1 вводная лекция.

План:

1. История языков программирования и процесса разработки ПО с начала (1950-е) до наших дней. Машинный код + перфокарты, языки ассемблера, ForTran/ALGOL, LISP, PASCAL, COBOL, C, SmallTalk, C++, Basic.
2. Подходы в разработке ПО. Спагетти-код, структурный подход, ООП, функциональщина.
3. Кроссплатформенность, переносимость, компиляция VS интерпретация и прочие причины возникновения Java.
4. Внезапно – C#. Следующий виток эволюции.
5. C# и .NET Framework

2 вводная лекция.

1. Какие платформы бывают.
2. Какие задачи встречаются.
3. Какие профессии нужны в IT?
4. Что нужно знать для становления junior .NET developer? И сколько времени это займет.
5. О чем мы расскажем на курсах – план занятий по шарпу.
6. Обзор других технологий и зарплат.



1 вводная лекция.

**1. История языков программирования и процесса разработки ПО с начала (1950-е) до наших дней. Машинный код + перфокарты, языки ассемблера, ForTran/ALGOL, LISP, PASCAL, COBOL, C, SmallTalk, C++, Basic.**

**2. Подходы в разработке ПО. Спагетти-код, структурный подход, ООП, функциональщина.**

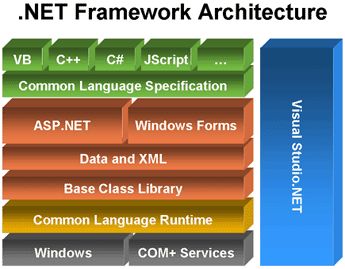
**3. Кроссплатформенность, переносимость, компиляция VS интерпретация и прочие причины возникновения Java.**

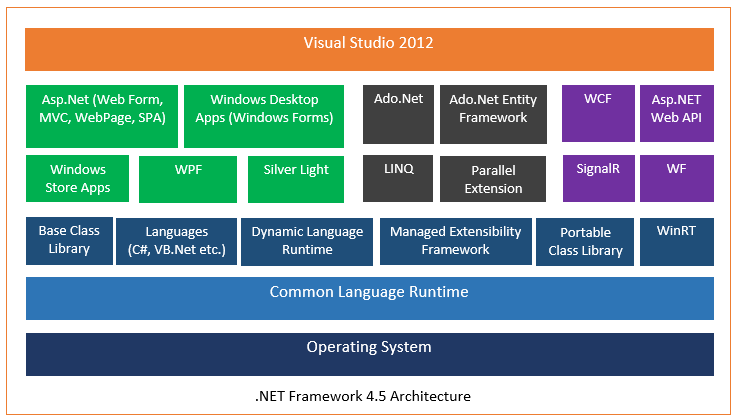
**4. C# и .NET Framework**

**.NET Framework** – платформа разработки, развертывания и выполнения распределенных приложений преимущественно в ОС Microsoft Windows. Она предоставляет объемный инструментарий для разработчиков, обеспечивает полную интеграцию языков, поддерживая межъязыковое наследование, обработку исключений и отладку. Программы, написанные на языках платформы .NET Framework запускаются не сами по себе, а в специальной среде (виртуальной машине) – **Common Language Runtime** – общеязыковая среда выполнения. Среда предоставляет такие функции, как управление памятью, отслеживание ошибок, проверка безопасности и другие.

В чем преимущество платформы .NET? Она позволяет разработчику уйти от большей части рутинной работы, автоматизируя ее, а именно – работа с памятью (выделение и сборка мусора), сборка приложения и другое. На платформе .NET Framework можно разрабатывать десктопные, web-приложения, приложения для мобильных устройств, игры (Unity + Hearthstone) и даже .

Далее на картинках можно увидеть приближенный взгляд на архитектуру платформы.





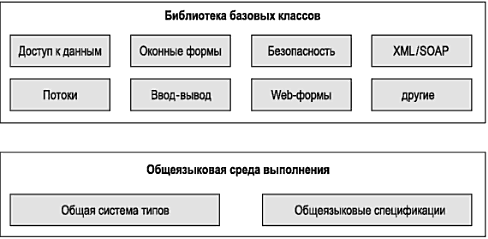
На самом нижнем уровне находится операционная система. Все, что выше - .NET Framework. Зависимости идут снизу-вверх.

Начинается он с **CLR** (**Common Language Runtime).** Над ней расположены **библиотеки базовых классов – BCL (Base Class Libraries)**. **BCL** не только инкапсулирует различные «примитивы», такие как потоки, файловый ввод-вывод, визуализация графики (и прочее), но и обеспечивает поддержку высокоуровневых сервисов. Над BCL расположены общеплатформенные технологии по работе с XML, данными из баз данных, LINQ-запросами, распределенными системами и сервисами. А над ними можно увидеть конкретные платформы для развертывания приложений - ASP.Net, desktop, WPF, Windows Phone и т.д.

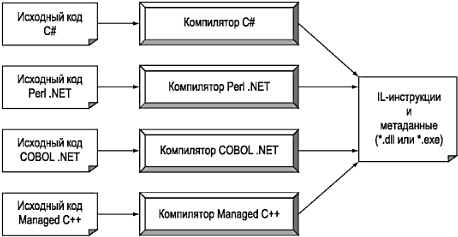
Помимо **CLR** и **BCL** есть еще две важные составляющие .NET Framework:

**Common Type System** – общеязыковая система типов. Спецификации CTS полностью описывают все возможные типы данных и программные конструкции, поддерживаемые средой выполнения, указывают, как эти элементы могут взаимодействовать друг с другом и как они представляются в формате метаданных .NET.

**Common Language Specification** – общеязыковые спецификации определяют подмножество общих типов и конструкций, понятных всем языкам, совместимым с платформой .NET. Типы данных, не выходящие за рамки CLS, будут поддерживаться всеми языками платформы.



Как же устроено межъязыковое взаимодействие? Программный код, написанный на любом языке, который поддерживается платформой .NET, компилируется дважды: сперва компилятор конкретного языка переводит код в промежуточный язык – **Common Intermediate Language (IL)–** а затем IL-инструкции компилируются под конкретный процессор. Выполняющий эту задачу компилятор называют JIT-компилятором (just-in-time).



Важно понимать, что двойная компиляция влияет на производительность. Если вы разрабатываете ПО для старого железа с сильно ограниченным объемом памяти, платформа .NET будет уступать C++. Напротив, на компьютерах хотя бы средней производительности имеются достаточные для .NET Framework ресурсы.

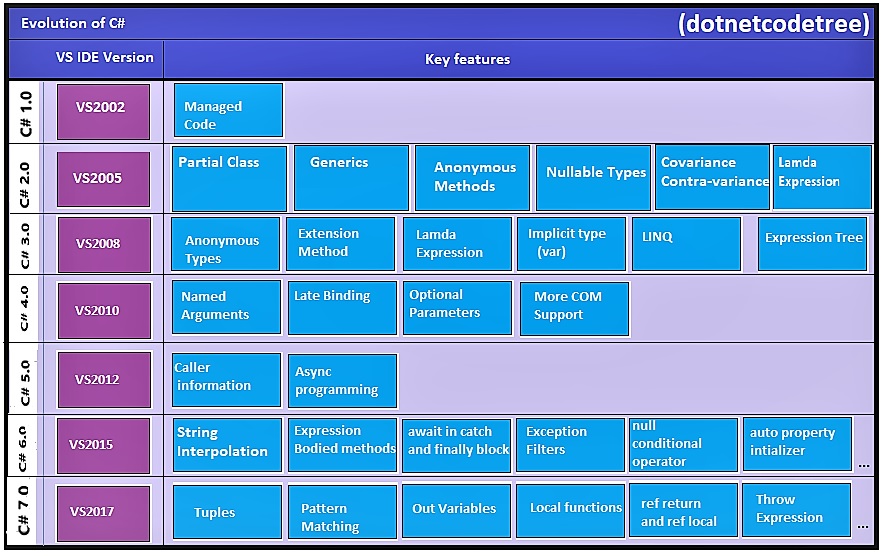
Далее на картинке более детально показан механизм выполнения любой программы, написанной на/для платформы .NET.

Язык C#.

Язык C# был разработан корпорацией Microsoft в конце 90-х годов как часть общей стратегии .NET. Впервые он был выпущен в виде альфа-версии в середине 2000 года, а официальный релиз первой версии состоялся в январе 2002 года. Главным разработчиком C# был Андерс Хейльсберг — один из ведущих в мире специалистов по языкам программирования, в 80-е годы он был автором очень удачной и имевшей большое значение разработки — языка Turbo Pascal, изящная реализация которого послужила образцом для создания всех последующих компиляторов.

Итак, C# - высокоуровневый, компилируемый (дважды) язык программирования высокого уровня, спроектированный и разработанный для взаимодействия с платформой .NET и призванный устранить множество неудобств, с которыми сталкивались разработчики/программисты в 80-90-е годы. Первая версия языка была во многом похожа на клон существующей уже десятый год Java, но затем язык стал стремительно развиваться, комьютини росло, появлялись новые фреймворки, порой даже кроссплатформенные. На текущий момент последней выпущенной версией языка является C# 7.0. Анонсирован выпуск версий 7.1 и 7.2.

На картинке ниже можно увидеть список изменений в каждой версии языка:



Вставить сюда примеры кода и демо.

.Net Framework сегодня. Кроссплатформенность, новый компилятор, .Net Standart и прочее.