Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

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

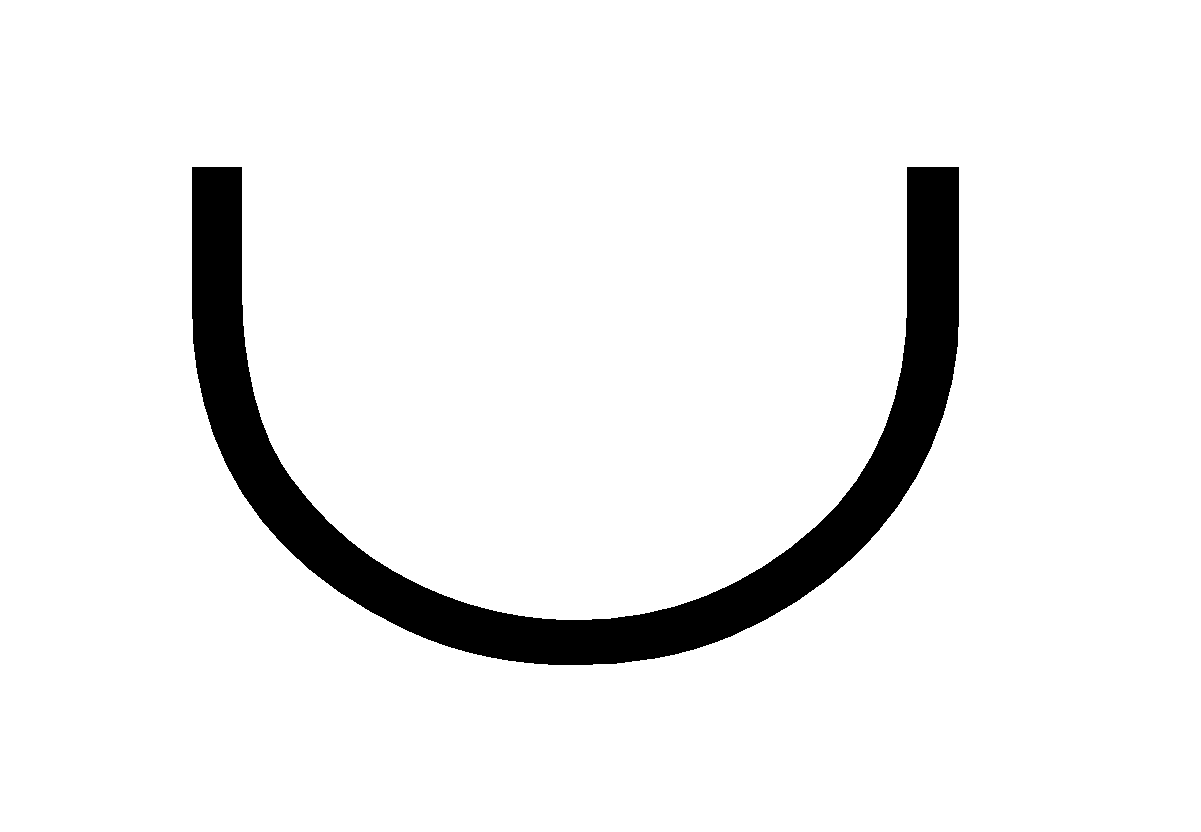
в) расщепления вершины

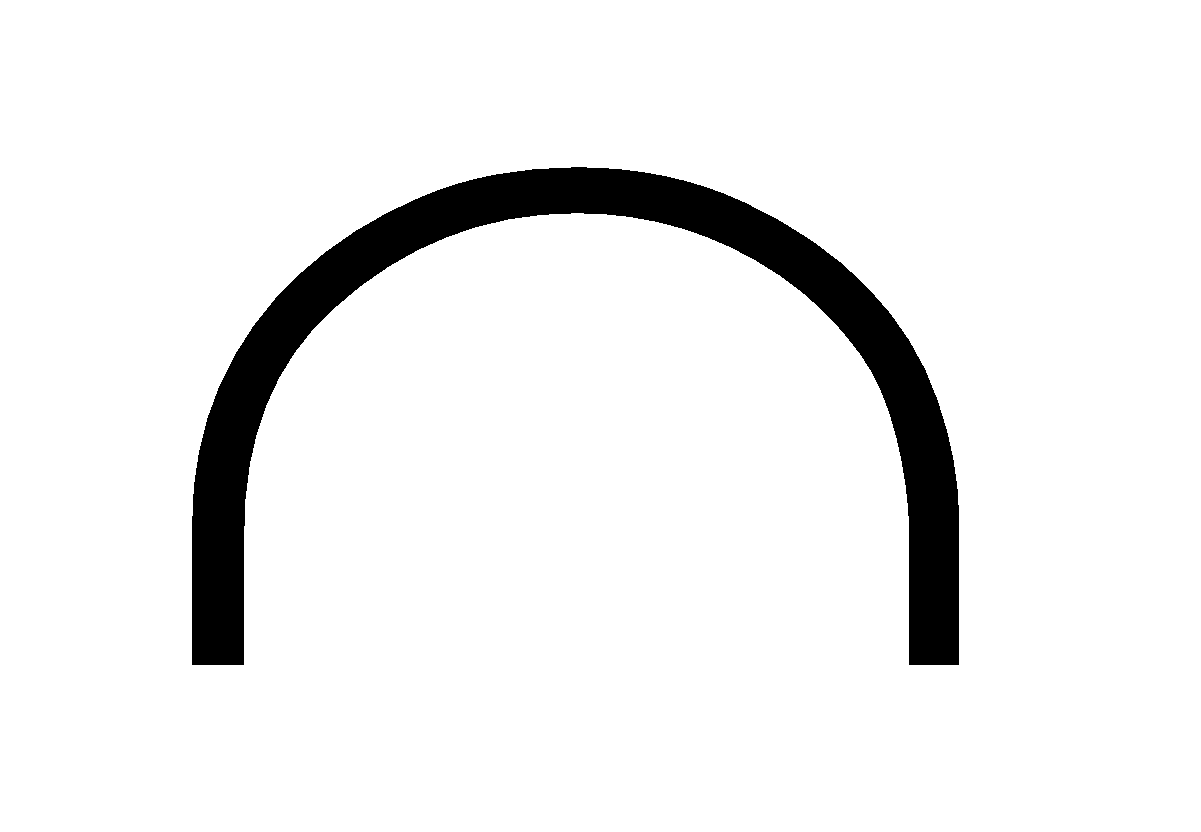
Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

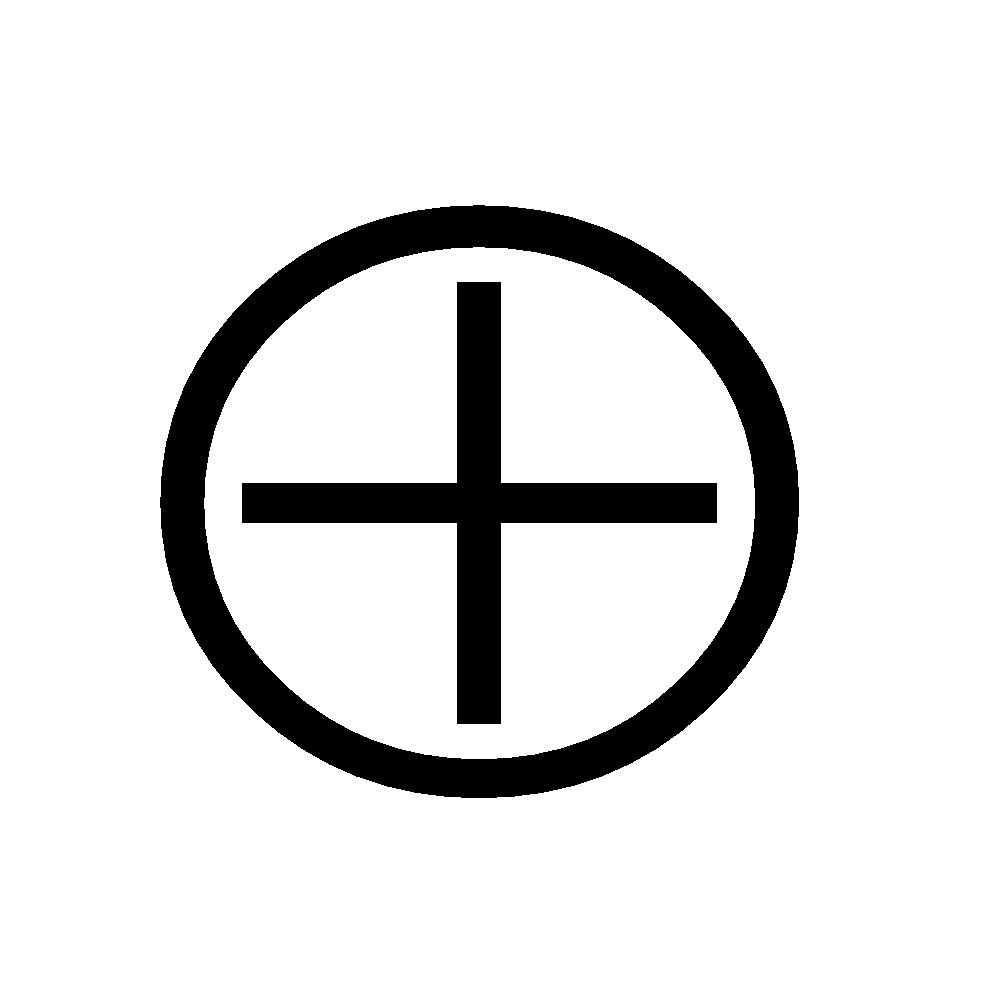
Результат выполнения операции выведите на экран.

**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 <iostream>

void Print1(int\* arr1, int size) {

static int\* arr = nullptr;

static int x = 0;

if (arr != arr1 - x) {

arr = arr1;

x = size;

cout << '\t';

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

cout << 'р' << ' ';

}

cout << endl << '\t';

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

cout << 'я' << ' ';

}

cout << endl << '\t';

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

cout << 'д' << ' ';

}

cout << endl << '\t';

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

cout << i + 1 << ' ';

}

cout << endl;

}

else x += size;

//

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

if (!j) cout << endl << "строка" << j + 1 << " ";

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

}

}

void Print2(int \*arr1, int \*arr2, int size) {

static int\* arr = nullptr;

static int x = 0;

if (arr != arr1 - x) {

arr = arr1;

x = size;

cout << endl << '\t';

for (int i = 0; i != size \* 2 + 1; i++) {

if (i == 5) {

cout << '\t' << '\t';

continue;

}

cout << 'р' << ' ';

}

cout << endl << '\t';

for (int i = 0; i != size \* 2 + 1; i++) {

if (i == 5) {

cout << '\t' << '\t';

continue;

}

cout << 'я' << ' ';

}

cout << endl << '\t';

for (int i = 0; i != size \* 2 + 1; i++) {

if (i == 5) {

cout << '\t' << '\t';

continue;

}

cout << 'д' << ' ';

}

cout << endl << '\t';

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

cout << i + 1 << ' ';

}

cout << '\t' << '\t';

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

cout << i + 1 << ' ';

}

cout << endl << endl;

}

else x += size;

//

for (int j = 0; j != size \* 2; j++) {

if(!j) cout << "строка" << j + 1 << " ";

if (j < size) {

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

}

else {

if (j == size) cout << '\t' << "строка" << j - 5 << " ";

cout << arr2[j - 5] << ' ';

}

}

}

int main()

{

srand(time(NULL));

system("chcp 1251");

system("cls");

// 1 матрица

int arr1[5][5]{ 0 };

for (int i = 0; i != 5; i++) {

int flag = 0; //если есть несвязанная вершина

for (int j = 0; j != 5; j++) {

int p = rand() % 101;

if (i == j) {

arr1[i][j] = 0;

continue;

}

if (arr1[i][j] == 1) {

flag++;

continue;

}

if (p > 65) {

flag++;

arr1[i][j] = 1;

arr1[j][i] = 1;

}

}

if (flag == 0) {

if (i <= 3) {

arr1[i][i + 1] = 1; //тк больше 4 не будет

arr1[i + 1][i] = 1;

}

else {

arr1[i][i - 1] = 1;

arr1[i - 1][i] = 1;

}

}

}

// 2 матрица

int arr2[5][5]{ 0 };

for (int i = 0; i != 5; i++) {

int flag = 0; //если есть несвязанная вершина

for (int j = 0; j != 5; j++) {

int p = rand() % 101;

if (i == j) {

arr2[i][j] = 0;

continue;

}

if (arr2[i][j] == 1) {

flag++;

continue;

}

if (p > 65) {

flag++;

arr2[i][j] = 1;

arr2[j][i] = 1;

}

}

if (flag == 0) {

if (i <= 3) {

arr2[i][i + 1] = 1; //тк больше 4 не будет

arr2[i + 1][i] = 1;

}

else {

arr2[i][i - 1] = 1;

arr2[i - 1][i] = 1;

}

}

}

for (int i = 0; i != 5; i++) {

Print2(arr1[i], arr2[i], 5);

cout << endl;

}

// задание 2.1 отождествление вершины

int top[2]; // вводим вершины

for(;;) {

cout << endl << endl << "Выберите 2 вершины: ";

cin >> top[0];

cin >> top[1];

if (top[0] > top[1]) {

int tmp = top[0];

top[0] = top[1];

top[1] = tmp;

}

if (top[0] == top[1] || !cin.good()) {

cout << "Вы ввели неправильное значение" << endl;

cin.clear();

cin.ignore(10, '\n');

continue;

}

break;

}

int anti\_top[3]; // тут остаются не введеные ввершины

for (int i = 0, j = 0; i != 5; i++, j++) {

if (i != top[0] && i != top[1]) anti\_top[j] = i;

else j--;

}

int arr3[4][4]{ 0 }; // результат отождествления

for (int i = 0; i != 2; i++) {

for (int j = 0, x = 0; j != 5; j++, x++) {

// результат соединения двух вершин записывается в наименьший из двух вершин

if (top[0] == j) continue;

if (top[1] == j && j != 4) j++; // игнорируем 2 вершину т.к. она удаляется

if (top[1] == 4 && j == 4) x--;

if (arr1[top[i]][j]) {

arr3[top[0]][x] = 1; // заполняем слияние двух вершин

arr3[x][top[0]] = 1; // остальные вершины если пересекались хотя бы с одним из 2 сливаемых вершин

}

}

}

arr3[top[0]][top[0]] = 0;

for (int i = 0; i != 3; i++) {

for (int j = 0, y = 0; j != 3; j++, y++) {

// цикл проходит по оставшимся вершинам

if (arr1[anti\_top[i]][anti\_top[j]]) {

if (anti\_top[j] > top[1] && anti\_top[i] > top[1]) {

arr3[anti\_top[i] - 1][anti\_top[j] - 1] = 1;

arr3[anti\_top[j] - 1][anti\_top[i] - 1] = 1;

}

else if (anti\_top[j] > top[1]) {

arr3[anti\_top[i]][anti\_top[j] - 1] = 1;

arr3[anti\_top[j] - 1][anti\_top[i]] = 1;

}

else if (anti\_top[i] > top[1]) {

arr3[anti\_top[i] - 1][anti\_top[j]] = 1;

arr3[anti\_top[j]][anti\_top[i] - 1] = 1;

}

else {

arr3[anti\_top[i]][anti\_top[j]] = 1;

arr3[anti\_top[j]][anti\_top[i]] = 1;

}

}

}

}

// выводим цикл на экран

for (int i = 0; i != 4; i++) {

Print1(arr3[i], 4);

}

// задание 2.2 стягивания ребра

int top2[2]; // вводим вершины

for (;;) {

cout << endl << endl << "Выберите 2 вершины: ";

cin >> top2[0];

cin >> top2[1];

if (top2[0] > top2[1]) {

int tmp = top2[0];

top2[0] = top2[1];

top2[1] = tmp;

}

if (top2[0] == top2[1] || !cin.good()) {

cout << "Вы ввели неправильное значение" << endl;

cin.clear();

cin.ignore(10, '\n');

continue;

}

if (!(arr1[top2[0]][top2[1]] && arr1[top2[1]][top2[0]])) {

cout << "Данные вершины не образуют ребро" << endl;

continue;

}

break;

}

int anti\_top2[3]; // тут остаются не введеные ввершины

for (int i = 0, j = 0; i != 5; i++, j++) {

if (i != top2[0] && i != top2[1]) anti\_top2[j] = i;

else j--;

}

int arr4[4][4]{ 0 }; // результат отождествления

for (int i = 0; i != 2; i++) {

for (int j = 0, x = 0; j != 5; j++, x++) {

// результат соединения двух вершин записывается в наименьший из двух вершин

if (top2[0] == j) continue;

if (top2[1] == j && j != 4) j++; // игнорируем 2 вершину т.к. она удаляется

if (top2[1] == 4 && j == 4) x--;

if (arr1[top2[i]][j]) {

arr4[top2[0]][x] = 1; // заполняем слияние двух вершин

arr4[x][top2[0]] = 1; // остальные вершины если пересекались хотя бы с одним из 2 сливаемых вершин

}

}

}

arr4[top2[0]][top2[0]] = 0;

for (int i = 0; i != 3; i++) {

for (int j = 0, y = 0; j != 3; j++, y++) {

// цикл проходит по оставшимся вершинам

if (arr1[anti\_top2[i]][anti\_top2[j]]) {

if (anti\_top2[j] > top2[1] && anti\_top2[i] > top2[1]) {

arr4[anti\_top2[i] - 1][anti\_top2[j] - 1] = 1;

arr4[anti\_top2[j] - 1][anti\_top2[i] - 1] = 1;

}

else if (anti\_top2[j] > top2[1]) {

arr4[anti\_top2[i]][anti\_top2[j] - 1] = 1;

arr4[anti\_top2[j] - 1][anti\_top2[i]] = 1;

}

else if (anti\_top2[i] > top2[1]) {

arr4[anti\_top2[i] - 1][anti\_top2[j]] = 1;

arr4[anti\_top2[j]][anti\_top2[i] - 1] = 1;

}

else {

arr4[anti\_top2[i]][anti\_top2[j]] = 1;

arr4[anti\_top2[j]][anti\_top2[i]] = 1;

}

}

}

}

// задание 2.3 расщепления вершины

int arr5[6][6] { 0 }; // новый массив с расщеплением

int top3;

for (;;) {

cout << endl << endl << "Выберите 1 вершину: ";

cin >> top3;

if (top3 > 4 || top3 < 0 || !cin.good()) {

cout << "Вы ввели неправильное значение" << endl;

cin.clear();

cin.ignore(10, '\n');

continue;

}

break;

}

for (int i = 0; i != 5; i++) {

for (int j = 0; j != 5; j++) {

if (!i) {

arr5[0][j + 1] = arr1[top3][j];

arr5[j + 1][0] = arr1[top3][j];

}

if (arr1[i][j]) arr5[i + 1][j + 1] = 1;

}

}

arr5[0][top3 + 1] = 1;

arr5[top3 + 1][0] = 1;

cout << endl;

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

Print1(arr5[i], 6);

}

cout << endl;

// задание 3.1 объединения графов

int arr6[5][5]{ 0 };

for (int i = 0; i != 5; i++) {

for (int j = 0; j != 5; j++) {

if(arr1[i][j] || arr2[i][j]) arr6[i][j] = 1;

}

}

cout << endl << "Объединение графов" << endl << endl;

for (int i = 0; i != 5; i++) {

Print1(arr6[i], 5);

}

cout << endl;

// задание 3.2 пересечения графов

int arr7[5][5]{ 0 };

for (int i = 0; i != 5; i++) {

for (int j = i + 1; j != 5; j++) {

if (arr1[i][j] && arr2[i][j]) {

arr7[i][j] = 1;

arr7[j][i] = 1;

}

}

}

cout << endl << "Пересечения графов" << endl << endl;

for (int i = 0; i != 5; i++) {

Print1(arr7[i], 5);

}

cout << endl;

// задание 3.3 кольцевая сумма графов

stack<int> Stack1;

stack<int> Stack2;

Stack1.push(0);

Stack2.push(0);

int nodes1[5]{ 0 }; // состояние вершин

int nodes2[5]{ 0 }; // состояние вершин

while (!Stack1.empty() && !Stack2.empty()) {

int node1 = Stack1.top();

int node2 = Stack2.top();

Stack1.pop();

Stack2.pop();

if (nodes1[node1] == 2 && nodes2[node2]) continue; // уже были

nodes1[node1] = 2;

nodes2[node2] = 2;

for (int i = 0; i != 5; i++) {

if (arr1[node1][i] && nodes1[i] != 2) {

Stack1.push(i);

nodes1[i] = 1;

}

if (arr2[node2][i] && nodes2[i] != 2) {

Stack2.push(i);

nodes2[i] = 1;

}

}

}

int n = 0;

for (int i = 0; i != 5; i++) {

if (!nodes1[i] || !nodes2[i]) {

n++;

}

}

if (n > 0) {

printf("Один из графов имеет изолированн%s вершин%c \n", n == 1 ? "ую" : "ые", n == 1 ? 'у' : 'ы');

cout << "Конец программы" << endl;

return 0;

}

int arr8[5][5]{ 0 };

for (int i = 0; i != 5; i++) {

for (int j = i + 1; j != 5; j++) {

if (arr1[i][j] != arr2[i][j]) {

arr8[i][j] = 1;

arr8[j][i] = 1;

}

}

}

cout << endl << "Кольцевая сумма графов" << endl << endl;

for (int i = 0; i != 5; i++) {

Print1(arr8[i], 5);

}

cout << 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 <iostream>

#include <stack>

#include <set>

#include <vector>

#include <algorithm>

using namespace std;

void Print1(int\* a, int size) {

static int\* x = a;

int static steps = 0;

if (x + steps \* size != a) {

steps = 0;

x = a;

}

if (x == a) {

cout << '\t';

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

cout << 'р' << ' ';

}

cout << endl;

cout << '\t';

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

cout << 'я' << ' ';

}

cout << endl;

cout << '\t';

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

cout << 'д' << ' ';

}

cout << endl;

cout << '\t';

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

cout << i << ' ';

}

cout << endl << endl;

}

cout << "строка" << steps << ' ';

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

cout << a[i] << ' ';

}

cout << endl;

steps++;

}

void Print2(int \*a, int \*b, int size) {

static int\* x = a;

int static step = 0;

if (x == a) {

cout << endl << endl;

cout << '\t';

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

cout << 'р' << ' ';

}

cout << '\t' << '\t' << '\t';

cout << '\t';

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

cout << 'р' << ' ';

}

cout << endl;

cout << '\t';

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

cout << 'я' << ' ';

}

cout << '\t' << '\t' << '\t';

cout << '\t';

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

cout << 'я' << ' ';

}

cout << endl;

cout << '\t';

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

cout << 'д' << ' ';

}

cout << '\t' << '\t' << '\t';

cout << '\t';

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

cout << 'д' << ' ';

}

cout << endl;

cout << '\t';

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

cout << i << ' ';

}

cout << '\t' << '\t' << '\t';

cout << '\t';

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

cout << i << ' ';

}

cout << endl << endl;

}

cout << "строка" << step << ' ';

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

cout << a[i] << ' ';

}

cout << "\t\t\t";

cout << "строка" << step << ' ';

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

cout << b[i] << ' ';

}

cout << endl;

step++;

}

int main()

{

srand(time(NULL));

system("chcp 1251");

system("cls");

/\*

stack<int> Stack;

stack<Edge> Edges;

int req;

Edge e;

int mas[7][7] = { { 0, 1, 1, 0, 0, 0, 1 }, // матрица смежности

{ 1, 0, 1, 1, 0, 0, 0 },

{ 1, 1, 0, 0, 0, 0, 0 },

{ 0, 1, 0, 0, 1, 0, 0 },

{ 0, 0, 0, 1, 0, 1, 0 },

{ 0, 0, 0, 0, 1, 0, 1 },

{ 1, 0, 0, 0, 0, 1, 0 } };

int nodes[7]; // вершины графа

for (int i = 0; i < 7; i++) // исходно все вершины равны 0

nodes[i] = 0;

cout << "N = ";

cin >> req;

req--;

Stack.push(0); // помещаем в очередь первую вершину

while (!Stack.empty())

{ // пока стек не пуст

int node = Stack.top(); // извлекаем вершину

Stack.pop();

if (nodes[node] == 2) continue;

nodes[node] = 2; // отмечаем ее как посещенную

for (int j = 6; j >= 0; j--)

{ // проверяем для нее все смежные вершины

if (mas[node][j] == 1 && nodes[j] != 2)

{ // если вершина смежная и не обнаружена

Stack.push(j); // добавляем ее в cтек

nodes[j] = 1; // отмечаем вершину как обнаруженную

e.begin = node; e.end = j;

Edges.push(e);

if (node == req) break;

}

}

cout << node + 1 << endl; // выводим номер вершины

}

cout << "Путь до вершины " << req + 1 << endl;

cout << req + 1;

while (!Edges.empty())

{

e = Edges.top();

Edges.pop();

if (e.end == req)

{

req = e.begin;

cout << " <- " << req + 1;

}

}

cin.get(); cin.get();

\*/

// 1 матрица

int arr1[5][5]{ 0 };

for (int i = 0; i != 5; i++) {

int flag = 0; //если есть несвязанная вершина

for (int j = 0; j != 5; j++) {

int p = rand() % 101;

if (i == j) {

arr1[i][j] = 0;

continue;

}

if (arr1[i][j] == 1) {

flag++;

continue;

}

if (p > 65) {

flag++;

arr1[i][j] = 1;

arr1[j][i] = 1;

}

}

if (flag == 0) {

if (i <= 3) {

arr1[i][i + 1] = 1; //тк больше 4 не будет

arr1[i + 1][i] = 1;

}

else {

arr1[i][i - 1] = 1;

arr1[i - 1][i] = 1;

}

}

}

// 2 матрица

int arr2[5][5]{ 0 };

for (int i = 0; i != 5; i++) {

int flag = 0; //если есть несвязанная вершина

for (int j = 0; j != 5; j++) {

int p = rand() % 101;

if (i == j) {

arr2[i][j] = 0;

continue;

}

if (arr2[i][j] == 1) {

flag++;

continue;

}

if (p > 65) {

flag++;

arr2[i][j] = 1;

arr2[j][i] = 1;

}

}

if (flag == 0) {

if (i <= 3) {

arr2[i][i + 1] = 1; //тк больше 4 не будет

arr2[i + 1][i] = 1;

}

else {

arr2[i][i - 1] = 1;

arr2[i - 1][i] = 1;

}

}

}

//заполнение списка смежности

vector< vector<int> > arr\_list1;

arr\_list1.resize(5);

vector< vector<int> > arr\_list2;

arr\_list2.resize(5);

for (int i = 0; i != 5; i++) {

for (int j = 0; j != 5; j++) {

if (arr1[i][j]) {

arr\_list1[i].push\_back(j);

}

}

// // // // //

for (int j = 0; j != 5; j++) {

if (arr2[i][j]) {

arr\_list2[i].push\_back(j);

}

}

}

for (int i = 0; i != 5; i++) {

cout << "Вершина " << i << " ";

for(int j = 0; j < arr\_list1[i].size(); j++){

cout << arr\_list1[i][j] << ' ';

}

cout << endl;

}

cout << endl << endl;

for (int i = 0; i != 5; i++) {

cout << "Вершина " << i << " ";

for (int j = 0; j < arr\_list2[i].size(); j++) {

cout << arr\_list2[i][j] << ' ';

}

cout << endl;

}

for (int i = 0; i != 5; i++) {

Print2(arr1[i], arr2[i], 5);

}

**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**

int top[2]; // вводим вершины

for (;;) {

cout << endl << endl << "Выберите 2 вершины: ";

cin >> top[0];

cin >> top[1];

if (top[0] > top[1]) {

int tmp = top[0];

top[0] = top[1];

top[1] = tmp;

}

if (top[0] == top[1] || !cin.good()) {

cout << "Вы ввели неправильное значение" << endl;

cin.clear();

cin.ignore(10, '\n');

continue;

}

break;

}

int anti\_top[3]; // тут остаются не введеные ввершины

for (int i = 0, j = 0; i != 5; i++, j++) {

if (i != top[0] && i != top[1]) anti\_top[j] = i;

else j--;

}

int arr3[4][4]{ 0 }; // результат отождествления

for (int i = 0; i != 2; i++) {

for (int j = 0, x = 0; j != 5; j++, x++) {

// результат соединения двух вершин записывается в наименьший из двух вершин

if (top[0] == j) continue;

if (top[1] == j && j != 4) j++; // игнорируем 2 вершину т.к. она удаляется

if (top[1] == 4 && j == 4) x--;

if (arr1[top[i]][j]) {

arr3[top[0]][x] = 1; // заполняем слияние двух вершин

arr3[x][top[0]] = 1; // остальные вершины если пересекались хотя бы с одним из 2 сливаемых вершин

}

}

}

arr3[top[0]][top[0]] = 0;

for (int i = 0; i != 3; i++) {

for (int j = 0, y = 0; j != 3; j++, y++) {

// цикл проходит по оставшимся вершинам

if (arr1[anti\_top[i]][anti\_top[j]]) {

if (anti\_top[j] > top[1] && anti\_top[i] > top[1]) {

arr3[anti\_top[i] - 1][anti\_top[j] - 1] = 1;

arr3[anti\_top[j] - 1][anti\_top[i] - 1] = 1;

}

else if (anti\_top[j] > top[1]) {

arr3[anti\_top[i]][anti\_top[j] - 1] = 1;

arr3[anti\_top[j] - 1][anti\_top[i]] = 1;

}

else if (anti\_top[i] > top[1]) {

arr3[anti\_top[i] - 1][anti\_top[j]] = 1;

arr3[anti\_top[j]][anti\_top[i] - 1] = 1;

}

else {

arr3[anti\_top[i]][anti\_top[j]] = 1;

arr3[anti\_top[j]][anti\_top[i]] = 1;

}

}

}

}

// выводим цикл на экран

for (int i = 0; i != 4; i++) {

Print1(arr3[i], 4);

}

cout << endl << endl;

// отождествление вершин в списке смежности

/// /// ///

vector< vector<int> > arr\_list3;

arr\_list3.resize(5);

for (int i = 0; i != arr\_list1.size(); i++) { //копируем arr\_list1s

for (int j = 0; j != arr\_list1[i].size(); j++) {

arr\_list3[i].push\_back(arr\_list1[i][j]);

}

}

vector<int>::iterator itg = find(arr\_list3[top[0]].begin(), arr\_list3[top[0]].end(), top[1]); //связанна ли 1 вершина со 2-ой

if (itg != arr\_list3[top[0]].end()) {

arr\_list3[top[0]].erase(itg, itg+1);

}

for (vector<int>::iterator i = arr\_list3[top[1]].begin(); i != arr\_list3[top[1]].end(); i++) {

if (\*i == top[0]) continue;

itg = find(arr\_list3[top[0]].begin(), arr\_list3[top[0]].end(), \*i);

if (itg != arr\_list3[top[0]].end() && \*itg == \*i) continue; // уже есть

arr\_list3[top[0]].push\_back(\*i);

}

sort(arr\_list3[top[0]].begin(), arr\_list3[top[0]].end()); // сортируем вершины по порядку

for (int i = 0; i < sizeof(anti\_top) / 4; i++) {

itg = find(arr\_list3[anti\_top[i]].begin(), arr\_list3[anti\_top[i]].end(), top[1]);

int c = count(arr\_list3[anti\_top[i]].begin(), arr\_list3[anti\_top[i]].end(), top[0]);

if (itg != arr\_list3[anti\_top[i]].end()) { // соединен со 2-ой вершиной

arr\_list3[anti\_top[i]].erase(itg);

if (c == 0) { // не соединен с 1-ой вершиной

arr\_list3[anti\_top[i]].push\_back(top[0]);

sort(arr\_list3[anti\_top[i]].begin(), arr\_list3[anti\_top[i]].end());

}

}

}

arr\_list3.erase(arr\_list3.begin() + top[1]); // удаляем 2 склеиваемую вершину

for (int i = 0; i != arr\_list3.size(); i++) { //копируем arr\_list1s

for (int j = 0; j != arr\_list3[i].size(); j++) {

if (arr\_list3[i][j] > top[1]) arr\_list3[i][j]--;

}

}

for (int i = 0; i != 4; i++) {

cout << "Вершина " << i << " ";

for (int j = 0; j < arr\_list3[i].size(); j++) {

cout << arr\_list3[i][j] << ' ';

}

cout << endl;

}

cout << endl << endl << endl << endl;

// задание 2.2 стягивания ребра

int top2[2]; // вводим вершины

for (;;) {

cout << endl << endl << "Выберите 2 вершины: ";

cin >> top2[0];

cin >> top2[1];

if (top2[0] > top2[1]) {

int tmp = top2[0];

top2[0] = top2[1];

top2[1] = tmp;

}

if (top2[0] == top2[1] || !cin.good()) {

cout << "Вы ввели неправильное значение" << endl;

cin.clear();

cin.ignore(10, '\n');

continue;

}

if (!(arr1[top2[0]][top2[1]] && arr1[top2[1]][top2[0]])) {

cout << "Данные вершины не образуют ребро" << endl;

continue;

}

break;

}

int anti\_top2[3]; // тут остаются не введеные ввершины

for (int i = 0, j = 0; i != 5; i++, j++) {

if (i != top2[0] && i != top2[1]) anti\_top2[j] = i;

else j--;

}

int arr4[4][4]{ 0 }; // результат отождествления

for (int i = 0; i != 2; i++) {

for (int j = 0, x = 0; j != 5; j++, x++) {

// результат соединения двух вершин записывается в наименьший из двух вершин

if (top2[0] == j) continue;

if (top2[1] == j && j != 4) j++; // игнорируем 2 вершину т.к. она удаляется

if (top2[1] == 4 && j == 4) x--;

if (arr1[top2[i]][j]) {

arr4[top2[0]][x] = 1; // заполняем слияние двух вершин

arr4[x][top2[0]] = 1; // остальные вершины если пересекались хотя бы с одним из 2 сливаемых вершин

}

}

}

arr4[top2[0]][top2[0]] = 0;

for (int i = 0; i != 3; i++) {

for (int j = 0, y = 0; j != 3; j++, y++) {

// цикл проходит по оставшимся вершинам

if (arr1[anti\_top2[i]][anti\_top2[j]]) {

if (anti\_top2[j] > top2[1] && anti\_top2[i] > top2[1]) {

arr4[anti\_top2[i] - 1][anti\_top2[j] - 1] = 1;

arr4[anti\_top2[j] - 1][anti\_top2[i] - 1] = 1;

}

else if (anti\_top2[j] > top2[1]) {

arr4[anti\_top2[i]][anti\_top2[j] - 1] = 1;

arr4[anti\_top2[j] - 1][anti\_top2[i]] = 1;

}

else if (anti\_top2[i] > top2[1]) {

arr4[anti\_top2[i] - 1][anti\_top2[j]] = 1;

arr4[anti\_top2[j]][anti\_top2[i] - 1] = 1;

}

else {

arr4[anti\_top2[i]][anti\_top2[j]] = 1;

arr4[anti\_top2[j]][anti\_top2[i]] = 1;

}

}

}

}

cout << endl;

for (int i = 0; i != 4; i++) {

Print1(arr4[i], 4);

}

cout << endl;

// стягивания ребра для списков смежности

vector< vector<int> > arr\_list4;

arr\_list4.resize(5);

for (int i = 0; i != arr\_list1.size(); i++) {

for (int j = 0; j != arr\_list1[i].size(); j++) {

arr\_list4[i].push\_back(arr\_list1[i][j]);

}

}

itg = find(arr\_list4[top2[0]].begin(), arr\_list4[top2[0]].end(), top2[1]);

if (itg != arr\_list4[top2[0]].end()) {

arr\_list4[top2[0]].erase(itg, itg + 1);

}

for (vector<int>::iterator i = arr\_list4[top2[1]].begin(); i != arr\_list4[top2[1]].end(); i++) {

if (\*i == top[0]) continue;

itg = find(arr\_list4[top2[0]].begin(), arr\_list4[top2[0]].end(), \*i);

if (itg != arr\_list4[top2[0]].end() && \*itg == \*i) continue; // уже есть

arr\_list4[top2[0]].push\_back(\*i);

}

sort(arr\_list4[top2[0]].begin(), arr\_list4[top2[0]].end()); // сортируем вершины по порядку

for (int i = 0; i < 3; i++) {

itg = find(arr\_list4[anti\_top2[i]].begin(), arr\_list4[anti\_top2[i]].end(), top2[1]);

int c = count(arr\_list4[anti\_top2[i]].begin(), arr\_list4[anti\_top2[i]].end(), top2[0]);

if (itg != arr\_list4[anti\_top2[i]].end()) { // соединен со 2-ой вершиной

arr\_list4[anti\_top2[i]].erase(itg);

if (c == 0) { // не соединен с 1-ой вершиной

arr\_list4[anti\_top2[i]].push\_back(top2[0]);

sort(arr\_list4[anti\_top2[i]].begin(), arr\_list4[anti\_top2[i]].end());

}

}

}

arr\_list4.erase(arr\_list4.begin() + top2[1]); // удаляем 2-ю склеиваемую вершину

for (int i = 0; i != arr\_list4.size(); i++) { //копируем arr\_list1s

for (int j = 0; j != arr\_list4[i].size(); j++) {

if (arr\_list4[i][j] > top[1]) arr\_list4[i][j]--;

}

}

for (int i = 0; i != 4; i++) {

cout << "Вершина " << i << " ";

for (int j = 0; j < arr\_list4[i].size(); j++) {

cout << arr\_list4[i][j] << ' ';

}

cout << endl;

}

cout << endl << endl;

// задание 2.3 расщепления вершины

int arr5[6][6]{ 0 }; // новый массив с расщеплением

int top3;

for (;;) {

cout << endl << endl << "Выберите 1 вершину: ";

cin >> top3;

if (top3 > 4 || top3 < 0 || !cin.good()) {

cout << "Вы ввели неправильное значение" << endl;

cin.clear();

cin.ignore(10, '\n');

continue;

}

break;

}

for (int i = 0; i != 5; i++) {

for (int j = 0; j != 5; j++) {

if (!i) {

arr5[0][j + 1] = arr1[top3][j];

arr5[j + 1][0] = arr1[top3][j];

}

if (arr1[i][j]) arr5[i + 1][j + 1] = 1;

}

}

arr5[0][top3 + 1] = 1;

arr5[top3 + 1][0] = 1;

cout << endl;

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

Print1(arr5[i], 6);

}

cout << endl;

// расщепления вершины смежные списки

vector< vector<int> > arr\_list5;

arr\_list5.resize(5);

for (int i = 0; i != arr\_list1.size(); i++) {

for (int j = 0; j != arr\_list1[i].size(); j++) {

arr\_list5[i].push\_back(arr\_list1[i][j]);

}

}

vector<int> q;

arr\_list5.push\_back(q);

for (int i = 0; i != arr\_list1[top3].size(); i++) {

arr\_list5[arr\_list1.size()].push\_back(arr\_list1[top3][i]);

}

arr\_list5[arr\_list1.size()].push\_back(top3);

for (int i = 0; i != arr\_list5[top3].size(); i++) {

arr\_list5[arr\_list5[top3][i]].push\_back(arr\_list5.size() - 1);

}

arr\_list5[top3].push\_back(arr\_list5.size() - 1);

sort(arr\_list5[arr\_list1.size()].begin(), arr\_list5[arr\_list1.size()].end());

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

cout << "Вершина " << i << " ";

for (int j = 0; j < arr\_list5[i].size(); j++) {

cout << arr\_list5[i][j] << ' ';

}

cout << endl;

}

cout << endl << endl;

// задание 3.1 объединения графов

int arr6[5][5]{ 0 };

for (int i = 0; i != 5; i++) {

for (int j = 0; j != 5; j++) {

if (arr1[i][j] || arr2[i][j]) arr6[i][j] = 1;

}

}

cout << endl << "Объединение графов" << endl << endl;

for (int i = 0; i != 5; i++) {

Print1(arr6[i], 5);

}

cout << endl;

// задание 3.2 пересечения графов

int arr7[5][5]{ 0 };

for (int i = 0; i != 5; i++) {

for (int j = i + 1; j != 5; j++) {

if (arr1[i][j] && arr2[i][j]) {

arr7[i][j] = 1;

arr7[j][i] = 1;

}

}

}

cout << endl << "Пересечения графов" << endl << endl;

for (int i = 0; i != 5; i++) {

Print1(arr7[i], 5);

}

cout << endl;

// задание 3.3 кольцевая сумма графов

stack<int> Stack1;

stack<int> Stack2;

Stack1.push(0);

Stack2.push(0);

int nodes1[5]{ 0 }; // состояние вершин

int nodes2[5]{ 0 }; // состояние вершин

while (!Stack1.empty() && !Stack2.empty()) {

int node1 = Stack1.top();

int node2 = Stack2.top();

Stack1.pop();

Stack2.pop();

if (nodes1[node1] == 2 && nodes2[node2]) continue; // уже были

nodes1[node1] = 2;

nodes2[node2] = 2;

for (int i = 0; i != 5; i++) {

if (arr1[node1][i] && nodes1[i] != 2) {

Stack1.push(i);

nodes1[i] = 1;

}

if (arr2[node2][i] && nodes2[i] != 2) {

Stack2.push(i);

nodes2[i] = 1;

}

}

}

int n = 0;

for (int i = 0; i != 5; i++) {

if (!nodes1[i] || !nodes2[i]) {

n++;

}

}

if (n > 0) {

printf("Один из графов имеет изолированн%s вершин%c \n", n == 1 ? "ую" : "ые", n == 1 ? 'у' : 'ы');

cout << "Конец программы" << endl;

return 0;

}

int arr8[5][5]{ 0 };

for (int i = 0; i != 5; i++) {

for (int j = i + 1; j != 5; j++) {

if (arr1[i][j] != arr2[i][j]) {

arr8[i][j] = 1;

arr8[j][i] = 1;

}

}

}

cout << endl << "Кольцевая сумма графов" << endl << endl;

for (int i = 0; i != 5; i++) {

Print1(arr8[i], 5);

}

cout << 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 простые структуры данных, где повторили приобретенные навыки с предыдущего курса.