Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Обход графа в глубину»

Выполнили студенты группы 20ВВ3:

Шадрин Данила

Кравцов Михаил

Ляушин Ринат

Приняли:

Митрохин М.А.

Юрова О.В.

Пенза 2021

**Лабораторное задание**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу

смежности для неориентированного графа G. Выведите матрицу на экран.

2. Для сгенерированного графа осуществите процедуру обхода в

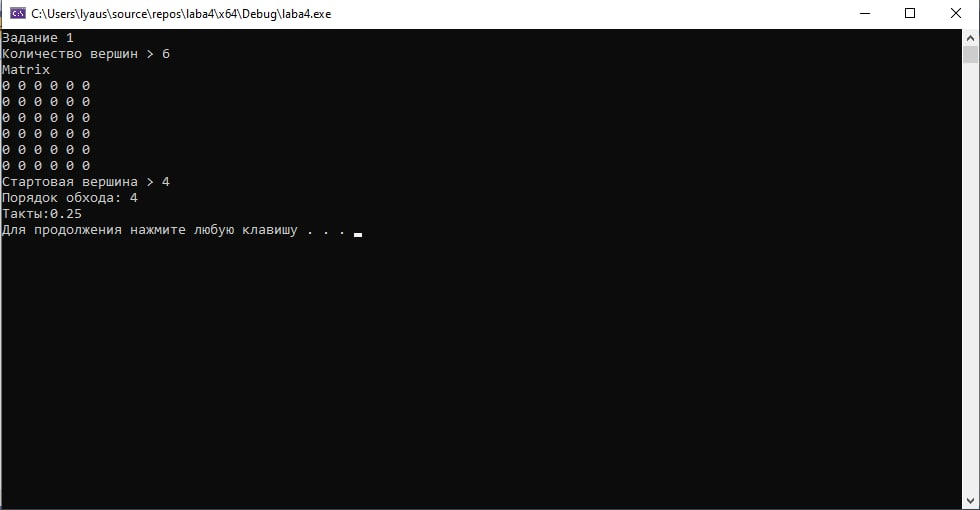
глубину, реализованную в соответствии с приведенным выше описанием.

**Ход работы:**

1. Код программы

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS |
|  | #include <iostream> |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <locale.h> |
|  | #include <conio.h> |
|  | #include <time.h> |
|  | #include <stack> |
|  | #include <vector> |
|  | using namespace std; |
|  | void dfs\_spis(int, const vector<vector<int>>&); |
|  | vector<bool> vis(0); |
|  |  |
|  | void DFS\_f(int\*\* graph, bool\* visited, int st, int n) |
|  | { |
|  | int r; |
|  | cout << st + 1 << " "; |
|  | visited[st] = true; |
|  | for (r = 0; r <= n; r++) |
|  | if ((graph[st][r] != 0) && (!visited[r])) |
|  | DFS\_f(graph, visited, r, n); |
|  | } |
|  |  |
|  | void Create\_matrix() { |
|  | int start, n, t = clock(); |
|  |  |
|  | cout << "Çàäàíèå 1" << endl; |
|  |  |
|  | cout << "Êîëè÷åñòâî âåðøèí > "; cin >> n; |
|  |  |
|  | bool\* visited = new bool[n]; |
|  |  |
|  | int\*\* matrix; |
|  | matrix = (int\*\*)malloc(n \* sizeof(int\*)); |
|  | for (int i = 0; i < n; ++i) { |
|  | //matrix[i] = new int[n]; |
|  | visited[i] = false; |
|  | } |
|  |  |
|  | int s = 1; |
|  | srand(time(NULL)); |
|  | for (int i = 0; i < n; i++) // ñîçäàíèå ìàòðèöû |
|  | { |
|  | matrix[i] = (int\*)malloc(n \* sizeof(int)); |
|  | matrix[i][i] = 0; |
|  | for (int j = s; j < n; j++) |
|  | { |
|  | if (rand() % 100 > 10) |
|  | { |
|  | matrix[i][j] = 0; |
|  | } |
|  | else { |
|  | matrix[i][j] = 1; |
|  | } |
|  | } |
|  | s++; |
|  | } |
|  | s = 1; |
|  | for (int i = 0; i < n; i++) // íåîðèåíòèðîâàííûé ãðàô |
|  | { |
|  | for (int j = s; j < n; j++) |
|  | matrix[j][i] = matrix[i][j]; |
|  | s++; |
|  | } |
|  | cout << "Matrix"; // âûâîä ìàòðèöû íà ýêðàí |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | cout << matrix[i][j] << " "; |
|  | } |
|  | } |
|  | cout << endl; |
|  | cout << "Ñòàðòîâàÿ âåðøèíà > "; cin >> start; |
|  | cout << "Ïîðÿäîê îáõîäà: "; |
|  | DFS\_f(matrix, visited, start - 1, n); |
|  |  |
|  | delete[]visited; |
|  |  |
|  | //DFS\_stack(matrix, visited, start - 1, n); |
|  |  |
|  | cout << "\nÒàêòû:" << ((float)t / CLOCKS\_PER\_SEC) << "\n"; |
|  | system("pause"); |
|  | } |
|  |  |
|  | void sp\_smej() { |
|  | int n, m; |
|  | cout << "Ñïèñêè ñìåæíîñòè" << endl; |
|  | cin >> n >> m; // ñòðîêè è ñòîëáöû |
|  |  |
|  | //èíèöèëèçèðóåì ñïèñîê ñìåæíîñòè è ìàññèâ ïîñåùåííûõ âåðøèí |
|  | vector<vector<int>> adj\_list(n, vector<int>()); |
|  | vis.resize(n, false); |
|  |  |
|  | //ââîä ðåáåð |
|  | for (int i = 0; i < m; i++) { |
|  | int x, y; |
|  | cin >> x >> y; |
|  | adj\_list[x - 1].push\_back(y - 1); |
|  | adj\_list[y - 1].push\_back(x - 1); |
|  | } |
|  | cout << endl; |
|  |  |
|  | dfs\_spis(0, adj\_list); //ïîèñê â ãëóáèíó |
|  |  |
|  | } |
|  |  |
|  | void dfs\_spis(int u, const vector<vector<int>>& matrix\_adjacency) { |
|  | vis[u] = true; |
|  | cout << "in " << u + 1 << endl; |
|  | for (auto i : matrix\_adjacency[u]) |
|  | if (!vis[i]) dfs\_spis(i, matrix\_adjacency); |
|  | cout << "out " << u + 1 << endl; |
|  | } |
|  |  |
|  | /\*void vector\_matrix() { |
|  |  |
|  | int n; |
|  |  |
|  | vector<vector<int>> mainV; |
|  |  |
|  | cout << "Êîëè÷åñòâî âåðøèí > "; cin >> n; |
|  | cout << endl; |
|  |  |
|  | mainV.resize(n); |
|  | int s = 1; |
|  | for (int i = 0; i < n; i++) { |
|  | mainV[i].resize(n); |
|  | for (int j = s; j < n; j++) { |
|  | if ((rand() % 100) > 10) { |
|  | int temp = 0; |
|  | mainV[i][j] = temp; |
|  | } |
|  | else { |
|  | int temps = 1; |
|  | mainV[i][j] = temps; |
|  | } |
|  | } |
|  | s++; |
|  | } |
|  | s = 1; |
|  |  |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = s; j < n; j++) { |
|  | mainV[j][i] = mainV[i][j]; |
|  | } |
|  | s++; |
|  | } |
|  |  |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = 0; j < n; j++) { |
|  | cout << mainV[i][j] << " "; |
|  | } |
|  | cout << endl; |
|  | } |
|  | }\*/ |
|  |  |
|  | /\*void DFS\_stack(int\*\* graph, bool\* visited, int st, int n) |
|  | { |
|  | stack<int> S; |
|  | vector<int> v(n); |
|  | int t; |
|  | S.push(n); |
|  | v[n]++; |
|  | cout << n + 1 << " "; |
|  | while (!S.empty()) |
|  | { |
|  | t = S.top(); |
|  | S.pop(); |
|  | for (int i = n - 1; i > 0; i--) |
|  | if (graph[t][i] && !v[i]) |
|  | { |
|  | S.push(i); |
|  | v[i]++; |
|  | cout << i + 1 << " "; |
|  | } |
|  | } |
|  | }\*/ |
|  |  |
|  | void main() |
|  | { |
|  | system("chcp 1251"); system("cls"); |
|  | Create\_matrix(); |
|  | //vector\_matrix(); |
|  | sp\_smej(); |
|  | } |

1. Результат работы программы.



Вывод: в ходе выполнения данной лабораторной работы разработали программу, согласно заданию.