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.

**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.**

Для управления первыми вычислительными машинами использовались перфокарты. Перфока́рта — [носитель информации](https://dic.academic.ru/dic.nsf/ruwiki/4843), предназначенный для использования системами автоматической обработки данных. Сделанная из тонкого [картона](https://dic.academic.ru/dic.nsf/ruwiki/133890), перфокарта представляет информацию наличием или отсутствием отверстий в определённых позициях карты (могу принести из института, Давыдкина на них же пишет примеры). Информация, которую они содержат, и есть программа – комбинации отверстий или их отсутствия, которые обозначают определённые действия, которые машина должна выполнить.

С развитием компьютерной техники появился [машинный язык](https://dic.academic.ru/dic.nsf/ruwiki/1029856), с помощью которого [программист](https://dic.academic.ru/dic.nsf/ruwiki/11604) мог задавать команды, оперируя с ячейками памяти, полностью используя возможности машины.

Машинный язык – это система команд конкретной [вычислительной машины](https://dic.academic.ru/dic.nsf/ruwiki/616), который [интерпретируется](https://dic.academic.ru/dic.nsf/ruwiki/15224) непосредственно [микропроцессором](https://dic.academic.ru/dic.nsf/ruwiki/1187537) или [микропрограммами](https://dic.academic.ru/dic.nsf/ruwiki/224644) данной вычислительной машины.

Однако использование большинства компьютеров на уровне машинного языка затруднительно. **(**Например, для организации чтения блока данных с гибкого диска программист может использовать 16 различных команд, каждая из которых требует 13 параметров, таких как номер блока на диске, номер сектора на дорожке и т. п. Когда выполнение операции с диском завершается, контроллер возвращает 23 значения, отражающие наличие и типы ошибок, которые надо анализировать.**)**

«Слова» на машинном языке называются инструкции, каждая из которых представляет собой одно элементарное действие для центрального процессора, такое, например, как считывание информации из ячейки памяти.

В случае, когда нужно иметь эффективную программу, вместо машинных языков используются близкие к ним машинно-ориентированные языки — [ассемблеры](https://dic.academic.ru/dic.nsf/ruwiki/5273). Команды языка ассемблера один в один соответствуют командам процессора и, фактически, представляют собой удобную символьную форму записи команд и аргументов. Также язык ассемблера обеспечивает связывание частей программы и данныx через метки, выполняемое при ассемблировании (для каждой метки высчитывается адрес, после чего каждое вхождение метки заменяется на этот адрес). Но даже работа с ассемблером достаточно сложна и требует специальной подготовки.

В 1954 году был создан первый язык высокого уровня — [Фортран](https://dic.academic.ru/dic.nsf/ruwiki/9042) ([англ.](https://dic.academic.ru/dic.nsf/ruwiki/6161) FORTRAN - FORmula TRANslator). Языки высокого уровня имитируют естественные языки, используя некоторые слова разговорного языка и общепринятые математические символы. Эти языки более удобны для человека, с помощью них, можно писать программы до нескольких тысяч строк длиной. Однако легко понимаемый в коротких программах, этот язык становился нечитаемым и трудно управляемым, когда дело касалось больших программ. Решение этой проблемы пришло после изобретения языков структурного программирования ([англ.](https://dic.academic.ru/dic.nsf/ruwiki/6161) structured programming language), таких как [Алгол](https://dic.academic.ru/dic.nsf/ruwiki/11120)(1958), [Паскаль](https://dic.academic.ru/dic.nsf/ruwiki/2495)(1970), [Си](https://dic.academic.ru/dic.nsf/ruwiki/7630)(1972).

В Алголе появилось представление о программе не как о свободной последовательности команд, а как о блочной структуре, состоящей из чётко описанных и отделённых друг от друга частей. Основной блок программы на Алголе — это сама главная программа. Она содержит свою исполняемую часть, заключённую в блок, ограниченный парой ключевых слов begin и end, а также описания подпрограмм. Каждая подпрограмма — это программа в миниатюре, имеющая собственные, описанные внутри неё данные, однозначно определённый интерфейс в виде имени и списка формальных параметров, и блок кода.

Язык Лисп был предложен в 1960 году и ориентирован на разработку программ для решения задач не численного характера. Английское название этого языка — LISP является аббревиатурой выражения LISt Processing (обработка списков) и хорошо подчеркивает основную область его применения. Понятие «список» оказалось очень емким.

В виде списков удобно представлять алгебраические выражения, графы, элементы конечных групп, множества, правила вывода и многие другие сложные объекты. Списки являются наиболее гибкой формой представления информации в памяти компьютеров. Неудивительно поэтому, что удобный язык, специально предназначенный для обработки списков, быстро завоевал популярность.

[Кобол](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B1%D0%BE%D0%BB) был разработан в 1959 году и предназначался прежде всего для написания программ для разработки бизнес приложений, а так же для работы в экономической сфере. Спецификация языка была создана в 1959 году. Создатели языка ставили своей целью сделать его машинонезависимым и максимально приближенным к естественному английскому языку. Обе цели были успешно достигнуты; программы на COBOL считаются понятными даже неспециалистам, поскольку тексты на этом языке программирования не нуждаются в каких-либо специальных комментариях (самодокументирующиеся программы).

Бе́йсик (от BASIC, сокращение от [англ.](https://dic.academic.ru/dic.nsf/ruwiki/6161) Beginner’s All-purpose Symbolic Instruction Code — универсальный код символических инструкций для начинающих)— семейство высокоуровневых [языков программирования](https://dic.academic.ru/dic.nsf/ruwiki/1465). Был разработан в [1963 году](https://dic.academic.ru/dic.nsf/ruwiki/680). Язык создавался как инструмент, с помощью которого студенты-непрограммисты могли самостоятельно создавать компьютерные программы для решения своих задач. Получил широкое распространение в виде различных диалектов, прежде всего как язык для [домашних компьютеров](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BC%D0%B0%D1%88%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80). К настоящему моменту претерпел существенные изменения, значительно отойдя от характерной для первых версий простоты, граничащей с примитивизмом, и превратившись в достаточно ординарный язык высокого уровня с типичным набором возможностей.

Язык Паскаль был создан [Никлаусом Виртом](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82,_%D0%9D%D0%B8%D0%BA%D0%BB%D0%B0%D1%83%D1%81) в 1968—1969 годах. Язык назван в честь французского математика, физика, литератора и философа [Блеза Паскаля](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D1%8C,_%D0%91%D0%BB%D0%B5%D0%B7), который создал первую в мире механическую машину, складывающую два числа. Первая публикация Вирта о языке датирована 1970 годом; представляя язык, автор в качестве цели его создания указывал построение небольшого и эффективного языка, способствующего хорошему стилю программирования, использующему [структурное программирование](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и структурированные данные.

Си — [стандартизированный](https://dic.academic.ru/dic.nsf/ruwiki/57195) [процедурный](https://dic.academic.ru/dic.nsf/ruwiki/40931) [язык программирования](https://dic.academic.ru/dic.nsf/ruwiki/1465), разработанный в начале [1970-х годов](https://dic.academic.ru/dic.nsf/ruwiki/708643). Си был создан для использования в [операционной системе (ОС)](https://dic.academic.ru/dic.nsf/ruwiki/2566) UNIX, но впоследствии портирован на многие другие операционные системы и стал одним из самых используемых языков программирования. Си ценят за его эффективность; он является самым популярным языком для создания [системного программного обеспечения](https://dic.academic.ru/dic.nsf/ruwiki/4850). Его также часто используют для создания [прикладных программ](https://dic.academic.ru/dic.nsf/ruwiki/4631). Несмотря на то, что Си не разрабатывался для новичков, он активно используется для обучения [программированию](https://dic.academic.ru/dic.nsf/ruwiki/8183). В дальнейшем синтаксис языка Си стал основой для многих других языков (см.: [Си-подобный синтаксис](https://dic.academic.ru/dic.nsf/ruwiki/249655)). Для языка Си характерны лаконичность, современный набор конструкций управления потоком выполнения, [структур данных](https://dic.academic.ru/dic.nsf/ruwiki/114700) и обширный набор операций.

Являясь одним из самых популярных языков программирования, C++ широко используется для разработки программного обеспечения. Область его применения включает создание [операционных систем](https://dic.academic.ru/dic.nsf/ruwiki/2566), разнообразных прикладных программ, [драйверов](https://dic.academic.ru/dic.nsf/ruwiki/49331) устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр). Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. C++ оказал огромное влияние на другие языки программирования, в первую очередь на [Java](https://dic.academic.ru/dic.nsf/ruwiki/1315) и [C#](https://dic.academic.ru/dic.nsf/ruwiki/1112).

Синтаксис C++ унаследован от языка [C](https://dic.academic.ru/dic.nsf/ruwiki/7630)и. Одним из принципов разработки было сохранение совместимости с Cи. Тем не менее, C++ не является в строгом смысле надмножеством Cи; множество программ, которые могут одинаково успешно транслироваться как [компиляторами](https://dic.academic.ru/dic.nsf/ruwiki/6953) C, так и компиляторами C++, довольно велико, но не включает все возможные программы на Cи.

Получилось чот много, если сказать несколько слов о каждом ЯП

Наверное надо куски кода повставлять ещё в качестве примеров

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

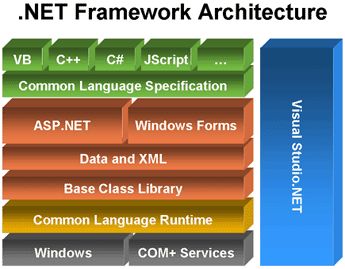
**3. Кроссплатформенность, переносимость, компиляция VS интерпретация и прочие причины возникновения Java.**

**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 этой платформы 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 - выделение памяти, ее освобождение (что было обязательно в C++). Также на ней можно разрабатывать и десктопные, и web-приложения, и приложения для мобильных устройств, и игры.

Далее на картинке можно увидеть приближенный взгляд на архитектуру платформы.

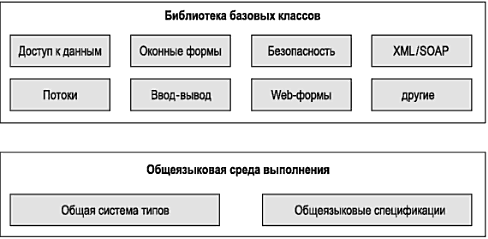


Все, что находится выше **Common Language Runtime**, контролируется средой, а та, в свою очередь, контролируется непосредственно операционной системой. Помимо среды выполнения, платформа предоставляет общие для всех своих языков технологии (выделены оранжевым), общеязыковую систему типов и общеязыковые спецификации.

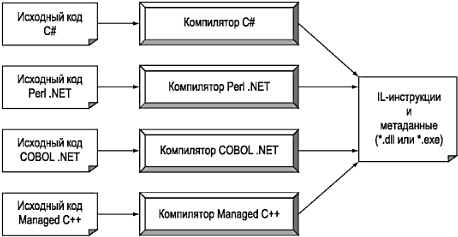
**Common Type System** – общеязыковая система типов. Спецификации CTS полностью описывают все возможные типы данных и программные конструкции, поддерживаемые средой выполнения, указывают, как эти элементы могут взаимодействовать друг с другом и как они представляются в формате метаданных .NET.

**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.

Также платформа предоставляет библиотеку базовых классов, доступную всем языкам программирования .NET. Она не только инкапсулирует различные «примитивы», такие как потоки, файловый ввод-вывод, визуализация графики (и прочее), но и обеспечивает поддержку высокоуровневых сервисов.



Как же устроено межъязыковое взаимодействие? Код на любом языке .NET компилируется дважды: сперва компилятор конкретного языка переводит код в промежуточный язык – **Common Intermediate Language –** 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. 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.

