Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №7

по курсу «Логика и основы алгоритмизации в инженерных задачах»

## на тему «Обход графа в глубину»

**Выполнил студент группы 21вв1:**

Вартанов А.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

**Задание 1:**

1. Сгенерируйте матрицу смежности для неориентированного графа G. Выведите матрицу на экран

2. Для сгенерированного графа осуществите процедуру обхода в глубину, реализованную в соответствии с приведённым выше описанием.

**Задание 2:\***

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

**Листинг:**

#include <stack>

#include <string>

#include <stdlib.h>

#include <stdio.h>

#include <limits>

#include <time.h>

using namespace std;

int i, j,size;

bool \*visited=new bool[200];

//матрица смежности

int \*\*graph;

int\*\* create\_mass(int size){

int \*\*graph = (int\*\*)malloc(sizeof(int\*)\*size);

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

graph[i] = (int\*)malloc(sizeof(int)\*size);

}

return graph;

}

int\*\* initializate(int\*\* graph, int size){

srand(time(NULL));

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

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

graph[i][j] = rand()%2;

}

}

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

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

if(i == j){

graph[i][j] = 0;

}

if(graph[i][j] = graph[j][i]){

graph[i][j] = graph[j][i];

}

}

}

return graph;

}

//поиск в глубину

void DFS(int st)

{

int r;

printf("%d",st+1);

visited[st]=true;

for (r=0; r<=size; r++)

if ((graph[st][r]!=0) && (!visited[r]))

DFS(r);

}

void main()

{

setlocale(LC\_ALL, "Rus");

int start = 1;

printf("\nВведите количество вершин (больше 2):");

scanf("%d", &size);

graph = create\_mass(size);

initializate(graph,size);

printf("\nМатрица смежности графа:");

for (i=0; i<size; i++){

printf("\n");

visited[i]=false;

for (j=0; j<size; j++){

printf("%d ",graph[i][j]);

}

}

bool \*vis=new bool[size];

printf("\nПорядок обхода рекурсивной функции:");

DFS(start-1);

delete []visited;

printf("\nПорядок обхода нерекурсивной функции:");

stack <int> Stack;

int nodes[200];

for (int i = 0; i < size; i++) nodes[i] = 0;

Stack.push(0);

while (!Stack.empty()) // пока стек не пуст

{

int node = Stack.top();

Stack.pop();

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

nodes[node] = 2;

for (int j = size - 1; j >= 0; j--)

{

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

{

Stack.push(j);

nodes[j] = 1;

}

}

printf("%d",node + 1);

}

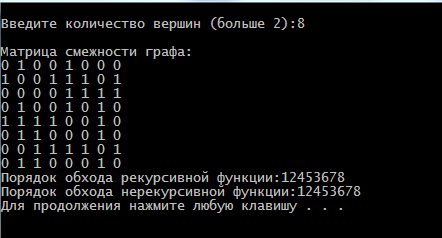
printf("\n");

system("pause");

}

**Результат работы программы:**

Задание 1.1, 1.2 и 2.1:



Программа выполняет «поиск в глубину» с помощью рекурсивной функции и не рекурсивной и полностью выполняет свою поставленную задачу, выводя верный результат.

**Вывод:**

Я научился создавать функцию поиска в глубину, реализовал преобразование рекурсивной функции обхода графа к не рекурсивной.