### 1. Что такое .NET Framework и из чего он состоит?

.NET Framework — это программная платформа от Microsoft, предназначенная для разработки и выполнения приложений, поддерживающих несколько языков программирования. Она включает в себя:

- CLR (Common Language Runtime) — исполняющая среда, которая управляет выполнением кода.

- FCL (Framework Class Library) — библиотека классов, содержащая базовые классы для работы с данными, файлами, сетью, потоками и т.д.

- Компиляторы для различных языков (C#, VB.NET и др.).

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

- CLR (Common Language Runtime) — среда выполнения, которая управляет выполнением .NET приложений, управляет памятью, безопасностью и потоками.

- FCL (Framework Class Library) или BCL (Base Class Library) — набор базовых классов .NET, таких как System, System.IO, System.Collections.

- CLI (Common Language Infrastructure) — стандарт, описывающий особенности разработки и выполнения кода на разных языках в .NET.

- IL (Intermediate Language) — промежуточный язык, в который компилируется исходный код на языках .NET перед выполнением.

### 3. Пояснить работу JIT-компилятора?

JIT (Just-In-Time) компилятор выполняет преобразование IL-кода в машинный код во время выполнения программы. Когда метод вызывается впервые, JIT компилирует его в код, понятный процессору.

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

CTS (Common Type System) — это система, которая определяет, как типы данных объявляются, используются и управляются в .NET. Она обеспечивает совместимость между языками, используемыми на платформе.

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

System.Object — это базовый класс для всех типов в .NET. Он определяет методы:

- ToString() — возвращает строковое представление объекта.

- Equals() — сравнивает два объекта на равенство.

- GetHashCode() — возвращает хэш-код объекта.

- GetType() — возвращает тип объекта.

### 6. Что находится в mscorlib.dll?

mscorlib.dll — это основная библиотека .NET Framework, содержащая базовые классы (System, System.IO, System.Threading) и реализации базовых типов данных (int, string и т.д.).

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

Сборка — это скомпилированный код в формате, который может исполняться CLR. Сборка состоит из:

- IL-кода.

- Метаданных.

- Манифеста сборки.

- Ресурсов (например, изображения, файлы).

### 8. Какие виды сборок существуют?

- Частные сборки — используются только внутри одного приложения.

- Общие сборки — могут использоваться несколькими приложениями и хранятся в GAC.

- Сателлитные сборки — содержат ресурсы для локализации.

### 9. Что такое assembly manifest?

Манифест сборки — это часть сборки, которая содержит метаданные о сборке, такие как версия, имя сборки, список файлов, зависимости и информация о безопасности.

### 10. Что такое GAC?

GAC (Global Assembly Cache) — это глобальный кэш сборок, где хранятся общие сборки, которые могут использоваться несколькими приложениями.

### 11. Чем managed code отличается от unmanaged code?

- Managed code — это код, управляемый CLR, который предоставляет управление памятью, безопасностью и выполнением.

- Unmanaged code — код, который выполняется вне CLR, например, код на C++.

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

Метод Main — это входная точка приложения в C#. Именно с него начинается выполнение программы. Он может возвращать void или int и может иметь параметры string[].

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

- Using директива — используется для подключения пространств имен.

- Using конструкция — используется для автоматического освобождения ресурсов (IDisposable), например, для работы с файлами.

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

Пространства имен организуют классы, структуры и другие типы внутри сборки. Сборка может содержать несколько пространств имен, а одно пространство имен может быть распределено между разными сборками.

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

Примитивные типы данных — это встроенные типы данных:

- Целые: byte, sbyte, short, ushort, int, uint, long, ulong.

- Вещественные: float, double, decimal.

- Символьные: char.

- Логические: bool.

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

Ссылочные типы хранят ссылку на объект в памяти:

- string

- Классы (class)

- Интерфейсы (interface)

- Делегаты (delegate)

- Массивы.

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

Типы-значения хранят данные непосредственно в памяти:

- Примитивные типы (int, bool).

- Структуры (struct).

- Перечисления (enum).

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

- Ссылочные типы хранят адрес в памяти, где хранятся данные.

- Значимые типы хранят сами данные непосредственно.

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

Упаковка — преобразование типа-значения в объект, оборачивающий этот тип в память кучи. Распаковка — извлечение значимого типа из объекта.

### 20. В чем заключается разница между int и System.Int32? double и System.Double?

int и System.Int32, double и System.Double — это одно и то же. Первое — это синтаксический сахар для базового типа .NET.

### 21. Для чего используется тип dynamic?

Тип dynamic используется для выполнения операций в рантайме без компиляционной проверки типов. Полезен при работе с COM, динамическими библиотеками.

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

- var — тип выводится на этапе компиляции.

- dynamic — тип выводится на этапе выполнения.

### 23. Что такое неявно типизированная переменная?

Неявно типизированная переменная используется с var, когда тип переменной выводится компилятором на основе присваиваемого значения.

### 24. Для чего используют Nullable тип?

Nullable тип (Nullable<T>) позволяет хранить значения типов-значений, которые могут быть null, например, int?.

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

Строковые литералы объявляются с помощью кавычек: "Hello". Операции: конкатенация, получение подстроки, сравнение, замена, поиск и т.д.

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

Строки можно задавать через обычные и интерполированные строки:

- string s = "Hello";

- string s = $"Hello, {name}";

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

Методы: Length, Substring, IndexOf, Replace, ToLower, ToUpper, Split и другие.

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

- Пустая строка — это строка с длиной 0 ("").

- null строка — это отсутствие строки.

### 29. Как можно выполнить сравнение строк?

Сравнение выполняется с помощью метода String.Equals или оператора ==.

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

String — неизменяемый тип. При изменении строки создается новая. StringBuilder — изменяемый, что делает его эффективнее при частых модификациях.

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

Команды Convert используются для явного преобразования типов, например, Convert.ToInt32(), Convert.ToString().

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

- Ввод: Console.ReadLine().

- Вывод: Console.WriteLine().

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

Одномерный: int[] arr = new int[] {1, 2, 3};.

Двумерный: `int[,]

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

**Ступенчатый массив** (jagged array) — это массив массивов, где каждый элемент может содержать массив различной длины. В отличие от двумерного массива, где все строки имеют одинаковую длину, в ступенчатом массиве каждая строка может быть разной.

**Пример создания ступенчатого массива на C#:**

csharp

Копировать код

int[][] jaggedArray = new int[3][]; // Массив из трех массивов

jaggedArray[0] = new int[] { 1, 2 };

jaggedArray[1] = new int[] { 3, 4, 5 };

jaggedArray[2] = new int[] { 6, 7, 8, 9 };

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

В **foreach** можно использовать любые типы, которые реализуют интерфейсы IEnumerable или IEnumerable<T>. Это включает массивы, списки, строки и любые другие коллекции.

**Пример использования foreach с массивом:**

csharp

Копировать код

int[] numbers = { 1, 2, 3, 4 };

foreach (int number in numbers)

{

Console.WriteLine(number);

}

**Пример использования foreach с List<T>:**

csharp

Копировать код

List<string> fruits = new List<string> { "Apple", "Banana", "Orange" };

foreach (string fruit in fruits)

{

Console.WriteLine(fruit);

}

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

**Кортеж** (Tuple) — это структура данных, позволяющая хранить несколько значений разных типов как единое целое. Кортежи удобны для возвращения нескольких значений из метода или для временного хранения данных без необходимости создания отдельного класса.

**Пример создания кортежа:**

csharp

Копировать код

var tuple = (1, "Hello", true);

Console.WriteLine(tuple.Item1); // 1

Console.WriteLine(tuple.Item2); // Hello

Console.WriteLine(tuple.Item3); // True

Также можно использовать именованные элементы:

csharp

Копировать код

var person = (Name: "John", Age: 30);

Console.WriteLine(person.Name); // John

Console.WriteLine(person.Age); // 30

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

**Локальная функция** — это функция, объявленная внутри другой функции (метода). Она может использовать параметры и переменные внешней функции, но видна только в её контексте (области видимости).

**Пример локальной функции:**

csharp

Копировать код

void OuterFunction()

{

int x = 10;

void LocalFunction()

{

Console.WriteLine(x); // Локальная функция имеет доступ к переменной x

}

LocalFunction();

}

Локальная функция доступна только в рамках метода, внутри которого она объявлена.

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

* **checked** — используется для включения проверки на переполнение арифметических операций. Если возникает переполнение, генерируется исключение OverflowException.
* **unchecked** — используется для отключения проверки на переполнение, что позволяет игнорировать переполнение и обрезать значения.

**Пример:**

csharp

Копировать код

int x = int.MaxValue;

checked

{

x++; // Генерируется исключение переполнения

}

unchecked

{

x++; // Переполнение игнорируется, и результат — минимальное значение int

}

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

По умолчанию используется **контекст unchecked** для арифметических операций, что означает, что переполнение игнорируется.

Чтобы переопределить это поведение:

* Локально для блока кода можно использовать ключевые слова **checked** или **unchecked**.
* В настройках проекта можно задать поведение по умолчанию как **checked** через компилятор C#.

**Пример явного указания контекста:**

csharp

Копировать код

int x = int.MaxValue;

int result = checked(x + 1); // Переполнение вызывает исключение

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

Ключевое слово **fixed** используется для работы с указателями в небезопасном коде. Оно фиксирует объект в памяти, предотвращая его перемещение сборщиком мусора (GC), что необходимо для прямого доступа к объектам через указатели.

**Пример использования fixed:**

csharp

Копировать код

unsafe

{

int[] numbers = { 1, 2, 3, 4 };

fixed (int\* p = numbers)

{

for (int i = 0; i < 4; i++)

{

Console.WriteLine(p[i]); // Прямой доступ к элементам массива через указатель

}

}

}

**Особенности:**

* Используется в небезопасном контексте (unsafe).
* Необходим для работы с указателями на управляемые объекты (например, массивы).
* Важно освобождать объект после завершения работы с указателями, чтобы позволить GC снова управлять его перемещением.