Министерство образования, культуры и исследований Республики Молдова

Технический университет Молдовы

Департамент программной инженерии и автоматики

***Отчет***

по лабораторной работе №2

### Предмет: Programarea în Reţea

**Тема:** мульипоточность

**Выполнил:** Жорин Иван

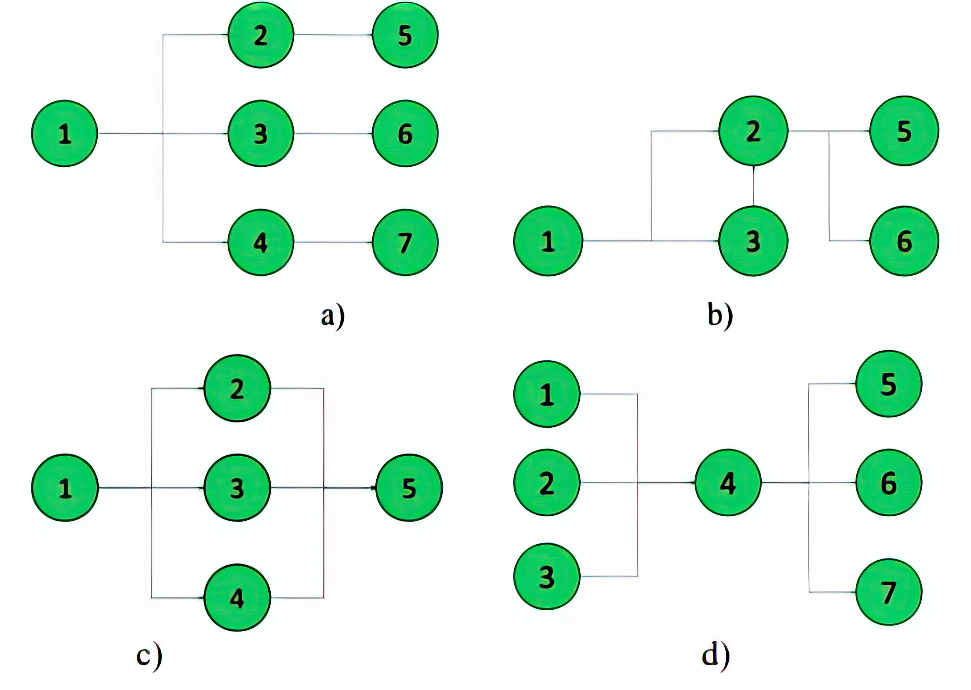
**Проверила:**  Ангилоглу А.

Кишинёв 2021

***Тема:*** Многопоточность

***Цель работы*:** понимание моделей параллельного исполнения и знание основных методов синхронизации действий на основе атомарных операций светофора; конкретная цель - создать приложение Java, которое обязательно будет использовать различные структуры в контексте параллельного выполнения.

***Многопоточность*** представляет свойство платформы или приложения, которое заключается в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени. При выполнении некоторых задач такое разделение может достичь более эффективного использования ресурсов вычислительной машины.



***КОД ПРОГРАММЫ:***

**#include <iostream>**

**#include <thread>**

**using namespace std;**

**void foo(int Z)**

**{**

**for (int i = 0; i < Z; i++) {**

**cout << "Поток использует функцию"**

**" указатель как вызываемый \n";**

**}**

**}**

**// Вызываемый объект 1**

**class thread\_obj {**

**public:**

**void operator()(int x)**

**{**

**for (int i = 0; i < x; i++)**

**cout << "Поток использует функцию"**

**" объект как вызываемый \n";**

**}**

**};**

**int main()**

**{**

**setlocale(LC\_ALL, "Russian");**

**cout << "Потоки 1, 2 , 3 "**

**"работают независимо" << endl;**

**thread th1(foo, 3);**

**thread th2(thread\_obj(), 3);**

**// Определение выражения лямбда**

**auto f = [](int x) {**

**for (int i = 0; i < x; i++)**

**cout << "Поток использует лямбду"**

**" выражение как вызываемое \n";**

**};**

**// Этот поток запускается с использованием**

**// выражения лямбда как вызываемого**

**thread th3(f, 3);**

**// Ожидание завершения потоков**

**// Ожидание завершения потока 1**

**th1.join();**

**// Ожидание завершения потока 2**

**th2.join();**

**// Ожидание завершения потока 3**

**th3.join();**

**return 0;**

**}**

**Результат работы программы:**

