**МИНОБРНАУКИ РОССИИ**

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

дисциплина: Дискретная математика

тема: «Отношения и их свойства»

|  |  |
| --- | --- |
|  | Выполнил: ст. группы  Проверил: Рязанов Ю.Д. |

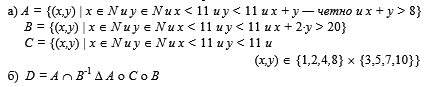
Белгород 20

**Цель занятия**: изучить способы задания отношений, операции над от ношениями и свойства отношений, научиться программно реализовывать операции и определять свойства отношений.

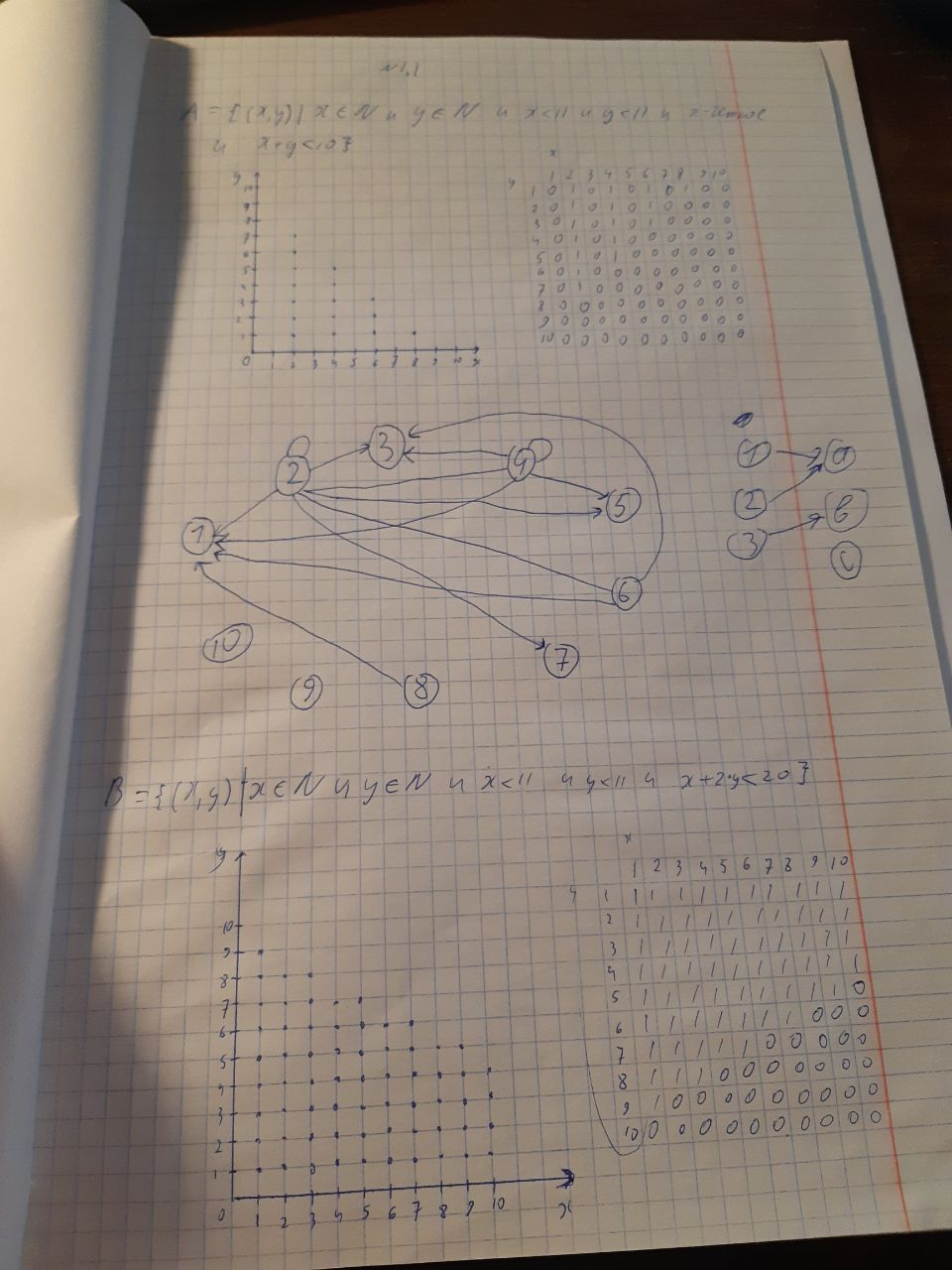
**Задания**

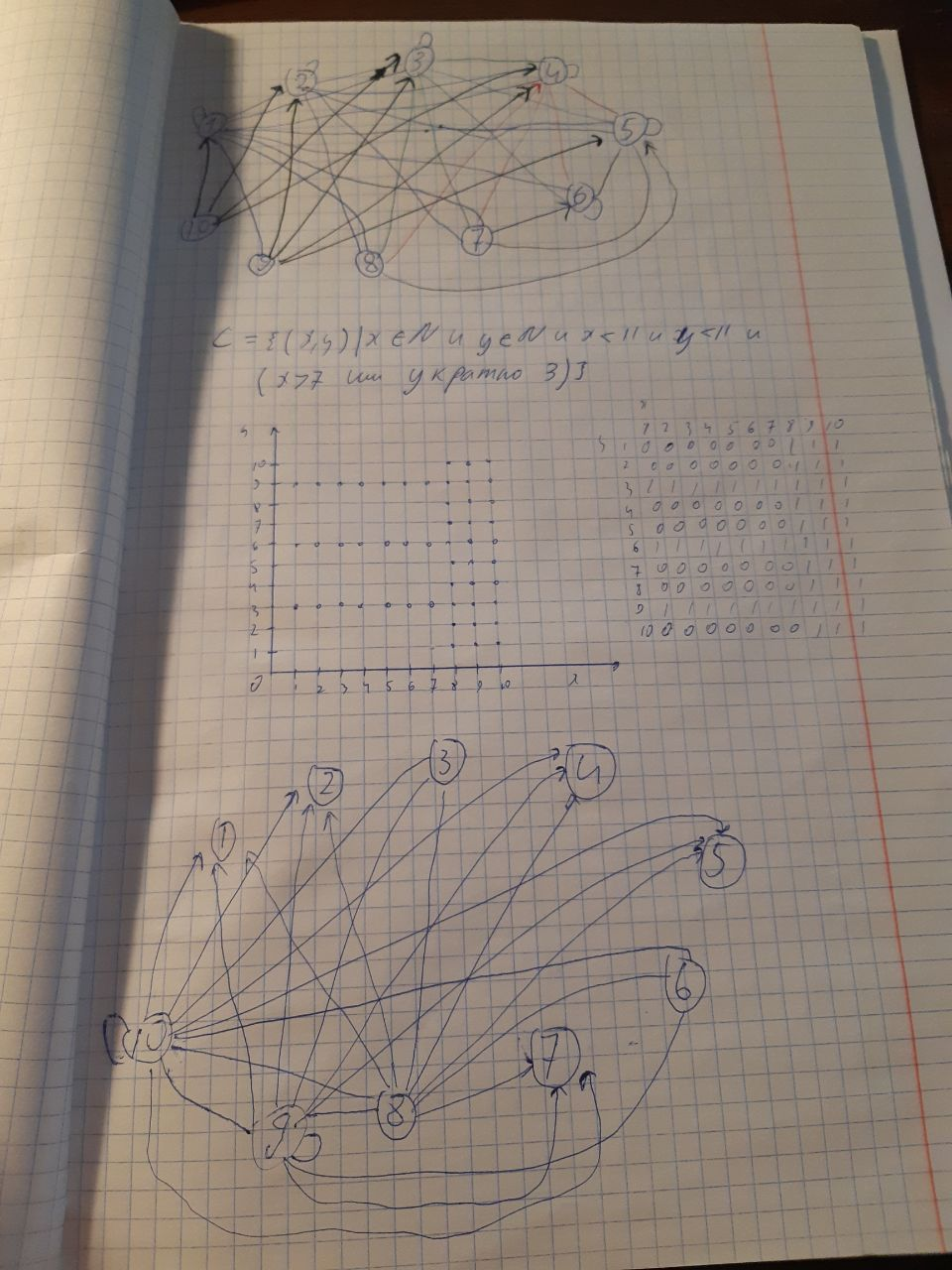
**Часть 1. Операции над отношениями**

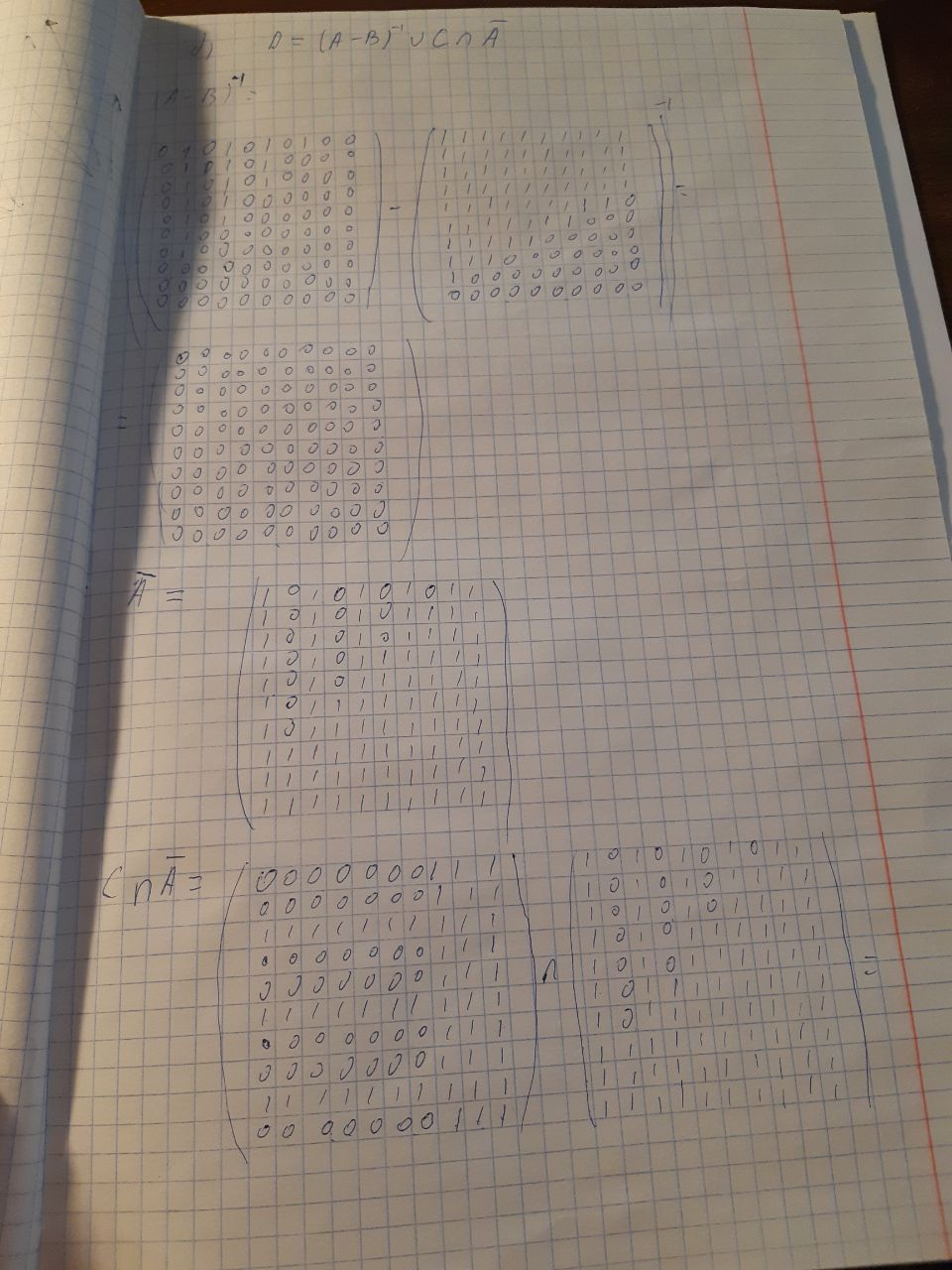
* 1. Представить отношения на множестве {1,2,3,4,5,6,7,8,9,10} (см.‖Варианты заданий‖, п.а) графиком, графом и матрицей.
  2. Вычислить значение выражения (см.‖Варианты заданий‖, п.б) при заданных отношениях (см.‖Варианты заданий‖, п.а).
  3. Написать программы, формирующие матрицы заданных отношений (см.‖Варианты заданий‖, п.а).
  4. Программно реализовать операции над отношениями.
  5. Написать программу, вычисляющую значение выражения (см. ―Варианты заданий‖, п.б) и вычислить его при заданных отношениях (см.‖Варианты заданий‖, п.а).

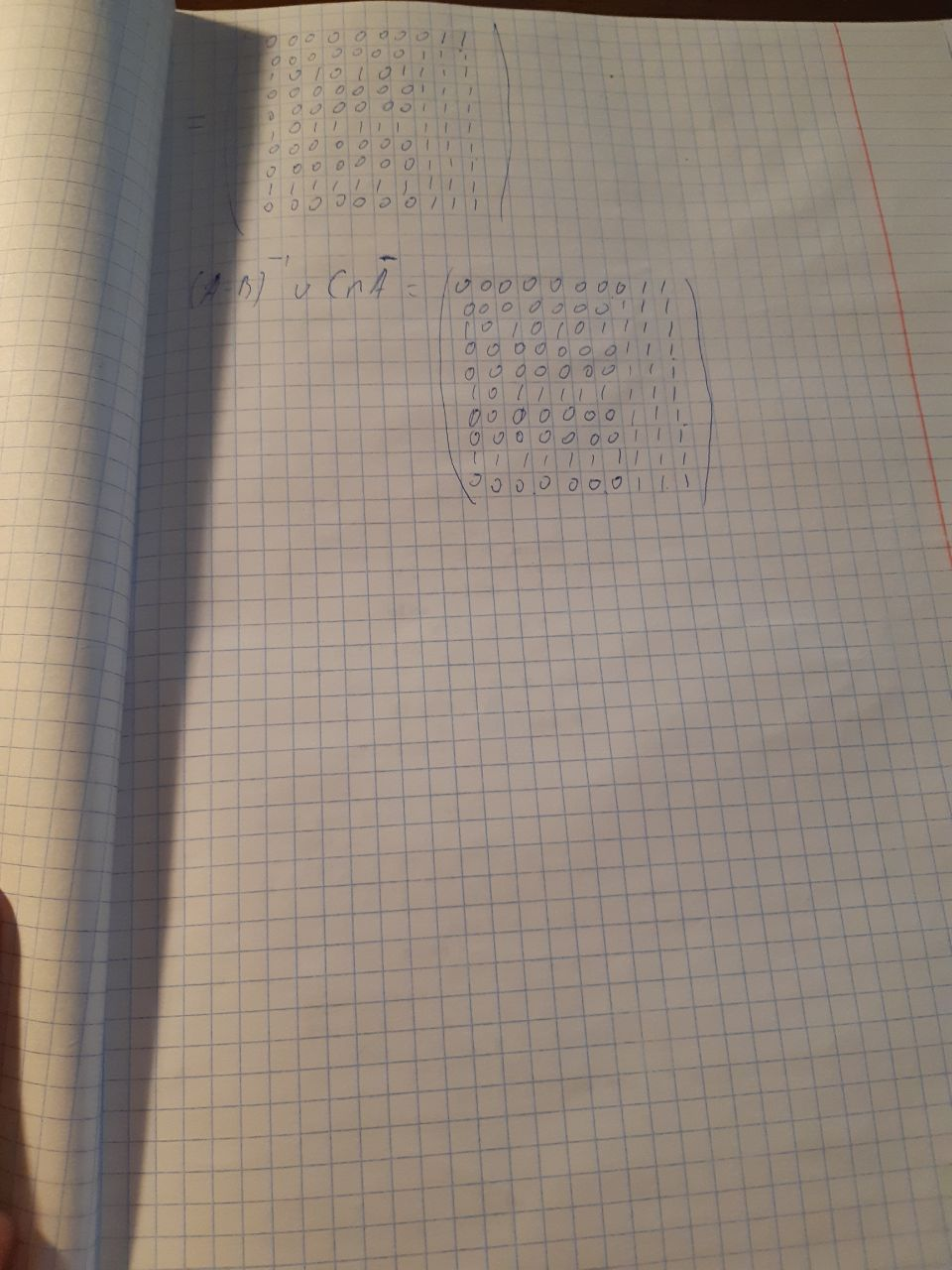


**Задание 1.1.**







**Задание 1.3**

#include <stdio.h> #include <stdlib.h> #define N 10

void new\_matrABC (char a[N][N], char b[N][N], char c[N][N])

***// Назначение: создает матрицы A, B, C размера N на N (N – const) из задания п.а.***

{

int i=0, j=0;

for (i=0;i<N;i++)

for(j=0;j<N;j++)

{

if ((i+j>8)&&((i+j)%2==0)) a[i][j]=1;

else a[i][j]=0;

}

for (i=0; i<N; i++)

for (j=0; j<N; j++)

{

if (((2\*j)+i) > 20) b[i][j]=1;

else b[i][j]=0;

}

for (i=0; i<N; i++)

for (j=0; j<N; j++)

{

if ((i==1)||(i==2)||(i==4)||(i==8))

{

if ((j==3)||(j==5)||(j==7)||(j==10))) c[i][j]=1;

}

}

}

void output\_matr(char a[N][N])

***// Назначение: выводит на экран матрицу A размера N на N, N – const.***

{

int i, j;

for (i=0;i<N;i++)

{

for (j=0;j<N;j++)

{

printf("%d", a[i][j]);

}

printf("\n");

}

printf("\n");

}

**Задание 1.4**

char raven (char a[N][N], char b[N][N])

***//Назначение: Возвращает 1, если матрицы отношений равны и 0 в противном случае***

{

size\_t i=0,j=0;

char f=1;

while(i<N && f)

{

j=1;

while(j<N && f)

{

f=a[i][j]==b[i][j];

j++;

}

i++;

}

return f;

}

char vkl (char a[N][N], char b[N][N])

***/\*Назначение: Возвращает 1, если матрица B включена в матрицу A и 0 в противном случае\*/***

{

char f=1, i=0,j=0;

while(i<N && f)

{

j=1;

while(j<N && f)

{

f=a[i][j]<=b[i][j];

j++;

}

i++;

}

return f;

}

void obed (char a[N][N], char b[N][N], char c[N][N])

***/\*Назначение: Записывает в матрицу С результат объединения матриц A и B\*/***

{

size\_t i=0, j=0;

for (i=0;i<N;i++)

for (j=0;j<N;j++)

c[i][j]=(a[i][j]+b[i][j]);

}

void peres (char a[N][N], char b[N][N], char c[N][N])

***/\*Назначение: Записывает в матрицу С результат пересечения матриц A и B\*/***

{

size\_t i=0, j=0;

for (i=0;i<N;i++)

for (j=0;j<N;j++)

c[i][j]=(a[i][j]\*b[i][j]);

}

void razn (char a[N][N], char b[N][N], char c[N][N])

***/\*Назначение: Записывает в матрицу С результат разности матриц A и B\*/***

{

size\_t i=0, j=0;

for (i=0;i<N;i++)

for (j=0;j<N;j++)

c[i][j]=(a[i][j]>b[i][j]);

}

void simrazn (char a[N][N], char b[N][N], char c[N][N])

***/\*Назначение: Записывает в матрицу С результат симметрической разности матриц A и B\*/***

{

size\_t i=0, j=0;

for (i=0;i<N;i++)

for (j=0;j<N;j++)

c[i][j]=(a[i][j]!=b[i][j]);

}

void dop (char a[N][N], char c[N][N])

***/\*Назначение: Записывает в матрицу С дополнение матрицы A \*/***

{

size\_t i=0, j=0;

for (i=0;i<N;i++)

for (j=0;j<N;j++)

c[i][j]=(1-a[i][j]);

}

void obrash (char a[N][N], char c[N][N])

***/\*Назначение: Записывает в матрицу С результат обращение матрицы А\*/***

{

size\_t i=0, j=0;

for (i=0;i<N;i++)

for (j=0;j<N;j++) c[i][j]=a[j][i];

}

void kompoz (char a[N][N], char b[N][N], char c[N][N])

***/\*Назначение: Записывает в матрицу С результат композиции матриц A и B\*/***

{

size\_t i=0, j=0, k=0;

char f=1;

for (i=0;i<N;i++)

for (j=0;j<N;j++)

{

c[i][j] = 0;

for (k=0; k<N; k++)

{

c[i][j]=(c[i][j]+a[i][j]\*b[i][j]);

}

}

}

**Задание 1.5**

void main()

***/\*Главная функция\*/***

{

char a[N][N], b[N][N], c[N][N], s[N][N], s2[N][N], s3[N][N], s4[N][N], s5[N][N];

new\_matrABC(a, b, c);

null\_matr(s);

null\_matr(s2);

null\_matr(s3);

null\_matr(s4);

null\_matr(s5);

obrash(b, s);

peres(a, s, s2);

kompoz(a, c, s3);

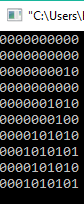
kompoz(s3, b, s4);

simrazn(s2, s3, s5);

output\_matr(s5);

}

Результат:



**Часть 2**

**Задание 2.3**

int refl(char a[N][N], int v) ***// Рефлексивность***

{

int i=0;

for (i=0; i<N; i++)

{

if (a[i][i]==0)

{

if (v) printf("Не рефлексивно, так как (%d,%d) не принадлежит отношению\n",i+1,i+1);

return 0;

}

}

if (v) printf("Рефлексивно\n");

return 1;

}

int anti\_refl(char a[N][N], int v)***//Антирефлексивность***

{

int i=0;

for (i=0; i<N; i++)

{

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

{

if (v) printf("Не антирефлексивно, так как (%d,%d) не принадлежит отношению\n",i+1,i+1);

return 0;

}

}

if (v) printf("Антирефлексивно\n");

return 1;

}

int simmetr (char a[N][N], int v)***//Симметричность***

{

int i=0, j=0;

for (i=0; i<N; i++)

for (j=0; j<N; j++)

{

if (a[i][j]!=a[j][i])

{

if (v) printf("Не симметрично, так как из %dR%d не следует %dR%d\n",i+1,j+1,j+1,i+1);

return 0;

}

}

if (v) printf("Симметрично\n");

return 1;

}

int anti\_simmetr (char a[N][N], int v)  ***//Антисимметричность***

{

int i=0, j=0;;

for (i=0; i<N; i++)

for (j=0; j<N; j++)

{

if (a[i][j]==a[j][i])

{

if (v) printf("Не антисимметрично, так как из %dR%d следует %dR%d\n",i+1,j+1,j+1,i+1);

return 0;

}

}

if (v) printf("Антимметрично\n");

return 1;

}

int polnoe (char a[N][N], int v) ***//Полное***

{

int i, j;

for (i=0; i<N; i++)

for(j=0; j<N; j++)

{

if (a[i][j]||a[j][i]!=1)

{

if (v) printf("Не полное\n");

return 0;

}

}

if (v) printf("Полное\n");

return 1;

}

int tranzit (char a[N][N], int v) ***//Транзитивное***

{

int i, j, k;

for (i=0; i<N; i++)

for (j=0; j<N; j++)

{

if (a[i][j]==0)

for (k=0; k<N; k++)

{

if (a[i][k]&&a[k][j])

{

if (v) printf("Не транзитивно, так как из %dR%d и %dR%d не следует %dR%d\n",i+1,k+1,k+1,j+1,i+1,j+1);

return 0;

}

}

}

if (v) printf("Транзитивно\n");

return 1;

}

int anti\_tranzit (char a[N][N], int v) ***//Антитранзитивное***

{

int i, j, k;

for (i=0; i<N; i++)

for (j=0; j<N; j++)

{

if (a[i][j]==1)

for (k=0; k<N; k++)

{

if (a[i][k]&&a[k][j])

{

if (v)

printf("Не антитранзитивно, так как из %dR%d и %dR%d следует %dR%d\n",i+1,k+1,k+1,j+1,i+1,j+1);

return 0;

}

}

}

if (v) printf("Антитранзитивно\n");

return 1;

}

int tolerant (char a[N][N])

{

return refl(a, 0)\*simmetr(a, 0);

}

int eqviv (char a[N][N])

{

return tolerant(a)\* tranzit(a, 0) ;

}

int poryadka (char a[N][N])

{

return anti\_tranzit(a, 0)\* tranzit(a, 0);

}

int nestr\_por (char a[N][N])

{

return refl(a, 0)\* poryadka(a);

}

int strog\_por (char a[N][N])

{

return poryadka(a)\* anti\_refl(a, 0);

}

int lin\_por (char a[N][N])

{

return poryadka(a)\* polnoe(a, 0);

}

int nestr\_lin\_por (char a[N][N])

{

return nestr\_por(a)\* polnoe(a, 0);

}

int str\_lin\_por (char a[N][N])

{

return strog\_por(a)\* polnoe(a, 0);

}

int main()

{

setlocale(LC\_ALL, "Rus");

char a[N][N], b[N][N], c[N][N], s[N][N], s2[N][N], s3[N][N], s4[N][N], s5[N][N];

new\_matrABC(a, b, c);

null\_matr(s); null\_matr(s2); null\_matr(s3); null\_matr(s4); null\_matr(s5);

obrash(b, s);

peres(a, s, s2);

kompoz(a, c, s3);

kompoz(s3, b, s4);

simrazn(s2, s3, s5);

output\_matr(s5);

printf("А:\n");

refl(a, 1); anti\_refl(a, 1); simmetr(a, 1); anti\_simmetr(a, 1); tranzit(a, 1); anti\_tranzit(a, 1); polnoe(a, 1);

if (tolerant(a)) printf("Толерантно\n");

if (eqviv(a)) printf("Эквивалентно\n");

if (poryadka(a))

{

printf("Порядка\n");

if (nestr\_por(a)) printf("Нестрогого порядка");

if (nestr\_lin\_por(a)) printf("Нестрого линейного порядка");

if (strog\_por(a)) printf("Строгого порядка");

if (str\_lin\_por(a)) printf("Строгого линейного порядка");

if (lin\_por(a)) printf("Линейного порядка");

}

printf("B:\n");

refl(b, 1); anti\_refl(b, 1); simmetr(b, 1); anti\_simmetr(b, 1); tranzit(b, 1); anti\_tranzit(b, 1); polnoe(b, 1);

if (tolerant(b)) printf("Толерантно\n");

if (eqviv(b)) printf("Эквивалентно\n");

if (poryadka(b))

{

printf("Порядка\n");

if (nestr\_por(b)) printf("Нестрогого порядка");

if (nestr\_lin\_por(b)) printf("Нестрого линейного порядка");

if (strog\_por(b)) printf("Строгого порядка");

if (str\_lin\_por(b)) printf("Строгого линейного порядка");

if (lin\_por(b)) printf("Линейного порядка");

}

printf("C:\n");

refl(c, 1); anti\_refl(c, 1); simmetr(c, 1); anti\_simmetr(c, 1); tranzit(c, 1); anti\_tranzit(c, 1); polnoe(c, 1);

if (tolerant(c)) printf("Толерантно\n");

if (eqviv(c)) printf("Эквивалентно\n");

if (poryadka(c))

{

printf("Порядка\n");

if (nestr\_por(c)) printf("Нестрогого порядка");

if (nestr\_lin\_por(c)) printf("Нестрого линейного порядка");

if (strog\_por(c)) printf("Строгого порядка");

if (str\_lin\_por(c)) printf("Строгого линейного порядка");

if (lin\_por(c)) printf("Линейного порядка");

}

}

