Министерство науки и высшего образования Российской федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №7

по курсу «Логика и основы алгоритмизации в инженерных задачах»

## на тему «Обход графа в глубину»

Выполнили ст. группы 22ВВВ1:

Уткин М.М.

Саветкин Д.Д.

Соколовский Е.В

Приняли:

К.э.н., доцент Акифьев И. В.

К.т.н., доцент Юрова О. В.

Пенза 2023

**Цель работы:**

Цель научиться использовать алгоритм обхода 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.

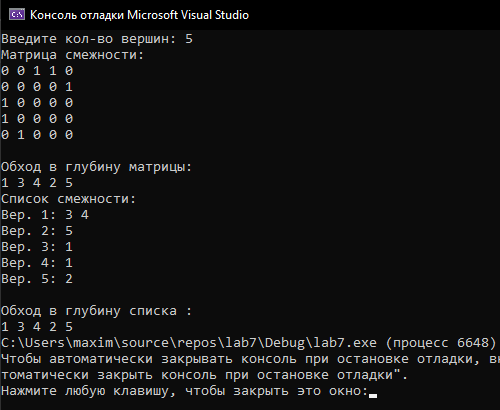
**Задание 2\***

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

**Ход работы:**

Создаем первый двумерный динамический массив. Обнуляем его диагональ. Заполняем верхнюю половину рандомно единицами и нулями. Отражаем на нижнюю часть. Считаем количество единиц в верхней части, для того чтобы определить размер графа. Проверяем количество единиц в строчке, чтобы определить какие вершины. Создает второй динамический массив. Строим его на основе первой матрицы. Так же выводим его размер и типы вершин.

**Результаты работы программы:**



**Вывод:**

В ходе лабораторной работы научился работать с обходом графа в глубину для матрицы смежности и списка смежности, также научился преобразовывать рекурсивной реализацию обхода графа к не рекурсивной.

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

// Обход в глубину для матрицы смежности

void Matr(int\*\* matrica, int v, int\* visited, int n) {

visited[v] = 1;

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

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

if (matrica[v][i] == 1 && !visited[i]) {

Matr(matrica, i, visited, n);

}

}

}

// Обход в глубину для списка смежности

void Spisok(int\*\* zxc, int v, int\* visited, int n) {

visited[v] = 1;

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

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

if (zxc[v][i] == 1 && !visited[i]) {

Spisok(zxc, i, visited, n);

}

}

}

int main() {

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int n;

printf("Введите кол-во вершин: ");

scanf("%d", &n);

int\*\* matrica = (int\*\*)malloc(n \* sizeof(int\*));

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

matrica[i] = (int\*)malloc(n \* sizeof(int));

}

// Матрица смежности

printf("Матрица смежности:\n");

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

for (int j = i; j < n; j++) {

if (i == j) {

matrica[i][j] = 0;

}

else {

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

matrica[j][i] = matrica[i][j];

}

}

}

// Вывод матрицы

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

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

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

}

printf("\n");

}

printf("\n");

int\* visited = (int\*)malloc(n \* sizeof(int));

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

visited[i] = 0;

}

// Обход матрицы в глубину

printf("Обход в глубину матрицы:\n");

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

int v = i;

if (!visited[v]) {

Matr(matrica, v, visited, n);

}

}

// Сброс посещенных вершин

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

visited[i] = 0;

}

int\*\* mass = (int\*\*)malloc(n \* sizeof(int\*));

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

mass[i] = (int\*)malloc(n \* sizeof(int));

}

// Заполнение списка смежности на основе матрицы

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

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

if (matrica[i][j] == 1) {

mass[i][j] = 1;

mass[j][i] = 1; // добавляем обратное ребро

}

}

}

// Вывод списка смежности

printf("\nСписок смежности:\n");

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

printf("Вер. %d: ", i + 1);

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

if (mass[i][j] == 1) {

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

}

}

printf("\n");

}

// Обход в глубину для списка

printf("\nОбход в глубину списка :\n");

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

int v = i;

if (!visited[v]) {

Spisok(mass, v, visited, n);

}

}

// Освобождение памяти

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

free(matrica[i]);

free(mass[i]);

}

free(matrica);

free(mass);

free(visited);

return 0;

}