|  |
| --- |
| Министерство 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 19ВВ3:  Литвинов А. Ю.  Принял:  Митрохин М. А. |
| Пенза  2020 |

**Цель работы**

Реализовать алгоритм реализации бинарных и унарных операций.

### **Лабораторное задание**

Создать алгоритм реализации отождествления вершин, стягивания, расщепления вершины, объединения, пересечения, кольцевой суммы.

**Листинг программы**

#include <iostream>

#include "time.h"

#include "windows.h"

#include "conio.h"

#include "stdio.h"

#pragma warning (disable : 4996)

int\*\* dynamic(int SIZE);

void generate(int\*\* mass, int SIZE);

void print(int\*\* mass, int SIZE);

int otojd(int\*\* mass, int SIZE);

void circle\_sum(int\*\* mass\_A, int\*\* mass\_B, int\*\* mass\_res, int SIZE);

void del(int\*\* mass, int SIZE);

int stag(int\*\* mass, int SIZE);

void unionn(int\*\* mass\_A, int\*\* mass\_B, int\*\* mass\_res, int SIZE);

void cross(int\*\* mass\_A, int\*\* mass\_B, int\*\* mass\_res, int SIZE);

void cross(int\*\* mass\_A, int\*\* mass\_B, int\*\* mass\_res, int SIZE) {

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

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

mass\_res[i][j] = mass\_A[i][j] & mass\_B[i][j];

}

}

}

void unionn(int\*\* mass\_A, int\*\* mass\_B, int\*\* mass\_res, int SIZE) {

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

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

mass\_res[i][j] = mass\_A[i][j] | mass\_B[i][j];

}

}

}

int stag(int\*\* mass, int SIZE) {

int point\_1, point\_2;

printf("\n Введите номера вершин, которые вы хотели бы стянуть (в порядке возрастания) : ");

scanf("%d%\*c%d", &point\_1, &point\_2);

if (point\_1 > point\_2) return SIZE;

point\_1--;

point\_2--;

if (mass[point\_1][point\_2] == 0) {

printf("\n Введенные вершины невозможно отождествить.\n");

return SIZE;

}

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

if (mass[point\_1][i] == 1 || mass[point\_2][i] == 1) {

mass[point\_1][i] = 1;

mass[i][point\_1] = 1;

}

}

for (int i = point\_2; i < SIZE - 1; i++) {

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

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

}

}

for (int i = point\_2; i < SIZE - 1; i++) {

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

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

}

}

mass[point\_1][point\_1] = 0;

printf(" (в результирующей матрице номера вершин начинная с %d-ой уменьшаются на единицу)\n", point\_2 + 2);

SIZE--;

return SIZE;

}

void del(int\*\* mass, int SIZE) {

bool g = true;

int point\_1;

printf("\n Введите номер вершины, которую вы хотете расщепить: ");

scanf("%d", &point\_1);

int arr[6];

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

arr[i] = mass[i][point\_1 - 1];

}

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

mass[6][i] = 0;

mass[i][6] = 0;

}

for (int i = 0; i < 6; i++) {

mass[point\_1 - 1][i] = 0;

mass[i][point\_1 - 1] = 0;

}

for (int i = 0; i < 7; i++) {

if (arr[i] != 0 && g == true) {

mass[point\_1 - 1][i] = 1;

mass[i][point\_1 - 1] = 1;

g = false;

}

else if (arr[i] != 0) {

mass[6][i] = 1;

mass[i][6] = 1;

g = true;

}

}

mass[6][point\_1 - 1] = 1;

mass[point\_1 - 1][6] = 1;

mass[6][6] = 0;

}

int\*\* dynamic(int SIZE) {

int\*\* mass;

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

for (int i = 0; i < SIZE; i++) mass[i] = (int\*)malloc(SIZE \* sizeof(int));

return mass;

}

void generate(int\*\* mass, int SIZE) {

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

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

if (j > i) {

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

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

}

else if (i == j) mass[i][j] = 0;

}

}

}

void print(int\*\* mass, int SIZE) {

printf(" ");

for (int k = 1; k <= SIZE; k++) printf(" %d", k);

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

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

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

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

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

}

printf("\n");

}

}

int otojd(int\*\* mass, int SIZE) {

int point\_1, point\_2;

printf("\n Введите номера вершин, которые вы хотели бы отождествить (в порядке возрастания) : ");

scanf("%d%\*c%d", &point\_1, &point\_2);

if (point\_1 > point\_2) return SIZE;

point\_1--;

point\_2--;

if (mass[point\_1][point\_2] == 0) {

printf("\n Введенные вершины невозможно отождествить.\n");

return SIZE;

}

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

if (mass[point\_1][i] == 1 || mass[point\_2][i] == 1) {

mass[point\_1][i] = 1;

mass[i][point\_1] = 1;

}

}

for (int i = point\_2; i < SIZE - 1; i++) {

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

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

}

}

for (int i = point\_2; i < SIZE - 1; i++) {

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

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

}

}

printf(" (в результирующей матрице номера вершин начинная с %d-ой уменьшаются на единицу)\n", point\_2 + 2);

SIZE--;

return SIZE;

}

void circle\_sum(int\*\* mass\_A, int\*\* mass\_B, int\*\* mass\_res, int SIZE) {

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

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

mass\_res[i][j] = mass\_A[i][j] ^ mass\_B[i][j];

}

}

}

int main()

{

int count = 6, count\_dynamic, count\_m1, countd = 6;

int count\_dynamic1, count\_dynamic2;

int\*\* mass1;

int\*\* mass2;

int\*\* mass3;

int\*\* mass5;

int\*\* mass\_res;

int\*\* mass\_res\_mul;

int\*\* mass4;

int\*\* mass6;

int\*\* mass8;

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

mass1 = dynamic(count);

mass2 = dynamic(count);

mass4 = dynamic(7);

mass5 = dynamic(count);

mass6 = dynamic(count);

mass\_res = dynamic(count);

generate(mass1, count);

generate(mass2, count);

printf(" Массив M1:\n\n");

print(mass1, count);

printf("\n Массив M2:\n\n");

print(mass2, count);

for (int i = 0; i < 6; i++) {

for (int j = 0; j < 6; j++) {

mass4[i][j] = mass1[i][j];

}

}

for (int i = 0; i < 6; i++) {

for (int j = 0; j < 6; j++) {

mass5[i][j] = mass1[i][j];

}

}

for (int i = 0; i < 6; i++) {

for (int j = 0; j < 6; j++) {

mass6[i][j] = mass1[i][j];

}

}

//A

count\_m1 = otojd(mass6, count);

printf("\n Массив Mass1 после операции отождествления:\n\n");

print(mass6, count\_m1);

//A

//B

count\_m1 = stag(mass5, count);

printf("\n Результат стягивания вершин:\n\n");

print(mass5, count\_m1);

//B

//C

del(mass4, 7);

printf("\n Результат расщепления вершины:\n\n");

print(mass4, 7);

//rashep

//C

//A2

printf("\n Результат после операции обЪединения Mass1 и Mass2:\n\n");

unionn(mass1, mass2, mass\_res, count);

print(mass\_res, count);

//A2

//B2

printf("\n Результат после операции пересечения Mass1 и Mass2:\n\n");

cross(mass1, mass2, mass\_res, count);

print(mass\_res, count);

//B2

//C3

printf("\n Результат после операции кольццевой суммы Mass1 и Mass2:\n\n");

circle\_sum(mass1, mass2, mass\_res, count);

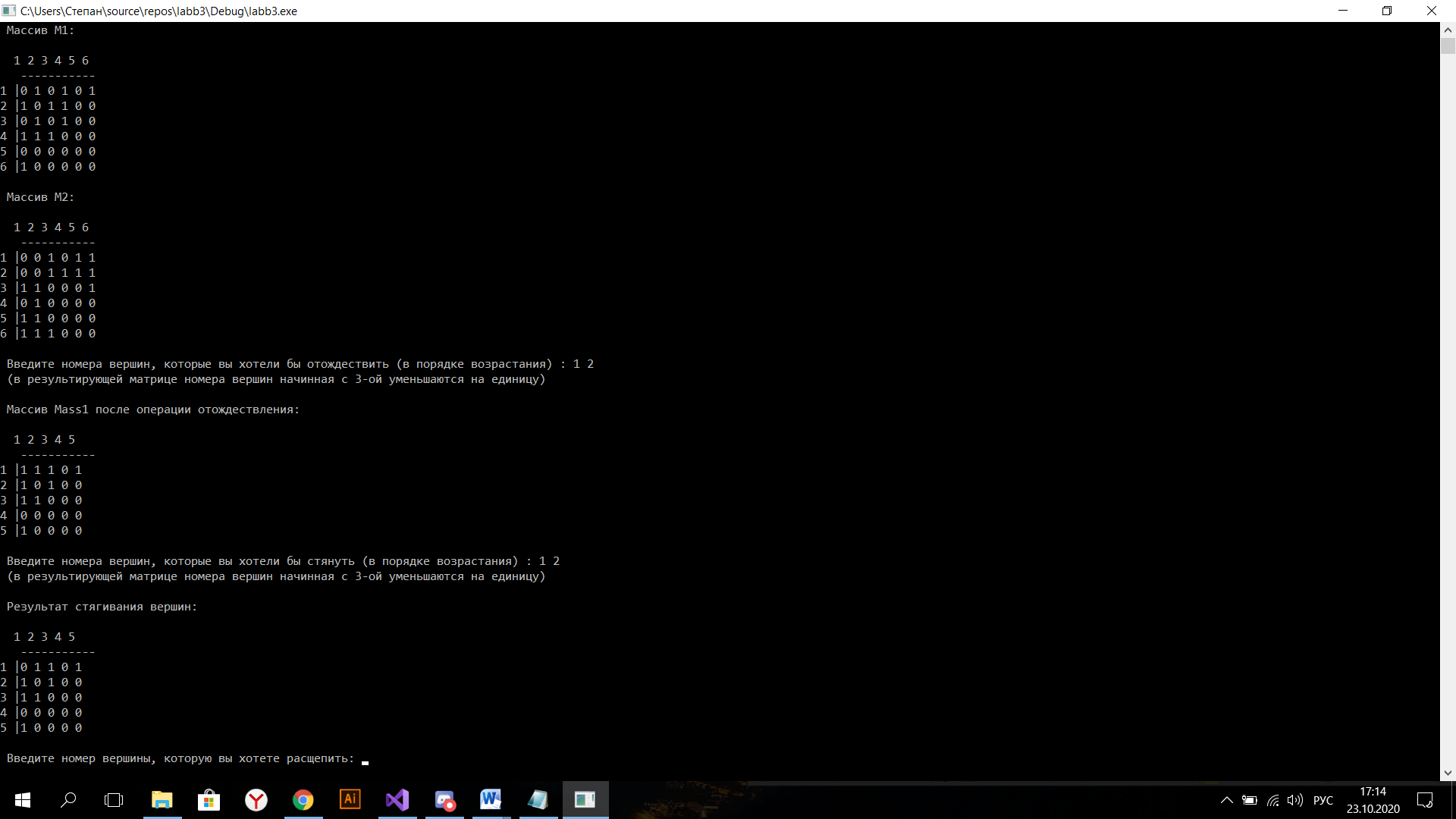
print(mass\_res, count);

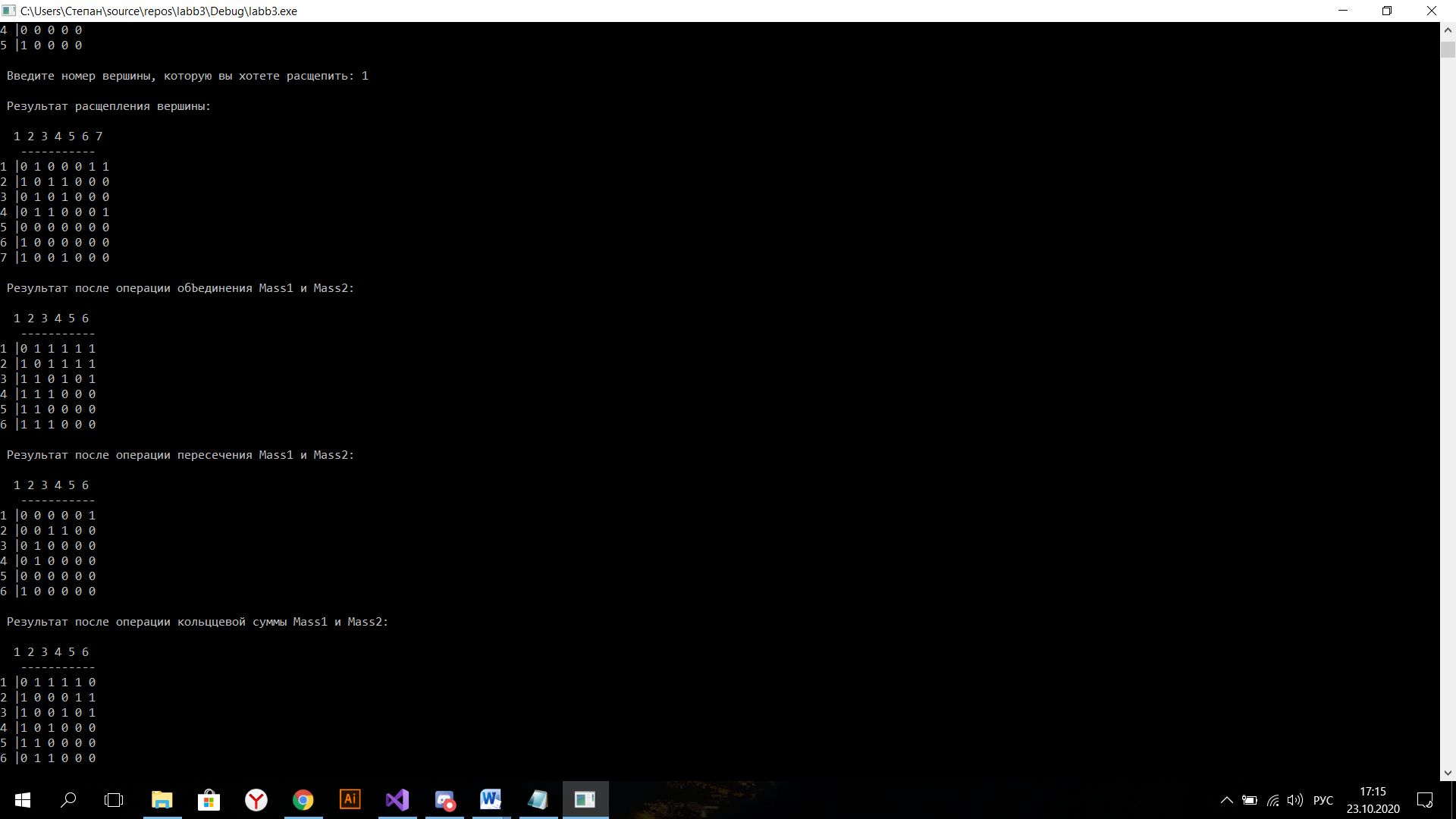
//C3

\_getch();

}

**Результат работы программы:**





**Вывод:** Реализовал бинарные и унарные операции над графами, представленными матрицей смежности.