

# Учебник. Сведения об отладке кода C++ с помощью Visual Studio

Статья • 04.01.2023 • Чтение занимает 9 мин

Область применения:  Visual Studio  Visual Studio для Mac  Visual Studio Code

В этом пошаговом руководстве рассматриваются возможности отладчика Visual Studio. Более полное описание функций отладчика см. в статье с [Знакомство с отладчиком Visual Studio](#). *Отладка приложения* обычно означает запуск и выполнение приложения с подключенным отладчиком. 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, просматривать выполнение ветви кода и т. д. Если вы не знакомы с процессом отладки, перед выполнением задач в этой статье рекомендуется прочесть документ об [отладке для начинающих](#).


Несмотря на то, что демонстрационное приложение написано на C++, большинство функций применимы к C#, Visual Basic, F#, Python, JavaScript и другим языкам, поддерживаемым Visual Studio (F# не поддерживает возможность "Изменить и продолжить". F# и JavaScript не поддерживают окно **Видимые**). Снимки экрана приведены для C++.

В этом руководстве рассмотрены следующие задачи:

- ✓ Запуск отладчика и попадание в точки останова.
- ✓ Использование команд для пошагового выполнения кода в отладчике.
- ✓ Проверка переменных в подсказках к данным и окнах отладчика.
- ✓ Просмотр стека вызовов

## Предварительные требования

У вас должна быть установлена среда Visual Studio и рабочая нагрузка **Разработка классических приложений на C++**.

Установите Visual Studio 2022 бесплатно со страницы [скачиваемых материалов Visual Studio 2022](#) , если еще не сделали этого.

Если вам нужно установить рабочую нагрузку, но вы уже используете Visual Studio, выберите пункт **Средства > Получить средства и компоненты...**, после чего

запустится Visual Studio Installer. Запускается Visual Studio Installer. Выберите рабочую нагрузку **Разработка классических приложений на C++**, а затем нажмите **Изменить**.

## Создание проекта

Сначала вы создадите проект консольного приложения на C++. Для этого типа проекта уже имеются все нужные файлы шаблонов, что избавляет вас от лишней работы.

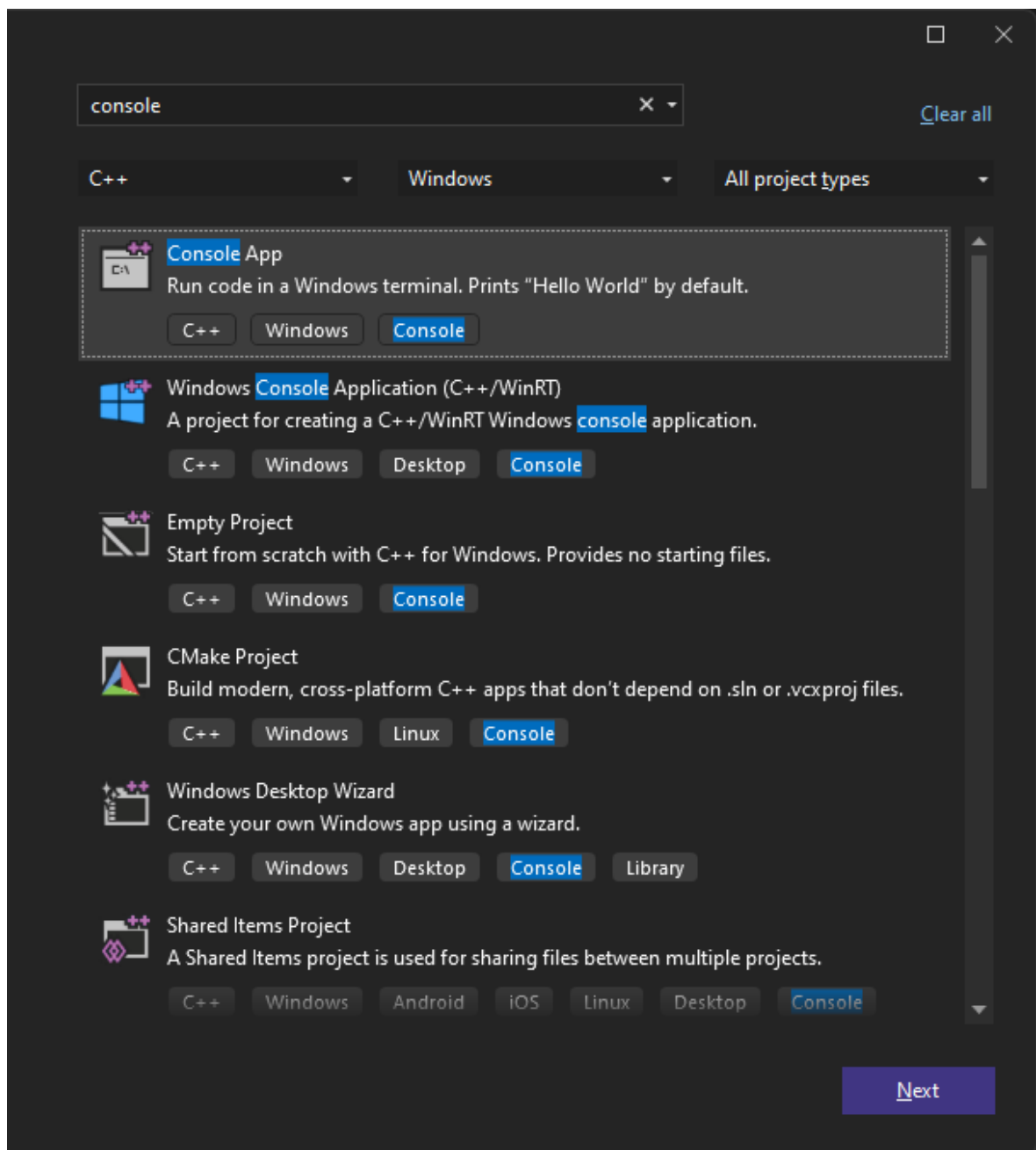
1. Запустите Visual Studio.

Если окно запуска не открыто, выберите **Файл > Окно запуска**.

2. На начальном экране выберите **Создать проект**.

3. В поле поиска окна **Создание проекта** введите *консоль*. Затем выберите **C++** в списке языков и **Windows** в списке платформ.

Применив фильтры языка и платформ, выберите шаблон **Консольное приложение** и нажмите кнопку **Далее**.



#### ⚠ Примечание

Если шаблон **Консольное приложение** отсутствует, его можно установить из окна **Создание проекта**. В сообщении **Не нашли то, что искали?** выберите ссылку **Установка других средств и компонентов**. После этого в Visual Studio Installer выберите рабочую нагрузку **Разработка классических приложений на C++**.

4. В поле **Имя проекта** окна **Настроить новый проект** введите *get-started-debugging*. Затем нажмите **Создать**.

Новый проект открывается в Visual Studio.

# Создание приложения

1. Откройте файл *get-started-debugging.cpp* и замените все его содержимое по умолчанию следующим кодом:

```
C++


#include <string>
#include <vector>
#include <iostream>

void SendMessage(const std::wstring& name, int msg)
{
    std::wcout << L"Hello, " << name << L"! Count to " << msg <<
    std::endl;
}

int main()
{
    std::vector<wchar_t> letters = { L'f', L'r', L'e', L'd', L' ',
    L's', L'm', L'i', L't', L'h' };
    std::wstring name = L"";
    std::vector<int> a(10);
    std::wstring key = L"";

    for (int i = 0; i < letters.size(); i++)
    {
        name += letters[i];
        a[i] = i + 1;
        SendMessage(name, a[i]);
    }
    std::wcin >> key;
    return 0;
}
```

## Запуск отладчика


1. Нажмите клавишу F5 (Отладка > Начать отладку) или кнопку Начать отладку  на панели инструментов отладки.

При нажатии клавиши F5 происходит запуск приложения с присоединенным отладчиком. Но пока мы не сделали ничего особенного, чтобы проанализировать код. Поэтому приложение будет просто загружено, и вы увидите выходные данные консоли.

```
cmd
```

```
Hello, f! Count to 1  
Hello, fr! Count to 2  
Hello, fre! Count to 3  
Hello, fred! Count to 4  
Hello, fred ! Count to 5  
Hello, fred s! Count to 6  
Hello, fred sm! Count to 7  
Hello, fred smi! Count to 8  
Hello, fred smit! Count to 9  
Hello, fred smith! Count to 10
```

В этом руководстве мы более подробно рассмотрим приложение с отладчиком и познакомимся с возможностями отладчика.

2. Остановите отладчик, нажав красную кнопку остановки  или клавиши **SHIFT + F5**.
3. В окне консоли нажмите клавишу **ВВОД**, чтобы закрыть его.


## Установка точки останова и запуск отладчика

1. В цикле `for` функции `main` установите точку останова, щелкнув левое поле следующей строки кода:

```
name += letters[i];
```

В месте установки точки останова появится красный круг .

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.

2. Нажмите клавишу **F5** или кнопку **Начать отладку** . Откроется приложение, и отладчик перейдет к строке кода, где задана точка останова.

```
11 int main()
12 {
13     std::vector<wchar_t> letters = { L'f', L'n', L'e', L'd', L' ', L's', L'm', L'i', L't', L'h' };
14     std::wstring name = L"";
15     std::vector<int> a(10);
16     std::wstring key = L"";
17
18     for (int i = 0; i < letters.size(); i++)
19     {
20         name += letters[i];
21         a[i] = i + 1;
22         SendMessage(name, a[i]);
23     }
24     std::wcin >> key;
25     return 0;
26 }
```

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

Если приложение еще не запущено, клавиша **F5** запускает отладчик и останавливается в первой точке останова. В противном случае **F5** продолжает выполнение приложения до следующей точки останова.

Точки останова полезны, если вам известны строка или раздел кода, которые вы хотите подробно изучить. Дополнительные сведения о различных типах точек останова, которые можно задать, например об условных точках останова, см. в разделе [Использование точек останова](#).

## Переход по коду в отладчике с помощью пошаговых команд

Здесь мы используем в основном сочетания клавиш, так как они позволяют быстро выполнять приложение в отладчике (эквивалентные команды, например команды меню, отображаются в круглых скобках).

1. Во время приостановки в цикле `for` в методе `main` дважды нажмите клавишу **F11** (или выберите **Отладка > Шаг с заходом**), чтобы перейти к вызову метода `SendMessage`.

После двойного нажатия клавиши **F11** вы должны находиться на следующей строке кода:

```
SendMessage(name, a[i]);
```

2. Еще раз нажмите клавишу **F11**, чтобы выполнить шаг с заходом в метод `SendMessage`.

Желтый указатель перемещается в метод `SendMessage`.

```

4
5 void SendMessage(const std::wstring& name, int msg)
6 {
7     std::wcout << L"Hello, " << name << L"! Count to " << msg << std::endl;
8 }
9

```

F11 — это команда **Шаг с заходом**, которая выполняет приложение с переходом к следующему оператору. 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"](#)).

Предположим, что вы закончили изучать метод `SendMessage` и хотите выйти из него, но остаться в отладчике. Это можно сделать с помощью команды **Шаг с выходом**.

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

Эта команда возобновляет выполнение приложения (и работу отладчика) до возврата данных текущим методом или текущей функции.

Вы должны вернуться в цикл `for` в методе `main`, приостановленный на вызове метода `SendMessage`.

4. Нажмите клавишу **F11** несколько раз, пока не вернетесь к вызову метода `SendMessage`.
5. Во время приостановки на вызове метода один раз нажмите клавишу **F10** (или выберите **Отладка > Шаг с обходом**).

```


11 int main()
12 {
13     std::vector<wchar_t> letters = { L'f', L'r', L'e', L'd', L' ', L's', L'm', L'i', L't', L'h' };
14     std::wstring name = L"";
15     std::vector<int> a(10);
16     std::wstring key = L"";
17
18     for (int i = 0; i < letters.size(); i++)
19     {
20         name += letters[i];
21         a[i] = i + 1;
22         SendMessage(name, a[i]);
23     }
24     std::wcin >> key;
25     return 0;
26 }

```

Обратите внимание, что в этот раз отладчик не заходит в метод `SendMessage`. Клавиша **F10** перемещает отладчик без захода в функции или методы в коде приложения (код продолжает выполняться). Нажав клавишу **F10** (а не **F11**) в вызове метода `SendMessage`, мы пропускаем код реализации для `SendMessage`.

(пока это нас не интересует). Дополнительные сведения о различных способах перемещения по коду см. в разделе [Навигация по коду в отладчике](#).

## Переход по коду с помощью команды "Выполнение до щелкнутого"

1. Нажмите клавишу F5, чтобы перейти к точке останова.
2. В редакторе кода прокрутите код вниз и наведите указатель мыши на функцию `std::wcout` в методе `SendMessage`, чтобы в левой части появилась зеленая кнопка **Выполнение до щелкнутого** . В подсказке для кнопки выводится "Выполнить до этого места".



### ⚠ Примечание


Кнопка **Выполнение до щелкнутого** впервые появилась в Visual Studio 2017. (Если кнопка с зеленой стрелкой отсутствует, воспользуйтесь клавишей F11, чтобы переместить отладчик в нужное место.)

3. Нажмите кнопку **Выполнение до щелкнутого** .

Отладчик перемещается к функции `std::wcout`.

Использование этой кнопки аналогично установке временной точки останова. Функция **Выполнение до щелкнутого** удобна для быстрой работы в видимой области кода приложения (можно щелкнуть в любом открытом файле).

## Быстрый перезапуск приложения

Нажмите кнопку **Перезапустить**  на панели инструментов отладки (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.

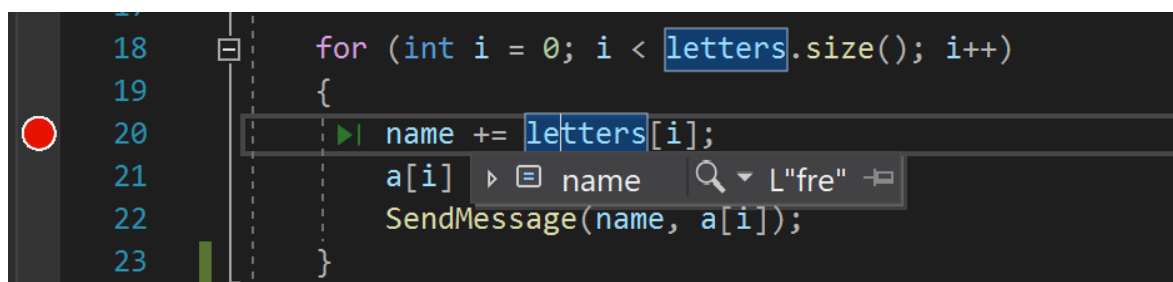


Отладчик еще раз останавливается в точке останова, ранее заданной вами в цикле `for`.

## Проверка переменных с помощью подсказок по данным

Функции, позволяющие проверять переменные, являются самыми полезными возможностями отладчика. Реализовывать эту задачу можно разными способами. Часто при попытке выполнить отладку проблемы пользователь старается выяснить, хранятся ли в переменных значения, которые требуются ему в определенное время.

1. При приостановке на операторе `name += letters[i]` наведите указатель мыши на переменную `letters` и увидите ее значение по умолчанию — `size={10}`.
2. Разверните переменную `letters`, чтобы просмотреть ее свойства, включая все элементы, которые она содержит.
3. Затем наведите указатель мыши на переменную `name`, чтобы просмотреть ее текущее значение — пустую строку.
4. Несколько раз нажмите клавишу **F5** (или выберите **Отладка>Продолжить**), чтобы выполнить несколько итераций по циклу `for`, каждый раз снова приостанавливая выполнение в точке останова и наводя указатель мыши на переменную `name`, чтобы просмотреть ее значение.



Значение переменной изменяется при каждой итерации цикла `for` — `f`, затем `fr`, `fre` и т. д.

Часто при отладке требуется быстро проверить значения свойств в переменных, чтобы убедиться, что в них хранятся ожидаемые значения. Советы по данным — отличный способ это сделать.

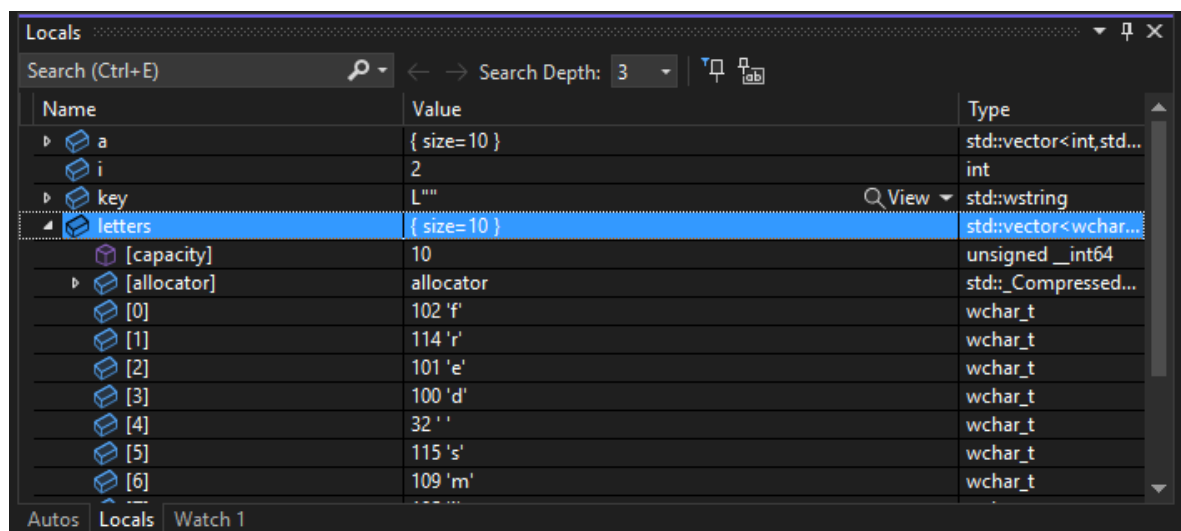
# Проверка переменных с помощью окон "Видимые" и "Локальные"

1. Взгляните на окно **Видимые** в нижней части редактора кода.

Если оно закрыто, откройте его во время приостановки в отладчике, выбрав **Отладка > Окна > Видимые**.

В окне **Видимые** отображаются переменные и их текущие значения. В окне **Видимые** отображаются все переменные, используемые в текущей или предыдущей строке (сведения о зависящем от языка поведении см. в соответствующей документации).

2. Затем посмотрите на окно **Локальные** на вкладке рядом с окном **Видимые**.
3. Разверните переменную `letters`, чтобы отобразить элементы, которые она содержит.



В окне **Локальные** показаны переменные, которые находятся в текущей области [↗](#), то есть текущем контексте выполнения.

## Установка контрольного значения

1. В основном окне редактора кода щелкните правой кнопкой мыши переменную `name` и выберите команду **Добавить контрольное значение**.

В нижней части редактора кода откроется окно **Контрольное значение**. В окне **Контрольное значение** можно указать переменную (или выражение), которую необходимо отслеживать.

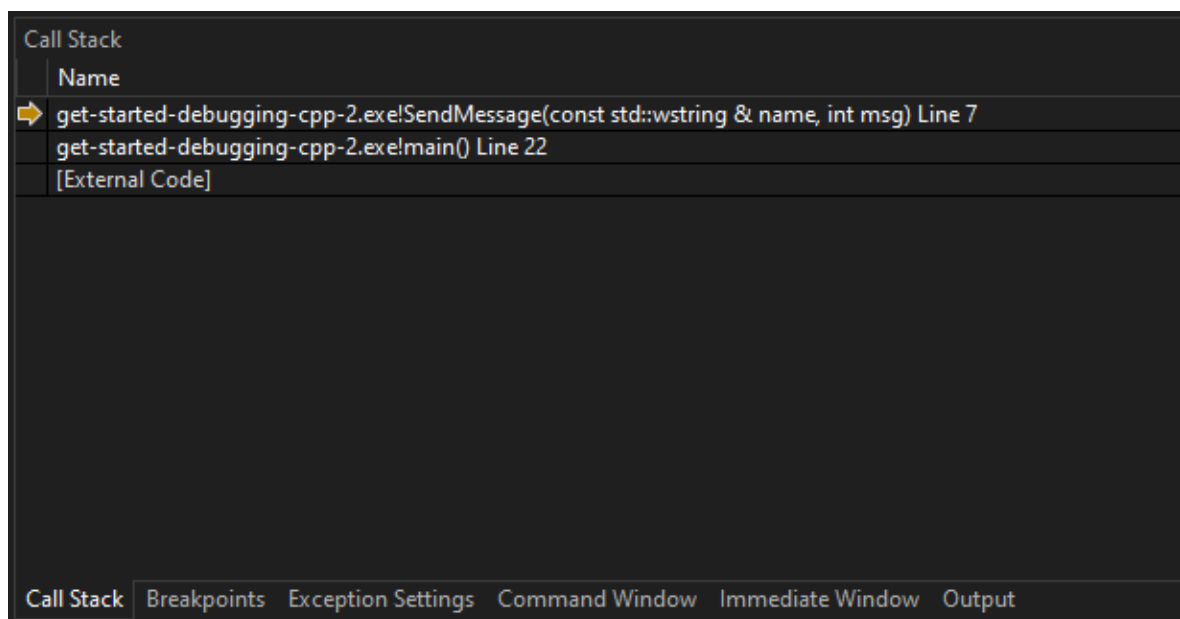
Теперь у вас есть контрольное значение, заданное для переменной `name`, и по мере перемещения по отладчику вы можете наблюдать за изменением его значения. В отличие от других окон переменных, в окне **Контрольное значение** всегда отображаются просматриваемые вами переменные (они выделяются серым цветом,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

## Просмотр стека вызовов

1. Во время приостановки в цикле `for` щелкните окно **Стек вызовов**, которое по умолчанию открыто в нижней правой области.

Если оно закрыто, откройте его во время приостановки в отладчике, выбрав **Отладка > Окна > Стек вызовов**.

2. Несколько раз нажмите клавишу **F11**, пока отладчик не приостановится в методе `SendMessage`. Взгляните на окно **Стек вызовов**.



В окне **Стек вызовов** показан порядок вызова методов и функций. В верхней строке приведена текущая функция (в данном приложении метод `SendMessage`). Во второй строке показано, что функция `SendMessage` была вызвана из метода `main` и т. д.

### ⓘ Примечание

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

Стек вызовов хорошо подходит для изучения и анализа потока выполнения приложения.

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

Для выполнения других задач можно воспользоваться контекстными меню из окна **Стек вызовов**. Например, можно вставлять точки останова в указанные функции, перемещать отладчик с помощью функции **Выполнение до текущей позиции** и изучать исходный код. Дополнительные сведения см. в разделе [Практическое руководство. просмотреть стек вызовов](#).

## Изменение потока выполнения

1. Дважды нажмите клавишу **F11**, чтобы запустить функцию `std::wcout`.
2. Приостановив отладчик в вызове метода `SendMessage`, с помощью мыши захватите желтую стрелку (указатель выполнения) в левой части и переместите ее вверх на одну строку — обратно в `std::wcout`.
3. Нажмите клавишу **F11**.

Отладчик повторно выполнит функцию `std::wcout` (вы увидите это в выходных данных окна консоли).

Изменяя поток выполнения, можно решать множество задач, например тестировать различные пути выполнения кода или повторно выполнять код без перезапуска отладчика.

### Предупреждение

Как правило, при работе с этой функцией необходимо соблюдать осторожность — вы увидите соответствующее предупреждение во всплывающей подсказке. Могут отображаться и другие предупреждения. При перемещении указателя предыдущее состояние приложения не возвращается.

4. Чтобы продолжить выполнение приложения, нажмите клавишу **F5**.

Поздравляем с завершением этого учебника!