



**Международный Казахско-Турецкий  
Университет  
Имени Ходжа Ахмеда Яссауи  
Факультет «Инженерия»  
Кафедра Компьютерной Инженерий**

**срс**

**тема**

**Какие инструменты отладки  
доступны в визуальном  
программировании?**

**Готовил: Муратов Нодир  
Группа: ААЖ-114**

**кентау-2023**

**Breakpoints**

**Debugging Panel**

**Output Window**

**Logging**

**Event Tracking**



# Breakpoints

**Breakpoints (брейкпоинты) - это фундаментальное понятие в разработке программного обеспечения и отладке. Это метки или конкретные точки в вашем исходном коде, которые вы устанавливаете, чтобы приостановить выполнение программы во время отладки. Брейкпоинты используются для анализа состояния программы, пошагового выполнения кода, анализа переменных и данных в этой конкретной точке, чтобы выявить и устранить проблемы или ошибки.**



**Вот как работают брейкпоинты и их значение в процессе отладки:**

**Установка брейкпоинтов:** В интегрированных средах разработки (IDE) или редакторах кода вы можете устанавливать брейкпоинты, нажимая на номера строк или конкретные строки кода в ваших исходных файлах. Эти строки обычно отмечены специальным визуальным индикатором, например, красной точкой.

**Запуск программы:** Когда вы запускаете программу в режиме отладки, программа выполняется как обычно, пока не достигнет брейкпоинта.

**Приостановка выполнения:** Когда программа достигает брейкпоинта, она останавливается или приостанавливается, что позволяет вам изучить ее текущее состояние. Вы можете увидеть значения переменных, стек вызовов и другую информацию о выполнении.

**Пошаговое выполнение кода:** После приостановки на брейкпоинте вы можете выбрать пошаговое выполнение кода. Обычно есть опции для входа в функции или методы, перехода через строки и выхода из функций. Это помогает вам навигировать по коду и понять, как программа ведет себя.

**Инспекция переменных:** Во время нахождения на брейкпоинте вы можете проверить значения переменных, чтобы понять их текущее состояние и проверить, вызывают ли они проблемы.

**Условные брейкпоинты:** Некоторые инструменты отладки позволяют устанавливать условные брейкпоинты. Они останавливают выполнение программы только в случае, если выполнено определенное условие. Это полезно, когда вы хотите остановить выполнение только при выполнении определенных критериев.

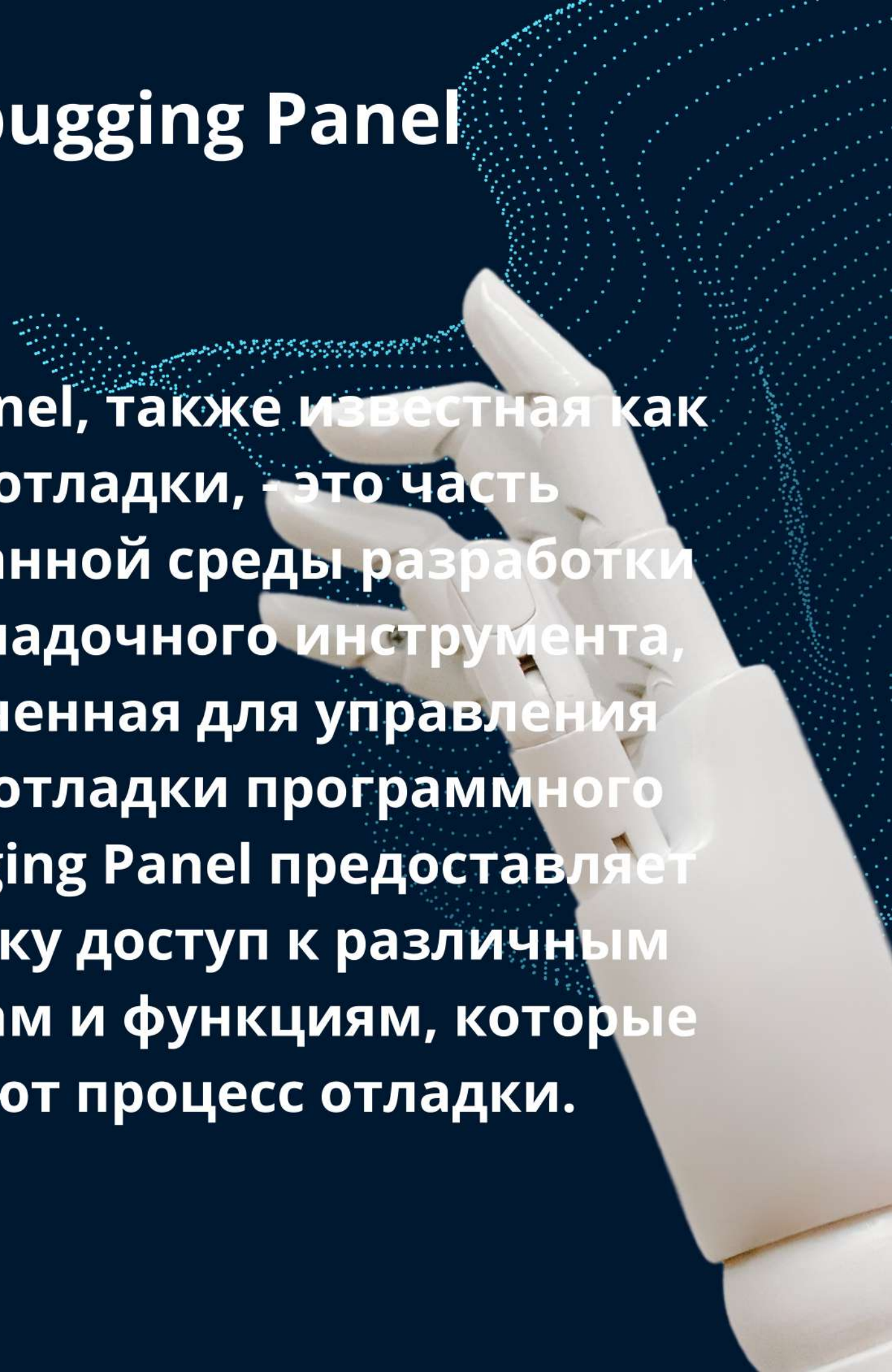
**Количество попаданий:** Вы также можете указать, сколько раз брейкпоинт должен быть достигнут, прежде чем программа остановится. Это полезно, когда вам нужно отлаживать определенную часть кода, которая выполняется несколько раз.

**Удаление или отключение брейкпоинтов:** Вы можете удалять или отключать брейкпоинты, когда они больше не нужны или вы хотите, чтобы программа выполнялась без прерываний.



# Debugging Panel

**Debugging Panel, также известная как панель отладки, - это часть интегрированной среды разработки (IDE) или отладочного инструмента, предназначенная для управления процессом отладки программного кода. Debugging Panel предоставляет разработчику доступ к различным инструментам и функциям, которые облегчают процесс отладки.**



**Обычно Debugging Panel содержит следующие элементы и функции:**

**Управление брейкпоинтами:** Вы можете устанавливать, удалять, активировать и деактивировать брейкпоинты в вашем коде с помощью Debugging Panel. Это позволяет вам контролировать точки останова в процессе отладки.

**Запуск и остановка отладки:** Debugging Panel предоставляет кнопки для запуска и остановки процесса отладки. Вы можете запускать программу в режиме отладки и приостанавливать ее выполнение по мере необходимости.

**Пошаговое выполнение:** Вы можете использовать Debugging Panel, чтобы выполнять код пошагово, одну инструкцию за другой, пока не достигнете нужной точки или брейкпоинта.

**Инспекция переменных:** Debugging Panel позволяет просматривать значения переменных и данных в текущем контексте выполнения программы. Это помогает вам понять, какие значения используются и какие проблемы могут возникнуть.

**Смотреть стек вызовов:** Вы можете просматривать стек вызовов, чтобы увидеть, какие функции или методы были вызваны и в каком порядке. Это полезно для понимания того, как программа выполняется и какие функции вызываются перед остановкой на брейкпоинте.

**Управление выполняемым потоком:** Некоторые Debugging Panel позволяют вам управлять потоками выполнения, изменяя их состояния, например, приостанавливая, возобновляя или завершая выполнение конкретных потоков.

**Debugging Panel является важной частью инструментов разработки, потому что оно облегчает процесс отладки, делает его более управляемым и позволяет разработчикам быстрее находить и устранять ошибки в своем коде.**



# OUTPUT WINDOW

**Output Window (Окно вывода) - это элемент интегрированных сред разработки (IDE) или других средств разработки программного обеспечения, предназначенный для отображения различных видов вывода и сообщений, которые связаны с процессом разработки и выполнения кода. Output Window обычно содержит текстовый или многострочный интерфейс, в котором разработчики исследуют, анализируют и отслеживают информацию, связанную с их проектами.**

**В Output Window могут отображаться следующие виды информации:**

**Сообщения об ошибках и предупреждения:** Когда компилятор или интерпретатор обнаруживает ошибки в коде, он выводит сообщения об ошибках и предупреждения в Output Window. Это помогает разработчикам быстро идентифицировать и исправлять проблемы в коде.

**Вывод отладки:** Вывод отладки, такой как значения переменных, стек вызовов и другие отладочные сообщения, могут быть направлены в Output Window во время отладки программы. Это помогает разработчикам понимать, как именно работает их код и какие данные передаются.

**Сообщения о выполнении программы:** Output Window может содержать сообщения о ходе выполнения программы, такие как информация о процессе загрузки и выполнения программы, а также любые другие события, связанные с выполнением кода.

**Сообщения о сборке проекта:** При сборке проекта, компилятор и среда разработки могут выводить информацию о процессе сборки, включая успешное завершение или ошибки, которые могли возникнуть.

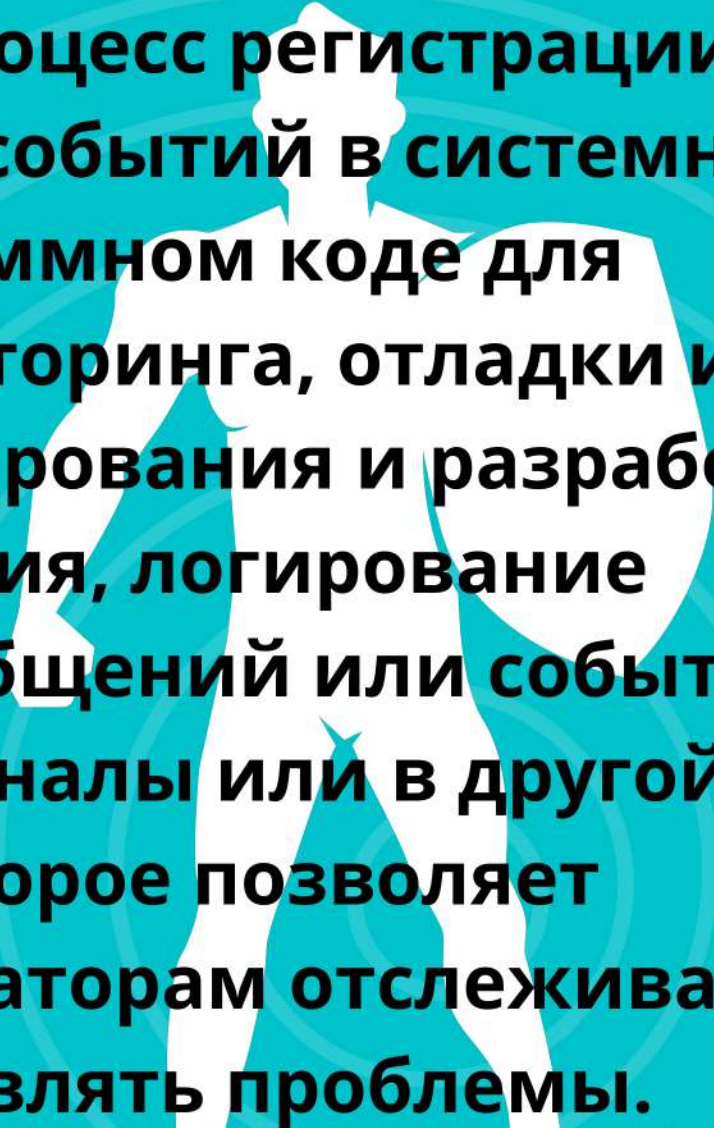
**Пользовательский вывод:** Разработчики также могут использовать Output Window для вывода пользовательской информации, такой как текстовые сообщения, отладочные журналы или результаты специфических операций, чтобы легче отслеживать выполнение программы.

**Output Window - полезный инструмент в процессе разработки, так как он предоставляет централизованный способ мониторинга и анализа информации, связанной с проектом. Он помогает разработчикам быстро реагировать на ошибки, следить за выполнением программы и делать выводы на основе вывода и сообщений, которые генерируются в процессе разработки.**



# Logging

**Логирование (logging) - это процесс регистрации и сохранения информации или событий в системном, прикладном или программном коде для последующего анализа, мониторинга, отладки или аудита. В контексте программирования и разработки программного обеспечения, логирование представляет собой запись сообщений или событий в специальные файлы, в журналы или в другой хранилище данных, которое позволяет разработчикам и администраторам отслеживать работу программы и выявлять проблемы.**



## Основные аспекты логирования включают:

**События и сообщения:** Логирование может включать в себя запись различных событий, сообщений, ошибок, предупреждений и другой информации, связанной с работой программы. Это может включать в себя информацию о запросах в веб-приложениях, ошибки в коде, события в приложениях, такие как вход пользователя, выполнение задач и многое другое.

**Уровни логирования:** Чтобы обеспечить баланс между объемом информации и читаемостью логов, часто используются различные уровни логирования, такие как отладка (debug), информация (info), предупреждение (warning), ошибка (error), фатальная ошибка (fatal), и т. д. Разработчики могут выбирать, какой уровень логирования использовать для каждого сообщения в зависимости от его важности.

**Системы логирования:** Для логирования в программировании часто используются специальные библиотеки и системы логирования, такие как log4j, log4net, Python's logging, и другие. Эти системы предоставляют инструменты и API для записи, фильтрации, форматирования и вывода логов.

**Место хранения логов:** Логи могут быть сохранены в различных местах, включая текстовые файлы, базы данных, системные журналы операционных систем и удаленные серверы. Выбор места зависит от конкретных требований и целей.



**Логирование имеет множество применений, таких как:**

**Отладка:** Логи позволяют разработчикам отслеживать состояние программы и искать ошибки в коде.

**Мониторинг и анализ:** Логи могут использоваться для мониторинга производительности и анализа поведения приложения в продакшене.

**Аудит:** В системах безопасности логи могут использоваться для аудита действий пользователей и поиска аномалий.

**Сбор статистики:** Логи позволяют собирать данные о использовании приложения, которые могут быть полезными для принятия бизнес-решений.

**Логирование является важной практикой в разработке программного обеспечения и системном администрировании, так как оно помогает поддерживать качество приложений, упрощает поиск и решение проблем и позволяет отслеживать работу программ в различных средах.**

# EVENT TRACKING

Event tracking (слежение за событиями) - это процесс сбора и анализа информации о событиях, которые происходят в приложениях, веб-сайтах, системах и других контекстах. Эти события могут быть разнообразными и включать в себя действия пользователей, операции в приложении, события системы и многое другое. Слежение за событиями важно для понимания поведения пользователей, анализа производительности и определения, каким образом используется приложение или система.



**Веб-аналитика:** Веб-сайты используют слежение за событиями для отслеживания действий пользователей, таких как клики, просмотры страниц, отправка форм, взаимодействие с элементами на странице и другие события. Это позволяет веб-мастерам и маркетологам анализировать поведение пользователей и улучшать пользовательский опыт.

**Мобильные приложения:** Мобильные приложения также используют слежение за событиями для отслеживания действий пользователей, таких как нажатия на кнопки, переходы между экранами, использование функций приложения и т. д.

**Информационные системы:** В корпоративных информационных системах и базах данных слежение за событиями может включать в себя мониторинг действий пользователей, изменения данных, создание и удаление записей и другие операции, которые могут быть важными для безопасности и аудита.

**Игровая индустрия:** В играх слежение за событиями позволяет разработчикам отслеживать, какие уровни проходят игроки, какие достижения они разблокируют, какие предметы они используют и другие игровые события.

**Системы мониторинга и аналитики:** Системы мониторинга сетей и серверов также используют слежение за событиями для отслеживания различных событий, таких как сбои в работе, события безопасности и производительность.

Слежение за событиями часто реализуется с использованием специальных инструментов и библиотек, которые позволяют собирать данные о событиях, а затем анализировать эти данные для выявления тенденций, проблем или возможностей для улучшения. Эта информация может быть весьма ценной для принятия решений в различных областях, включая маркетинг, разработку приложений, безопасность и мониторинг.