Пензенский государственный университет Кафедра «Вычислительная техника»

**Отчет**о выполнении лабораторной работы №4.1 по дисциплине “Логика и основа алгоритмизации в инженерных задачах” на тему**:**

**Обход графа в глубину**

Выполнили студенты гр. 19ВВ4:   
Хлыстов А.Ю.  
Привалов А.Э.

Проверили:

Юрова О.В.  
Митрохин М. А.

Пенза, 2020 г.

**Название**Обход графа в глубину

**Цель работы**Реализация алгоритма обхода графа в глубину.

**Лабораторное задание**1. Сгенерируйте (используя генератор случайных чисел) две матрицу

смежности для неориентированного графа G. Выведите сгенерированные

матрицы на экран.

2. Для сгенерированного графа осуществите процедуру обхода в

глубину, реализованную в соответствии с приведенным выше описанием.

Задание 2

1. Для матричной формы представления графов выполните

преобразование рекурсивной реализации обхода графа к не рекурсивной.

**Код программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <conio.h>

#include <iostream>

#include <locale.h>

// GLOBAL

int\* NUM;

int\* history;

int flag = 0;

struct graph {

int\*\* matrix;

int\* names;

int size;

}m;

int\*\* createMatrix(int size) {

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

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

matrix[i] = (int\*)calloc(size, sizeof(int));

}

return matrix;

}

void fillMatrix(graph m) {

int value;

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

for (int j = i + 1; j < m.size; j++) {

value = rand() % 10;

if (value < 3)

value = 0;

else

value = 1;

m.matrix[i][j] = value;

m.matrix[j][i] = m.matrix[i][j];

}

}

}

void outputMatrix(graph m) {

printf\_s(" ");

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

printf\_s("%d ", i);

}

printf\_s("\n");

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

printf\_s("%d ", i);

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

printf\_s("%d ", m.matrix[i][j]);

}

printf\_s("\n");

}

}

int null\_exist(int\* NUM, int size) {

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

if (NUM[i] != 1) {

return i;

}

}

return -1;

}

int\* generate\_list(int size, int value) {

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

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

NUM[i] = value;

}

return NUM;

}

void DFS(int v) {

NUM[v] = 1;

printf("%d ", v);

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

if (m.matrix[v][i] == 1 && NUM[i] == 0) {

DFS(i);

}

}

}

int DFS\_no\_recursion(int v) {

int i = 0;

history[i] = v;

NUM[v] = 1;

if (flag == 0) {

printf("%d ", v);

flag = -1;

}

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

if (m.matrix[v][j] == 1 && NUM[j] == 0) {

i++;

history[i] = j;

printf("%d ", j);

return history[i];

}

}

history[i] = -1;

i--;

return history[i];

}

int main() {

setlocale(LC\_ALL, "Rus");

int v = 0;

m.size = 5;

m.matrix = createMatrix(m.size);

fillMatrix(m);

outputMatrix(m);

printf("\n");

NUM = generate\_list(m.size, 0);

printf("Рекурсивная реализация DFS:\n");

while (true) {

if ((v = null\_exist(NUM, m.size)) != -1) {

DFS(v);

printf("\n");

}

break;

}

///

printf("Нерекурсивная реализация DFS:\n");

history = generate\_list(m.size, -1);

NUM = generate\_list(m.size, 0);

int last;

while ((v = null\_exist(NUM, m.size)) != -1) {

last = v;

do {

last = DFS\_no\_recursion(last);

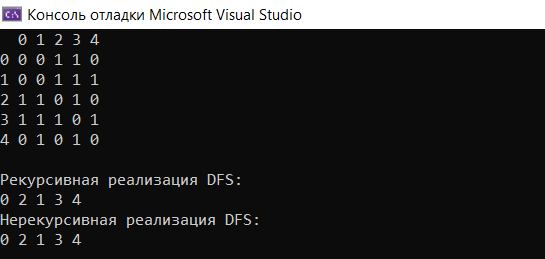
} while (history[0] != -1);

printf("\n");

}

}

**Результаты работы программы**



**Рисунок 1 – результат выполнения задания**

**Выводы**В процессе выполнения лабораторной работы был реализован алгоритм обхода массива в глубину.