Министерство высшего образования и науки Российской Федерации

Пензенский Государственный Университет

Кафедра «Вычислительная техника»

Выполнили:

студенты группы 20ВВ2

Аверочкин П.С.

Зиновьев Я.М.

Принял:

к. т. н. доцент Юрова О.В.

д. т. н., профессор Митрохин М.А.

Пенза 2020

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Л и ОА в ИС»

на тему «Обход графа в глубину»

Ход работы:

Программа задает динамический двумерный массив, размерность которого задается с клавиатуры пользователем. Он в свою очередь описывает собой граф. Каждая единичка в массиве обозначает ребро между двумя смежными вершинами. С помощью функции rand() программа генерирует значения массива, состоящие из 0 или 1. Обязательным условием является наличие нулей по диагонали, что означает невозможность вершин быть смежными с самими с собой, т. е. отсутствие петель. Массив выводится на экран. Функция DFS является основной рабочей частью программы. Она совершает сам обход и счет пройденных вершин в графе.

Суть обхода: DFS работает рекурсивно, т. е. ссылается сама на себя по новым координатам. При обходе определенной вершины она ставит 1 в дополнительном массиве, отмечая тем самым, что она побывала здесь и дальнейший обход этой точки невозможен. В результате выводится номера вершин по очередности обхода программой.

Результат работы программы:



Рис 1 – Результат работы программы

Листинг:

#include <iostream>

#include <ctime>

#include <malloc.h>

#include <conio.h>

using namespace std;

int\* vis = NULL, n, \*\* p = NULL;

void DFS(int num) {

cout << num+1 << " ";

vis[num] = 1;

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

if (p[num][i] == 0 && vis[i] == 0) {

DFS(i);

}

}

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(0)); //по-настоящему случайная генерация массива

cout << "Введите размер матрицы: ";

cin >> n;

cout << "Массив: \n";

vis = (int\*)calloc(n, sizeof(int));

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

{

vis[i] = 0;

}

p = (int\*\*)calloc(n, sizeof(int));

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

{

p[i] = (int\*)calloc(n, sizeof(int));

}

cout.precision(3 \* n);

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

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

p[i][m] = 0 + rand() % 2;

p[i][i] = 0;

cout << p[i][m] << " ";

}

cout << endl;

}

cout << endl<< "Обход: ";

DFS(0);

return 0;

}

Вывод:

Мы научились реализовывать алгоритм обхода графа в глубину.