1. **Что такое .Net Framework и из чего он состоит?**

**.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" \o "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" \o "Common Language Runtime), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

1. **Поясните, что такое CLR-среда.**

Название среды — общеязыковая среда выполнения (Common Language Runtime, CLR) — говорит само за себя: это среда выполнения, которая подходит для разных языков программирования. Основные возможности CLR (управление памятью, загрузка сборок, безопасность, обработка исключений, синхронизация) доступны в любых языках программирования, использующих эту среду. Например, при обработке ошибок среда выполнения опирается на исключения, а значит, во всех языках программирования, использующих эту среду выполнения, сообщения об ошибках передаются при помощи механизма исключений. Или, например, среда выполнения позволяет создавать программные потоки, а значит, во всех языках программирования, использующих эту среду, тоже могут создаваться потоки.

1. **Что такое FCL?**

Одним из компонентов .NET Framework является FCL (Framework Class Library) — набор сборок в формате DLL, содержащих несколько тысяч определений типов, каждый из которых предоставляет некоторую функциональность.

1. **Какая наименьшая исполнимая единица в .NET?**

сборка

1. **Что такое IL?**

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.

**6. Пояснить работу JIT-компилятора?**

Когда мы запускам [С#](http://dream-land.by/developer/csharp" \t "_blank) приложение то есть открываем **exe файл**, тогда начинает работать **CLR**.   
**CLR** исполняет каждую строчку [IL](http://dream-land.by/developer/csharp/sborka" \t "_blank) кода в 2 шага.   
**IL** (промежуточный код) находится в [сборке (exe файл)](http://dream-land.by/developer/csharp/sborka" \t "_blank)

**Шаг 1**

[**IL**](http://dream-land.by/developer/csharp/sborka) (промежуточный код) компилируется в комманды процессора (0001 0011 1101… ). Этим занимается **JIT компилятор** в [CLR](http://dream-land.by/developer/csharp/clr) среде.   
  
Вместо компиляции всего приложения , **JIT** (Just-In-Time) компилятор просто компилирует каждую порцию кода при вызове. Если промежуточный код однажды скомпилирован, то результирующий машинный исполняемый код сохраняется до момента завершения работы приложения, поэтому его перекомпиляция при повторных обращениях к нему не требуется.   
  
[В Microsoft аргументируют](https://docs.microsoft.com/ru-ru/previous-versions/visualstudio/visual-studio-2008/ht8ecch6(v=vs.90)" \t "_blank), что такой процесс более эффективен, чем компиляция всего приложения при запуске, поскольку высока вероятность того, что крупные фрагменты кода приложения на самом деле не будут выполняться при каждом запуске.

**Шаг 2**

Выполняются комманды процессора.

**7. Что такое CTS (Common Type System)?**

Поскольку типы занимают центральное место в CLR, компания Microsoft разработала формальную спецификацию CTS (Common Type System), которая описывает способ определения и поведение типов.

CTS также задает правила видимости типов и доступа к членам типа. Например, помечая тип как открытый (ключевое слово public), вы тем самым экспортируете этот тип, делая его видимым и доступным для любой сборки. С другой стороны, пометка типа на уровне сборки (ключевое слово internal в C#) делает его видимым и доступным для кода той же сборки. Таким образом, CTS устанавливает правила, по которым сборки формируют границу видимости типа, а CLR обеспечивает выполнение правил видимости.

Кроме того, CTS определяет правила, управляющие наследованием, работой виртуальных методов, сроком жизни объектов и т. д. Эти правила разрабатывались для выражения семантики, выражаемой средствами современных языков программирования. Собственно, вам вообще не придется изучать правила CTS как таковые, потому что выбранный вами язык предоставляет собственный синтаксис и правила работы с типами. Синтаксис конкретного языка преобразуется в IL, «язык» CLR, в процессе генерирования сборки на стадии компиляции.

**8. Какие аспекты поведения определяет тип System.Object?**

**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

**9. Что находится в MSCorLib.dll?**

MSCorLib.dll — специальный файл, в котором находятся все основные типы:

Byte, Char, String, Int32 и т. д. В действительности, эти типы используются так часто, что компилятор C# обращается к этой сборке (MSCorLib.dll) автоматически.

**10.Что такое частные и общие сборки?**

Частные находятся в каталоге программы, а общие в GAC

Частные доступны только одному приложению, общие доступны всем приложениям?

**11.Что такое assembly manifest?**

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.

**12.Что такое GAC?**

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.

**13.Чем managed code отличается от unmanaged code?**

Это означает, что CLR заботится об управлении памятью, о совместимости между платформами, о безопасности кода и так далее

**14.Как и для чего определен метод Main?**

Метод Main — это точка входа приложения C#. (Библиотекам и службам точка входа в виде метода Main не требуется.) Когда приложение запускается, первым вызывается именно метод Main.

**15.Варианты использования директивы using( using Directive ) в C#.**

* Для разрешения использования типов в пространстве имен, чтобы не нужно было квалифицировать использование типа в этом пространстве имен:

using System.Text;

* Для разрешения доступа к статическим членам и вложенным типам без необходимости квалифицировать доступ с помощью имени типа.

using static System.Math;

* Чтобы создать псевдоним для пространства имен или типа. Это называется директивой using static.

using Project = PC.MyCompany.Project;

**16.Как связаны между собой сборки и пространства имен?**

перед использованием using SomeNameSpace;, вы должны указать в проекте ссылку на сборку, в которой объявлены типы данного пространства имен. Но это никак не влияет на то, что сборки и пространства имен не имеют жесткой связи на уровне языка и платформы. Пространства имен - всего лишь агрегатная приставка к имени типа для явного указания принадлежности типа к определенной группе логически связанных типов, не более того. А вот типы, в отличие от пространств имен, имеют жесткую связь со сборкой в которой находятся, об этом можно подробнее почитать в первой ссылке.

**17.Что такое примитивные типы данных? Перечислите их**

Типы данных, которые поддерживаются компилятором напрямую, называются примитивными (primitive types); у них существуют прямые аналоги в библиотеке классов .NET Framework Class Library (FCL).

**18.Что такое ссылочные типы? Какие типы относятся к ним?**

Память для ссылочных типов всегда выделяется из управляемой кучи, а оператор C# new возвращает адрес в памяти, где размещается сам объект. При работе со ссылочными типами необходимо учитывать следующие обстоятельства, относящиеся к производительности приложения:

 -память для ссылочных типов всегда выделяется из управляемой кучи;

 -каждый объект, размещаемый в куче, содержит дополнительные члены, подлежащие инициализации;

 -незанятые полезной информацией байты объекта обнуляются (это касается полей);

 -размещение объекта в управляемой куче со временем инициирует сборку мусора.

Тип object

Тип string

Классы (class)

Интерфейсы (interface)

Делегаты (delegate)

**19.Какие типы относятся к типам-значениям?**

Целочисленные типы (byte, sbyte, char, short, ushort, int, uint, long, ulong)

Типы с плавающей запятой (float, double)

Тип decimal

Тип bool

Перечисления enum

Структуры (struct)

**20. В чем отличие между ссылочными и значимыми типами данных?**

Ссылочныым типам память выделяется из управляемой кучи, значимым -- из стека.

**21.Что такое упаковка и распаковка значимых типов?**

Для преобразования значимого типа в ссылочный служит упаковка (boxing). Распаковка – из ссылочного в значимый.

**22.Для чего используется тип dynamic?**

Тип является статическим типом, но объект типа dynamic обходит проверку статического типа. В большинстве случаев он функционирует, как тип object. Во время компиляции предполагается что элементы с типом dynamic поддерживают любые операции. Поэтому разработчику не нужно следить за тем, откуда объект получает свое значение — из интерфейса API COM, из динамического языка, такого как IronPython, из объектной модели DOM HTML, из отражения или из любого другого места программы. Но если код является недействительным, во время выполнения перехватываютсяошибки.

**23.Что такое неявно типизированная переменная?**

Локальные переменные можно объявлять без указания конкретного типа. Ключевое слово var указывает, что компилятор должен вывести тип переменной из выражения справа от оператора инициализации. Выведенный тип может быть встроенным, анонимным, определяемым пользователем либо типом, определяемым в библиотеке классов .NET Framework.

**24.Для чего используют Nullable тип?**

Как известно, переменная значимого типа не может принимать значение null; ее содержимым всегда является значение соответствующего типа. Именно поэтому типы и называют значимыми. Но в некоторых ситуациях такой подход создает проблемы. Например, при проектировании базы данных тип данных столбца можно определить как 32-разрядное целое, что в FCL соответствует типу Int32. Однако в столбце базы может отсутствовать значение, что соответствует значению null, и это — вполне стандартная ситуация. А это создаст проблемы при работе с базой данных средствами .NET Framework, ведь общеязыковая среда (CLR) не позволяет представить значение типа Int32 как null.

**25.Как объявить строковый литерал? Какие операции можно выполнять со строкой?**

"This is text"

Конкатенация, вставка подстроки, сравнение строк, разделение строк, обрезка, удаление, замена, смена регистра

**26.Какие есть способы для задания и инициализации строк?**

Создавать сроки можно, как используя переменную типа string и присваивая ей значение, так и применяя один из конструкторов класса String:

string s1 = "hello";

string s2 = null;

string s3 = new String('a', 6); // результатом будет строка "aaaaaa"

string s4 = new String(new char[]{'w', 'o', 'r', 'l', 'd'});

Конструктор String имеет различное число версий. Так, вызов конструктора new String('a', 6) создаст строку "aaaaaa". И так как строка представляет ссылочный тип, то может хранить значение null.

**27.Какие методы есть у типа String?**

Compare: сравнивает две строки с учетом текущей культуры (локали) пользователя

CompareOrdinal: сравнивает две строки без учета локали

Contains: определяет, содержится ли подстрока в строке

Concat: соединяет строки

CopyTo: копирует часть строки или всю строку в другую строку

EndsWith: определяет, совпадает ли конец строки с подстрокой

Format: форматирует строку

IndexOf: находит индекс первого вхождения символа или подстроки в строке

Insert: вставляет в строку подстроку

Join: соединяет элементы массива строк

LastIndexOf: находит индекс последнего вхождения символа или подстроки в строке

Replace: замещает в строке символ или подстроку другим символом или подстрокой

Split: разделяет одну строку на массив строк

Substring: извлекает из строки подстроку, начиная с указанной позиции

ToLower: переводит все символы строки в нижний регистр

ToUpper: переводит все символы строки в верхний регистр

Trim: удаляет начальные и конечные пробелы из строки

**29.Как можно выполнить сравнение строк?**

Метод Equal и операторы сравнения

**30.В чем отличие типов String и StringBuilder?**

Прежде всего, объект String представляет собой неизменяемую строку. Когда мы выполняем какой-нибудь метод класса String, система создает новый объект в памяти с выделением ему достаточного места. Удаление первого символа - не самая затратная операция. Однако когда подобных операций множество, а объем текста, для которого надо выполнить данные операции, также не самый маленький, то издержки при потере производительности становятся более существенными.

Чтобы выйти из этой ситуации во фреймворк .NET был добавлен новый класс StringBuilder, который находится в пространстве имен System.Text. Этот класс представляет динамическую строку.

**31.Поясните явные преобразования переменных с помощью команд Convert.**

Для явных преобразований необходим оператор приведения. Приведение требуется, когда при преобразовании может быть потеряна информация, или когда преобразование может завершиться неудачей по другим причинам. К типичным примерам относится числовое преобразование в тип, который имеет меньшую точность или меньший диапазон значений, а также преобразование экземпляра базового класса в производный класс.

**32.Как выполнить консольный ввод/вывод?**

Console.Writeline(); Console.Readline();

**33.Приведите примеры определения и инициализации одномерных и двумерных массивов**

int[] mas = new int[] {1,2,3};

int[,] mas1 = new int[,] {{1,2,3},{4,5,6}};

34.Что такое ступенчатый массив? Как его задать?

Ступенчатый массив представляет собой массив массивов, в котором длина каждого массива может быть разной. Следовательно, ступенчатый массив может быть использован для составления таблицы из строк разной длины.

**35.Какие типы можно использовать в foreach? Приведите пример**

Массивы, коллекции. Классы в которых реализован интерфейс System.Collections.IEnumerable.

**36.Что такое кортеж? Для чего и как он используется?**

Кортежный тип (tuple type) — это тип, который содержит коллекцию свойств, каким-то образом связанных друг с другом. Кортежи могут передаваться в качестве параметров в метод, могут быть возвращаемым результатом функции, либо использоваться иным образом.

**37.Что такое локальная функция?**

Локальные функции представляют функции, определенные внутри других методов. При использовании локальных функций следует помнить, что они не могут иметь модификаторов доступа (public, private, protected). Нельзя определить в одном методе несколько локальных функций с одним и тем же именем, даже если у них будет разный список параметров. Кроме того, не имеет значения, определены локальные функции до своего вызова или после.