**Знакомство с отладчиком Visual Studio**

* Статья
* 09.12.2021
* Чтение занимает 5 мин
  + [](https://github.com/Mikejo5000)
  + [](https://github.com/olprod)
  + [](https://github.com/iamqinmu)
  + [](https://github.com/OpenLocalizationService)

Были ли сведения на этой странице полезными?

В этом разделе представлены средства отладчика, предоставляемые Visual Studio. В контексте Visual Studio *отладка приложения* обычно означает запуск приложения с подключенным отладчиком (то есть в режиме отладчика). При этом в отладчике доступно множество способов наблюдения за выполнением кода. Вы можете пошагово перемещаться по коду и просматривать значения, хранящиеся в переменных, задавать контрольные значения для переменных, чтобы отслеживать изменение значений, изучать путь выполнения кода и т. д. Если вы не знакомы с процессом отладки, перед выполнением задач в этом разделе рекомендуется прочесть документ об [отладке для начинающих](https://docs.microsoft.com/ru-ru/visualstudio/debugger/debugging-absolute-beginners?view=vs-2022).

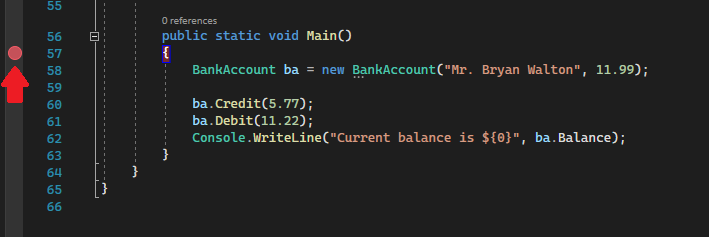
Описанные здесь функции применяются к C#, C++, Visual Basic, JavaScript и другим языкам, поддерживаемым Visual Studio (если не указано иное).

**Установка точки останова и запуск отладчика**

Точки останова полезны, если вам известны строка или раздел кода, которые вы хотите подробно изучить в среде выполнения. Дополнительные сведения о различных типах точек останова, которые можно задать, например об условных точках останова и точках останова в функциях, см. в разделе [Использование точек останова](https://docs.microsoft.com/ru-ru/visualstudio/debugger/using-breakpoints?view=vs-2022).

Для отладки нужно запустить приложение с отладчиком, подключенным к процессу приложения. Для этого чаще всего используется клавиша **F5** (**Отладка > Начать отладку**). Однако сейчас у вас, возможно, не задано ни одной точки останова для проверки кода приложения, поэтому мы сначала зададим их, а затем начнем отладку. Точки останова — это один из самых простых и важных компонентов надежной отладки. Точка останова указывает, где Visual Studio следует приостановить выполнение кода, чтобы вы могли проверить значения переменных или поведение памяти либо выполнение ветви кода.

Если вы открыли файл в редакторе кода, точку останова можно задать, щелкнув в поле слева от строки кода.

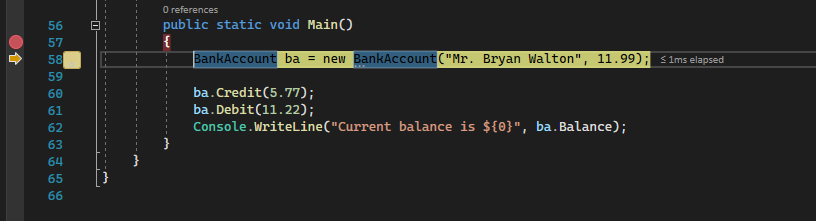


Нажмите клавишу **F5** (**Отладка > Начать отладку**) или **Начать отладку** кнопку Начать отладку на панели инструментов отладки, при этом отладчик выполняется до первой встреченной точки останова. Если приложение еще не запущено, при нажатии клавиши F5 запускается отладчик и выполняется остановка в первой точке останова.

**Переход по коду в отладчике с помощью пошаговых команд**

Мы указываем сочетания клавиш для большинства команд, так как они ускоряют навигацию по коду вашего приложения. (Аналогичные команды, такие как команды меню, отображаются в круглых скобках.) Дополнительные сведения об использовании команд пошагового выполнения см. в разделе [Навигация по коду в отладчике](https://docs.microsoft.com/ru-ru/visualstudio/debugger/navigating-through-code-with-the-debugger?view=vs-2022).

Для запуска приложения с подключенным отладчиком нажмите клавишу **F11** (**Отладка > Шаг с заходом**). F11 — это команда **Шаг с заходом**, которая выполняет приложение с переходом к следующему оператору. При запуске приложения с помощью клавиши F11 отладчик останавливается на первом выполняемом операторе.



Желтая стрелка представляет оператор, на котором приостановлен отладчик. В этой же точке приостанавливается выполнение приложения (этот оператор пока не выполнен).

Клавишу F11 удобно использовать для более детальной проверки потока выполнения. (Мы также покажем другие варианты более быстрого перемещения по коду.) По умолчанию отладчик пропускает непользовательский код (дополнительные сведения см. в статье об [отладке в режиме "Только мой код"](https://docs.microsoft.com/ru-ru/visualstudio/debugger/just-my-code?view=vs-2022)).

**Примечание**

В управляемом коде вы увидите диалоговое окно с запросом о том, хотите ли вы получать уведомления при автоматическом обходе свойств и операторов (поведение по умолчанию). Если вы хотите изменить этот параметр позже, отключите параметр **Шаг с обходом свойств и операторов** в меню **Сервис > Параметры** в разделе **Отладка**.

**Шаг с обходом по коду для пропуска функций**

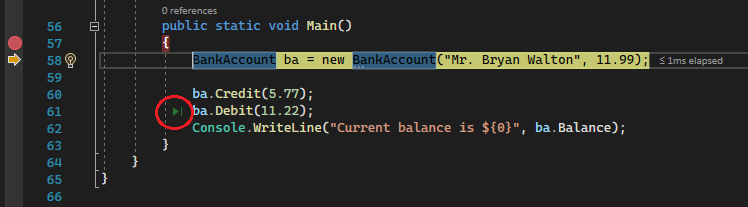
Когда вы находитесь на строке кода, представляющей собой вызов функции или метода, можно нажать клавишу **F10** (**Отладка > Шаг с обходом**) вместо F11.

Клавиша F10 продолжает выполнение отладчика без захода в функции или методы в коде приложения (код продолжает выполняться). Нажав клавишу F10, вы можете обойти код, который вас не интересует. Так можно быстро перейти к важному для вас коду. Дополнительные сведения об использовании команд пошагового выполнения см. в разделе [Навигация по коду в отладчике](https://docs.microsoft.com/ru-ru/visualstudio/debugger/navigating-through-code-with-the-debugger?view=vs-2022).

**Быстрое выполнение до точки в коде с помощью мыши**

Использование кнопки **Выполнение до щелкнутого** аналогично установке временной точки останова. Кроме того, эта команда удобна для быстрой работы в видимой области кода приложения. **Выполнение до щелкнутого** можно использовать в любом открытом файле. Дополнительные сведения об этой функции и аналогичных функциях навигации см. в разделе [Выполнение до определенного места в коде](https://docs.microsoft.com/ru-ru/visualstudio/debugger/navigating-through-code-with-the-debugger?view=vs-2022#run-to-a-specific-location-or-function).

Находясь в отладчике, наведите курсор на строку кода, пока слева не появится кнопка **выполнения до щелкнутого** (Выполнить до этого места) Снимок экрана: кнопка выполнения до щелкнутого в отладчике Visual Studio. Кнопка указывает на то, что выполнение должно происходить до строки, в которой размещена кнопка. отображается слева.



**Примечание**

Кнопка **выполнения до щелкнутого** (Выполнить до этого места) доступна начиная с версии Visual Studio 2017.

Нажмите кнопку **выполнения до щелкнутого** (Выполнить до этого места). Отладчик продолжает выполнение до строки кода, которую вы щелкнули.

**Вывод отладчика из текущей функции**

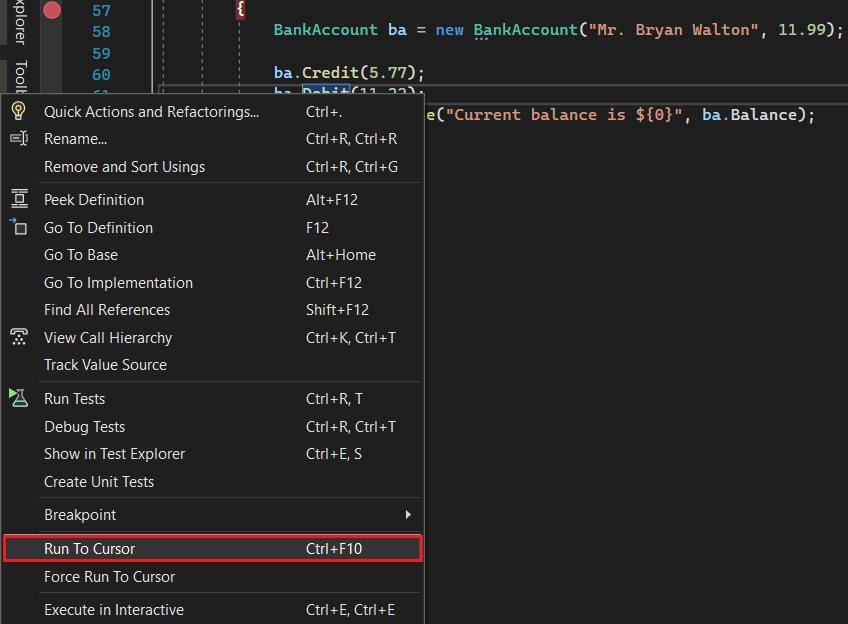
В некоторых случаях может потребоваться продолжить сеанс отладки, однако полностью проведя отладчик сквозь текущую функцию.

Нажмите клавиши **SHIFT+F11** (или выберите **Отладка > Шаг с выходом**).

Эта команда возобновляет выполнение приложения (и перемещает отладчик) до возврата текущей функции.

**Выполнить до текущей позиции**

Если вы находитесь в режиме редактирования кода (то есть работа отладчика не приостановлена), щелкните правой кнопкой мыши строку кода в приложении и выберите команду **Выполнить до текущей позиции** (или нажмите клавиши **CTRL**+**F10**). Эта команда запускает отладку и задает временную точку останова на текущей строке кода. Дополнительные сведения об этой функции и аналогичных функциях навигации см. в разделе [Выполнение до определенного места в коде](https://docs.microsoft.com/ru-ru/visualstudio/debugger/navigating-through-code-with-the-debugger?view=vs-2022#run-to-a-specific-location-or-function).



Если имеются заданные точки останова, отладчик приостанавливается в первой достигнутой точке останова.

Нажимайте клавишу **F5**, пока не достигнете строки кода, для которой выбрали **Выполнить до текущей позиции**.

Эта команда удобна, когда вы редактируете код и хотите быстро задать временную точку останова и одновременно запустить отладчик.

**Примечание**

Вы можете использовать функцию **Выполнить до текущей позиции** в окне **Стек вызовов** во время отладки.

**Быстрый перезапуск приложения**

Нажмите кнопку **Перезапустить** Перезапустить приложение на панели инструментов отладки (или нажмите клавиши **CTRL+SHIFT+F5**).

Кнопка **Перезапустить** позволяет сэкономить время, затрачиваемое на остановку приложения и перезапуск отладчика. Отладчик приостанавливается в первой точке останова, достигнутой при выполнении кода.

Если вы хотите остановить отладчик и вернуться в редактор кода, вы можете нажать красную кнопку Остановить отладку вместо **Перезапустить**.

**Редактирование кода в реальном времени**

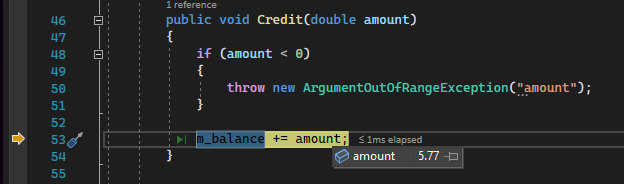
Visual Studio 2022 поддерживает динамическое редактирование кода в процессе отладки. Дополнительные сведения см. в следующих разделах.

* [Создание и отладка выполняющегося кода](https://docs.microsoft.com/ru-ru/visualstudio/debugger/hot-reload?view=vs-2022)
* [Создание и отладка выполняющегося кода XAML с помощью Горячей перезагрузки XAML](https://docs.microsoft.com/ru-ru/visualstudio/xaml-tools/xaml-hot-reload?view=vs-2022)
* [Изменить и продолжить](https://docs.microsoft.com/ru-ru/visualstudio/debugger/edit-and-continue?view=vs-2022)

**Проверка переменных с помощью подсказок по данным**

Теперь, когда вы немного освоились, у вас есть хорошая возможность проверить состояние приложения (переменные) с помощью отладчика. Функции, позволяющие проверять переменные, являются одними из самых полезных в отладчике. Реализовывать эту задачу можно разными способами. Часто при попытке выполнить отладку проблемы пользователь старается выяснить, хранятся ли в переменных значения, которые требуются в определенном состоянии приложения. Подробные сведения об использовании подсказок по данным см. в разделе [Просмотр значений данных в подсказках по данным](https://docs.microsoft.com/ru-ru/visualstudio/debugger/view-data-values-in-data-tips-in-the-code-editor?view=vs-2022).

В режиме приостановки в отладчике наведите указатель мыши на объект, чтобы увидеть его текущее значение или значение по умолчанию.



Если переменная имеет свойства, объект можно развернуть, чтобы увидеть все его свойства.

Часто при отладке бывает необходимо быстро проверить значения свойств для объектов. Лучше всего для этого подходят подсказки по данным.

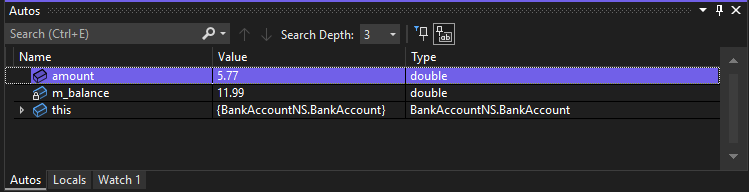
**Совет**

В большинстве поддерживаемых языков можно изменять код во время сеанса отладки. Дополнительные сведения см. в разделе [**Изменить и продолжить**](https://docs.microsoft.com/ru-ru/visualstudio/debugger/edit-and-continue?view=vs-2022).

**Проверка переменных с помощью окон "Видимые" и "Локальные"**

В окне **Видимые** отображаются переменные вместе с текущим значением и типом. Окно **Видимые** содержит все переменные, используемые в текущей строке или предыдущей строке (в C++ в окне отображаются переменные в трех предыдущих строках кода; сведения о зависящем от языка поведении см. в соответствующей документации). Дополнительные сведения об использовании этих окон см. в статье [Проверка переменных в окнах "Видимые" и "Локальные"](https://docs.microsoft.com/ru-ru/visualstudio/debugger/autos-and-locals-windows?view=vs-2022).

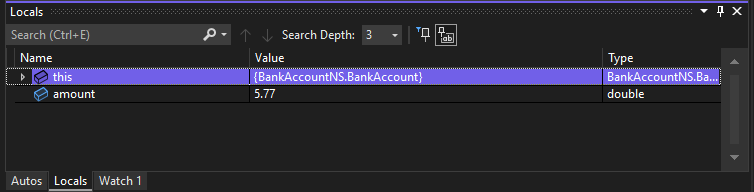
Во время отладки взгляните на окно **Видимые** в нижней части редактора кода.



**Примечание**

В JavaScript окно **Локальные** поддерживается, а окно **Видимые** — нет.

Взгляните в окно **Локальные**. В окне **Локальные** показаны переменные, которые находятся в текущей области.

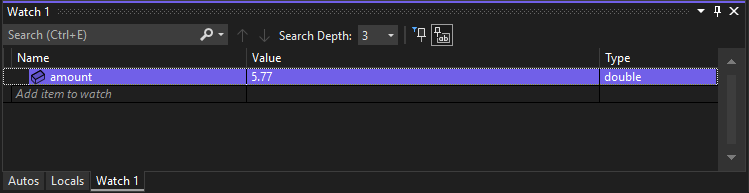


В этом примере объекты this и f находятся в области действия. Дополнительные сведения см. в статье [Проверка переменных в окнах "Видимые" и "Локальные"](https://docs.microsoft.com/ru-ru/visualstudio/debugger/autos-and-locals-windows?view=vs-2022).

**Установка контрольного значения**

В окне **Контрольное значение** можно указать переменную (или выражение), которую необходимо отслеживать. Дополнительные сведения см. в статье [Установка контрольных значений с помощью окон "Контрольное значение" и "Быстрая проверка"](https://docs.microsoft.com/ru-ru/visualstudio/debugger/watch-and-quickwatch-windows?view=vs-2022).

Во время отладки щелкните правой объект кнопкой мыши и выберите пункт **Добавить контрольное значение**.



В этом примере у вас есть контрольное значение, заданное для объекта, и по мере перемещения по отладчику вы можете наблюдать за изменением его значения. В отличие от других окон переменных, в окне **Контрольное значение** всегда отображаются просматриваемые вами переменные (они выделяются серым цветом, когда находятся вне области действия).

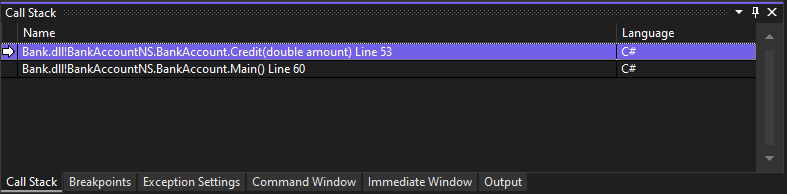
**Просмотр стека вызовов**

В окне **Стек вызовов** показан порядок вызова методов и функций. В верхней строке показана текущая функция. Во второй строке показана функция или свойство, из которого она вызывалась, и т. д. Стек вызовов хорошо подходит для изучения и анализа потока выполнения приложения. Дополнительные сведения см. в статье о [просмотре стека вызовов](https://docs.microsoft.com/ru-ru/visualstudio/debugger/how-to-use-the-call-stack-window?view=vs-2022).

**Примечание**

Окно **Стек вызовов** аналогично перспективе "Отладка" в некоторых интегрированных средах разработки, например Eclipse.

Во время отладки щелкните окно **Стек вызовов**, которое по умолчанию открыто в нижней правой области.

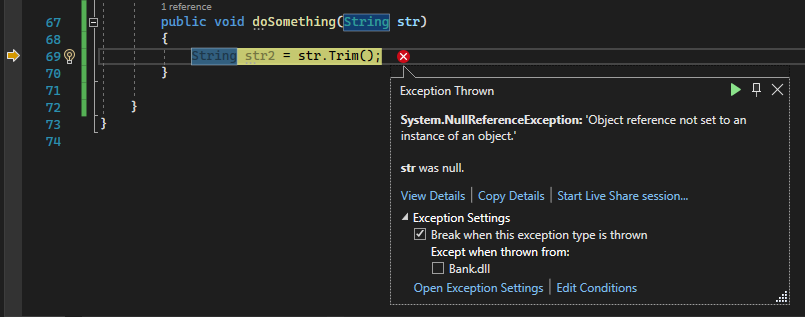


Дважды щелкните строку кода, чтобы просмотреть исходный код. При этом также изменится текущая область, проверяемая отладчиком. Это не перемещает отладчик.

Для выполнения других задач можно воспользоваться контекстными меню из окна **Стек вызовов**. Например, можно вставлять точки останова в указанные функции, перезапускать приложение с помощью функции **Выполнение до текущей позиции** и изучать исходный код.

**Проверка исключения**

Когда приложение выдает исключение, отладчик переходит к строке кода, вызвавшей исключение. Подробные сведения см. в разделе [Проверка исключения с помощью помощника по исправлению ошибок](https://docs.microsoft.com/ru-ru/visualstudio/debugger/exception-helper?view=vs-2022).



В этом примере **Помощник по исправлению ошибок** отображает исключение System.NullReferenceException и ошибку с сообщением о том, что для экземпляра объекта не задана ссылка на объект. Также он сообщает, что строковое значение имело значение NULL при попытке вызвать метод Trim.

Помощник по исправлению ошибок — это отличная функция, которая помогает отлаживать ошибки. Используя помощник по исправлению ошибок, вы также можете, например, просмотреть сведения об ошибке и добавить контрольное значение. При необходимости вы также можете изменить условия для возникновения конкретного исключения. См. дополнительные сведения об обработке исключений в коде в разделе [Приемы и инструменты отладки](https://docs.microsoft.com/ru-ru/visualstudio/debugger/write-better-code-with-visual-studio?view=vs-2022).

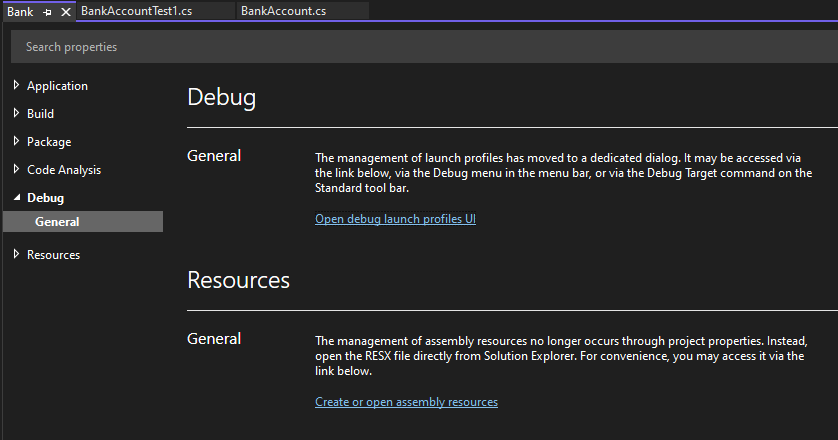
Разверните узел **Параметры исключений**, чтобы просмотреть дополнительные параметры для обработки исключения этого типа, однако в рамках этого тура ничего менять не требуется.

**Настройка отладки**

Вы можете настроить проект для сборки, как описано в статье о [конфигурации отладки или выпуска](https://docs.microsoft.com/ru-ru/visualstudio/debugger/how-to-set-debug-and-release-configurations?view=vs-2022), настроить свойства проекта для отладки или настроить [общие параметры](https://docs.microsoft.com/ru-ru/visualstudio/debugger/how-to-specify-debugger-settings?view=vs-2022) для отладки. Кроме того, можно настроить отладчик для вывода пользовательских сведений с помощью таких компонентов, как атрибута [DebuggerDisplay](https://docs.microsoft.com/ru-ru/visualstudio/debugger/using-the-debuggerdisplay-attribute?view=vs-2022) либо [платформы NatVis](https://docs.microsoft.com/ru-ru/visualstudio/debugger/create-custom-views-of-native-objects?view=vs-2022) для C/C++.

Свойства отладки зависят от типа проекта. Например, можно задать аргумент, который будет передан в приложение при запуске. Чтобы получить доступ к свойствам проекта приложения, щелкните имя проекта правой кнопкой мыши в обозревателе решений и выберите **Свойства**. Свойства отладки обычно отображаются на вкладке **Сборка** или **Отладка** в зависимости от типа проекта.

Начиная с Visual Studio 2022, вкладка **Отладка** для проектов .NET содержит ссылку на пользовательский интерфейс профилей запуска отладки, где можно задать свойства, связанные с отладкой.



**Использование точек останова в отладчике Visual Studio**

* Статья
* 14.12.2021
* Чтение занимает 6 мин
  + [](https://github.com/Mikejo5000)
  + [](https://github.com/olprod)
  + [](https://github.com/iamqinmu)
  + [](https://github.com/OpenLocalizationService)
  + [](https://github.com/yishengjin1413)

Были ли сведения на этой странице полезными?

Точки останова — это один из важнейших инструментов разработчика для выполнения отладки. Точки останова устанавливаются везде, где нужно приостановить выполнение отладчика. Например, вы хотите просмотреть состояние переменных кода или стек вызовов в определенной точке останова. Если вы пытаетесь разрешить предупреждение или проблему при использовании точек останова, см. раздел [Устранение неполадок точек останова в отладчике Visual Studio](https://docs.microsoft.com/ru-ru/visualstudio/debugger/troubleshooting-breakpoints?view=vs-2022).

**Примечание**

Если вам известна задача или проблема, которую вы пытаетесь решить, но вы не знаете, какой тип точки останова следует использовать, см. статью [**Часто задаваемые вопросы. Поиск нужной функции отладки в Visual Studio**](https://docs.microsoft.com/ru-ru/visualstudio/debugger/find-your-debugging-task?view=vs-2022#pause-running-code).

**Задание точек останова в исходном коде**

Можно установить точку останова в любой строке исполняемого кода. Например, в следующем коде C# можно задать точку останова в строке кода с присвоением значения переменной (int testInt = 1), в цикле for или любом коде в цикле for. Точка останова не может быть задана в сигнатурах метода, объявлениях пространства имен или класса, а также в объявлениях переменной, не содержащих присваивания и методов задания или получения.

Чтобы задать точку останова в исходном коде, щелкните в крайнем левом поле рядом со строкой кода. Можно также выбрать строку и нажать клавишу **F9**, выбрать команду **Отладка** > **Переключить точку останова** или щелкнуть правой кнопкой мыши и выбрать команду **Точка останова** > **Вставить точку останова**. В месте установки точки останова в левом поле появится красная точка.

Для большинства языков, включая C#, точка останова и текущие строки выполнения выделяются автоматически. Для кода C++ можно включить выделение точек останова и текущих строк, выбрав **Сервис** (или **Отладка**) > **Параметры** > **Отладка** > **Выделить всю исходную строку для точек останова и текущий оператор (только C++)** .

При отладке выполнение приостанавливается в точке останова до выполнения кода в этой строке. Символ точки останова отображается с желтой стрелкой.

В точке останова в следующем примере значение testInt по-прежнему равно 1. Таким образом, значение не изменилось с момента инициализации переменной (значение 1), так как инструкция, выделенная желтым, еще не была выполнена.

Когда отладчик останавливается в точке останова, можно просмотреть текущее состояние приложения, включая [значения переменных](https://docs.microsoft.com/ru-ru/visualstudio/debugger/debugger-feature-tour?view=vs-2022#inspect-variables-with-data-tips) и [стека вызова](https://docs.microsoft.com/ru-ru/visualstudio/debugger/how-to-use-the-call-stack-window?view=vs-2022).

Ниже приведены некоторые общие инструкции по работе с точками останова.

* Точка останова представляет собой переключатель. Можно щелкнуть ее, нажать клавишу **F9** или выбрать **Отладка** > **Переключить точку останова**, чтобы удалить или снова вставить ее.
* Чтобы отключить точку останова, не удаляя ее, наведите указатель мыши на нее или щелкните ее правой кнопкой мыши и выберите **Отключить точку останова**. Отключенные точки останова отображаются как пустые точки в левом поле или окне **Точки останова**. Чтобы повторно включить точку останова, наведите указатель мыши на нее или щелкните ее правой кнопкой мыши и выберите **Включить точку останова**.
* Задайте условия и действия, добавьте и измените метки или экспортируйте точку останова, щелкнув ее правой кнопкой мыши и выбрав нужную команду или наведя на нее курсор и нажав на значок **Параметры**.

**Действия точки останова и точки трассировки**

*Точка трассировки* — это точка останова, которая выводит сообщение в окно **вывода**. Точка трассировки может играть роль временного оператора трассировки в языке программирования и не приостанавливает выполнение кода. Вы создаете точку трассировки, задавая особое действие в окне **Параметры точки останова**. Подробные инструкции см. в разделе [Использование точек трассировки в отладчике Visual Studio](https://docs.microsoft.com/ru-ru/visualstudio/debugger/using-tracepoints?view=vs-2022).

**Условия точки останова**

Можно управлять тем, где и когда выполняется точка останова, задавая условия. Условие может быть любым допустимым выражением, которое распознает отладчик. Дополнительные сведения о допустимых выражениях см. в разделе [Выражения в отладчике](https://docs.microsoft.com/ru-ru/visualstudio/debugger/expressions-in-the-debugger?view=vs-2022).

**Задание условия для точки останова.**

1. Щелкните правой кнопкой мыши символ точки останова и выберите пункт **Условия** (или нажмите клавиши **ALT** + **F9**, **C**). Или наведите курсор на символ точки останова, выберите значок **Параметры**, а затем выберите **Условия** в окне **Параметры точки останова**.

Вы также можете щелкнуть правой кнопкой мыши в дальнем левом поле рядом со строкой кода и выбрать **Вставить условную точку останова** в контекстном меню.

Можно также задать условия в окне **Точки останова**, щелкнув правой кнопкой мыши точку останова и выбрав пункт **Параметры**, а затем **Условия**.

1. В раскрывающемся списке выберите **Условное выражение**, **Количество обращений** или **Фильтр** и задайте соответствующее значение.
2. Выберите **Закрыть** или нажмите клавиши **CTRL**+**ВВОД**, чтобы закрыть окно **Параметры точки останова**. Или в окне **Точки останова** выберите **ОК**, чтобы закрыть диалоговое окно.

Точки останова с установленными условиями отображаются с символом **+** в исходном коде и окне **Точки останова**.

**Создание условного выражения**

Если вы выбрали **Условное выражение**, можно выбрать одно из двух условий: **Имеет значение true** или **При изменении**. Выберите значение **Имеет значение true**, чтобы прервать выполнение при истинности выражения, или значение **При изменении**, если требуется прервать выполнение при изменении значения выражения.

В следующем примере задается применение точки останова, только если значение testInt равно **4**:

В следующем примере задается применение точки останова, только если значение testInt меняется:

Если условие точки останова имеет недопустимый синтаксис, появится предупреждающее сообщение. Если указать условие для точки останова с недопустимой семантикой, но допустимым синтаксисом, предупреждающее сообщение появится при достижении точки останова в первый раз. В любом случае отладчик прерывает выполнение при попадании на недопустимую точку останова. Точка останова пропускается, только если условие допустимо и принимает значение false.

**Примечание**

Для поля **При изменении** отладчик не обрабатывает первое вычисление условия как изменение, поэтому при первом вычислении выражения точка останова не сработает.

**Использование идентификаторов объектов в условных выражениях (только C# и F#)**

Иногда возникает необходимость наблюдать за поведением определенного объекта. Например, может потребоваться выяснить, почему объект был вставлен в коллекцию несколько раз. В C# и F# можно создавать идентификаторы объектов для определенных экземпляров [ссылочных типов](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/reference-types) и использовать их в условиях точек останова. Идентификатор объекта создается службами отладки среды CLR и связан с объектом.

**Создание идентификатора объекта**

1. Установите точку останова в коде после создания объекта.
2. Запустите отладку и, когда выполнение приостановится на точке останова, выберите **Отладка** > **Windows** > **Локальные** (или нажмите клавиши **CTRL** + **ALT** + **V**, **L**), чтобы открыть окно **Локальные**.

Найдите конкретный экземпляр объекта в окне **Локальные**, щелкните его правой кнопкой мыши и выберите **Сделать идентификатор объекта**.

В окне **$** вы должны увидеть символ **Локальные** . Это и есть идентификатор объекта.

1. Добавьте новую точку останова в точке, которую нужно исследовать, например в точке добавления объекта в коллекцию. Щелкните правой кнопкой мыши точку останова и выберите пункт **Условие**.
2. Используйте идентификатор объекта в поле **Условное выражение**. Например, если переменная item является объектом, который должен быть добавлен в коллекцию, выберите **Имеет значение true** и введите **item == $<n>** , где <n> — номер идентификатора объекта.

Выполнение прервется в точке добавления объекта в коллекцию.

Чтобы удалить идентификатор объекта, щелкните правой кнопкой мыши переменную в окне **Локальные** и выберите пункт **Удалить идентификатор объекта**.

**Примечание**

Идентификаторы объектов создают слабые ссылки и не предотвращают сборку мусора для объекта. Они действительны только в рамках текущего сеанса отладки.

**Установка условия количества обращений**

Если есть подозрение, что цикл в коде начинает неправильно вести себя после определенного числа итераций, можно установить точку останова для остановки выполнения после указанного количества обращений, вместо того чтобы многократно нажимать клавишу **F5** для достижения этой итерации.

В разделе **Условия** в окне **Параметры точки останова** выберите **Количество обращений**, а затем укажите число итераций. В следующем примере задается выполнение точки останова при каждой второй итерации:

**Задание условия фильтра**

Вы можете ограничить точку останова, сделав ее срабатывание возможным лишь на определенных устройствах или в определенных процессах или потоках.

В разделе **Условия** в окне **Параметры точки останова** выберите **Фильтр**, а затем введите одно или несколько из следующих выражений.

* MachineName = "имя"
* ProcessId = значение
* ProcessName = "имя"
* ThreadId = значение
* ThreadName = "имя"

Значения строк следует заключить в двойные кавычки. Для комбинации условий можно использовать знаки & (И), || (ИЛИ), ! (НЕ) и скобки.

**Задание точек останова функции**

Выполнение можно прерывать при вызове функции. Это полезно, например, если известно имя функции, но не ее расположение. Это также полезно, если у вас есть функции с одинаковым именем и вы хотите приостановить их все (например, перегруженные функции или функции в разных проектах).

**Установка точки останова функции:**

1. Выберите **Отладка** > **Создать точку останова** > **Точка останова функции** или нажмите клавиши **CTRL** + **K**, **B**.

Можно также выбрать **Создать** > **Точка останова функции** в окне **Точки останова**.

1. В диалоговом окне **Новая точка останова функции** заполните поле **Имя функции**.

Чтобы уточнить функцию, сделайте следующее.

* + Используйте полное имя функции.

Пример: Namespace1.ClassX.MethodA()

* + Добавьте типы параметров перегруженной функции.

Пример: MethodA(int, string)

* + Используйте символ "!", чтобы указать модуль.

Пример: App1.dll!MethodA

* + Используйте оператор контекста в машинном коде C++.

{function, , [module]} [+<line offset from start of method>]

Пример: {MethodA, , App1.dll}+2

1. В раскрывающемся списке **Язык** выберите язык функции.
2. Нажмите кнопку **ОК**.

**Задание точки останова функции по адресу памяти (только в машинном коде C++)**

Вы можете использовать адрес объекта, чтобы установить точку останова функции в методе, вызываемом для конкретного экземпляра класса. Например, если имеется адресуемый объект типа my\_class, вы можете задать точку останова функции в методе my\_method, вызываемом экземпляром.

1. Установите точку останова после создания экземпляра класса.
2. Найдите адрес экземпляра (например, 0xcccccccc).
3. Выберите **Отладка** > **Создать точку останова** > **Точка останова функции** или нажмите клавиши **CTRL** + **K**, **B**.
4. В поле **Имя функции** добавьте следующий текст и выберите язык **C++** .

C++Копировать

((my\_class \*) 0xcccccccc)->my\_method

**Задание точек останова в данных (.NET Core 3.x или .NET 5 или более поздних версий)**

Точки останова в данных прерывают выполнение при изменении свойства конкретного объекта.

**Задание точки останова в данных**

1. В проекте .NET Core начните отладку и подождите, пока не будет достигнута точка останова.
2. В окне **Видимые**, **Контрольные значения** или **Локальные** щелкните правой кнопкой мыши свойство и выберите **Прервать выполнение при изменении значения** в контекстном меню.

Точки останова в данных в .NET Core не будут работать для следующих элементов:

* Свойства, которые не могут быть расширены в подсказках, окнах "Локальные", "Видимые" или "Контрольные значения"
* Статические переменные
* Классы с атрибутом DebuggerTypeProxy
* Поля внутри структур

**Задание точек останова в данных (только машинный код C++)**

Точки останова в данных приостанавливают выполнение, когда изменяется значение, хранящееся в указанном адресе памяти. Если значение только считывается, но не изменяется, выполнение программы не прерывается.

**Задание точки останова в данных**

1. В проекте C++ начните отладку и подождите, пока не будет достигнута точка останова. В меню **Отладка** выберите команду **Создать точку останова** > **Точка останова в данных**.

Можно также выбрать **Создать** > **Точка останова в данных** в окне **Точки останова** или щелкнуть правой кнопкой мыши элемент **Видимые**, **Контрольные значения** или **Локальные** и выбрать **Прервать выполнение при изменении значения** в контекстном меню.

1. В поле **Адрес** введите адрес памяти или выражение, результатом вычисления которого будет адрес памяти. Например, для прерывания при изменении содержимого переменной &avar введите avar .
2. В раскрывающемся списке **Счетчик байтов** введите количество байтов, за которыми должен наблюдать отладчик. Например, если выбрать **4**, отладчик будет наблюдать за четырьмя байтами начиная с &avar и выполнит прерывание, если какой-либо из этих байтов изменит значение.

Точки останова в данных не работают при следующих условиях.

* Процесс, для которого не выполняется отладка, производит запись в область памяти.
* Область памяти совместно используется двумя или более процессами.
* Область памяти обновляется в ядре. Например, если память передается в функцию ReadFile 32-разрядной версии Windows, то память будет обновляться из режима ядра и отладчик не будет прерывать выполнение при обновлении.
* Контрольное значение превышает 4 байта на 32-разрядном оборудовании и 8 байт на 64-разрядном оборудовании. Это ограничение архитектуры x86.

**Примечание**

* Точки останова в данных зависят от конкретных адресов памяти. Адрес переменной меняется между сеансами отладки, поэтому точки останова в данных автоматически отключаются в конце каждого сеанса отладки.
* Если установить точку останова по данным на локальную переменную, точка останова остается включенной при завершении выполнения функции, но адрес памяти больше не применяется, поэтому поведение точки останова будет непредсказуемым. Если установить точку останова в данных на локальную переменную, рекомендуется удалить или отключить точку останова до окончания функции.

**Добавление зависимой точки останова**

Зависимые точки останова могут сработать только при условии срабатывания другой точки останова. Поэтому в сложном сценарии, например при отладке многопоточного приложения, вы можете настроить дополнительные точки останова после срабатывания определенной точки останова. Это упрощает отладку кода по часто используемым путям, например для игрового цикла или служебного API, так как точки останова в таких функциях можно настроить для включения только в случае вызова функции в определенной части приложения.

**Чтобы задать зависимую точку останова, сделайте следующее:**

1. Наведите указатель мыши на символ точки останова, щелкните значок **Параметры**, а затем выберите **Включить только при срабатывании следующей точки останова** в окне "Параметры точек останова".
2. В раскрывающемся списке выберите необходимую точку останова, от которой будет зависеть текущая точка останова.

Выберите **Закрыть** или нажмите клавиши **CTRL+ВВОД**, чтобы закрыть окно "Параметры точки останова". Или в окне "Точки останова" выберите **ОК**, чтобы закрыть диалоговое окно.

Чтобы задать зависимую точку останова, вы также можете воспользоваться контекстным меню по щелчку правой кнопкой мыши.

1. Щелкните правой кнопкой мыши в крайнем левом поле рядом со строкой кода и выберите в контекстном меню пункт **Вставить зависимую точку останова**.

* Зависимые точки останова не работают, если в приложении присутствует только одна точка останова.
* Зависимые точки останова преобразуются в обычные точки останова, если предварительная условная точка останова была удалена.

**Добавление временной точки останова**

Такая точка останова позволяет прервать выполнение кода только единожды. Отладчик Visual Studio приостанавливает выполняющееся приложение только один раз для этой точки, а после срабатывания немедленно удаляет ее.

**Чтобы задать временную точку останова, сделайте следующее:**

1. Наведите указатель мыши на символ точки останова, щелкните значок **Параметры**, а затем выберите **Удалить точку останова после срабатывания** в окне "Параметры точек останова".
2. Выберите **Закрыть** или нажмите клавиши **CTRL+ВВОД**, чтобы закрыть окно "Параметры точки останова". Или в окне "Точки останова" выберите **ОК**, чтобы закрыть диалоговое окно.

Чтобы задать временную точку останова, вы также можете воспользоваться контекстным меню по щелчку правой кнопкой мыши.

1. Щелкните правой кнопкой мыши в крайнем левом поле рядом со строкой кода и выберите в контекстном меню пункт **Вставить временную точку останова**.

Или просто воспользуйтесь клавишами **F9+SHIFT+ALT** и установите временную точку останова в нужной строке.

**Управление точками останова в окне "Точки останова"**

Используйте окно **Точки останова** для просмотра всех точек останова в вашем решении и управления ими. Это централизованное расположение особенно полезно в больших решениях или сложных сценариях отладки, в которых точки останова являются критически важными.

В окне **Точки останова** можно выполнять поиск, сортировку, фильтрацию, включение, отключение или удаление точек останова. Можно также задать условия и действия или добавить новую функцию или точку останова в данных.

Чтобы открыть окно **Точки останова**, выберите **Отладка** > **Windows** > **Точки останова** или нажмите клавиши **CTRL**+**ALT**+**B**.

Чтобы выбрать столбцы для отображения в окне **Точки останова**, выберите **Показать столбцы**. Выберите заголовок столбца, чтобы отсортировать список точек останова по этому столбцу.

**Метки точки останова**

Используйте метки для сортировки и фильтрации списка точек останова в окне **Точки останова**.

1. Чтобы добавить метку к точке останова, щелкните правой кнопкой мыши точку останова в исходном коде или в окне **Точки останова**, а затем выберите **Изменить метки**. Добавьте новую метку или выберите существующую, а затем выберите **ОК**.
2. Отсортируйте список точек останова в окне **Точки останова**, выбрав **Метки**, **Условия** или другие заголовки столбцов. Можно выбрать отображаемые столбцы, нажав **Показать столбцы** на панели инструментов.

**Экспорт и импорт точек останова**

Чтобы сохранить или предоставить состояние и расположение точек останова, их можно экспортировать или импортировать.

* Чтобы экспортировать одну точку останова в XML-файл, щелкните правой кнопкой мыши точку останова в исходном коде или окне **Точки останова** и выберите **Экспорт** или **Экспорт выбранных**. Выберите расположение экспорта и щелкните **Сохранить**. Расположением по умолчанию является папка решения.
* Чтобы экспортировать несколько точек останова, в окне **Точки останова** установите флажки рядом с точками останова или введите условия поиска в поле **поиска**. Нажмите на значок **Экспортировать все точки останова, соответствующие текущим критериям поиска** и сохраните файл.
* Чтобы экспортировать все точки останова, снимите все флажки и оставьте поле **поиска** пустым. Нажмите на значок **Экспортировать все точки останова, соответствующие текущим критериям поиска** и сохраните файл.
* Чтобы импортировать точки останова, в окне **Точки останова** выберите значок **Импортировать точки останова из файла**, перейдите к расположению XML-файла и нажмите **Открыть**.

**Установка точек останова из окон отладчика**

Можно также задать точки останова из окон отладчика **Стек вызовов** и **Дизассемблирование**.

**Задание точки останова в окне стека вызовов**

Чтобы прервать выполнение на инструкции или строке, к которой возвращается вызывающая функция, установите соответствующую точку останова в окне **Стек вызовов**.

**Задание точки останова в окне стека вызовов**

1. Чтобы открыть окно **Стек вызовов**, необходимо приостановить процесс отладки. Выберите **Отладка** > **Windows** > **Стек вызовов** или нажмите клавиши **CTRL**+**ALT**+**C**.
2. В окне **Стек вызовов** щелкните правой кнопкой мыши вызывающую функцию и выберите **Точка останова** > **Вставить точку останова** или нажмите **F9**.

В левом поле стека вызовов рядом с именем вызова функции появится символ точки останова.

В окне **Точки останова** точка останова стека вызова будет представлена как адрес с областью памяти, который соответствует следующей исполняемой инструкции в функции.

Отладчик приостанавливает выполнение на этой инструкции.

Более подробную информацию о стеке вызовов см. в разделе [Практическое руководство. Использование окна стека вызова](https://docs.microsoft.com/ru-ru/visualstudio/debugger/how-to-use-the-call-stack-window?view=vs-2022).

Сведения о визуальном отслеживании точек останова во время выполнения кода см. в разделе [Сопоставление методов в стеке вызовов при отладке](https://docs.microsoft.com/ru-ru/visualstudio/debugger/map-methods-on-the-call-stack-while-debugging-in-visual-studio?view=vs-2022).

**Задание точки останова в окне дизассемблирования**

1. Чтобы открыть окно **Дизассемблирование**, необходимо приостановить процесс отладки. Выберите **Отладка** > **Windows** > **Дизассемблирование** или нажмите клавиши **CTRL**+**ALT**+**D**.
2. В окне **Дизассемблирование** щелкните в левом поле инструкции, на которой нужно прервать выполнение. Можно также выбрать строку и нажать клавишу **F9** или щелкнуть правой кнопкой мыши и выбрать **Точка останова** > **Вставить точку останова**.