МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИМЕНИ И. С. ТУРГЕНЕВА»

                                                                                                                                                                   Кафедра информационных систем и цифровых технологий

**ОТЧЁТ**

по лабораторной работе № 2

на тему: «Структурное тестирование программного обеспечения»

по дисциплине «Качество и тестирование программного обеспечения»

Выполнили: Банных М.А., Мельников А.Е.

Институт приборостроения, автоматизации и информационных

технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 21ПГ

Проверил: Олькина Е.В.

Отметка о зачёте:

Дата: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025г.

Орёл, 2025

**1.1 Уточняющие вопросы к требованиям заказчика:**

1. Как будет производиться стирка?
2. Зачем вводить ограничение на минимальную загрузку? Какова максимальная?
3. Что из себя представляют быстрая и полная стирки? Только эти два режима?
4. До каких температур должен осуществляться нагрев?
5. Какая информация должна быть выведена на дисплей?
6. С какой целью машина должна подключаться к Wi-Fi?

**1.2 Структурное тестирование алгоритма по методу базового пути.**

Код алгоритма:

#include <iostream>

using namespace std;

const int MAX\_NODES = 10;

int graph[MAX\_NODES][MAX\_NODES];

int visited[MAX\_NODES];

int parent[MAX\_NODES];

void addEdge(int u, int v) {

graph[u][v] = 1;

graph[v][u] = 1;

}

bool bfsPath(int start, int end, int numNodes) {

int queue[MAX\_NODES], front = 0, rear = 0;

for (int i = 0; i < numNodes; i++) {

visited[i] = 0;

parent[i] = -1;

}

queue[rear++] = start;

visited[start] = 1;

while (front < rear) {

int node = queue[front++];

if (node == end) {

return true;

}

for (int neighbor = 0; neighbor < numNodes; neighbor++) {

if (graph[node][neighbor] == 1 && !visited[neighbor]) {

visited[neighbor] = 1;

parent[neighbor] = node;

queue[rear++] = neighbor;

}

}

}

return false;

}

void printPath(int start, int end) {

if (parent[end] == -1) {

cout << "Путь не найден." << endl;

return;

}

int path[MAX\_NODES];

int pathLength = 0;

for (int at = end; at != -1; at = parent[at]) {

path[pathLength++] = at;

}

cout << "Путь от " << start << " до " << end << ": ";

for (int i = pathLength - 1; i >= 0; i--) {

cout << path[i] << (i > 0 ? " -> " : "");

}

cout << endl;

}

int main() {

setlocale(LC\_ALL, "Russian");

int numNodes = 6;

addEdge(1, 2);

addEdge(1, 3);

addEdge(2, 4);

addEdge(3, 4);

addEdge(4, 5);

int start, end;

while (true) {

cout << "Введите начальную и конечную вершины (от 1 до " << numNodes-1 << "): ";

cin >> start >> end;

if (start < 1 || start > numNodes-1 || end < 1 || end > numNodes-1) {

cout << "Некорректный ввод. Попробуйте снова." << endl << endl;

continue;

}

if (bfsPath(start, end, numNodes)) {

printPath(start, end);

break;

}

else {

cout << "Путь не найден. Попробуйте снова." << endl;

}

}

return 0;

}

Вычисление цикломатической сложности:

1. 24 дуги – 19 узлов + 2 = 7

2. 6 предикатных узлов + 1 = 7

Итого 7 регионов.

1-2-3-...-6-7-8-9-10-19

1-2-3-...-6-7-8-9-10-11-12-13-14-15-16-17-18-19

1-2-3-...-6-7-8-9-10-11-12-13-14-18-19

1-2-3-...-6-7-8-9-10-11-12-13-18-19

1-2-3-...-6-7-8-9-10-11-12-19

1-2-3-...-6-7-8-19

1-2-3-6... НЕВОЗМОЖНО (цикл гарантированно производит несколько итераций)

