МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительной техники»

**ОТЧЁТ**

по лабораторной работе №6

по курсу «Л и ОА в ИЗ»

на тему «Унарные и бинарные операции над графами»

Выполнили:

Студенты группы 22ВВС1

Костин Максим

Макеева Дарья

Приняли:

Акифьев И.В.

Юрова О.В.

Пенза 2023

**Лабораторноезадание*:***

### 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.

### 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* = *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.

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <conio.h>

#include <locale.h>

#include <stdlib.h>

#include <time.h>

int oto(int v1, int v2, int\*\* matr, int n1, int g) {

int k = 0;

int\*\* matr3 = (int\*\*)malloc(sizeof(int\*) \* n1);

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

matr3[i] = (int\*)malloc(sizeof(int) \* n1);

if (g == 1) {

if (matr[v1][v2] == 0) {

printf("Ребра не существует!");

return 0;

}

}

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

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

matr3[i][j] = matr[i][j];

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

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

if ((i == v1 || i == v2) && matr3[i][j] != 1) {

matr3[i][j] = matr3[v1][j] + matr3[v2][j];

k = i;

}

}

}

int k1 = (v1 < v2) ? v1 : v2;

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

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

if (matr3[k1][j] == 1) {

matr3[j][k1] = 1;

}

}

}

if (g == 1)

matr3[k1][k1] = 0;

printf("\n Матрица: \n№ ");

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

if (i != k)

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

}

printf("\n\_\_|");

for (int i = 0; i < n1 - 1; i++) {

printf("\_\_");

}

printf("\n");

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

{

if (i != k) {

printf("%d | ", i + 1);

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

if (j != k)

printf("%d ", matr3[i][j]);

printf("\n");

}

}

}

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

free(matr3[i]);

}

free(matr3);

return 0;

}

int ras(int\*\* matr, int v1, int n1) {

n1 += 1;

int\*\* matr3 = (int\*\*)malloc(sizeof(int\*) \* n1);

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

matr3[i] = (int\*)malloc(sizeof(int) \* n1);

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

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

matr3[i][j] = 0;

for (int i = 0; i < n1 - 1; i++) {

for (int j = 0; j < n1 - 1; j++) {

if (i != v1 && j != v1)

matr3[i][j] = matr[i][j];

}

}

for (int i = 0; i < n1 - 1; i++) {

for (int j = 0; j < n1 - 1; j++) {

matr3[v1][j] = matr[v1][j];

matr3[i][v1] = matr[i][v1];

matr3[n1 - 1][j] = matr[v1][j];

matr3[i][n1 - 1] = matr[i][v1];

}

}

matr3[v1][n1 - 1] = 1;

matr3[n1 - 1][v1] = 1;

printf("\n Матрица M1: \n№ ");

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

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

}

printf("\n\_\_|");

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

printf("\_\_");

}

printf("\n");

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

{

printf("%d | ", i + 1);

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

printf("%d ", matr3[i][j]);

printf("\n");

}

}

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

free(matr3[i]);

}

free(matr3);

return 0;

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int n0 = 0, n = 0;

int kr0 = 0, kr1 = 0;

printf("Введите количество вершин: ");

scanf\_s("%d", &n0);

int\*\* matr1 = (int\*\*)malloc(sizeof(int\*) \* n0);

for (int i = 0; i < n0; i++)

matr1[i] = (int\*)malloc(sizeof(int) \* n0);

for (int i = 0; i < n0; i++)

for (int j = 0; j < n0; j++)

matr1[i][j] = 0;

for (int i = 0; i < n0; i++) {

for (int j = i; j < n0; j++) {

matr1[i][j] = rand() % 2;

matr1[j][i] = matr1[i][j];

if (i == j)

matr1[i][j] = 0;

}

}

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

for (int j = i; j < n; j++)

if (matr1[i][j] == 1) {

kr0 += 1;

}

printf("\n Матрица M1: \n№ ");

for (int i = 0; i < n0; i++) {

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

}

printf("\n\_\_|");

for (int i = 0; i < n0; i++) {

printf("\_\_");

}

printf("\n");

for (int i = 0; i < n0; ++i) {

{

printf("%d | ", i + 1);

for (int j = 0; j < n0; ++j)

printf("%d ", matr1[i][j]);

printf("\n");

}

}

printf("\n--------------------------------------------------------\n");

printf("Введите количество вершин: ");

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

int\*\* matr2 = (int\*\*)malloc(sizeof(int\*) \* n);

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

matr2[i] = (int\*)malloc(sizeof(int) \* n);

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

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

matr2[i][j] = 0;

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

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

matr2[i][j] = rand() % 2;

matr2[j][i] = matr2[i][j];

if (i == j)

matr2[i][j] = 0;

}

}

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

for (int j = i; j < n; j++)

if (matr2[i][j] == 1) {

kr1 += 1;

}

printf("\n Матрица M2: \n№ ");

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

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

}

printf("\n\_\_|");

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

printf("\_\_");

}

printf("\n");

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

{

printf("%d | ", i + 1);

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

printf("%d ", matr2[i][j]);

printf("\n");

}

}

int ver1 = 0, ver2 = 0;

int no = (n > n0) ? n : n0;

int v = 0;

while (1) {

int g = 0;

printf("\nВведите номер операции:\n1)Отожествление вершин 2)Стягивание ребра 3)Расщепление 4)Бинарные операции 5)Выйти\n");

scanf\_s("%d", &v);

switch (v) {

case 1:

printf("Выберите матрицу с которой хотите работать(М1-1, М2-2):");

scanf\_s("%d", &v);

switch (v) {

case 1:

printf("Укажите вершины для отожествления:");

scanf\_s("%d %d", &ver1, &ver2);

if ((ver1 < 1 || ver1 > n0) || (ver2 < 1 || ver2 > n0)) {

printf("Нет вершины!");

}

else

oto(ver1 - 1, ver2 - 1, matr1, n0, 0);

break;

case 2:

printf("Укажите вершины для отожествления:");

scanf\_s("%d %d", &ver1, &ver2);

if ((ver1 < 1 || ver1 > n) || (ver2 < 1 || ver2 > n)) {

printf("Нет вершины!");

}

else

oto(ver1 - 1, ver2 - 1, matr2, n, 0);

break;

}

break;

case 2:

g = 1;

printf("Выберите матрицу с которой хотите работать(М1-1, М2-2):");

scanf\_s("%d", &v);

switch (v) {

case 1:

printf("Укажите вершины между которыми ребро для стягивания:");

scanf\_s("%d %d", &ver1, &ver2);

if ((ver1 < 1 || ver1 > n0) || (ver2 < 1 || ver2 > n0)) {

printf("Нет вершины!");

}

else

oto(ver1 - 1, ver2 - 1, matr1, n0, g);

break;

case 2:

printf("Укажите вершины между которыми ребро для стягивания:");

scanf\_s("%d %d", &ver1, &ver2);

if ((ver1 < 1 || ver1 > n) || (ver2 < 1 || ver2 > n)) {

printf("Нет вершины!");

}

else

oto(ver1 - 1, ver2 - 1, matr2, n, g);

break;

}

break;

case 3:

printf("Выберите матрицу с которой хотите работать(М1-1, М2-2):");

scanf\_s("%d", &v);

switch (v) {

case 1:

printf("Укажите вершину, которую надо расщепить:");

scanf\_s("%d", &ver1);

if (ver1 < 1 || ver1 > n0) {

printf("Нет вершины!");

}

else

ras(matr1, ver1 - 1, n0);

break;

case 2:

printf("Укажите вершину, которую надо расщепить:");

scanf\_s("%d", &ver1);

if (ver1 < 1 || ver1 > n) {

printf("Нет вершины!");

}

else

ras(matr2, ver1 - 1, n);

break;

}

break;

case 4:

printf("\nВведите номер операции:\n1)Объединение 2)Пересечение 3)Кольцевая сумма\n");

scanf\_s("%d", &v);

switch (v) {

case 1:

{

int\*\* matr = (int\*\*)malloc(sizeof(int\*) \* no);

for (int i = 0; i < no; i++)

matr[i] = (int\*)malloc(sizeof(int) \* no);

for (int i = 0; i < no; i++) {

for (int j = 0; j < no; j++) {

if (n0 == no && n != no)

matr[i][j] = matr1[i][j];

if (n == no && n0 != no)

matr[i][j] = matr2[i][j];

if (n == n0)

matr[i][j] = 0;

}

}

int ne = (n < n0) ? n : n0;

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

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

matr[i][j] = matr1[i][j] | matr2[i][j];

}

}

printf("\n Матрица M3: \n№ ");

for (int i = 0; i < no; i++) {

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

}

printf("\n\_\_|");

for (int i = 0; i < no; i++) {

printf("\_\_");

}

printf("\n");

for (int i = 0; i < no; ++i) {

{

printf("%d | ", i + 1);

for (int j = 0; j < no; ++j)

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

printf("\n");

}

}

for (int i = 0; i < no; i++) {

free(matr[i]);

}

free(matr);

}

break;

case 2:

{

int\*\* matr = (int\*\*)malloc(sizeof(int\*) \* no);

for (int i = 0; i < no; i++)

matr[i] = (int\*)malloc(sizeof(int) \* no);

for (int i = 0; i < no; i++) {

for (int j = 0; j < no; j++) {

matr[i][j] = 0;

}

}

int ne = (n < n0) ? n : n0;

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

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

matr[i][j] = matr1[i][j] & matr2[i][j];

}

}

printf("\n Матрица M3: \n№ ");

for (int i = 0; i < no; i++) {

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

}

printf("\n\_\_|");

for (int i = 0; i < no; i++) {

printf("\_\_");

}

printf("\n");

for (int i = 0; i < no; ++i) {

{

printf("%d | ", i + 1);

for (int j = 0; j < no; ++j)

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

printf("\n");

}

}

for (int i = 0; i < no; i++) {

free(matr[i]);

}

free(matr);

}

break;

case 3:

{

int\*\* matr = (int\*\*)malloc(sizeof(int\*) \* no);

for (int i = 0; i < no; i++)

matr[i] = (int\*)malloc(sizeof(int) \* no);

for (int i = 0; i < no; i++) {

for (int j = 0; j < no; j++) {

if (n0 == no && n != no)

matr[i][j] = matr1[i][j];

if (n == no && n0 != no)

matr[i][j] = matr2[i][j];

if (n == n0)

matr[i][j] = 0;

}

}

int ne = (n < n0) ? n : n0;

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

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

if ((matr1[i][j] == 1 && matr2[i][j] == 0) || (matr1[i][j] == 0 && matr2[i][j] == 1))

matr[i][j] = 1;

else

matr[i][j] = 0;

}

}

int iz = 0, iz1 = 0;

int\* iz2;

for (int i = 0; i < no; i++) {

iz = 0;

for (int j = 0; j < no; j++) {

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

iz++;

}

}

if (iz == 0)

iz1++;

}

if (iz1 != 0) {

iz2 = (int\*)malloc(iz1 \* sizeof(int));

int ki = 0;

for (int i = 0; i < no; i++) {

iz = 0;

for (int j = 0; j < no; j++) {

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

iz++;

}

}

if (iz == 0) {

iz2[iz] = i;

ki++;

}

}

printf("\n Матрица M3: \n№ ");

for (int i = 0; i < no; i++) {

for (int j = 0; j < iz1; j++) {

if (i != iz2[j])

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

}

}

printf("\n\_\_|");

for (int i = 0; i < no; i++) {

printf("\_\_");

}

printf("\n");

for (int i = 0; i < no; ++i) {

for (int h = 0; h < iz1; h++) {

if (i != iz2[h]) {

printf("%d | ", i + 1);

for (int j = 0; j < no; ++j)

for (int h1 = 0; h1 < iz1; h1++) {

if (j != iz2[h1]) {

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

}

}

printf("\n");

}

}

}

free(iz2);

}

else {

printf("\n Матрица M3: \n№ ");

for (int i = 0; i < no; i++) {

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

}

printf("\n\_\_|");

for (int i = 0; i < no; i++) {

printf("\_\_");

}

printf("\n");

for (int i = 0; i < no; ++i) {

{

printf("%d | ", i + 1);

for (int j = 0; j < no; ++j)

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

printf("\n");

}

}

}

for (int i = 0; i < no; i++) {

free(matr[i]);

}

free(matr);

}

break;

}

break;

case 5:

for (int i = 0; i < n0; i++) {

free(matr1[i]);

}

free(matr1);

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

free(matr2[i]);

}

free(matr2);

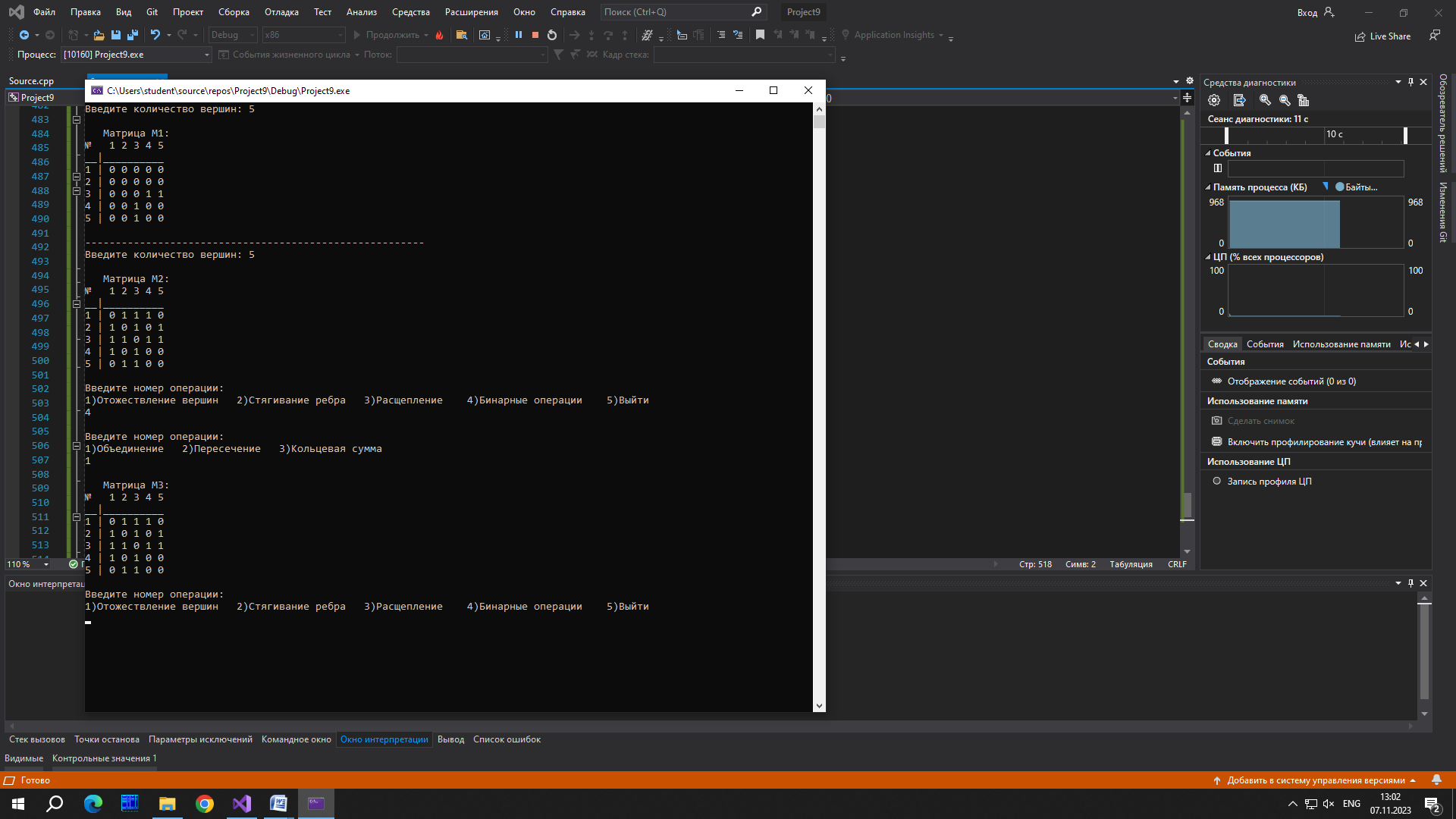
return 0;

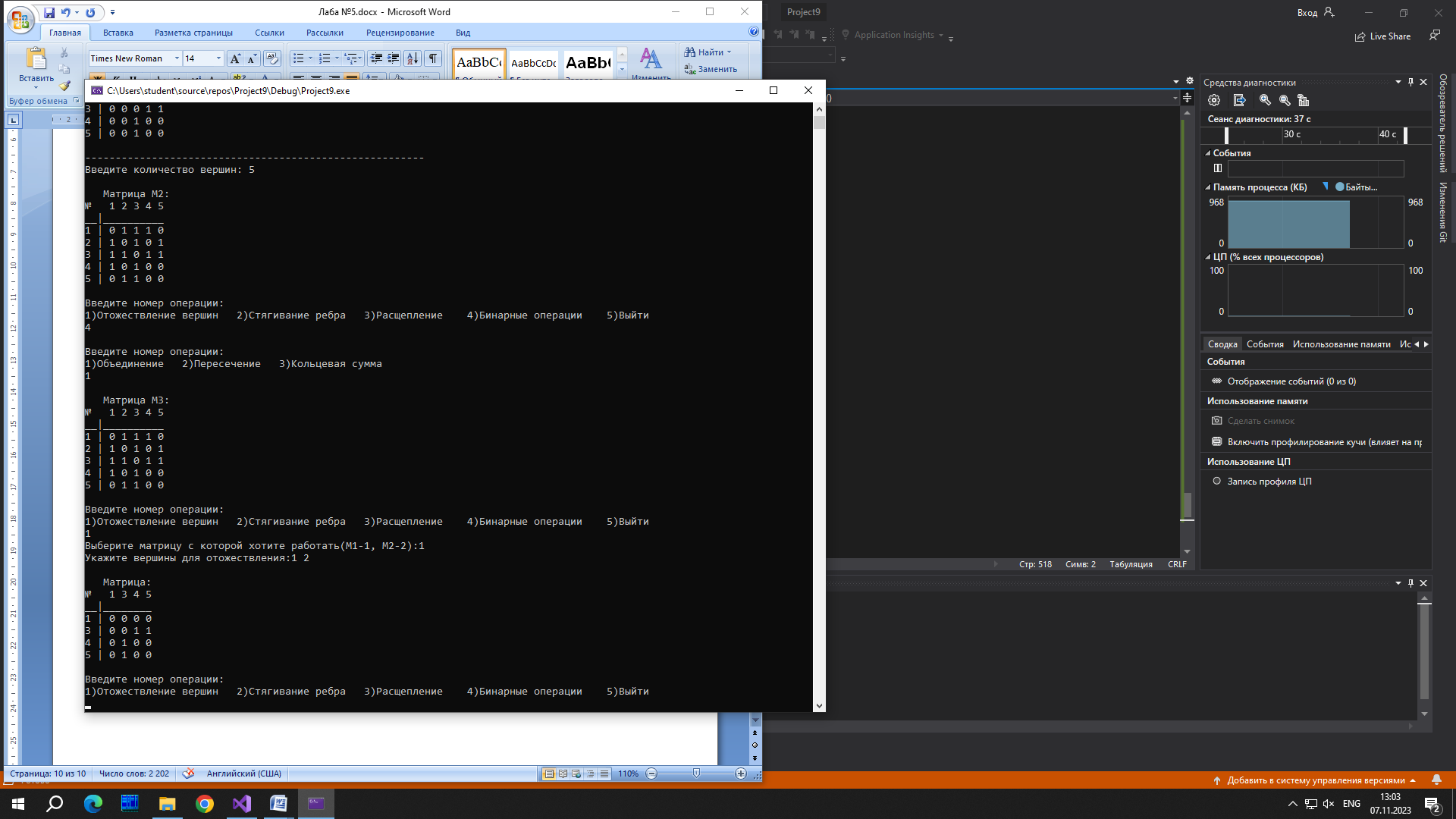
}

}

}

**Результат работы программы**

****

****

**Вывод**

В ходе выполнения лабораторной работы научились выполнять унарные и бинарные операции над неориентированными графами.