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-223
Пахомов Владислав Андреевич

Проверили:
ст. пр. Рязанов Юрий Дмитриевич
ст. пр. Бондаренко Татьяна Владимировна

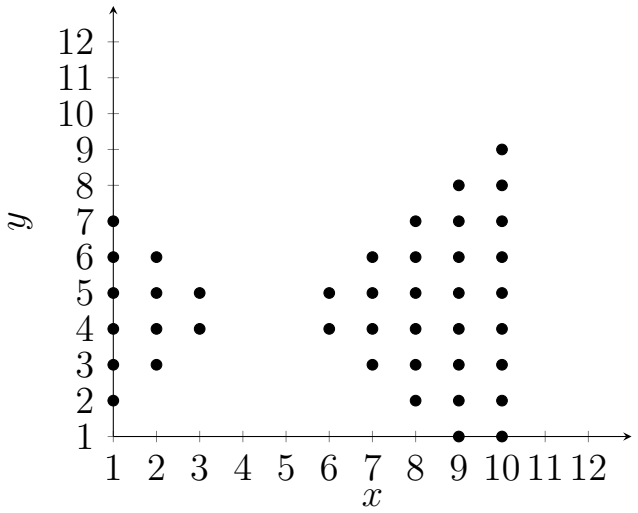
Белгород 2023 г.

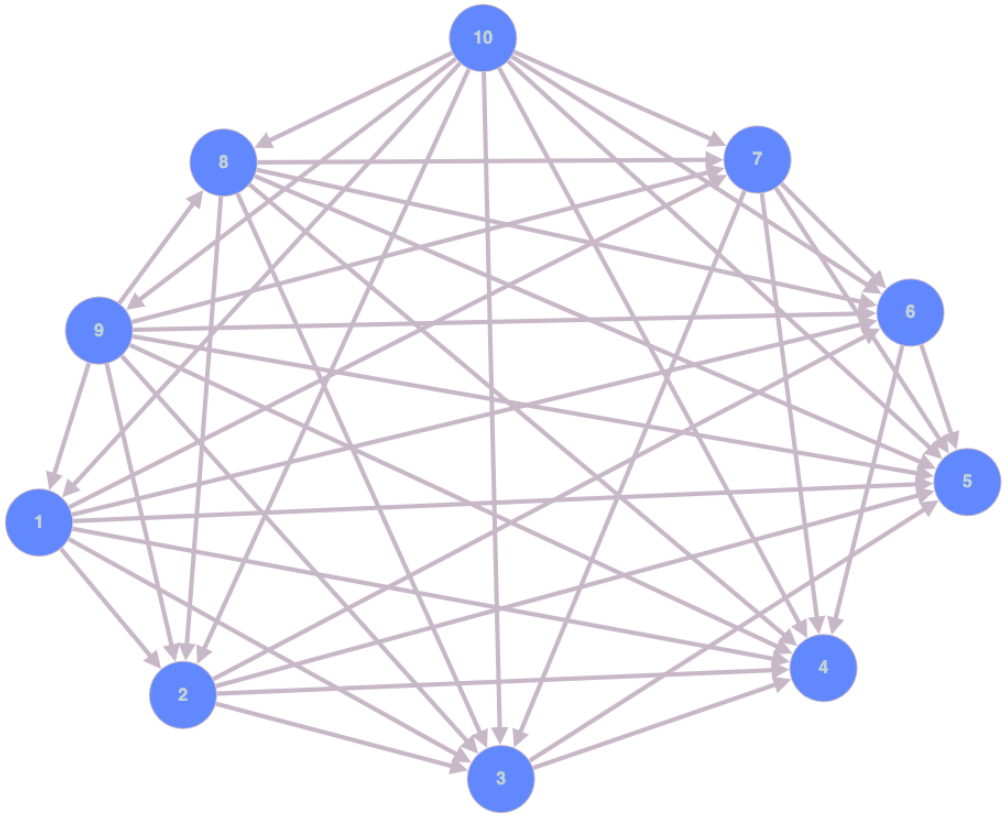
Лабораторная работа №3.1
Отношения и их свойства
Вариант 10

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

Часть 1. Операции над отношениями

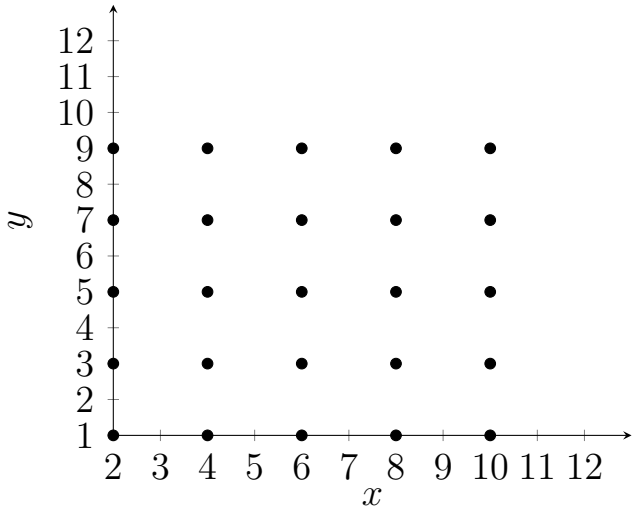
1.1. Представить отношения (см. ”Варианты заданий”, п.а) графиком, графом и матрицей.
$A = \{(x, y) | x \in N$ *и* $y \in N$ *и* $x < 11$ *и* $y < 11$ *и* $(x < y < (9 - x)$ *или* $(9 - x) < y < x)\}$

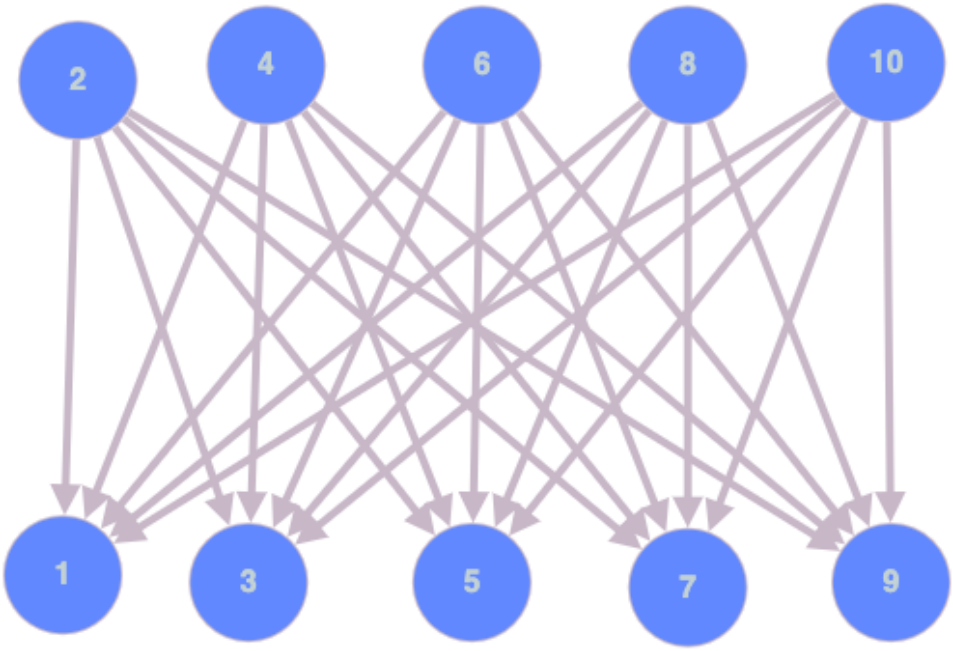




	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	1	1	1	1	1	1	0	0	0
2	0	0	1	1	1	1	0	0	0	0
3	0	0	0	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	1	1	0	0	0	0	0
7	0	0	1	1	1	1	0	0	0	0
8	0	1	1	1	1	1	1	0	0	0
9	1	1	1	1	1	1	1	1	0	0
10	1	1	1	1	1	1	1	1	1	0

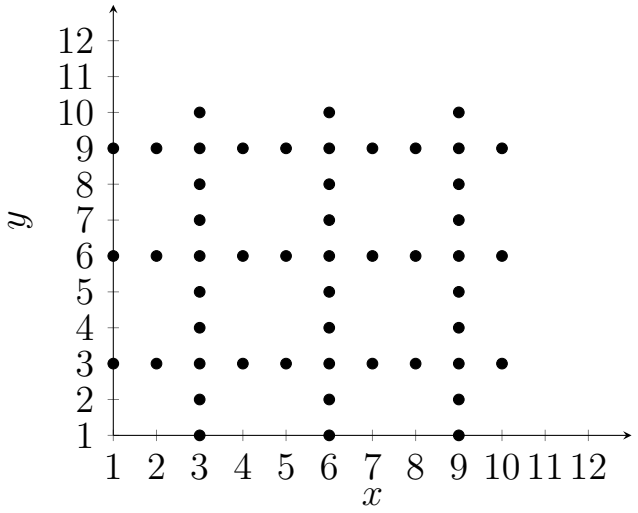
$B = \{(x, y) | x \in N$ *и* $y \in N$ *и* $x < 11$ *и* $y < 11$ *и* x *- чётно и* y *- нечётно*$\}$

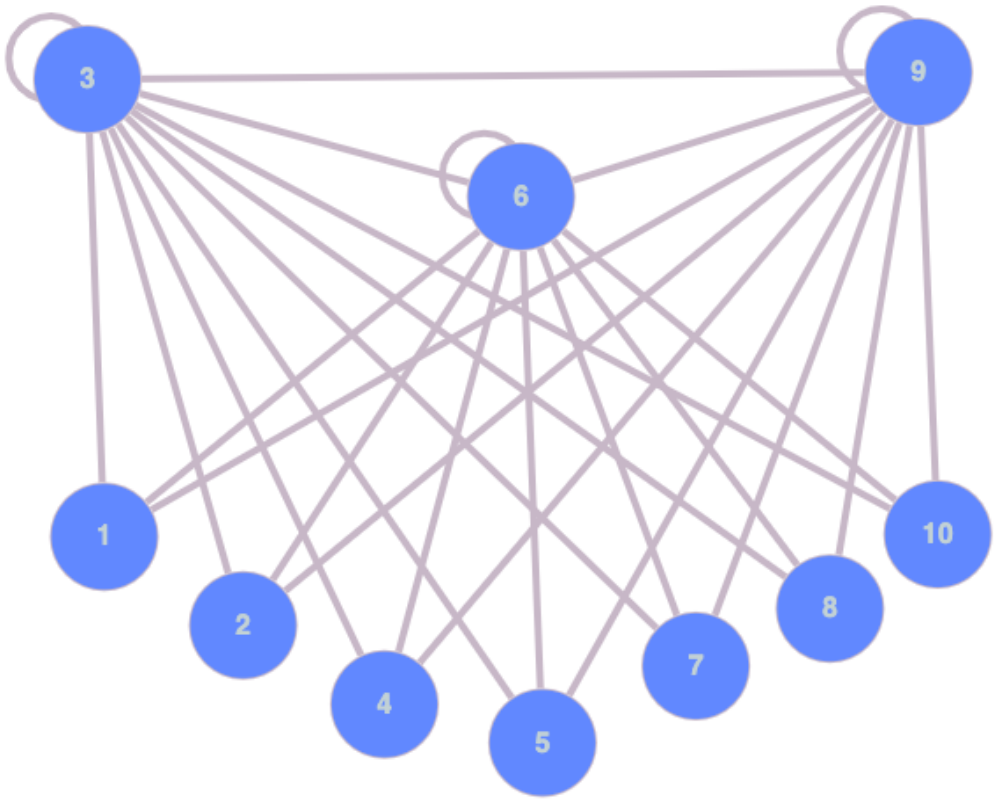




	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	1	0	1	0	1	0
3	0	0	0	0	0	0	0	0	0	0
4	1	0	1	0	1	0	1	0	1	0
5	0	0	0	0	0	0	0	0	0	0
6	1	0	1	0	1	0	1	0	1	0
7	0	0	0	0	0	0	0	0	0	0
8	1	0	1	0	1	0	1	0	1	0
9	0	0	0	0	0	0	0	0	0	0
10	1	0	1	0	1	0	1	0	1	0

$C = \{(x, y) | x \in N$ *и* $y \in N$ *и* $x < 11$ *и* $y < 11$ *и* $x \cdot y$ кратно трём$\}$





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

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

$D = A \circ B^2 - \overline{C} \cup C^{-1}$

$D = A \overset{1}{\circ} B \overset{2}{\circ} B \overset{5}{-} \overline{C}^{\,3} \overset{6}{\cup} C^{-1\,4}$

1) $A \circ B =$

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

2) $\_1 \circ B =$

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

3) $\overline{C} =$

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

4) $C^{-1} =$

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

5) $\_2 - \_3 =$

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

6) $\_5 \cup \_4 =$

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

$D =$

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

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

main.cpp

```
#include "../../libs/alg/alg.h"

bool predA(int x, int y) {
^^Ireturn (x < y && y < (9 - x)) || ((9 - x) < y && y < x);
}

bool predB(int x, int y) {
^^Ireturn x % 2 == 0 && y % 2 != 0;
```

```
}

bool predC(int x, int y) {
^^Ireturn (x * y) % 3 == 0 ;
}

int main() {
^^IBoolMatrixRelation a(10, predA);
^^Istd::cout << a << std::endl;
^^I
^^IBoolMatrixRelation b(10, predB);
^^Istd::cout << b << std::endl;
^^I
^^IBoolMatrixRelation c(10, predC);
^^Istd::cout << c << std::endl;
}
```

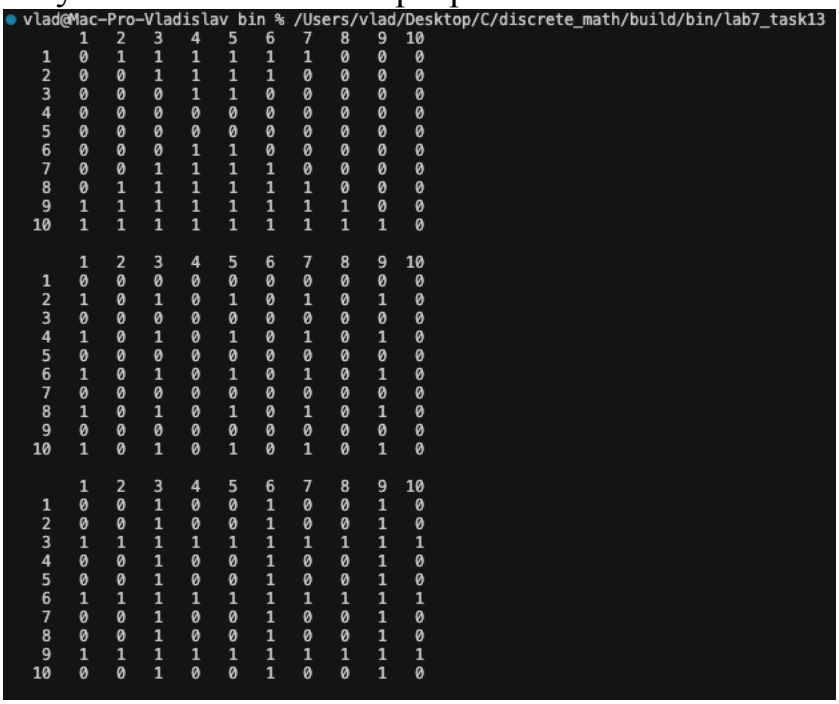
alg.h (объявление методов класса)

```
class BoolMatrixRelation
{
^^Iprivate:
^^Istd::vector<std::vector<bool>> data;
^^Iint size;
^^I
^^Ipublic:
^^IBoolMatrixRelation(const int size, bool (*pred)(int, int));
^^I~BoolMatrixRelation();
^^Ifriend std::ostream& operator<<(std::ostream& out, BoolMatrixRelation &val) {
^^I^^Iout << std::setw(3) << "" << " ";
^^I^^Ifor (int i = 1; i <= val.size; i++) {
^^I^^I^^Iout << std::setw(3) << i << " ";
^^I^^I}
^^I^^Iout << "\n";
^^I^^I
^^I^^Ifor (int x = 0; x < val.size; x++) {
^^I^^I^^Iout << std::setw(3) << x + 1 << " ";
^^I^^I^^Ifor (int y = 0; y < val.size; y++) {
^^I^^I^^I^^Iout << std::setw(3) << val.data[x][y] << " ";
^^I^^I^^I}
^^I^^I^^I
^^I^^I^^Iout << "\n";
^^I^^I}
^^I^^I
^^I^^Ireturn out;
^^I}
};
```

task13.cpp (реализация методов класса)

```
#include "../alg.h"

BoolMatrixRelation::BoolMatrixRelation(const int size, bool (*pred)(int, int)) {
^^Ithis->size = size;
^^I
^^Ifor (int x = 1; x <= size; x++) {
^^I^^Istd::vector<bool> val;
^^I^^I
^^I^^Ifor (int y = 1; y <= size; y++) {
^^I^^I^^Ival.push_back(pred(x, y));
^^I^^I}
^^I^^I
^^I^^Ithis->data.push_back(val);
^^I}
}

BoolMatrixRelation::~BoolMatrixRelation() {}
```

Результат выполнения программы:



1.4. Программно реализовать операции над отношениями.
Немного модифицируем класс BoolMatrixRelation. *alg.h* (объявление методов класса)

```
class BoolMatrixRelation
{
^^Iprivate:
^^Istd::vector<std::vector<bool>> data;
^^Iint size;
^^I
^^Istatic BoolMatrixRelation getDefault() {
^^I^^Ireturn BoolMatrixRelation();
^^I}
^^I
```

```
^^Ipublic:
^^IBoolMatrixRelation(const int size, std::function<bool (int, int)> pred);
^^IBoolMatrixRelation() {
^^I^^Ithis->size = 0;
^^I}
^^I~BoolMatrixRelation();
^^I
^^Ibool includes(BoolMatrixRelation b);
^^Ibool equals(BoolMatrixRelation b);
^^Ibool includesStrict(BoolMatrixRelation b);
^^IBoolMatrixRelation unite(BoolMatrixRelation b);
^^IBoolMatrixRelation intersect(BoolMatrixRelation b);
^^IBoolMatrixRelation diff(BoolMatrixRelation b);
^^IBoolMatrixRelation symDiff(BoolMatrixRelation b);
^^IBoolMatrixRelation non();
^^IBoolMatrixRelation transpose();
^^IBoolMatrixRelation compose(BoolMatrixRelation b);
^^IBoolMatrixRelation pow(int p);
^^I
^^Ifriend std::ostream& operator<<(std::ostream& out, BoolMatrixRelation &val) {
^^I^^Iout << std::setw(3) << "" << " ";
^^I^^Ifor (int i = 1; i <= val.size; i++) {
^^I^^I^^Iout << std::setw(3) << i << " ";
^^I^^I}
^^I^^Iout << "\n";
^^I^^I
^^I^^Ifor (int x = 0; x < val.size; x++) {
^^I^^I^^Iout << std::setw(3) << x + 1 << " ";
^^I^^I^^Ifor (int y = 0; y < val.size; y++) {
^^I^^I^^I^^Iout << std::setw(3) << val.data[x][y] << " ";
^^I^^I^^I}
^^I^^I^^I
^^I^^I^^Iout << "\n";
^^I^^I}
^^I^^I
^^I^^Ireturn out;
^^I}
};
```

task13.cpp (реализация методов класса)

```
#include "../alg.h"

BoolMatrixRelation::BoolMatrixRelation(const int size, std::function<bool (int, int)> pred)
{
^^Ithis->size = size;
^^I
^^Ifor (int x = 1; x <= size; x++)
^^I{
^^I^^Istd::vector<bool> val;
^^I^^I
^^I^^Ifor (int y = 1; y <= size; y++)
^^I^^I{
```

```
^^I^^I^^Ival.push_back(pred(x, y));
^^I^^I}
^^I^^I
^^I^^Ithis->data.push_back(val);
^^I}
}

BoolMatrixRelation::~BoolMatrixRelation() {}
```

task14.cpp (реализация методов класса)

```
#include "../alg.h"

bool BoolMatrixRelation::includes(BoolMatrixRelation b)
{
^^Iif (this->size != b.size) return false;
^^I
^^Ifor (int i = 0; i < size; i++) {
^^I^^Ifor (int j = 0; j < size; j++) {
^^I^^I^^Iif (data[i][j] && !b.data[i][j])
^^I^^I^^Ireturn false;
^^I^^I}
^^I}
^^Ireturn true;
}
bool BoolMatrixRelation::equals(BoolMatrixRelation b)
{
^^Iif (this->size != b.size) return false;
^^I
^^Ifor (int i = 0; i < size; i++) {
^^I^^Ifor (int j = 0; j < size; j++) {
^^I^^I^^Iif (data[i][j] != b.data[i][j])
^^I^^I^^Ireturn false;
^^I^^I}
^^I}
^^Ireturn true;
}
bool BoolMatrixRelation::includesStrict(BoolMatrixRelation b)
{
^^Ireturn (*this).includes(b) && !(*this).equals(b);
}
BoolMatrixRelation BoolMatrixRelation::unite(BoolMatrixRelation b)
{
^^Iif (this->size != b.size) return BoolMatrixRelation::getDefault();
^^I
^^Ireturn BoolMatrixRelation(size, [this, &b](int x, int y) {
^^I^^Ireturn data[x - 1][y - 1] || b.data[x - 1][y - 1];
^^I});
}
BoolMatrixRelation BoolMatrixRelation::intersect(BoolMatrixRelation b)
{
^^Iif (this->size != b.size) return BoolMatrixRelation::getDefault();
^^I
```

```
^^Ireturn BoolMatrixRelation(size, [this, &b](int x, int y) {
^^I^^Ireturn data[x - 1][y - 1] && b.data[x - 1][y - 1];
^^I});
}
BoolMatrixRelation BoolMatrixRelation::diff(BoolMatrixRelation b)
{
^^Iif (this->size != b.size) return BoolMatrixRelation::getDefault();
^^I
^^Ireturn BoolMatrixRelation(size, [this, &b](int x, int y) {
^^I^^Ireturn data[x - 1][y - 1] && !b.data[x - 1][y - 1];
^^I});
}
BoolMatrixRelation BoolMatrixRelation::symDiff(BoolMatrixRelation b)
{
^^Iif (this->size != b.size) return BoolMatrixRelation::getDefault();
^^I
^^Ireturn BoolMatrixRelation(size, [this, &b](int x, int y) {
^^I^^Ireturn data[x - 1][y - 1] ^ b.data[x - 1][y - 1];
^^I});
}
BoolMatrixRelation BoolMatrixRelation::non()
{
^^Ireturn BoolMatrixRelation(size, [this](int x, int y) {
^^I^^Ireturn !data[x - 1][y - 1];
^^I});
}
BoolMatrixRelation BoolMatrixRelation::transpose()
{
^^Ireturn BoolMatrixRelation(size, [this](int x, int y) {
^^I^^Ireturn data[y - 1][x - 1];
^^I});
}
BoolMatrixRelation BoolMatrixRelation::compose(BoolMatrixRelation b)
{
^^Iif (this->size != b.size) return BoolMatrixRelation::getDefault();
^^I
^^Ireturn BoolMatrixRelation(size, [this, &b](int x, int y) {
^^I^^Ifor (int z = 0; z < size; z++) {
^^I^^I^^Iif (data[x - 1][z] && b.data[z][y - 1])
^^I^^I^^Ireturn true;
^^I^^I}
^^I^^I
^^I^^Ireturn false;
^^I});
}
BoolMatrixRelation BoolMatrixRelation::pow(int p)
{
^^Iif (p < 0) return transpose();
^^Iif (p == 0) return BoolMatrixRelation(size, [](int x, int y){return x == y;});
^^Iif (p == 1) return *this;
^^I
^^IBoolMatrixRelation lowP = pow(p - 1);
^^Ireturn compose(lowP);
}
```

1.5. Написать программу, вычисляющую значение выражения (см. “Варианты заданий”, п.б) и вычислить его при заданных отношениях (см.”Варианты заданий”, п.а).
main.cpp

```
#include "../../libs/alg/alg.h"

bool predA(int x, int y) {
^^Ireturn (x < y && y < (9 - x)) || ((9 - x) < y && y < x);
}

bool predB(int x, int y) {
^^Ireturn x % 2 == 0 && y % 2 != 0;
}

bool predC(int x, int y) {
^^Ireturn (x * y) % 3 == 0 ;
}

int main() {
^^IBoolMatrixRelation a(10, predA);
^^IBoolMatrixRelation b(10, predB);
^^IBoolMatrixRelation c(10, predC);
^^IBoolMatrixRelation d = (((a.compose(b)).compose(b)).diff(c.non())).unite(c.pow(-1));
^^I
^^Istd::cout << d << std::endl;
}
```

Результат выполнения программы:

```
vlad@Mac-Pro-Vladislav bin % /Users/vlad/Desktop/C/discrete_math/build/bin/lab7_task15
    1  2  3  4  5  6  7  8  9  10
 1  0  0  1  0  0  1  0  0  1  0
 2  0  0  1  0  0  1  0  0  1  0
 3  1  1  1  1  1  1  1  1  1  1
 4  0  0  1  0  0  1  0  0  1  0
 5  0  0  1  0  0  1  0  0  1  0
 6  1  1  1  1  1  1  1  1  1  1
 7  0  0  1  0  0  1  0  0  1  0
 8  0  0  1  0  0  1  0  0  1  0
 9  1  1  1  1  1  1  1  1  1  1
10  0  0  1  0  0  1  0  0  1  0
```

Значение формулы D, вычисленное вручную, и результат выполненния программы совпали.

Часть 2. Свойства отношений

2.1. Определить основные свойства отношений (см. ”Варианты заданий”, п.а).

	A	B	C
Рефлексивность			
Антирефлексивность	+	+	
Симметричность			+
Антисимметричность	+	+	
Транзитивность	+	+	
Антитранзитивность		+	
Полнота			

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

	A	B	C
Толерантно			
Эквивалентно			
Порядка	+	+	
Нестрогого порядка			
Строгого порядка	+	+	
Линейного порядка			
Нестрогого линейного порядка			
Строгого линейного порядка			

2.3. Написать программу, определяющую свойства отношения, в том числе толерантности, эквивалентности и порядка, и определить свойства отношений (см. ”Варианты заданий”, п.а). *main.cpp*

```
#include "../../libs/alg/alg.h"

bool predA(int x, int y) {
^^Ireturn (x < y && y < (9 - x)) || ((9 - x) < y && y < x);
}

bool predB(int x, int y) {
^^Ireturn x % 2 == 0 && y % 2 != 0;
}

bool predC(int x, int y) {
^^Ireturn (x * y) % 3 == 0 ;
}

void outputProperties(std::string name, BoolMatrixRelation a) {
^^Istd::cout << "Properties for " << name << "\n";
^^Istd::cout << "Is reflexive? " << a.isReflexive() << "\n";
^^Istd::cout << "Is antireflexive? " << a.isAntiReflexive() << "\n";
^^Istd::cout << "Is symmetric? " << a.isSymmetric() << "\n";
^^Istd::cout << "Is antisymmetric? " << a.isAntiSymmetric() << "\n";
^^Istd::cout << "Is transitive? " << a.isTransitive() << "\n";
^^Istd::cout << "Is antitransitive? " << a.isAntiTransitive() << "\n";
^^Istd::cout << "Is full? " << a.isFull() << "\n";
^^Istd::cout << "Is tolerant? " << a.isTolerant() << "\n";
^^Istd::cout << "Is equivalent? " << a.isEquivalent() << "\n";
^^Istd::cout << "Is ordered? " << a.isOrdered() << "\n";
^^Istd::cout << "Is ordered non strict? " << a.isOrderedNonStrict() << "\n";
^^Istd::cout << "Is ordered strict? " << a.isOrderedStrict() << "\n";
^^Istd::cout << "Is ordered linear? " << a.isOrderedLinear() << "\n";
^^Istd::cout << "Is ordered linear non strict? " << a.isOrderedLinearNonStrict() << "\n";
^^Istd::cout << "Is ordered linear strict? " << a.isOrderedLinearStrict() << "\n" << std::endl;
}

int main() {
^^IBoolMatrixRelation a(10, predA);
^^IBoolMatrixRelation b(10, predB);
```

```
^^IBoolMatrixRelation c(10, predC);
^^I
^^IoutputProperties("A", a);
^^IoutputProperties("B", b);
^^IoutputProperties("C", c);
}
```

---

alg.h

```
class BoolMatrixRelation
{
^^Iprivate:
^^Istd::vector<std::vector<bool>> data;
^^Iint size;
^^I
^^Istatic BoolMatrixRelation getDefault() {
^^I^^Ireturn BoolMatrixRelation();
^^I}
^^I
^^Ipublic:
^^IBoolMatrixRelation(const int size, std::function<bool (int, int)> pred);
^^IBoolMatrixRelation() {
^^I^^Ithis->size = 0;
^^I}
^^I~BoolMatrixRelation();
^^I
^^Ibool includes(BoolMatrixRelation b);
^^Ibool equals(BoolMatrixRelation b);
^^Ibool includesStrict(BoolMatrixRelation b);
^^IBoolMatrixRelation unite(BoolMatrixRelation b);
^^IBoolMatrixRelation intersect(BoolMatrixRelation b);
^^IBoolMatrixRelation diff(BoolMatrixRelation b);
^^IBoolMatrixRelation symDiff(BoolMatrixRelation b);
^^IBoolMatrixRelation non();
^^IBoolMatrixRelation transpose();
^^IBoolMatrixRelation compose(BoolMatrixRelation b);
^^IBoolMatrixRelation pow(int p);
^^I
^^Istatic BoolMatrixRelation getIdentity(int size);
^^Istatic BoolMatrixRelation getUniversum(int size);
^^I
^^Ibool isEmpty();
^^I
^^Ibool isReflexive();
^^Ibool isAntiReflexive();
^^Ibool isSymmetric();
^^Ibool isAntiSymmetric();
^^Ibool isTransitive();
^^Ibool isAntiTransitive();
^^Ibool isFull();
^^Ibool isTolerant();
^^Ibool isEquivalent();
^^Ibool isOrdered();
```

```
^^Ibool isOrderedNonStrict();
^^Ibool isOrderedStrict();
^^Ibool isOrderedLinear();
^^Ibool isOrderedLinearNonStrict();
^^Ibool isOrderedLinearStrict();
^^I
^^Ifriend std::ostream& operator<<(std::ostream& out, BoolMatrixRelation &val) {
^^I^^Iout << std::setw(3) << "" << " ";
^^I^^Ifor (int i = 1; i <= val.size; i++) {
^^I^^I^^Iout << std::setw(3) << i << " ";
^^I^^I}
^^I^^Iout << "\n";
^^I^^I
^^I^^Ifor (int x = 0; x < val.size; x++) {
^^I^^I^^Iout << std::setw(3) << x + 1 << " ";
^^I^^I^^Ifor (int y = 0; y < val.size; y++) {
^^I^^I^^I^^Iout << std::setw(3) << val.data[x][y] << " ";
^^I^^I^^I}
^^I^^I^^I
^^I^^I^^Iout << "\n";
^^I^^I}
^^I^^I
^^I^^Ireturn out;
^^I}
};
```

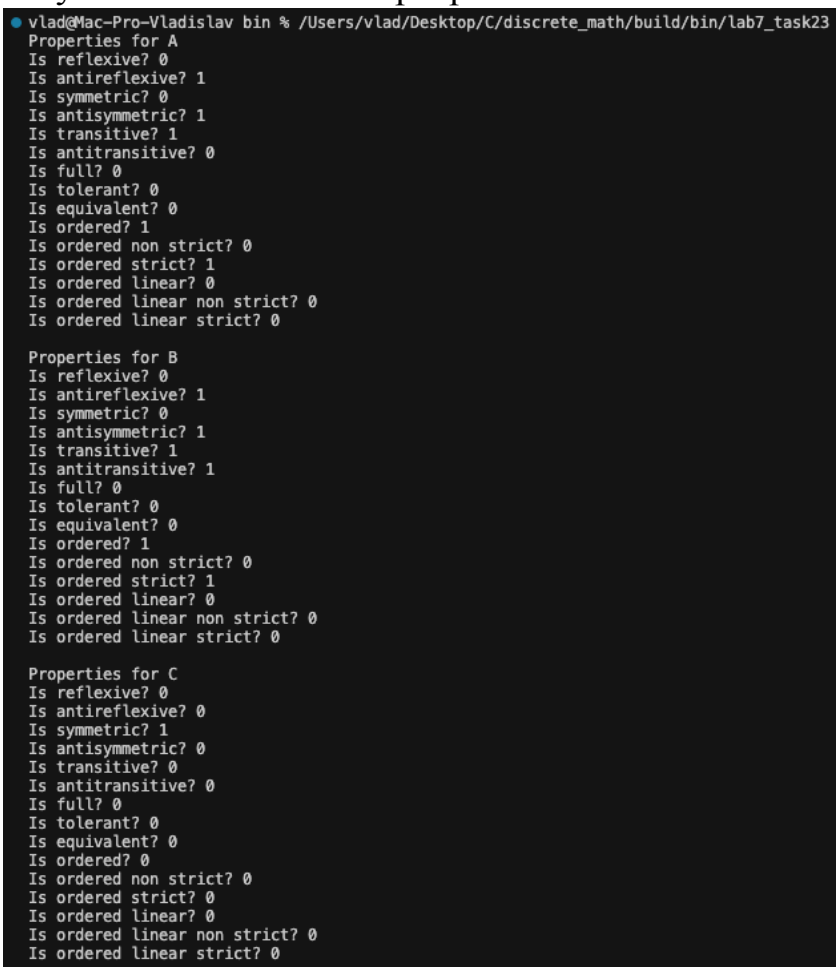
task23.cpp

```
#include "../alg.h"

bool BoolMatrixRelation::isEmpty() {
^^Ifor (int i = 0; i < size; i++) {
^^I^^Ifor (int j = 0; j < size; j++) {
^^I^^I^^Iif (data[i][j]) return false;
^^I^^I}
^^I}
^^I
^^Ireturn true;
}

BoolMatrixRelation BoolMatrixRelation::getIdentity(int size) {
^^Ireturn BoolMatrixRelation(size, [](int x, int y) {
^^I^^Ireturn x == y;
^^I});
}
BoolMatrixRelation BoolMatrixRelation::getUniversum(int size) {
^^Ireturn BoolMatrixRelation(size, [](int x, int y) {
^^I^^Ireturn true;
^^I});
}

bool BoolMatrixRelation::isReflexive()
{
```

```
^^Ireturn BoolMatrixRelation::getIdentity(size).includes(*this);
}
bool BoolMatrixRelation::isAntiReflexive()
{
^^Ireturn (*this).intersect(BoolMatrixRelation::getIdentity(size)).isEmpty();
}
bool BoolMatrixRelation::isSymmetric()
{
^^Ireturn (*this).equals((*this).pow(-1));
}
bool BoolMatrixRelation::isAntiSymmetric()
{
^^Ireturn (*this).intersect((*this).pow(-1)).includes(BoolMatrixRelation::getIdentity(size));
}
bool BoolMatrixRelation::isTransitive()
{
^^Ireturn ((*this).pow(2)).includes((*this));
}
bool BoolMatrixRelation::isAntiTransitive()
{
^^Ireturn ((*this).pow(2)).intersect((*this)).isEmpty();
}
bool BoolMatrixRelation::isFull()
{
^^Ireturn
↪  (*this).unite(BoolMatrixRelation::getIdentity(size)).unite((*this).pow(-1)).equals(BoolMatrixRelation::getUniversum(s
}
bool BoolMatrixRelation::isTolerant()
{
^^Ireturn isReflexive() && isSymmetric();
}
bool BoolMatrixRelation::isEquivalent()
{
^^Ireturn isReflexive() && isSymmetric() && isTransitive();
}
bool BoolMatrixRelation::isOrdered()
{
^^Ireturn isAntiSymmetric() && isTransitive();
}
bool BoolMatrixRelation::isOrderedNonStrict()
{
^^Ireturn isOrdered() && isReflexive();
}
bool BoolMatrixRelation::isOrderedStrict()
{
^^Ireturn isOrdered() && isAntiReflexive();
}
bool BoolMatrixRelation::isOrderedLinear()
{
^^Ireturn isOrdered() && isFull();
}
bool BoolMatrixRelation::isOrderedLinearNonStrict()
{
^^Ireturn isOrderedNonStrict() && isFull();
```

```
}
bool BoolMatrixRelation::isOrderedLinearStrict()
{
^^Ireturn isOrderedStrict() && isFull();
}
```

Результат выполнения программы:



Вывод: в ходе лабораторной работы изучили способы задания отношений, операции над отношениями и свойства отношений, научились программно реализовывать операции и определять свойства отношений.