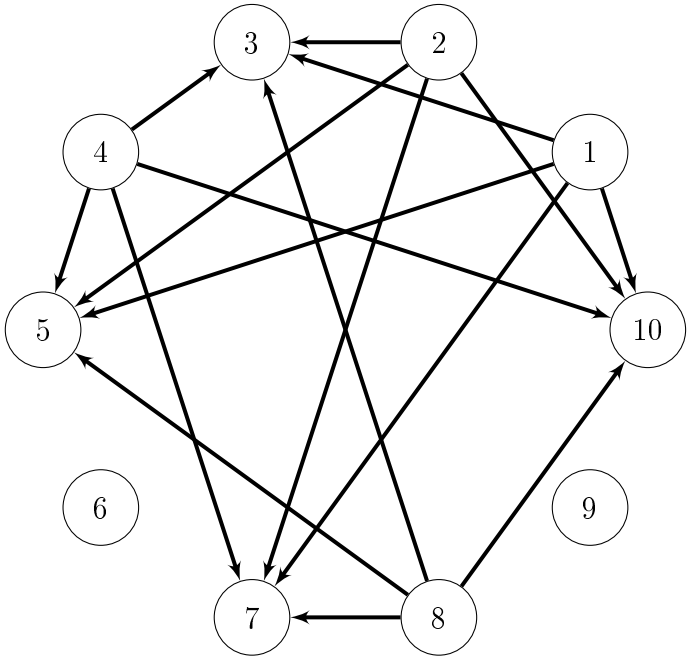
Матрица, представляющая это отношение, будет выглядеть следующим образом:

0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1

Наконец, перейдем к отношению C. Оно задается декартовым произведением двух множеств, поэтому его элементам соответствуют точки, для которых $x \in \{1, 2, 4, 8\}$ и $y \in \{3, 5, 7, 10\}$.



Ему соответствует следующий граф (который, кстати, не имеет ребер вовсе).



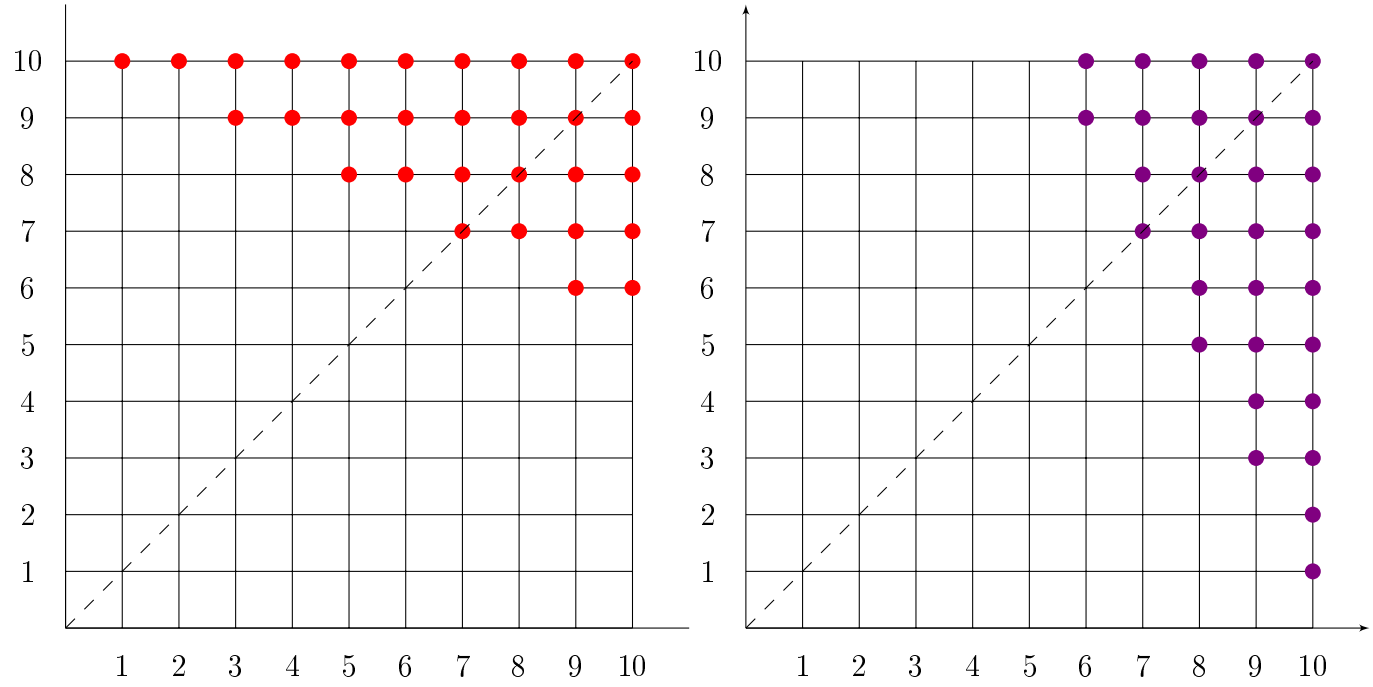
И матрица этого соотношения будет иметь вид:

0	0	1	0	1	0	1	0	0	1
0	0	1	0	1	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

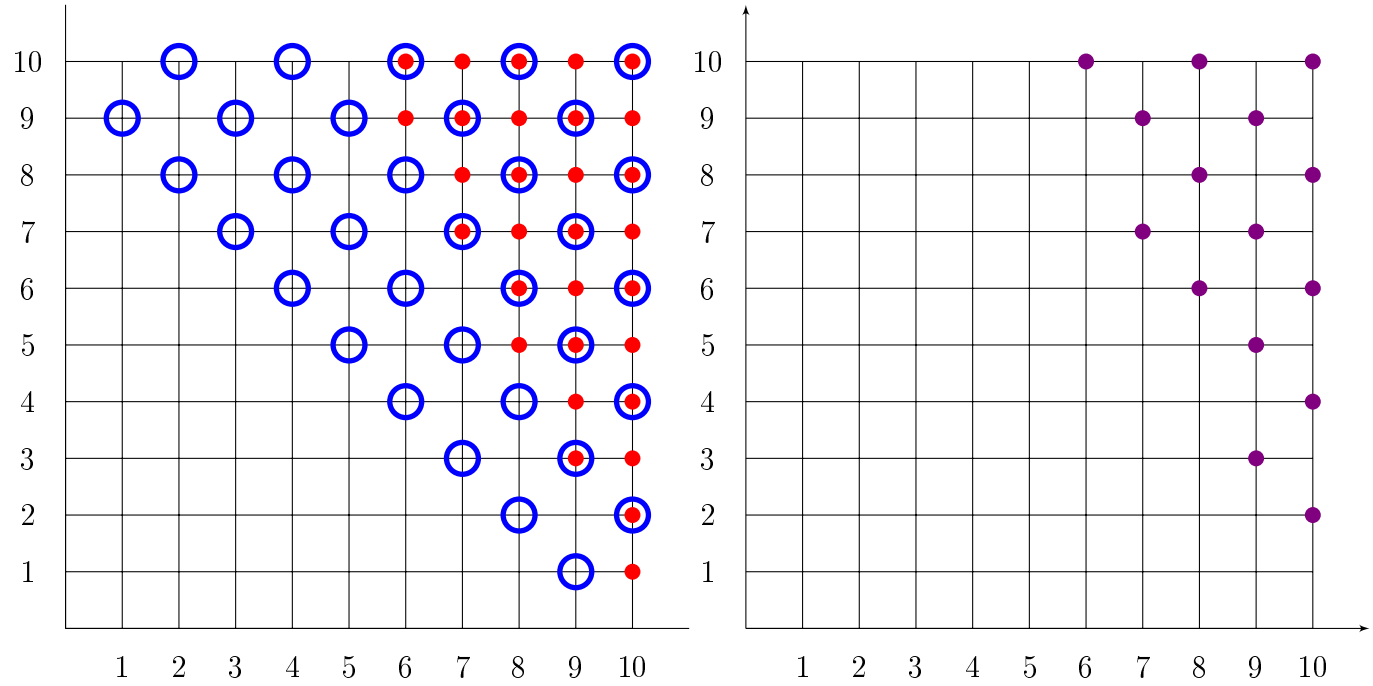
Задание 1.2

Текст задания: Вычислить значение выражения («Варианты заданий», пункт **б)**) при заданных отношениях («Варианты заданий», пункт **а)**).

Чтобы получить отношение B^{-1}, нужно поменять местами элементы x и y в каждой паре. Если мы говорим о представлении отношения с помощью графика, то графиком отношения B^{-1} будет график отношения B, зеркально отраженный по оси $y = x$ (отметим эту ось пунктиром). Изобразим слева график исходного отношения B, а справа — его отраженный график, то есть график отношения B^{-1}.



Перейдем к следующему действию, $A \cup B^{-1}$, где нам нужно произвести операцию пересечения. Результат ее также представим графически. На графике слева отобразим красными точками элементы отношения B^{-1} и синими окружностями элементы отношения A; на графике справа фиолетовыми точками отметим элементы, входящие в оба отношения.



Перейдем к следующему действию, операции композиции AoC. Здесь уже не станем выполнять ее графически, попробуем выполнить цепочку рассуждений. Посмотрев на графики отношения A и C, приведенные в данной работе, можем увидеть, что:

- В отношении C не пусты только образы следующих элементов: 1, 2, 4, 8. В образы этих элементов, в свою очередь, входят одни и те же элементы 3, 5, 7, 10. Далее мы рассмотрим

отношение A и то, не пусты ли в нем прообразы тех элементов, которые имеют образы в отношении C.

- В отношении A прообраз элемента 1 не пустой — в него входит элемент 9. То есть в отношение — результат композиции AoC войдут пары, принадлежащие декартову произведению $\{9\} \times \{3, 5, 7, 10\}$.

- Прообраз элемента 2 в отношении A также не пустой — в него входят элементы 8, 10. То есть в отношение — результат композиции AoC войдут пары, принадлежащие декартову произведению $\{8, 10\} \times \{3, 5, 7, 10\}$.

- Прообраз элемента 4 в отношении A не пустой — в него входят элементы 6, 8, 10. То есть в отношение — результат композиции AoC войдут пары, принадлежащие декартову произведению $\{6, 8, 10\} \times \{3, 5, 7, 10\}$.

- Прообраз элемента 8 в отношении A не пустой — в него входят элементы 2, 4, 6, 8, 10. То есть в отношение — результат композиции AoC войдут пары, принадлежащие декартову произведению $\{2, 4, 6, 8, 10\} \times \{3, 5, 7, 10\}$.

- Вполне очевидно, что декартовы произведения $\{8, 10\} \times \{3, 5, 7, 10\}$ и $\{6, 8, 10\} \times \{3, 5, 7, 10\}$ будут подмножествами декартова произведения $\{2, 4, 6, 8, 10\} \times \{3, 5, 7, 10\}$. Поскольку второй множитель этого произведения такой же, как у декартова произведения $\{9\} \times \{3, 5, 7, 10\}$, их можно объединить. В итоге мы можем утверждать, что результатом композиции AoC будет соответствие $\{2, 4, 6, 8, 9, 10\} \times \{3, 5, 7, 10\}$.

Рассуждая аналогичным образом, хотя и слегка изменив порядок рассуждений, проведем композицию $(AoC)oB$.

- В отношении AoC не пусты только прообразы следующих элементов: 3, 5, 7, 10. В прообразы этих элементов, в свою очередь, входят одни и те же элементы 2, 4, 6, 8, 9, 10. Далее мы рассмотрим отношение B и то, не пусты ли в нем образы тех элементов, которые имеют прообразы в отношении AoC.

- В отношении B образ элемента 3 не пустой — в него входят элементы 9, 10. То есть в отношение — результат композиции $AoCoB$ войдут пары, принадлежащие декартову произведению $\{2, 4, 6, 8, 9, 10\} \times \{9, 10\}$.

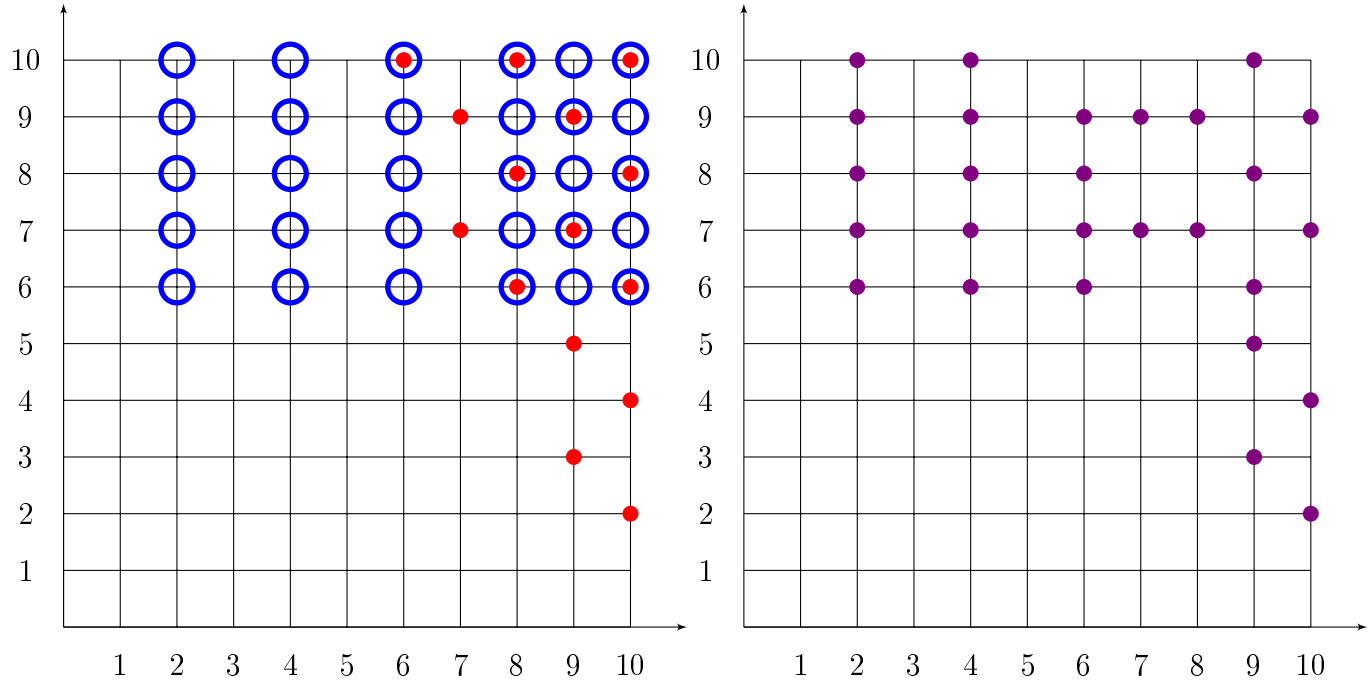
- Образ элемента 5 в отношении B также не пустой — в него входят элементы 8, 9, 10. То есть в отношение — результат композиции $AoCoB$ войдут пары, принадлежащие декартову произведению $\{2, 4, 6, 8, 9, 10\} \times \{8, 9, 10\}$.

- Образ элемента 7 в отношении B не пустой — в него входят элементы 7, 8, 9, 10. То есть в отношение — результат композиции $AoCoB$ войдут пары, принадлежащие декартову произведению $\{2, 4, 6, 8, 9, 10\} \times \{7, 8, 9, 10\}$.

- Образ элемента 10 в отношении B не пустой — в него входят элементы 6, 7, 8, 9, 10. То есть в отношение — результат композиции $AoCoB$ войдут пары, принадлежащие декартову произведению $\{2, 4, 6, 8, 9, 10\} \times \{6, 7, 8, 9, 10\}$.

- Очевидно, что все упомянутые выше декартовы произведения будут подмножествами декартова произведения $\{2, 4, 6, 8, 9, 10\} \times \{6, 7, 8, 9, 10\}$. Порожденное им множество пар и будет являться результатом композиции $AoCoB$.

Наконец, осталось провести последнюю операцию, симметрическую разность отношений $A \cap B^{-1} \triangle (AoCoB)$. На графике слева отобразим красными точками элементы отношения $A \cap B^{-1}$ и синими окружностями элементы отношения $AoCoB$; на графике справа фиолетовыми точками отметим элементы, входящие только в одно из отношений.



Задание 1.3

Текст задания: Написать программы, формирующие матрицы заданных отношений («Варианты заданий», пункт **а)**).

Первым делом для выполнения этого задания потребуется создать структуру, которая отображала бы отношение, и написать для работы с ней несколько базовых функций, позволяющих добавлять и удалять её элементы. Будем представлять отношение как матрицу, элемент которой в строке i и столбце j равен 1, если в отношение входит пара (i, j), и 0, если такая пара в отношение не входит. Можно было бы представить матрицу в памяти компьютера как массив указателей на массивы булевых значений (каждый массив является отдельной строкой), но в целях оптимизации (и учитывая тот факт, что мы не собираемся работать с отношениями на множествах большой мощности) мы представим матрицу как массив целых чисел, где каждое число представляет строку матрицы — ведь его можно записать в двоичном виде как последовательность нулей и единиц. Таким образом, j-тый разряд числа, являющегося i-тым элементом массива, показывает, входит ли в отношение пара (i, j), а получать доступ к разрядам числа будем, используя побитовые операции.

Примечание: в матрицах нумерация столбцов и строк идет с единицы, что будет учитываться в функциях доступа к элементу.

```c
#include <stdio.h>
#include <stdbool.h>
#include <malloc.h>
#include <assert.h>

#ifndef BIN_REL_DEF_IO
#define BIN_REL_DEF_IO

int max2 (int x, int y) {
    return (x > y) ? x : y;
}

//Структура, отображающая отношение на множестве {1, ..., max_value};
//Какие упорядоченные пары присутствуют в этом отношении, показывает
   массив целых чисел values; целые числа в двоичной записи
   представляют собой строки матрицы, отображающей отношение
typedef struct {
    int max_value;
    int *values;
} bin_relation;

//Возвращает пустое отношение на множестве {1, ..., max_value},
   проверяя при этом, что max_value не превышает 32 количество(
   двоичных разрядов в типе int)
bin_relation bin_relation_createEmpty(int max_value) {
    if (max_value > 32) {
        fprintf(stderr, "Too large power of set");
        exit(1);
    }
    int *values = (int*) malloc (max_value * sizeof(int));
    for (int i = 0; i < max_value; i++) {
        values[i] = 0;
    }
    return (bin_relation) {max_value, values};
}

//возвращает 1, если упорядоченная пара (x, y) входит в множество a, и
   0 в противном случае
bool bin_relation_getValue(bin_relation a, int x, int y) {
    return (a.values[x-1] >> (y-1)) & 1;
}

//Меняет ячейку матрицы отношения a, соответствующую паре (x, y), на
   значение value
void bin_relation_changeValue(bin_relation *a, int x, int y, bool
   value) {
    if (value) {
```

```
        a->values[x-1] |= 1 << (y-1);
    } else {
        a->values[x-1] &= ~(1 << (y-1));
    }
}

//Осуществляет ввод матрицы отношения по адресу a
void bin_relation_input(bin_relation *a) {
    for (int i = 1; i <= a->max_value; i++) {
        for (int j = 1; j <= a->max_value; j++) {
            int cur_value;
            scanf("%d", &cur_value);
            assert(cur_value == 1 || cur_value == 0);
            bin_relation_changeValue(a, i, j, cur_value);
        }
    }
}

//Осуществляет вывод матрицы отношения a
void bin_relation_matrixPrint(bin_relation a) {
    for (int i = 1; i <= a.max_value; i++) {
        for (int j = 1; j <= a.max_value; j++) {
            printf("%d ", bin_relation_getValue(a, i, j));
        }
        printf("\n");
    }
}

//Осуществляет вывод упорядоченных пар, входящих в отношение a
void bin_relation_pairPrint(bin_relation a) {
    for (int i = 1; i <= a.max_value; i++) {
        bool are_pairs_for_cur_i = false;
        for (int j = 1; j <= a.max_value; j++) {
            if (bin_relation_getValue(a, i, j)) {
                are_pairs_for_cur_i = true;
                printf("(%d, %d); ", i, j);
            }
        }
        if (are_pairs_for_cur_i) {
            printf("\n");
        }
    }
}
#endif
```

../bin_relations/bin_relation_definition_input_output.c

Используя созданную структуру и функции к ней, напишем три функции без аргументов,

возвращающие соответственно заданные условием отношения A, B, C.

```c
#include "bin_relation_definition_input_output.c"

//Возвращает отношение A, заданное условием варианта 6 лабораторной работы 3.1 см(. Варианты" заданий", пункт а))
bin_relation relation_A_variant6 () {
    bin_relation A = bin_relation_createEmpty(10);
    for (int i = 1; i <= 10; i++) {
        for (int j = 1; j <= 10; j++) {
            if (i + j > 8 && (i + j) % 2 == 0) {
                bin_relation_changeValue(&A, i, j, 1);
            }
        }
    }
    return A;
}

//Возвращает отношение B, заданное условием варианта 6 лабораторной работы 3.1 см(. Варианты" заданий", пункт а))
bin_relation relation_B_variant6 () {
    bin_relation B = bin_relation_createEmpty(10);
    for (int i = 1; i <= 10; i++) {
        for (int j = 1; j <= 10; j++) {
            if (i + 2*j > 20) {
                bin_relation_changeValue(&B, i, j, 1);
            }
        }
    }
    return B;
}

//Возвращает отношение C, заданное условием варианта 6 лабораторной работы 3.1 см(. Варианты" заданий", пункт а))
bin_relation relation_C_variant6 () {
    bin_relation C = bin_relation_createEmpty(10);
    int c1[4] = {1, 2, 4, 8};
    int c2[4] = {3, 5, 7, 10};
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            bin_relation_changeValue(&C, c1[i], c2[j], 1);
        }
    }
    return C;
}
```

../bin_relations/ABC_forming.c

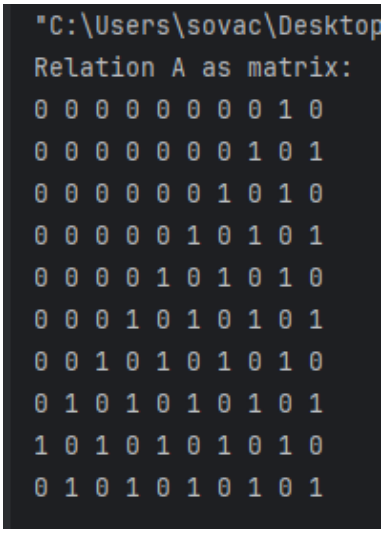
Для вывода матриц этих отношений на экран функция main должна иметь следующий вид:

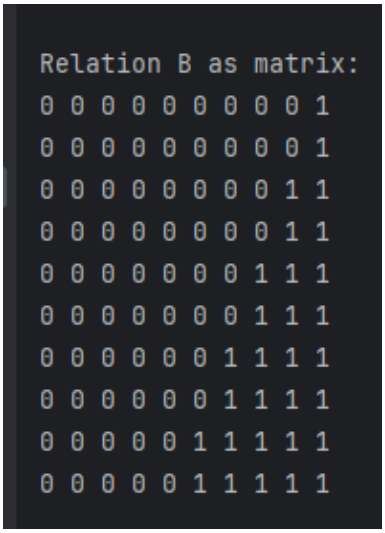
```c
#include <stdio.h>
#include "bin_relation_definition_input_output.c"
#include "ABC_forming.c"


int main() {
    bin_relation A = relation_A_variant6();
    bin_relation B = relation_B_variant6();
    bin_relation C = relation_C_variant6();

    printf("Relation A as matrix: \n");
    bin_relation_matrixPrint(A);
    printf("\nRelation B as matrix: \n");
    bin_relation_matrixPrint(B);
    printf("\nRelation C as matrix: \n");
    bin_relation_matrixPrint(C);
}
```

../bin_relations/main1.c





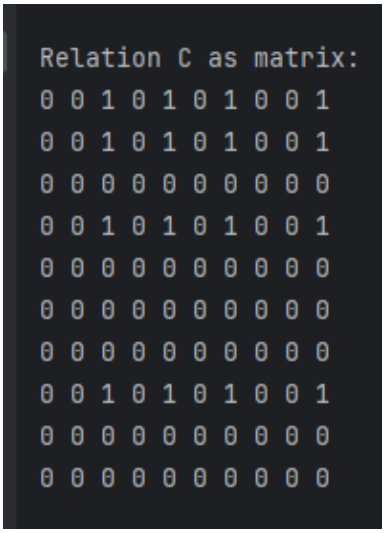


Рис. 1: Матрица отношения A Рис. 2: Матрица отношения B Рис. 3: Матрица отношения C

Легко убедиться, что матрицы отношений A, B, C аналогичны матрицам, которые мы составили в задании 1.1.

Задание 1.4

Текст задания: Программно реализовать операции над отношениями.

Напишем функции, реализующие операции сравнения отношений (проверка на включение, равенство, строгое включение), операции, аналогичные операциям над множествами (пересечение, объединение, разность, симметрическая разность, дополнение) и операции, не определенные для множеств (обращение, композиция, степень отношения). Здесь стоит заметить, что, поскольку строки бинарной матрицы представлены целыми числами, с помощью побитовых операций их можно в одно действие сравнить, поняв, какие элементы встречаются только в одной, в обеих или ни в одной из матриц. Таким образом, многие операции можно реализовать без использования вложенных циклов.

```c
#include <stdbool.h>
#include "bin_relation_definition_input_output.c"

#ifndef BIN_REL_OPERATIONS
#define BIN_REL_OPERATIONS

//Возвращает 1, если отношение a включено в отношение b, и 0 в противном случае
bool bin_relation_inclusion(bin_relation a, bin_relation b) {
    bool is_inclusion = true;
    for (int i = 0; i < max2(a.max_value, b.max_value) && is_inclusion; i++) {
        is_inclusion = (b.values[i] | a.values[i]) == b.values[i];
    }
    return is_inclusion;
}

//Возвращает 1, если отношения a и b равны, и 0 в противном случае
bool bin_relation_equality(bin_relation a, bin_relation b) {
    bool is_equality = true;
    for (int i = 0; i < max2(a.max_value, b.max_value) && is_equality; i++) {
        is_equality = b.values[i] == a.values[i];
    }
    return is_equality;
}

//Возвращает 1, если отношение a строго включено в отношение b, и 0 в противном случае
bool bin_relation_strictInclusion(bin_relation a, bin_relation b) {
    bool is_inclusion = true;
    bool is_equality = true;
    for (int i = 0; i < max2(a.max_value, b.max_value); i++) {
        is_inclusion &= (b.values[i] | a.values[i]) == b.values[i];
        is_equality &= b.values[i] == a.values[i];
    }
    return is_inclusion && !is_equality;
}
```

```
35
36 //Возвращает отношение – объединение отношений a и b
37 bin_relation bin_relation_union(bin_relation a, bin_relation b) {
38     bin_relation c = bin_relation_createEmpty(max2(a.max_value,
       b.max_value));
39     for (int i = 0; i < c.max_value; i++) {
40         c.values[i] = a.values[i] | b.values[i];
41     }
42     return c;
43 }
44
45 //Возвращает отношение – пересечение отношений a и b
46 bin_relation bin_relation_intersection(bin_relation a, bin_relation b)
       {
47     bin_relation c = bin_relation_createEmpty(max2(a.max_value,
       b.max_value));
48     for (int i = 0; i < c.max_value; i++) {
49         c.values[i] = a.values[i] & b.values[i];
50     }
51     return c;
52 }
53
54 //Возвращает отношение – разность отношений a и b
55 bin_relation bin_relation_difference(bin_relation a, bin_relation b) {
56     bin_relation c = bin_relation_createEmpty(max2(a.max_value,
       b.max_value));
57     for (int i = 0; i < c.max_value; i++) {
58         c.values[i] = a.values[i] & ~b.values[i];
59     }
60     return c;
61 }
62
63 //Возвращает отношение – симметрическую разность отношений a и b
64 bin_relation bin_relation_symmetricalDifference(bin_relation a,
       bin_relation b) {
65     bin_relation c = bin_relation_createEmpty(max2(a.max_value,
       b.max_value));
66     for (int i = 0; i < c.max_value; i++) {
67         c.values[i] = a.values[i] ^ b.values[i];
68     }
69     return c;
70 }
71
72 //Возвращает отношение – дополнение до универсального отношения на
       множестве {1, ..., a.max_value} отношения a
73 bin_relation bin_relation_complement(bin_relation a) {
74     bin_relation c = bin_relation_createEmpty(a.max_value);
```

```
75      for (int i = 0; i < c.max_value; i++) {
76          c.values[i] = ~a.values[i] & ((1<<a.max_value) - 1);
77      }
78      return c;
79  }
80
81  //Возвращает отношение – обращение отношения a
82  bin_relation bin_relation_conversion(bin_relation a) {
83      bin_relation c = bin_relation_createEmpty(a.max_value);
84      for (int i = 1; i <= c.max_value; i++) {
85          for (int j = 1; j <= c.max_value; j++) {
86              bin_relation_changeValue(&c, i, j,
        bin_relation_getValue(a, j, i));
87          }
88      }
89      return c;
90  }
91
92  //Возвращает отношение – композицию отношений a и b
93  bin_relation bin_relation_composition(bin_relation a, bin_relation b) {
94      bin_relation c = bin_relation_createEmpty(max2(a.max_value,
        b.max_value));
95      for (int i = 1; i <= c.max_value; i++) {
96          for (int j = 1; j <= c.max_value; j++) {
97              if (bin_relation_getValue(a, i, j)) {
98                  for (int k = 1; k <= c.max_value; k++) {
99                      c.values[i-1] |= bin_relation_getValue(b, j, k) <<
        (k-1);
100                 }
101             }
102         }
103     }
104     return c;
105 }
106
107 //Возвращает отношение – степень degree отношения a, то есть
        композицию отношения a с самим собой, выполненную degree-1 раз
108 bin_relation bin_relation_degree(bin_relation a, int degree) {
109     bin_relation c = a;
110     for (int i = 1; i < degree; i++) {
111         c = bin_relation_composition(c, a);
112     }
113     return c;
114 }
115 #endif
```

../bin_relations/bin_relations_operations.c

Задание 1.5

Текст задания: Написать программу, вычисляющую значение выражения («Варианты заданий», пункт **б)**) и вычислить его при заданных отношениях («Варианты заданий», пункт **а)**).

Теперь, когда основные операции над отношениями реализованы программно, нам не составит труда вычислить значение выражения и вывести на экран отношение D. Для этого в соответствующей функции надо лишь в нужном порядке выполнить все операции (иногда сохраняя промежуточный результат в отдельных переменных для удобочитаемости) и вернуть окончательный результат вычислений. В теле функции main мы сначала вычисляем отношения A, B, C с помощью функций, написанных для решения задания 1.3, а потом вызываем функцию для вычисления отношения D, передавая отношения A, B, C как аргументы. Полученное отношение можно вывести на экран как в виде матрицы, так и в виде перечисления упорядоченных пар.

```c
#include <stdio.h>
#include "bin_relation_definition_input_output.c"
#include "ABC_forming.c"
#include "bin_relations_operations.c"


bin_relation expression_D_variant6 (bin_relation A, bin_relation B, bin_relation C) {
    bin_relation exp_left = bin_relation_intersection(A, bin_relation_conversion(B));
    bin_relation exp_right = bin_relation_composition(bin_relation_composition(A, C), B);
    bin_relation result = bin_relation_symmetricalDifference(exp_left, exp_right);
    return result;
}

int main() {
    bin_relation A = relation_A_variant6();
    bin_relation B = relation_B_variant6();
    bin_relation C = relation_C_variant6();

    bin_relation D = expression_D_variant6(A, B, C);
    printf("Relation D as matrix: \n");
    bin_relation_matrixPrint(D);
    printf("\nRelation D as list of pairs: \n");
    bin_relation_pairPrint(D);
}
```

../bin_relations/main2.c

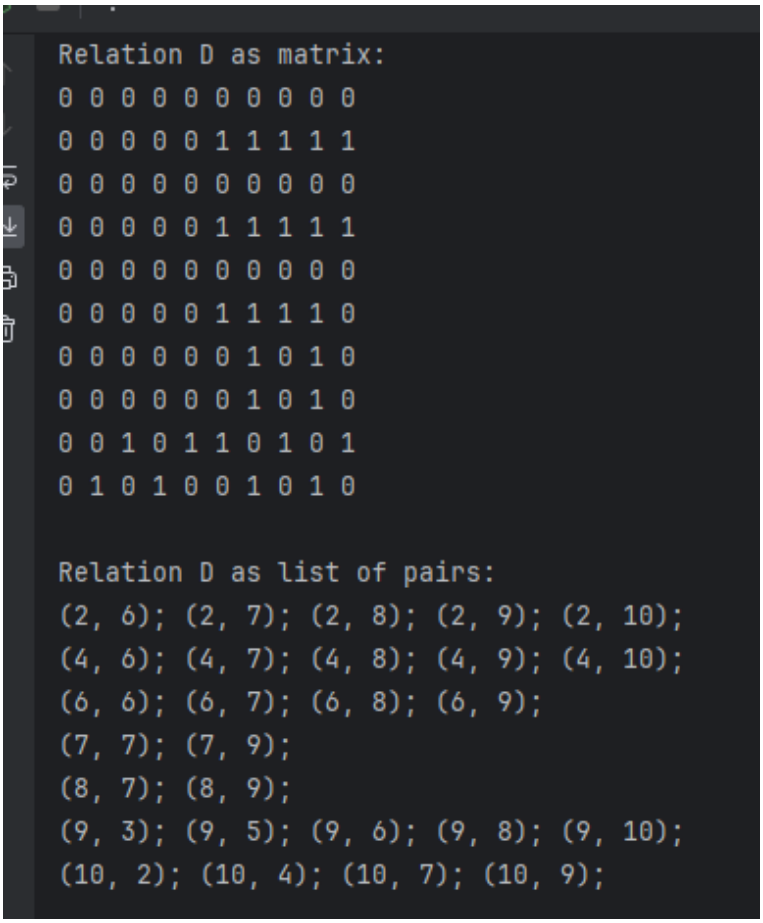


Рис. 4: Вывод значения выражения D в двух формах

Матрица отношения D соответствует графику, который мы построили в задании 1.2.

Часть 2. Свойства отношений

Задание 2.1

Текст задания: Определить основные и производные свойства отношений («Варианты заданий», пункт **а)**).

Определим основные свойства отношения A, для удобства снова приведя его матрицу:

0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	1	0	1
0	0	0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1	0	1
0	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1

Анализируя эту матрицу, мы можем сказать, что:

- Отношение A **не является ни рефлексивным, ни антирефлексивным**; мы ясно видим, что главная диагональ матрицы содержит как нули, так и единицы. К примеру, рефлексивным отношение A не является хотя бы потому, что в него не входит пара $(1, 1)$, а антирефлексивным не является, потому что в него входит пара $(5, 5)$.

- Отношение A **является симметричным**; это можно как проследить в самой матрице, так и принять как следствие из факта, что вхождение или невхождение пары в отношение определяется суммой элементов, которая не зависит от их порядка. Очевидно, что если отношение A является симметричным, то **антисимметричным** оно точно **не является**.

- Отношение A **не является ни транзитивным, ни антитранзитивным**. К примеру, транзитивным отношение A не является хотя бы потому, что в него входят пары $(1, 9)$ и $(9, 3)$, но не входит пара $(1, 3)$. Антитранзитивным оно также не является хотя бы потому, что в него входят пары $(6, 8)$ и $(8, 4)$, а также пара $(6, 4)$.

- Отношение A **не является полным**, хотя бы потому, что в нее не входит ни пара $(1, 2)$, ни пара $(2, 1)$.

Поскольку отношение A не рефлексивно, его уже **нельзя** назвать **толерантным и эквивалентным**, а поскольку оно не является антисимметричным, то оно **не обладает свойством порядка** и его частными случаями (свойствами **нестрогого порядка, строгого порядка, линейного порядка, нестрогого линейного и строгого линейного порядка**).

Проделаем то же с отношением B, матрица которого выглядит так:

0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1

Анализируя эту матрицу, мы можем сказать, что:

- Отношение B **не является ни рефлексивным, ни антирефлексивным**; главная диагональ матрицы содержит как нули, так и единицы. К примеру, рефлексивным отношение B не является хотя бы потому, что в него не входит пара $(1, 1)$, а антирефлексивным не является, потому что в него входит пара $(7, 7)$.

- Отношение B **не является ни симметричным, ни антисимметричным**. К примеру, его нельзя назвать симметричным хотя бы потому, что в него входит пара $(3, 9)$, но не входит пара $(9, 3)$. Антисимметричным его нельзя назвать хотя бы потому, что в него входят одновременно пары $(7, 9)$ и $(9, 7)$.

- Отношение B **не является ни транзитивным, ни антитранзитивным**. К примеру, транзитивным отношение B не является хотя бы потому, что в него входят пары $(3, 9)$ и $(9, 7)$, но не входит пара $(3, 7)$. Антитранзитивным оно также не является хотя бы потому, что в него входят пары $(7, 8)$ и $(8, 10)$, а также пара $(7, 10)$.

- Отношение B **не является полным** хотя бы потому, что в нее не входит ни пара $(7, 3)$, ни пара $(3, 7)$.

Поскольку отношение B не рефлексивно, его уже **нельзя** назвать **толерантным и эквивалентным**, а поскольку оно не является антисимметричным, то оно **не обладает** свойством **порядка** и его частными случаями (свойствами **нестрогого порядка, строгого порядка, линейного порядка, нестрогого линейного и строгого линейного порядка**).

Наконец, перейдем к отношению C, матрица которого выглядит так:

0	0	1	0	1	0	1	0	0	1
0	0	1	0	1	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Анализируя эту матрицу, мы можем сказать, что:

- Отношение C **не является рефлексивным** уже потому, что в него не входит пара $(1, 1)$. Но это отношение **является антирефлексивным**: главная диагональ матрицы содержит только нули, поскольку это отношение образовано декартовым произведением непересекающихся множеств, и в нем не может быть пар вида (x, x).

- Отношение C **не является симметричным** уже потому, что в него входит пара $(1, 3)$, но не входит пара $(1, 3)$. Однако это отношение **является антисимметричным**; это можно как проследить в самой матрице, так и принять как следствие из факта, что отношение образовано декартовым произведением непересекающихся множеств (обозначим их как X и Y). Уже поэтому в отношение не могут входить одновременно пары (x, y) и (y, x), поскольку если элемент x принадлежит множеству X, он не будет входить в множество Y.

- Снова обратимся к тому факту, что отношение C образовано декартовым произведением неких непересекающихся множеств X и Y. Если в отношение входит пара вида (x, z), то справедливо утверждение, что $z \in Y$, соответственно, $z \notin X$ и пара вида (z, y) не может входить в отношение. Получается, в отношении C, не существует двух пар вида (x, z) и (z, y). Для любой тройки элементов x, y, z в транзитивном отношении справедливо следствие $xRz\&zRy \to xRy$; а в антитранзитивном — следствие $xRz\&zRy \to \overline{xRy}$. Но для любой тройки элементов в отношении C конъюнкция $xRz\&zRy$ будет ложной, и любое следствие из нее будет истинным. Соотвественно, выполняются сразу оба условия, и отношение C **одновременно и транзитивно, и антитранзитивно**.

- Отношение C **не является полным**, хотя бы потому, что в нее не входит ни пара $(2, 4)$, ни пара $(4, 2)$.

Поскольку отношение C не рефлексивно, его уже **нельзя** назвать **толерантным и эквивалентным**. Оно **обладает свойством порядка** (антисимметрично и транзитивно), при этом **не обладает свойством нестрогого порядка** (так как не рефлексивно), но **обладает свойством строгого порядка** (антирефлексивно). Оно **не обладает свойством линейного порядка** (поскольку не является полным) и его частными случаями (свойствами **нестрогого линейного и строгого линейного порядка**).

Задание 2.2

Текст задания: Для каждого основного свойства написать функцию, которая определяет, обладает ли этим свойством заданное отношение. Если отношение не обладает свойством, то функция должна дать этому объяснение.

Для каждого основного свойства отношений (рефлексивности, антирефлексивности, симметричности, антисимметричности, транзитивности, антитранзитивности, полноты) напишем функции, определяющие, обладает ли отношение этим свойством. Если отношение обладает этим свойством, на экран выводится сообщение об этом; если отношение им не обладает, то сообщение об этом также будет содержать пару или пары, вхождение или невхождение которых в отношение доказывает это. Однако все эти функции, помимо вывода, возвращают соответствующее булево значение («истину», если отношение обладает данным свойством, и «ложь»,

если не обладает) — его можно сохранить в переменной и использовать в программе в дальнейшем. Пример этого — последняя функций файла, которая вызывает все функции для основным свойств, сохраняя булевы значения в отдельных переменных и используя их для определения производных свойств. При этом для производных свойств сообщения выводятся только тогда, когда отношение обладает ими, а если оно не обладает некоторым производным свойством, то это свойство не упоминается. Впрочем, сообщение об отсутствии производных свойств выводится в эстетических целях.

```
#include <stdbool.h>
#include "bin_relation_definition_input_output.c"

#ifndef BIN_REL_PROPERTIES
#define BIN_REL_PROPERTIES

//Возвращает 1, если отношение a рефлексивно, и 0 в противном случае, а также печатает соответствующее сообщение
bool bin_relation_is_reflexive(bin_relation a) {
    for (int x = 1; x <= a.max_value; x++) {
        if (!bin_relation_getValue(a, x, x)) {
            printf("This relation is not reflexive, because the pair (%d, %d) doesn't belong it.\n", x, x);\
            return false;
        }
    }
    printf("This relation is reflexive.\n");
    return true;
}

//Возвращает 1, если отношение нтирефлексивноа, и 0 в противном случае, а также печатает соответствующее сообщение
bool bin_relation_is_antireflexive(bin_relation a) {
    for (int x = 1; x <= a.max_value; x++) {
        if (bin_relation_getValue(a, x, x)) {
            printf("This relation is not antireflexive, because the pair (%d, %d) belongs it.\n", x, x);
            return false;
        }
    }
    printf("This relation is antireflexive.\n");
    return true;
}

//Возвращает 1, если отношение симметрично, и 0 в противном случае, а также печатает соответствующее сообщение
bool bin_relation_is_symmetrical(bin_relation a) {
    for (int x = 1; x <= a.max_value; x++) {
        for (int y = x+1; y <= a.max_value; y++) {
```

```
35              if (bin_relation_getValue(a, x, y) &&
    !bin_relation_getValue(a, y, x)) {
36                  printf("This relation is not symmetrical, because the
    pair (%d, %d) belongs it and the pair (%d, %d) doesn't.\n", x, y,
    y, x);
37                  return false;
38              }
39          }
40      }
41      printf("This relation is symmetrical.\n");
42      return true;
43  }
44
45  //Возвращает 1, если отношение антисимметрично, и 0 в противном
    случае, а также печатает соответствующее сообщение
46  bool bin_relation_is_antisymmetrical(bin_relation a) {
47      for (int x = 1; x <= a.max_value; x++) {
48          for (int y = x+1; y <= a.max_value; y++) {
49              if (bin_relation_getValue(a, x, y) &&
    bin_relation_getValue(a, y, x)) {
50                  printf("This relation is not antisymmetrical, because
    both pairs (%d, %d) and (%d, %d) belong it.\n", x, y, y, x);
51                  return false;
52              }
53          }
54      }
55      printf("This relation is antisymmetrical.\n");
56      return true;
57  }
58
59  //Возвращает 1, если отношение транзитивно, и 0 в противном случае, а
    также печатает соответствующее сообщение
60  bool bin_relation_is_transitive(bin_relation a) {
61      for (int x = 1; x <= a.max_value; x++) {
62          for (int z = 1; z <= a.max_value; z++) {
63              if (bin_relation_getValue(a, x, z) && a.values[z-1] != 0) {
64                  for (int y = 1; y <= a.max_value; y++) {
65                      if (bin_relation_getValue(a, z, y) &&
    !bin_relation_getValue(a, x, y)) {
66                          printf("This relation is not transitive,
    because both pairs (%d, %d) and (%d, %d) belong it, but pair (%d,
    %d) doesn't.\n", x, z, z, y, x, y);
67                          return false;
68                      }
69                  }
70              }
71          }
```

```
72      }
73      printf("This relation is transitive.\n");
74      return true;
75  }
76
77  //Возвращает 1, если отношение антитранзитивно, и 0 в противном
        случае, а также печатает соответствующее сообщение
78  bool bin_relation_is_antitransitive(bin_relation a) {
79      for (int x = 1; x <= a.max_value; x++) {
80          for (int z = 1; z <= a.max_value; z++) {
81              if (bin_relation_getValue(a, x, z) && a.values[z-1] != 0) {
82                  for (int y = 1; y <= a.max_value; y++) {
83                      if (bin_relation_getValue(a, z, y) &&
        bin_relation_getValue(a, x, y)) {
84                          printf("This relation is not antitransitive,
        because both pairs (%d, %d) and (%d, %d) belong it, and also the
        pair (%d, %d) does.\n", x, z, z, y, x, y);
85                          return false;
86                      }
87                  }
88              }
89          }
90      }
91      printf("This relation is antitransitive.\n");
92      return true;
93  }
94
95  //Возвращает 1, если отношение полно, и 0 в противном случае, а также
        печатает соответствующее сообщение
96  bool bin_relation_is_full(bin_relation a) {
97      for (int x = 1; x <= a.max_value; x++) {
98          for (int y = x+1; y <= a.max_value; x++) {
99              if (!bin_relation_getValue(a, x, y) &&
        !bin_relation_getValue(a, y, x)) {
100                 printf("This relation is not full, because none of the
        pairs (%d, %d) and (%d, %d) belong it.\n", x, y, y, x);
101                 return false;
102             }
103         }
104     }
105     printf("This relation is full.\n");
106     return true;
107 }
108
109 //Выводит на экран сообщения, обладает или не обладает отношение а
        каждым из основных свойств, а также выводит производные свойства,
        которыми обладает это отношение
```

```
110 void bin_relation_print_all_properties(bin_relation a) {
111     printf("\tSimple properties of a relation: \n");
112     bool is_reflexive = bin_relation_is_reflexive(a);
113     bool is_antireflexive = bin_relation_is_antireflexive(a);
114     bool is_symmetrical = bin_relation_is_symmetrical(a);
115     bool is_antisymmetrical = bin_relation_is_antisymmetrical(a);
116     bool is_transitive = bin_relation_is_transitive(a);
117     bool is_antitransitive = bin_relation_is_antitransitive(a);
118     bool is_full = bin_relation_is_full(a);
119 
120     bool has_derived_properties = false;
121 
122     printf("\tDerived properties of a relation: \n");
123     if (is_reflexive && is_symmetrical) {
124         has_derived_properties = true;
125         printf("This relation has the tolerance property.\n");
126         if (is_transitive) {
127             printf("This relation has the equivalence property.\n");
128         }
129     }
130     if (is_antisymmetrical && is_transitive) {
131         has_derived_properties = true;
132         printf("This relation has the order property.\n");
133         if (is_reflexive) {
134             printf("This relation has the non-strict order
    property.\n");
135         }
136         if (is_antireflexive) {
137             printf("This relation has the strict order property.\n");
138         }
139         if (is_full) {
140             printf("This relation has the linear order property.\n");
141             if (is_reflexive) {
142                 printf("This relation has the non-strict linear order
    property.\n");
143             }
144             if (is_antireflexive) {
145                 printf("This relation has the non-strict linear order
    property.\n");
146             }
147         }
148     }
149     if (!has_derived_properties) {
150         printf("This relation hasn't derived properties.\n");
151     }
152 }
153 
```

```
154 #endif
```

../bin_relations/bin_relations_properties.c

Задание 2.3

Текст задания: 2.3. Написать программу, определяющую свойства отношения и определить свойства отношений («Варианты заданий», пункт **а)**).

Учитывая что в задании 2.2 мы задали функцию, выводящую на экран информацию о всех основных и производных свойствах отношения, текущее задание сводится к тому, чтобы вызвать ее для каждого из отношений A, B, C.

```
1  #include <stdio.h>
2  #include "bin_relation_definition_input_output.c"
3  #include "ABC_forming.c"
4  #include "bin_relations_properties.c"
5
6
7  int main() {
8      bin_relation A = relation_A_variant6();
9      bin_relation B = relation_B_variant6();
10     bin_relation C = relation_C_variant6();
11
12     bin_relation I = bin_relation_createEmpty(4); //Можно заменить
       значение max_value на 1, чтобы увидеть, какие свойства при этом
       меняются
13     for (int x = 1; x <= I.max_value; x++) {
14         bin_relation_changeValue(&I, x, x, 1);
15     }
16
17     printf("All properties of relation A: \n");
18     bin_relation_print_all_properties(A);
19     printf("\nAll properties of relation B: \n");
20     bin_relation_print_all_properties(B);
21     printf("\nAll properties of relation C: \n");
22     bin_relation_print_all_properties(C);
23
24
25     printf("\nAll properties of relation I with power of set = %d:
       \n", I.max_value);
26     bin_relation_print_all_properties(I);
27 }
```

../bin_relations/main3.c

```
All properties of relation A:
        Simple properties of a relation:
This relation is not reflexive, because the pair (1, 1) doesn't belong it.
This relation is not antireflexive, because the pair (5, 5) belongs it.
This relation is symmetrical.
This relation is not antisymmetrical, because both pairs (1, 9) and (9, 1) belong it.
This relation is not transitive, because both pairs (1, 9) and (9, 1) belong it, but pair (1, 1) doesn't.
This relation is not antitransitive, because both pairs (1, 9) and (9, 9) belong it, and also the pair (1, 9) does.
This relation is not full, because none of the pairs (1, 2) and (2, 1) belong it.
        Derived properties of a relation:
This relation hasn't derived properties.
```

Рис. 5: Вывод свойств отношения A

```
All properties of relation B:
        Simple properties of a relation:
This relation is not reflexive, because the pair (1, 1) doesn't belong it.
This relation is not antireflexive, because the pair (7, 7) belongs it.
This relation is not symmetrical, because the pair (1, 10) belongs it and the pair (10, 1) doesn't.
This relation is not antisymmetrical, because both pairs (6, 9) and (9, 6) belong it.
This relation is not transitive, because both pairs (1, 10) and (10, 6) belong it, but pair (1, 6) doesn't.
This relation is not antitransitive, because both pairs (1, 10) and (10, 10) belong it, and also the pair (1, 10) does.
This relation is not full, because none of the pairs (1, 2) and (2, 1) belong it.
        Derived properties of a relation:
This relation hasn't derived properties.
```

Рис. 6: Вывод свойств отношения B

Выведенные на экран сообщения соответствуют умозаключениям, сделанным в задании 2.1.

Вывод: бинарное отношение — частный случай бинарного соответствия, подмножество квадрата некоторого множества на само себя. Гораздо чаще приходится работать с бинарными отношениями, чем с бинарными соответствиями: отношения позволяют обозначить связи между элементами одного множества — соответственно, между элементами, имеющими одинаковую природу. На основе уже существующих отношений можно создавать новые, отображающие более сложные связи; для этого используются операции над отношениями, как аналогичные операциям над множествами, так и не определенные для них. Сами отношения обладают характеристиками, наиболее важные из которых имеют отдельные названия и называются основными свойствами; производные свойства являются комбинацией основных. И в ходе выполнения этой лабораторной работы мы как научились вручную выполнять операции над отношениями и определять их свойства, так и реализовали функции для быстрого автоматизированного вычисления выражений с отношениями и определения их свойств.

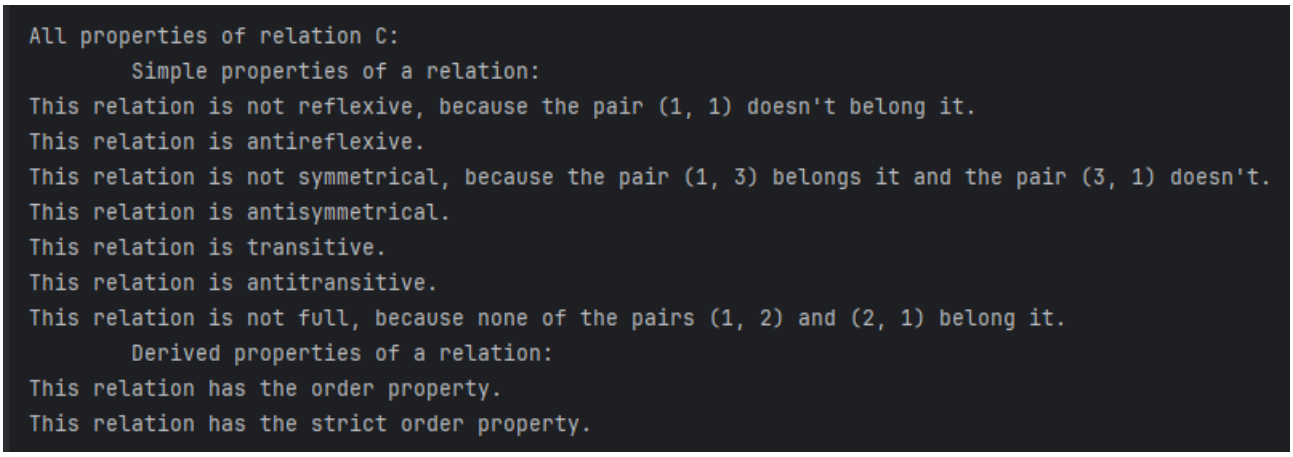


Рис. 7: Вывод свойств отношения C