**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**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

дисциплина: Дискретная математика

тема: «Отношения и их свойства»

Выполнил: ст. группы ПВ-21

Донцов Александр Алексеевич

Проверил: Рязанов Ю.Д.

Белгород 2018

**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.

A = {(x,y) | x N и y N и x < 11 и y < 11 и (x и y четные)}

A = {(2,2), (2,4), (2,6), (2,8), (2,10), (4,2), (4,4), (4,6), (4,8), (4,10), (6,2), (6,4), (6,8), (6,10), (8,2), (8,4), (8,6), (8,8), (8,10), (10,2), (10,4), (10,6), (10,8), (10,10)}

0000000000

0101010101

0000000000

0101010101

0000000000

0101010101

0000000000

0101010101

0000000000

0101010101

B = {(x,y) | x N и y N и x < 11 и y < 11 и |x – y| < 5}

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

1111100000

1111110000

1111111000

1111111100

1111111110

0111111111

0011111111

0001111111

0000111111

0000011111



C = {(1;1), (1;2), (1;3), (1;4), (1;5), (1;6), (1;7), (1;8), (1;9), (1;10), (2;2), (2;4), (2;6), (2;8), (2;10), (3;3), (3;6), (3;9), (4;2), (4;4), (4;6), (4;8), (4;10), (5;5), (5;10), (6;6), (7;7), (8;8), (9;9), (10;10)}

1111111111

0101010101

0010010010

0101010101

0000100001

0000010000

0000001000

0001000100

0010010010

0000000001

1.2. Вычислить значение выражения (см.‖Варианты заданий‖, п.б) при заданных отношениях (см.‖Варианты заданий‖, п.а).



#include <stdio.h>

#include <stdlib.h>

#define N 10

//1.3. Написать программы, формирующие матрицы заданных отношений (см.Варианты заданий, п.а).

//A = {(x,y) | x ∈ N и y ∈ N и x < 11 и y < 11 и (x и y четные)}

void get\_A(int \*\*a){

int i,j;

for(i = 1; i <= N; i++)

for(j = 1; j <= N; j++)

if((i % 2 == 0)&&(j % 2 == 0))

a[i][j] = 1;

else

a[i][j] = 0;

return;

}

//B = {(x,y) | x ∈ N и y ∈ N и x < 11 и y < 11 и |x – y| < 5}

void get\_B(int \*\*b){

int i,j;

for(i = 1; i <= N; i++)

for(j = 1; j <= N; j++)

if(abs(i - j) < 5)

b[i][j] = 1;

else

b[i][j] = 0;

return;

}

//C = {(x,y) | x ∈ N и y ∈ N и x < 11 и y < 11 и y^2 кратно x}

void get\_C(int \*\*c){

int i,j;

for(i = 1; i <= N; i++)

for(j = 1; j <= N; j++)

if((j\*j) % i == 0)

c[i][j] = 1;

else

c[i][j] = 0;

return;

}

// отношение пустое ? 1:0

int isVoid(int \*\*a) {

for (int i = 1; i <= N; i++)

for (int j = 1; j <= N; j++)

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

return 0;

return 1;

}

// универсум ? 1:0

int isUniv(int \*\*a) {

for (int i = 1; i <= N; i++)

for (int j = 1; j <= N; j++)

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

return 0;

return 1;

}

//1.4. Программно реализовать операции над отношениями.

int ravn\_ot(int \*\*a, int \*\*b){ //равенство отношений

int i,j,f = 1;

for(i = 1; (i <= N)&&(f == 1); i++)

for(j = 1; (j <= N)&&(f == 1); j++)

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

return f;

}

int podmn\_ot(int \*\*a, int \*\*b){ //отношение а - подмножество b

int i,j,f = 1;

for(i = 1; (i <= N)&&(f == 1); i++)

for(j = 1; (j <= N)&&(f == 1); j++)

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

return f;

}

void ob\_ot(int \*\*a, int \*\*b, int \*\*c){ //объединение отношений

int i,j;

for(i = 1; i <= N; i++)

for (j = 1; j <= N; j++)

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

return;

}

void per\_ot(int \*\*a, int \*\*b, int \*\*c){ //пересечение отношений

int i,j;

for(i = 1; i <= N; i++)

for(j = 1; j <= N; j++)

c[i][j] = a[i][j]&&b[i][j];

return;

}

void razn\_ot(int \*\*a, int \*\*b, int \*\*c){ //разность отношений

int i,j;

for(i = 1; i <= N; i++)

for(j = 1; j <= N; j++)

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

return;

}

void sim\_razn\_ot(int \*\*a, int \*\*b, int \*\*c){ //симметрическая разность отношений

int i,j;

for(i = 1; i <= N; i++)

for(j = 1; j <= N; j++)

c[i][j] = !((a[i][j] == 1)&&(b[i][j] == 1));

return;

}

void dop\_ot(int \*\*a, int \*\*c){ //дополнение отношения

int i,j;

for(i = 1; i <= N; i++)

for(j = 1; j <= N; j++)

c[i][j] = !a[i][j];

return;

}

void obr\_ot(int \*\*a, int \*\*c){ //обращение отношения

int i,j;

for(i = 1; i <= N; i++)

for(j = 1; j <= N; j++)

c[i][j] = a[j][i];

return;

}

void comp\_ot(int \*\*a, int \*\*b, int \*\*c){ //композиция отношений

int i,j,k;

for(i = 1; i <= N; i++)

for(j = 1; j <= N; j++)

{

c[i][j] = 0;

for(k = 1; k <= N; k++)

c[i][j] = c[i][j] || (a[i][k] && b[k][j]);

}

return;

}

int\*\* get\_mem(){

int n = N + 1;

int \*\* res = (int \*\*)malloc(n \* sizeof(int \*));

for (int i = 0; i < n ; i++)

res[i] = (int \*)malloc(n \* sizeof(int));

return res;

}

void free\_mem(int \*\*a){

int n = N + 1;

for (int i = 0; i < n; i++)

free(a[i]);

free(a);

}

void write\_arr(int \*\*a){

int i,j;

for(i = 1; i <= N; i++){

for(j = 1; j <= N; j++)

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

printf("\n");

}

}

//-------------Лог. выражения-----------------

int identically(int \*\*a, int \*\*b) {//равенство

int f = 1;

int x = 1;

int y;

while (x <= N && f) {

y = 1;

while (y <= N && f) {

f = a[x][y] = b[x][y];

y++;

}

x++;

}

return f;

}

int inclusion(int \*\*a, int \*\*b) {//включение

int f = 1;

int x = 1;

int y;

while (x <= N && f) {

y = 1;

while (y <= N && f) {

f = a[x][y] <= b[x][y] ? 1 : 0;

y++;

}

x++;

}

return f;

}

//строгое включение

int sinclusion(int \*\*a, int \*\*b) {

if (inclusion(a, b) && !identically(a, b))

return 1;

else

return 0;

}

int \*\*creatTochd() {//пораждение тождественного отношения

int \*\*a = get\_mem();

for (int i = 1; i <= N; i++)

for (int j = 1; j <= N; j++)

a[i][j] = (i == j) ? 1 : 0;

return a;

}

//------------------Свойства отношений---------------------

int reflex(int \*\*r) { //рефлексивно+

int i = 1, j = 1;

int res = 1;

while(i <= N){

if(r[i][i] == 0){

res = 0;

printf("No petli %d ", i);

}

i++;

}

return res;

}

int areflex(int \*\*r) { //антирефлексивно+

int i = 1, j = 1;

int res = 1;

while(i <= N){

if(r[i][i] == 1){

res = 0;

printf("Petly %d ", i);

}

i++;

}

return res;

}

int simetr(int \*\*r) { //симметрично+

int i = 1, j = 1;

int res = 1;

while(i <= N){

while(j <= N){

if(r[i][j] != r[j][i]){

printf("No symmetry %d ", i);

res = 0;

}

j++;

}

i++;

}

return res;

}

int antSimetr(int \*\*r) { //антисимметрично+

int i = 1, j = 1;

int res = 1;

while(i <= N){

while(j <= N){

if(r[i][j] == r[j][i]){

printf("Symmetry %d ", i);

res = 0;

}

j++;

}

i++;

}

return res;

}

int trans(int \*\*r) { //транзитивно+

int i = 1, j = 1;

int res = 1;

int \*\*a = get\_mem();

comp\_ot(r, r, a);

while(i <= N){

while(j <= N){

if(a[i][j] && !r[i][j]){

res = 0;

printf("Only one train %d %d ", i, j);

}

j++;

}

i++;

}

free\_mem(a);

return res;

}

int antTrans(int \*\*r) { //антитранзитивно+

int i = 1, j = 1;

int res = 1;

int \*\*a = get\_mem();

comp\_ot(r, r, a);

while(i <= N){

while(j <= N){

if(!(a[i][j] && r[i][j])){

res = 0;

printf("No one train %d %d", i, j);

}

j++;

}

i++;

}

free\_mem(a);

return res;

}

int full(int \*\*r) { //полно+

int i = 1, j = 1;

int res = 1;

while(i <= N){

while(j <= N){

if(r[i][j] == 0){

printf("no train %d %d ", i, j);

res = 0;

}

j++;

}

i++;

}

return res;

}

//------производные свойст--------

// толерантное

int toller(int \*\*r) {

if (reflex(r) || simetr(r))

return 1;

else

return 0;

}

// эквивалентности

int equal(int \*\*r) {

if ((reflex(r) || simetr(r)) && trans(r))

return 1;

else

return 0;

}

// порядка

int poradok(int \*\*r) {

if (antSimetr(r) && trans(r))

return 1;

else

return 0;

}

// нестрогого порядка

int neStrogPoradok(int \*\*r) {

if (poradok(r) && reflex(r))

return 1;

else

return 0;

}

// строгого порядка

int StrogPoradok(int \*\*r) {

if (poradok(r) && areflex(r))

return 1;

else

return 0;

}

// линейного порядка

int LinePoradok(int \*\*r) {

if (poradok(r) && full(r))

return 1;

else

return 0;

}

// нестрогого линейного порядка

int neStrogLinePoradok(int \*\*r) {

if (neStrogPoradok(r) && full(r))

return 1;

else

return 0;

}

// строгого линейного порядка

int StrogLinePoradok(int \*\*r) {

if (StrogPoradok(r) && full(r))

return 1;

else

return 0;

}

//1.5. Написать программу, вычисляющую значение выражения (см. ―Варианты заданий, п.б) и вычислить его при заданных отношениях (см.Варианты заданий, п.а).

void main()

{

int \*\*a = get\_mem();

get\_A(a);

printf("A = \n");

write\_arr(a);

printf("\n\n");

int \*\*b = get\_mem();

get\_B(b);

printf("B = \n");

write\_arr(b);

printf("\n\n");

int \*\*c = get\_mem();

get\_C(c);

printf("C = \n");

write\_arr(c);

printf("\n\n");

int \*\*d = get\_mem();

printf("---- properties A ----\n");

printf("reflexively? %d\n",reflex(a));

printf("antireflexive? %d\n",areflex(a));

printf("symmetrically? %d\n",simetr(a));

printf("antisymmetrically? %d\n",antSimetr(a));

printf("transitive? %d\n",trans(a));

printf("antitransitively? %d\n",antTrans(a));

printf("full? %d\n",full(a));

printf("tolerant? %d\n",toller(a));

printf("equivalence? %d\n",equal(a));

printf("poradok? %d\n",poradok(a));

printf("neStrogPoradok? %d\n",neStrogPoradok(a));

printf("StrogPoradok? %d\n",StrogPoradok(a));

printf("LinePoradok? %d\n",LinePoradok(a));

printf("neStrogLinePoradok? %d\n",neStrogLinePoradok(a));

printf("StrogLinePoradok? %d\n",StrogLinePoradok(a));

printf("---- properties B ----\n");

printf("reflexively? %d\n",reflex(b));

printf("antireflexive? %d\n",areflex(b));

printf("symmetrically? %d\n",simetr(b));

printf("antisymmetrically? %d\n",antSimetr(b));

printf("transitive? %d\n",trans(b));

printf("antitransitively? %d\n",antTrans(b));

printf("full? %d\n",full(b));

printf("tolerant? %d\n",toller(b));

printf("equivalence? %d\n",equal(b));

printf("poradok? %d\n",poradok(b));

printf("neStrogPoradok? %d\n",neStrogPoradok(b));

printf("StrogPoradok? %d\n",StrogPoradok(b));

printf("LinePoradok? %d\n",LinePoradok(b));

printf("neStrogLinePoradok? %d\n",neStrogLinePoradok(b));

printf("StrogLinePoradok? %d\n",StrogLinePoradok(b));

printf("---- properties C ----\n");

printf("reflexively? %d\n",reflex(c));

printf("antireflexive? %d\n",areflex(c));

printf("symmetrically? %d\n",simetr(c));

printf("antisymmetrically? %d\n",antSimetr(c));

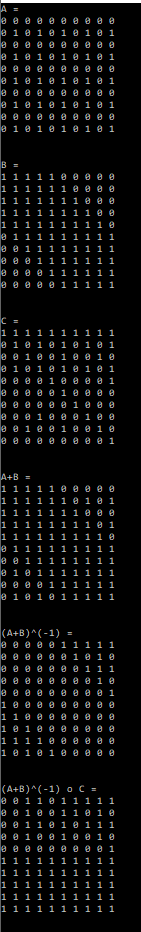
printf("transitive? %d\n",trans(c));

printf("antitransitively? %d\n",antTrans(c));

printf("full? %d\n",full(c));

printf("tolerant? %d\n",toller(c));

printf("equivalence? %d\n",equal(c));

 printf("poradok? %d\n",poradok(c));

printf("neStrogPoradok? %d\n",neStrogPoradok(c));

printf("StrogPoradok? %d\n",StrogPoradok(c));

printf("LinePoradok? %d\n",LinePoradok(c));

printf("neStrogLinePoradok? %d\n",neStrogLinePoradok(c));

printf("StrogLinePoradok? %d\n",StrogLinePoradok(c));

printf("\n\n");

ob\_ot(a,b,d);//объединение отношений

printf("A+B = \n");

write\_arr(d);

printf("\n\n");

dop\_ot(d, d);//дополнение отношений

printf("(A+B)^(-1) = \n");

write\_arr(d);

printf("\n\n");

comp\_ot(d, c, b);//композиция отношений

printf("(A+B)^(-1) o C = \n");

write\_arr(b);

printf("\n\n");

comp\_ot(a,a,d);//возведение в квадрат

printf("A^2 = \n");

write\_arr(d);

printf("\n\n");

sim\_razn\_ot(b,d, a);//симметрическая разность отношений

printf("(A+B)^(-1) o C (del) A^2 = \n");

write\_arr(a);

printf("\n\n");

free\_mem(a);

free\_mem(b);

free\_mem(c);

free\_mem(d);

return;

}

