Как писать техническую документацию для программ на C#

Рано или поздно в жизни каждого разработчика наступает момент, когда он не понимает, как работает его код. Выясняем, что с этим делать.

Поделиться



 vlada\_maestro / shutterstock

Программисты часто сталкиваются с тем, что не могут прочитать код. Такое случается постоянно: когда только приходят в новый проект, когда проверяют код коллеги или — так бывает чаще всего — когда смотрят результат своей же работы. Чтобы этого избежать, нужно писать и читать документацию.

Разработчики имеют дело с двумя основными видами документации:

1. **Пользовательская документация**. Это руководство по эксплуатации программ. Обычно оно нужно для сложных профессиональных инструментов. Если же пользователи не могут сами разобраться в приложении пиццерии, то лучше доработать интерфейс — добровольно никто инструкцию читать не станет.
2. **Техническая документация**. Это пояснения для программистов, которые будут использовать или дорабатывать существующий код. Они помогут быстро вникнуть в проект и начать работать. Или же продолжить писать программу после долгого перерыва.

К сожалению, не все разработчики (практически никто) любят читать документацию. Любителей писать её и того меньше. Однако делать это очень важно, потому что сложно поддерживать проект без документации.



[](https://skillbox.ru/media/authors/evgenij-kucheryavyj/)

**Евгений Кучерявый**

Пишет о разработке сайтов, в свободное время создает игры. Мечтает открыть свою студию и выпускать ламповые RPG.

**Содержание**

* [Два вида документации](https://skillbox.ru/media/code/kak_pisat_tekhnicheskuyu_dokumentatsiyu/#1)
* [Правила хорошего тона в составлении документации](https://skillbox.ru/media/code/kak_pisat_tekhnicheskuyu_dokumentatsiyu/#2)
* [Где писать документацию в C#](https://skillbox.ru/media/code/kak_pisat_tekhnicheskuyu_dokumentatsiyu/#3)
* [Как создать файл документации](https://skillbox.ru/media/code/kak_pisat_tekhnicheskuyu_dokumentatsiyu/#4)
* [Заключение](https://skillbox.ru/media/code/kak_pisat_tekhnicheskuyu_dokumentatsiyu/#5)

**Правила хорошего тона в составлении документации**

Составляя документацию, стоит следовать определенным правилам — они помогают сделать ее более понятной.

**1. Документация нужна не всегда**

Если программа одноразовая, не стоит тратить время на написание пояснений. Например, если нужен небольшой скрипт, который будет написан за пять минут и использован 1-2 раза.

**2. Документация нужна не везде**

Также не нужно писать пояснения ко всему. Если код написан хорошо, то по названиям уже будет понятно, что это такое и зачем оно используется. Например, легко догадаться, что метод *int Sum (int a, int b)* возвращает результат сложения двух чисел.

Исключение можно сделать, если речь идет об API или фреймворке, которыми пользуются многие разработчики: они не всегда видят исходный код, но могут использовать классы и методы. Поэтому им важно иметь список доступных методов. В этом случае задокументировать всё нужно просто для галочки.

**3. Документация должна быть точной**

Очень важно уметь ясно выражать свои мысли. Нужно предельно точно описывать, что делает тот или иной фрагмент кода. Для этого стоит давать как можно более короткие определения. Например:

/// <summary>

/// Сообщение в чате.

/// </summary>

class Message

{

…

/// <summary>

/// Текст сообщения.

/// </summary>

public string Text

{

get { return this.text; }

}

}

В этом фрагменте кода объем документации к классу и его свойству не превышает одного предложения. Этого достаточно, чтобы было понятно, что это такое и для чего его нужно использовать.

**4. Документация должна быть сухой**

Хотя канцеляризмов нужно избегать, писать надо максимально сухо и по делу. Никаких стихов, метафор, аналогий или шуток — всё это может быть забавным, но не добавляет ясности.

**5. В документации не должно быть старого кода**

Этот правило больше касается обычных комментариев, чем самой документации. Однако оно очень важное, поэтому приведено здесь.

Никогда не храните в коде старые методы и операторы, даже если они задокументированы. Если что-то не используется в текущий момент — это мусор, от которого нужно избавиться.

Если есть сомнения пригодится ли еще этот код, его лучше сохранить в системе контроля версий — именно для этого ее и придумали.

Дальше речь пойдет о том, как писать техническую документацию для программ на C#. Вся работа будет вестись в Visual Studio.

**Где писать документацию в C#**

Вариантов много. Например, можно сделать это в Word или Google Docs, тогда разработчики смогут скачивать файл из интернета. Некоторые хранят инструкции в печатном виде, но это плохой вариант, потому что документация быстро устаревает.

Лучший способ — писать всё прямо в коде программы. Тогда у каждого разработчика будет доступ к документации в любое время. Самый примитивный вариант — использовать комментарии.

В C# есть два вида комментариев. Однострочные:

int a = 11 + 12; //Это однострочный комментарий

И многострочные:

/\* Начало комментария

Это

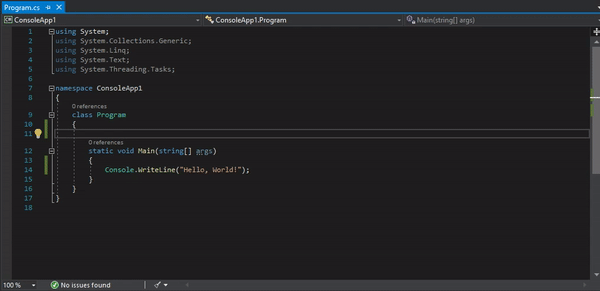
Многострочный

Комментарий

Конец комментария \*/

Компилятор во время сборки игнорирует комментарии и просто вырезает их, поэтому на работу программы они не влияют.

Более продвинутый вариант — использовать XML. Чтобы вставить XML-комментарий, нужно перед названием класса, поля, свойства или метода поставить тройной слеш.



После этого автоматически будет создано два элемента:

1. **Summary** — общий комментарий. В нем пишут, что делает метод или для чего нужен класс.
2. **Param** — комментарий об аргументе. В нем указывается, какое значение надо передать.

Практически все инструменты, в том числе и Visual Studio, поддерживают вывод подсказок, которые подгружаются из документации. И теперь, если навести на метод *Main ()*или его аргумент, то можно увидеть, что было написано в комментарии.

Такой способ намного лучше, потому что человеку вообще не нужно ничего открывать, чтобы определить, что делает какой-нибудь фрагмент кода. Конечно, наличие XML создает визуальный шум, но его можно просто скрыть.

Еще можно использовать такие XML-элементы, как:

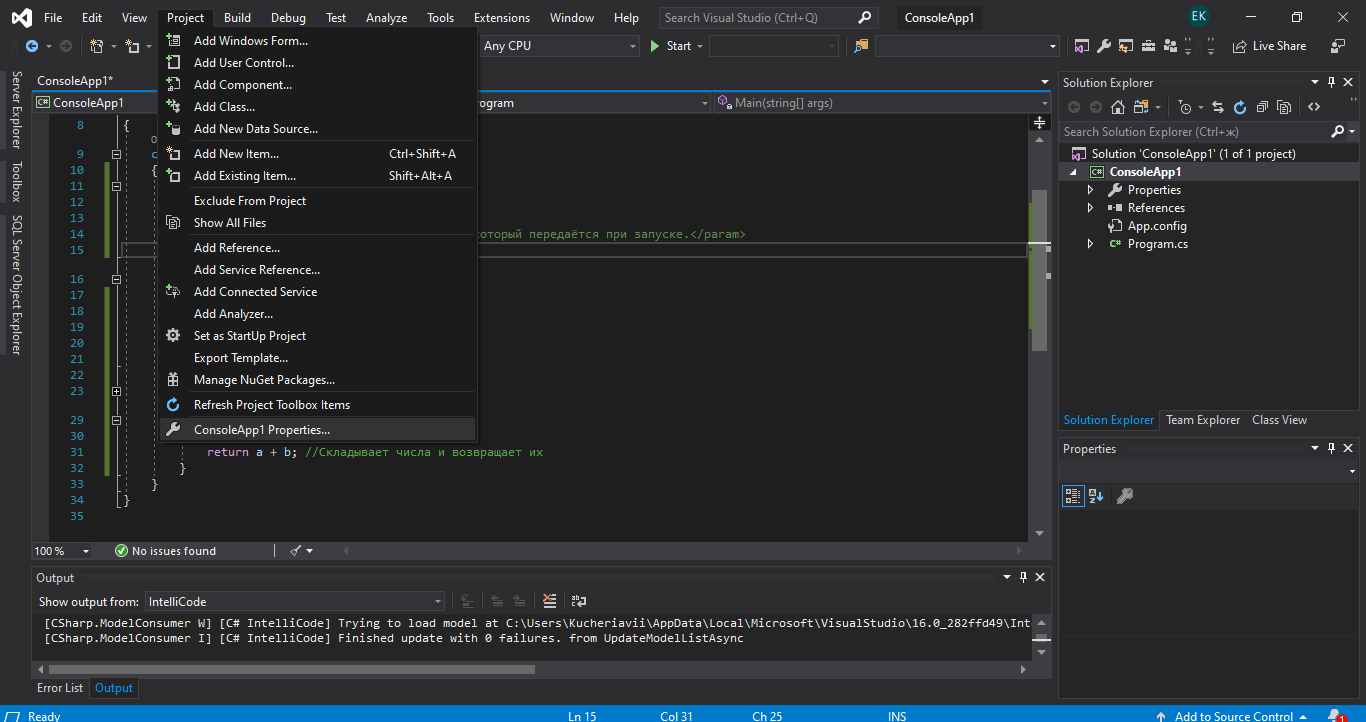
* **Returns** — возвращаемое значение;
* **Value** — значение свойства;
* **Exception** — исключение;
* **Remarks** — ремарка к основному комментарию.

Таких элементов очень много, подробнее почитать о них можно в [документации Microsoft](https://docs.microsoft.com/ru-ru/dotnet/csharp/codedoc). Цель же этой статьи — показать, как документировать код, чтобы разбираться в проекте стало легче, а не сложнее.

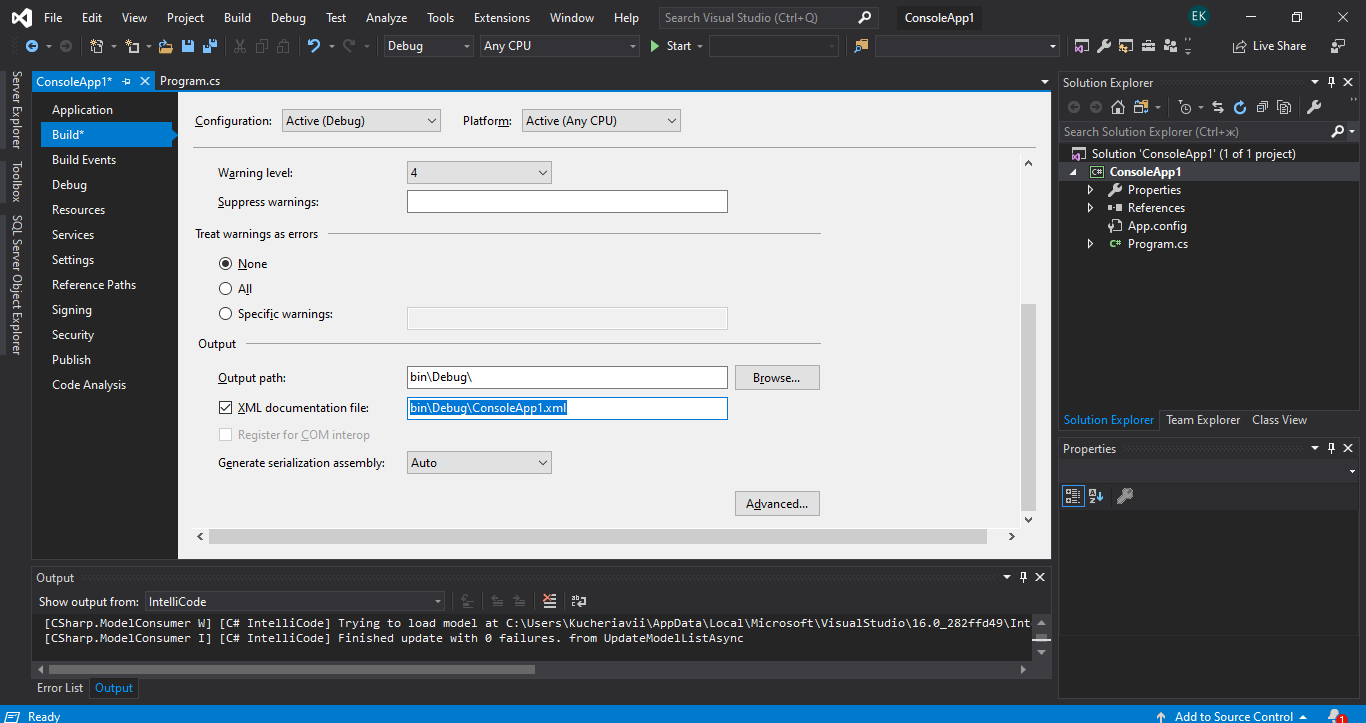
**Как создать файл документации**

Иногда все-таки нужно сохранить документацию вне кода. Чаще всего ее сохраняют в HTML-формате, а потом загружают на сайт, чтобы разработчики имели к ней доступ.

Для этого сначала нужно зайти в настройки проекта:

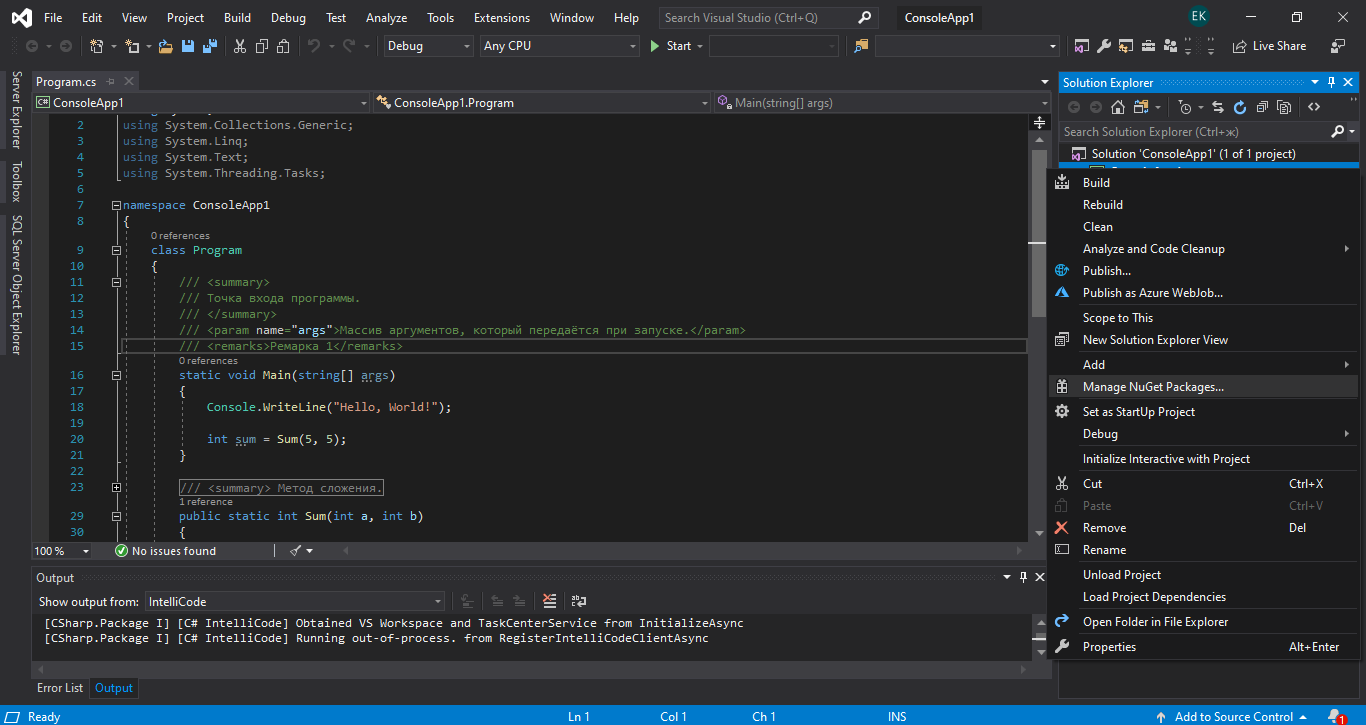


А потом перейти во вкладку *Build* и поставить галочку *XML documentation file*:

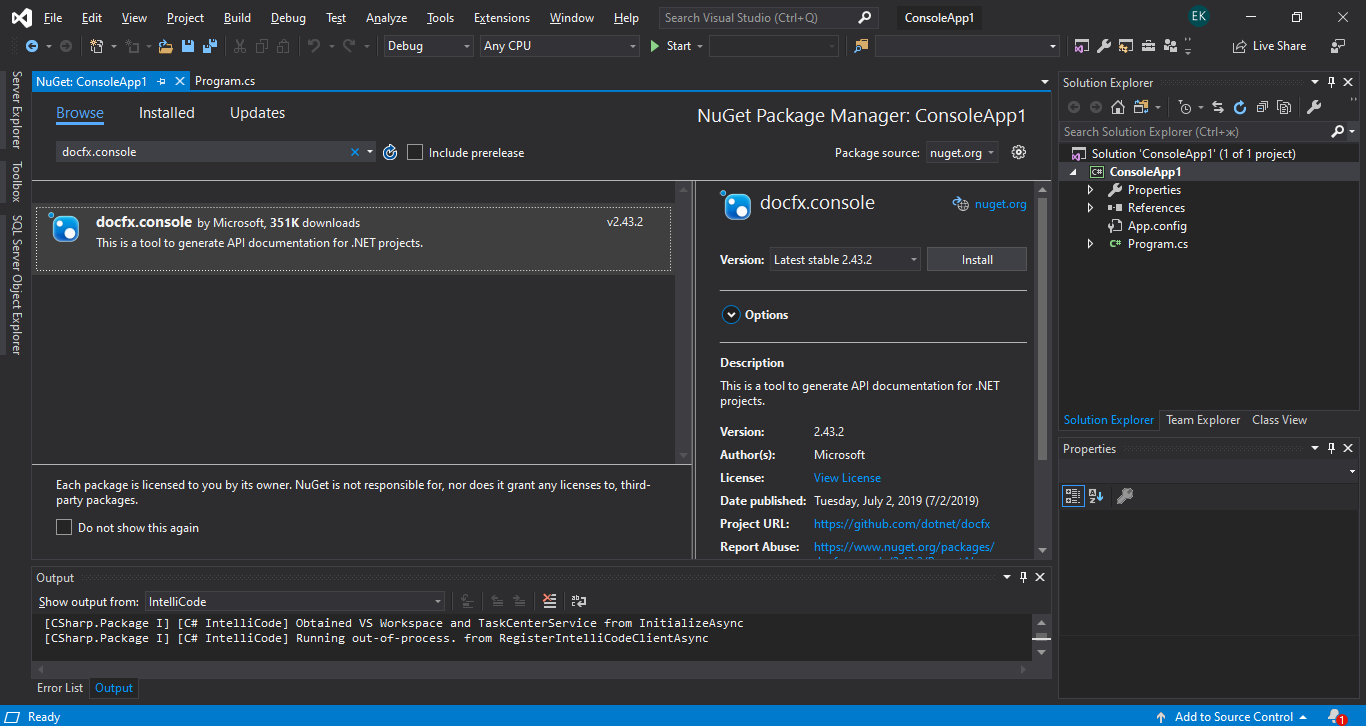


Теперь вместе с компиляцией программы будет создаваться файл с документацией в формате XML. Его можно преобразовать в HTML с помощью специальных утилит. Microsoft для этого рекомендует использовать [DocFX](https://dotnet.github.io/docfx/) или [Sandcastle](https://github.com/EWSoftware/SHFB).

Рассмотрим на примере DocFX. Его можно скачать с помощью NuGet Package Manager в Visual Studio. Для этого нажмите на проект правой кнопкой мыши и выберите пункт *Manage NuGet Packages*:



Затем перейдите во вкладку *Browse* и введите в поле поиска название *docfx.console*, а потом нажмите *Install*:



После нужно подтвердить установку и согласиться с условиями лицензионного соглашения.

Теперь при сборке проекта будет создаваться папка*\_site*, в которой находится сайт с документацией. Однако это касается класса *Program*, поэтому чтобы проверить работу DocFX, нужно добавить какой-нибудь класс:

namespace ConsoleApp1

{

/// <summary>

/// Класс, который представляет пользователя.

/// </summary>

public class User

{

private string name;

/// <summary>

/// Конструктор класса.

/// </summary>

/// <param name="name">Имя пользователя.</param>

public User(string name)

{

this.name = name;

}

/// <summary>

/// Меняет имя пользователя.

/// </summary>

/// <param name="name">Имя пользователя.</param>

public void ChangeName(string name)

{

this.name = name;

}

/// <summary>

/// Метод для вывода имени пользователя.

/// </summary>

/// <returns>Возвращает имя пользователя.</returns>

public string GetName()

{

return this.name;

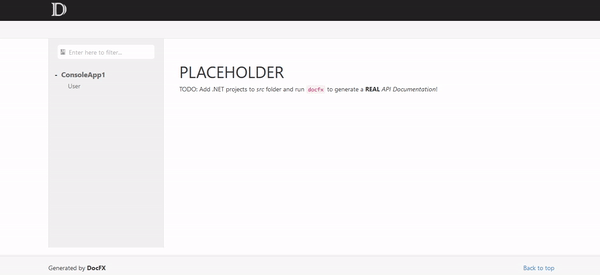
}

}

}

После компиляции проекта с таким классом можно проверить сайт. Его главная страница будет пуста — она нужна для того, чтобы вкратце описать свою программу. Сама же документация находится по адресу *api/index.html*.

Вот как она выглядит:



**Заключение**

Многим, и мне в том их числе, гораздо интереснее писать код, а не описывать его. Однако хорошая документация очень важна, если над проектом работает несколько человек или если это API, которым будут пользоваться сторонние программисты.

Кроме того, хорошая практика разработки — когда сначала пишется документация, а потом создаются классы и методы, которые должны ей соответствовать.