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

Халилов Алмаз

Китаев Андрей

Новиков Иван

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пеза 2021

# **Название**

**Оценка времени выполнения программ**

**Цель работы** – изучение обхода графов в глубину.

# **Методические указания**

Обход графа – одна из наиболее распространенных операций с

графами. Задачей обхода является прохождение всех вершин в графе.

Обходы применяются для поиска информации, хранящейся в узлах графа,

нахождения связей между вершинами или группами вершин и т.д.

Одним из способов обхода графов является поиск в глубину. Идея

такого обхода состоит в том, чтобы начав обход из какой-либо вершины

всегда переходить по первой встречающейся в процессе обхода связи в

следующую вершину, пока существует такая возможность. Как только в

процессе обхода исчерпаются возможности прохода, необходимо вернуться

на один шаг назад и найти следующий вариант продвижения. Таким образом,

итерационно выполняя описанные операции, будут пройдены все доступные

для прохождения вершины. Чтобы не заходить повторно в уже пройденные

вершины, необходимо их пометить как пройденные.

Таким образом, можно предложить следующую рекурсивную

реализацию алгоритма обхода в глубину.

Вход: G – матрица смежности графа.

Выход: номера вершин в порядке их прохождения на экране.

Алгоритм ПОГ

1.1. для всех i положим NUM[i] = False пометим как не посещенную

1.2. ПОКА существует новая вершина v

1.3. ВЫПОЛНЯТЬ DFS (v).

Алгоритм DFS(v):

2.1. пометить v как посещенную NUM[v] = True;

2.2. вывести на экран v;

2.3. ДЛЯ i = 1 ДО size\_G ВЫПОЛНЯТЬ

2.4. ЕСЛИ G(v,i) = = 1И NUM[i] = = False

2.5. ТО

2.6. {

2.7. DFS(i);

2.8. }

Реализация состоит из подготовительной части, в которой все вершины

помечаются как не помеченные (п.1.1) и осуществляется запуск процедуры

обхода для вершин графа (п.1.2, 1.3). И непосредственно процедуры обхода,

которая помечает текущую (т.е. ту, в которой на текущей итерации находится

алгоритм) вершину как посещенную (п. 2.1). Затем выводит номер текущей

вершины на экран (п.2.2) и в цикле просматривает v-ю строку матрицы

смежности графа G(v,i). Как только алгоритм встречает смежную с v не

посещенную вершину (п.2.4), то для этой вершины вызывается процедура

обхода (п.2.7).

# **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

**Ход Работы**

**Листинг**

#define \_CRT\_NONSTDC\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include <string.h>

#include <windows.h>

int i = 0;

int j = 0;

int size = 0;

int\*\* g = 0;

void z1() {

srand(time(NULL));

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

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

{

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

}

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

{

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

{

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

}

}

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

{

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

{

if (i == j)

g[i][j] = 0;

else

g[i][j] = g[j][i];

}

}

printf("Матрица: \n");

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

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

printf("%2d", g[i][j]);

}

printf("\n");

}

}

void DFS(int\*\* g, int\* NUM, int v)

{

NUM[v] = 1;

printf(" %d ", v);

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

{

if ((g[v][i] == 1) && (NUM[i] == 0))

DFS(g, NUM, i);

}

}

void z2(int\*\* g)

{

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

int v;

printf("Точка входа: ");

scanf("%d", &v);

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

{

NUM[i] = 0;

}

printf("\n");

DFS(g, NUM, v);

}

int main(void)

{

setlocale(LC\_ALL, "RUS");

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

printf("Введите размер матрицы: ");

scanf("%d", &size);

z1();

printf("\n");

z2(g);

printf("\n");

system("pause");

}

**Задание 1:**

1. Сгенерировали (используя генератор случайных чисел) матрицу

смежности для неориентированного графа G и вывели матрицу на экран.

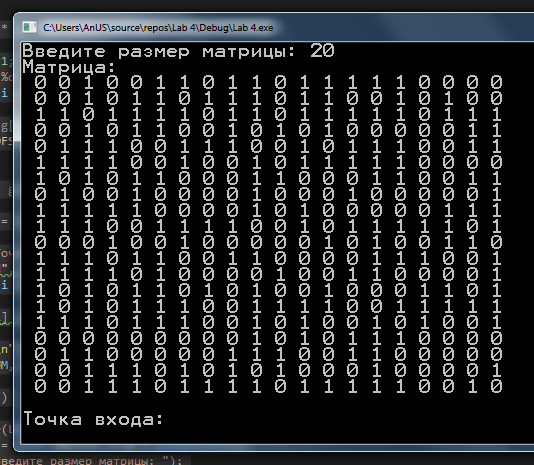


Рисунок 1. Генерация массива

2. Для сгенерированного графа осуществили процедуру обхода в

глубину, реализованную в соответствии с приведенным выше описанием.

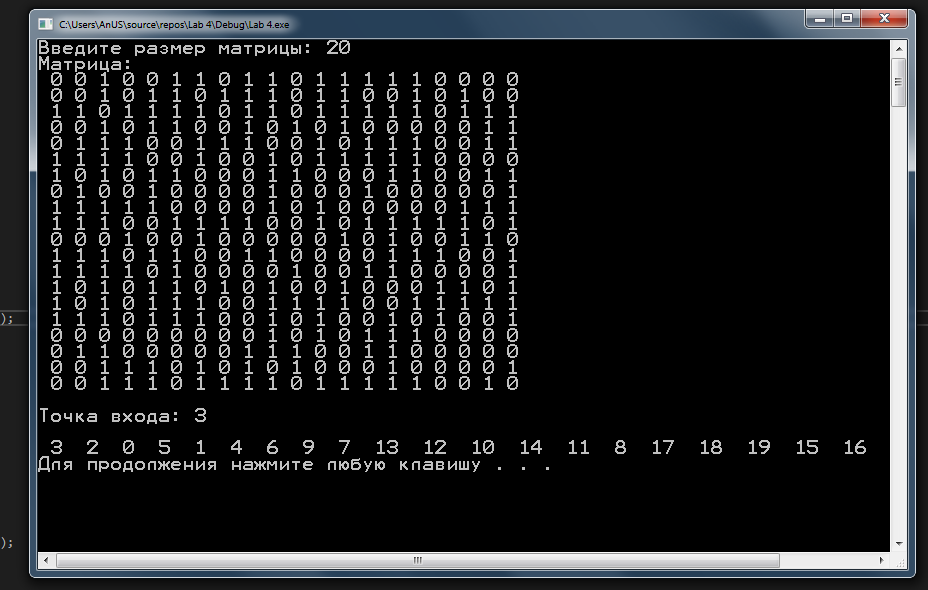


Рисунок 2. Ввод точки входа и результат выполнения программы.

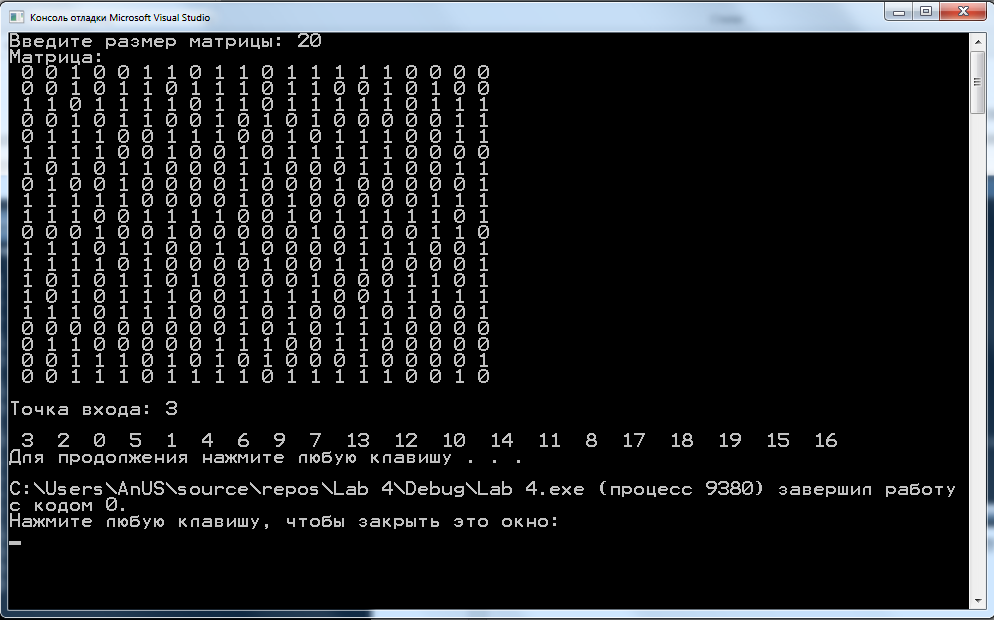


Рисунок 2. Завершение работы программы.

**Вывод:** изучили обход графа – одну из наиболее распространенных операций с графами. Написали и протестировали программу, которая создает граф определенного размера и обходит его.