Министерство образования и науки Российской Федерации

Пензенский государственный университет

Факультет вычислительной техники

**Отчёт**

По лабораторной работе №3

По курсу "ЛиОАвИЗ"

Выполнили:

Студент группы 20ВВ1

Репин И.В.

Приняла:

Юрова О.В.

Пенза 2021

**Название:** Унарные и бинарные операции над графами

**Цель работы:** выполнить ряд заданий

**Лабораторное задание:**

Задание 1

1. Сгенерируйте (используя генератор случайных чисел) две матрицы M1, М2 смежности неориентированных помеченных графов G1, G2. Выведите сгенерированные матрицы 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.

Задание 3

1. Для матричной формы представления графов выполните операцию:

а) объединения G = G1 G2

б) пересечения G = G1 G2

в) кольцевой суммы G = G1 G2

Результат выполнения операции выведите на экран.

**Описание метода решения задачи:**

Задание 1.

Заполняем двумерный массив случайными 0 и 1, следим за тем что бы значения были симметричны относительно главное оси

Задание 2.

А) получаем значения путей вершины 1 и 2, создаем вершину 3 по следующему правилу: если в хотя бы из одной из 2 вершин есть путь в вершину n, то в результирующей вершине тоже будет путь в вершину n, на месте вершины 1 записываем вершину 3, удаляем вершину 2.

Б) Проверяем, смежны ли выбранные вершины, если да, то смотреть п. А), если нет, просим повторить выбор

В) Получаем вершину 1, разделяем значения ее путей на половину, первую половину переписываем в вершину 2, вторую оставляем в вершине 1, переписываем значения в вершине 1 на 0, убираем петли при наличии и делаем вершины 1 и 2 смежными, добавляем вершину 2 в матрицу

Задание 3.

А) Получаем матрицу 1 и 2, создаем матрицу 3 по следующему правилу: если хотя бы в одной матрице есть путь из точки n в точку m, то в матрице 3 он тоже будет. Выводим матрицу 3 как результат

Б) Получаем матрицу 1 и 2, если количество точек в матрицах не равно – выводим ошибку и переходим далее, иначе создаем матрицу 3 по следующему правилу: если в матрице 1 И 2 есть путь из точки n в точку m, то в матрице 3 он будет, иначе не будет. Выводим матрицу 3 как результат

В) Получаем матрицу 1 и 2, если количество точек в матрицах не равно – выводим ошибку и переходим далее, иначе создаем матрицу 3 по следующему правилу: если в матрице 1 И 2 есть путь или нет пути из точки n в точку m, то в матрице 3 его не будет, иначе будет. Выводим матрицу 3 как результат

**Листинг:**

// лб3.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

int main()

{

int num, num2, generate;

printf("Input 1st array size: ");

scanf("%i", &num);

printf("Input 2nd array size: ");

scanf("%i", &num2);

int\*\* m1 = new int\*[num];

int\*\* m2 = new int\*[num2];

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

m1[i] = new int[num];

}

for (int i = 0; i < num2; i++) {

m2[i] = new int[num];

}

srand(time(NULL));

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

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

if (j == i) {

m1[i][j] = 0;

}

else if(j>i){

generate = rand() % 100;

if (generate >= 30) {

m1[i][j] = 1;

}else {

m1[i][j] = 0;

}

}

else {

m1[i][j] = m1[j][i];

}

}

}

for (int i = 0; i < num2; i++) {

for (int j = 0; j < num2; j++) {

if (j == i) {

m2[i][j] = 0;

}

else if (j > i) {

generate = rand() % 100;

if (generate >= 30) {

m2[i][j] = 1;

}

else {

m2[i][j] = 0;

}

}

else {

m2[i][j] = m2[j][i];

}

}

}

//print array

printf("m1 array:");

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

printf("\n");

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

printf("%4i", m1[i][j]);

}

}

printf("\nm2 array:");

for (int i = 0; i < num2; i++) {

printf("\n");

for (int j = 0; j < num2; j++) {

printf("%4i", m2[i][j]);

}

}

//2 задание

//Отождествление

int u, v, z, x;

printf("\nidentification\n");

printf("1st m1 point (from 0 to n): ");

scanf("%i", &u);

printf("2nd m1 point (from 0 to n): ");

scanf("%i", &v);

int\* ptsu = new int [num];

int\* ptsv = new int [num];

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

if (i == u) {

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

ptsu[j] = m1[i][j];

}

}else if (i == v) {

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

ptsv[j] = m1[i][j];

}

}

}

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

if (ptsu[i] == 0) {

ptsu[i] = ptsv[i];

}

}

if (u >= v) {

z = v;

x = u;

}

else {

z = u;

x = v;

}

int\*\* newm1 = new int\* [num-1];

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

newm1[i] = new int[num];

}

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

if (i == z) {

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

newm1[i][j] = ptsu[j];

}

}

else {

if (i < x) {

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

newm1[i][j] = m1[i][j];

}

}

else {

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

newm1[i][j] = m1[i+1][j];

}

}

}

}

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

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

if (j >= x) {

newm1[i][j] = newm1[i][j+1];

}

}

}

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

if (i != z) {

newm1[i][z] = newm1[z][i];

}

}

printf("\nnew m1 array:");

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

printf("\n");

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

printf("%4i", newm1[i][j]);

}

}

num = num - 1;

//стягивание ребра

int passed = 0;

printf("\nrib retraction\n");

while (passed == 0) {

printf("1st m1 point (from 0 to n): ");

scanf("%i", &u);

printf("2nd m1 point (from 0 to n): ");

scanf("%i", &v);

passed = newm1[u][v];

if ((passed == 0) or (u == v)) {

printf("Can't retraction this points! Try again\n");

passed = 0;

}

}

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

if (i == u) {

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

ptsu[j] = newm1[i][j];

}

}

else if (i == v) {

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

ptsv[j] = newm1[i][j];

}

}

}

if (u >= v) {

z = v;

x = u;

}

else {

z = u;

x = v;

}

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

if (ptsu[i] == 0) {

ptsu[i] = ptsv[i];

}

}

int\*\* idenm1 = new int\* [num -1];

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

idenm1[i] = new int[num];

}

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

if (i == z) {

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

idenm1[i][j] = ptsu[j];

}

}

else {

if (i < x) {

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

idenm1[i][j] = newm1[i][j];

}

}

else {

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

idenm1[i][j] = newm1[i + 1][j];

}

}

}

}

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

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

if (j >= x) {

idenm1[i][j] = idenm1[i][j + 1];

}

}

}

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

if (i != z) {

idenm1[i][z] = idenm1[z][i];

}

}

printf("\niden m1 array:");

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

printf("\n");

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

printf("%4i", idenm1[i][j]);

}

}

//расщепление вершины

printf("\nm1 point to split (from 0 to n): ");

scanf("%i", &u);

num = num - 1;

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

if (i == u) {

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

ptsu[j] = idenm1[i][j];

}

}

}

int border = num/2;

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

if (i <= border) {

ptsv[i] = ptsu[i];

ptsu[i] = 0;

}

else {

ptsv[i] = 0;

}

}

ptsu[num] = 1;

ptsv[u] = 1;

ptsv[num] = 0;

int\*\* splitm1 = new int\* [num+1];

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

splitm1[i] = new int[num + 1];

}

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

if (i == u) {

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

splitm1[i][j] = ptsu[j];

}

}

else if (i == num) {

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

splitm1[i][j] = ptsv[j];

}

}

else {

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

if (j < num) {

splitm1[i][j] = idenm1[i][j];

}

else {

splitm1[i][j] = ptsv[i];

}

}

}

}

printf("\nsplit m1 array:");

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

printf("\n");

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

printf("%4i", splitm1[i][j]);

}

}

//3 задание

//объединение врешин

num++;

int unum;

if (num >= num2) {

unum = num;

for (int i = num2; i < unum; i++) {

m2[i] = new int[unum];

}

}

else {

unum = num2;

for (int i = num; i < unum; i++) {

splitm1[i] = new int[unum];

}

}

int\*\* unionm = new int\* [unum];

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

unionm[i] = new int[unum];

}

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

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

if ((m2[i][j] == 1) or (splitm1[i][j] == 1)) {

unionm[i][j] = 1;

}

else {

unionm[i][j] = 0;

}

}

}

printf("\nunion array:");

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

printf("\n");

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

printf("%4i", unionm[i][j]);

}

}

//пересечение

if (num != num2) {

printf("\nError! m1 and m2 should have same shapes\n");

}

else {

int\*\* combinem = new int\* [num];

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

combinem[i] = new int[num];

}

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

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

if ((m2[i][j] == 1) and (splitm1[i][j] == 1)) {

combinem[i][j] = 1;

}

else {

combinem[i][j] = 0;

}

}

}

printf("\ncombine array:");

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

printf("\n");

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

printf("%4i", combinem[i][j]);

}

}

}

//кольцевая сумма

if (num != num2) {

printf("\nError! m1 and m2 should have same shapes\n");

}

else {

int\*\* summm = new int\* [num];

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

summm[i] = new int[num];

}

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

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

if (m2[i][j] == splitm1[i][j]) {

summm[i][j] = 0;

}

else {

summm[i][j] = 1;

}

}

}

printf("\nsumm array:");

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

printf("\n");

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

printf("%4i", summm[i][j]);

}

}

}

}

**Результаты работы программы:**

1. Задание:

Результаты работы программы представлен на рисунке 1

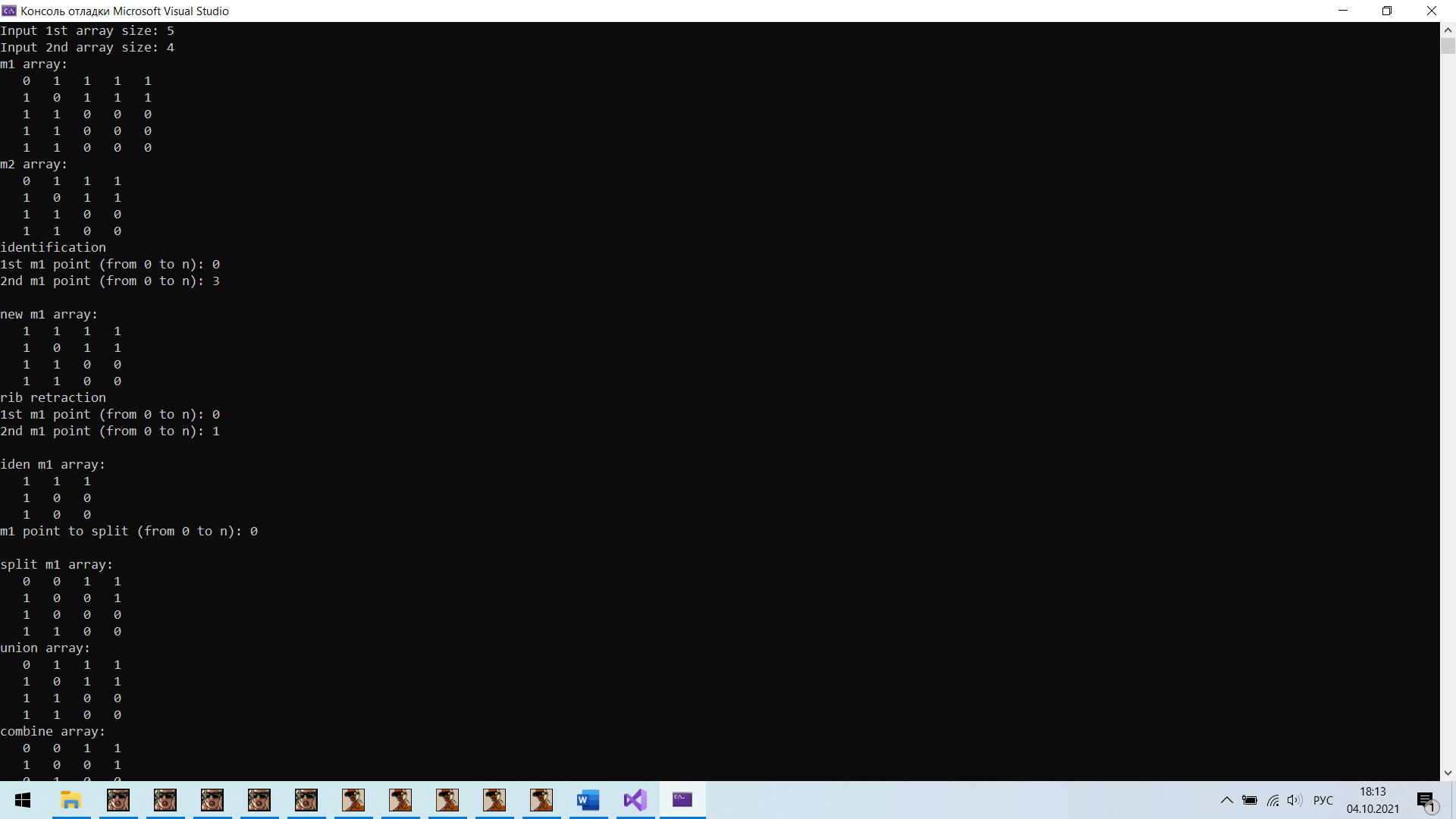
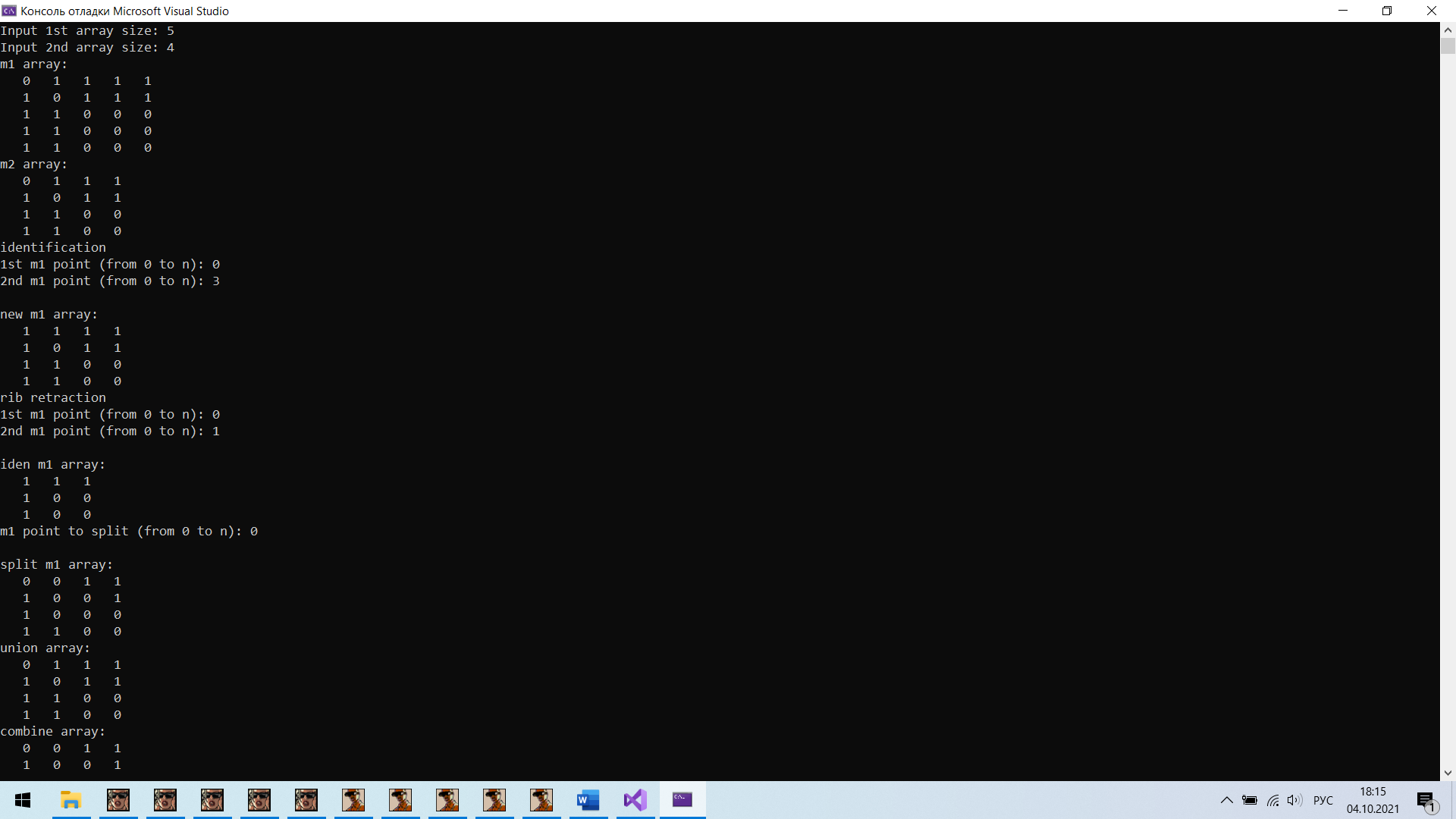


Рисунок 1.

2 Задание:

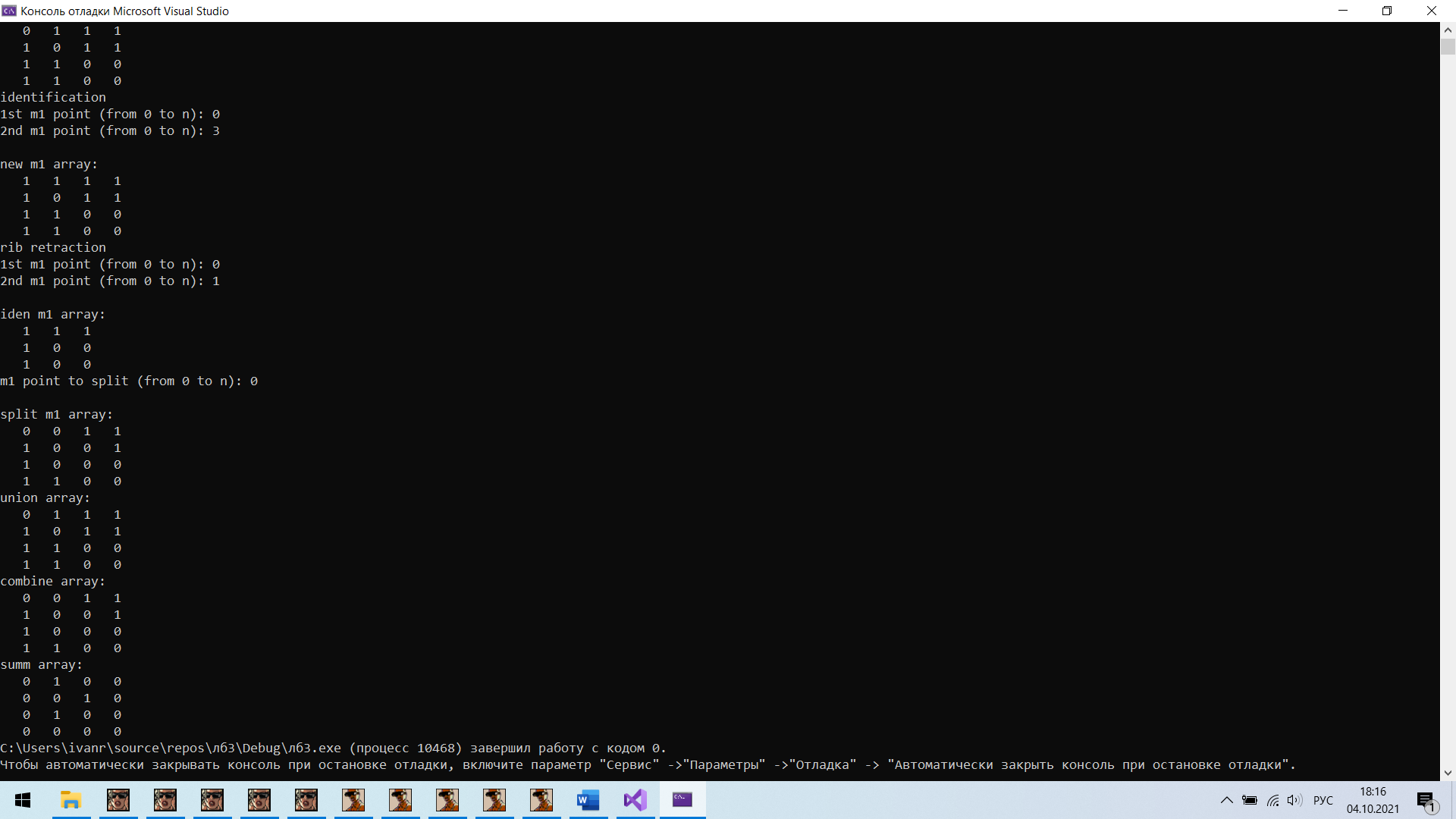
Результат работы программы представлен на рисунке 2



Риснуок 2

3 Задание:

Результат работы программы представлен на рисунке 3



Риснуок 3

**Вывод:**

Получили навыки работы с матричными представлениями графов