- значимые типы (Value Types)

1) числовые

1.1 числа со знаком (sbyte, short, int, long)

1.2 числа без знака (byte, ushort, uint, ulong)

1.3 дробные числа (float, double, decimal)

2) логические (bool)

3) символьные (char)

- ссылочные типы

1) строки (string)

2) объекты (object)

-пользовательские

Класс является основным пользовательским типом. Синтаксис объявления класса в C# следующий:

*<модификаторы>* class *<имя класса>*

{

*[<элементы класса>]*

}

## Допустимые элементы класса

**1. Поле.** Синтаксис объявления поля класса совпадает с синтаксисом оператора объявления переменной[[1]](#footnote-1) (тип поля должен всегда быть указан явно, использование var не допускается).

**2. Константа.** Синтаксис объявления константы в классе аналогичен синтаксису, применяемому при объявлении константы в теле метода.

Следующие элементы класса будут подробно рассмотрены в дальнейшем.

**3. Метод.** Методы описывают функциональность класса.

**4. Свойство.** Свойства класса призваны предоставить защищенный доступ к полям.

**5. Индексатор.** Индексатор – это свойство-коллекция, отдельный элемент которого доступен по индексу.

**6. Конструктор.** Задача конструктора – начальная инициализация объекта или класса.

**7. Финализатор.** Финализатор автоматически вызывается сборщиком мусора и содержит завершающий код для объекта.

**8. Событие.** События представляют собой механизм рассылки уведомлений различным объектам.

**9. Операция.** Язык C# допускает перегрузку некоторых операций для объектов класса.

**10. Вложенный пользовательский тип.** Описание класса может содержать описание другого пользовательского типа – класса, структуры, перечисления, интерфейса, делегата. Обычно вложенные типы выполняют вспомогательные функции и явно вне основного типа не используются.

*модификаторы доступа*:

* private. Элемент с данным модификатором доступен только в том типе, в котором определён. Например, поле доступно только в содержащем его классе.
* protected. Элемент виден в типе, в котором определен, и в наследниках этого типа (даже если наследники расположены в других сборках).
* internal. Элемент доступен без ограничений, но только в той сборке, где описан.
* protected internal. Комбинация модификаторов protected и internal. Элемент виден в содержащей его сборке без ограничений, а вне сборки – только в наследниках типа (т.е. protected *или* internal[[2]](#footnote-2)).
* public. Элемент доступен без ограничений как в той сборке, где описан, так и в других сборка, к которым подключается сборка с элементом.

Инструкция **Structure** определяет тип составного типа значения, который можно настроить. Структуры поддерживают работу со многими из тех элементов, с которыми работают классы. Например, структуры могут иметь свойства и процедуры, реализовывать интерфейсы и иметь параметризованные конструкторы. Однако в отношении наследования, объявлений и использования между структурами и классами существуют значительные различия. Также классы относятся к ссылочным типам, а структуры — к типам значений. **Structure** можно использовать только на уровне пространства имен или модуля. Это означает, что *контекст объявления* для структуры должен быть исходным файлом, пространством имен, классом, структурой, модулем или интерфейсом, и не может быть процедурой или блоком. Уровни доступа можно настроить с помощью модификаторов доступа.

* **Вложения.** Можно определить одну структуру внутри другой. Внешняя структура называется *содержащей структурой*, а внутренние структуры называются *вложенными структурами*. Однако невозможно будет получить доступ к членам вложенной структуры через содержащую структуру. Вместо этого следует объявить переменную типа данных вложенной структуры.
* **Объявление члена.**Необходимо объявить все элементы структуры. Ничто не может наследоваться от структуры. Можно объявить ноль или более общей переменной или собственным, события без общего доступа в структуре. Структура не может содержать только константы, свойства и процедуры, даже если некоторые из них не используются совместно.
* **Инициализация.** Инициализация значений неиспользуемых совместно данных-членов структуры не может быть частью их объявлений.Инициализация таких данных-членов должна выполняться на структуре с помощью параметризованного конструктора или путем присваивания значения члену после создания экземпляра структуры.
* **Наследование.** Структура не может наследовать от типа, отличного от [ValueType](https://msdn.microsoft.com/ru-ru/library/system.valuetype.aspx), от которого наследуют все структуры. В частности, одна структура не может наследовать от другой.

Нельзя использовать [Инструкция Inherits](https://msdn.microsoft.com/ru-ru/library/1yk8s7sk.aspx) в определении структуры, даже для указания [ValueType](https://msdn.microsoft.com/ru-ru/library/system.valuetype.aspx).

* **Реализация.** Если структура использует [Оператор Implements](https://msdn.microsoft.com/ru-ru/library/7z6hzchx.aspx), необходимо реализовать каждый член, определенный каждым интерфейсом, заданным в *interfacenames*.

### Поведение

* **Уровень доступа.** Внутри структуры можно объявить каждый член со своим собственным уровнем доступа. Все члены структуры по умолчанию имеют доступ [public](https://msdn.microsoft.com/ru-ru/library/9dc6we3z.aspx). Обратите внимание: если структура сама по себе имеет более ограниченный уровень доступа, это автоматически ограничивает доступ к ее членам, даже если отрегулировать их уровни доступа модификатором доступа.
* **Область действия.** Область действия структуры — ее пространство имен, класс, структура или модуль. Областью действия каждого члена структуры является вся структура.
* **Время существования.** Для самой структуры не определено время существования. Вместо этого каждый экземпляр данной структуры имеет время существования, независимое от всех других экземпляров. Время существования экземпляра начинается, когда он создается условием [Оператор new](https://msdn.microsoft.com/ru-ru/library/77s47661.aspx). Оно заканчивается, когда заканчивается время существования переменной, содержащей его. Нельзя увеличить время существования экземпляра структуры. Приближение к функциональности статической структуры обеспечивается модулем.

У членов структуры время существования зависит от того, как и где они были объявлены.

* **Квалификация.** Код вне структуры должен определять имя члена с именем этой структуры.

Если код внутри вложенной структуры делает неопределенную ссылку на элемент программирования, сначала элементы ищется во вложенной структуре, затем в его содержащей структуре, и т.д., вплоть до внешнего содержащего элемента.

* **Потребление ресурсов памяти**. Как и для всех составных типов данных, нельзя точно подсчитать общее потребление ресурсов памяти структуры простым сложением номинальных объемов памяти, занимаемых отдельными членами структуры. Более того, нельзя однозначно полагать, что порядок расположения элементов в памяти такой же, как и порядок их объявления. Если требуется контролировать расположение структуры в памяти, можно применить атрибут [StructLayoutAttribute](https://msdn.microsoft.com/ru-ru/library/system.runtime.interopservices.structlayoutattribute.aspx) в операторе **Structure**.

1. Как правило, идентификаторы полей снабжаются неким оговоренным префиксом. [↑](#footnote-ref-1)
2. В CLR имеется модификатор доступа, соответствующий protected **и** internal. При помощи языка C# такой уровень доступа описать нельзя. [↑](#footnote-ref-2)