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

.Net Framework это платформа для разработки использующая среду CLR.

CLR Common Language Runtume.

.Net Framework состоит из CLR и библиотеки классов .NET Framework.

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

CLR (Common language Runtime) – среда выполнения которая подходит для разных языков программирования. Возможности CLR: управление памятью, загрузка сборок, безопасность,

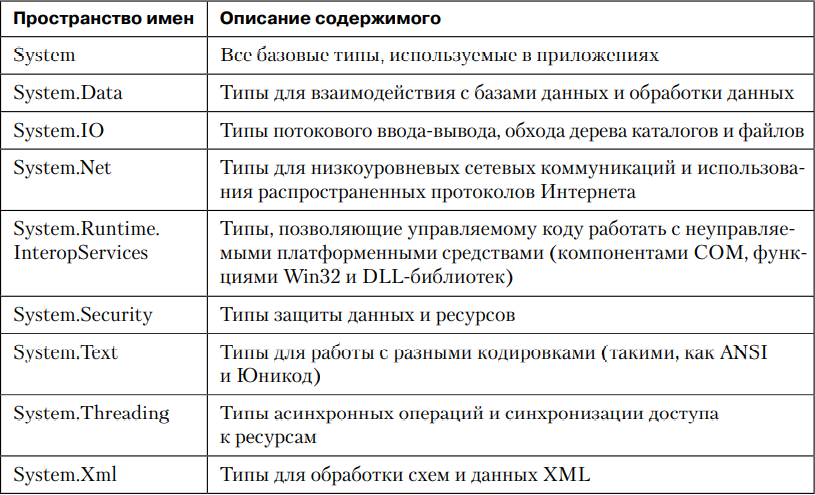
Обработка исключений, синхронизация.

(BCL – Baseclasslibrary стандартная библиотека классов платформы .NETFRAMEWORK короче говоря это и есть FCL)FCL (Framework Class Library) – Один из компонентов .Net FrameWork.

FCL – представляет набор сборок в формате DLL, которые содержат много тысяч определений типов, где каждый из типов предоставляет некоторую функциональность.

Грубо говоря сборка DLL это подключаемая библиотека описывающая некоторые типы.

Вот некоторые пространства имен которые включены в FCL:



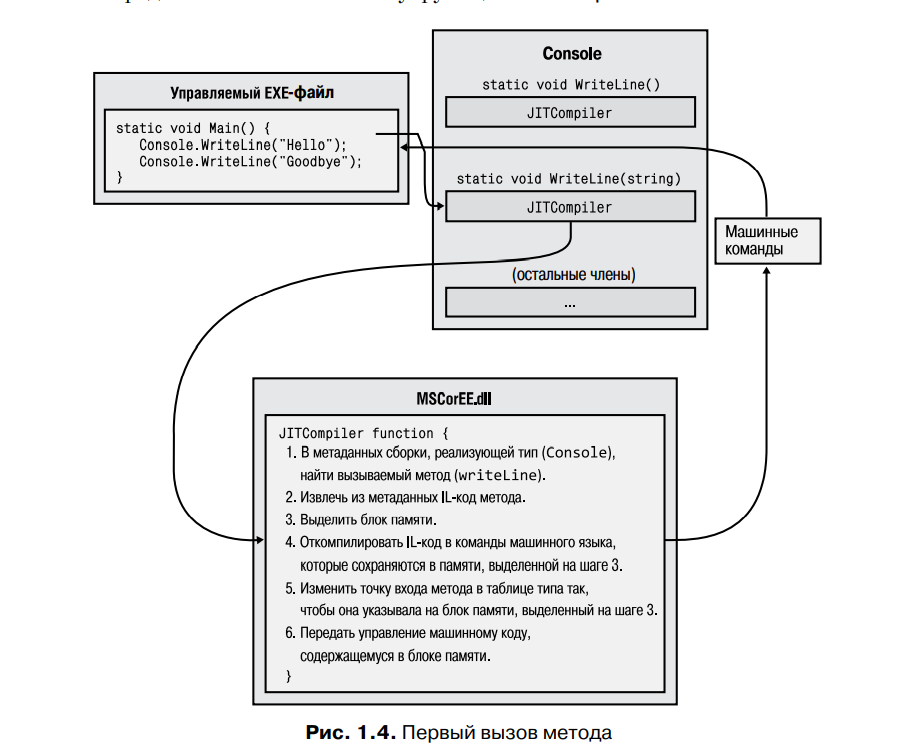
CLI – Common Language Infrastructure. Это кроссплатформенная цепочка инструментов для разработки, сборки, запуска и публикаций приложений .NET. Интерфейс CLI (интерфейс командной строки) входит в пакет .NET.

СТАНДАРТНЫЕ УСТАНОВЛЕННЫЕ КОМАНДЫ: new, restore, build и тд.

IL – Intermediate Language. CLR компиляторы генерируют IL код , этот код иногда называют управляемым так как CLR управляет его выполннением. По синтаксису IL напоминает ассемблер , поэтому его часто называют высокоуровневым ассемблером. Промежуточный язык, разработанный фирмой [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) для платформы [.NET Framework](https://ru.wikipedia.org/wiki/Microsoft_.NET). [JIT](https://ru.wikipedia.org/wiki/JIT)-компилятор CIL является частью [CLR](https://ru.wikipedia.org/wiki/Common_Language_Runtime).

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

Jit компилятор занимается преобразованием IL кода в машинные команды. JIT компилятор среды CLR.



Когда метод Main первый раз обращается к методу WriteLine, вызывается функция JITCompiler. Она отвечает за компиляцию IL-кода вызываемого метода в собственные команды процессора. Поскольку IL-код компилируется непосредственно перед выполнением («just in time»), этот компонент CLR часто называют JIT-компилятором.

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 комады, 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.

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

CTS это спецификация описывающая способ определения и поведения типов.

Согласно этой спецификации тип может содержать нуль и более членов.

Поле – переменная, являющаяся частью состояния объекта. Поле идентифицируется именем и типом.

Метод – функция выполняющая операцию с объектом, часто с изменением его состояния. Метод обладает именем, сигнатурой и модификаторами. Сигнатура определяет количество параметров(и порядок их следования), типы параметров, наличие возвращаемого значения, и если оно имеется тип значения, возвращаемого методом.

Свойство — с точки зрения вызывающей стороны выглядит как поле, но в реализации типа представляет собой метод (или два). Свойства позволяют организовать проверку параметров или состояния объекта перед обращением к значению и/или вычислять его значение только при необходимости. Кроме того, они упрощают синтаксис работы с данными и позволяют создавать «поля», доступные только для чтения или записи.

Событие — используется для создания механизма оповещения между объектом и другими заинтересованными объектами. Например, кнопка может поддерживать событие, оповещающее другие объекты о щелчке на ней.

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

А вот еще одно правило CTS: все типы должны быть производными (прямо или опосредованно) от предопределенного типа System.Object (то есть от типа Object из пространства имен System). Тип Object является корнем иерархии типов, а следовательно, гарантирует, что каждый экземпляр типа обладает минимальным набором аспектов поведения. А если говорить конкретнее, тип System.Object позволяет сделать следующее:

1. **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.

Далее основной поток вызывает определенный в библиотеке MSCorEE.dll метод, который инициализирует CLR, загружает сборку EXE, а затем вызывает ее метод Main, в котором содержится точка входа. На этом процедура запуска управляемого приложения считается завершенной.

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

На самом деле среда CLR работает не с модулями, а со сборками. Сборка (assembly) — это абстрактное понятие, понять смысл которого на первых порах бывает нелегко. Во-первых, сборка обеспечивает логическую группировку одного или нескольких управляемых модулей или файлов ресурсов. Во-вторых, это наименьшая единица многократного использования, безопасности и управления версиями. Сборка может состоять из одного или нескольких файлов — все зависит от выбранных средств и компиляторов. В контексте среды CLR сборкой называется то, что мы обычно называем компонентом.

Сборка позволяет разделить логическое и физическое представления компонента, поддерживающего многократное использование, безопасность и управление версиями. Разбиение программного кода и ресурсов на разные файлы полностью определяется желаниями разработчика

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

Все эти компоненты могут находиться в одном файле, и тогда сборка представляет один единственный файл в формате exe или dll.

1. **Какие виды сборок существуют?**

Существует несколько способов создания сборок. Можно использовать средства разработки, такие как Visual Studio, позволяющие создавать файлы DLL или EXE. Чтобы создать сборки с использованием модулей из других средств разработки, можно воспользоваться средствами из Windows SDK. Для создания динамических сборок также можно использовать интерфейсы CLR такие, как [System.Reflection.Emit](https://docs.microsoft.com/ru-ru/dotnet/api/system.reflection.emit).

1. **Что такое assembly manifest?**

Любая сборка, статическая или динамическая, содержит коллекцию данных с описанием того, как ее элементы связаны друг с другом. Эти метаданные содержатся в манифесте сборки. Манифест сборки содержит все метаданные, необходимые для задания требований сборки к версиям и удостоверения безопасности, а также все метаданные, необходимые для определения области действия сборки и разрешения ссылок на ресурсы и классы. Манифест сборки может храниться в PE-файле (EXE или DLL) с кодом MSIL или же в отдельном PE-файле, содержащем только данные манифеста.

Для сборки с одним связанным файлом манифест включается в PE-файл, чтобы получить однофайловую сборку. Создать многофайловую сборку можно, включив в нее отдельный файл манифеста или же добавив манифест в один из PE-файлов сборки.

Манифест сборки предназначен для следующих задач:

перечисление файлов, составляющих сборку;

сопоставление ссылок на типы и ресурсы сборки с файлами, содержащими объявления и реализации этих типов и ресурсов;

перечисление других сборок, от которых зависит эта сборка;

обеспечение косвенного обращения пользователей сборки к подробностям ее реализации;

предоставление собственного описания сборки;

**10.Что такое GAC?**

global assembly cache, GAC) - глобальным кэшем сборок.

Место,

где располагаются совместно используемые сборки, называют глобальным кэшем

сборок (global assembly cache, GAC). Точное местонахождение GAC – подробность

реализации, которая может изменяться в будущих версиях .NET Framework. Тем

не менее обычно GAC находится в каталоге

%SystemRoot%\Microsoft.NET\Assembly

GAC имеет иерархическое строение и содержит множество вложенных каталогов,

имена которых генерируются по определенному алгоритму. Ни в коем случае не

следует копировать файлы сборок в GAC вручную — вместо этого надо использовать

инструменты, созданные специально для этой цели. Эти инструменты «знают» внутреннюю структуру GAC и умеют генерировать надлежащие имена каталогов.

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

Управляемый код (managed code) – это код управляемый общеязыковой средой CLR. Это означает, что что управляющая среда имеет возможность приостановить выполнение приложения и получить специфическую информацию о состоянии приложения в любой момент его исполнения. Исходный код используемого языка программирования транслируется в управляемый код CIL (его еще называют высокоуровневым ассемблером).

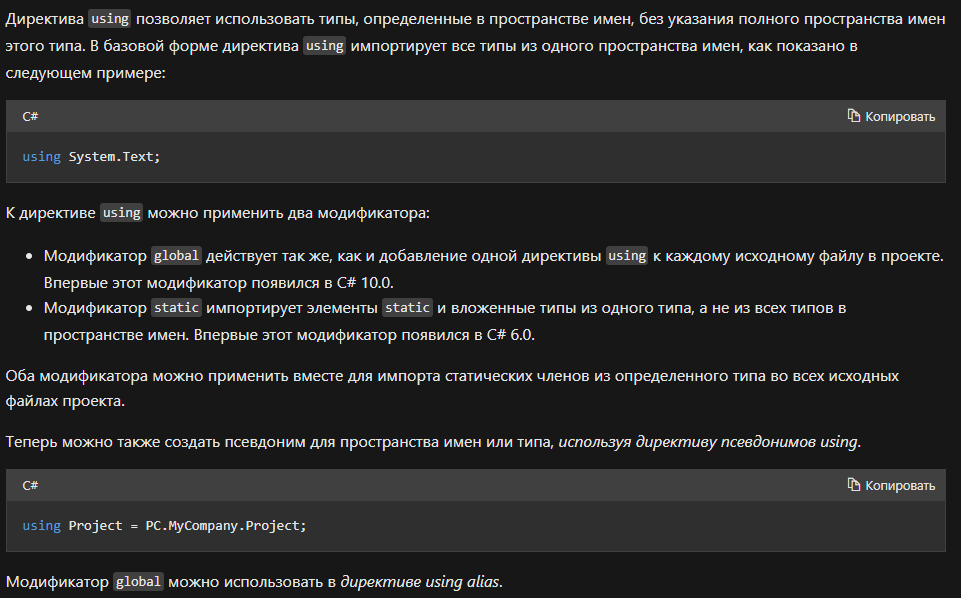
После создания библиотеки классов или приложения, исходный код продолжает храниться на языке CIL. А при запуске приложения или обращении к библиотеке выполняется Just-In-Time (JIT) компиляция приложения в машинный код, специфичный именно для того окружения, на котором осуществляется исполнение приложения. При этом важной особенностью является то, что выполняется компиляция только той части приложения или библиотеки, к которой выполняется обращение (ну и связей естественно). Это позволяет увеличить производительность работы системы и экономить ресурсы.

Неуправляемый код (unmanaged code) – это код, транслируемый сразу в машинный исполняемый код и исполняется операционной системой напрямую.

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

языке C# метод Main() — это главный метод программы, ее входная точка, является частью класса. Класс, в котором находится этот метод, по умолчанию называется Program.

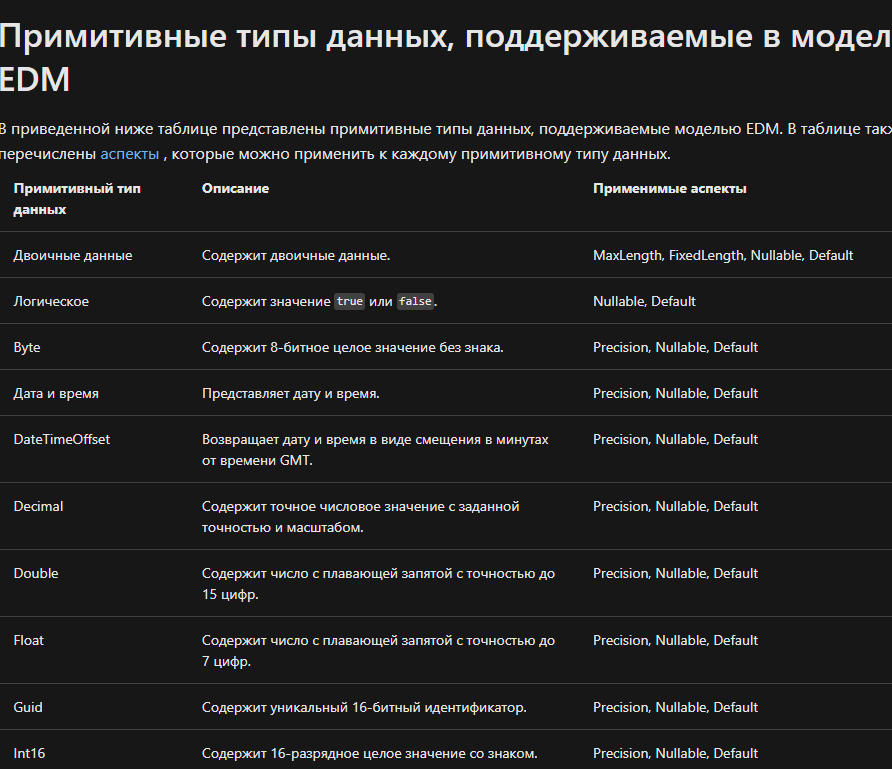
**13.Варианты использования директивы using( using Directive ) в C#.**



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

Пространство имен — это совокупность классов, логически связанных между собой. Между сборками и пространствами имен нет прямого соответствия: в сборке может хранится несколько пространств имен, а разные классы одного пространства имен могут быть определены в разных сборках.

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



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

Переменная ссылочного типа содержит не данные, а ссылку на них. Сами данные в этом случае уже хранятся в куче. Куча - это область памяти, в которой размещаются управляемые объекты, и работает сборщик мусора. Сборщик мусора освобождает все ресурсы и объекты, которые уже не нужны.

К ссылочным типам относятся:

- Классы (class);

- Интерфейсы (interface);

- Делегаты (delegate);

- Тип object;

- Тип string.

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

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 типы содержат ссылку на данные, а типы значения содержат сами данные.

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

Упаковка и распаковка значимых типов в C# — доступный для программиста механизм преобразования размерных типов данных языка C# из значимых в ссылочные и обратно через задействование свойств фундаментального базового класса Object.

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

int - примитив, доступный в компиляторе C#, в то время как Int32 является типом FCL и следовательно доступен для всех языков, соответствующих CLS.

Грубо говоря int это псевдоним для System.Int 32

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

Хотя C# относится к статически типизированным языкам, в последних версиях языка были добавлены некоторые динамические возможности. Так, начиная с .NET 4.0 была добавлена новая функциональность под названием DLR (Dynamic Language Runtime). DLR представляет среду выполнения динамических языков, например, таких языков как IronPython и IronRuby.

Чтобы понять значение данного нововведение, нужно осознавать разичие между языками со статической и динамической типизицией. В языках со статической типизацией выявление всех типов и их членов - свойств и методов происходит на этапе компиляции. А в динамических языках системе ничего не известно о свойствах и методах типов вплоть до выполнения.

Благодаря этой среде DLR C# может создавать динамические объекты, члены которых выявляются на этапе выполнения программы, и использовать их вместе с традиционными объектами со статической типизацией.

Ключевым моментом использования DLR в C# является применение типов dynamic. Это ключевое слово позволяет опустить проверку типов во время компиляции. Кроме того, объекты, объявленные как dynamic, могут в течение работы программы менять свой тип.

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

var является просто сокращением для объявления нормального типа, где вы позволяете компилятору угадать правильный тип.

dynamic - новый (статический) тип, где все проверки выполняются во время выполнения, а не компилятором.

Var и динамический тип определения.

var во время компиляции, а динамические - во время выполнения.

в объявлении var и инициализации оба являются обязательными, как постоянная переменная, а

в динамической инициализации может быть во время выполнения, например переменные readonly.

в var type любой тип определяется в то время, когда инициализация не может изменить следующую, но

dynamic может использовать любой тип, даже пользовательский тип данных.

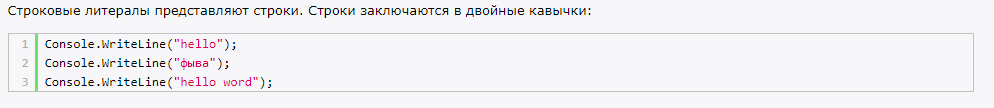
**23.Что такое неявно типизированная переменная?**

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.

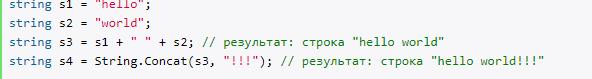
**24.Для чего используют Nullable тип?**

Nullable-типы могут пригодиться при работе с базой данных. Некоторые поля таблицы базы данных могут принимать значение null. Например, значение булевого поля любит ли человек кошек может оставаться неизвестным. Или более практический пример. На главной странице этого сайта есть лента с уроками, и в ней несколько последних, остальные на страницах 2,3,…Номер страницы передается аргументом в http запросе ("…?page=2"), но для первого захода на сайт страницу 1 указывать в запросе не очень красиво, используя nullable-тип, переменная int, которая отвечает за номер страницы, в таком случае принимает значение null.

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



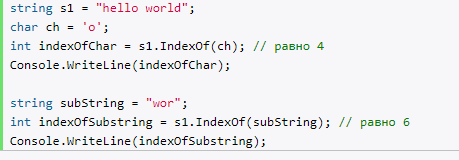
Со строками можно производить конкатенацию



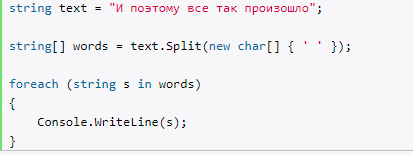
Сравнение строк

(Я думаю тут итак понятно как делать)

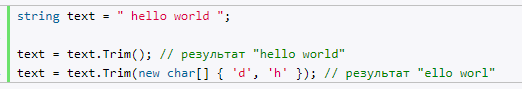
Поиск по строке



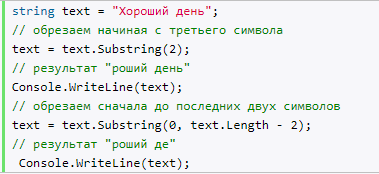
Разделение строк



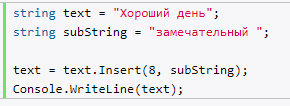
Обрезка строки



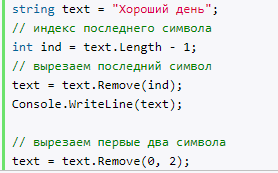
2 сп-сб



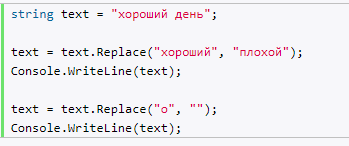
Вставка



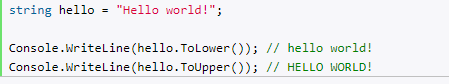
Удаление строк



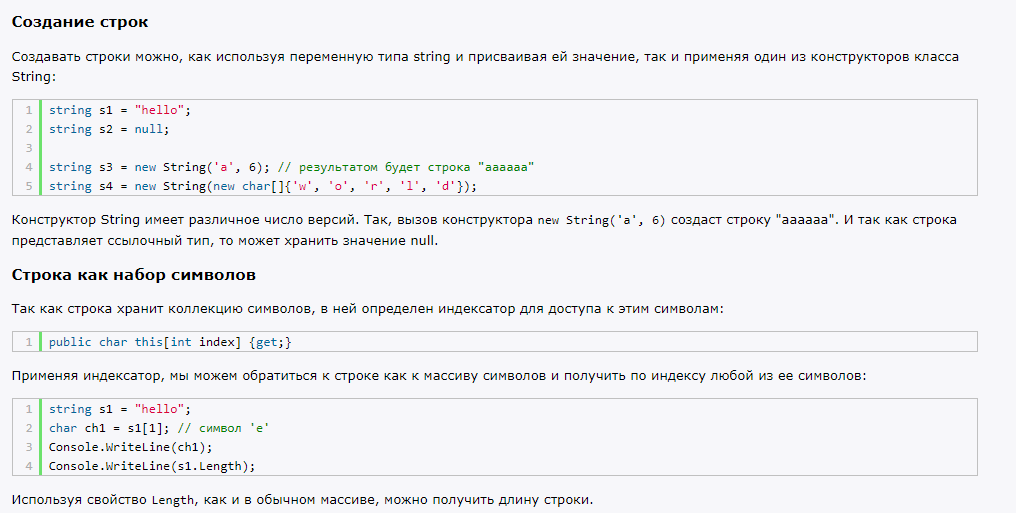
Замена



Смена регистра

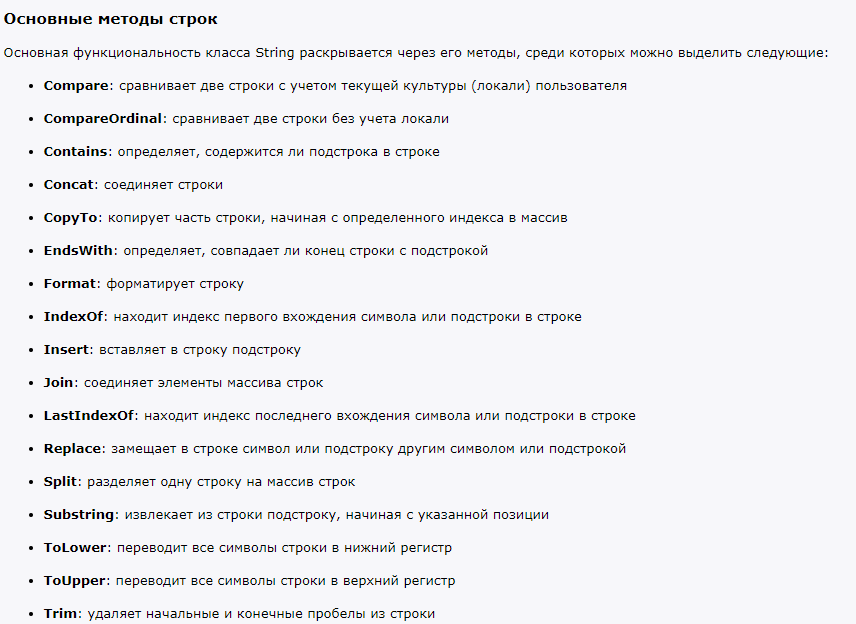


**26.Какие есть способы для задания и инициализации строк?**



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

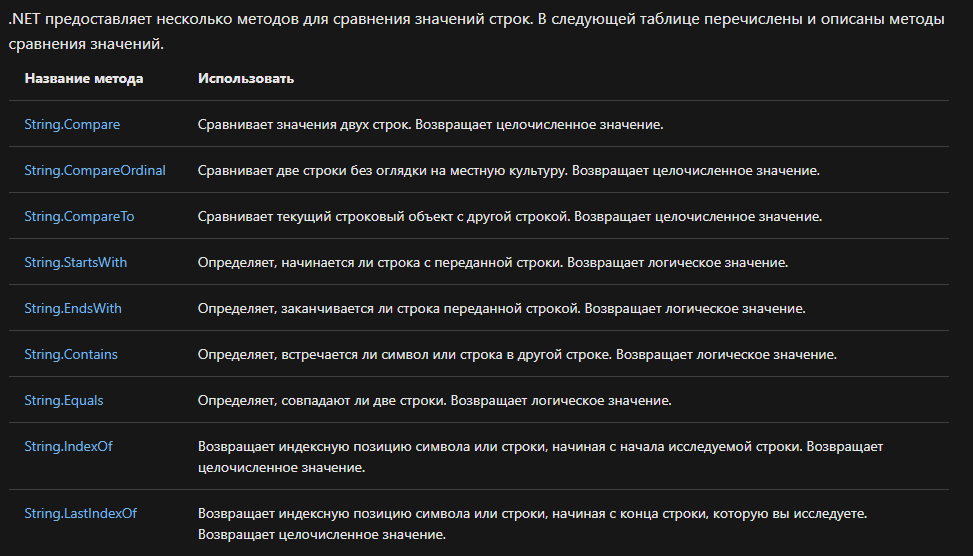
БЛЯ КАК МЕНЯ ЗАЕБАЛИ ОДНОТИПНЫЕ ВОПРОСЫ ГДЕ ОТВЕТЫ ОДНИ И ТЕЖЕ



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

NULL означает отсутствие значения (т. е. нет значения ), а пустая строка означает, что есть строковое значение нулевой длины.

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



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

Экземпляр string является неизменяемым. Вы не можете изменить его после того, как он был создан. Любая операция, которая, как представляется, изменяет строку, вместо этого возвращает новый экземпляр

Когда вам нужна изменяемая строка, например, та, которую вы строите по частям или где вы меняете много вещей, вам понадобится StringBuilder , который является буфером символов, которые можно изменить. Это, по большей части, влияет на производительность. Если вам нужна изменяемая строка, а вместо этого вы делаете это с обычным экземпляром string , то в конечном итоге вы создадите и уничтожите множество объектов без необходимости, в то время как сам экземпляр StringBuilder изменится, отрицая необходимость во многих новых объектах.

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

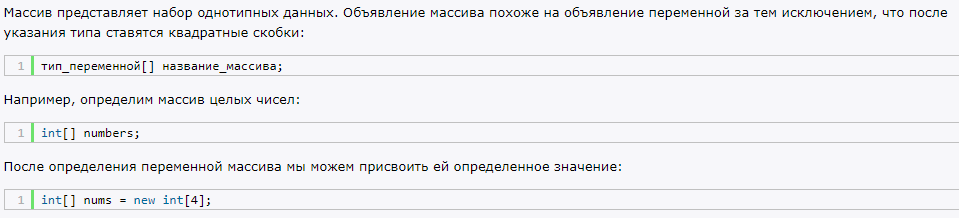
Преобразования с использованием вспомогательных классов. Чтобы выполнить преобразование между несовместимыми типами, например целыми числами и объектами System.DateTime или шестнадцатеричными строками и массивами байтов, можно использовать классы System.BitConverter и System.Convert, а также методы Parse встроенных числовых типов, такие как Int32.Parse. Дополнительные сведения см. в руководствах по преобразованию массива байтов в значение типа int, преобразованию строки в число и преобразованию из шестнадцатеричных строк в числовые типы.

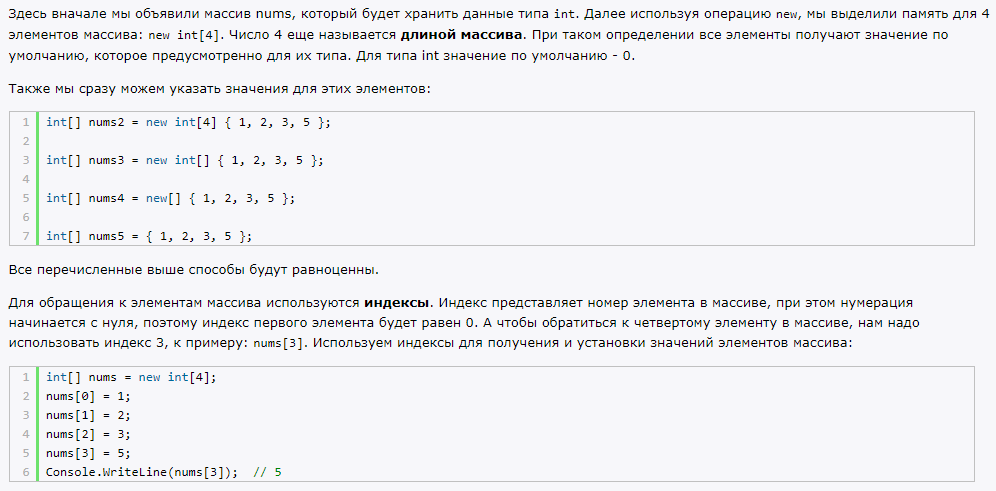
**32.Как выполнить консольный ввод/вывод?**

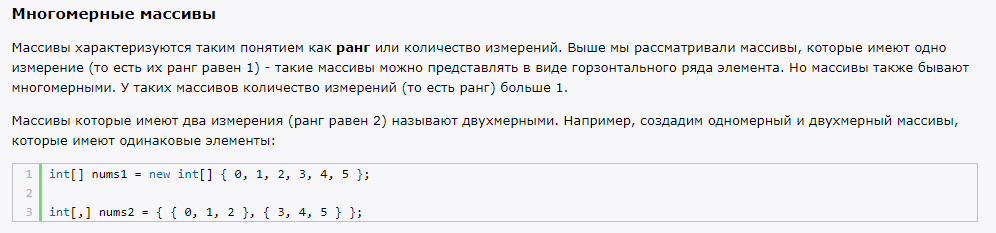
System.Console.WriteLine(a);

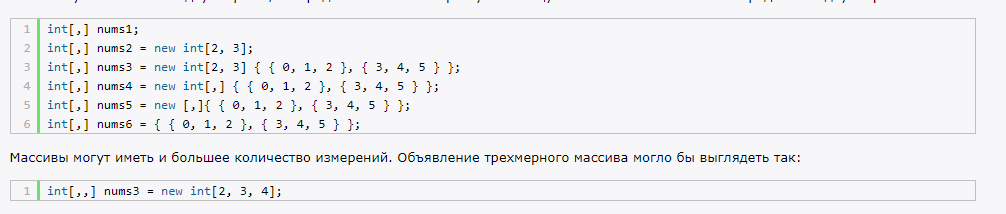
System.Console.ReadLine();

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

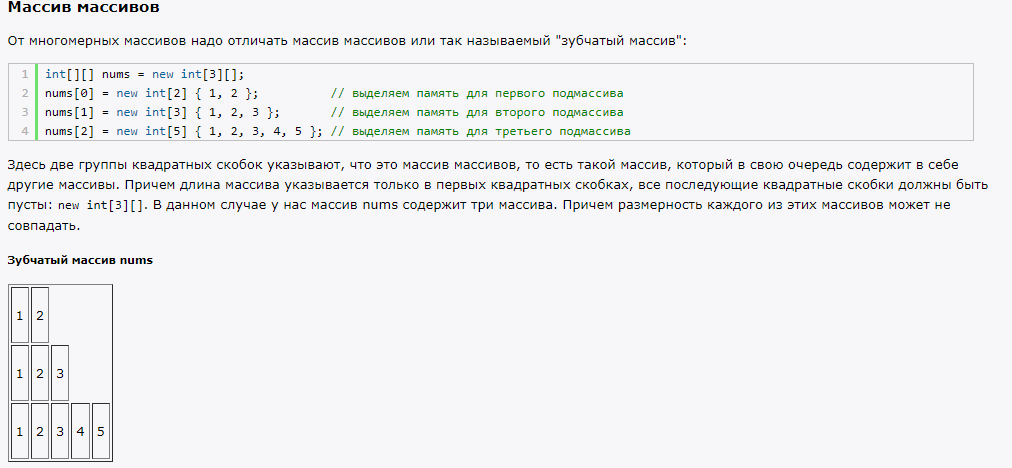




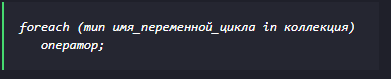




**34.Что такое ступенчатый массив? Как его задать?**



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



Непонятный вопрос если честно, если вопрос какие данные можно перечислять то массивы  
а использовать в перечислениях Int, double,float и т.д.

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

Кортежи предоставляют удобный способ для работы с набором значений, который был добавлен в версии C# 7.0.

Кортеж представляет набор значений, заключенных в круглые скобки:

var tuple = (5, 10);

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

Локальная функция это функция у которой область видимости находиться только внутри финурных скобок самой функции. Грубо говоря все переменные и массивы находящиеся внутри скобок локальной функции будут видны компилятору только после запуска\обращения к самой функции.

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

В C# имеется специальное средство, связанное с генерированием исключений, возникающих при переполнении в арифметических вычислениях. Как известно, результаты некоторых видов арифметических вычислений могут превышать диапазон представления чисел для типа данных, используемого в вычислении. В этом случае происходит так называемое переполнение.

В C# допускается указывать, будет ли в коде сгенерировано исключение при переполнении, с помощью ключевых слов checked и unchecked. Так, если требуется указать, что выражение будет проверяться на переполнение, следует использовать ключевое слово checked, а если требуется проигнорировать переполнение — ключевое слово unchecked.

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

checked - используется для явного включения проверки переполнения при выполнении арифметических операций и преобразований с данными целого типа. По умолчанию выражение, содержащее только константные значения, вызывает ошибку компилятора в том случае, если результат его вычисления выходит за допустимые пределы значений конечного типа. Если выражение содержит одно или несколько неконстантных значений, компилятор не выполняет проверку переполнения.

· unchecked - используется для подавления проверки переполнения при выполнении арифметических операций и преобразований с данными целого типа. Если в непроверяемом контексте результатом выполнения выражения является значение, выходящее за допустимые пределы значений конечного типа, то переполнение не помечается. Вычисление в приведенном ниже примере выполняется в блоке или выражении unchecked, поэтому факт превышения результатом максимального значения целого числа игнорируется.

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

В C# оператор fixed можно использовать для создания буфера с массивом фиксированного размера в структуре данных. Буферы фиксированного размера полезны при написании методов, взаимодействуя с источниками данных из других языков или платформ. Фиксированный массив может принимать любые атрибуты или модификаторы, разрешенные для обычных членов структуры. Единственное ограничение заключается в том, что тип массива должен быть bool byte char short int long sbyte ushort uint ulong float double

**ДОПОЛНИТЕЛЬНО  
это ссылки на статьи которые лучше прочесть**

[**https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-tuples**](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-tuples)

[**https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/nullable-value-types**](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/nullable-value-types)

[**https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/types/casting-and-type-conversions**](https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/types/casting-and-type-conversions)

[**https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/functional/discards**](https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/functional/discards)