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

Microsoft.NET (.NET Framework) – программная платформа. Содержит следующие основные компоненты: CLR (Common Language Runtime) – общеязыковая среда исполнения, виртуальная машина на которой исполняются все приложения, работающие в среде .NET. Реализация CLI VES компанией Microsoft. Компилятор JIT(Just in Time). MSIL(Microsoft IL) – реализация CLI CIL компанией Microsoft. FCL (Framework Class Library) – реализация CLI BCL компанией Microsoft. Можно рассматривать, как API CRL.

С точки зрения программирования, .NET Framework состоит из двух основных составляющих:

* общеязыковая исполнительная среда CLR (Common Language Runtime);
* библиотека базовых классов.

**2. Что такое CLR, FCL/BCL, CLI, IL?**

CLR (Common Language Runtime) – Среда Времени Выполнения или Виртуальная Машина. Обеспечивает выполнение сборки (управление памятью, загрузка сборок, безопасность, обработка исключений, синхронизация)

FCL (.NET Framework Class Library) – соответствующая CLS спецификации объектно-ориентированная библиотека классов, интерфейсов и системы типов (типов-значений)

**Base Class Library,** *BCL* — стандартная библиотека классов платформы «[.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework)». Программы, написанные на любом из языков, поддерживающих платформу .NET, могут пользоваться классами и методами BCL — создавать объекты классов, вызывать их методы, наследовать необходимые классы BCL и т. д.

**CLI** ([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) *Common Language Infrastructure*) — спецификация общеязыковой инфраструктуры. Спецификация CLI определяет, в частности, архитектуру исполнительной системы .NET — [CLR](https://ru.wikipedia.org/wiki/CLR) и сервисы, предоставляемые CLR выполняемым программам, классы, предоставляемые библиотекой [BCL](https://ru.wikipedia.org/wiki/Base_Class_Library), синтаксис и мнемонику общего промежуточного языка ([CIL](https://ru.wikipedia.org/wiki/MSIL)).

**Common Intermediate Language** (сокращённо **CIL**)  - «высокоуровневый ассемблер» виртуальной машины [.NET](https://ru.wikipedia.org/wiki/Microsoft_.NET). Промежуточный язык, разработанный фирмой [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft) для платформы [.NET Framework](https://ru.wikipedia.org/wiki/Microsoft_.NET).

IL - объектно-ориентированный машинный язык не зависящий от процессора ILAsm exe – ассемблер ILDasm exe - дизассемблер IL

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

Для выполнения какого-либо метода его IL-код должен быть преобразован в машинные команды. Этим занимается JIT-компилятор (Just-In-Time) среды CLR.

Функции JITCompiler известен вызываемый метод и тип, в котором он определен. JITCompiler ищет в метаданных соответствующей сборки IL-код вызываемого метода. Затем JITCompiler проверяет и компилирует IL-код в машинные команды, которые сохраняются в динамически выделенном блоке памяти. После этого JITCompiler возвращается к структуре внутренних данных типа, созданной средой CLR, и заменяет адрес вызываемого метода адресом блока памяти, содержащего готовые машинные команды. В завершение JITCompiler передает управление коду в этом блоке памяти.

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

система типов, которая содержит полное описание всех возможных типов данных и программных конструкций, которые поддерживаются общеязыковой исполнительной средой CLR. Также здесь описывается то, как эти сущности могут взаимодействовать между собою.

Типами могут быть классы, интерфейсы, структуры, перечисления, делегаты.

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

сравнить два экземпляра на равенство;

получить хеш-код экземпляра;

запросить фактический тип экземпляра;

выполнить поверхностное (поразрядное) копирование экземпляра;

получить строковое представление текущего состояния экземпляра.

**6. 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. Главная сборка размещается в файле “mscorlib.dll”.

**7. Что такое «сборка»? Из чего состоит сборка .NET?**

Сборка (assembly) — 1) это абстрактное понятие, для логической группировки одного или нескольких управляемых модулей или файлов ресурсов.

2) дискретная единица многократно используемого кода внутри CLR

Сборки имеют следующие составляющие:

* Манифест, который содержит метаданные сборки
* Метаданные типов. Используя эти метаданные, сборка определяет местоположение типов в файле приложения, а также места размещения их в памяти
* Собственно код приложения на языке MSIL, в который компилируется код C#
* Ресурсы

**8. Какие виды сборок существуют?**

с нестрогими именами (weakly named assemblies)

со строгими именами (strongly named assemblies). ♣ подписаны при помощи пары ключей, уникально идентифицирующей издателя сборки (безопасность, управление ее версиями, развертывание в любом месте пользовательского жесткого диска или в Интернете) ♣ атрибуты: имя файла (без расширения), номер версии, идентификатор регионального стандарта и открытый ключ.

**9. Что такое assembly manifest?**

набор таблиц метаданных файлы, которые входят в сборку, общедоступные экспортируемые типы, файлы ресурсов или данных

Манифест (manifest) — это еще один набор таблиц метаданных, которые в основном содержат имена файлов, составляющих сборку. Кроме того, эти таблицы описывают версию и региональные стандарты сборки, ее издателя, общедоступные экспортируемые типы, а также все составляющие сборку файлы.

В манифесте размещается информация:

* о текущей версии сборки;
* сведения о культуре (локализация строчных и графических ресурсов);
* перечень ссылок на все внешние сборки, которые нужны для правильного функционирования.

**10.Что такое GAC?**

Global Assembly Cache **-** Место, где располагаются совместно используемые сборки.

**11.Чем managed code отличается от unmanaged code**

IL-код иногда называют управляемым (managed code), потому что CLR управляет его выполнением.

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

Main является точкой входа в программу

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

использование директивы using импортирует все имена из заданного пространства имён в окружающее пространство имён

форма директивы using, позволяющая создать псевдоним для отдельного типа или пространства имен

Директива using используется в следующих трех целях.

* 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;

Ключевое слово using также используется для создания операторов using, которые помогают обеспечить правильную обработку объектов [IDisposable](https://docs.microsoft.com/ru-ru/dotnet/api/system.idisposable), таких как файлы и шрифты.

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

Скорее всего от вас хотят услышать, что перед использованием using SomeNameSpace;, вы должны указать в проекте ссылку на сборку, в которой объявлены типы данного пространства имен. Но это не ни как 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.

Сборка предназначена для сохранения пространств имен (namespaces). Пространства имен содержат типы. Типами могут быть классы, делегаты, интерфейсы, перечисления, структуры.

Сборка может содержать любое количество пространств имен. Любое пространство имен может содержать любое количество типов (классов, интерфейсов, структур, перечислений, делегатов).

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

Примитивный тип - это **несоставной** тип данных, тоесть такие типы как int, double, boolean, single, byte и т.д., занимающие в памяти строго определенное количество байт. String, Object. enum уже не примитивные, а составные типы данных, тоесть состоящие из ряда примитивных.

sbyte 8 ,short 16 , int 32 ,ong 64 , byte 8 , ushort 16 , uint 32 , ulong 64 , char 16 , bool 8 , float 32 , double 64 , decimal 128 , string , object .

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

В переменных ссылочных типов хранятся ссылки на их данные (объекты)

Ссылочные типы:

* Тип object
* Тип string
* Классы (class)
* Интерфейсы (interface)
* Делегаты (delegate)

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

переменные типа значений содержат свои данные непосредственно.

Типы значений:

* Целочисленные типы (byte, sbyte, short, ushort, int, uint, long, ulong)
* Типы с плавающей запятой (float, double)
* Тип decimal
* Тип bool
* Тип char
* Перечисления enum
* Структуры (struct)

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

Ссылочные хранят ссылку на объект, а значимые непосредственно само значение****

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

Упаковкой (boxing) называется процесс преобразования типа значения в тип System.Object или в тип интерфейса, который реализуется данным типом значением. распаковка (unboxing) получение указателя на исходный значимый тип (поля данных), содержащийся в объектею Объекты значимого типа существуют в двух формах: неупакованной (unboxed) и упакованной (boxed). Ссылочные типы бывают только в упакованной форме.

**20.В чем заключается разница между int и System.Int32? double и System.Double и т.д.?**

int - это псевдоним для System.Int32. То же самое.

double - это алиас класса Double, т.е. в коде пишем **double** тоже что new **System.Double**

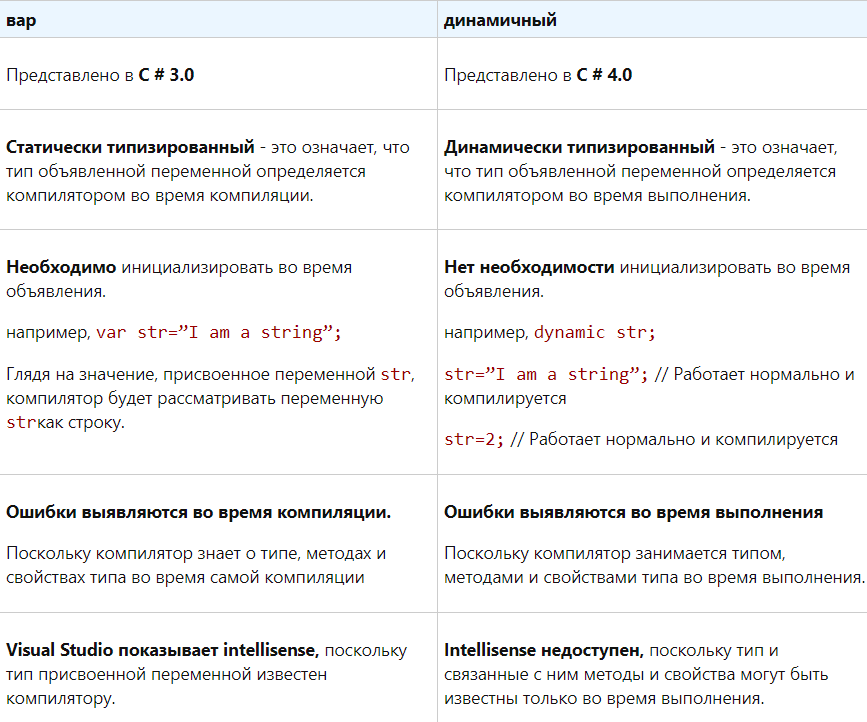
**21.Для чего используется тип dynamic?**

Назначение : Отражения или коммуникации с другими компонентами

Использование: для членов класса - поля, свойства/индексаторы, структур , для метода, делегата, или унарных/бинарных операторов

Что происходит: может получить какое угодно начальное значение, и на протяжении времени его существования это значение может быть заменено новым

Ограничение динамических типов - могут использоваться только для обращения к членам экземпляров ( должна ссылаться на объект) - не могут использовать лямбда-выражения или анонимные методы C# при вызове метода - не могут воспринимать расширяющие методы

**22.В чем заключается главное отличие между var и dynamic? **

****

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

**Неявная** **типизация** — **типизация**, при которой **переменная** остается строго **типизированной**, а её тип определяется компилятором, исходя из присваиваемого значения.

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

применяется только для типов значений

Проверка на равенство объектов (равны не только, когда они имеют ненулевые значения, которые совпадают, но и когда оба объекта равны null**)**

int? x1 = null; int? x2 = null; System.Console.Write(x1 == x2); //True

Преобразование int? x1 = null; if (x1.HasValue) { int x2 = (int)x1;//явное int? x3 = x2; //неявное к T? long? x4 = x2; //неявное расширяющее }

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

String path; path = "C:\\Windows\\regedit.exe"; path = "C:/Windows/regedit.exe";

присваивание (=);

проверка на равенство содержимого (==);

проверка на неравенство (!=);

обращение по индексу ([]);

сцепление (конкатенация) строк (+)

>,< , >=,<= - сравнивают ссылки!!!!!!!!

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

char[] a = { '0', '0', '0' }; // создание массива символов:

string s; // инициализация отложена

string t = "qqq"; // инициализация строковым литералом

string u = new string(' ', 20); // с пом. конструктора

string v = new string(a); // создание из массива символов

**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, начиная с определенного индекса в массив
* **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

**28.В чем отличие пустой и null строки?**

Пустая строка — экземпляр объекта System.String, содержащий 0 символов: string s = ""; Для пустых строк можно вызывать методы.

Строки со значениями null не ссылаются на экземпляр объекта System.String, попытка вызвать метод для строки null вызовет исключение NullReferenceException. строки null можно использовать в операциях объединения и сравнения с другими строками.

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

|  |  |
| --- | --- |
| [String.Compare](https://docs.microsoft.com/ru-ru/dotnet/api/system.string.compare) | Сравнивает значения двух строк. Возвращает целочисленное значение. |
| Отрицательное целое число | Первая строка предшествует второй в порядке сортировки.  -или-  Первая строка имеет значение null. |
| 0 | Первая и вторая строка равны.  -или-  Обе строки имеют значение null. |
| Положительное целое число  -или-  1 | Первая строка следует за второй в порядке сортировки.  -или-  Вторая строка имеет значение null. |

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

Класс String является immutable - Вы не можете модифицировать объект String, но можете заменить его созданием нового экземпляра.

Класс StringBuffer является mutable - использовать StringBuffer или StringBuilder следует тогда, когда вы хотите модифицировать содержимое. StringBuilder был добавлен в пятой джаве и он во всем идентичен классу StringBuffer, за исключением того, что он не синхронизирован, что делает его значительно шустрее. Но цена скорости - небезопасное поведение в мультипоточной среде.

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

Преобразует значение одного базового типа данных к другому базовому типу данных.

double dNumber = 23.15;

int iNumber = System.Convert.ToInt32(dNumber);//23

bool bNumber = System.Convert.ToBoolean(dNumber);//true

string strNumber = System.Convert.ToString(dNumber);//”23.15”

char chrNumber = System.Convert.ToChar(strNumber[0]);//”2”

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

статический класс System.Console

int x = Console.ReadLine();

Console.WriteLine((char)x);

String s = Console.ReadLine();

Console.WriteLine(s);

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

тип[] имя;

тип[] имя = new тип [ размерность ];

тип[] имя = { список инициализаторов };

тип[] имя = new тип [] { список инициализаторов };

тип[] имя = new тип [ размерность ] { список инициализаторов };

Многомерные

тип[,] имя;

тип[,] имя = new тип [ разм\_1, разм\_2 ];

тип[,] имя = { список инициализаторов };

тип[,] имя = new тип [,] { список инициализаторов };

тип[,] имя = new тип [ разм\_1, разм\_2 ] { список инициализаторов };

**34.Что такое ступенчатый массив? Как его задать?**

тип[][] имя;

int[][] a = { new int[3], new int[5], new int[4] };

foreach (int[] x in a) {

foreach (int b in x) Console.Write("\t" + b);

Console.WriteLine(); }

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

Цикл foreach предназначен для перебора элементов в контейнерах, в том числе в массивах

foreach (тип\_иден. название\_иден. in контейнер) { операторы }

цикл работает только на чтение, но не на запись элементов (наполнять нельзя)

String[] collection = new String[] { "1-й элемент", "2-й элемент", "3-й элемент" };

//Последовательно выводим в консоль элементы массива

foreach (String element in collection) { Console.WriteLine(element); }

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

Кортежи (tuple) комбинируют объекты различных типов (от одного до восьми). Типы и выражения

Свойства: создается один раз и остается неименным (все свойства доступны только для чтения) позволяют использовать методы CompareTo, Equals, GetHashCode и ToString, свойство Size реализуют интерфейсы IStructuralEquatable, IStructuralComparable и IComparable (можно сравнивать)

**37.Что такое локальная функция? Какова область ее видимости?**

Локальные функции - вспомогательная функция - внутри метода, в котором вызывается. Аргументы внешнего метода и его локальные переменные доступны для локальной функции

**38. В чем разница между кодом, заключенным в блок checked и кодом, заключенным в блок unchecked?**

Строка кода C# может выполняться как в проверенном, так и в непроверенном контексте. В проверенном контексте арифметическое переполнение создает исключение. В непроверяемом контексте арифметическое переполнение игнорируется и результат усекается путем удаления старших разрядов, которые не помещаются в целевой тип данных.

* [checked](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/checked). Укажите проверенный контекст.
* [unchecked](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/unchecked). Укажите непроверенный контекст

**39.Какой контекст (checked/unchecked) применяется по умолчанию? Как можно переопределить это поведение?**

**40.Для чего используется ключевое слово fixed? Каковы особенности его использования?**

Fixed - Ключевое слово позволяет «закрепить» локальную систему в стеке, чтобы предотвратить их сбор или перемещение во время сборки мусора. Он используется для сценариев низкого уровня программирования.

Указатель из выражения фиксируется с помощью fixed ключевого слова и привязывается к идентификатору с помощью use ключевого слова. Семантика этого объекта аналогична управлению ресурсами с помощью use ключевого слова. Указатель фиксирован, пока находится в области видимости, и, когда он выходит за пределы области, он больше не является фиксированным. fixed не может использоваться вне контекста use привязки. Необходимо привязать указатель к имени с помощью use .

Использование метода fixed должно выполняться в выражении в функции или в методе. Его нельзя использовать в области действия уровня скрипта или модуля.