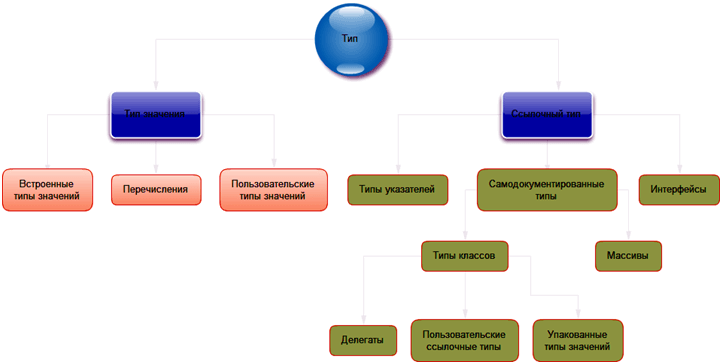
**Общая система типов данных в среде .NET. Размерные и ссылочные типы данных.**

Common Type System (сокр. CTS, рус. Стандартная система типов) — часть .NET Framework, формальная спецификация, определяющая, как какой-либо тип (класс, интерфейс, структура, встроенный тип данных) должен быть определён для его правильного выполнения средой .NET. Кроме того, данный стандарт определяет, как определения типов и специальные значения типов представлены в компьютерной памяти. Целью разработки CTS было обеспечение возможности программам, написанным на различных языках программирования, легко обмениваться информацией. Как это принято в языках программирования, тип может быть описан как определение набора допустимых значений и допустимых операций над этими значениями.



- Тип – базовый класс, представляющий тип.  
- Типы значений – базовый класс, представляющий любой тип значений.  
- Ссылочные типы – любые типы, которые доступны по ссылке и хранятся в куче.  
- Встроенные типы значений – включают большинство стандартных примитивных типов, представляющих числа, булевские значения, или символы.  
- Перечисления – наборы перечислимых значений.  
- Пользовательские типы значений – типы, которые определены в исходном коде и сохраняются как типы значений. В терминологии C# это означает любые структуры.  
- Интерфейсные типы – интерфейсы.  
- Типы указателей – указатели.  
- Самодокументированные типы – типы данных, которые представляют информацию о себе, полезную для сборщика мусора.  
- Массивы – любой тип, содержащий массив объектов.  
- Типы классов – самодокументированные типы, но не массивы.

- Делегаты – типы, определённые в исходном коде и сохраняемые как ссылочные. В терминологии C# это любые классы.  
- Упакованные типы значений – типы значений, временно помещённые в ссылки, благодаря чему могут сохраняться в куче.

*Типы значений:*

Типы значений являются производными от System.ValueType, являющегося производным от System.Object. Типы, производные от System.ValueType, имеют особое поведение в среде CLR. Переменные типа значения напрямую содержат их значения, что означает, что память встроена в контекст, в котором объявлена переменная. Не существует отдельного размещения кучи или служебных данных сборки мусора для переменных типа значения.

Существует две категории типов значений: *структура и перечисление*.

Встроенные числовые типы являются структурами, и к их свойствам и методам можно получить доступ.

Но значения объявляются и присваиваются им, как если бы они были простыми не статическими типами.

Типы значений являются запечатанными, что означает, например, что нельзя произвести тип от System.Int32, и нельзя определить структуру для наследования от любого пользовательского класса или структуры, поскольку структура может наследовать только от System.ValueType. Однако структура может реализовать один или несколько интерфейсов. Можно выполнить приведение типа структуры в тип интерфейса; это приведет к операции упаковки-преобразования для создания программы-оболочки структуры внутри объекта ссылочного типа в управляемой куче. Операции упаковки-преобразования возникают при передаче типа значения методу, принимающему System.Object в качестве входного параметра.

Ключевое слово struct используется для создания собственных пользовательских типов значений. Обычно структура используется как контейнер для небольшого набора связанных переменных.

Другой категорией типов значений является *перечисление*. Перечисление определяет набор именованных интегральных констант. Например, перечисление System.IO.FileMode в библиотеке классов платформы .NET Framework содержит набор именованных констант целого типа, которые задают, как должен быть открыт файл.

Все перечисления наследуются от System.Enum, который наследуется от System.ValueType. Все правила, применимые к структурам, также применяются к перечислениям.

*Ссылочные типы:*

Тип, определенный как класс, делегат, массив или интерфейс, является ссылочным типом. Во время выполнения при объявлении переменной ссылочного типа переменная содержит значение null до явного создания экземпляра объекта с помощью оператора new или назначения его объекту, который был создан в другом месте, с помощью new, as shown in the following example.

Интерфейс должен быть инициализирован вместе с объектом класса, который его реализует.

При создании объекта память размещается в управляемой куче, и переменная хранит только ссылку на расположение объекта. Для типов в управляемой куче требуются служебные данные и при их размещении, и при их удалении функциональной возможностью автоматического управления памятью среды CLR, также известной как сборка мусора. Однако сборка мусора также в высокой степени оптимизирована, и в большинстве сценариев она не создает проблем с производительностью.

Все массивы являются ссылочными типами, даже если их члены являются типами значений. Массивы являются неявно производными от класса System.Array, но объявляются и используются они с упрощенным синтаксисом, предоставленным C#.

Ссылочные типы полностью поддерживают наследование. При создании класса можно наследовать от любого другого интерфейса или класса, который не определен как запечатанный, а другие классы могут наследовать от этого класса и переопределять виртуальные методы.