**55. Монитор, условные переменные. Мониторы Хоара и Меса.**

**Идея:** объединить механизм синхронизации и защищаемые данные.

**Монитор** – это конструкция языка программирования, поддерживающая управляемый доступ к разделяемым данным.

**Монитор инкапсулирует:**

* разделяемые критические данные
* функции, использующие разделяемые данные
* синхронизацию выполнения параллельных потоков, вызывающих указанные функции

**Свойства:**

* Доступ к данным, расположенным в мониторе, реализуется только посредством вызова предоставленных функций, код синхронизации добавляется компилятором.
* Монитор обеспечивает автоматическое взаимное исключение (достаточно для работы с критическими данными использовать только методы монитора)
  + только один поток может находиться в мониторе в любой момент времени
  + если второй поток пытается вызвать метод монитора, он переходит в состояние ожидания до выхода из монитора первого потока
* Мониторы во многих случаях проще в использовании, чем семафоры

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.**Схема:**

Автоматически добавлен код синхронизации. Он обеспечивает, что по отношению к каждому конкретному монитору в каждый момент времени функция будет выполнятся только одним потоком, все остальные потоки будут заблокированы на входе в монитор.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.**Проблема:**

Пусть решаем задачу «Производители-потребители» (Р – потоки-производители,   
С – потоки-потребители).

Предположим, что первым в монитор вошёл поток-потребитель и вызвал функцию «Потребить»:

* Заполненных записей нет (буфер пуст), он не может продолжить (и выйти)
* Монитор занят, производитель не может в него войти

Получили **взаимоблокировку (deadlock).**

**Решение** – концепция условной переменной.

**Условная переменная (condition variable, cv)** – символизирует некоторое событие, наступление которого может ожидать поток, выполняющий функцию монитора.

**Операции с условными переменными:**

* **wait(cv)** – блокирующая функция
  + снять блокировку монитора, после чего другой поток может войти в него
  + ожидать, пока какой-либо другой поток не подаст условный сигнал
* **signal(cv)**
  + пробудить максимум один поток, ожидающий наступления события
  + если нет ожидающих потоков, информация о сигнале теряется, т.е. выполнение операции может не изменить состояние ни одного объекта или потока
* **broadcast(cv)**
  + пробудить все потоки, ожидающие наступления события  
    (если таких потоков нет, то ничего не произойдёт, т.е. выполнение операции может не изменить состояние ни одного объекта или потока)

**Решение задачи «Производители-потребители»:**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

* Если поток не может выполнить операцию «Произвести/Потребить», он временно освобождает доступ к монитору
* Какой поток получит сигнал при его поступлении определяется реализацией и прикладному программисту неизвестно
* Способ обработки вызова wait(cv) зависит от вида монитора

**Проблема условных переменных:** условная переменная является таким объектом, смысл которой знает только программист (компилятор не может её сопоставить ни с чем). Также сложно проконтролировать правильность использования функций из набора: wait, signal, broadcast.

**Виды мониторов:** различия в обслуживании signal(cv)

1. **Мониторы Хоара:**

* **signal(cv)** означает:
  + выбирается и немедленно запускается ожидающий поток
  + поток, пославший сигнал, блокируется и остаётся блокированным всё время, пока выполняется поток, которого он вывел из состояния ожидания
  + пославший сигнал поток перед посылкой сигнала должен восстановить инварианты монитора (!) — он не должен оставлять ожидающему потоку монитор в некорректном состоянии
* (Явный «+»)Если поток разбужен, гарантируется выполнение условия, которого он ожидал
* **Обычно не реализуются** по причине сложности использования и неоднозначности реализации broadcast(cv)

1. Мониторы Меса:

* **signal(cv)** означает:
  + ожидающий поток переводится в состояние «готов к выполнению», а поток, пославший сигнал, продолжает исполнение
  + ожидавший поток может продолжить работу в мониторе при его освобождении, при этом он будет конкурировать за вход со всеми потоками, вызывающими функцию монитора или получившими сигнал
  + поток, пославший сигнал, может не восстанавливать состояние монитора вплоть до выхода из него
* Если поток вышел из состояния ожидания, это означает, что произошли какие-то изменения, при этом условие ожидания может не быть выполнено
* Обычно реализуются именно мониторы Меса