Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №7

по курсу “Логика и основа алгоритмизации в ИЗ”

на тему “Обход графа в глубину”



Пенза 2022

**Название**

Определение характеристик графов.

**Цель работы:** выполнить лабораторные указания 1-2 выполняя обход графа в глубину.

**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.

3. Реализуйте процедуру обхода в глубину для графа, представленного списками смежности.

**Задание 2**

1. Для матричной формы представления графов выполните преобразование рекурсивной реализации обхода графа к не рекурсивной.

**Задание 1:**

**Листинг**

**#include <locale>**

**#include <iostream>**

**#include <algorithm>**

**#include <stack>**

**using namespace std;**

**void algorithm(int\*\* G, int\* visited, int size, int n, int space);**

**void algorithm2(int\*\* G, int\* visited, int n, int space);**

**void DFS(int\*\* G, int size, int variant, int n = 0) {**

**if (n < size) cout << "Вершина = " << n << endl;**

**else return;**

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

**for (int i = 0; i != size; i++) visited[i] = 0;**

**if (variant <= 0) algorithm(G, visited, size, n, 1);**

**else algorithm2(G, visited, n, 1);**

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

**if (n >= size) return;**

**else DFS(G, size, variant, ++n);**

**}**

**void algorithm(int\*\* G, int\* visited, int size, int n, int space) {**

**if (visited[n] == 1) return;**

**visited[n] = 1;**

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

**if (G[n][i] == 0 || visited[i] == 1) continue;**

**for (int j = 0; j != space; j++) cout << "\_\_\_\_";**

**cout << i << endl;**

**algorithm(G, visited, size, i, space + 1);**

**}**

**}**

**void algorithm2(int\*\* G, int\* visited, int n, int space) {**

**if (visited[n] == 1) return;**

**visited[n] = 1;**

**for (int i = 1; i < G[n][0]; i++) {**

**if (visited[G[n][i]] == 1) continue;**

**for (int j = 0; j < space; j++) cout << "\_\_\_\_";**

**cout << G[n][i] << endl;**

**algorithm2(G, visited, G[n][i], space + 1);**

**}**

**}**

**int main()**

**{**

**setlocale(LC\_ALL, "rus");**

**srand(static\_cast<unsigned int>(time(NULL)));**

**int count;**

**cout << "Введитк кол-во вершин: ";**

**cin >> count;**

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

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

**arr[i] = new int[count];**

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

**arr[i][j] = 0;**

**}**

**}**

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

**for (int j = i + 1; j < count; j++) {**

**int r = rand() % 101;**

**if (r > 40) {**

**arr[i][j] = 1;**

**arr[j][i] = 1;**

**}**

**else {**

**arr[i][j] = 0;**

**arr[j][i] = 0;**

**}**

**arr[i][i] = 0;**

**}**

**}**

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

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

**cout << arr[i][j] << ' ';**

**}**

**cout << endl;**

**}**

**cout << endl;**

**DFS(arr, count, 0);**

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

**//Реализуйте процедуру обхода в глубину для графа, представленного списками смежности.**

**int\*\* arr2 = new int\* [count];**

**cout << "Список смежности" << endl;**

**for (int i = 0; i < count; i++) { //создаем список смежности**

**int x = 0;**

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

**if (arr[i][j] == 1) x++;**

**}**

**arr2[i] = new int[x + 1];**

**arr2[i][0] = x + 1; //записываем ее размер**

**cout << i << " ";**

**for (int j = 0, h = 1; j < count; j++) {**

**if (arr[i][j] == 1) {**

**arr2[i][h] = j;**

**h++;**

**cout << j;**

**}**

**}**

**cout << endl;**

**}**

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

**DFS(arr2, count, 2);**

**// сделать обход в глубину не рекурсивно**

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

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

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

**cout << arr[i][j] << ' ';**

**}**

**cout << endl;**

**}**

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

**cout << endl;**

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

**cout << "Вершина " << i << endl;**

**for (int i = 0; i != count; i++) visited[i] = 0;**

**stack<int> Stack;**

**Stack.push(i);**

**while (!Stack.empty()) {**

**int node = Stack.top();**

**Stack.pop();**

**if (visited[node] == 1) continue;**

**visited[node] = 1;**

**for (int i = count - 1; i >= 0; i--) {**

**if (arr[node][i] == 0 || visited[i] == 1) continue;**

**Stack.push(i);**

**}**

**cout << node << " ";**

**}**

**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 <locale>**

**#include <iostream>**

**#include <algorithm>**

**#include <stack>**

**using namespace std;**

**void algorithm(int\*\* G, int\* visited, int size, int n, int space);**

**void algorithm2(int\*\* G, int\* visited, int n, int space);**

**void DFS(int\*\* G, int size, int variant, int n = 0) {**

**if (n < size) cout << "Вершина = " << n << endl;**

**else return;**

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

**for (int i = 0; i != size; i++) visited[i] = 0;**

**if (variant <= 0) algorithm(G, visited, size, n, 1);**

**else algorithm2(G, visited, n, 1);**

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

**if (n >= size) return;**

**else DFS(G, size, variant, ++n);**

**}**

**void algorithm(int\*\* G, int\* visited, int size, int n, int space) {**

**if (visited[n] == 1) return;**

**visited[n] = 1;**

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

**if (G[n][i] == 0 || visited[i] == 1) continue;**

**for (int j = 0; j != space; j++) cout << "\_\_\_\_";**

**cout << i << endl;**

**algorithm(G, visited, size, i, space + 1);**

**}**

**}**

**void algorithm2(int\*\* G, int\* visited, int n, int space) {**

**if (visited[n] == 1) return;**

**visited[n] = 1;**

**for (int i = 1; i < G[n][0]; i++) {**

**if (visited[G[n][i]] == 1) continue;**

**for (int j = 0; j < space; j++) cout << "\_\_\_\_";**

**cout << G[n][i] << endl;**

**algorithm2(G, visited, G[n][i], space + 1);**

**}**

**}**

**int main()**

**{**

**setlocale(LC\_ALL, "rus");**

**srand(static\_cast<unsigned int>(time(NULL)));**

**int count;**

**cout << "Введитк кол-во вершин: ";**

**cin >> count;**

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

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

**arr[i] = new int[count];**

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

**arr[i][j] = 0;**

**}**

**}**

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

**for (int j = i + 1; j < count; j++) {**

**int r = rand() % 101;**

**if (r > 40) {**

**arr[i][j] = 1;**

**arr[j][i] = 1;**

**}**

**else {**

**arr[i][j] = 0;**

**arr[j][i] = 0;**

**}**

**arr[i][i] = 0;**

**}**

**}**

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

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

**cout << arr[i][j] << ' ';**

**}**

**cout << endl;**

**}**

**cout << endl;**

**DFS(arr, count, 0);**

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

**//Реализуйте процедуру обхода в глубину для графа, представленного списками смежности.**

**int\*\* arr2 = new int\* [count];**

**cout << "Список смежности" << endl;**

**for (int i = 0; i < count; i++) { //создаем список смежности**

**int x = 0;**

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

**if (arr[i][j] == 1) x++;**

**}**

**arr2[i] = new int[x + 1];**

**arr2[i][0] = x + 1; //записываем ее размер**

**cout << i << " ";**

**for (int j = 0, h = 1; j < count; j++) {**

**if (arr[i][j] == 1) {**

**arr2[i][h] = j;**

**h++;**

**cout << j;**

**}**

**}**

**cout << endl;**

**}**

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

**DFS(arr2, count, 2);**

**// сделать обход в глубину не рекурсивно**

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

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

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

**cout << arr[i][j] << ' ';**

**}**

**cout << endl;**

**}**

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

**cout << endl;**

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

**cout << "Вершина " << i << endl;**

**for (int i = 0; i != count; i++) visited[i] = 0;**

**stack<int> Stack;**

**Stack.push(i);**

**while (!Stack.empty()) {**

**int node = Stack.top();**

**Stack.pop();**

**if (visited[node] == 1) continue;**

**visited[node] = 1;**

**for (int i = count - 1; i >= 0; i--) {**

**if (arr[node][i] == 0 || visited[i] == 1) continue;**

**Stack.push(i);**

**}**

**cout << node << " ";**

**}**

**cout << endl;**

**}**

**return 0;**

**}**

**Результаты работы программы:**

Результаты работы программы показан на рисунке 1.

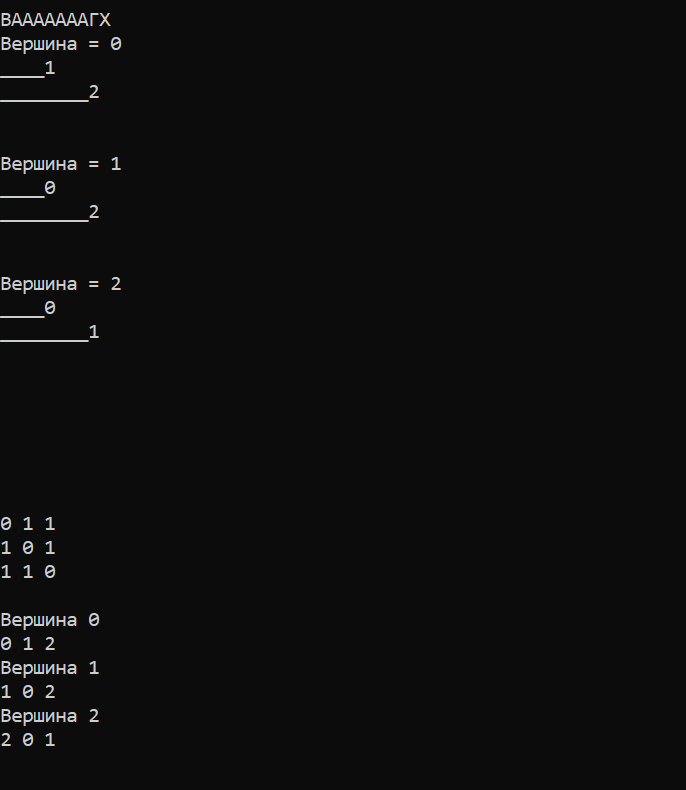
****

Рис. 1

Результат получился правильный, все действия совершены верно.

Результат работы программы, показанный на рисунке 1.