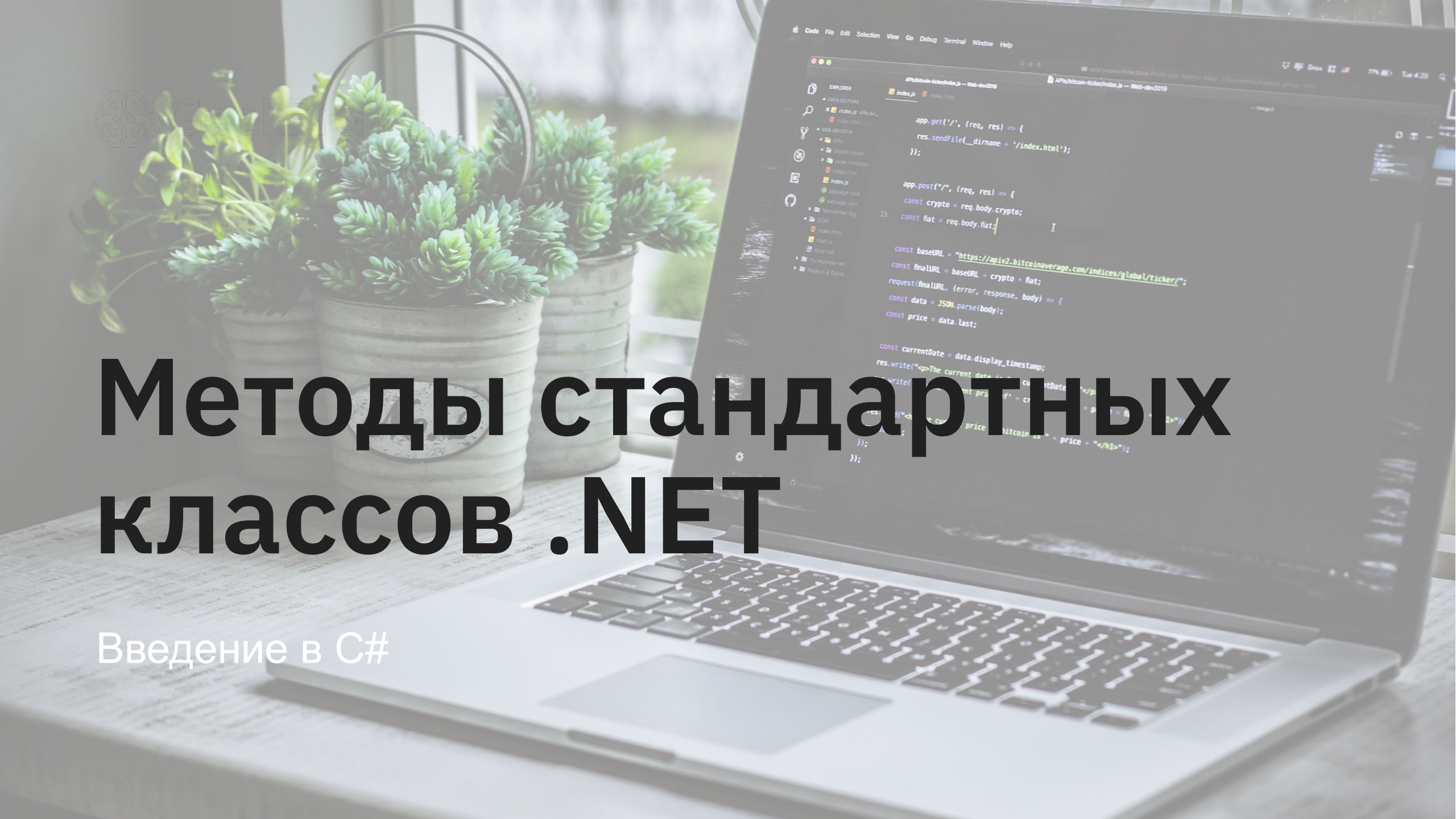


Методы стандартных классов .NET

Введение в C#

План урока

1. Функции и методы.
2. Память в C#.
3. Рекурсивный вызов метода.

Функции и методы

Что такое функции?

Функция — это фрагмент программного кода, к которому можно обратиться из другого места программы. Функция может содержать входные аргументы, с помощью которых мы можем изменять её поведение, а также передавать ей необходимые значения. Также функция может содержать результирующее значение.

Пример функции: функция Main

Назначение функций

Функции в программировании решают ряд задач:

1. Позволяют разбить сложную программу на набор маленьких функций, с которыми проще взаимодействовать (вносить правки и разбираться в их работе).
2. Позволяют переиспользовать код и избежать излишнего дублирования кода (о дублировании кода мы поговорим далее в этом уроке).

Пример: в домашних заданиях часто был одинаковый код валидации ввода с консоли - его можно вынести в отдельную функцию.

Метод

Метод - это название функции в объектно ориентированном программировании. Важный момент - в C# все функции описываются внутри классов.

В дальнейшем мы будем использовать именно этот термин.

Объявление методов

```
static string GetUserName()
{
    Console.WriteLine("Привет! Как тебя зовут?");
    string userName = Console.ReadLine();
    return userName;
}

static string MakeGreeting(string userName)
{
    return $"Привет, {userName}!";
}
```

Сигнатура метода

Сигнатура метода включает в себя имя метода, тип возвращаемого значения, входные параметры и различные необязательные модификаторы, которые мы разберем позднее.

Пример:

```
static string MakeGreeting(string userName)
```

Кортеж

Кортеж - это упорядоченный набор переменных, фиксированной длины.

Пример:

(string name, int age)

Назначение кортежа

Бывают ситуации, когда из метода следует вернуть несколько значений. На практике в таких случаях чаще применяют классы, но также можно воспользоваться возможностью языка C# под названием named tuple (именованный кортеж)

Принцип DRY

Существует принцип DRY (Don’t Repeat Yourself), который сводится к тому, что каждая единица знания должна быть описана в коде единожды.

Пример: только один метод должен знать, как форматировать данные пользователя, а все остальные части приложения должны использовать данный метод.

Стандартные методы классов

В .NET существует класс Object, который является базовым для всех классов .NET. Это означает, что любой другой класс обладает теми же методами, что и класс Object. Например, класс Object определяет метод ToString для строкового представления содержимого объекта. Когда мы выводим на консоль объекты, отличные от строковых, у данных объектов вызывает метод ToString. Он всегда определен в любом объекте, так как описан в Object.

Тернарный оператор

Тернарный оператор - это оператор, обладающий тремя операторами и работающий следующим образом:

Если значение первого операнда - истина, возвращается значение и выполняется второй операнд. В противном случае, выполняется третий операнд. При этом важно помнить, что типы второго и третьего операндов должны совпадать.

Запись:

```
<условие> ? <операнд2> : <операнд3>
```

Память в C#

Stack:

- имеет фиксированный малый размер;
- хранит локальные переменные, аргументы функций;

Heap:

- размер ограничен объемом ОЗУ;
- хранит данные ссылочного типа, ресурсы, и прочее;

Stack

Стек (stack) — специальная область памяти компьютера, в которой хранятся локальные переменные, созданные каждым методом (включая Main). Название этой области происходит из-за использования одноименной структуры данных. Стек — это структура данных, организованная по принципу LIFO (last in, first out).

Stack Frame

Стек состоит из кадров (stack frame). Каждый раз, когда мы вызываем какой-нибудь метод, в области стека создается новый стековый кадр, в который по мере выполнения метода будут помещаться локальные переменные, объявленные внутри этого метода. Затем каждый раз, когда метод завершает работу, стековый кадр удаляется из памяти, а следовательно, и все переменные, объявленные внутри метода.

Размер стека

Обычно для 32-битных приложений размер стека составляет 1 Мб, а для 64-битных — 4 Мб.

Переполнение стека

Достаточно распространенная ошибка, когда стек переполняется, например, в результате рекурсии. В этом случае программа экстренно завершает работу. Проблема этой ошибки, в том, что ее невозможно корректно обработать. Мы не можем записать данные из памяти на диск, т.к. не сможем вызвать ни одного метода.

Heap

Heap (куча) - это динамическая область памяти. Когда какому-либо приложению требуется память, запрашиваемый размер памяти выделяется под данное приложение из общей кучи. В отличие от стека, в котором кадры расположены друг за другом, в куче нет строгости в порядке выделения памяти. Мы просто запрашиваем определённый объём памяти и получаем его в каком-либо участке оперативки.

Размер кучи

Размер кучи достаточно большой, но он ограничен. Как минимум размером доступной нам оперативной памяти. Для 32-битных систем - это всего 1,5ГБ. Будьте аккуратны - не загружайте в память большие файлы целиком.

Типы данных

ValueType:

- целочисленные типы;
- символы.

ReferenceType:

- объекты: строки, массивы, и прочее.

Value type

- Как правило хранятся в стеке
- Размер известен заранее
- Значение переменной хранится в области памяти которая под нее выделена
- При вызове метода в метод передается копия переменной
- При объявлении переменной место под нее выделяется сразу

Примеры: char, int, bool, long, double, decimal, float

Reference type

- Хранятся в куче
- Могут занимать большой объем
- В стеке хранится ссылка на адрес в памяти
- При вызове метода в него передается ссылка на объект
- При объявлении переменной место под нее НЕ выделяется сразу

Примеры: string, Array, object

Сравнение типов

Value type

Сравниваем значения

Reference type

Сравниваем ссылки

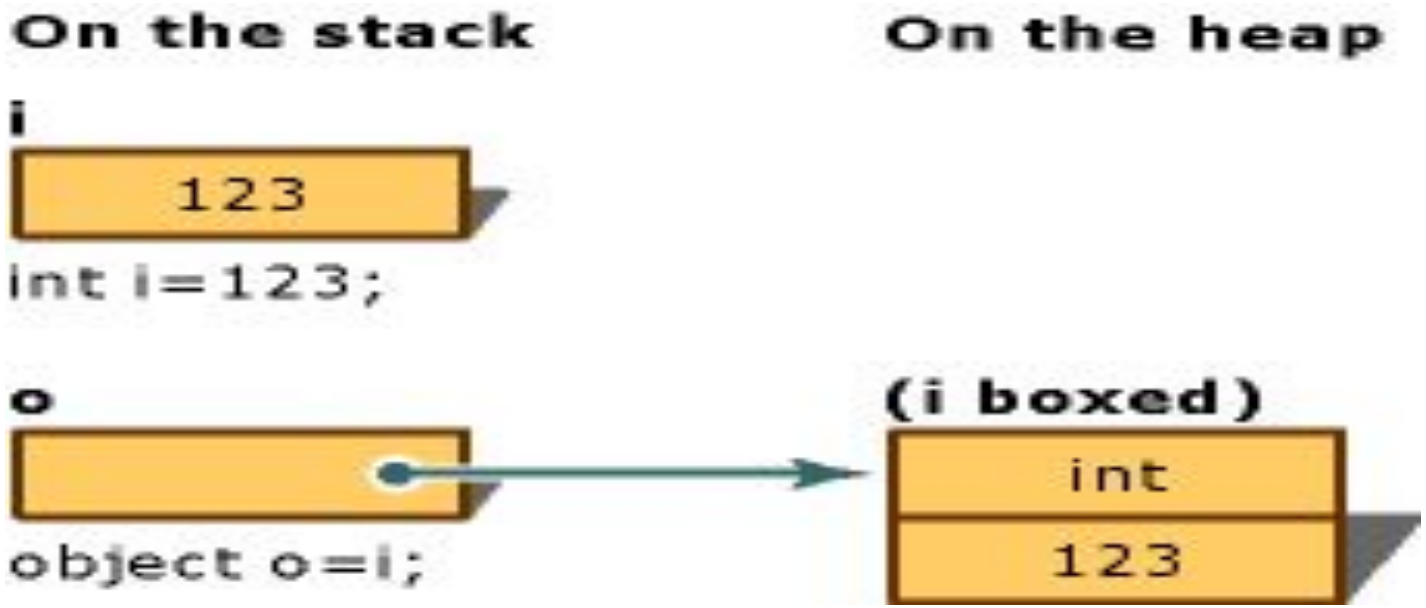
Интернирование строк

На этапе сборки программного кода, компилятор создает пул (своего рода “базу”) всех встречающихся в коде строк. При этом, если строка является результатом какого-либо простого выражения, то такие выражения вычисляются на месте и их значение также попадает в пул интернированных строк. Все строки в нем уникальны, и все переменные, имеющие данное значение, указывают на адрес данной строки в памяти.

Упаковка и распаковка Value type

В некоторых случаях нам может понадобиться обрабатывать типы значений наряду со ссылочными типами. В таких случаях мы можем применить к ним операцию упаковки — процесс преобразования типа значения в тип `object`:

```
int i = 123;
object o = i;
```



Ключевое слово ref

Ключевое слово `ref` применяется в тех случаях, когда нам нужно передать или изменить значение по ссылке.

Возвращение ссылок на значения

Ключевое слово `ref` позволяет нам возвращать ссылки на возвращаемое значение.

Ключевое слово out

Ключевое слово `out` схоже по использованию с ключевым словом `ref`, за исключением того, что при использовании `ref` перед передачей переменную необходимо инициализировать. Одним из вариантов использования `out` является передача наружу дополнительных сведений для булевых методов.

Ключевое слово params

Ключевое слово `params` позволяет методу принимать массив однотипных аргументов, при этом во время вызова метода мы будем указывать значения через запятую так, как если бы это были отдельные аргументы.

Рекурсивный вызов метода

```
1 static int GetFactorial(int number)
2 {
3     if (number == 1) // терминальное условие
4     {
5         return number;
6     }
7     return number * GetFactorial(number - 1); // рекурсивный вызов
8 }
```