**Цель работы:** научиться работать с стеком и очередями

**Задание:** написать программу, которая по заданным номерам остановок I и J определяет наиболее быстрый путь перемещения пассажира из остановки I в остановку J с использованием имеющихся маршрутов автобусов при условий, что время движения между соседними остановками у всех маршрутов одинаково и в 3 раза меньше времени изменения маршрута. Кроме того, автобусы могут двигаться в обоих направлениях.

**Листинг программы:**

**Файл main.cpp**

#include <iostream>

#include <vector>

#include <unordered\_map>

#include <queue>

#include <tuple>

#include <limits>

using namespace std;

const int MOVE\_COST = 1;

const int SWITCH\_COST = 3;

const int INF = numeric\_limits<int>::max();

struct Edge {

int to;

int weight;

int routeId;

};

int main() {

vector<vector<int>> routes = {

{1, 2, 3, 4},

{3, 5, 6, 4},

{6, 7, 8}

};

unordered\_map<int, vector<Edge>> graph;

for (int routeId = 0; routeId < routes.size(); ++routeId) {

const vector<int>& route = routes[routeId];

for (size\_t i = 0; i + 1 < route.size(); ++i) {

int from = route[i];

int to = route[i + 1];

graph[from].push\_back({to, MOVE\_COST, routeId});

graph[to].push\_back({from, MOVE\_COST, routeId});

}

}

int start, end;

cout << "Введите начальную остановку (I): ";

cin >> start;

cout << "Введите конечную остановку (J): ";

cin >> end;

priority\_queue<

tuple<int, int, int>,

vector<tuple<int, int, int>>,

greater<tuple<int, int, int>>

> pq;

unordered\_map<int, unordered\_map<int, int>> dist;

for (int routeId = 0; routeId < routes.size(); ++routeId) {

for (int stop : routes[routeId]) {

if (stop == start) {

pq.push(make\_tuple(0, start, routeId));

dist[start][routeId] = 0;

}

}

}

while (!pq.empty()) {

int cost, stop, route;

tie(cost, stop, route) = pq.top();

pq.pop();

if (stop == end) {

cout << "Минимальное время: " << cost << endl;

return 0;

}

for (const Edge& edge : graph[stop]) {

int nextCost = cost + ((edge.routeId == route) ? MOVE\_COST : SWITCH\_COST);

if (dist[edge.to].find(edge.routeId) == dist[edge.to].end()

|| nextCost < dist[edge.to][edge.routeId]) {

dist[edge.to][edge.routeId] = nextCost;

pq.push(make\_tuple(nextCost, edge.to, edge.routeId));

}

}

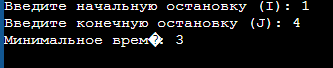
}

cout << "Путь не найден." << endl;

return 0;

}

**Результат выполнения программы:**



**Вывод:** в ходе выполнения лабораторной работы я научился работать с стеком и очередями в c++, а также в данной лабораторной работе был реализован алгоритм поиска кратчайшего пути на автобусных маршрутах с учётом **времени перемещения и пересадок**.