Министерство образования и науки Российской Федерации Казанский Национальный Исследовательский технический университет  
им. А.Н. Туполева – КАИ

Кафедра Систем автоматизированного проектирования

**Отчет по лабораторной работе №5**

по дисциплине «Методы программирования САПР»

Выполнил:

Студент группы 4215

Закиров И. Р.

Проверил:

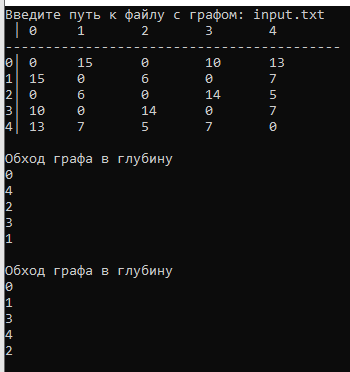
Преподаватель кафедры САПР

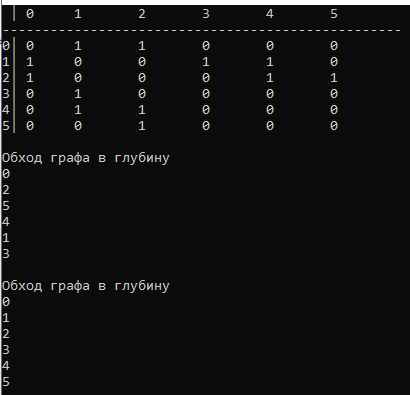
Голубович Д. А.

Казань 2020

**Задание:** Составьте программу, реализующую поиск по графу в глубину и в ширину. Входные данных программа получает из текстового файла.

**Ход работы**





**Исходный код**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Graph

{

public:

int vertex\_num;

int edges\_num;

int\*\* adj\_matrix;

Graph(string filePath)

{

readGraph(filePath);

printGraph();

}

void readGraph(string filePath) {

ifstream inFile(filePath);

int v1, v2, edge;

inFile >> vertex\_num >> edges\_num;

adj\_matrix = new int\* [vertex\_num];

for (int i = 0; i < vertex\_num; i++) {

adj\_matrix[i] = new int[vertex\_num];

for (int j = 0; j < vertex\_num; j++) adj\_matrix[i][j] = 0;

}

for(int e = 0; e < edges\_num; e++)

{

inFile >> v1 >> v2 >> edge;

adj\_matrix[v1][v2] = adj\_matrix[v2][v1] = edge;

}

inFile.close();

}

void printGraph() {

cout << " ¦ ";

for (int i = 0; i < vertex\_num; i++) cout << i << "\t ";

cout << endl;

for (int i = 0; i < 8 \* vertex\_num + 2; i++) cout << "-";

cout << endl;

for (int i = 0; i < vertex\_num; i++) {

cout << i << "¦ ";

for (int j = 0; j < vertex\_num; j++) cout << adj\_matrix[i][j] << "\t ";

cout << endl;

}

cout << endl;

}

};

void DFS(Graph graph, int Node)

{

bool\* Visited = new bool[graph.vertex\_num];

for (int i = 0; i < graph.vertex\_num; i++) Visited[i] = false;

int\* List = new int[graph.vertex\_num];

int Head;

for (int i = 0; i < graph.vertex\_num; i++) List[i] = 0;

Head = 0;

List[Head] = Node;

Visited[Node] = true;

while (Head >= 0) {

Node = List[Head];

Head--;

cout << Node << endl;

for (int i = 0; i < graph.vertex\_num; i++) {

if (graph.adj\_matrix[Node][i] && !Visited[i]) {

Head++;

List[Head] = i;

Visited[i] = true;

}

}

}

}

void BFS(Graph graph, int Node)

{

bool\* Visited = new bool[graph.vertex\_num];

for (int i = 0; i < graph.vertex\_num; i++) Visited[i] = false;

int\* List = new int[graph.vertex\_num];

int Count, Head;

for (int i = 0; i < graph.vertex\_num; i++) List[i] = 0;

Count = Head = 0;

List[Count++] = Node;

Visited[Node] = true;

while (Head < Count) {

Node = List[Head++];

cout << Node << endl;

for (int i = 0; i < graph.vertex\_num; i++) {

if (graph.adj\_matrix[Node][i] && !Visited[i]) {

List[Count++] = i;

Visited[i] = true;

}

}

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

string path;

cout << "Введите путь к файлу с графом: ";

cin >> path;

Graph graph = Graph(path);

cout << "Обход графа в глубину" << endl;

DFS(graph, 0);

cout << endl << "Обход графа в глубину" << endl;

BFS(graph, 0);

}

**Вывод:** В ходе лабораторной работы были написана программа, реализующая алгоритмы на графах.