Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

за 3 семестр

По дисциплине: «Дискретная математика»

Тема: «Отношения»

Выполнил:

Студент 2 курса

Группы ПО-6(1)

Мартынович Д. М.

Проверил:

Глущенко Т.А.

Брест 2021

Цель: изучить теорию по теме «Отношения», ознакомиться с понятиями, данными в теории. По данному варианту программно реализовать предложенные задачи.

Вариант 10

**Задание 1**

|  |  |  |
| --- | --- | --- |
| *№* | *Отношение* | *Отношение* |
| *10.* |  |  |

На множестве *A = {1,2,3,4,5,6,7,8,9,10}* заданы отношения  и  согласно вашему варианту. Варианты заданий указаны в *таблице 1*.

1. Для заданных отношений составить матрицы отношений. Построить орграфы отношений.

//посторение отношения R1

int\*\* R1(int\* A, int& num\_lines) {

int num\_columns = 2,

temp = 0,

\*\* R = new int\* [num\_lines];

for (int i = 0; i < num\_lines; i++) {

R[i] = new int[num\_columns];

}

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

if (A[i]%A[j]==4) {

R[temp][0] = A[i];

R[temp][1] = A[j];

temp++;

}

}

}

num\_lines = temp;

return R;

}

//посторение отношения R2

int\*\* R2(int\* A, int& num\_lines) {

int num\_columns = 2,

temp = 0,

\*\* R = new int\* [num\_lines];

for (int i = 0; i < num\_lines; i++) {

R[i] = new int[num\_columns];

}

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

if (A[i] + A[j] == 8) {

R[temp][0] = A[i];

R[temp][1] = A[j];

temp++;

}

}

}

num\_lines = temp;

return R;

}

//построение матриц

int\*\* matr(int\* A, int\*\* R, int num\_lines) {

cout << "Матрица отношения:" << endl;

int\*\* matr = new int\* [10];

for (int i = 0; i < 10; i++) {

matr[i] = new int[10];

}

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

matr[i][j] = 0;

}

}

for (int i = 0; i < num\_lines; i++) {

int x, y;

for (int j = 0; j < 10; j++) {

if (A[j] == R[i][0]) {

x = j;

}

if (A[j] == R[i][1]) {

y = j;

}

}

matr[x][y] = 1;

}

for (size\_t i = 0; i < num\_lines; i++)

{

for (size\_t j = 0; j < num\_lines; j++)

{

cout << matr[i][j];

}

cout << endl;

}

return matr;

}

Изображение выглядит как текст

Автоматически созданное описание

1. Найти обратные отношения и дополнения отношений.

int\*\* obratnoe(int\*\* matr) {

cout << "Обратное отноешние:" << endl;

int\*\* result;

result = new int\* [10];

for (int i = 0; i < 10; i++) {

result[i] = new int[10];

}

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

result[j][i] = matr[i][j];

}

}

for (size\_t i = 0; i < 10; i++)

{

for (size\_t j = 0; j < 10; j++)

{

cout << result[i][j];

}

cout << endl;

}

return result;

}

int\*\* dopolnit(int\*\* matr) {

cout << "Дополнение отноешния:" << endl;

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

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

matr[i][j] = 1;

}

else if (matr[i][j] == 1) {

matr[i][j] = 0;

}

}

}

for (size\_t i = 0; i < 10; i++)

{

for (size\_t j = 0; j < 10; j++)

{

cout << matr[i][j];

}

cout << endl;

}

return matr;

}

**

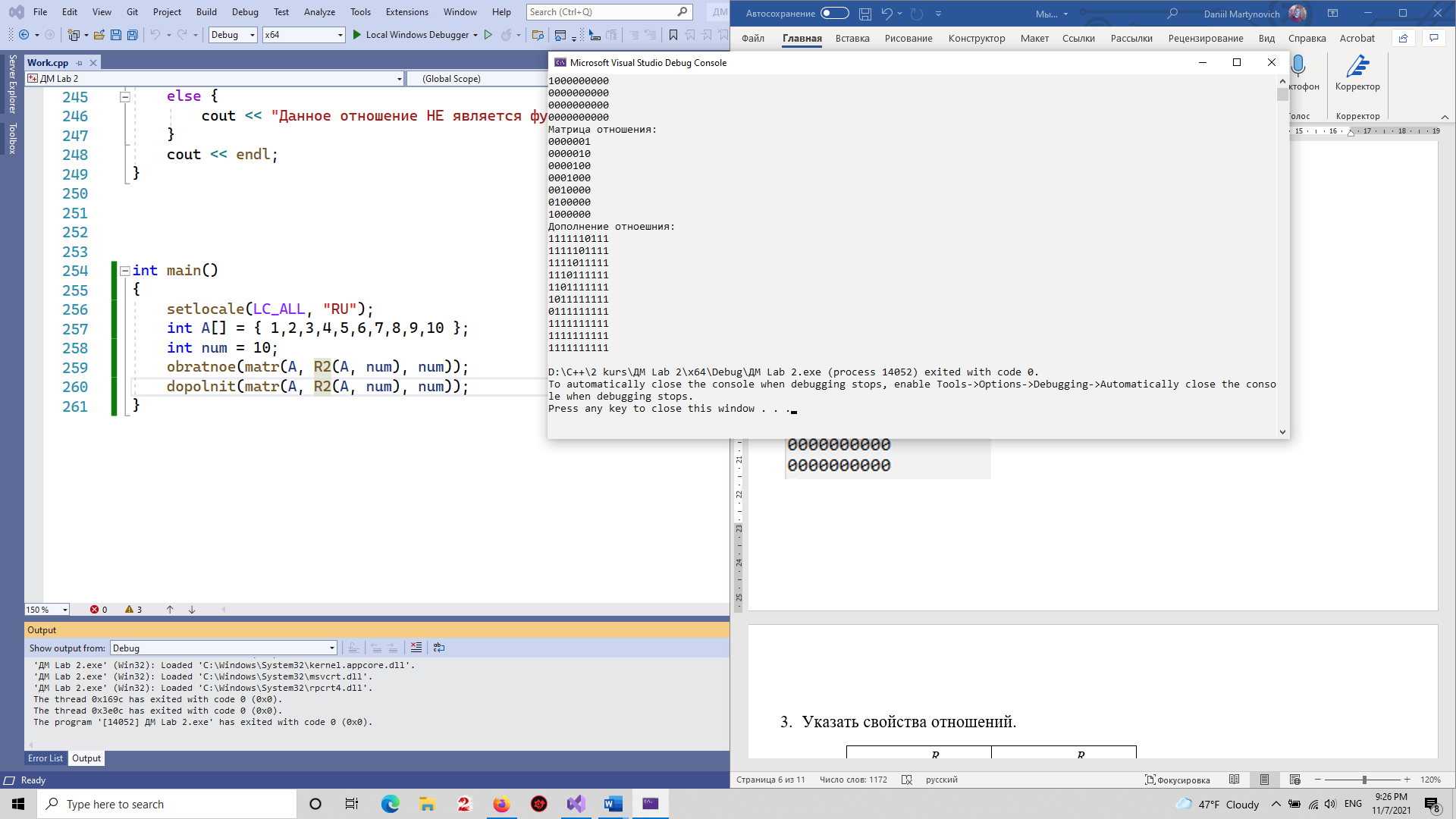
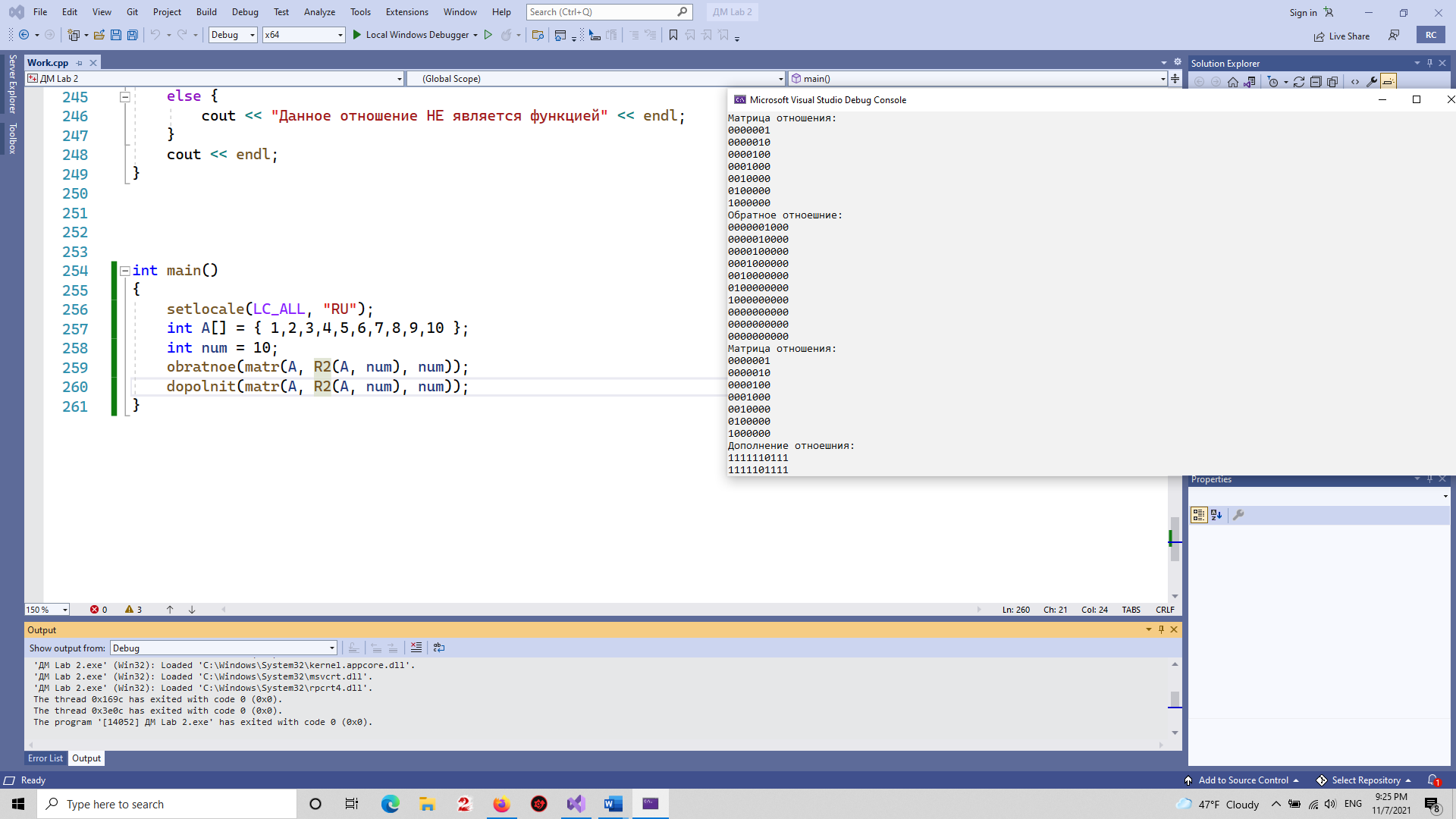
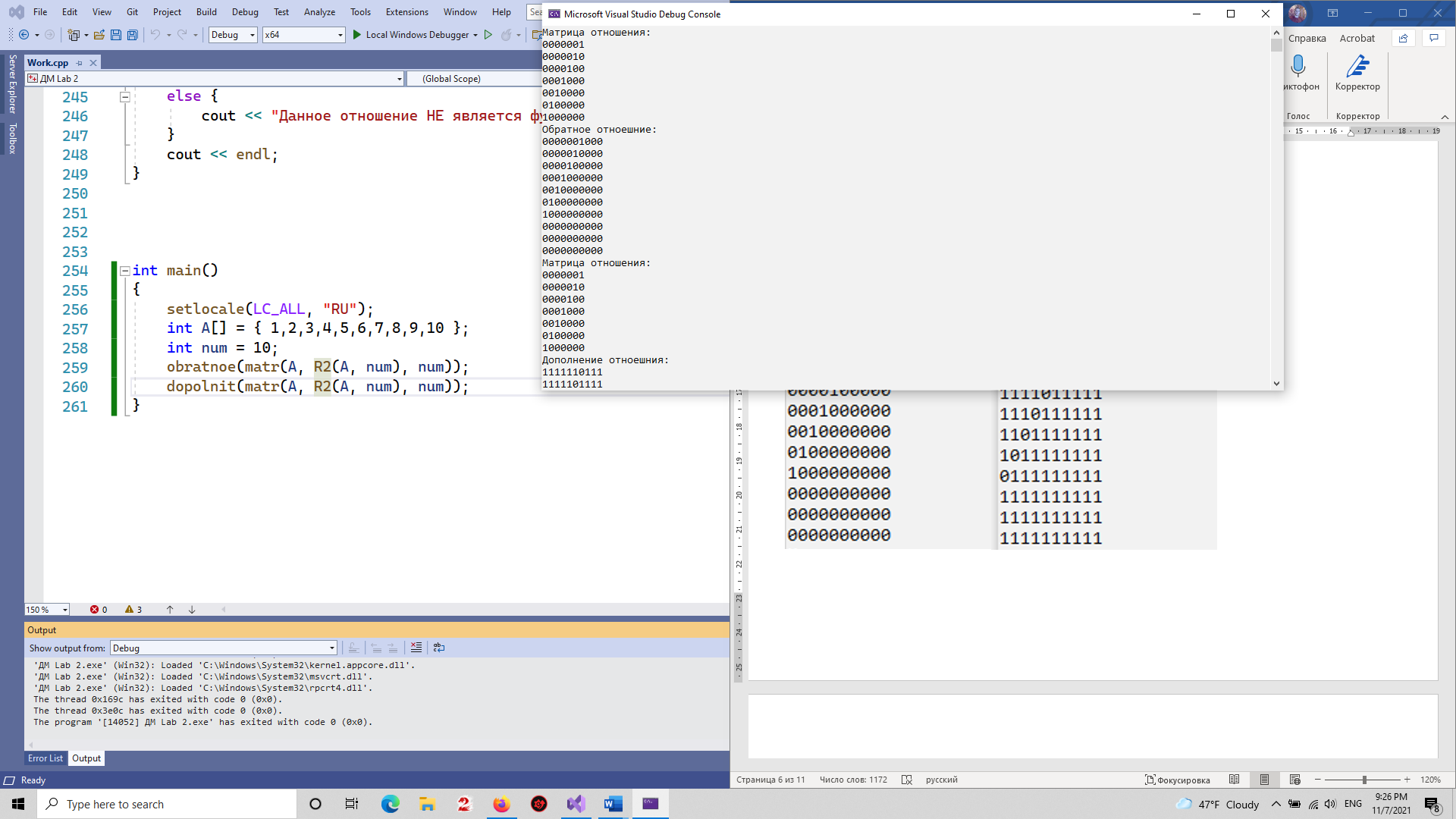
Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, ноутбук, компьютер

Автоматически созданное описание

**



1. Указать свойства отношений.

|  |  |
| --- | --- |
| Отношение | Отношение |
| Не рефлексивно  Антирефлексивно  Симметрично  Не антисимметрично  Не транзитивно | Не рефлексивно  Антирефлексивно  Симметрично  Не антисимметрично  Не транзитивно |

1. Для отношения, не обладающего свойством транзитивности, построить транзитивное замыкание алгоритмом *Флойда-Уоршолла*.

int\*\* trans(int\*\* matr) {

cout << "Транзитвное замыкание" << endl;

for (int k = 0; k < 10; k++) {

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

matr[i][j] = matr[i][j] || matr[i][k] && matr[k][j];

};

};

};

for (size\_t i = 0; i < 10; i++)

{

for (size\_t j = 0; j < 10; j++)

{

cout << matr[i][j];

}

cout << endl;

}

return matr;

}

Изображение выглядит как текст

Автоматически созданное описание

1. Найти композицию  и  отношений, определить обладает ли операция композиции отношений свойством коммутативности.

int\*\* compos(int\*\* R1, int\*\* R2) {

int\*\* R = new int\* [10];

for (int i = 0; i < 10; i++) {

R[i] = new int[10];

}

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

R[i][j] = 0;

}

}

cout << "Композиция:" << endl;

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

for (int k = 0; k < 10; k++) {

R[i][j] |= R2[i][k] & R1[k][j];

}

}

}

for (size\_t i = 0; i < 10; i++)

{

for (size\_t j = 0; j < 10; j++)

{

cout << R[i][j];

}

cout << endl;

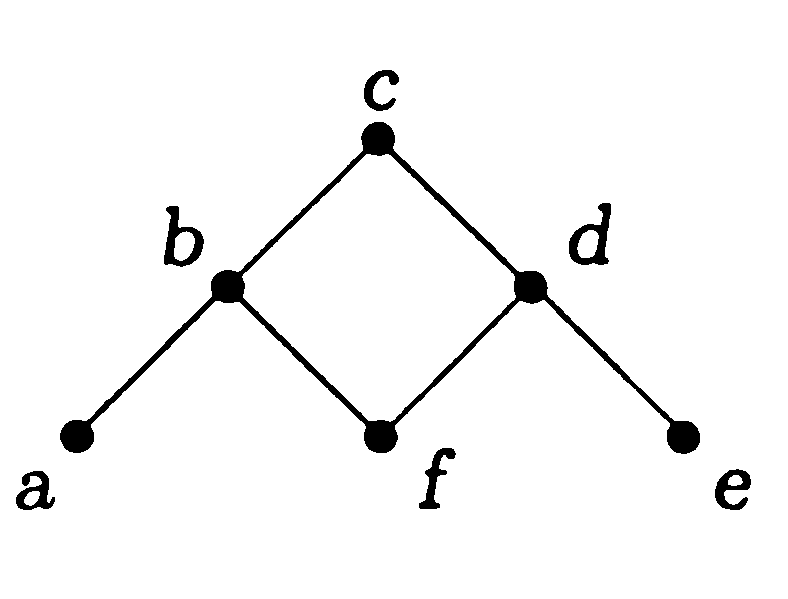
}

return R;

}

**Задание 2**

На подмножестве *A* натуральных чисел *N*,  задано отношение частичного порядка: *a* – делитель *b*. *ЧУ*-множество,  задано диаграммой *Хассе*. Указать:

 Изображение выглядит как часы

Автоматически созданное описание

1. Элементы *ЧУ*- множества .

{ 2,3,5,6,15,30 }

1. Пример сравнимых и несравнимых элементов.

Несравнимы: 2 и 3, 6 и 15, 2 и 15.

Сравнимы: 2 и 6, 3 и 15, 6 и 30.

1. Наибольший элемент *ЧУ-множества* (если он существует). 30
2. Наименьший элемент *ЧУ-множества* (если он существует). –
3. Максимальный и минимальный элементы *ЧУ-множества*.

Максимальный: 30.

Минимальные: 2, 3, 5.

1. Является ли рассматриваемое *ЧУ-множество* верхней или нижней полу

полурешеткой (или и тем, и другим)?

Верхняя полурешетка

**Часть 2**

На множествах *B = {1,2,3,4}* и *C = {5,6,7,8}* задано отношение *S*  согласно вашему варианту. Определить является ли заданное отношение функцией и если да, определить тип функции (инъекция, сюръекция, биекция). Задание проиллюстрировать графически

void process() {

int sum = 0,

B[4] = { 1,2,3,4 },

sizeB = 4,

C[4] = { 5,6,7,8 },

sizeC = 4,

S[4][2] = { {1,8},{2,7},{3,5},{4,6} },

matr[4][4];

bool inject = true,

surject = true;

cout << "Отношение (1,8),(2,7),(3,5),(4,6)" << endl;

for (int i = 0, k = 0; i < sizeB; i++) {

for (int j = 0; j < sizeC; j++) {

if (B[i] == S[k][0] && C[j] == S[k][1]) {

matr[i][j] = 1;

k++;

}

else matr[i][j] = 0;

cout << matr[i][j] << " ";

}cout << endl;

}

for (int i = 0; i < sizeB; i++) {

for (int j = 0; j < sizeC; j++) {

sum += matr[i][j];

}

}

if (sum == sizeB) {

cout << "Данное отношение является функцией. ";

sum = 0;

for (int i = 0; i < sizeC; i++) {

for (int j = 0; j < sizeB; j++) {

sum += matr[j][i];

}

if ((sum > 1) && (inject == true)) {

inject = false;

}

if ((sum == 0) && (surject == true)) {

surject = false;

}

sum = 0;

}

if (surject == true && inject == true) {

cout << "Биективная. ";

}

else

if (surject == true) {

cout << "Сюръективная. ";

cout << "Не инъективная. ";

}

else {

cout << "Не сюръективная. ";

if (inject == true) {

cout << "Инъективная. ";

}

else {

cout << "Не инъективная. ";

}

}

}

else {

cout << "Данное отношение НЕ является функцией" << endl;

}

cout << endl;

}

Изображение выглядит как текст

Автоматически созданное описание

***Задание 2.***

В коробке лежат 10 конфет в красных обертках, 10 в синих, 10 в желтых. Какое наименьшее число конфет нужно вытянуть не глядя, чтобы среди них обязательно оказались 2 конфеты в обертках:

Одного цвета 4

Разных цветов 11

Красного 22

Не синего 12

Вывод: изучил базовые понятия теории множеств и операции над множествами.