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”

Выполнили студенты группы 22ВВВ2:

Разин Д.С

Беккаревич К.А

Приняли:

Акифьев И. В.

Юрова О.В

Пенза 2023

**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.

**Задание №1.1.2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <clocale>

#include <time.h>

#define MAX\_VERTICES 100

// Генерация случайной матрицы смежности

void generrand(int matrix [MAX\_VERTICES][MAX\_VERTICES], int numVertices, int numEdges) {

int i, j;

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

for (j = i; j < numVertices; j++) {

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

matrix[j][i] = matrix[i][j];

if (i == j) matrix[i][j] = 0;

}

}

}

// Обход в глубину

void depthFirstSearch(int matrix[MAX\_VERTICES][MAX\_VERTICES], bool visited[MAX\_VERTICES], int currentVertex, int numVertices) {

printf("Пройдена вершина: %d\n", currentVertex);

visited[currentVertex] = true;

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

if (matrix[currentVertex][i] == 1 && !visited[i]) {

depthFirstSearch(matrix, visited, i, numVertices);

}

}

}

int main() {

srand (time(NULL));

setlocale(LC\_ALL, "Russian");

int numVertices;

int numEdges;

int matrix;

printf("Введите желаемое количесвто вершин - ");//Задаем размер матрицы смежности

scanf("%d", &numVertices);

printf("\nВведите желаемое количесвто ребер - ");

scanf("%d", &numEdges);

int adjMatrix[MAX\_VERTICES][MAX\_VERTICES];

generrand(adjMatrix, numVertices, numEdges);

printf("\nМатрица смежности:\n");

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

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

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

}

printf("\n");

}

bool visited[MAX\_VERTICES];

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

visited[i] = false;

}

int startVertex;

printf("Введите стартовую вершину (от 0 до %d): ", numVertices - 1);

scanf("%d", &startVertex);

if (startVertex >= 0 && startVertex < numVertices) {

printf("Результат обхода в глубину, начиная с вершины %d:\n", startVertex);

depthFirstSearch(adjMatrix, visited, startVertex, numVertices);

}

else {

printf("Некорректный ввод стартовой вершины.\n");

}

getchar();

getchar();

return 0;

}

**Задание №1.3**

**#define**\_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<clocale>

#defineMAX\_VERTICES 100

// Функция для добавления вершины в список смежности

voidaddEdge(int\* adjacencyLists[], intu, intv) {

adjacencyLists[u][v] = 1;

adjacencyLists[v][u] = 1;

}

// Функциядлявыводаспискасмежности

voidprintAdjacencyList(int\* adjacencyLists[], intnumVertices) {

for (inti = 0; i<numVertices; i++) {

printf("Список смежности для вершины %d: ", i);

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

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

printf("%d -> ", j);

}

}

printf("||| \n");

}

}

// Функция для обхода в глубину

voidDFS(int\* adjacencyLists[], intnumVertices, intvertex, intvisited[]) {

visited[vertex] = 1;

printf("Посещеннаявершина: %d\n", vertex);

for (inti = 0; i<numVertices; i++) {

if (adjacencyLists[vertex][i] == 1 && !visited[i]) {

DFS(adjacencyLists, numVertices, i, visited);

}

}

}

intmain() {

setlocale(LC\_ALL, "Russian");

intnumVertices = 6;

int\* adjacencyLists[MAX\_VERTICES];

for (inti = 0; i<numVertices; i++) {

adjacencyLists[i] = (int\*)malloc(numVertices \* sizeof(int));

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

adjacencyLists[i][j] = 0; // Инициализация нулями

}

}

addEdge(adjacencyLists, 0, 5);

addEdge(adjacencyLists, 0, 2);

addEdge(adjacencyLists, 1, 3);

addEdge(adjacencyLists, 2, 4);

addEdge(adjacencyLists, 3, 5);

intvisited[MAX\_VERTICES] = { 0 };

printf("Списоксмежности:\n");

printAdjacencyList(adjacencyLists, numVertices);

intstartVertex;

h1:

printf("Введитестартовуювершину (от 0 до %d): ", numVertices - 1);

scanf("%d", &startVertex);

if (startVertex>= 0 &&startVertex<numVertices) {

//printf("Результат обхода в глубину, начиная с вершины %d:\n", startVertex);

printf("\n");

DFS(adjacencyLists, numVertices, startVertex, visited);

}

else {

printf("Ерор\n");

goto h1;

}

return 0;

}

**Задание №2**

#include <stdio.h>

#include <stdlib.h>

#include <clocale>

#include <time.h>

#define MAX\_VERTICES 100

// Функция для добавления вершины в стек

void push(int stack[], int\* top, int vertex) {

stack[++(\*top)] = vertex;

}

// Функция для извлечения вершины из стека

int pop(int stack[], int\* top) {

return stack[(\*top)--];

}

// Функция для обхода в глубину без использования рекурсии

void DFS(int adjacencyMatrix[][MAX\_VERTICES], int numVertices, int startVertex) {

int visited[MAX\_VERTICES] = { 0 };

int stack[MAX\_VERTICES];

int top = -1;

push(stack, &top, startVertex);

printf("Результат обхода в глубину:\n");

while (top != -1) {

int currentVertex = pop(stack, &top);

if (visited[currentVertex] == 1) {

continue;

}

printf("Посещена вершина: %d\n", currentVertex);

visited[currentVertex] = 1;

for (int i = numVertices - 1; i >= 0; i--) {

if (adjacencyMatrix[currentVertex][i] == 1 && visited[i] == 0) {

push(stack, &top, i);

}

}

}

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(NULL));

int numVertices;

int adjacencyMatrix[MAX\_VERTICES][MAX\_VERTICES] = { 0 };

printf("Введите количество вершин - : ");

scanf\_s("%d", &numVertices);

printf("\n");

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

{

for (int j = i; j < numVertices; j++)

{

adjacencyMatrix[i][j] = 0 + rand() % 2;

adjacencyMatrix[j][i] = adjacencyMatrix[i][j];

if (i == j) adjacencyMatrix[i][j] = 0;

}

}

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

{

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

{

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

}

printf("\n");

}

printf("\n");

int startVertex;

printf("Введите с какой вершины хотите начать (от 0 до %d): ", numVertices - 1);

scanf\_s("%d", &startVertex);

DFS(adjacencyMatrix, numVertices, startVertex);

getchar();

getchar();

return 0;

}

**Результаты работы программ**

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

**Рисунок 1 - Результат работы программы lab7.12**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

**Рисунок 2 - Результат работы программы lab7.1.3**

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

**Рисунок 3 - Результат работы программы lab7.2**

**Вывод**

В результате выполнения лабораторной работы были успешно реализованы алгоритмы построения матрицы смежности, усвоены и практически проверены навыки в алгоритме реализации проверки графа в глубину.