Пензенский государственный университет  
Кафедра «Вычислительной техники»

**Отчет**по лабораторной работе №4  
по дисциплине: «Логика и основы алгоритмизации в инженерных задачах»  
на тему: «Обход графа в глубину»

**Выполнил студент группы 19ВВ1:**

Балалаев А. А.

**Приняли:**

д.т.н. профессор Митрохин М. А.

к.т.н. Юрова О. В.

Пенза 2020.

**Цель работы:** изучить обход в глубину, научится реализовывать его как через рекурсию, так и без неё, научится применять обход в глубину на списках.

**Теория**

Большое число задач, связанных с графами требует перебора вершин графа,

т.е. просмотра каждой вершины в точности один раз (задачи поиска).

**Обход в глубину** осуществляется с некоторой вершины вдоль ребер графа, до

попадания в лист. После этого нужно возвращаться назад вдоль пройденного пути, пока

не будет обнаружена вершина, у которой есть еще не посещенная смежная вершина, и

затем двигаться в направлении не посещённой вершины. Эти действия повторяются до

возврата в начальную вершину после посещения всех остальных.

Т.о. основная идея поиска в глубину – сначала полностью исследовать одну ветку вглубь

и только потом переходить к другим веткам.

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 &quot;не посещенную&quot;;

1.2. ПОКА существует &quot;новая&quot; вершина v

1.3. ВЫПОЛНЯТЬ DFS (v).

Алгоритм DFS(v):

2.1. пометить v как &quot;посещенную&quot; 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. }

**Стек** – это структура данных, в которой элементы поддерживают принцип LIFO (“Last in – first out”): пос

**PUSH** – кладёт элемент на вершину стека

**POP** – снимает элемент с вершины стека, перемещая вершину к следующему элементу

ледним зашёл – первым вышел. Или первым зашёл – последним вышел.

**Структура** — это объединение нескольких объектов, возможно, различного типа под одним именем, которое является типом структуры. В качестве объектов могут выступать переменные, массивы, указатели и другие структуры.  
  
Структуры позволяют трактовать группу связанных между собой объектов не как множество отдельных элементов, а как единое целое. Структура представляет собой сложный тип данных, составленный из простых типов.

**Односвязный список** – структура данных, в которой каждый элемент (узел) хранит информацию, а также ссылку на следующий элемент. Последний элемент списка ссылается на NULL.

**Код программы**

// la4.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#define \_CRT\_SECURE\_NO\_WARNINGS

#define N 5

#include <time.h>

#include <locale.h>

#include <iostream>

void rand\_Zap(int\*\* mat, int n)

{

srand(time(NULL));

printf("G\n");

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

for (int j = 0; j < n; j++)

{

if (i == j)

{

mat[i][j] = 0;

}

if (i < j) {

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

mat[j][i] = mat[i][j];

}

}

}

void print\_G(int\*\* mat, int n)

{

printf(" ");

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

printf("%3d", i + 1);

printf("\n\n");

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

{

printf("%2d", i + 1);

for (int j = 0; j < n; j++)

{

printf("%3d", mat[i][j]);

}

printf("\n");

}

printf("\n");

}

void DFS(int s, int\*\* G, int n, int\* vis)

{

vis[s] = 1;

printf("%d\t", s + 1);

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

{

if (G[s][i] == 1 && vis[i] == 0)

DFS(i, G, n, &vis[0]);

}

}

struct stack {

int elem[N];

int top;

};

typedef struct Spisok {

int value;

struct Spisok\* next;

}smej;

smej\* create(int num) {

// Выделение памяти под корень списка

smej\* tmp = new smej;

// Присваивание имя вершине

tmp->value = num;

// Присваивание указателю на следующий элемент значения NULL

tmp->next = NULL;

return(tmp);

}

void addElem(int data, smej\* head) {

// Выделение памяти под корень списка

smej\* tmp = new smej;

// Присваивание значения узлу

tmp->value = data;

// Присваивание указателю на следующий элемент значения NULL

tmp->next = NULL;

// Присваивание новому указателю указателя head.

// Присваивание выполняется для того, чтобы не потерять указатель на «голову» списка

smej\* p = head;

// Сдвиг указателя p в самый конец первоначального списка

while (p->next != NULL)

p = p->next;

// Присваивание указателю p -> next значения указателя tmp (созданный новый узел)

p->next = tmp;

}

void print(smej\* head) {

smej\* s = head;

if (s != NULL) { //номер вершины

printf("%d: ", s->value + 1);

s = s->next;

}

while (s != NULL) //номера узлов

{

// Вывод значения узла

printf("%d ", s->value + 1);

// Сдвиг указателя к следующему узлу

s = s->next;

}

}

void spisokSmej(int \*\*Matrix, smej\* G\_S[N]) {

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

Spisok\* tmp = create(i);

for (int j = 0; j < N; j++)

{

if (Matrix[i][j] == 1)

addElem(j, tmp);

}

G\_S[i] = tmp;

}

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

{

print(G\_S[i]);

printf("\n");

}

printf("\n");

}

void DFSsp(smej\* (&P)[N], int ver, int\* num) {

num[ver] = 1;

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

smej\* s = P[ver];

s = s->next;

while (s != NULL) {

if (num[s->value] == 0) {

DFSsp(P, s->value, &num[0]);

}

s = s->next;

}

}

int main()

{

setlocale(LC\_ALL, "Rus");

/\*int N = 1;

scanf("%d", &N);\*/

int\*\* G = (int\*\*)malloc(N \* sizeof(int\*));

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

G[i] = (int\*)malloc(N \* sizeof(int));

}

int\* vis, num[N], s;

vis = (int\*)malloc(N \* sizeof(int));

struct stack\* stk;

stk = new stack;

Spisok\* G\_S[N];

rand\_Zap(G, N);

print\_G(G, N);

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

{

vis[i] = 0;

}

printf("Введите вершину для начала обхода\n");

scanf("%d", &s);

while ((s < 1) || (s > N))

{

printf("Ошибка.\n");

scanf("%d", &s);

}

printf("\n");

s--;

printf("Результат:\n");

DFS(s, G, N, &vis[0]);

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

num[i] = 0;

}

printf("\n");

printf("Представление графа в виде списка смежности: \n");

spisokSmej(G, G\_S);

printf("Введите номер вершины для начала обхода: ");

scanf("%d", &s);

s--;

DFSsp(G\_S, s, &num[0]);

getchar();

getchar();

}

**Результат работы кода**

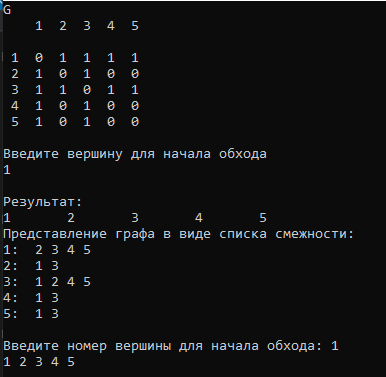


Рис.1 Результат работы программы

**Вывод:** изучил обход в глубину, научился его реализовывать.