Давайте по порядку разберем все вопросы по C# и .NET Framework:

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

- .NET Framework — это платформа для разработки и выполнения приложений, предоставляющая API для работы с операционной системой, управления памятью и безопасности, а также языки программирования.

- Основные компоненты .NET Framework:

- \*\*CLR (Common Language Runtime)\*\* — среда выполнения, отвечающая за выполнение и управление кодом.

- \*\*FCL/BCL (Framework/BCL Class Library)\*\* — библиотека классов, предоставляющая базовые классы для работы с файлами, потоками, строками и др.

- \*\*CLI (Common Language Infrastructure)\*\* — стандарт, описывающий среду выполнения программ.

- \*\*IL (Intermediate Language)\*\* — промежуточный язык, в который компилируется код перед его выполнением.

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

- \*\*CLR (Common Language Runtime)\*\* — виртуальная машина для выполнения кода на платформе .NET. Отвечает за управление памятью, безопасность, сборку мусора.

- \*\*FCL (Framework Class Library)\*\* — полная библиотека классов, входящая в .NET Framework. \*\*BCL (Base Class Library)\*\* — ее базовая часть.

- \*\*CLI (Common Language Infrastructure)\*\* — стандарт для создания приложений с поддержкой работы на разных языках программирования.

- \*\*IL (Intermediate Language)\*\* — промежуточный язык, в который компилируется код .NET перед выполнением.

### 3. \*\*Пояснить работу JIT-компилятора\*\*

- \*\*JIT (Just-in-Time) компилятор\*\* — это компилятор, который компилирует IL код в машинный код непосредственно перед его выполнением. Это позволяет оптимизировать работу программы и учитывать особенности среды выполнения.

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

- \*\*CTS (Common Type System)\*\* — это система, которая определяет, как должны работать все типы данных в .NET. Она гарантирует совместимость типов, определенных на разных языках программирования, внутри .NET Framework.

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

- \*\*System.Object\*\* — базовый класс для всех типов данных в .NET. Он определяет общие методы:

- `ToString()` — возвращает строковое представление объекта.

- `Equals()` — проверяет равенство объектов.

- `GetHashCode()` — возвращает хэш-код объекта.

- `GetType()` — возвращает тип объекта.

### 6. \*\*Что находится в mscorlib.dll?\*\*

- \*\*mscorlib.dll\*\* — это библиотека, содержащая основные классы .NET Framework, включая типы для работы с коллекциями, строками, числами, потоками и базовыми типами данных (например, `System.Int32`, `System.String`).

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

- \*\*Сборка (Assembly)\*\* — это единица развертывания и выполнения приложения в .NET. Состоит из:

- IL-кода

- Метаданных

- Манифеста сборки (assembly manifest), который содержит информацию о сборке, версиях и зависимостях.

### 8. \*\*Какие виды сборок существуют?\*\*

- \*\*Частные сборки\*\* — используются только внутри одного приложения.

- \*\*Общие сборки\*\* — могут использоваться несколькими приложениями и хранятся в глобальном кэше сборок (GAC).

### 9. \*\*Что такое assembly manifest?\*\*

- \*\*Манифест сборки\*\* — это метаданные, которые содержат информацию о сборке: версия, культура, зависимости, информация о типах и ресурсах.

### 10. \*\*Что такое GAC?\*\*

- \*\*GAC (Global Assembly Cache)\*\* — это место, где хранятся общие сборки, доступные для всех приложений на компьютере.

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

- \*\*Managed code\*\* — код, который выполняется под управлением CLR. Он управляется системой сборки мусора и безопасен с точки зрения управления памятью.

- \*\*Unmanaged code\*\* — код, который напрямую управляет памятью и выполняется без CLR, например, код на C++.

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

- \*\*Main()\*\* — это точка входа в C# программу. Он может возвращать `void` или `int`, и может принимать аргументы командной строки в виде массива строк `string[] args`.

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

- Используется для:

1. Объявления пространства имен: `using System;`.

2. Автоматического освобождения ресурсов: `using (var stream = new StreamWriter(...)) { ... }`.

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

- Пространства имен могут располагаться в разных сборках, и сборка может содержать несколько пространств имен.

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

- Примитивные типы данных — это встроенные типы, такие как:

- `int`, `double`, `bool`, `char`, `float`, `byte`, `short`, `long`, `decimal`.

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

- \*\*Ссылочные типы\*\* — это типы данных, которые хранятся в куче, и переменная хранит ссылку на объект:

- `string`, классы (`class`), интерфейсы, делегаты, массивы.

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

- \*\*Значимые типы\*\* хранятся в стеке:

- Все примитивные типы (`int`, `double`), структуры (`struct`), перечисления (`enum`).

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

- Значимые типы копируются целиком при передаче, а ссылочные типы передают только ссылку на объект в памяти.

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

- \*\*Упаковка (boxing)\*\* — преобразование значимого типа в объект (ссылочный тип).

- \*\*Распаковка (unboxing)\*\* — обратное преобразование объекта в значимый тип.

### 20. \*\*В чем заключается разница между int и System.Int32?\*\*

- \*\*int\*\* — это алиас для \*\*System.Int32\*\*. Они полностью идентичны.

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

- \*\*dynamic\*\* позволяет использовать переменные, тип которых определяется на этапе выполнения, а не компиляции.

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

- \*\*var\*\* — это статически типизированная переменная, тип которой определяется во время компиляции.

- \*\*dynamic\*\* — переменная с типом, который определяется во время выполнения.

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

- \*\*Неявно типизированная переменная\*\* — переменная, тип которой выводится компилятором автоматически на основе присвоенного значения (используется `var`).

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

- \*\*Nullable<T>\*\* позволяет значимым типам, таким как `int` или `bool`, принимать значение `null`.

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

- Строковые литералы объявляются с использованием кавычек: `"Hello"`.

- Основные операции: конкатенация, сравнение, поиск подстроки, изменение регистра.

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

- Строки можно инициализировать через литералы: `string s = "Hello";`.

- Использовать интерполяцию строк: `$"Hello, {name}!"`.

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

- `Substring()`, `Contains()`, `IndexOf()`, `ToLower()`, `ToUpper()`, `Replace()` и др.

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

- \*\*Пустая строка\*\* (`""`) имеет длину 0, но существует. \*\*null\*\* означает, что строки нет.

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

- Через методы: `String.Equals()`, операторы `==` или `!=`.

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

- \*\*String\*\* — неизменяемый тип, каждая операция создает новую строку.

- \*\*StringBuilder\*\* — изменяемый тип, который позволяет эффективно изменять содержимое строки.

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

- Используется класс `Convert` для преобразования типов: `Convert.ToInt32()`, `Convert

.ToString()`.

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

- Для ввода: `Console.ReadLine()`.

- Для вывода: `Console.WriteLine()`.

### 33. \*\*Примеры определения и инициализации одномерных и двумерных массивов.\*\*

- Одномерный массив: `int[] array = new int[5];`.

- Двумерный массив: `int[,] matrix = new int[2, 3];`.

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

- \*\*Ступенчатый массив\*\* — массив массивов. Пример: `int[][] jaggedArray = new int[3][];`.

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

- Любой тип, реализующий интерфейс `IEnumerable`. Пример:

```csharp

foreach (int num in array) { ... }

```

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

- \*\*Кортеж\*\* — это структура, которая позволяет хранить несколько значений разных типов. Пример:

```csharp

(int, string) person = (1, "John");

```

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

- \*\*Локальная функция\*\* — функция, объявленная внутри другой функции. Ее область видимости ограничена содержащей функцией.

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

- \*\*checked\*\* проверяет арифметические переполнения и выбрасывает исключение.

- \*\*unchecked\*\* игнорирует переполнения.

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

- По умолчанию используется \*\*unchecked\*\*. Можно явно задать контекст с помощью ключевых слов `checked` или `unchecked`.

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

- \*\*fixed\*\* закрепляет объект в памяти, чтобы указатель на него не был перемещен сборщиком мусора. Используется в небезопасном коде для работы с указателями.

### 40. \*\*Область видимости функций