Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Унарные и бинарные операции над графами»

Выполнил:

студент группы 19ВВ2

Ходакова Д.А.

Приняли:

Митрохин М. А.

Пенза 2020

# **Цель работы**

Освоить унарные и бинарные операции над графами

# **Лабораторное задание**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) две матрицы M 1 ,

М 2 смежности неориентированных помеченных графов G 1 , G 2 . Выведите

сгенерированные матрицы на экран.

2. \* Для указанных графов преобразуйте представление матриц

смежности в списки смежности. Выведите полученные списки на экран.

**Задание 2**

1. Для матричной формы представления графов выполните операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

Результат выполнения операции выведите на экран.

2. \* Для представления графов в виде списков смежности выполните

операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

Результат выполнения операции выведите на экран.

**Задание 3**

1. Для матричной формы представления графов выполните операцию:

а) объединения *G* = *G*1  *G*2

б) пересечения *G* = *G*1  *G*2

в) кольцевой суммы *G* = *G*1  *G*2

Результат выполнения операции выведите на экран.

**Задание 4 \***

1. Для матричной формы представления графов выполните операцию декартова произведения графов *G = G*1X *G*2.

Результат выполнения операции выведите на экран.

# **Объяснение вывода программы**

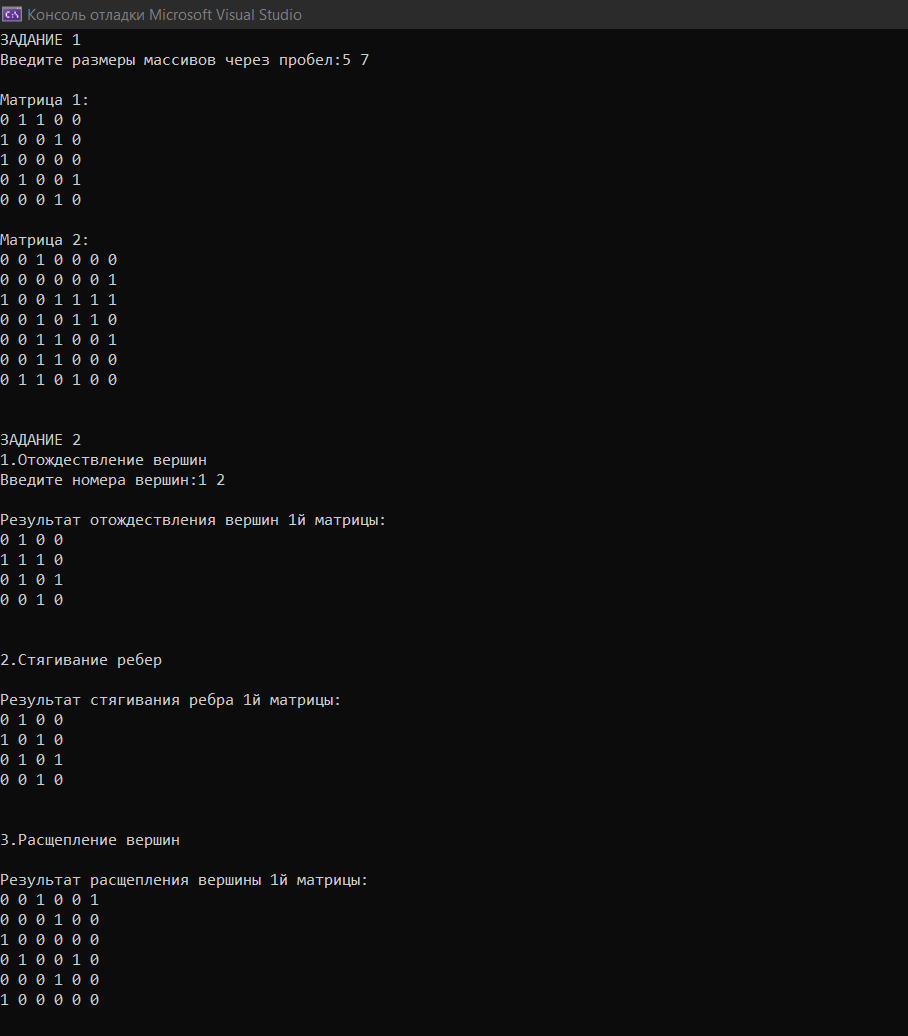
В начале работы программы просит нас ввести размеры матриц. После чего выводятся рандомно сгенерированные матрицы. Далее программа просит ввести номер вершин, с которыми Вы бы хотели совершить операции. После чего на экран выводятся результаты выполнения заданий: отождествления вершин, расщепления вершин, стягивания ребер, объединения матриц, пересечения матриц, кольцевой суммы.

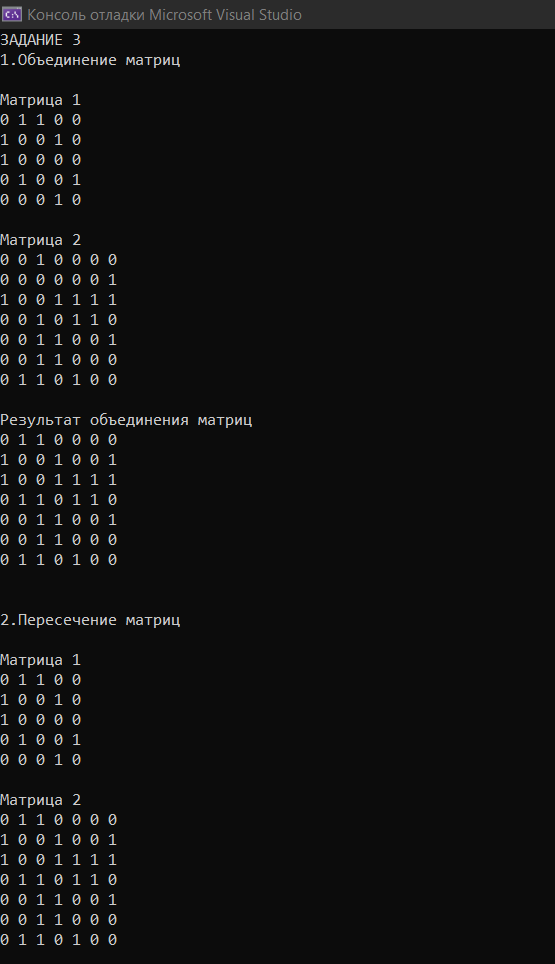
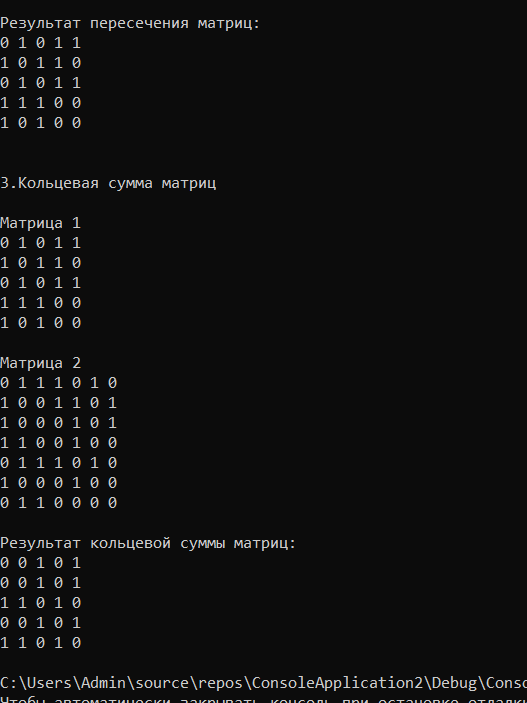
**Отождествление вершин** (иногда называется **стягиванием вершин**) не используется ограничение, что *стягивание* должно проводиться с вершинами, инцидентными одному ребру (таким образом, стягивание ребра является частным случаем отождествления вершин). Эта операция может быть проведена с любой парой (или подмножеством) вершин в графе. Рёбра между двумя *стягиваемыми* вершинами иногда удаляются. Если v и v' — вершины различных компонент графа G, то мы можем создать новый граф G' путём отождествления v и v' в G в новую вершину ***v*** в G'.

**Расщепление вершин** означает замену одной вершины на две, и эти две новые вершины смежны вершинам, которым была смежна исходная вершина. Операция является обратной отождествлению вершин.

**Стягивание пути** производится с множеством рёбер в [пути](https://ru.wikipedia.org/wiki/%D0%9F%D1%83%D1%82%D1%8C_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)), которые *стягиваются*, образуя одно ребро между конечными вершинами пути. Рёбра, инцидентные вершинам вдоль пути, либо исключаются, либо случайным образом (или по некой системе) соединяются с одной из конечных точек.

## **Результаты работы программы**

рис.1

рис.2рис.3

## **Вывод**

Научились выполнять над графами различные операции: отождествление вершин, стягивание ребра, расщепление вершины.

### **Листинг**

#include <stdio.h>

#include <random>

#include <time.h>

#include <locale.h>

#include <windows.h>

#include <string.h>

using namespace std;

void cpy(int\*\* mass1, int\*\* mass2, int n) {

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

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

mass1[i][j] = mass2[i][j];

}

}

}

void generate(int\*\* mass, int n) { // заполнение массивов

int i, r, c;

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

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

r = rand() % 2;

c = rand() % 2;

if (r == 1) {

mass[i][j] = 1;

}

else

mass[i][j] = 0;

if (i == j) {

mass[i][j] = 0;

}

}

}

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

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

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

mass[i][j] = mass[j][i];

}

}

}

}

void otoz(int s, int s1, int\*\* mass, int n) { //отождествление

int i, j;

printf("Результат отождествления вершин 1й матрицы:\n");

for (i = 0; i < n; i++) { //заносим 1ю матрицу

for (j = 0; j < n; j++) {

if (mass[s][j] > mass[s1][j])

mass[s1][j] = 1;

if (mass[i][s1] > mass[s1][j])

mass[i][s1] = 1;

}

}

int l = 0, g = 0;

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

for (j = 0; j < n; j++) {

if (mass[s][j] == 1)

g = 1;

if (mass[i][s] == 1)

l = 1;

mass[s][j] = 0;

if (g == 1)

mass[s1][j] = g;

mass[i][s] = 0;

if (l == 1)

mass[i][s1] = l;

g = 0;

l = 0;

mass[s1][s1] = 1;

}

}

//выполняем отождествление и вывод матрицы

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

for (j = 0; j < n; j++) {

if (i == s || j == s) {

continue;

}

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

}

if (i != s) {

printf("\n");

}

}

}

void styag(int s, int s1, int\*\* mass1, int n) { //стягивание

int g = 0, l = 0;

printf("Результат стягивания ребра 1й матрицы:\n");

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

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

if (mass1[s][j] == 1)

g = 1;

if (mass1[i][s] == 1)

l = 1;

mass1[s][j] = 0;

if (g == 1)

mass1[s1][j] = g;

mass1[i][s] = 0;

if (l == 1)

mass1[i][s1] = l;

g = 0;

l = 0;

mass1[s1][s1] = 0;

}

}

for (int i = 0; i < n; i++) { //выполняем стягивание и вывод матрицы

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

if (i == s || j == s) {

continue;

}

printf("%d ", mass1[i][j]);

}

if (i != s) {

printf("\n");

}

}

}

void ras(int s, int s1, int\*\* mass2, int n, int\*\* mass3) { //расщепление

printf("\nРезультат расщепления вершины 1й матрицы:\n");

for (int i = 0; i < n + 1; i++) {

for (int j = 0; j < n + 1; j++) {

mass2[i][j] = 0;

}

}

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

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

mass2[i][j] = mass3[i][j];

}

}

for (int i = 0; i < n + 1; i++) { //расщепление вершины 1й матрицы

for (int j = 0; j < n + 1; j++) {

if (j < n / 2 && mass2[s][j] == 1) {

mass2[n][j] = 1;

mass2[s][j] = 0;

}

if (i < n / 2 && mass2[i][s] == 1) {

mass2[i][n] = 1;

mass2[i][s] = 0;

}

printf("%d ", mass2[i][j]); //вывод результата

}

printf("\n");

}

}

void obe(int\*\* mass\_p1, int\*\* mass\_p2, int n, int n1) { //объединение

int i, j;

printf("Матрица 1\n");

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

for (j = 0; j < n; j++) {

printf("%d ", mass\_p1[i][j]); //вывод матрицы

}

printf("\n");

}

printf("\nМатрица 2\n");

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

printf("%d ", mass\_p2[i][j]); //вывод матрицы

}

printf("\n");

}

//выполнение операции

if (n > n1) {

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

if (mass\_p1[i][j] != 1 && mass\_p2[i][j] == 1) {

mass\_p1[i][j] = 1;

}

}

}

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

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

printf("%d ", mass\_p1[i][j]);

}

}

}

if (n1 > n) {

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

for (j = 0; j < n; j++) {

if (mass\_p1[i][j] == 1 && mass\_p2[i][j] != 1) {

mass\_p2[i][j] = 1;

}

}

}

printf("\nРезультат объединения матриц:\n");

for (int i = 0; i < n1; i++) {

for (int j = 0; j < n1; j++) {

printf("%d ", mass\_p2[i][j]);

}

printf("\n");

}

}

}

void pere(int\*\* mass\_o1, int\*\* mass\_o2, int n, int n1) { //пересечение

int i, j;

printf("Матрица 1\n");

for (i = 0; i < n; i++) { //вывод матрицы1

for (j = 0; j < n; j++) {

printf("%d ", mass\_o1[i][j]);

}

printf("\n");

}

printf("\nМатрица 2\n");

for (i = 0; i < n1; i++) { //вывод матрицы2

for (j = 0; j < n1; j++) {

printf("%d ", mass\_o2[i][j]);

}

printf("\n");

}

printf("\n");

if (n > n1) { //выполнение операции

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

if (mass\_o2[i][j] == 1 && mass\_o1[i][j] == 1)

mass\_o1[i][j] = 1;

else mass\_o1[i][j] = 0;

}

}

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

printf("%d ", mass\_o1[i][j]);

}

printf("\n");

}

}

if (n1 > n) {

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

for (j = 0; j < n; j++) {

if (mass\_o2[i][j] == 1 && mass\_o1[i][j] == 1)

mass\_o2[i][j] = 1;

else mass\_o2[i][j] = 0;

if (n > n1) {

mass\_o2[i][j] = 0;

}

}

}

printf("Результат пересечения матриц:\n");

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

for (j = 0; j < n; j++) {

printf("%d ", mass\_o2[i][j]);

}

printf("\n");

}

}

printf("\n");

}

void kolz(int\*\* mass\_k1, int\*\* mass\_k2, int n, int n1) { //кольцевая сумма

int i, j;

printf("\nМатрица 1\n");

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

for (j = 0; j < n; j++) {

printf("%d ", mass\_k1[i][j]);

}

printf("\n");

}

printf("\nМатрица 2\n");

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

printf("%d ", mass\_k2[i][j]);

}

printf("\n");

}

printf("\n");

if (n >= n1) { //выполнение операции

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

if (mass\_k1[i][j] != 1 && mass\_k2[i][j] == 1 || mass\_k2[i][j] != 1 && mass\_k1[i][j] == 1) {

mass\_k1[i][j] = 1;

}

else mass\_k1[i][j] = 0;

}

}

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

printf("%d ", mass\_k1[i][j]);

}

printf("\n");

}

}

if (n1 > n) {

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

for (j = 0; j < n; j++) {

if (mass\_k1[i][j] != 1 && mass\_k2[i][j] == 1 || mass\_k2[i][j] != 1 && mass\_k1[i][j] == 1) {

mass\_k2[i][j] = 1;

}

else mass\_k2[i][j] = 0;

}

}

printf("Результат кольцевой суммы матриц:\n");

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

for (j = 0; j < n; j++) {

printf("%d ", mass\_k2[i][j]);

}

printf("\n");

}

}

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int s, s1, n, n1;

printf("ЗАДАНИЕ 1\n");

printf("Введите размеры массивов через пробел:");

scanf\_s("%d %d", &n, &n1);

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

int\*\* mass1 = (int\*\*)malloc(sizeof(int) \* n);

int\*\* mass\_p1 = (int\*\*)malloc(sizeof(int) \* n);

int\*\* mass\_o1 = (int\*\*)malloc(sizeof(int) \* n);

int\*\* mass\_k1 = (int\*\*)malloc(sizeof(int) \* n);

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

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

mass1[i] = (int\*)malloc(sizeof(int) \* n);

mass\_p1[i] = (int\*)malloc(sizeof(int) \* n);

mass\_o1[i] = (int\*)malloc(sizeof(int) \* n);

mass\_k1[i] = (int\*)malloc(sizeof(int) \* n);

}

int\*\* massre = (int\*\*)malloc(sizeof(int) \* n1);

int\*\* mass\_p2 = (int\*\*)malloc(sizeof(int) \* n1);

int\*\* mass\_o2 = (int\*\*)malloc(sizeof(int) \* n1);

int\*\* mass\_k2 = (int\*\*)malloc(sizeof(int) \* n1);

for (int i = 0; i < n1; i++) {

massre[i] = (int\*)malloc(sizeof(int) \* n1);

mass\_p2[i] = (int\*)malloc(sizeof(int) \* n1);

mass\_o2[i] = (int\*)malloc(sizeof(int) \* n1);

mass\_k2[i] = (int\*)malloc(sizeof(int) \* n);

}

int\*\* mass2 = (int\*\*)malloc(sizeof(int) \* n + 1);

for (int i = 0; i < n + 1; i++) {

mass2[i] = (int\*)malloc(sizeof(int) \* n + 1);

}

int\*\* mass3 = (int\*\*)malloc(sizeof(int) \* n);

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

mass3[i] = (int\*)malloc(sizeof(int) \* n);

}

int i = 0;

int j = 0, k = 0, m = 0;

int a;

printf("\nМатрица 1:\n");

generate(mass, n);

generate(mass\_p2, n1);

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

for (j = 0; j < n; j++) {

mass1[i][j] = mass[i][j];

mass\_o1[i][j] = mass1[i][j];

mass\_p1[i][j] = mass1[i][j];

mass\_k1[i][j] = mass1[i][j];

}

}

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

for (j = 0; j < n; j++) {

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

}

printf("\n");

}

printf("\nМатрица 2:\n");

for (i = 0; i < n1; i++) {

for (j = 0; j < n1; j++) {

printf("%d ", mass\_p2[i][j]);

}

printf("\n");

}

printf("\n");

printf("\nЗАДАНИЕ 2\n");

printf("1.Отождествление вершин\n");

printf("Введите номера вершин:");

scanf\_s("%d %d", &s, &s1);

printf("\n");

cpy(mass3, mass, n);

otoz(s, s1, mass3, n);

printf("\n");

printf("\n2.Стягивание ребер\n");

printf("\n");

cpy(mass3, mass, n);

styag(s, s1, mass3, n);

printf("\n");

printf("\n3.Расщепление вершин\n");

cpy(mass3, mass, n);

ras(s, s1, mass2, n, mass3);

printf("\n");

printf("\nЗАДАНИЕ 3\n");

printf("1.Объединение матриц\n");

cpy(mass\_k2, mass\_p2, n1);

printf("\n");

obe(mass\_p1, mass\_p2, n, n1);

printf("\n");

printf("\n2.Пересечение матриц\n");

cpy(mass\_o2, mass\_p2, n1);

printf("\n");

pere(mass\_o1, mass\_o2, n, n1);

printf("\n3.Кольцевая сумма матриц\n");

kolz(mass\_k1, mass\_k2, n, n1);

}