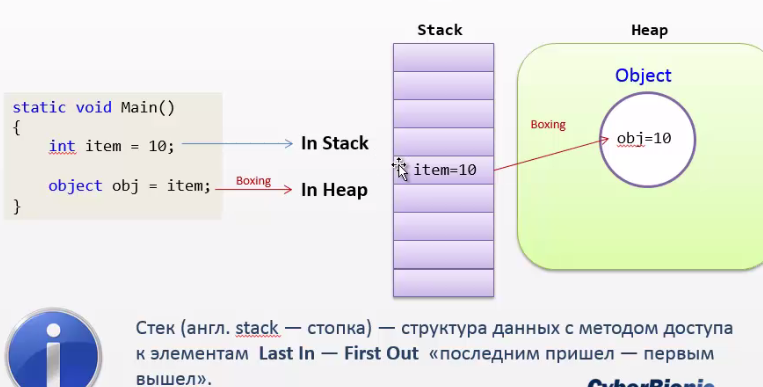
Копируется в экземпляр типа Object значение 10



Упаковка и распаковка.

// Упаковка (Boxing) - преобразование структурного типа (типа значения) в ссылочный тип

// (Object или любой другой тип интерфейса, реализуемый этим типом значения).

// Когда тип значения упаковывается средой CLR, она создает программу-оболочку значения внутри

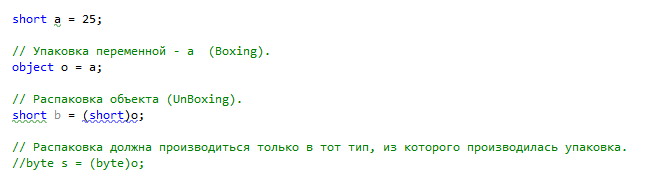
// System.Object и сохраняет ее в управляемой куче.

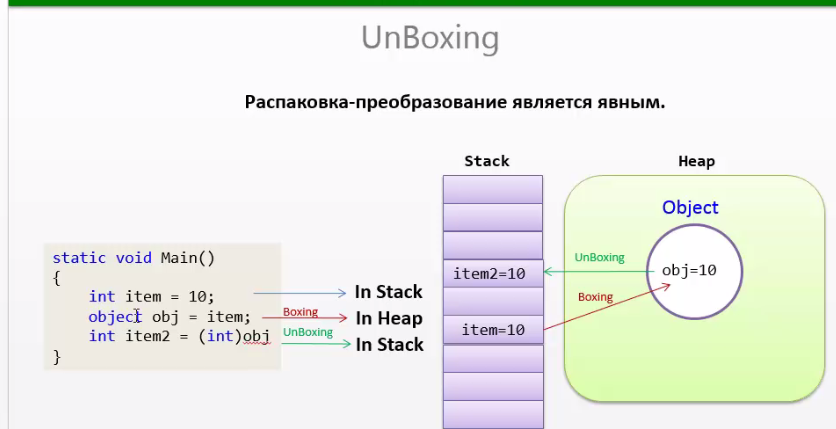
// Распаковка (Unboxing) - преобразование ссылочного типа в структурный тип.

// Операция распаковки-преобразования извлекает тип значения из объекта.

// Перед распаковкой среда выполнения проверяет совместимость между объектом и структурой,

// в которую будет происходить распаковка.





Упаковка структурного типа может быть :

В Object

ValueType

Либо в один из интерфейсов который реализовывает структура

Если много объектов и они не велики, то следует создавать их в стеке (пример самосвала с мышами в куче, зоопарк)

При выведении на консоль лучше упаковывать самостоятельно, если выводить одно и то же значение

#if INEFFICIENT

// При компиляции следующей строки v упакуется

// три раза, расходуя и время, и память

Console.WriteLine("{0}, {1}, {2}", v, v, v);

#else

// Следующие строки дают тот же результат,

// но выполняются намного быстрее и расходуют меньше памяти

Object o = v; // Упакуем вручную v (только единожды)

// При компиляции следующей строки код упаковки не создается

Console.WriteLine("{0}, {1}, {2}", o, o, o);

#endif

Помните, мы говорили, что неупакованные значимые типы «легче» ссылочных,

поскольку:

 память в управляемой куче им не выделяется;

 у них нет дополнительных членов, присущих каждому объекту в куче: указателя

на типовой объект и индекса блока синхронизации.

Поскольку неупакованные значимые типы не имеют индекса блока синхрони-

зации, то не может быть и нескольких потоков, синхронизирующих свой доступ

к экземпляру через методы типа System.Threading.Monitor (или инструкция lock

языка C#).

При вызове виртуальных методов не определенный в структуре она упаковывается

Вызов ToString. При вызове ToString упаковка p1 не требуется. Казалось бы,

тип p1 должен быть упакован, так как ToString — метод, унаследованный от ба-

зового типа, System.ValueType. Обычно для вызова виртуального метода нужен

указатель на типовой объект, а поскольку p1 является неупакованным значимым

типом, то нет ссылки на типовой объект Point. Однако JIT-компилятор видит,

что метод ToString переопределен в Point, и создает код, который напрямую

(невиртуально) вызывает ToString. Компилятор знает, что полиморфизм здесь

невозможен, коль скоро Point является значимым типом, а значимые типы не

могут применяться для другого типа в качестве базового и по-другому реали-

зовывать виртуальный метод. Ели бы метод ToString из Point во внутренней

реализации вызывал base.ToString(), то экземпляр значимого типа был бы

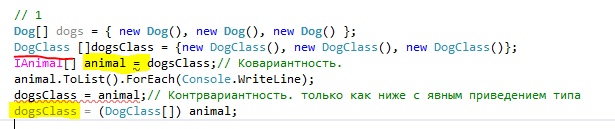
упакован при вызове метода ToString типа System.ValueType.



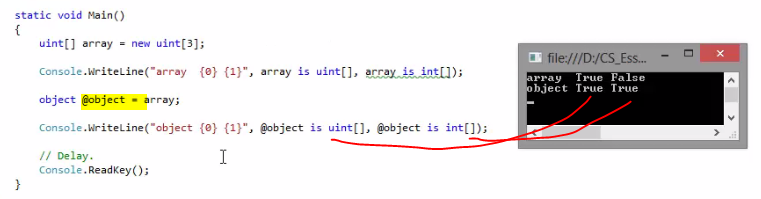
Ковариантность - это неявный обкаст массива (именно не явный)

Контрвариантность – это неявный даункаст массива

Ковариантность массивов ссылочных типов существует (только Ковариантность ), а типов значений нет (не Ковариантность, не Контрвариантность)

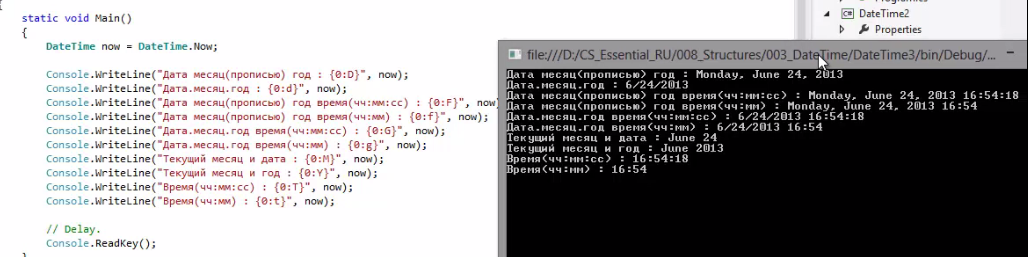


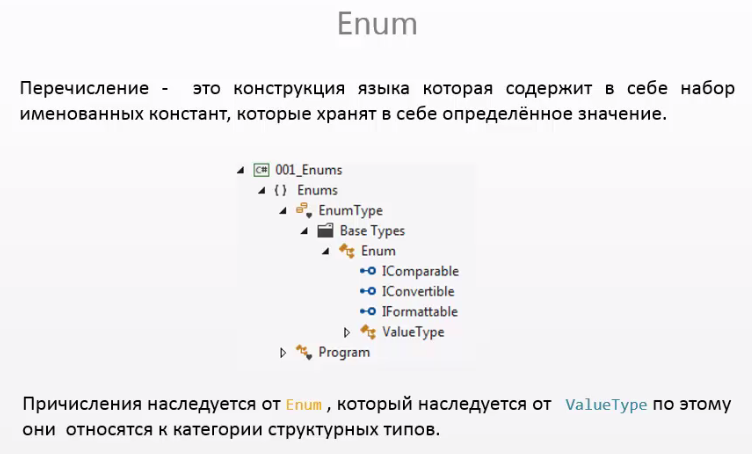
Массив приводим к ссылочному типу Object



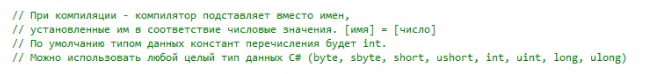
DateTime

Форматированный вывод даты и времени

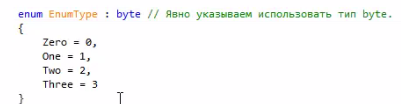


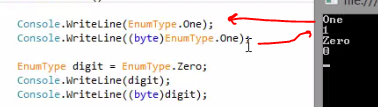


Использовать можно только 8 целочисленных типов и только их алиасы (псевдонимы)

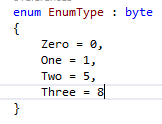


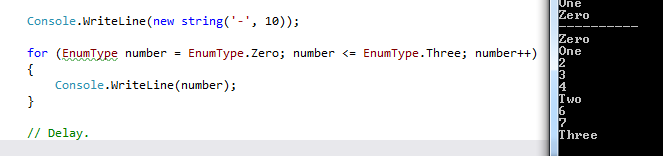
Создаем перечисление с именем EnumType в котором создаем четыре именованные константы типа byte и присваиваем им значение



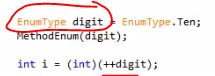


Можно пробежать циклом по перечислению, где пропуск (в данном примере от 1-5 и 5-8)будут подставляться целые числа

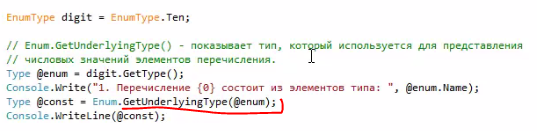




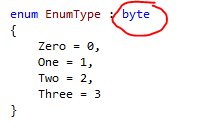
Копию можем увеличить применив к ней операцию инкремента



Определяет какой тип значения используется для перечисления

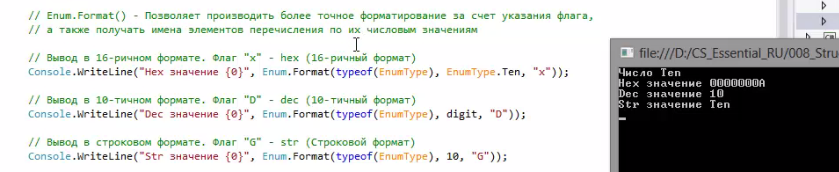




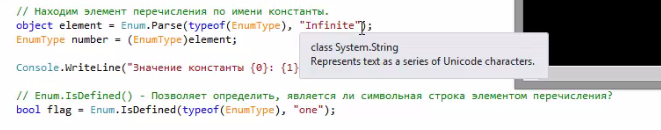


Enum.Format выводит значение именованных констант

1. Вариант принимаем саму константу типа
2. Переменная копия константы
3. Передаем число



Поиск перечисления по имени и проверка на наличие константы в перечислении



Получить все элементы перечисления типа

