# Федеральное государственное бюджетное образовательное учреждение высшего образования «Алтайский государственный технический университет им. И.И. Ползунова»

Факультет (институт) Информационных технологий

Кафедра Прикладная математика	-
	цищен с оценкой
Отчет	
по лабораторной ра	
<u>Разработка и анализ про</u>	ограмм обхода графа
(название лабораторно	й работы)
по дисциплине	ры данных
(наименование дисц	иплины)
ЛР 09.03.04.13	TO 000.
(обозначение докум	мента)
Студент группы ПИ-21	А.А.Лихтинфельд
Преподаватель доцент, доцент	(инициалы, фамилия) А.И.Потупчик
(должность, ученое звание)	(инициалы, фамилия)

### Задание:

- 1. Разработать и отладить программу на языке C++, реализующую работу с графом в соответствии с вариантом. Выполнить оценку временной и емкостной сложности программы.
- 2. Исходные данные поместить в файл input.dat
- 3. Результаты вывести на экран. Исходные данные и результаты вывести также в выходной файл output.dat

## Вариант №3

Найти самый длинный простой путь в графе.

# Текст программы:

```
#include <iostream>
#include <vector>
#include inits.h>
#include <fstream>
#include <locale>
using namespace std;
class Graph {
  int V; // Количество вершин
  vector<vector<int>> adj; // Список смежности
  vector<br/>bool> visited; // Массив для отслеживания посещенных вершин
  int longestPathLength;
  vector<int> longestPath;
  void DFS(int v, vector<int>& path) {
    visited[v] = true;
    path.push back(v);
    // Если текущий путь длиннее найденного ранее, обновляем его
    if (path.size() > longestPathLength) {
       longestPathLength = path.size();
       longestPath = path;
    }
    // Рекурсивно посещаем все смежные вершины
    for (int i : adj[v]) {
       if (!visited[i]) {
         DFS(i, path);
    }
    // Возвращаемся назад и помечаем вершину как непосещенную
    path.pop back();
```

```
visited[v] = false;
  }
public:
  Graph(int V): V(V), adj(V), visited(V, false), longestPathLength(0) {}
  void addEdge(int v, int w) {
     adj[v].push back(w); // Добавляем ребро в граф
  }
  void findLongestPath() {
     for (int i = 0; i < V; i++) {
       vector<int> path;
       DFS(i, path); // Ищем самый длинный путь из каждой вершины
    ofstream outputFile("output.dat");
    cout << "Самый длинный путь имеет длину: " << longestPathLength << endl;
    cout << "Путь: ";
    for (int v : longestPath) {
       cout << v << " ";
       outputFile << v << " ";
    cout << endl;
    outputFile.close();
    cout << "Результат поиска записан в файл output.dat" << endl;
  }
};
void input_file(int numVertices, int numEdges) {
  // Открытие выходного файла для сохранения графа
  ofstream outputFile("input.dat");
  outputFile << numVertices << " " << numEdges << endl;
  // Считывание ребер и сохранение их в файл
  for (int i = 0; i < numEdges; ++i) {
    int from, to;
    cout << "Введите ребро (от, до): ";
    cin >> from >> to;
    outputFile << from << " " << to << endl;
  }
  outputFile.close();
  cout << "Граф успешно сохранен в файле input.dat" << endl;
}
int main() {
```

```
setlocale(LC ALL, "Rus");
int numVertices, numEdges;
// Ввод количества вершин и ребер с консоли
cout << "Введите количество вершин: ";
cin >> numVertices;
cout << "Введите количество ребер: ";
cin >> numEdges;
input file(numVertices, numEdges);
ifstream inputFile("input.dat");
inputFile >> numVertices >> numEdges;// Считываем количество вершин и рёбер
Graph g(numVertices);
for (int i = 0; i < numEdges; ++i) {
  int from, to;
  inputFile >> from >> to;// Считываем ребро
  g.addEdge(from, to);
}
g.findLongestPath();
inputFile.close();
return 0;
```

# Пример работы программы:

}

```
Консоль отладки Microsoft Visual Studio

Введите количество вершин: 4

Введите количество ребер: 5

Введите ребро (от, до): 0 1

Введите ребро (от, до): 1 3

Введите ребро (от, до): 2 3

Введите ребро (от, до): 2 1

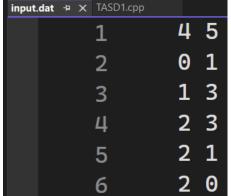
Введите ребро (от, до): 2 0

Граф успешно сохранен в файле input.dat

Самый длинный путь имеет длину: 4

Путь: 2 0 1 3

Результат поиска записан в файл output.dat
```





<u>Оценка временной сложности программы:</u> O(V+E), где V - количество вершин, E - количество рёбер

<u>Оценка ёмкостной сложности программы:</u> O(V+E), где V - количество вершин, E - количество рёбер