МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

**«Исследование работы многопоточных приложений. Изучение механизмов синхронизации потоков.»**

Отчет по лабораторной работе № 7 по дисциплине

«Технологии программирования»

Выполнил студент группы: ПИб-2301\_\_\_\_\_\_\_\_\_\_\_\_\_ /Морозов Д.А.

Проверил преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М.Л.

**Киров 2023**

**Цели работы:**

Разработать многопоточное приложения, в соответствии с указанным преподавателем вариантом и механизмом синхронизации. Оценить уместность предложенного механизма синхронизации при решении  задачи.

**Задание:**

Пять философов сидят возле круглого стола. Они проводят жизнь, чередуя приемы пищи и размышления. В центре стола находится большое блюдо спагетти. Спагетти длинные и запутанные, философам тяжело управляться с ними, поэтому каждый из них. чтобы съесть порцию, должен пользоваться двумя вилками. К несчастью, философам дали только пять вилок. Между каждой парой философов лежит одна вилка, поэтому эти высококультурные и предельно вежливые люди договорились, что каждый будет пользоваться только теми вилками, которые лежат рядом с ним (слева и справа). Написать многопоточную программу, моделирующую поведение философов с помощью семафоров. Программа должна избегать фатальной ситуации, в которой все философы голодны, но ни один из них не может взять обе вилки (например, каждый из философов держит по одной вилки н не хочет отдавать ее). Решение должно быть симметричным, то есть все потоки-философы должны выполнять один и тот же код.

**Описание взаимодействия потоков:**

Общая цель задачи заключается в том, чтобы симулировать ситуацию, в которой пять философов сидят за круглым столом и чередуют время между обедом и размышлениями, используя вилки, которые находятся между ними.

Каждый философ представлен отдельным потоком, запущенным функцией Philosopher. Каждый поток выполняет следующие шаги в бесконечном цикле:

1. Философ начинает думать, устанавливая свое состояние THINKING и выводя сообщение об этом.
2. Затем философ становится голодным, устанавливая свое состояние HUNGRY и выводя сообщение об этом.
3. Философ пытается получить доступ к еде, вызывая WaitForSingleObject для ожидания семафора philSemaphore. Этот семафор служит ограничением на количество философов, которые могут кушать одновременно (в данном случае, максимум двое).
4. Философ проверяет наличие свободных вилок слева и справа. Если одна из вилок занята, он освобождает семафор и ждет снова.
5. Если обе вилки свободны, философ берет их, устанавливает свое состояние EATING и выводит сообщение об этом.
6. Философ кушает в течение 3 секунд с помощью std::this\_thread::sleep\_for.
7. После окончания приема пищи философ освобождает вилки, устанавливает их состояние как свободные и освобождает семафор philSemaphore, чтобы позволить другим философам начать свой обед.

В функции main создается семафор philSemaphore с начальным значением 2 (два философа начинают кушанье, максимальное количество философов). Затем все вилки устанавливаются в состояние доступности. Создаются потоки для каждого философа, которые выполняют функцию **Philosopher**. Затем выводится сообщение о начале обеда. Когда пользователь нажимает клавишу Enter, потоки философов завершаются и семафор

**Листинг кода:**

main.cpp

#include <iostream>

#include <thread>

#include <chrono>

#include <windows.h>

const int numPhilosophers = 5;

HANDLE philSemaphore;

bool forksAvailable[numPhilosophers];

//состояния философов

enum PhilosopherState {

THINKING,

HUNGRY,

EATING

};

PhilosopherState philosopherStates[numPhilosophers];

void Philosopher(int philosopherId) {

int leftFork = philosopherId;

int rightFork = (philosopherId + 1) % numPhilosophers;

while (true) {

// Философ думает

philosopherStates[philosopherId] = THINKING;

std::cout << "Philosopher " << philosopherId << " is thinking." << std::endl;

std::this\_thread::sleep\_for(std::chrono::seconds(1));

// Философ голоден

philosopherStates[philosopherId] = HUNGRY;

std::cout << "Philosopher " << philosopherId << " is hungry." << std::endl;

// Ожидание кушанья философа

WaitForSingleObject(philSemaphore, INFINITE);

// Проверяем свободные вилки

while (!forksAvailable[leftFork] || !forksAvailable[rightFork]) {

// Если нет ни одной свободной – освобождаем и начинаем снова

ReleaseSemaphore(philSemaphore, 1, NULL);

WaitForSingleObject(philSemaphore, INFINITE);

}

// Забираем вилки

forksAvailable[leftFork] = false;

forksAvailable[rightFork] = false;

// Философ кушает

philosopherStates[philosopherId] = EATING;

std::cout << "Philosopher " << philosopherId << " is eating." << std::endl;

std::this\_thread::sleep\_for(std::chrono::seconds(3));

// Освобождаем вилки

forksAvailable[leftFork] = true;

forksAvailable[rightFork] = true;

// Освобождаем поток

ReleaseSemaphore(philSemaphore, 1, NULL);

}

}

int main() {

// Создаем семафор с начальным значением 2 (два философа начинаю кушанье, максимальное – количество философов)

philSemaphore = CreateSemaphore(NULL, 2, numPhilosophers, NULL);

// Освобождаем все вилки в самом начале

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

forksAvailable[i] = true;

}

// Создаем потоки философы

std::thread philosopherThreads[numPhilosophers];

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

philosopherThreads[i] = std::thread(Philosopher, i);

}

std::cout << "Dinner started!" << std::endl;

std::cin.ignore();

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

philosopherThreads[i].join();

}

// Закрываем семафоры

CloseHandle(philSemaphore);

std::cout << "Dinner finished!" << std::endl;

return 0;

}

**Экранные формы:**

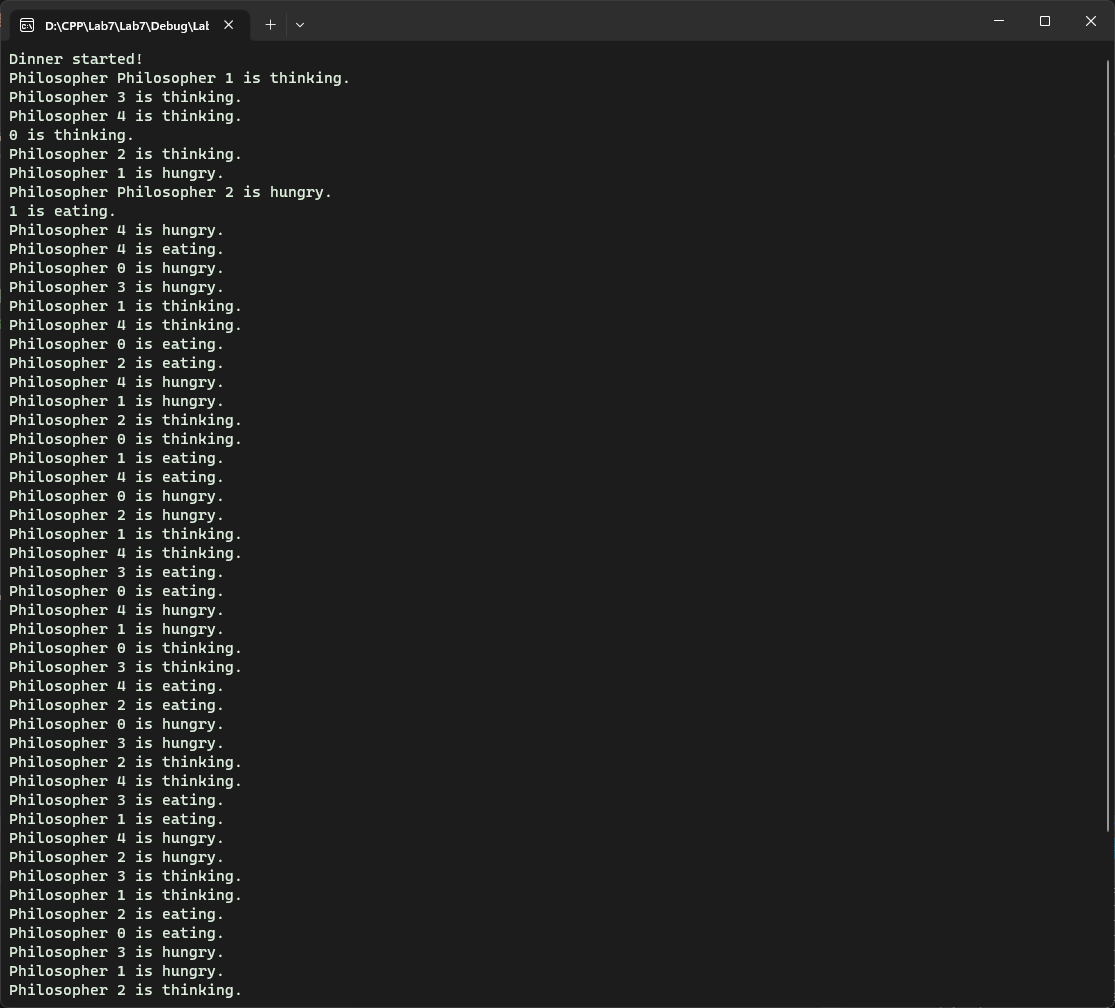
****

Рис. 1 – обед философов в случае, если время на раздумье меньше, чем на поглощение еды

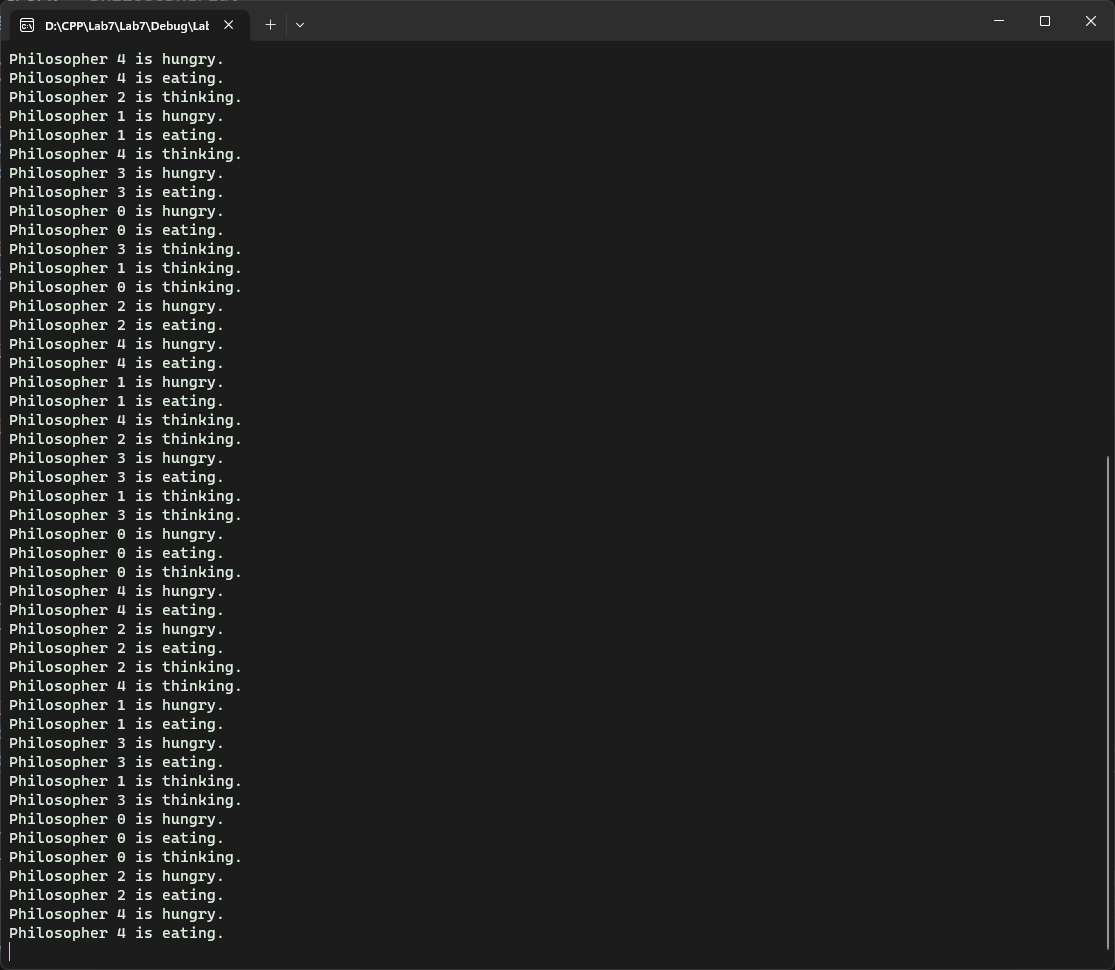
****

Рис. 2 – обед философов в случае, если время на раздумье больше, чем на поглощение еды

**Вывод:**

В результате данной лабораторной работы было создано многопоточное приложение, имитирующее ситуацию с пятью философами, сидящими за круглым столом. С использованием семафора `philSemaphore` и синхронизации потоков было достигнуто эффективное управление доступом к общим ресурсам (вилкам). Это позволило избежать проблем с взаимоблокировкой и голоданием философов, обеспечивая справедливое распределение возможности кушать.

Использование семафора в данной реализации оказалось целесообразным, так как он позволил ограничить количество философов, которые могут одновременно обедать, и гарантировал, что каждый философ будет иметь доступ к двум вилкам одновременно, избегая ситуаций, когда два философа могут одновременно взять одну и ту же вилку.

Главный поток успешно дожидался завершения работы всех потоков философов, используя метод `join`, что обеспечивает корректное завершение работы приложения и предотвращает возникновение неопределенного поведения.

Таким образом, с использованием семафора и синхронизации потоков было достигнуто успешное моделирование ситуации с философами и эффективное управление доступом к общим ресурсам, обеспечивая справедливое распределение шанса на кушанье и предотвращая проблемы с взаимоблокировкой и голоданием.