# Правила именования

Допустимые идентификаторы должны соответствовать этим правилам. Компилятор C# выдает ошибку для любого идентификатора, который не соответствует этим правилам:

* Идентификаторы должны начинаться с буквы или символа подчеркивания (\_).
* Идентификаторы могут содержать буквенные символы Юникода, десятичные цифровые символы, соединительные символы Юникода, объединяющие символы Юникода или символы форматирования Юникода.

Вы можете объявить идентификаторы, соответствующие ключевым словам C#, используя префикс @ в идентификаторе. Префикс @ не является частью имени идентификатора. Например, @if объявляет идентификатор с именем if. Эти дословные идентификаторы предназначены в первую очередь для взаимодействия с идентификаторами, объявленными в других языках.

# Соглашения об именах

Помимо правил, соглашения об именах идентификаторов используются во всех API .NET. Эти соглашения обеспечивают согласованность имен, но компилятор не применяет их. Вы можете использовать различные соглашения в проектах.

По соглашению программы C# используют PascalCase для имен типов, пространства имен и всех открытых членов. Кроме того, dotnet/docs команда использует следующие соглашения, принятые из стиля кодирования команды среды выполнения .NET:

* Имена интерфейсов начинаются с заглавной буквы I.
* Типы атрибутов заканчиваются словом Attribute.
* Типы перечисления используют единственное существительное для нефлагов и множественное существительное для флагов.
* Идентификаторы не должны содержать два последовательных символа подчеркивания (\_). Эти имена зарезервированы для идентификаторов, созданных компилятором.
* Используйте значимые и описательные имена для переменных, методов и классов.
* Предпочитайте ясность по поводу краткости.
* Используйте PascalCase для имен классов и имен методов.
* Используйте верблюдье Регистр для аргументов метода, локальных переменных и частных полей.
* Используйте PascalCase для имен констант, как полей, так и локальных констант.
* Поля частного экземпляра начинаются с подчеркивания (\_).
* Статические поля начинаются с s\_. Это соглашение не является поведением Visual Studio по умолчанию, а не частью рекомендаций по проектированию Платформы, но настраивается в editorconfig.
* Избегайте использования аббревиаций или акронимов в именах, за исключением широко известных и принятых аббревиаций.
* Используйте значимые и описательные пространства имен, которые соответствуют нотации обратного доменного имени.
* Выберите имена сборок, представляющие основную цель сборки.
* Избегайте использования однобуквенных имен, за исключением простых счетчиков циклов. Кроме того, в примерах синтаксиса, описывающих синтаксис конструкций C#, часто используются следующие однобуквные имена, соответствующие соглашению, используемому в спецификации языка C#. Примеры синтаксиса являются исключением из правила.
  + Используется S для структур C для классов.
  + Используется M для методов.
  + Используется v для переменных p для параметров.
  + Используется r для ref параметров.

В следующих примерах рекомендации, относящиеся к элементам, помеченным public, также применимы при работе с protected элементами и protected internal элементами, которые предназначены для отображения внешним вызывающим элементам.

# Регистр Pascal

Используйте pascal casing (PascalCasing) при именовании class, Interfacestructили delegate типа.

1. **public** **class** DataService
2. {
3. }
4. **public** record PhysicalAddress(
5. **string** Street,
6. **string** City,
7. **string** StateOrProvince,
8. **string** ZipCode);
9. **public** **struct** ValueCoordinate
10. {
11. }
12. **public** **delegate** **void** DelegateType(**string** message);

При присвоении имен interface помимо добавления к имени префикса I используйте прописные и строчные буквы в стиле языка Pascal. Этот префикс четко указывает потребителям, что это interface.

1. **public** **interface** IWorkerQueue
2. {
3. }

При именовании public элементов типов, таких как поля, свойства, события, используйте регистр pascal. Кроме того, используйте pascal casing для всех методов и локальных функций.

1. **public** **class** ExampleEvents
2. {
3. // A public field, these should be used sparingly
4. **public** **bool** IsValid;
6. // An init-only property
7. **public** IWorkerQueue WorkerQueue { **get**; init; }
9. // An event
10. **public** **event** Action EventProcessing;
12. // Method
13. **public** **void** StartEventProcessing()
14. {
15. // Local function
16. **static** **int** CountQueueItems() => WorkerQueue.Count;
17. // ...
18. }
19. }

При написании позиционных записей используйте регистр pascal для параметров, так как

1. **public** record PhysicalAddress(
2. **string** Street,
3. **string** City,
4. **string** StateOrProvince,
5. **string** ZipCode);

они — общедоступные свойства записи.

# "Верблюжий" стиль

При именовании private или internal полях и префиксе \_их используйте верблюдьи регистры ("верблюдьи"). Используйте регистр верблюда при именовании локальных переменных, включая экземпляры типа делегата.

1. **public** **class** DataService
2. {
3. **private** IWorkerQueue \_workerQueue;
4. }

При работе с полями static, являющимися private или internal, используйте префикс s\_, а для статического потока используйте t\_.

1. **public** **class** DataService
2. {
3. **private** **static** IWorkerQueue s\_workerQueue;
5. [ThreadStatic]
6. **private** **static** TimeSpan t\_timeSpan;
7. }

При написании параметров методов используйте "верблюжий" стиль.

1. **public** T SomeMethod<T>(**int** someNumber, **bool** isValid)
2. {
3. }

# Правила именования параметров типа

Следующие рекомендации применяются к параметрам типа для параметров универсального типа. Параметры типа — это заполнители аргументов в универсальном типе или универсальном методе. Дополнительные сведения о параметрах универсального типа см. в руководстве по программированию на C#.

* Укажите параметры универсального типа с описательными именами, если только одно имя буквы не является полностью понятным, а описательное имя не добавит значения.
* **public** **interface** ISessionChannel<TSession> { /\*...\*/ }
* **public** **delegate** TOutput Converter<TInput, TOutput>(TInput from);
* **public** **class** List<T> { /\*...\*/ }
* Рекомендуется использовать T в качестве имени параметра типа для типов с одним параметром типа буквы.
* **public** **int** IComparer<T>() { **return** 0; }
* **public** **delegate** **bool** Predicate<T>(T item);
* **public** **struct** Nullable<T> where T : **struct** { /\*...\*/ }
* Используйте префикс "T" для описательных имен параметров типа.
* **public** **interface** ISessionChannel<TSession>
* {
* TSession Session { **get**; }
* }
* Указывайте ограничения, связанные с параметром типа, в его имени. Например, параметр, ограниченный ISession для вызова TSession.

# Дополнительные соглашения об именовании

* В примерах, где нет директив using, рекомендуется использовать полные указания для пространства имен. Если известно, что пространство имен импортируется в проект по умолчанию, вам не нужно указывать полные имена из этого пространства имен. Полные имена можно разбить после точки (.), если они слишком длинны для одной строки, как показано в следующем примере.
* var currentPerformanceCounterCategory = **new** System.Diagnostics.
* PerformanceCounterCategory();
* Вам не нужно изменять имена объектов, созданных с помощью инструментов разработки Visual Studio, чтобы привести их в соответствие с другими рекомендациями.

# Ссылки на документацию

1. Правила и соглашения об именовании идентификаторов C# - <https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/coding-style/identifier-names>
2. Общие соглашения о коде C# - <https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/coding-style/coding-conventions>