**Задание 2.1.**

Определить основные свойства отношений (см. п.а).

Отношение А

Отношение антирефлексивно

Отношение не симметрично

Отношение не антисимметрично

Отношение транзитивно

Отношение не полно

Отношение B

Отношение антирефлексивно

Отношение не симметрично

Отношение не антисимметрично

Отношение транзитивно

Отношение не полно

Отношение C

Отношение не рефлексивно

Отношение не антирефлексивно

Отношение симметрично

Отношение не транзитивно

Отношение не антитранзитивно

Отношение не полно

**Задание 2.2.**

Определить, являются ли заданные отношения отношениями толерантности, эквивалентности и порядка.

Отношение R является отношением

— толерантности, если оно рефлексивно, симметрично;

— эквивалентности, если оно рефлексивно, симметрично и транзитивно;

— порядка, если оно антисимметрично и транзитивно.

То есть:

Отношение А

* антирефлексивно, не симметрично не является отношением толерантности
* антирефлексивно, не симметрично не является отношением эквивалентности
* не симметрично не является отношением порядка

Отношение В

* антирефлексивно, не симметрично не является отношением толерантности
* антирефлексивно, не симметрично не является отношением эквивалентности
* не симметрично не является отношением порядка

Отношение С

* не рефлексивно не является отношением толерантности
* не рефлексивно не является отношением эквивалентности
* не транзитивно не является отношением порядка

**Задание 2.3.**

Написать программу, определяющую свойства отношения, в том числе толерантности, эквивалентности и порядка, и определить свойства отношений (см. п.а).

#include <stdio.h>

#include <stdlib.h>

#include "locale.h"

//#define N 10+1

int main()

{

setlocale(LC\_ALL, "Rus");

int i, N=10+1;

int \*\*a;

a=(int \*\*)calloc(N,sizeof(int\*));

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

a[i]=(int \*)calloc(N,sizeof(int));

int \*\*b;

b=(int \*\*)calloc(N,sizeof(int\*));

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

b[i]=(int \*)calloc(N,sizeof(int));

int \*\*c;

c=(int \*\*)calloc(N,sizeof(int\*));

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

c[i]=(int \*)calloc(N,sizeof(int));

form\_A(a, N); printf ("\nОтношение А\n"); output (a,N);

form\_B(b, N); printf ("\nОтношение B\n"); output (b,N);

form\_C(c, N); printf ("\nОтношение C\n"); output (c,N);

printf ("\nОтношение А\n"); check (a,N);

printf ("\nОтношение B\n"); check (b,N);

printf ("\nОтношение C\n"); check (c,N);

}

int form\_A (int \*\*a, int N)

{

int x,y;

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

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

if ((x<y)&&(y<9-x) || ((9-x<y)&&(y<x)))

a[x][y]=1;

return 0;

}

int form\_B (int \*\*b, int N)

{

int x,y;

for (y=1;y<N;y+=2)

for (x=2;x<N;x+=2)

b[x][y]=1;

return 0;

}

int form\_C (int \*\*c, int N)

{

int x,y;

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

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

if (x\*y%3==0)

c[x][y]=1;

return 0;

}

void output (int \*\*a, int N)

{

int x,y;

printf ("\n");

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

{

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

printf ("%d ",a[x][y]);

printf ("\n");

}

printf ("\n");

}

//рефлексивность

int refl (int \*\*a, int N)

{

int i;

for (i=1;i<N && a[i][i];i++);

if (i==N)

printf ("\nОтношение рефлексивно\n");

else

printf ("\nОтношение не рефлексивно\n");

return (i==N);

}

//антирефлексивность

int arefl (int \*\*a, int N)

{

int i;

for (i=1;i<N && !a[i][i];i++);

if (i==N)

printf ("\nОтношение антирефлексивно\n");

else

printf ("\nОтношение не антирефлексивно\n");

return (i==N);

}

//симметричность

int simm (int \*\*a, int N)

{

int x,y,fl=1;

for (x=1;x<N && fl;x++)

for (y=x+1;y<N && fl;y++)

fl=a[x][y]==a[y][x];

if (fl)

printf ("\nОтношение симметрично\n");

else

printf ("\nОтношение не симметрично\n");

return fl;

}

//антисимметричность

int asimm (int \*\*a, int N)

{

int x,y,fl=1;

for (x=1;x<N-1 && fl;x++)

for (y=x+1;y<N && fl; y++)

fl= a[x][y]!=a[y][x];

if (fl)

printf ("\nОтношение антисимметрично\n");

else

printf ("\nОтношение не антисимметрично\n");

return fl;

}

//транзитивность

int tranz (int \*\*a, int N)

{

int x,y,z,fl=1;

for (x=1;x<N && fl;x++)

for (y=1;y<N && fl;y++)

for(z=1;z<N && fl;z++)

fl=!((a[x][z]&&a[z][y]))||a[x][y];

if (fl)

printf ("\nОтношение транзитивно\n");

else

printf ("\nОтношение не транзитивно\n");

return fl;

}

//антитранзитивность

int atranz (int \*\*a, int N)

{

int x,y,z,fl=1;

for (x=1;x<N && fl;x++)

for (y=1;y<N && fl;y++)

for(z=1;z<N && fl;z++)

fl=((a[x][z]&&a[z][y]))||a[x][y];

if (fl)

printf ("\nОтношение антитранзитивно\n");

else

printf ("\nОтношение не антитранзитивно\n");

return fl;

}

//полнота

int poln (int \*\*a, int N)

{

int x,y,fl=1;

for (x=1;x<N && fl;x++)

for (y=x+1;y<N && fl;y++)

fl = (a[x][y] || a[y][x]);

if (fl)

printf ("\nОтношение полно\n");

else

printf ("\nОтношение не полно\n");

return fl;

}

int check (int \*\*a, int N)

{

int r,t,s,ar,as,at,p;

r=refl(a,N);

ar=arefl(a,N);

s=simm(a,N);

as=asimm(a,N);

t=tranz(a,N);

at=atranz(a,N);

p=poln(a,N);

if(s&&r)

printf ("\nОтношение толерантности\n");

if (r&&s&&t)

printf("\nОтношение эквивалентности\n");

if (as&&t)

printf("\nОтношение порядка\n");

if (as&&t&&r)

printf("\nОтношение нестрогого порядка\n");

if (as&&t&&ar)

printf("\nОтношение строгого порядка\n");

if (as&&t&&p)

printf("\nОтношение линейного порядка\n");

if (as&&t&&r&&p)

printf("\nОтношение нестрогого линейного порядка\n");

if (as&&t&&ar&&p)

printf("\nОтношение строгого линейного порядка порядка\n");

}