МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6

по дисциплине
«Структуры и алгоритмы обработки данных»

Тема. Основные алгоритмы работы с графами

Выполнил студент группы ИКБО-25-22 Ракитин В.А.

Принял доцент Бузыкова Ю.С.

Москва 2023

Цель: изучить теорию и научиться программировать основные алгоритмы работы с графами.

Ход работы

Задание 1

Реализация программы

Код основной программы представлен в листинге 1.1.

Листинг 1.1 – Код основной программы

```
#include <iostream>
#include <list>
#include <array>
#include <limits>

using namespace std;

// Создаем класс графа
class Graph {
    int V;
    list<pair<int, int>>* adj;

public:
    Graph(int V);

    void addEdge(int u, int v, int w);

    void printSolution(const array<int, 100>& dist, int n);

    void shortestPath(int startVertex, int endVertex);
};

// Инициализируем граф
Graph::Graph(int V) {
    this->V = V;
    adj = new list<pair<int, int>>[V];
}

// Добавляем ребра
void Graph::addEdge(int u, int v, int w) {
    adj[u].push_back(make_pair(v, w));
    adj[v].push_back(make_pair(u, w));
}

// Реализуем алгоритм Беллмана-Форда
void Graph::shortestPath(int startVertex, int endVertex) {
    array<int, 100> dist;

    for (int i = 0; i < V; i++)
        dist[i] = numeric_limits<int>::max();

    dist[startVertex] = 0;

    for (int i = 1; i <= V - 1; i++) {
        for (int j = 0; j < V; j++) {
            for (auto x : adj[j]) {
                if (dist[j] != numeric_limits<int>::max() && dist[j] +
x.second < dist[x.first])
                    dist[x.first] = dist[j] + x.second;
```

```
                }
            }
        }

        printSolution(dist, endVertex);
}

// Выводим кратчайший путь
void Graph::printSolution(const array<int, 100>& dist, int endVertex) {
    cout << "Вершина\t\tРасстояние от источника\n";
    for (int i = 0; i < V; i++)
        cout << i << "\t\t" << dist[i] << "\n";
}

int main() {
    setlocale(0, "ru");
    int V, E, tempX, tempY, tempZ;

    cout << "Введите количество вершин: ";
    cin >> V;

    cout << "Введите количество ребер: ";
    cin >> E;

    Graph g(V);

    for (int i = 0; i < E; i++) {
        cout << "Введите край " << i + 1 << " (Источник, Назначение, Вес) :
";
        cin >> tempX >> tempY >> tempZ;
        g.addEdge(tempX, tempY, tempZ);
    }

    int startVertex, endVertex;

    cout << "Введите начальную вершину: ";
    cin >> startVertex;

    cout << "Введите конечную вершину: ";
    cin >> endVertex;

    g.shortestPath(startVertex, endVertex);

    return 0;
}
```

Результаты тестирования

На рисунке 1.1 представлено тестирование программы.

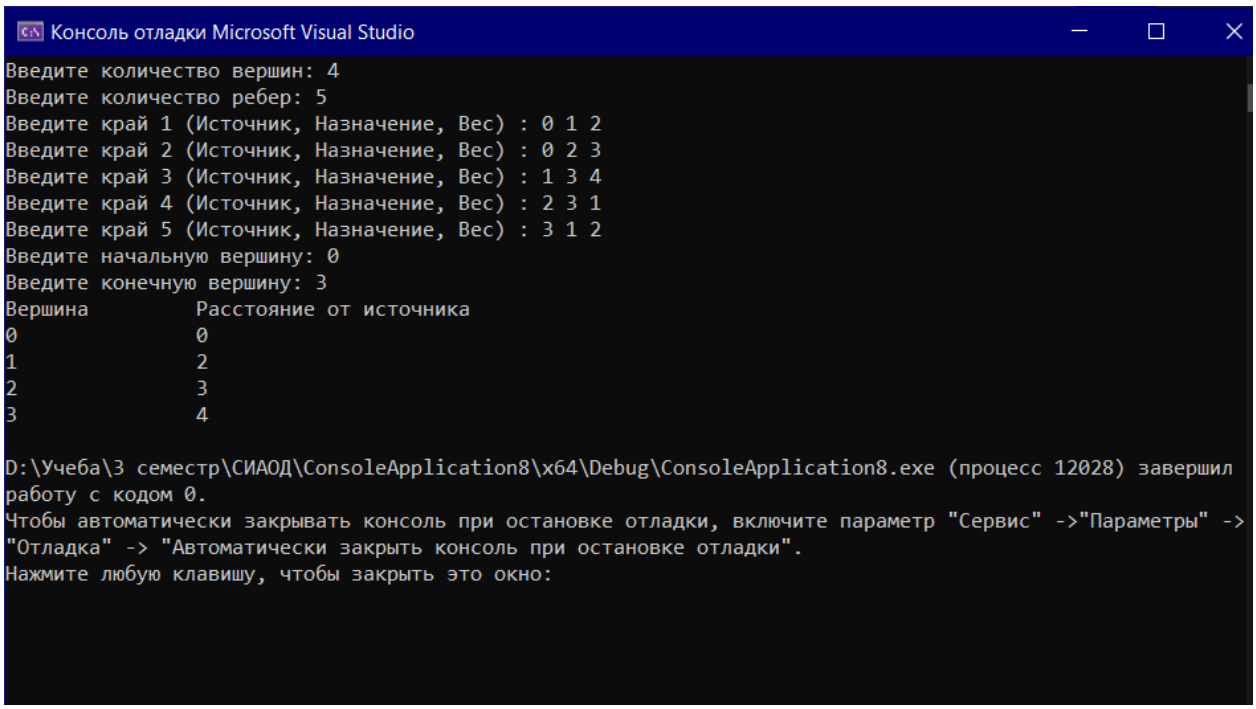

Рисунок 1.1 – Результат тестирования на примере введенных данных

Вывод: в ходе выполнения практической работы были получены навыки разработки алгоритмов работы с графами.