Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Операционные системы**

Студент: Козека Е. М.

ФИТ 3 курс 4 группа

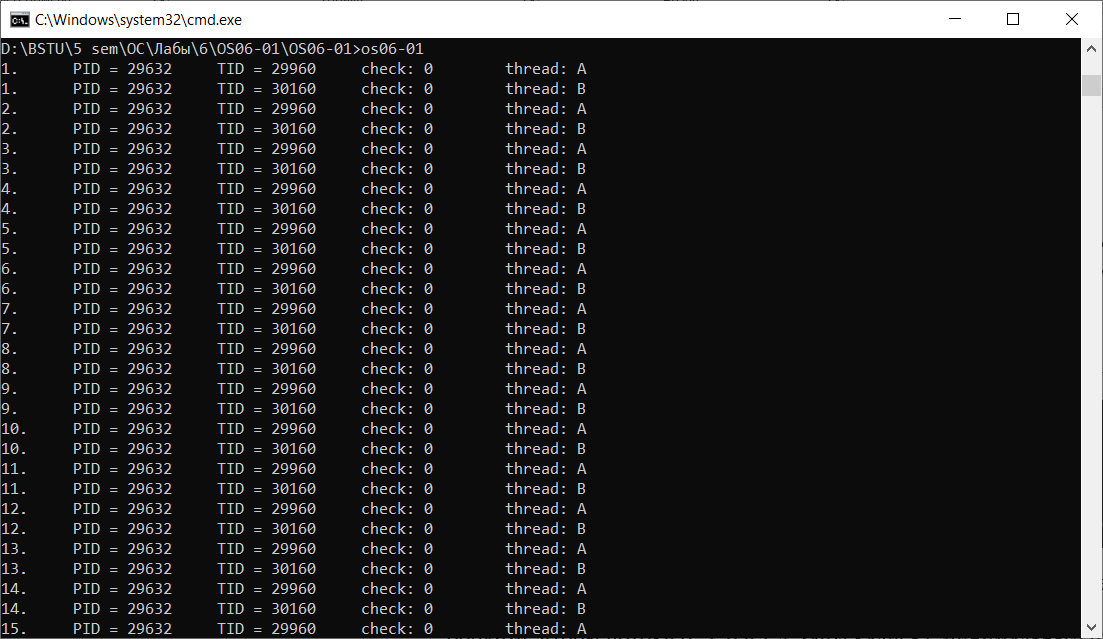
Преподаватель: Комкова А. В.

Минск 2024

**Лабораторная работа №6. Синхронизация**

**Задание 1.** Напишите ассемблерный код с применением команд BTS или BTR, демонстрирующий реализацию механизма синхронизации двух потоков одного процесса и поясните его работу.

Результат выполнения в консоли:



Переменная check используется для синхронизации потоков. В ней будет храниться битовая информация о том, находится ли поток в критической секции.

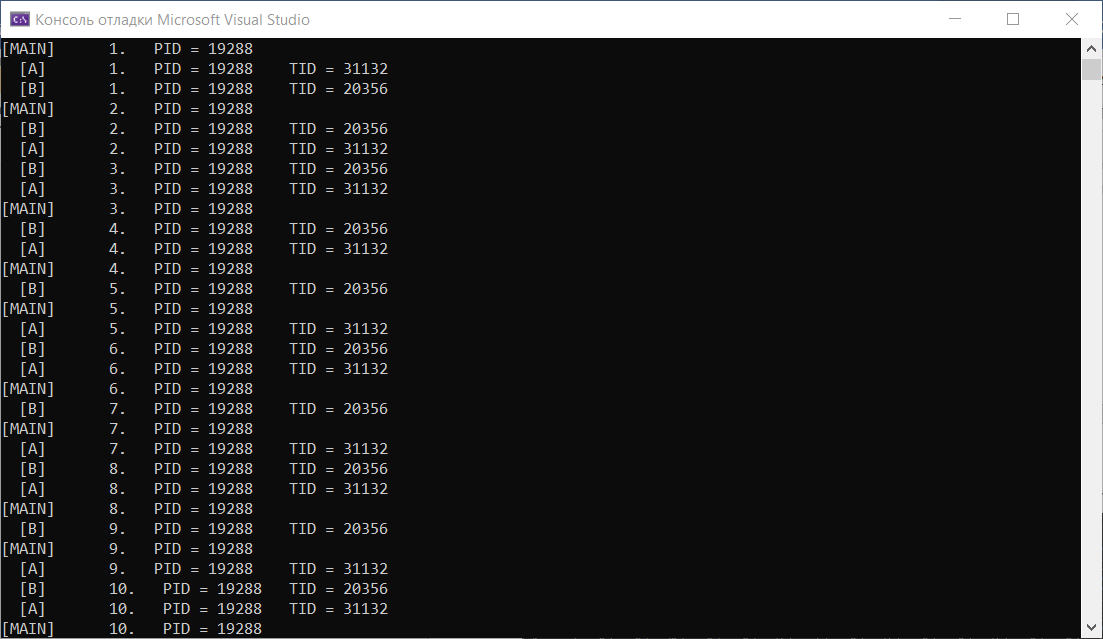
При входе в критическую сессию (EnterCriticalSectionAssembly) команда BTS устанавливает бит 0 переменной check. Если бит уже был установлен, выполнение переходит обратно к метке CriticalSection, и попытка повторяется.

При выходе из критической секции (LeaveCriticalSectionAssembly) команда BTR сбрасывает бит 0 переменной check.

Основной поток выполнения (loop) выводит информацию о себе, включая идентификаторы процесса и потока, а также текущее состояние check. Поток входит в критическую секцию при i == 30 и выходит при i == 60.

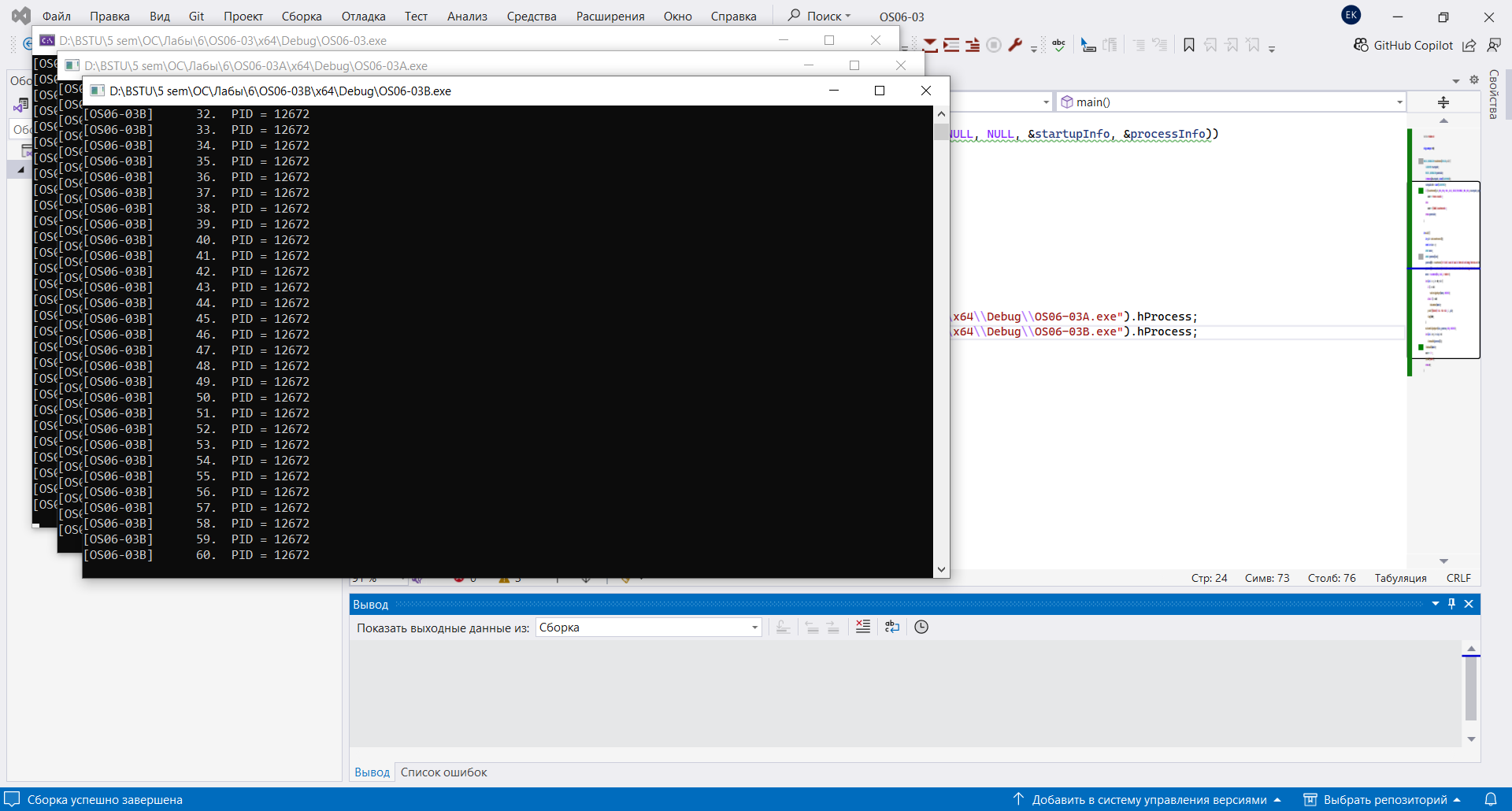
**Задание 2.** Разработайте приложение OS06\_02, запускающее два дочерних потока A и B. Все потоки выполняют циклы в 90 итераций, выводящие имена потоков и номера итерации с задержкой в 0.1 сек. Приложение OS06\_02 синхронизирует выполнение потоков main, A и B с помощью механизма critical section. Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60 в каждом потоке.

Результат выполнения в консоли:



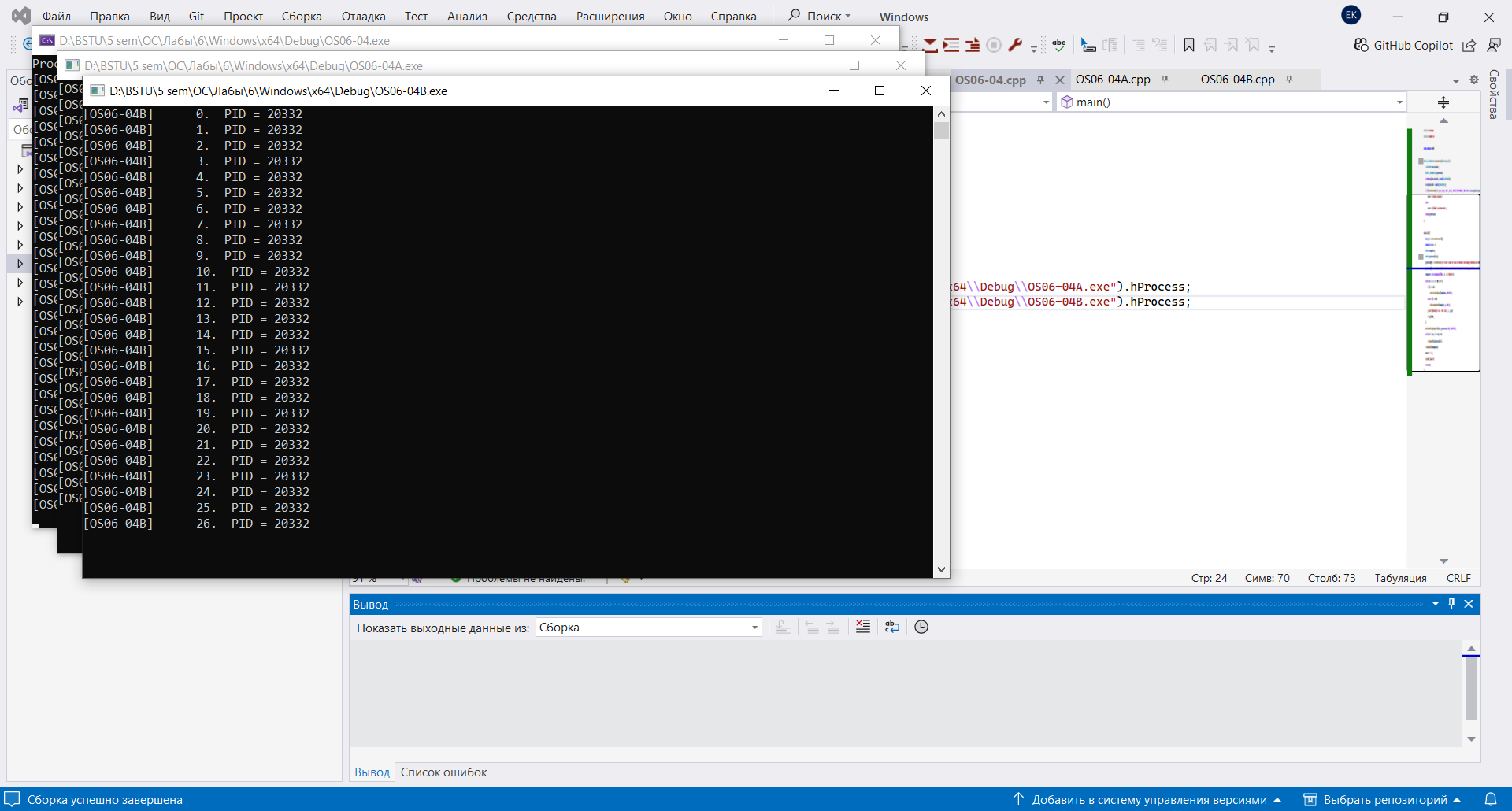
**Задание 3.** Разработайте приложение OS06\_03, запускающее два дочерних процесса OS06\_03A и OS06\_03B с выводом в отдельные консоли. Все процессы выполняют циклы в 90 итераций, выводящие имена процессов и номера итерации с задержкой в 0.1 сек. Приложение OS06\_03 синхронизирует выполнение процессов OS06\_03, OS06\_03A и OS06\_03B с помощью механизма mutex. Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60.

Результат выполнения в консоли:



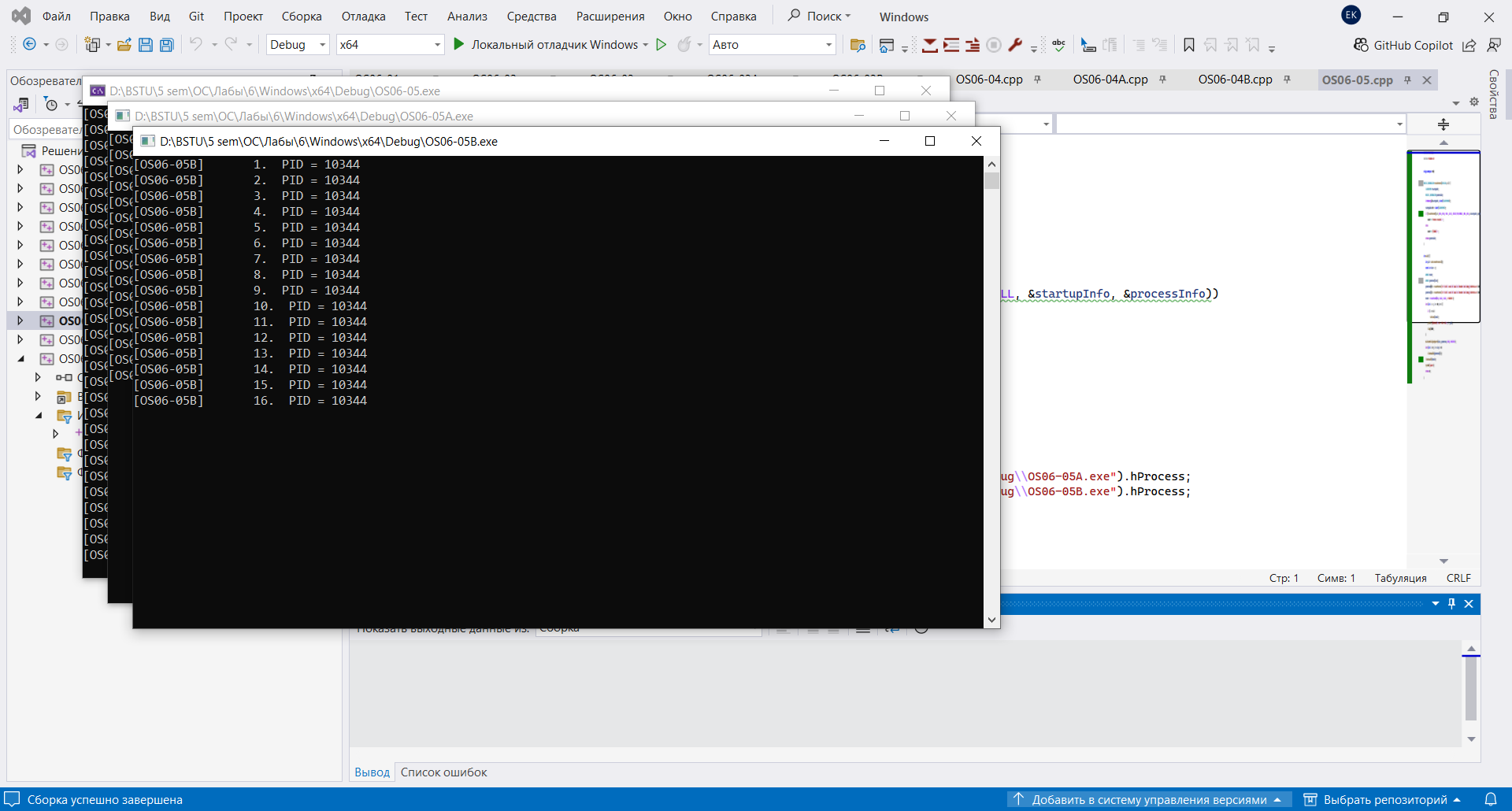
**Задание 4.** Разработайте приложение OS06\_04, запускающее два дочерних процесса OS06\_04A и OS06\_04B с выводом в отдельные консоли. Все процессы выполняют циклы в 90 итераций, выводящие имена процессов и номера итерации с задержкой в 0.1 сек. Приложение OS06\_04 синхронизирует выполнение процессов OS06\_04, OS06\_04A и OS06\_04B с помощью механизма semaphore. Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60 одного (любого) процесса и двух других процессов. Другими словами, итерации с 30 по 60 должны одновременно выполняться только в двух из трех процессов.

Результат выполнения в консоли:



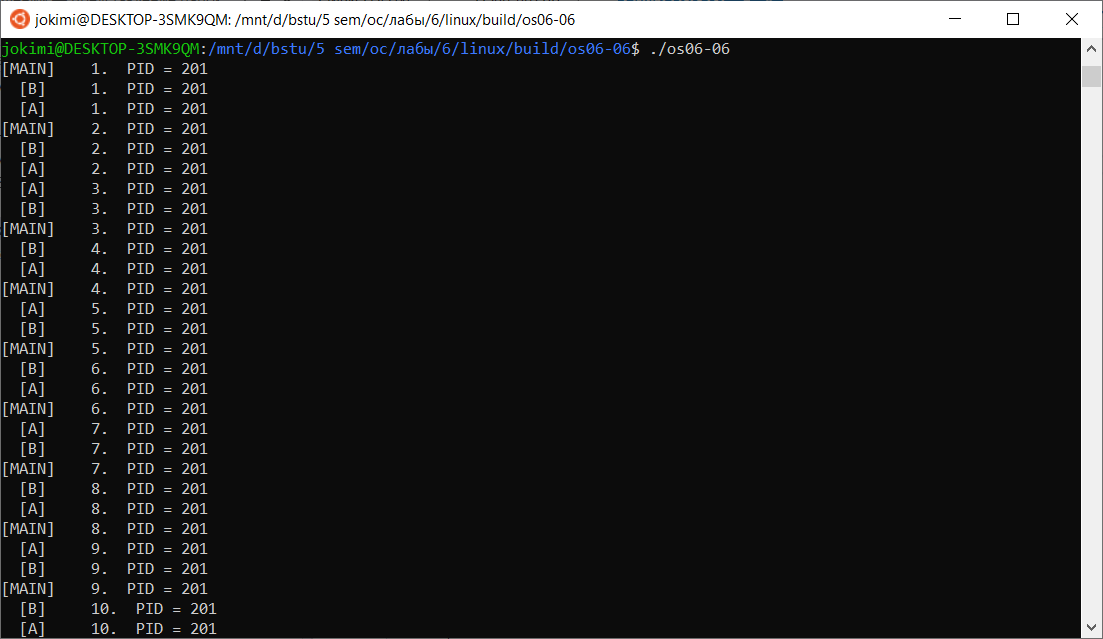
**Задание 5.** Разработайте приложение OS06\_05, запускающее два дочерних процесса OS06\_05A и OS06\_05B с выводом в отдельные консоли. Все процессы выполняют циклы в 90 итераций, выводящие имена процессов и номера итерации с задержкой в 0.1 сек. Приложение OS06\_05 синхронизирует выполнение процессов OS06\_05, OS06\_05A и OS06\_05B с помощью механизма event. Синхронизация должна обеспечивать выполнение приложения в два этапа: 1) выполнение итераций с 1 по 15 процесса OS06\_05; 2) одновременное выполнение всех трех процессов: OS06\_05 – продолжает выполнение итераций; процессы OS06\_05A и OS06\_05B выполняются начиная с первой итерации.

Результат выполнения в консоли:



**Задание 6.** Разработайте приложение OS06\_06, запускающее два дочерних потока A и B. Все потоки выполняют циклы в 90 итераций, выводящие имена потоков и номера итерации с задержкой в 0.1 сек. Приложение OS06\_06 синхронизирует выполнение потоков main, A и B с помощью механизма mutex. Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60 в каждом потоке.

Результат выполнения в консоли:



Ответы на вопросы:

**1. Дайте определение понятию «синхронизация потоков».**

Понятие «синхронизация потоков» описывает методы и механизмы, используемые для упорядочивания доступа к общим ресурсам или обмена информацией между потоками. Синхронизация потоков позволяет избежать гонок данных и других проблем, возникающих при параллельном выполнении кода.

**2. Объясните понятие «взаимная блокировка».**

«Взаимная блокировка» возникает, когда два или более потока находятся в состоянии ожидания ресурсов, которые контролируют другие потоки. Каждый из этих потоков блокирует ресурс, который требуется другому потоку для продолжения выполнения, при этом сам ожидает ресурс, контролируемый другим потоком. В результате ни один из потоков не может продолжить свое выполнение, поскольку он ожидает освобождения ресурса, которое не произойдет. Взаимная блокировка является нежелательным состоянием и может привести к замедлению или даже зависанию программы.

**3. Перечислите механизмы авторизации OS.**

**Парольная авторизация** (правильный пароль для доступа к системе или определенным ресурсам), **авторизация по ключу** (криптографический ключ для проверки подлинности и авторизации пользователя), **авторизация на основе сертификатов** (цифровой сертификат, который содержит информацию о его подлинности), **биометрическая авторизация** (физиологические или поведенческие характеристики пользователя, такие как отпечатки пальцев, сканирование сетчатки глаза или распознавание голоса), **авторизация на основе ролей** (доступ к определенным ресурсам в зависимости от своей роли или привилегий в системе)

**4. Поясните в чем разница между механизмом mutex и semaphore.**

Основная разница между mutex и semaphore заключается в их использовании. Mutex применяется для обеспечения взаимного исключения в отношении конкретного ресурса, в то время как semaphore используется для контроля доступа к определенному количеству ресурсов.

**5. Почему mutex, semaphore, event создают объект ядра OS, а critical section нет.**

Mutex, semaphore и event создают объекты ядра операционной системы потому, что они предоставляют механизмы синхронизации, которые могут использоваться между разными процессами или потоками в рамках операционной системы. Эти объекты ядра предоставляют надежные и масштабируемые механизмы синхронизации, поддерживаемые операционной системой.

Критическая секция представляет собой легковесный механизм синхронизации, который может использоваться только внутри одного процесса для синхронизации доступа к общим данным между потоками в этом процессе.