Министерство высшего образования и науки Российской Федерации

Пензенский Государственный Университет

Кафедра «Вычислительная техника»

ОТЧЕТ

Лабораторная работа №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»  
на тему «Унарные и бинарные операции над графами»

Выполнил студент группы 20ВВ2:

Пантелеев И. А.

Приняли:

Митрохин М. А.

Юрова О. В.

Пенза 2021

**Название**

Унарные и бинарные операции над графами.

**Цель работы**

Научиться производить унарные и бинарные операции над графами.

**Лабораторное задание**

**Задание 1:**

1. Сгенерируйте (используя генератор случайных чисел) две матрицы 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

б) пересечения

в) кольцевой суммы

Результат выполнения операции выведите на экран.

**Задание 4 \***

1. Для матричной формы представления графов выполните операцию

декартова произведения графов G = G 1 X G 2 .

Результат выполнения операции выведите на экран.

**Листинг программы:**

#include <iostream>

using namespace std;

int i, j;

void InitMatrix(int\*\* G, int n)

{

srand(time(NULL));

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

{

G[i][i] = 0;

for (j = 0; j < i; j++)

{

if (rand() % 100 <= 50) //заполнение графа случайными числами

{

G[i][j] = G[j][i] = 1;

}

else

{

G[i][j] = G[j][i] = 0;

}

}

}

}

void OutMatrix(int\*\* G, int n)

{

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

{

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

{

cout << " " << G[i][j];

}

cout << endl;

}

cout << endl;

}

void Сontraction(int\*\* M1, int\*\* M4, int n, int f, int s)

{

int k = 0;

int l = 0;

M1[f][s] = M1[s][f] = 0;

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

{

if (i != f && i != s)

{

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

{

if (j != f && j != s)

{

if(M1[i][j]==1)

M4[k][l] = 1;

l++;

}

}

l = 0;

k++;

}

}

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

{

if ((i == f || i == s))

{

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

{

if (j != s && j != f)

{

if (M1[i][j] == 1)

{

M4[n - 2][l] = 1;

M4[l][n - 2] = 1;

}

l++;

}

}

l = 0;

}

}

}

void Split(int\*\* M1, int\*\* M3, int size, int pos)

{

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

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

M3[i][j] = M1[i][j];

if (j == pos) {

if (i == pos) { M3[i][size] = 1; M3[size][i] = M3[i][size]; }

else { M3[i][size] = M1[i][pos]; M3[size][i] = M1[i][pos]; }

}

}

}

}

void SumMatrix(int\*\* M1, int\*\* M2, int size)

{

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

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

{

M1[i][j] += M2[i][j];

if (M1[i][j] == 2)

M1[i][j] = 1;

}

}

}

void CrossMatrix(int\*\* M1, int\*\* M2, int size)

{

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

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

{

if (M1[i][j] != M2[i][j])

M1[i][j] = 0;

}

}

}

void RingSum(int\*\* M1, int\*\* M2, int size)

{

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

{

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

{

if (M1[i][j] == M2[i][j])

M1[i][j] = 0;

else

M1[i][j] = 1;

}

}

}

void DeleteMass(int\*\* G, int n)

{

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

delete[] G[i];

delete[] G;

}

void Decart(int\*\* M1, int\*\* M2, int\*\* M5, int size)

{

for (int m = 0; m < size; m++)

{

for (i = size \* m; i < size + size \* m; i++)

{

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

{

for (j = size \* n; j < size + size \* n; j++)

{

if ((m == n) && (i == j))

M5[i][j] = 0;

if ((m == n) && (i != j))

M5[i][j] = M1[i - size \* m][j - size \* n];

if ((m != n) && (i - size \* m == j - size \* n))

M5[i][j] = M2[m][n];

if ((m != n) && (i - size \* m != j - size \* n))

M5[i][j] = 0;

}

}

}

}

}

void main()

{

setlocale(LC\_ALL, "Rus");

int Number, pos, first, second, size;

cout << "Введите размер графа:" << endl;

cin >> size;

int\*\* M1 = new int\* [size];

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

M1[i] = new int[size];

int\*\* M2 = new int\* [size];

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

M2[i] = new int[size];

int\*\* M3 = new int\* [size + 1];

for (i = 0; i < size + 1; i++)

{

M3[i] = new int[size + 1];

for (j = 0; j < size + 1; j++)

M3[i][j] = 0;

}

int\*\* M4 = new int\* [size - 1];

for (i = 0; i < size - 1; i++)

{

M4[i] = new int[size - 1];

for (j = 0; j < size - 1; j++)

M4[i][j] = 0;

}

int\*\* M5 = new int\* [size \* size];

for (i = 0; i < size\*size; i++)

{

M5[i] = new int[size\*size];

for (j = 0; j < size\*size; j++)

M5[i][j] = 0;

}

InitMatrix(M1, size);

cout << "\nМатрца смежности графа\n";

OutMatrix(M1, size);

cout << "Введите тип операции:\n1-Отождествление\n2-Стягивание\n3-Расщепление\n4-Объединение\n5-Пересечение\n6-Кольцевая сумма\n7-Декартовое произведение\n8-Выход\n\n";

cin >> Number;

switch (Number)

{

case 1:

cout << "Введите отождествляемые вершины" << endl;

cin >> first >> second;

Сontraction(M1, M4, size, first - 1, second - 1);

OutMatrix(M4, size - 1);

break;

case 2:

cout << "Введите стягивающиеся вершины" << endl;

cin >> first >> second;

if (M1[first-1][second-1] == 1)

{

Сontraction(M1, M4, size, first - 1, second - 1);

OutMatrix(M4, size - 1);

}

else cout << "Невозможно стянуть данные вершины";

break;

case 3:

cout << "Введите вершину для расщепления" << endl;

cin >> pos;

Split(M1, M3, size, pos - 1);

cout<<"\nИтоговая матрица\n";

OutMatrix(M3, size + 1);

break;

case 4:

cout<<"\nВторая матрица\n";

InitMatrix(M2, size);

OutMatrix(M2, size);

SumMatrix(M1, M2, size);

cout<<"Итоговая матрица\n";

OutMatrix(M1, size);

break;

case 5:

cout<<"\nВторая матрица\n";

InitMatrix(M2, size);

OutMatrix(M2, size);

CrossMatrix(M1, M2, size);

cout<<"Итоговая матриц\n";

OutMatrix(M1, size);

break;

case 6:

cout<<"\nВторая матрица\n";

InitMatrix(M2, size);

OutMatrix(M2, size);

RingSum(M1, M2, size);

cout<<"Итоговая матрица\n";

OutMatrix(M1, size);

break;

case 7:

cout << "\nВторая матрица\n";

InitMatrix(M2, size);

OutMatrix(M2, size);

Decart(M1, M2, M5, size);

cout << "Итоговая матрица\n";

OutMatrix(M5, size \* size);

break;

case 8:

exit(1);

}

DeleteMass(M1, size);

DeleteMass(M2, size);

DeleteMass(M3, size + 1);

DeleteMass(M4, size - 1);

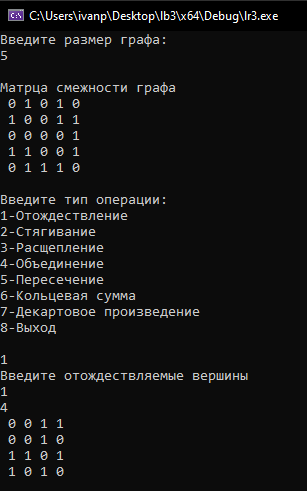
system("pause>>void");

}

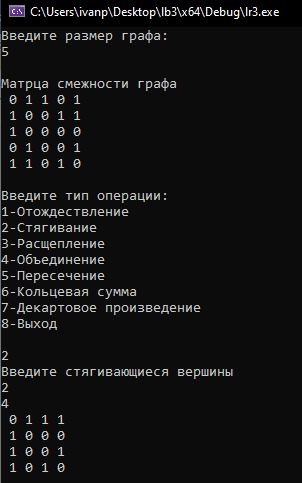
Результаты работы программы:

Работа программы с матрицами смежности

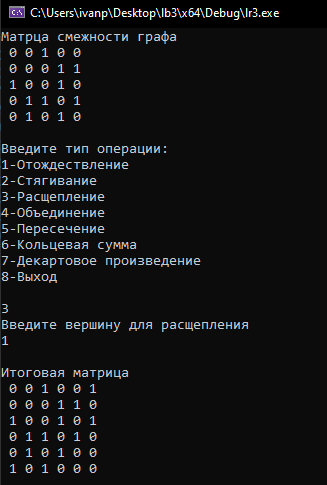
1 задание:



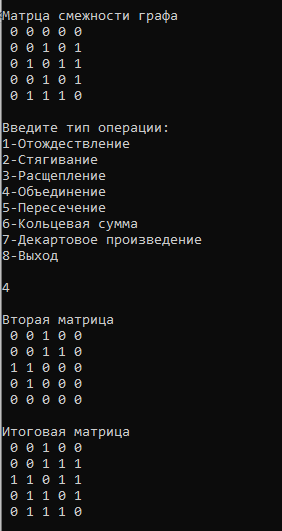
2 задание:



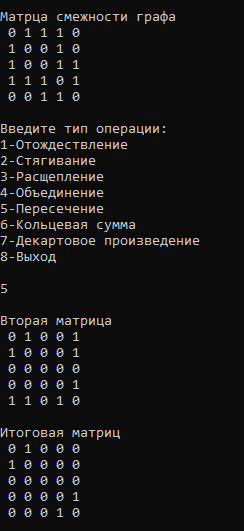
3 задание:



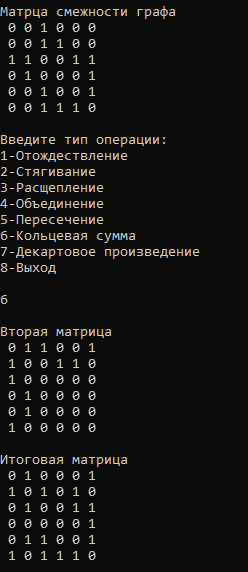
4 задание:



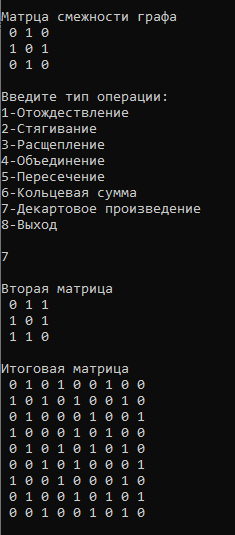
5 задание:



6 задание:



7 задание:



**Вывод:** научился производить унарные