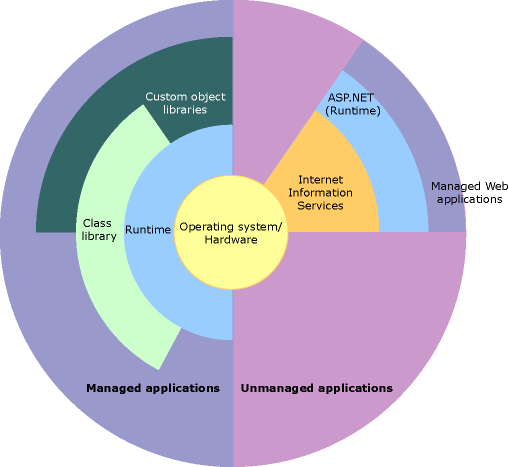
**1. Что такое .Net Framework и из чего он состоит?**

Платформа .NET Framework — это технология, которая поддерживает создание и выполнение веб-служб и приложений Windows. .NET Framework состоит из общеязыковой среды выполнения (CLR) и библиотеки классов .NET Framework.



<https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview>

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

Think of the runtime (**CLR**) as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness. <https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview>

**FCL** (Framework Class Library, библиотека классов фреймворка) и **BCL** (Base Class Library, базовая библиотека классов) — это набор библиотек, предоставляющих основные функции для разработки приложений на .NET. FCL включает в себя BCL, которая предоставляет основные типы данных, коллекции, обработку исключений, ввод-вывод и многое другое. Эти библиотеки создают основу для создания сложных приложений, облегчая доступ к основным функциональностям .NET​ <https://learn.microsoft.com/en-us/dotnet/standard/class-library-overview>

**CLI** (Command-Line Interface, интерфейс командной строки) — это кроссплатформенная цепочка инструментов для разработки, сборки, выполнения и публикации .NET приложений, используемая через команды в терминале. [<https://learn.microsoft.com/en-us/dotnet/core/tools/>]

**IL** (Intermediate Language, промежуточный язык), также известный как CIL (Common Intermediate Language), — это низкоуровневый язык программирования, на который компилируются .NET программы перед их выполнением CLR. IL представляет собой платформонезависимый код, который может быть трансформирован в машинный код конкретной платформы при помощи JIT-компиляции (Just-In-Time). Это позволяет программам .NET работать на различных операционных системах и устройствах​ <https://learn.microsoft.com/en-us/dotnet/standard/glossary>

**3. Пояснить работу JIT-компилятора**

JIT компилятор преобразует IL в машинный код, понятный процессору. Just-in-time (JIT) компиляция — это метод, при котором код переводится в машинный язык во время выполнения программы, а не заранее. Это позволяет оптимизировать производительность под конкретное окружение и облегчает разработку, не требуя создания различных версий кода для разных аппаратных платформ.

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

Система общих типов определяет, как типы объявляются, используются и управляются в среде CLR, а также является важной частью поддержки среды выполнения для межъязыковой интеграции.

<https://learn.microsoft.com/en-us/dotnet/standard/base-types/common-type-system>

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

Тип System.Object в .NET определяет несколько аспектов поведения, которые наследуются всеми другими типами в .NET, так как System.Object является базовым классом для всех типов. Основные аспекты поведения, определяемые этим типом, включают:

1. Методы для работы с экземплярами: Equals(Object), GetHashCode(), ToString()
2. Методы для управления памятью: MemberwiseClone(), Finalize()
3. Типовая информация: GetType()

**6. Что находится в mscorlib dll?**

mscorlib.dll (основная библиотека Microsoft) — это базовая сборка .NET Framework, содержащая базовые и важные классы, используемые всеми приложениями .NET. Он включает в себя реализацию основных классов, таких как System.Object, System.String и System.Collections и других.

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

Сборка — это набор типов и ресурсов, созданных для совместной работы и образующих логическую функциональную единицу. Сборки принимают форму исполняемых файлов (.exe) или файлов динамической библиотеки (.dll) и являются строительными блоками приложений .NET. Они предоставляют общеязыковой среде выполнения информацию, необходимую для понимания реализаций типов.

В .NET и .NET Framework вы можете построить сборку из одного или нескольких файлов исходного кода. В .NET Framework сборки могут содержать один или несколько модулей.

<https://learn.microsoft.com/en-us/dotnet/standard/assembly/>

**8. Какие виды сборок существуют?**

Сборка со строгим именем (strong named assembly) создается с использованием закрытого ключа, соответствующего открытому ключу, распространяемому вместе со сборкой, и самой сборки. Сборка включает манифест сборки, который содержит имена и хэши всех файлов, составляющих сборку.

Weakly named assemblies не имеют уникального идентификатора

<https://learn.microsoft.com/en-us/dotnet/standard/assembly/strong-named>

1. **Что такое assembly manifest?**

Каждая сборка имеет файл манифеста сборки, который позволяет сборке иметь само описание. Подобно оглавлению, манифест сборки содержит:

* Идентификатор сборки (ее имя и версия).
* Таблица файлов, описывающая все остальные файлы, составляющие сборку
* Список ссылок на сборки, который представляет собой список всех внешних зависимостей, таких как DLL-файлы или другие файлы.
* Региональные стандарты

<https://learn.microsoft.com/en-us/dotnet/standard/assembly/manifest>

[https://learn.microsoft.com/en-us/dotnet/standard/assembly/#assembly-manifest](https://learn.microsoft.com/en-us/dotnet/standard/assembly/%23assembly-manifest)

**10. Что такое GAC?**

Место, где располагаются совместно используемые сборки, называют глобальным кэшем сборок (global assembly cache, GAC). В глобальном кэше сборок хранятся сборки, специально предназначенные для совместного использования несколькими приложениями на компьютере.

<https://learn.microsoft.com/en-us/dotnet/framework/app-domains/gac>

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

Управляемый код — это код, выполнением которого управляет среда выполнения. В этом случае рассматриваемая среда выполнения называется Common Language Runtime или CLR, независимо от реализации (например, Mono, .NET Framework или .NET Core).

В неуправляемом мире программист отвечает практически за все. Реальная программа, по сути, представляет собой двоичный файл, который операционная система (ОС) загружает в память и запускает. Все остальное, от управления памятью до вопросов безопасности, — бремя программиста.

<https://learn.microsoft.com/en-us/dotnet/standard/managed-code>

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

Метод Main — это точка входа приложения C#. Когда приложение запускается, первым вызывается именно метод Main. В программе на C# может существовать только одна точка входа.

Main должен быть объявлен внутри класса или структуры. Включающий класс может быть static. Метод Main должен быть static, может иметь любые модификаторы доступа

<https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/program-structure/main-command-line>

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

Директива using позволяет использовать типы, определенные в пространстве имен, без указания полного пространства имен этого типа. В своей базовой форме директива using импортирует все типы из одного пространства имен, как показано в следующем примере:

using System.Text;

Можно применить два модификатора к директиве using: модификатор global имеет тот же эффект, что и добавление той же директивы using к каждому исходному файлу в вашем проекте.

Модификатор static импортирует статические члены и вложенные типы из одного типа, а не импортирует все типы в пространстве имен.

Вы также можете создать псевдоним для пространства имен или типа с помощью директивы using alias:

using Project = PC.MyCompany.Project;

<https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/using-directive>

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

Сборки предоставляют физическую упаковку для типов, определенных в пространстве имен. Они содержат скомпилированный код и ресурсы, которые могут быть развернуты и повторно использованы. Одна сборка может содержать несколько пространств имен. Пространство имен может быть распределено по нескольким сборкам. Манифест сборки содержит информацию обо всех типах и пространствах имен, определенных в сборке, а также их зависимости и другие метаданные.

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

Примитивные типы - это несоставной тип данных, то есть такие типы как int, double, boolean, single, byte и т.д., занимающие в памяти строго определенное количество байт.

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

В C# существует два типа типов: ссылочные типы и типы значений. Переменные ссылочных типов хранят ссылки на свои данные (объекты), а переменные типов значений непосредственно содержат их данные. При использовании ссылочных типов две переменные могут ссылаться на один и тот же объект; следовательно, операции с одной переменной могут повлиять на объект, на который ссылается другая переменная.

Следующие ключевые слова используются для объявления ссылочных типов: class, interface, delegate, record

C# также предоставляет следующие встроенные ссылочные типы: dymanic, object, string

<https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/reference-types>

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

К значимым типам относятся целочисленные числовые типы (int, short и др.), числовые типы с плавающей запятой (float, double, decimal), bool, char, tuple

<https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/value-types>

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

При использовании ссылочных типов две переменные могут ссылаться на один и тот же объект; следовательно, операции с одной переменной могут повлиять на объект, на который ссылается другая переменная. При использовании типов значений каждая переменная имеет свою собственную копию данных, и операции с одной переменной не могут влиять на другую (за исключением переменных параметров in, ref и out).

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

Упаковка — это процесс преобразования типа значения в тип object или в любой тип интерфейса, реализованный этим типом значения. Когда среда CLR упаковывает тип значения, она помещает значение в экземпляр System.Object и сохраняет его в управляемой куче. Распаковка извлекает тип значения из объекта. Упаковка неявная; распаковка явная. Концепция упаковки и распаковки лежит в основе унифицированного представления C# о системе типов, в которой значение любого типа можно рассматривать как объект.

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

каждое ключевое слово типа C# (кроме dymanic) является псевдонимом соответствующего типа .NET. Они взаимозаменяемы. Например, следующие объявления объявляют переменные одного и того же типа:

int a = 123;

System.Int32 b = 123;

<https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/built-in-types>

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

Тип dynamic указывает, что использование переменной и ссылок на ее члены обходит проверку типа во время компиляции. Вместо этого эти операции выполняются во время выполнения. В большинстве случаев он работает как тип object. Переменные типа dymanic компилируются в переменные типа object. Таким образом, динамический тип существует только во время компиляции, а не во время выполнения.

[https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/reference-types#the-dynamic-type](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/reference-types%23the-dynamic-type)

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

Основное отличие между var и dynamic заключается в том, когда определяется тип переменной и когда выполняется проверка типов: var используется для статически типизированного кода с проверкой типов на этапе компиляции, а dynamic - для сценариев, требующих динамической типизации и проверки типов на этапе выполнения.

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

Переменные, объявленные с использованием ключевого слова var, без явного типа. Ключевое слово var указывает компилятору определить тип переменной из выражения в правой части оператора инициализации.

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

Nullable типы в C# используются для представления значений типов, которые могут быть неопределенными (null). Они применяются в ситуациях, когда необходимо указать, что переменная может иметь отсутствие значения. Это особенно полезно для значимых типов (например, int, double), которые по умолчанию не могут быть null.

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

Строковые литералы и заканчиваются одинарной двойной кавычкой (") в одной строке.

Операции над строками: объединение строк, сравнение строк, поиск в строке, разделение строк, обрезка строки, вставка, удаление строк, замена, смена регистра, сцепление, поиск подстроки, разбиение строки на слова

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

Используя ключевого слова string, класс String, вместе со строковым литералом или массивом символов для инициализации

27. Какие методы есть у типа String?

* **Compare**: сравнивает две строки с учетом текущей культуры (локали) пользователя
* **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
* **IndexOf**: находит индекс первого вхождения символа или подстроки в строке
* **LastIndexOf**: находит индекс последнего вхождения символа или подстроки в строке
* **IndexOfAny:** находит индекс первого вхождения одного из символов заданного массива
* **Insert**: вставляет в строку подстроку
* **Join**: соединяет элементы массива строк
* **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
* И другие

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

Пустая строка ("") — это строка с длиной 0, тогда как null строка не указывает ни на какой объект в памяти

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

Используя оператор « == », методы Equals, Compare

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

String: Неизменяемый тип (immutable). После создания объект String не может быть изменен. Любое изменение создает новый объект.

StringBuilder: Изменяемый тип (mutable). Объект StringBuilder может изменяться без создания новых объектов, что делает его более эффективным для частых операций изменения строки.

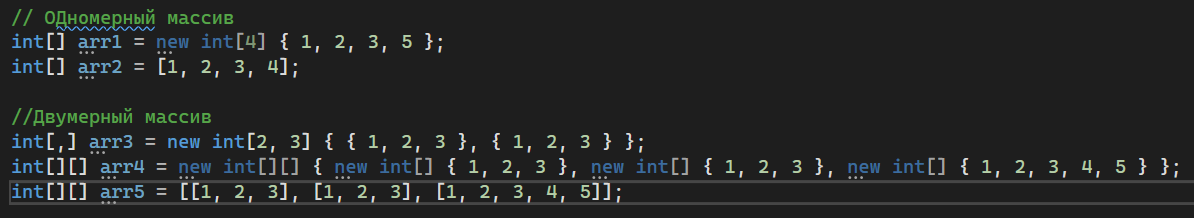
**31. Поясните явные преобразования переменных с помощью команд Convert.**

Класс Convert предоставляет методы для явного преобразования (преобразования типов) различных типов данных. Эти методы позволяют конвертировать значения одного типа в другой, обеспечивая при этом контроль над возможными исключениями и обеспечивая надежность кода. Методы класса Convert могут также округлять значения при преобразовании.

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

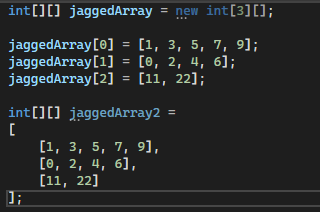
Используя методы ReadLine, Read, WriteLine, Write класса Console.

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



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

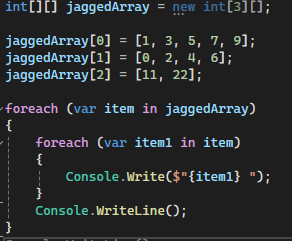
Cтупенчатый массив — это массив, элементами которого являются массивы, возможно, разных размеров. Cтупенчатый массив иногда называют «массивом массивов».



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

Оператор foreach выполняет оператор или блок операторов для каждого элемента в экземпляре типа, который реализует интерфейс System.Collections.IEnumerable или System.Collections.Generic.IEnumerable<T>. Его можно использовать с экземпляром любого типа, который удовлетворяет следующим условиям:

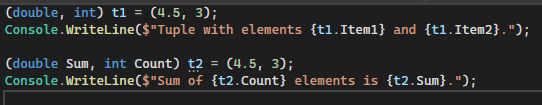
* Тип имеет общедоступный метод GetEnumerator без параметров. Метод GetEnumerator может быть методом расширения типа.
* Возвращаемый тип метода GetEnumerator имеет общедоступное свойство Current и общедоступный метод MoveNext без параметров, тип возвращаемого значения которого — bool.



[https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/statements/iteration-statements#the-foreach-statement](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/statements/iteration-statements%23the-foreach-statement)

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

Функция кортежей обеспечивает краткий синтаксис для группировки нескольких элементов данных в упрощенную структуру данных. В следующем примере показано, как можно объявить переменную-кортеж, инициализировать ее и получить доступ к ее элементам данных:



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

Локальные функции — это методы типа, вложенные в другой член. Их можно вызывать только из содержащего их члена. Локальные функции могут быть объявлены и вызваны из:

* Методы, особенно методы итераторов и асинхронные методы. Конструкторы.
* Средства доступа к свойствам
* Средства доступа к событиям
* Анонимные методы
* Лямбда-выражения
* Финализаторы
* Другие локальные функции

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

Операторы checked и unchecked задают контекст проверки переполнения для арифметических операций и преобразований целочисленного типа. Когда происходит переполнение целочисленной арифметики, контекст проверки переполнения определяет, что происходит. В checked выбрасывается исключение System.OverflowException; если в константном выражении происходит переполнение, возникает ошибка во времени компиляции. В unchecked результат операции усекается путем отбрасывания всех старших битов, которые не соответствуют типу назначения. Например, в случае сложения происходит переход от максимального значения к минимальному.

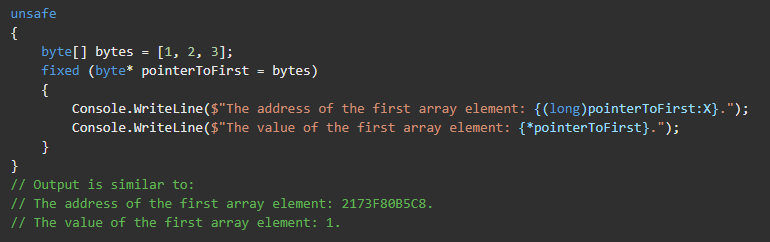
**39. Какой контекст (checked/unchecked) применяется по умолчанию? Как можно переопределить это поведение?**

Если вы не укажете контекст проверки переполнения, значение параметра компилятора CheckForOverflowUnderflow определяет контекст по умолчанию для непостоянных выражений. По умолчанию значение этой опции не установлено, а арифметические операции и преобразования целочисленного типа выполняются в непроверяемом контексте.

Переопределить контекст можно используя блоки checked {} и unchecked {}

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

Оператор fixed не позволяет сборщику мусора перемещать перемещаемую переменную и объявляет указатель на эту переменную. Адрес фиксированной или закрепленной переменной не меняется во время выполнения инструкции. Вы можете использовать объявленный указатель только внутри соответствующего фиксированного оператора. Объявленный указатель доступен только для чтения и не может быть изменен:



Вы можете использовать оператор fixed только в небезопасном контексте. Код, содержащий небезопасные блоки, должен быть скомпилирован с параметром компилятора AllowUnsafeBlocks.