Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

**Кафедра автоматизированных систем управления (АСУ)**

**Графы**

Отчет по лабораторной работе № 2 по дисциплине «Структуры и алгоритмы обработки данных в ЭВМ»

Выполнил: студент

специальности 09.03.01, группа   
з-436У-а, поток 75

Красноперов В.С.

«14» октября 2017 г.

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**г. Калининград 2017**

Содержание

[Тема работы 3](#_Toc495770748)

[Цель работы 3](#_Toc495770749)

[Индивидуальное задание 3](#_Toc495770750)

[Алгоритм решения задачи 3](#_Toc495770751)

[Результаты работы программы 3](#_Toc495770752)

[Выводы 3](#_Toc495770753)

[Приложение А. Листинг программы 3](#_Toc495770754)

# Тема работы

Алгоритмы работы с графами

# Цель работы

Получить практические навыки представления графов в памяти ЭВМ, реализовать на языке программирования C/C ++ алгоритмы работы с графами.

# Индивидуальное задание

Используя метод поиска в глубину, найти и вывести путь в ориентированном графе между двумя вершинами. Номера начальной и конечной вершин ввести с клавиатуры. Граф задать в текстовом файле матрицей вeсов

# Алгоритм решения задачи

1. Осуществить ввод номера первой вершины
2. Осуществить ввод номера второй вершины
3. Заполнить матрицу весов в памяти
4. Методом поиска в глубину найти путь до второй вершины
5. Вывести результат
6. Освободить память

# Результаты работы программы

Enter first node:3

Enter second node:4

Path is: 3 > 1 > 2 > 4

# Выводы

Выполняя данную лабораторную работу я получил практические навыки работы с графами и их представлениями в памяти ЭВМ

# 

# Приложение А. Листинг программы

#include <stdlib.h>

#include <string.h>

#include <fstream>

#include <iostream>

using namespace std;

string find\_path(int size, bool\* visited, int\*\* graph, int node1, int node2, string current\_path = "") {

visited[node1] = true;

if (node1 == node2) {

return current\_path;

}

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

{

int next = graph[node1][i];

if (next != 0) {

string path = find\_path(size, visited, graph, i, node2, current\_path + " > " + to\_string(i + 1));

if (path != "") {

return path;

}

}

}

return "";

}

int main()

{

int n1, n2;

printf("Enter first node:");

scanf("%d", &n1);

printf("Enter second node:");

scanf("%d", &n2);

ifstream stream("graph.txt");

int size;

stream >> size;

int\*\* graph = new int\*[size];

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

{

graph[i] = new int[size];

}

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

{

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

{

stream >> graph[i][j];

}

}

bool\* visited = new bool[size];

string path = find\_path(size, visited, graph, n1 - 1, n2 - 1);

if (path == "") {

cout << "There is no path";

} else {

cout << "Path is: " << n1 << path << "\n";

}

free(graph);

free(visited);

return 0;

}