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 №3.4**

по дисциплине: Дискретная математика

тема: «Упорядоченные множества»

Выполнил: ст. группы ВТ-231

Масленников Даниил Александрович

Проверили:

Островский Алексей Мичеславович

Рязанов Юрий Дмитриевич

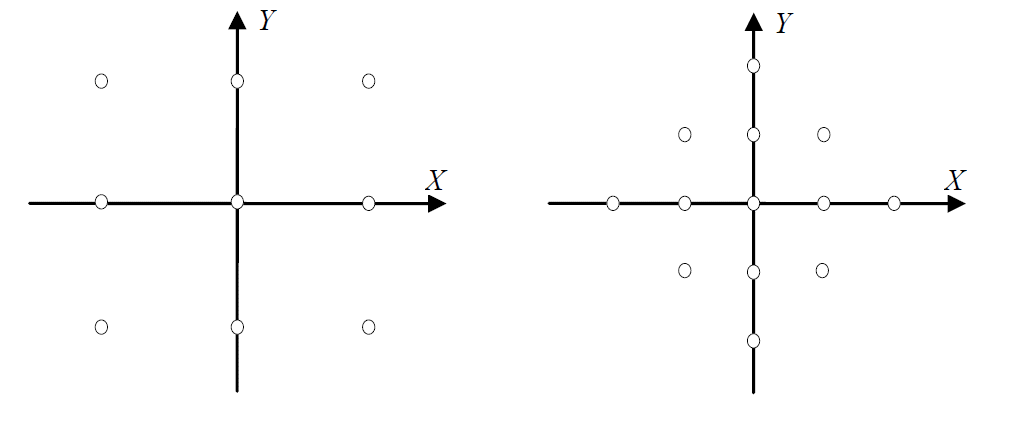
Белгород 2024 г.

Л а б о р а т о р н а я р а б о т а № 3.4

**Цель работы:** изучить упорядоченные множества, алгоритм топологической сортировки, научиться представлять множества диаграммами Хассе, находить минимальные (максимальные) и наименьшие (наибольшие) элементы упорядоченного множества.

**Вариант 12**

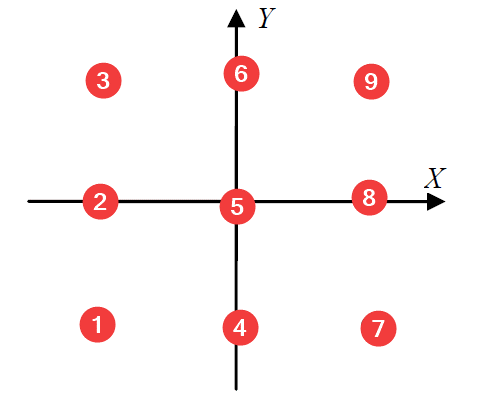
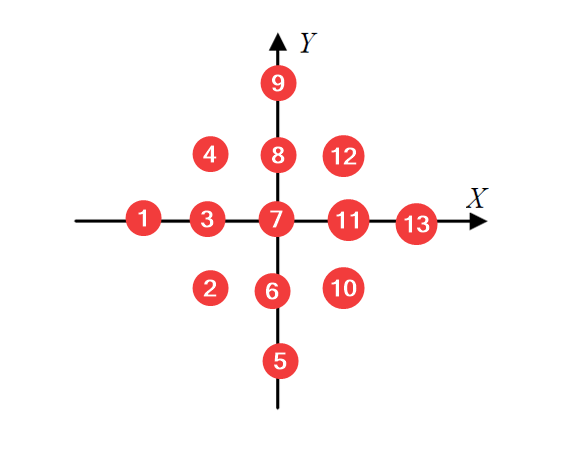
Даны множества точек на плоскости М1, М2 и отношение порядка. Для определения отношения на множестве точек примем следующие обозначения: ax — абсцисса точки a; ay — ордината a. На рис. 3 координаты правой верхней точки считать (1,1). Координаты самой верхней точки считать (0,2), а координаты самой правой точки считать (2,0).





**Задания**

1. Написать программы, формирующие матрицы отношений в соответствии с вариантом задания, на множествах М1 и М2.

Программа:

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX\_SIZE 100

void createRelationship(int matrix[MAX\_SIZE][2], int size, bool newMatrix[MAX\_SIZE][MAX\_SIZE]) {

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

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

int ax = matrix[i][0];

int ay = matrix[i][1];

int bx = matrix[j][0];

int by = matrix[j][1];

newMatrix[i][j] = (ax \* ay < bx \* by);

}

}

}

void matrixOutput(bool matrix[MAX\_SIZE][MAX\_SIZE], int size) {

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

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

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

}

printf("\n");

}

}

int main() {

int m1[9][2];

int index = 0;

for (int x = -1; x <= 1; x++) {

for (int y = -1; y <= 1; y++) {

m1[index][0] = x;

m1[index][1] = y;

index++;

}

}

int m2[13][2];

index = 0;

for (int x = -2; x <= 2; x++) {

for (int y = -2; y <= 2; y++) {

if ((abs(x) <= 1 && abs(y) <= 1) && !(y == 0 && x == 0)) {

m2[index][0] = x;

m2[index][1] = y;

index++;

}

}

}

bool matrixRel1[MAX\_SIZE][MAX\_SIZE] = { false };

bool matrixRel2[MAX\_SIZE][MAX\_SIZE] = { false };

createRelationship(m1, 9, matrixRel1);

createRelationship(m2, 13, matrixRel2);

printf("M1:\n");

matrixOutput(matrixRel1, 9);

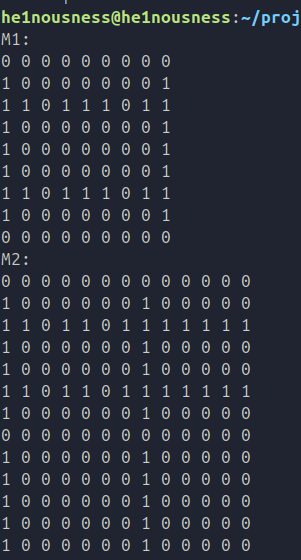
printf("M2:\n");

matrixOutput(matrixRel2, 13);

return 0;

}

Результат работы программы:



2. Написать программы, формирующие матрицы отношения доминирования по матрицам отношения порядка.

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX\_SIZE 100

bool createDominance(bool matrix[MAX\_SIZE][MAX\_SIZE], int size, bool newMatrix[MAX\_SIZE][MAX\_SIZE]) {

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

matrix[i][i] = 0;

}

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

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

if (matrix[i][j] == 1) {

bool flag = true;

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

if (matrix[i][k] && matrix[k][j]) {

flag = false;

break;

}

}

if (flag) {

newMatrix[i][j] = 1;

} else {

newMatrix[i][j] = 0;

}

} else {

newMatrix[i][j] = 0;

}

}

}

return true;

}

void matrixOutput(bool matrix[MAX\_SIZE][MAX\_SIZE], int size) {

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

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

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

}

printf("\n");

}

}

int main() {

int m1[9][2];

int index = 0;

for (int x = -1; x <= 1; x++) {

for (int y = -1; y <= 1; y++) {

m1[index][0] = x;

m1[index++][1] = y;

}

}

int m2[13][2];

index = 0;

for (int x = -2; x <= 2; x++) {

for (int y = -2; y <= 2; y++) {

if ((abs(x) <= 1 && abs(y) <= 1) || (y == 0 || x == 0)) {

m2[index][0] = x;

m2[index++][1] = y;

}

}

}

bool matrixRel1[MAX\_SIZE][MAX\_SIZE] = { false };

bool matrixRel2[MAX\_SIZE][MAX\_SIZE] = { false };

bool matrixDom1[MAX\_SIZE][MAX\_SIZE] = { false };

bool matrixDom2[MAX\_SIZE][MAX\_SIZE] = { false };

createRelationship(m1, 9, matrixRel1);

createRelationship(m2, 13, matrixRel2);

createDominance(matrixRel1, 9, matrixDom1);

createDominance(matrixRel2, 13, matrixDom2);

printf("M1 Dominance:\n");

matrixOutput(matrixDom1, 9);

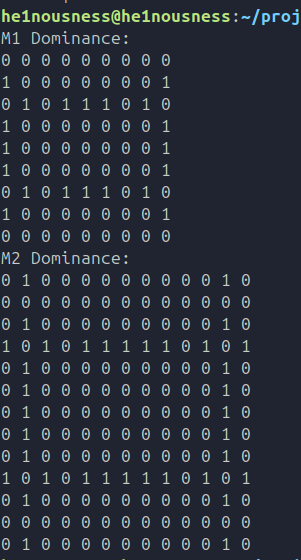
printf("M2 Dominance:\n");

matrixOutput(matrixDom2, 13);

return 0;

}

Результат работы программы:



3. Написать программу, реализующую алгоритм топологической сортировки по матрице отношения доминирования.

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <math.h>

#define MAX\_SIZE 100

void sortTopological(bool matrix[MAX\_SIZE][MAX\_SIZE], int size, int newMatrix[MAX\_SIZE]) {

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

int counter = 0;

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

counter += matrix[j][i];

}

newMatrix[i] = counter;

}

}

void sortTopologicalOutput(int matrix1[MAX\_SIZE], int size, int matrix2[MAX\_SIZE][2]) {

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

bool flagOfNull = 0;

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

if (matrix1[j] == 0) {

flagOfNull = 1;

}

}

if (flagOfNull) {

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

matrix1[j]--;

}

} else {

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

if (matrix1[j] > 0) {

matrix1[j]--;

}

}

}

}

int val = -1;

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

bool flag = 0;

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

if (matrix1[j] == val) {

flag = 1;

break;

}

}

if (flag) {

printf("%d level: ", i);

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

if (matrix1[j] == val) {

printf("%d ", j + 1);

}

}

printf("\n");

}

val--;

}

}

int main() {

int m1[9][2];

int index = 0;

for (int x = -1; x <= 1; x++) {

for (int y = -1; y <= 1; y++) {

m1[index][0] = x;

m1[index++][1] = y;

}

}

int m2[13][2];

index = 0;

for (int x = -2; x <= 2; x++) {

for (int y = -2; y <= 2; y++) {

if (abs(x) <= 1 && abs(y) <= 1 || (y == 0 || x == 0)) {

m2[index][0] = x;

m2[index++][1] = y;

}

}

}

bool matrixRel1[MAX\_SIZE][MAX\_SIZE] = { 0 };

bool matrixRel2[MAX\_SIZE][MAX\_SIZE] = { 0 };

createRelationship(m1, 9, matrixRel1);

createRelationship(m2, 13, matrixRel2);

bool matrixDom1[MAX\_SIZE][MAX\_SIZE] = { 0 };

bool matrixDom2[MAX\_SIZE][MAX\_SIZE] = { 0 };

createDominance(matrixRel1, 9, matrixDom1);

createDominance(matrixRel2, 13, matrixDom2);

int matrixTop1[MAX\_SIZE] = { 0 };

int matrixTop2[MAX\_SIZE] = { 0 };

sortTopological(matrixDom1, 9, matrixTop1);

sortTopological(matrixDom2, 13, matrixTop2);

printf("M1 Topological Sort:\n");

sortTopologicalOutput(matrixTop1, 9, m1);

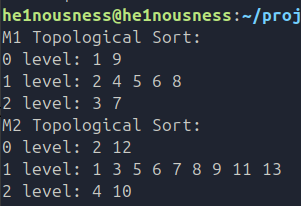
printf("M2 Topological Sort:\n");

sortTopologicalOutput(matrixTop2, 13, m2);

return 0;

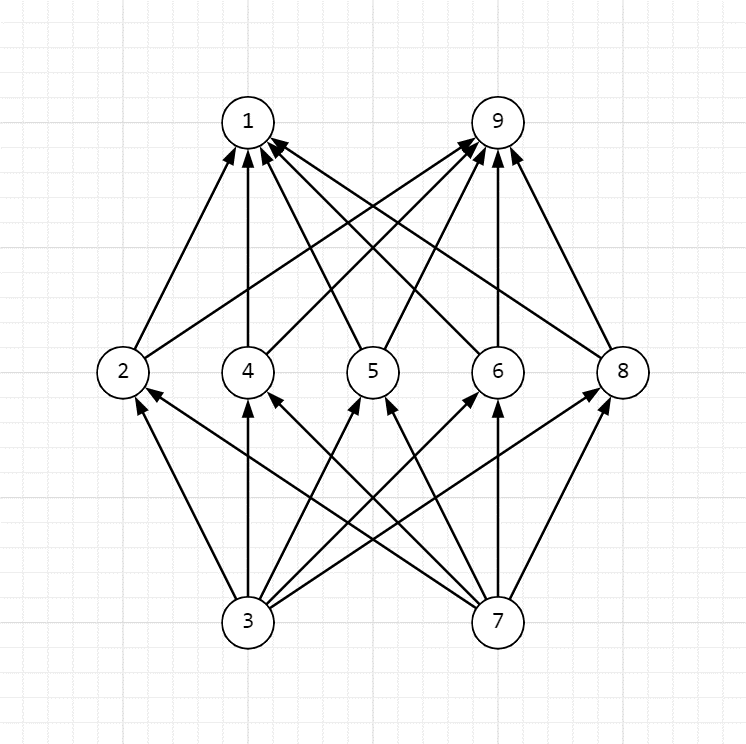
}

Результат работы программы:

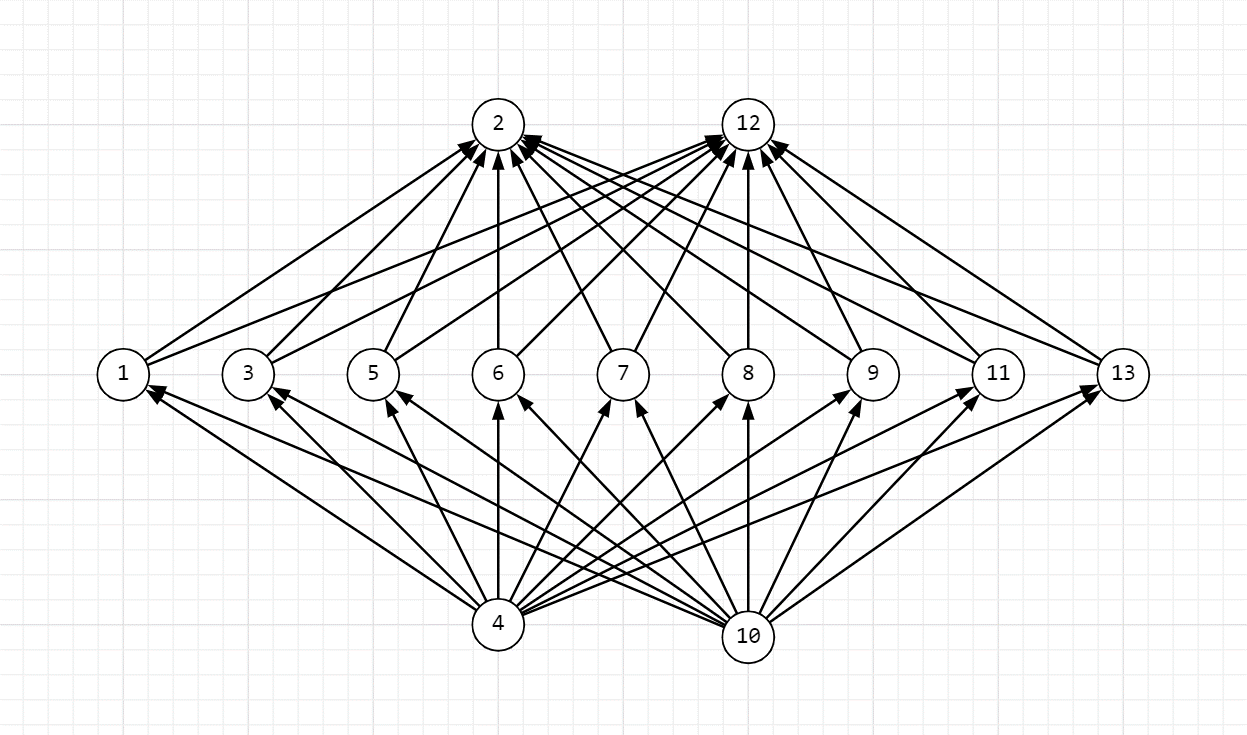


4. Изобразить диаграмму Хассе отношения доминирования на множествах М1 и М2.

М1:



М2:



5. Найти минимальные и максимальные элементы множеств М1 и М2.

|  |  |  |
| --- | --- | --- |
|  | Минимальный элемент | Максимальный элемент |
| M1 | 3, 7 | 1, 9 |
| M2 | 4, 10 | 2, 12 |

6. Найти, если существуют, наименьший и наибольший элементы множеств М1 и М2.

|  |  |  |
| --- | --- | --- |
|  | Наименьший элемент | Наибольший элемент |
| M1 | нет | нет |
| M2 | нет | нет |

**Вывод**

**Вывод**: в ходе работы я изучил упорядоченные множества, алгоритм топологической сортировки, научился представлять множества диаграммами Хассе, находить минимальные (максимальные) и наименьшие (наибольшие) элементы упорядоченного множества.