1. **.NET Framework** — [программная платформа](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0#.D0.BA.D1.80.D0.BE.D1.81.D1.81.D0.BF.D0.BB.D0.B0.D1.82.D1.84.D0.BE.D1.80.D0.BC.D0.B5.D0.BD.D0.BD.D0.BE.D0.B5_.D0.BF.D1.80.D0.BE.D0.B3.D1.80.D0.B0.D0.BC.D0.BC.D0.BD.D0.BE.D0.B5_.D0.BE.D0.B1.D0.B5.D1.81.D0.BF.D0.B5.D1.87.D0.B5.D0.BD.D0.B8.D0.B5), выпущенная компанией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) в [2002 году](https://ru.wikipedia.org/wiki/2002_%D0%B3%D0%BE%D0%B4). Основой платформы является общеязыковая среда исполнения [Common Language Runtime (CLR)](https://ru.wikipedia.org/wiki/Common_Language_Runtime), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.
2. Центральной частью каркаса .NET является его общеязыковая исполняющая среда, известная как Common Language Runtime (CLR) или .NET runtime. Код, выполняемый под управлением CLR, часто называют управляемым кодом. С точки зрения программирования под термином ***исполняющая среда*** может пониматься коллекция внешних служб, которые требуются для выполнения скомпилированной единицы программного кода.
3. FCL –– объектно-ориентированная библиотека классов, интерфейсов, и системы типов(типов-значений).
4. Сборка.
5. ***IL*** *(Intermediate Language) - код, содержащий набор инструкций, не зависящих от платформы. Иными словами, после компиляции исходного кода он преобразуется не в код для какой-то определенной платформы, а в промежуточный код на языке IL.*

***JIT-компиляция***

Процесс компиляции CIL-инструкций в соответствующий машинный код, при котором JIT-компилятор будет помещать результаты в кэш в соответствии с тем, как того требует целевая операционная система.

1. **Common Type System** (сокр. CTS, [рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Общая система типов*) — часть [.NET Framework](https://ru.wikipedia.org/wiki/.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](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и специальные значения типов представлены в компьютерной памяти  
   1. **тип System.Object позволяет сделать следующее:   
       сравнить два экземпляра на равенство;   
       получить хеш-код экземпляра;  
       запросить фактический тип экземпляра;   
       выполнить поверхностное (поразрядное) копирование экземпляра;   
       получить строковое представление текущего состояния экземпляра.**
2. MSCorLib.dll — это особый файл в том смысле, что в нем находятся все основные типы, представляющие байты, целочисленные, символьные, строковые и другие значения. В действительности, эти типы используются так часто, что компилятор C# ссылается на эту сборку автоматически
3. ***Манифест сборки*** *(assembly manifest) - это внутренняя часть сборки, которая позволяет ей быть самоописанной. Assembly manifest позволяет идентифицировать сборку, указывает файлы, которые включаются в реализацию сборки, описывает типы и ресурсы, используемые в сборке, указывает зависимости от других сборок, а также набор прав доступа, которые необходимы сборке для корректной работы. Эта информация используется во время выполнения для разрешения ссылок, проверку корректности версий, проверку целостности загруженных сборок.*
4. ***GAC*** *— это глобальный кэш сборок. В нем хранятся совместно используемые сборки. Обычно это каталог С:\Windows\Assembly\GAC. Этот каталог имеет определенную структуру, в котором хранятся подкаталоги, имена которых сгенерированы по определенному алгоритму. В GAC можно поместить только сборки со строгими именами. Для того, чтобы поместить сборку в GAC, используют специальный инструмент GACUtil.exe, который знает всю внутреннюю структуру GAC и может генерировать имена подкаталогов надлежащим образом. Регистрировать в GAC сборки необходимо для того, чтобы избежать конфликтов имен сборок. Приведем пример: две компании выпустили сборку и назвали ее одним именем Calculus. Если мы скопируем эту сборку в каталог, в котором уже находится сборка с таким же именем, то мы затрем сборку, которая ранее могла использоваться каким-то приложением. Это приложение с новой сборкой теперь работать не сможет. Решением этой проблемы будет регистрация этих двух сборок в GAC, в котором для каждой будет создан отдельный каталог.  
   КАТАЛОГ, РАЗМЕЩАЮЩИЙ В СЕБЕ БОЛЬШИНСТВО СБОРОК .Net Framework.*
5. **Управля́емый код** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *managed code*) — термин, введённый фирмой [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), для обозначения кода [программы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), исполняемой под «управлением» [виртуальной машины](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) [.NET](https://ru.wikipedia.org/wiki/.NET_Framework)[[1]](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4#cite_note-Abrams-1)[[2]](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4#cite_note-Abrams-ru-2)[[3]](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4#cite_note-3) — [Common Language Runtime](https://ru.wikipedia.org/wiki/Common_Language_Runtime) или [Mono](https://ru.wikipedia.org/wiki/Mono). При этом [машинный код](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) называется **неуправля́емым кодом** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *unmanaged code*).
6. В каждом приложении на C# должен присутствовать один метод Main, указывающий точку начала выполнения программы. В языке C# Main пишется с прописной буквы
7. Директива using разрешает использование типов в пространстве имен, поэтому уточнение использования типа в этом пространстве имен не требуется.  
   Также директива using позволяет создавать псевдонимы пространства имен или типа
8. Главное отличие - это где храниться значение переменной. Для ссылочных типов - это куча, для значимых - это стек. При этом, ссылочная переменная (сама ссылка) храниться на стеке. Значимыми могут быть только экземпляры структур, в противоположность им ссылочные - это экземпляры классов. Все встроенные типы (int, double и т.д.) - это структуры
9. Динамическую переменную от переменной, объявленной неявно или через ссылку System.Object, значительно отличает то, что она не является строго типизированной. Другими словами, динамические данные не типизированы статически. Для компилятора C# это выглядит так, что элемент данных, объявленный с ключевым словом dynamic, может получить какое угодно начальное значение, и на протяжении времени его существования это значение может быть заменено новым (и возможно, не связанным с первоначальным).
10. начиная с версии C# 3.0, компилятору предоставляется возможность самому определить тип локальной переменной, исходя из значения, которым она инициализируется. Такая переменная называется неявно типизированной  
    Неявно типизированная переменная объявляется с помощью ключевого слова **var** и должна быть непременно инициализирована
11. Чтобы превратить значимый тип в тип, допускающий значение null, нужно при объявлении переменной непосредственно после указания ее (значимого) типа поставить **знак ?**:
12. **https://professorweb.ru/my/csharp/charp\_theory/level4/4\_7.php**
13. Основная особенность NULLа заключается в том, что он не равен ничему, даже другому NULLу. С ним нельзя сравнить какое-либо значение с помощью любых операторов: =, <, >, like… Даже выражение NULL != NULL не будет истинным, ведь нельзя однозначно сравнить одну неизвестность с другой. Кстати, ложным это выражение тоже не будет, потому что при вычислении условий Oracle не ограничивается состояниями ИСТИНА и ЛОЖЬ. Из-за наличия элемента неопределённости в виде NULLа существует ещё одно состояние — НЕИЗВЕСТНО.
14. Строки не поддерживают операторы < и >. Вместо них для сравнения строк нужно использовать строковой метод CompareTo, который возвращает 1 если первая строка предшествует второй, -1  если вторая строка предшествует первой и 0 если строки равны:
15. *Данные, хранящиеся в классе System.String и есть неизменяемые (immutable). Класс System.StringBuilder разрабатывался так, чтобы над изменяемой строкой можно было проделать множество операций. То есть при каждой операции над объектом класса System.String происходит перенос данных в новую область памяти, что влияет на производительность программы.*
16. **кортежи (tuple)** могут комбинировать объекты различных типов
17. Локальные функции представляют функции, определенные внутри других методов. Локальные функции доступны начиная с версии C# 7.0 (Visual Studio 2017).

1. Что такое .Net Framework и из чего он состоит? 2. Поясните, что такое CLR-среда. 3. Что такое FCL? 4. Какая наименьшая исполнимая единица в .NET? 5. Что такое IL? 6. Пояснить работу JIT-компилятора? 7. Что такое CTS (Common Type System)? 8. Какие аспекты поведения определяет тип System.Object? 9. Что находится в MSCorLib dll? 10. Что такое частные и общие сборки? 11. Что такое assembly manifest? 12. Что такое GAC? 13. Чем managed code отличается от unmanaged code 14. Как и для чего определен метод Main? 15. Варианты использования директивы using( using Directive ) в C#. 16. Как связаны между собой сборки и пространства имен? 17. Что такое примитивные типы данных? Перечислите их. 18. Что такое ссылочные типы? Какие типы относятся к ним? 19. Какие типы относятся к типам-значениям? 20. В чем отличие между ссылочными и значимыми типами данных? 21. Что такое упаковка и распаковка значимых типов? 22. Для чего используется тип dynamic? 23. Что такое неявно типизированная переменная? 24. Для чего используют Nullable тип? 25. Как объявить строковый литерал? Какие операции можно выполнять со строкой? 26. Какие есть способы для задания и инициализации строк? 27. Какие методы есть у типа String? 28. В чем отличие пустой и null строки? 29. Как можно выполнить сравнение строк? 30. В чем отличие типов String и StringBuilder? 31. Поясните явные преобразования переменных с помощью команд Convert. 32. Как выполнить консольный ввод/вывод? 33. Приведите примеры определения и инициализации одномерных и двумерных массивов. 34. Что такое ступенчатый массив? Как его задать? 35. Какие типы можно использовать в foreach? Приведите пример. 36. Что такое кортеж? Для чего и как он используется? 37. Что такое локальная функция?