Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу “Логика и основа алгоритмизации в ИЗ”

на тему “ Унарные и бинарные операции над графами”



Пенза 2022

**Название**

Унарные и бинарные операции над графами.

**Цель работы:** выполнить лабораторные указания 1-4 реализуя 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.

**Задание 2**

1. Для матричной формы представления графов выполните операцию:

а) отождествления вершин

б) стягивания ребра

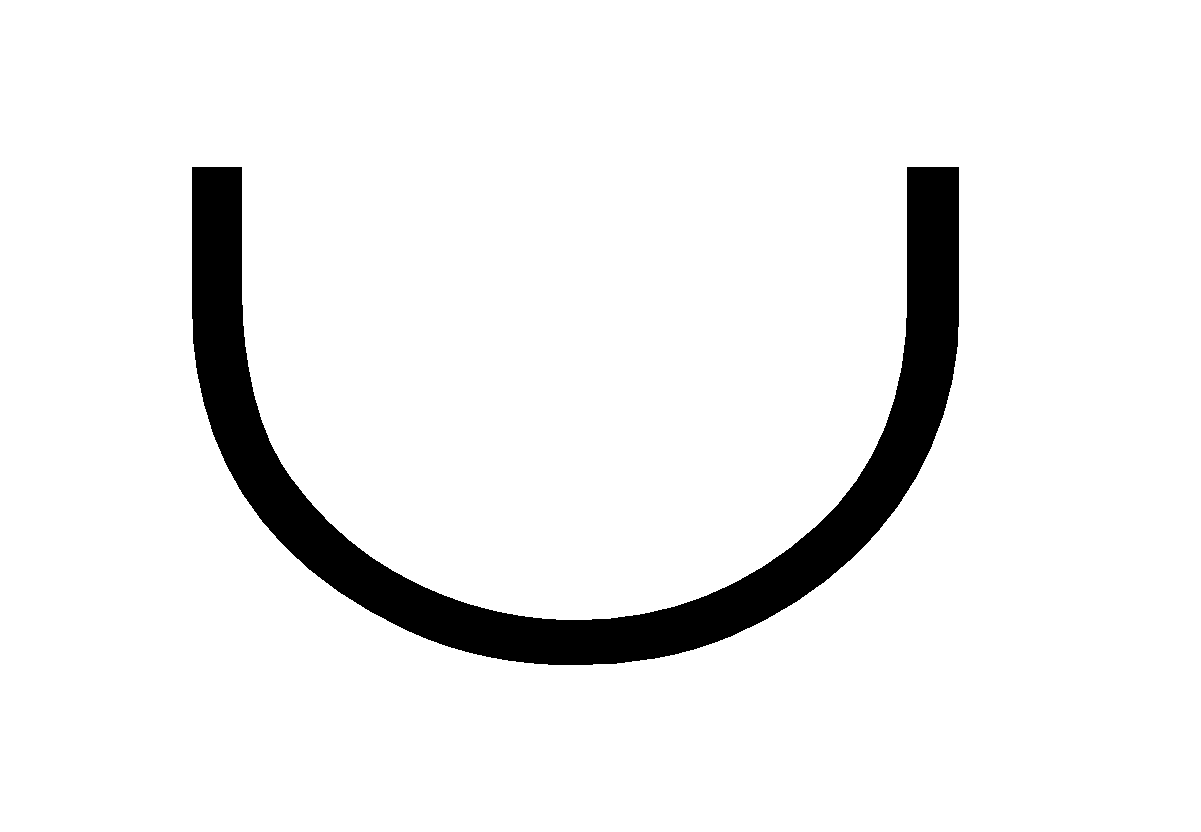
в) расщепления вершины

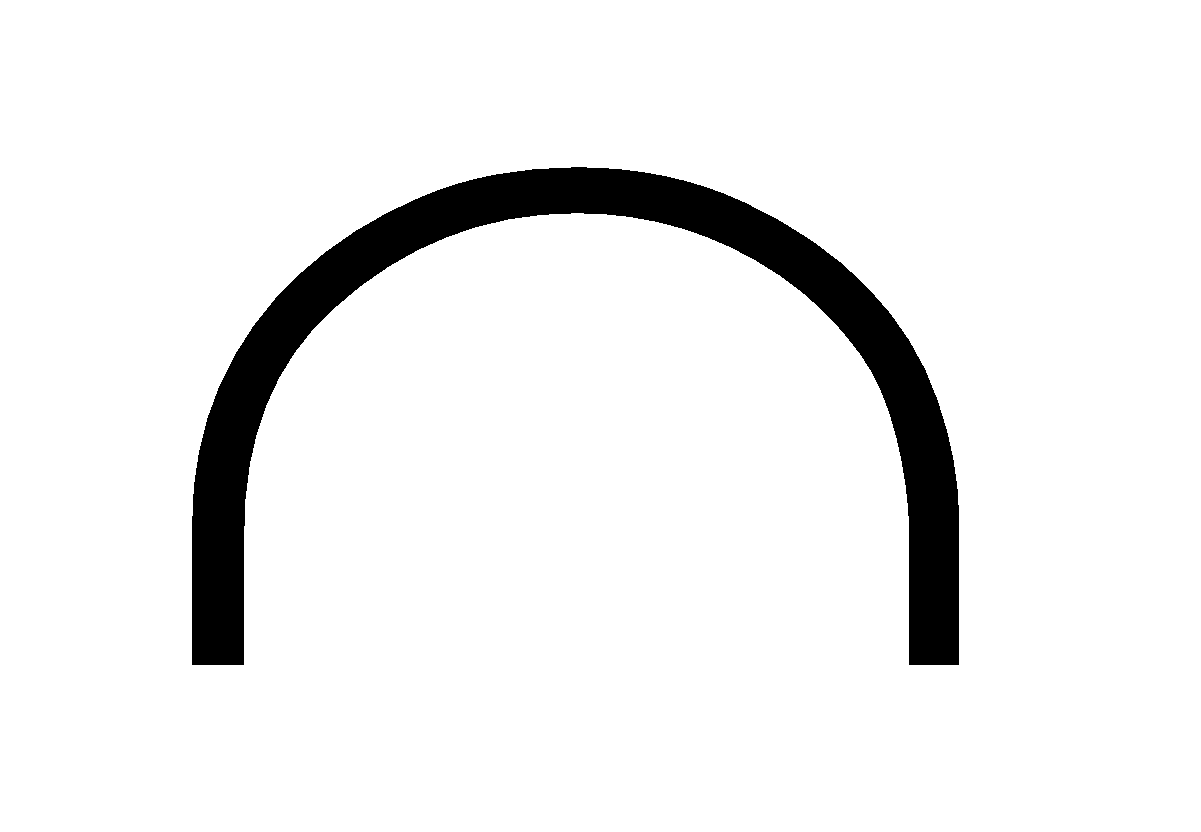
Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

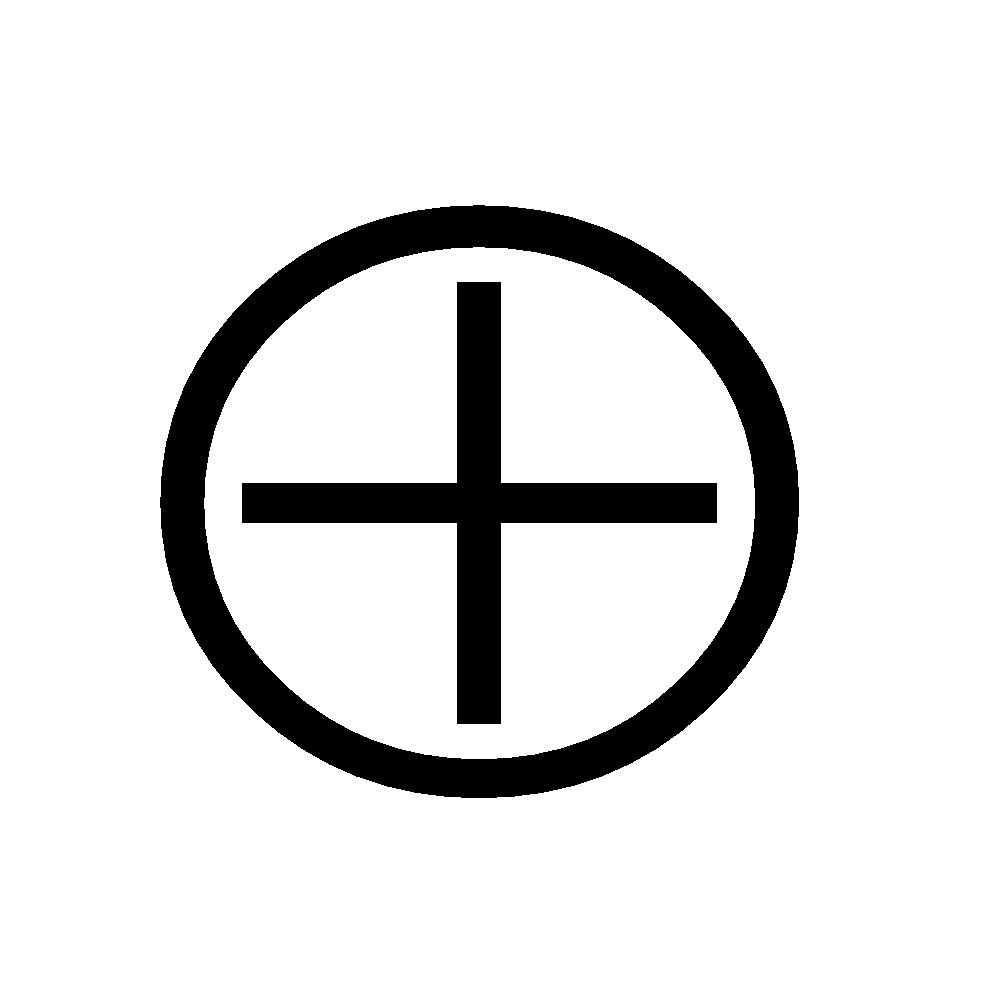
Результат выполнения операции выведите на экран.

**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.

**Задание 1:**

**Листинг**

#include <ctime>

#include <locale>

#include <iostream>

#include <vector>

#include <map>

#include <algorithm>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int count = 3;

cout << "Введите кол-во вершин 1 массива " << endl;

cin >> count;

if (count < 3) count = 3;

int\*\* arr1 = new int\* [count];

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

int\* a = new int[count] {0};

arr1[i] = a;

}

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

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

int p = rand() % 101;

if (i == j) {

arr1[i][j] = 0;

continue;

}

if (p > 70) {

int weight = rand() % 10;

arr1[i][j] = weight;

arr1[j][i] = weight;

}

}

}

for (int i = 0; i != count; i++) {

for (int j = 0; j != count; j++) {

cout << arr1[i][j] << ' ';

}

cout << endl;

}

int diam = 0;

int radi = 10000;

int\* d = new int[count]; // минимальное расстояние

int\* v = new int[count]; // посещенные вершины

int temp, minindex, min;

vector<int> max\_top; // Переферифные вершины

vector<int> min\_top; // Центральные вершины

// алгоритм Дейскстры

for (int i = 0; i != count; i++) {

int begin\_index = i;

//Инициализация вершин и расстояний

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

{

d[i] = 10000;

v[i] = 1;

}

d[begin\_index] = 0;

// Шаг алгоритма

do {

minindex = 10000;

min = 10000;

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

{ // Если вершину ещё не обошли и вес меньше min

if ((v[i] == 1) && (d[i] < min))

{ // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != 10000)

{

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

{

if (arr1[minindex][i] > 0)

{

temp = min + arr1[minindex][i];

if (temp < d[i])

{

d[i] = temp;

}

}

}

v[minindex] = 0;

}

} while (minindex < 10000);

// Вывод кратчайших расстояний до вершин

printf("\nКратчайшие расстояния до вершин: \n");

for (int i = 0; i < count; i++) printf("%-6d ", d[i]);

cout << endl;

for (int i = 0; i != count; i++) { // находим Переферийные и центральные вершины

if (d[i] != 10000 && diam < d[i]) {

diam = d[i];

max\_top.clear();

max\_top.push\_back(i);

}

if (max\_top.size() != 0 && diam == d[i] && find(max\_top.begin(), max\_top.end(), i) == max\_top.end()) max\_top.push\_back(i);

if (d[i] != 0 && radi > d[i]) {

radi = d[i];

min\_top.clear();

min\_top.push\_back(i);

}

if (min\_top.size() != 0 && radi == d[i] && find(min\_top.begin(), min\_top.end(), i) == min\_top.end()) min\_top.push\_back(i);

}

}

cout << endl << endl << "Диаметр = " << diam << endl;

cout << "Радиус = " << radi << endl;

cout << "периферийные вершины: ";

for (int i = 0; i != max\_top.size(); i++) {

cout << max\_top[i] << ' ';

}

cout << endl << "Центральные вершины: ";

for (int i = 0; i != min\_top.size(); i++) {

cout << min\_top[i] << ' ';

}

cout << endl;

for (int i = 0; i != count; i++) {

int edge\_count = 0;

for (int j = 0; j != count; j++) {

if (arr1[i][j] != 0) edge\_count++;

}

if (edge\_count == 0) cout << "Вершина " << i << " Изолированная" << endl;

else if (edge\_count == 1) cout << "Вершина " << i << " Концевая" << endl;

else if (edge\_count == count - 1) cout << "Вершина " << i << " Доминирующая " << endl;

else cout << "Вершина " << i << " просто вершина" << endl;

}

//

//Доп. задание

vector< vector< int> > arr2;

arr2.resize(count);

for (int i = 0; i != count; i++) {

for (int j = 0; j != count; j++) {

if (arr1[i][j] != 0) {

arr2[i].push\_back(j);

arr2[i].push\_back(arr1[i][j]);

}

}

}

for (int i = 0; i != count; i++) {

for (int j = 0; j < arr2[i].size(); j+=2) {

cout << arr2[i][j] << ' ';

}

cout << endl;

}

diam = 0;

radi = 10000;

max\_top.clear(); // Переферифные вершины

min\_top.clear(); // Центральные вершины

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

int begin\_index = i;

//Инициализация вершин и расстояний

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

{

d[i] = 10000;

v[i] = 1;

}

d[begin\_index] = 0;

// Шаг алгоритма

do {

minindex = 10000;

min = 10000;

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

{ // Если вершину ещё не обошли и вес меньше min

if ((v[i] == 1) && (d[i] < min))

{ // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != 10000)

{

int j = 1;

int x = 0;

for (int i = 0; i < arr2[minindex].size(); i += 2)

{

if (arr2[minindex][j] > 0)

{

temp = min + arr2[minindex][j];

if (temp < d[arr2[minindex][i]])

{

d[arr2[minindex][i]] = temp;

}

}

x++;

j += 2;

}

v[minindex] = 0;

}

} while (minindex < 10000);

// Вывод кратчайших расстояний до вершин

printf("\nКратчайшие расстояния до вершин: \n");

for (int i = 0; i < count; i++) printf("%-6d ", d[i]);

cout << endl;

for (int i = 0; i != count; i++) { // находим Переферийные и центральные вершины

if (d[i] != 10000 && diam < d[i]) {

diam = d[i];

max\_top.clear();

max\_top.push\_back(i);

}

if (max\_top.size() != 0 && diam == d[i] && find(max\_top.begin(), max\_top.end(), i) == max\_top.end()) max\_top.push\_back(i);

if (d[i] != 0 && radi > d[i]) {

radi = d[i];

min\_top.clear();

min\_top.push\_back(i);

}

if (min\_top.size() != 0 && radi == d[i] && find(min\_top.begin(), min\_top.end(), i) == min\_top.end()) min\_top.push\_back(i);

}

}

cout << endl << endl << "Диаметр = " << diam << endl;

cout << "Радиус = " << radi << endl;

cout << "периферийные вершины: ";

for (int i = 0; i != max\_top.size(); i++) {

cout << max\_top[i] << ' ';

}

cout << endl << "Центральные вершины: ";

for (int i = 0; i != min\_top.size(); i++) {

cout << min\_top[i] << ' ';

}

cout << endl;

for (int i = 0; i != count; i++) {

int edge\_count = arr2[i].size() / 2;

if (edge\_count == 0) cout << "Вершина " << i << " Изолированная" << endl;

else if (edge\_count == 1) cout << "Вершина " << i << " Концевая" << endl;

else if (edge\_count == count - 1) cout << "Вершина " << i << " Доминирующая " << endl;

else cout << "Вершина " << i << " просто вершина" << endl;

}

return 0;

}

**Результаты работы программы:**

Результаты работы программы показан на рисунке 1.

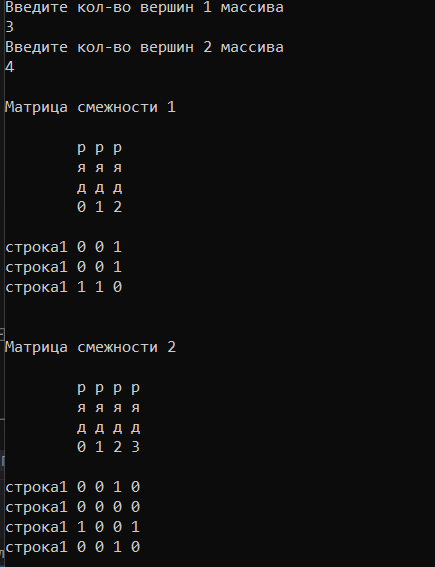
****

Рис. 1

Результат получился правильный, все действия совершены верно.

Результат работы программы, показанный на рисунке 1.

**Задание 2:**

**Листинг**

#include <ctime>

#include <locale>

#include <iostream>

#include <vector>

#include <map>

#include <algorithm>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int count = 3;

cout << "Введите кол-во вершин 1 массива " << endl;

cin >> count;

if (count < 3) count = 3;

int\*\* arr1 = new int\* [count];

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

int\* a = new int[count] {0};

arr1[i] = a;

}

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

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

int p = rand() % 101;

if (i == j) {

arr1[i][j] = 0;

continue;

}

if (p > 70) {

int weight = rand() % 10;

arr1[i][j] = weight;

arr1[j][i] = weight;

}

}

}

for (int i = 0; i != count; i++) {

for (int j = 0; j != count; j++) {

cout << arr1[i][j] << ' ';

}

cout << endl;

}

int diam = 0;

int radi = 10000;

int\* d = new int[count]; // минимальное расстояние

int\* v = new int[count]; // посещенные вершины

int temp, minindex, min;

vector<int> max\_top; // Переферифные вершины

vector<int> min\_top; // Центральные вершины

// алгоритм Дейскстры

for (int i = 0; i != count; i++) {

int begin\_index = i;

//Инициализация вершин и расстояний

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

{

d[i] = 10000;

v[i] = 1;

}

d[begin\_index] = 0;

// Шаг алгоритма

do {

minindex = 10000;

min = 10000;

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

{ // Если вершину ещё не обошли и вес меньше min

if ((v[i] == 1) && (d[i] < min))

{ // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != 10000)

{

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

{

if (arr1[minindex][i] > 0)

{

temp = min + arr1[minindex][i];

if (temp < d[i])

{

d[i] = temp;

}

}

}

v[minindex] = 0;

}

} while (minindex < 10000);

// Вывод кратчайших расстояний до вершин

printf("\nКратчайшие расстояния до вершин: \n");

for (int i = 0; i < count; i++) printf("%-6d ", d[i]);

cout << endl;

for (int i = 0; i != count; i++) { // находим Переферийные и центральные вершины

if (d[i] != 10000 && diam < d[i]) {

diam = d[i];

max\_top.clear();

max\_top.push\_back(i);

}

if (max\_top.size() != 0 && diam == d[i] && find(max\_top.begin(), max\_top.end(), i) == max\_top.end()) max\_top.push\_back(i);

if (d[i] != 0 && radi > d[i]) {

radi = d[i];

min\_top.clear();

min\_top.push\_back(i);

}

if (min\_top.size() != 0 && radi == d[i] && find(min\_top.begin(), min\_top.end(), i) == min\_top.end()) min\_top.push\_back(i);

}

}

cout << endl << endl << "Диаметр = " << diam << endl;

cout << "Радиус = " << radi << endl;

cout << "периферийные вершины: ";

for (int i = 0; i != max\_top.size(); i++) {

cout << max\_top[i] << ' ';

}

cout << endl << "Центральные вершины: ";

for (int i = 0; i != min\_top.size(); i++) {

cout << min\_top[i] << ' ';

}

cout << endl;

for (int i = 0; i != count; i++) {

int edge\_count = 0;

for (int j = 0; j != count; j++) {

if (arr1[i][j] != 0) edge\_count++;

}

if (edge\_count == 0) cout << "Вершина " << i << " Изолированная" << endl;

else if (edge\_count == 1) cout << "Вершина " << i << " Концевая" << endl;

else if (edge\_count == count - 1) cout << "Вершина " << i << " Доминирующая " << endl;

else cout << "Вершина " << i << " просто вершина" << endl;

}

//

//Доп. задание

vector< vector< int> > arr2;

arr2.resize(count);

for (int i = 0; i != count; i++) {

for (int j = 0; j != count; j++) {

if (arr1[i][j] != 0) {

arr2[i].push\_back(j);

arr2[i].push\_back(arr1[i][j]);

}

}

}

for (int i = 0; i != count; i++) {

for (int j = 0; j < arr2[i].size(); j+=2) {

cout << arr2[i][j] << ' ';

}

cout << endl;

}

diam = 0;

radi = 10000;

max\_top.clear(); // Переферифные вершины

min\_top.clear(); // Центральные вершины

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

int begin\_index = i;

//Инициализация вершин и расстояний

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

{

d[i] = 10000;

v[i] = 1;

}

d[begin\_index] = 0;

// Шаг алгоритма

do {

minindex = 10000;

min = 10000;

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

{ // Если вершину ещё не обошли и вес меньше min

if ((v[i] == 1) && (d[i] < min))

{ // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != 10000)

{

int j = 1;

int x = 0;

for (int i = 0; i < arr2[minindex].size(); i += 2)

{

if (arr2[minindex][j] > 0)

{

temp = min + arr2[minindex][j];

if (temp < d[arr2[minindex][i]])

{

d[arr2[minindex][i]] = temp;

}

}

x++;

j += 2;

}

v[minindex] = 0;

}

} while (minindex < 10000);

// Вывод кратчайших расстояний до вершин

printf("\nКратчайшие расстояния до вершин: \n");

for (int i = 0; i < count; i++) printf("%-6d ", d[i]);

cout << endl;

for (int i = 0; i != count; i++) { // находим Переферийные и центральные вершины

if (d[i] != 10000 && diam < d[i]) {

diam = d[i];

max\_top.clear();

max\_top.push\_back(i);

}

if (max\_top.size() != 0 && diam == d[i] && find(max\_top.begin(), max\_top.end(), i) == max\_top.end()) max\_top.push\_back(i);

if (d[i] != 0 && radi > d[i]) {

radi = d[i];

min\_top.clear();

min\_top.push\_back(i);

}

if (min\_top.size() != 0 && radi == d[i] && find(min\_top.begin(), min\_top.end(), i) == min\_top.end()) min\_top.push\_back(i);

}

}

cout << endl << endl << "Диаметр = " << diam << endl;

cout << "Радиус = " << radi << endl;

cout << "периферийные вершины: ";

for (int i = 0; i != max\_top.size(); i++) {

cout << max\_top[i] << ' ';

}

cout << endl << "Центральные вершины: ";

for (int i = 0; i != min\_top.size(); i++) {

cout << min\_top[i] << ' ';

}

cout << endl;

for (int i = 0; i != count; i++) {

int edge\_count = arr2[i].size() / 2;

if (edge\_count == 0) cout << "Вершина " << i << " Изолированная" << endl;

else if (edge\_count == 1) cout << "Вершина " << i << " Концевая" << endl;

else if (edge\_count == count - 1) cout << "Вершина " << i << " Доминирующая " << endl;

else cout << "Вершина " << i << " просто вершина" << endl;

}

return 0;

}

**Результаты работы программы:**

Результаты работы программы показан на рисунке 1.

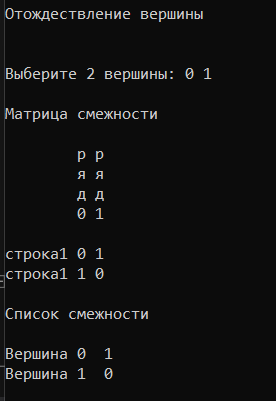
****

Рис. 1

Результат получился правильный, все действия совершены верно.

Результат работы программы, показанный на рисунке 1.

**Задание 3:**

**Листинг**

#include <ctime>

#include <locale>

#include <iostream>

#include <vector>

#include <map>

#include <algorithm>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int count = 3;

cout << "Введите кол-во вершин 1 массива " << endl;

cin >> count;

if (count < 3) count = 3;

int\*\* arr1 = new int\* [count];

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

int\* a = new int[count] {0};

arr1[i] = a;

}

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

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

int p = rand() % 101;

if (i == j) {

arr1[i][j] = 0;

continue;

}

if (p > 70) {

int weight = rand() % 10;

arr1[i][j] = weight;

arr1[j][i] = weight;

}

}

}

for (int i = 0; i != count; i++) {

for (int j = 0; j != count; j++) {

cout << arr1[i][j] << ' ';

}

cout << endl;

}

int diam = 0;

int radi = 10000;

int\* d = new int[count]; // минимальное расстояние

int\* v = new int[count]; // посещенные вершины

int temp, minindex, min;

vector<int> max\_top; // Переферифные вершины

vector<int> min\_top; // Центральные вершины

// алгоритм Дейскстры

for (int i = 0; i != count; i++) {

int begin\_index = i;

//Инициализация вершин и расстояний

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

{

d[i] = 10000;

v[i] = 1;

}

d[begin\_index] = 0;

// Шаг алгоритма

do {

minindex = 10000;

min = 10000;

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

{ // Если вершину ещё не обошли и вес меньше min

if ((v[i] == 1) && (d[i] < min))

{ // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != 10000)

{

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

{

if (arr1[minindex][i] > 0)

{

temp = min + arr1[minindex][i];

if (temp < d[i])

{

d[i] = temp;

}

}

}

v[minindex] = 0;

}

} while (minindex < 10000);

// Вывод кратчайших расстояний до вершин

printf("\nКратчайшие расстояния до вершин: \n");

for (int i = 0; i < count; i++) printf("%-6d ", d[i]);

cout << endl;

for (int i = 0; i != count; i++) { // находим Переферийные и центральные вершины

if (d[i] != 10000 && diam < d[i]) {

diam = d[i];

max\_top.clear();

max\_top.push\_back(i);

}

if (max\_top.size() != 0 && diam == d[i] && find(max\_top.begin(), max\_top.end(), i) == max\_top.end()) max\_top.push\_back(i);

if (d[i] != 0 && radi > d[i]) {

radi = d[i];

min\_top.clear();

min\_top.push\_back(i);

}

if (min\_top.size() != 0 && radi == d[i] && find(min\_top.begin(), min\_top.end(), i) == min\_top.end()) min\_top.push\_back(i);

}

}

cout << endl << endl << "Диаметр = " << diam << endl;

cout << "Радиус = " << radi << endl;

cout << "периферийные вершины: ";

for (int i = 0; i != max\_top.size(); i++) {

cout << max\_top[i] << ' ';

}

cout << endl << "Центральные вершины: ";

for (int i = 0; i != min\_top.size(); i++) {

cout << min\_top[i] << ' ';

}

cout << endl;

for (int i = 0; i != count; i++) {

int edge\_count = 0;

for (int j = 0; j != count; j++) {

if (arr1[i][j] != 0) edge\_count++;

}

if (edge\_count == 0) cout << "Вершина " << i << " Изолированная" << endl;

else if (edge\_count == 1) cout << "Вершина " << i << " Концевая" << endl;

else if (edge\_count == count - 1) cout << "Вершина " << i << " Доминирующая " << endl;

else cout << "Вершина " << i << " просто вершина" << endl;

}

//

//Доп. задание

vector< vector< int> > arr2;

arr2.resize(count);

for (int i = 0; i != count; i++) {

for (int j = 0; j != count; j++) {

if (arr1[i][j] != 0) {

arr2[i].push\_back(j);

arr2[i].push\_back(arr1[i][j]);

}

}

}

for (int i = 0; i != count; i++) {

for (int j = 0; j < arr2[i].size(); j+=2) {

cout << arr2[i][j] << ' ';

}

cout << endl;

}

diam = 0;

radi = 10000;

max\_top.clear(); // Переферифные вершины

min\_top.clear(); // Центральные вершины

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

int begin\_index = i;

//Инициализация вершин и расстояний

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

{

d[i] = 10000;

v[i] = 1;

}

d[begin\_index] = 0;

// Шаг алгоритма

do {

minindex = 10000;

min = 10000;

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

{ // Если вершину ещё не обошли и вес меньше min

if ((v[i] == 1) && (d[i] < min))

{ // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != 10000)

{

int j = 1;

int x = 0;

for (int i = 0; i < arr2[minindex].size(); i += 2)

{

if (arr2[minindex][j] > 0)

{

temp = min + arr2[minindex][j];

if (temp < d[arr2[minindex][i]])

{

d[arr2[minindex][i]] = temp;

}

}

x++;

j += 2;

}

v[minindex] = 0;

}

} while (minindex < 10000);

// Вывод кратчайших расстояний до вершин

printf("\nКратчайшие расстояния до вершин: \n");

for (int i = 0; i < count; i++) printf("%-6d ", d[i]);

cout << endl;

for (int i = 0; i != count; i++) { // находим Переферийные и центральные вершины

if (d[i] != 10000 && diam < d[i]) {

diam = d[i];

max\_top.clear();

max\_top.push\_back(i);

}

if (max\_top.size() != 0 && diam == d[i] && find(max\_top.begin(), max\_top.end(), i) == max\_top.end()) max\_top.push\_back(i);

if (d[i] != 0 && radi > d[i]) {

radi = d[i];

min\_top.clear();

min\_top.push\_back(i);

}

if (min\_top.size() != 0 && radi == d[i] && find(min\_top.begin(), min\_top.end(), i) == min\_top.end()) min\_top.push\_back(i);

}

}

cout << endl << endl << "Диаметр = " << diam << endl;

cout << "Радиус = " << radi << endl;

cout << "периферийные вершины: ";

for (int i = 0; i != max\_top.size(); i++) {

cout << max\_top[i] << ' ';

}

cout << endl << "Центральные вершины: ";

for (int i = 0; i != min\_top.size(); i++) {

cout << min\_top[i] << ' ';

}

cout << endl;

for (int i = 0; i != count; i++) {

int edge\_count = arr2[i].size() / 2;

if (edge\_count == 0) cout << "Вершина " << i << " Изолированная" << endl;

else if (edge\_count == 1) cout << "Вершина " << i << " Концевая" << endl;

else if (edge\_count == count - 1) cout << "Вершина " << i << " Доминирующая " << endl;

else cout << "Вершина " << i << " просто вершина" << endl;

}

return 0;

}

**Результаты работы программы:**

Результаты работы программы показан на рисунке 1.

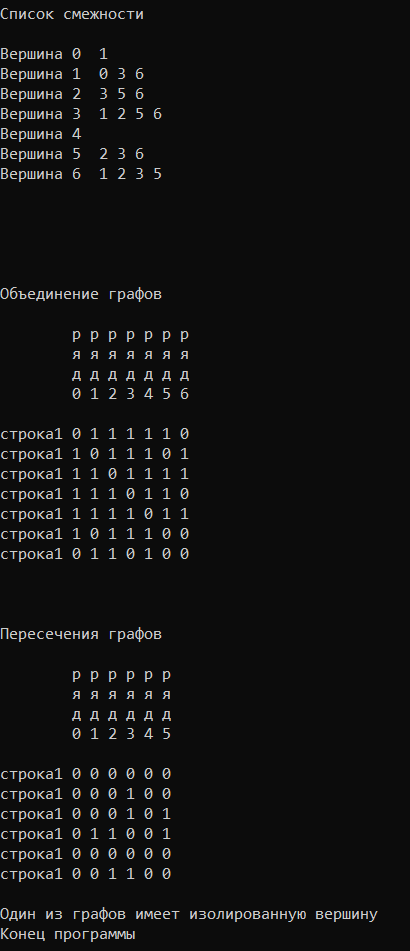
****

Рис. 1

Результат получился правильный, все действия совершены верно.

Результат работы программы, показанный на рисунке 1.

**Вывод:** мы выполнили лабораторные указания 1-3 используя знания использования бинарных и унарных функций над графами, где повторили приобретенные навыки с предыдущего курса.