

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
тема: «Отношения и их свойства»

Выполнил: ст. группы ПВ-202

Аладиб язан

Проверил:

Рязанов Юрий Дмитриевич

Бондаренко Татьяна

Владимировна

Лабораторная работа № 3.1

«Отношения и их свойства»

Цель работы:

изучить способы задания отношений, операции над отношениями и свойства отношений, научиться программно реализовывать операции и определять свойства отношений.

Задания к работе:

Часть 1. Операции над отношениями

- 1.1. Представить отношения (см. «Варианты заданий», п.а) графически, графом и матрицей.
- 1.2. Вычислить значение выражения (см. «Варианты заданий», п.б) при заданных отношениях (см. «Варианты заданий», п.а).
- 1.3. Написать программы, формирующие матрицы заданных отношений (см. «Варианты заданий», п.а).
- 1.4. Программно реализовать операции над отношениями.
- 1.5. Написать программу, вычисляющую значение выражения (см. «Варианты заданий», п.б) и вычислить его при заданных отношениях (см. «Варианты заданий», п.а).

Часть 2. Свойства отношений

- 2.1. Определить основные свойства отношений (см. «Варианты заданий», п.а).
- 2.2. Определить, являются ли заданные отношения отношениями толерантности, эквивалентности и порядка.
- 2.3. Написать программу, определяющую свойства отношений, в том числе толерантности, эквивалентности и порядка, и определить свойства отношений (см. «Варианты заданий», п.а).

Задание варианта №2:

а)

$$A = \{(x, y) \mid x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x - \text{четно и } y > x\}$$

$$B = \{(x, y) \mid x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x - y = 1 \text{ или } x + y = 11)\}$$

$$C = \{(x, y) \mid x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x, y) \in \{2, 5, 8, 9, 10\} \times \{1, 3, 5, 10\}\}$$

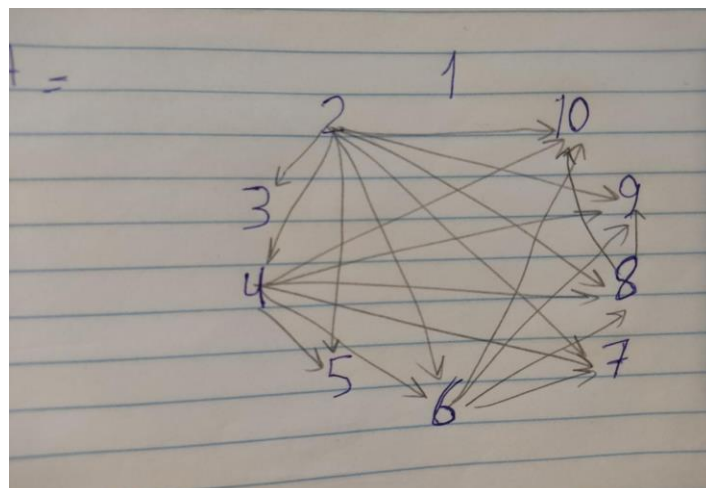
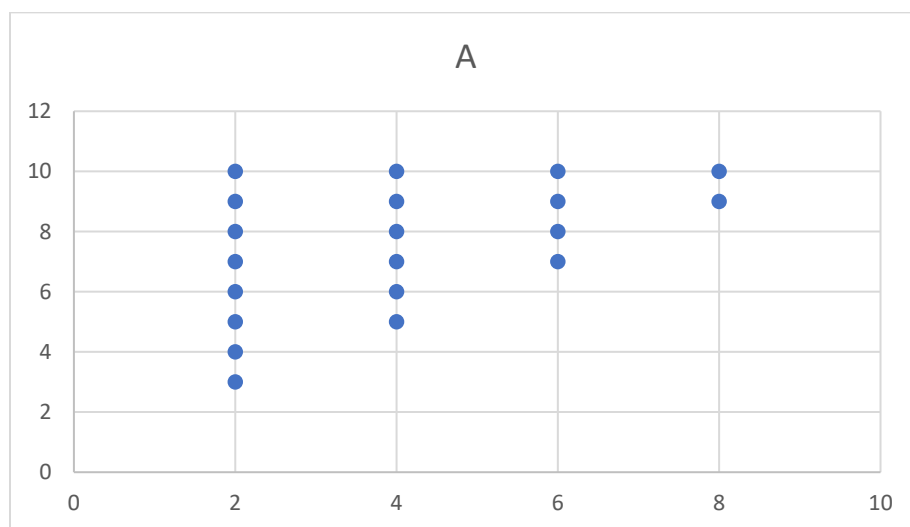
б)

$$D = A^{-1} \cap B \cup \bar{A} - C$$

1.1)

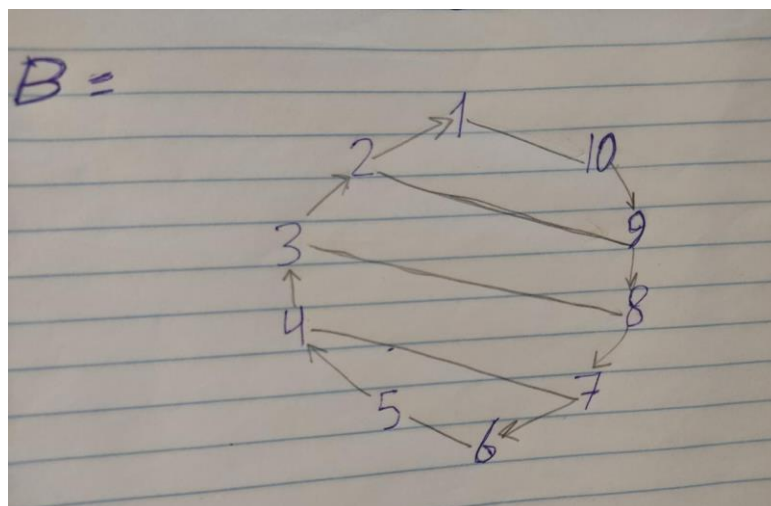
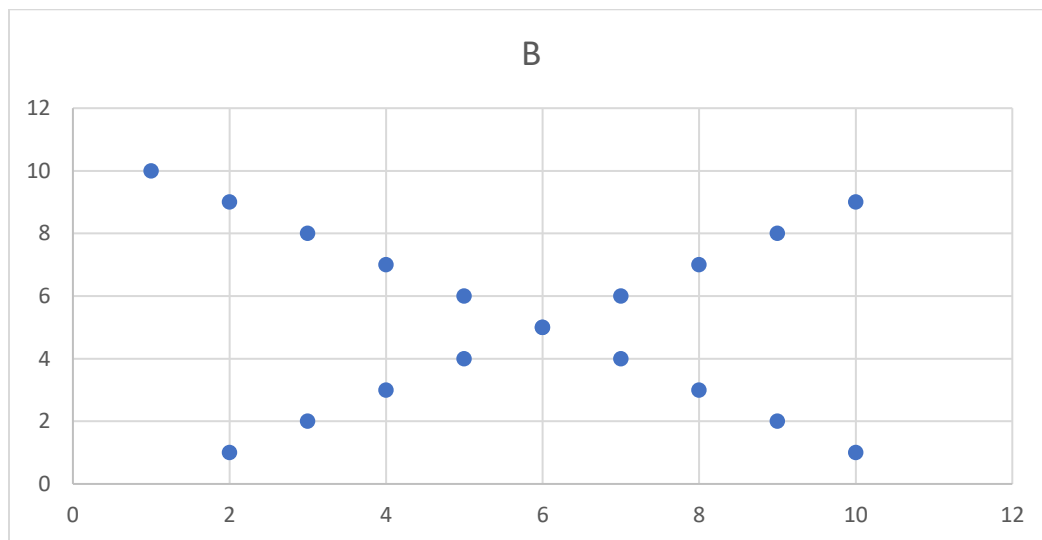
$$A = \{(x, y) \mid x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x - \text{четно и } y > x\}$$

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	1	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	1	1	1	1
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	1	1	1
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	1
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0



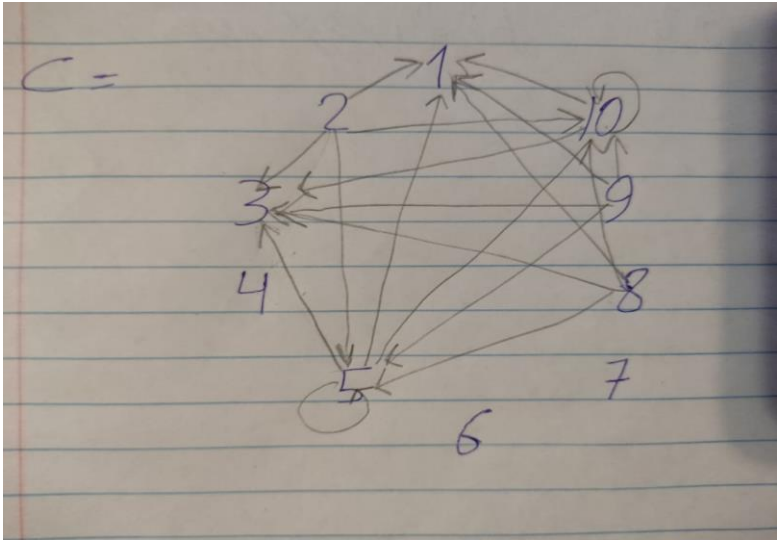
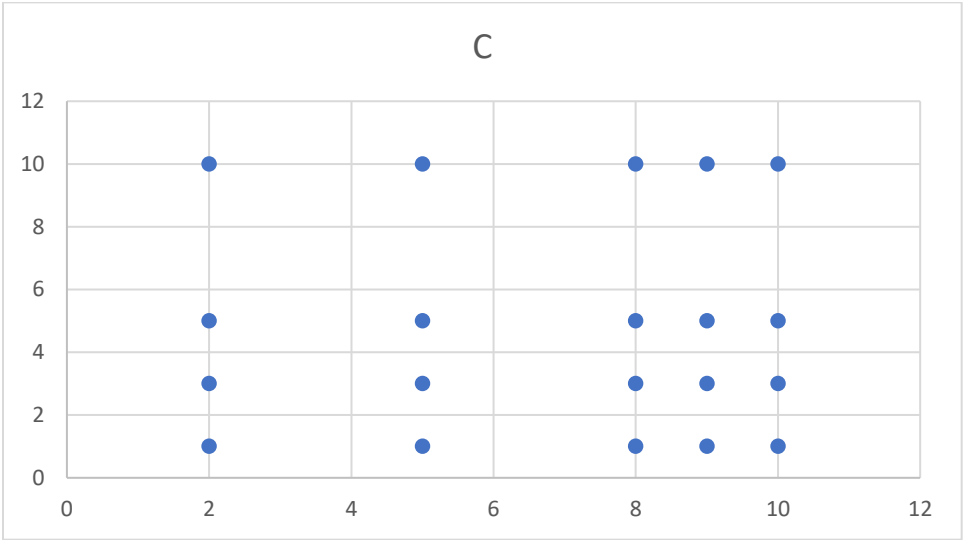
$$B = \{(x, y) \mid x \in N \cup y \in N \cup x < 11 \cup y < 11 \cup (x - y = 1 \text{ или } x + y = 11)\}$$

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	1
2	1	0	0	0	0	0	0	0	1	0
3	0	1	0	0	0	0	0	1	0	0
4	0	0	1	0	0	0	1	0	0	0
5	0	0	0	1	0	1	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0
7	0	0	0	1	0	1	0	0	0	0
8	0	0	1	0	0	0	1	0	0	0
9	0	1	0	0	0	0	0	1	0	0
10	1	0	0	0	0	0	0	0	1	0



$$C = \{(x,y) \mid x \in N \cup y \in N \cup x < 11 \cup y < 11 \cup (x,y) \in \{2,5,8,9,10\} \times \{1,3,5,10\}\}$$

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	1	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	1	0	1	0	1	0	0	0	0	1
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	1	0	1	0	1	0	0	0	0	1
9	1	0	1	0	1	0	0	0	0	1
10	1	0	1	0	1	0	0	0	0	1



1.2)

$$D = A^{-1} \cap B \cup \bar{A} - C$$

$A = \{(2,3)(2,4)(2,5)(2,6)(2,7)(2,8)(2,9)(2,10)(4,5)(4,6)(4,7)(4,8)(4,9)(4,10)(6,7)(6,8)(6,9)(6,10)(8,9)(8,10)\}$

$B = \{(1,10)(2,1)(2,9)(3,2)(3,8)(4,3)(4,7)(5,4)(5,6)(6,5)(7,4)(7,6)(8,3)(8,7)(9,2)(9,8)(10,1)(10,9)\}$

$C = \{(2,1)(2,3)(2,5)(2,10)(5,1)(5,3)(5,5)(5,10)(8,1)(8,3)(8,5)(8,10)(9,1)(9,3)(9,5)(9,10)(10,1)(10,3)(10,5)(10,10)\}$

$A^{-1} = \{(3,2)(4,2)(5,2)(5,4)(6,2)(6,4)(7,2)(7,4)(7,6)(8,2)(8,4)(8,6)(9,2)(9,4)(9,6)(9,8)(10,2)(10,4)(10,6)(10,8)\}$

$\bar{A} = \{(1,1)(1,2)(1,3)(1,4)(1,5)(1,6)(1,7)(1,8)(1,9)(1,10)(2,1)(2,2)(3,1)(3,2)(3,3)(3,4)(3,5)(3,6)(3,7)(3,8)(3,9)(3,10)(4,1)(4,2)(4,3)(4,4)(5,1)(5,2)(5,3)(5,4)(5,5)(5,6)(5,7)(5,8)(5,9)(5,10)(6,1)(6,2)(6,3)(6,4)(6,5)(6,6)(7,1)(7,2)(7,3)(7,4)(7,5)(7,6)(7,7)(7,8)(7,9)(8,1)(8,2)(8,3)(8,4)(8,5)(8,6)(8,7)(8,8)(9,1)(9,2)(9,3)(9,4)(9,5)(9,6)(9,7)(9,8)(9,9)(9,10)(10,1)(10,2)(10,3)(10,4)(10,5)(10,6)(10,7)(10,8)(10,9)(10,10)\}$

$A^{-1} \cap B = \{(3,2)(5,4)(7,4)(7,6)(9,2)(9,8)\}$

$A^{-1} \cap B \cup \bar{A} = \{(1,1)(1,2)(1,3)(1,4)(1,5)(1,6)(1,7)(1,8)(1,9)(1,10)(2,1)(2,2)(3,1)(3,2)(3,3)(3,4)(3,5)(3,6)(3,7)(3,8)(3,9)(3,10)(4,1)(4,2)(4,3)(4,4)(5,1)(5,2)(5,3)(5,4)(5,5)(5,6)(5,7)(5,8)(5,9)(5,10)(6,1)(6,2)(6,3)(6,4)(6,5)(6,6)(7,1)(7,2)(7,3)(7,4)(7,5)(7,6)(7,7)(7,8)(7,9)(8,1)(8,2)(8,3)(8,4)(8,5)(8,6)(8,7)(8,8)(9,1)(9,2)(9,3)(9,4)(9,5)(9,6)(9,7)(9,8)(9,9)(9,10)(10,1)(10,2)(10,3)(10,4)(10,5)(10,6)(10,7)(10,8)(10,9)(10,10)\}$

$D = A^{-1} \cap B \cup \bar{A} - C = \{(1,1)(1,2)(1,3)(1,4)(1,5)(1,6)(1,7)(1,8)(1,9)(1,10)(2,2)(3,1)(3,2)(3,3)(3,4)(3,5)(3,6)(3,7)(3,8)(3,9)(3,10)(4,1)(4,2)(4,3)(4,4)(5,2)(5,4)(5,6)(5,7)(5,8)(5,9)(6,1)(6,2)(6,3)(6,4)(6,5)(6,6)(7,1)(7,2)(7,3)(7,4)(7,5)(7,6)(7,7)(7,8)(7,9)(8,2)(8,4)(8,6)(8,7)(8,8)(9,2)(9,4)(9,6)(9,7)(9,8)(9,9)(10,2)(10,4)(10,6)(10,7)(10,8)(10,9)\}$

1.3/1.4)

```
#include <stdio.h>
#define n 11

// Union
void Union(int A[n][n], int B[n][n], int C[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            C[x][y] = A[x][y] || B[x][y];
        }
    }
}

// Intersection
void Intersection(int A[n][n], int B[n][n], int C[n][n]) {
    for (int x = 1; x < n; x++) {
```

```

        for (int y = 1; y < n; y++) {
            C[x][y] = A[x][y] && B[x][y];
        }
    }
}

// Difference
void Difference(int A[n][n], int B[n][n], int C[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            C[x][y] = A[x][y] > B[x][y];
        }
    }
}

// Symmetric Difference
void Symmetric_Difference(int A[n][n], int B[n][n], int C[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            C[x][y] = (!A[x][y] && B[x][y]) || (A[x][y] && !B[x][y]);
        }
    }
}

// Complement A
void Complement(int A[n][n], int C[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            C[x][y] = !A[x][y];
        }
    }
}

// Inverse A
void Inverse(int A[n][n], int C[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            C[x][y] = A[y][x];
        }
    }
}

// Equality
int Equality(int A[n][n], int B[n][n]) {
    int r = 1;

```

```

        for (int x = 1; x < n && r == 1; x++) {
            for (int y = 1; y < n && r == 1; y++) {
                r = A[x][y] == B[x][y];
            }
        }
        return r;
    }

// Inclusion
int Inclusion(int A[n][n], int B[n][n]) {
    int r = 1;
    for (int x = 1; x < n && r == 1; x++) {
        for (int y = 1; y < n && r == 1; y++) {
            r = A[x][y] <= B[x][y];
        }
    }
    return r;
}

// Composition (superposition or multiplication)
void Composition(int A[n][n], int B[n][n], int C[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            C[x][y] = 0;
            for (int z = 1; z < n; z++) {
                C[x][y] = C[x][y] || (A[x][z] && B[z][y]);
            }
        }
    }
}

// Formation of relation matrix A
void ResultA(int A[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            if ((x%2==0) && y>x)
                A[x][y] = 1;
        }
    }
}

// Formation of relation matrix B
void ResultB(int B[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {

```



```

        if (x-y==1 || x+y==11)
            B[x][y] = 1;
    }
}

// Formation of relation matrix C
void ResultC(int C[n][n]) {
    int Cx[5]={2,5,8,9,10},Cy[4]={1,3,5,10};
    for (int x = 0; x < 5; x++) {
        for (int y = 0; y < 4; y++) {
            C[Cx[x]][Cy[y]]=1;
        }
    }
}

void Print_Relation(int A[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            printf("%d ", A[x][y]);
        }
        printf("\n");
    }
}

void Equal0(int A[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            A[x][y] = 0;
        }
    }
}

int main() {
    int A[n][n];
    int B[n][n];
    int C[n][n];
    int D[n][n];
    int R1[n][n];
    int R2[n][n];
    int R3[n][n];
    int R4[n][n];
    Equal0(A);
    Equal0(B);
    Equal0(C);

```

```

    ResultA(A);
    ResultB(B);
    ResultC(C);
    printf("A = \n");
    ResultA(A);
    Print_Relation(A);
    printf("_____ \n");
    printf("B = \n");
    ResultB(B);
    Print_Relation(B);
    printf("_____ \n");
    printf("C = \n");
    ResultC(C);
    Print_Relation(C);
    printf("_____ \n");
    printf("A^-1 = \n");
    Inverse(A,R1);
    Print_Relation(R1);
    printf("_____ \n");
    printf("~A = \n");
    Complement(A,R2);
    Print_Relation(R2);
    printf("_____ \n");
;
    printf("A^-1 ∩ B = \n");
    Intersection(R1,B,R3);
    Print_Relation(R3);
    printf("_____ \n");
    printf("A^-1 ∩ B ∪ ~A = \n");
    Union(R3,R2,R4);
    Print_Relation(R4);
    printf("_____ \n");
    printf("D = A^-1 ∩ B ∪ ~A - C = \n");
    Difference(R4,C,D);
    Print_Relation(D);
    return 0;
}

```

```
"C:\Users\yazon\Desktop\Clien C\cmake-build-debug\Clien_C.exe"
A =
0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 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 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
-----
B =
0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 1 0 0
0 0 1 0 0 0 1 0 0 0
0 0 0 1 0 1 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 0 1 0 0 0 0
0 0 1 0 0 0 1 0 0 0
0 1 0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0 1 0
-----
```

```
C =
0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 1 0 0 0 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
1 0 1 0 1 0 0 0 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
1 0 1 0 1 0 0 0 0 0 1
1 0 1 0 1 0 0 0 0 0 1
1 0 1 0 1 0 0 0 0 0 1

-----
A^*-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 1 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0 0
0 1 0 1 0 1 0 0 0 0 0 0
0 1 0 1 0 1 0 0 0 0 0 0
0 1 0 1 0 1 0 1 0 1 0 0
0 1 0 1 0 1 0 1 0 1 0 0
```

```

~A =
1 1 1 1 1 1 1 1 1 1
1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
-----
A^~1 и B =
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0

```

```

A^1 и B или ~A =
1 1 1 1 1 1 1 1 1 1
1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
-----
D = A^1 и B или ~A - C =
1 1 1 1 1 1 1 1 1 1
0 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0 0 0
0 1 0 1 0 1 1 1 1 0
1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 1 0 1 0 1 1 1 0 0
0 1 0 1 0 1 1 1 1 0
0 1 0 1 0 1 1 1 1 0

```

Часть 2. Свойства отношений

2.1)

Отношения	А	В	С
рефлексивность	-	-	-
антирефлексивность	+	+	-
симметричность	-	-	-
антисимметричность	+	-	-
транзитивность	+	-	+
антитранзитивность	-	+	-
полнота	-	-	-

2.2)

Отношения	A	B	C
толерантность	-	-	-
эквивалентность	-	-	-
порядок	+	-	-

2.3)

```
#include <stdio.h>
#define n 11

// reflexivity
int Reflexive(int A[n][n]){
    int r = 1;
    for (int i = 1; i < n && r==1; i++) {
        r = A[i][i];
    }
    return r;
}

// anti-reflexivity
int AntiReflexive(int A[n][n]){
    int r = 1;
    for (int i = 1; i < n; i++) {
        r = !A[i][i];
    }
    return !r;
}

// symmetry
int Symmetric(int A[n][n]){
    int r = 1;
    for (int i = 1; i < n && r==1; i++) {
        for (int j = 1; j < n && r==1; j++) {
            r = A[i][j] == A[j][i];
        }
    }
    return r;
}

// anti-symmetry
int AntiSymmetric(int A[n][n]){
    int r=1;
    for (int i = 1; i <n && r==1; i++) {
```

```

        for (int j = 1; j < n && r==1; j++) {
            if (A[i][j] == 1)
                r = !A[j][i];
        }
    }
    return r;
}

// transitivity
int Transitive(int A[n][n]){
    int r = 1;
    for (int i = 1; i < n && r==1; i++) {
        for (int j = 1; j < n && r==1; j++) {
            for (int k = 1; k < n && r==1; k++) {
                if(A[i][j] && A[j][k])
                    r=(A[i][k]==1);
            }
        }
    }
    return r;
}

// anti-transitivity
int AntiTransitive(int A[n][n]){
    int r = 1;
    for (int i = 1; i < n && r==1; i++) {
        for (int j = 1; j < n && r==1; j++) {
            for (int k = 1; k < n && r==1; k++) {
                if(A[i][j] && A[j][k])
                    r=!(A[i][k]==1);
            }
        }
    }
    return r;
}

// completeness
int Dence(int A[n][n]){
    int r=1;
    for (int i = 1; i < n && r==1; i++) {
        for (int j = 1; j < n && r==1; j++) {
            r = (A[i][j] == 1) || (A[j][i] == 1);
        }
    }
    return r;
}

```

```

}

// tolerance
int Tolerance(int A[n][n]){
    int r = (Reflexive(A) == 1) && (Symmetric(A) == 1);
    return r;
}

// equivalence
int Equivalence(int A[n][n]){
    int r = (Tolerance(A) == 1) && (Transitive(A) == 1);
    return r;
}

// order
int Order(int A[n][n]){
    int r = (AntiSymmetric(A) == 1) && (Transitive(A) == 1);
    return r;
}

// Formation of relation matrix A
void ResultA(int A[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            if ((x%2==0) && y>x)
                A[x][y] = 1;
        }
    }
}

// Formation of relation matrix B
void ResultB(int B[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            if (x-y==1 || x+y==11)
                B[x][y] = 1;
        }
    }
}

// Formation of relation matrix C
void ResultC(int C[n][n]) {
    int Cx[5]={2,5,8,9,10},Cy[4]={1,3,5,10};
    for (int x = 0; x < 5; x++) {
        for (int y = 0; y < 4; y++) {

```

```

        C[Cx[x]][Cy[y]]=1;
    }
}

void Equal0(int A[n][n]) {
    for (int x = 1; x < n; x++) {
        for (int y = 1; y < n; y++) {
            A[x][y] = 0;
        }
    }
}

void Check(int A[n][n]){
    if(Reflexive(A)==1)
        printf("R is reflexive\n");
    if(AntiReflexive(A)!=1)
        printf("R is anti-reflexive\n");
    if(Symmetric(A)==1)
        printf("R is symmetric\n");
    if(AntiSymmetric(A)==1)
        printf("R is anti-symmetric\n");
    if(Transitive(A)==1)
        printf("R is transitive\n");
    if(AntiTransitive(A)==1)
        printf("R is anti-transitive\n");
    if(Dence(A)==1)
        printf("R is complete\n");
    if(Tolerance(A)==1)
        printf("R is tolerant\n");
    if(Equivalence(A)==1)
        printf("R is equivalent\n");
    if(Order(A)==1)
        printf("R is ordered\n");
}

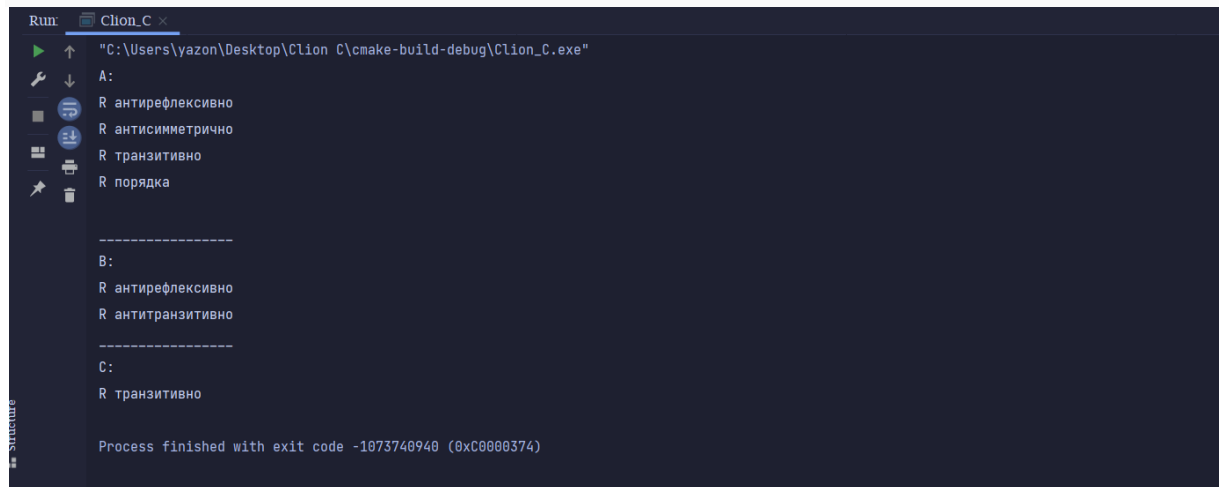
int main(){
    int A[n][n];
    int B[n][n];
    int C[n][n];
    Equal0(A);
    Equal0(B);
    Equal0(C);
    ResultA(A);
    ResultB(B);
}

```

```

    ResultC(C);
    printf("_____ \n");
    Check(A);
    printf("_____ \n");
    Check(B);
    printf("_____ \n");
    Check(C);
    return 0;
}

```



```

Run: Clion_C x
  "C:\Users\yazon\Desktop\Clion_C\cmake-build-debug\Clion_C.exe"
  A:
  R антирефлексивно
  R антисимметрично
  R транзитивно
  R порядка
  -----
  B:
  R антирефлексивно
  R антитранзитивно
  -----
  C:
  R транзитивно

Process finished with exit code -1073740940 (0xC0000374)

```