Вот ответы на ваши вопросы по .NET и C#:

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

\*\*.NET Framework\*\* — это программная платформа от Microsoft для создания и выполнения приложений на Windows. Он состоит из нескольких ключевых компонентов:

- \*\*Common Language Runtime (CLR)\*\*: Исполняющая среда, которая управляет выполнением программ и предоставляет услуги, такие как сборка мусора и управление памятью.

- \*\*Framework Class Library (FCL)\*\*: Библиотека классов, предоставляющая широкий набор функций для выполнения различных задач, таких как работа с файлами, сетью и графическим интерфейсом.

- \*\*Base Class Library (BCL)\*\*: Подмножество FCL, которое содержит основные классы и типы данных, используемые в .NET.

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

- \*\*CLR (Common Language Runtime)\*\*: Исполняющая среда .NET, которая управляет выполнением программ, обеспечивает безопасность и управление памятью.

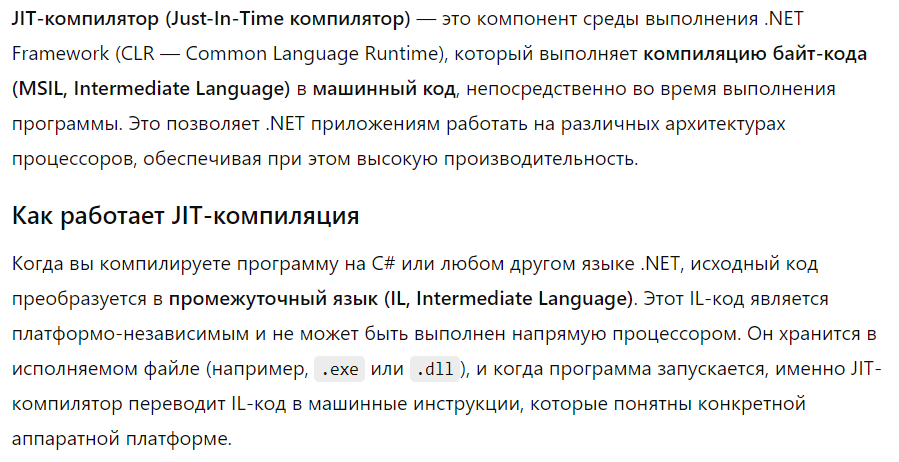
- \*\*FCL (Framework Class Library)\*\*: Библиотека классов, предоставляющая функциональные возможности для приложений .NET.

- \*\*BCL (Base Class Library)\*\*: Основная часть FCL, содержащая базовые классы, такие как коллекции, ввод/вывод и базовые типы данных.

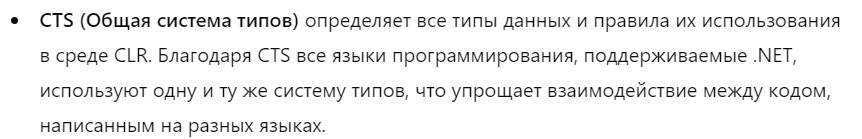
- \*\*CLI (Common Language Infrastructure)\*\*: Спецификация, которая описывает стандартные способы взаимодействия языков программирования и платформы .NET. Включает CLR, CTS, и IL.

- \*\*IL (Intermediate Language)\*\*: Промежуточный язык, в который компилируется код .NET. Код IL затем исполняется CLR.

### 3. Поясните работу JIT-компилятора.



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



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

\*\*System.Object\*\* — это базовый тип для всех типов в .NET. Он определяет основные методы, которые все типы наследуют:

- `Equals()`: Сравнение объектов.

- `GetHashCode()`: Получение хэш-кода объекта.

- `GetType()`: Получение типа объекта.

- `ToString()`: Преобразование объекта в строку.

### 6. Что находится в mscorlib.dll?

\*\*mscorlib.dll\*\* содержит основные классы .NET Framework, включая типы данных, классы ввода/вывода, коллекции и основные функциональные возможности, такие как `System.String`, `System.Int32` и `System.Object`.

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

\*\*Сборка\*\* — это абстрактное понятие, для логической группировки одного или нескольких управляемых модулей или файлов ресурсов.

. Сборка состоит из:

- \*\*Манифеста сборки\*\*: Информация о сборке, включая ее имя, версию и список зависимостей.

- \*\*Метаданных\*\*: Описание типов, методов и других данных.

- \*\*Кода\*\*: IL-код, который будет исполняться CLR.

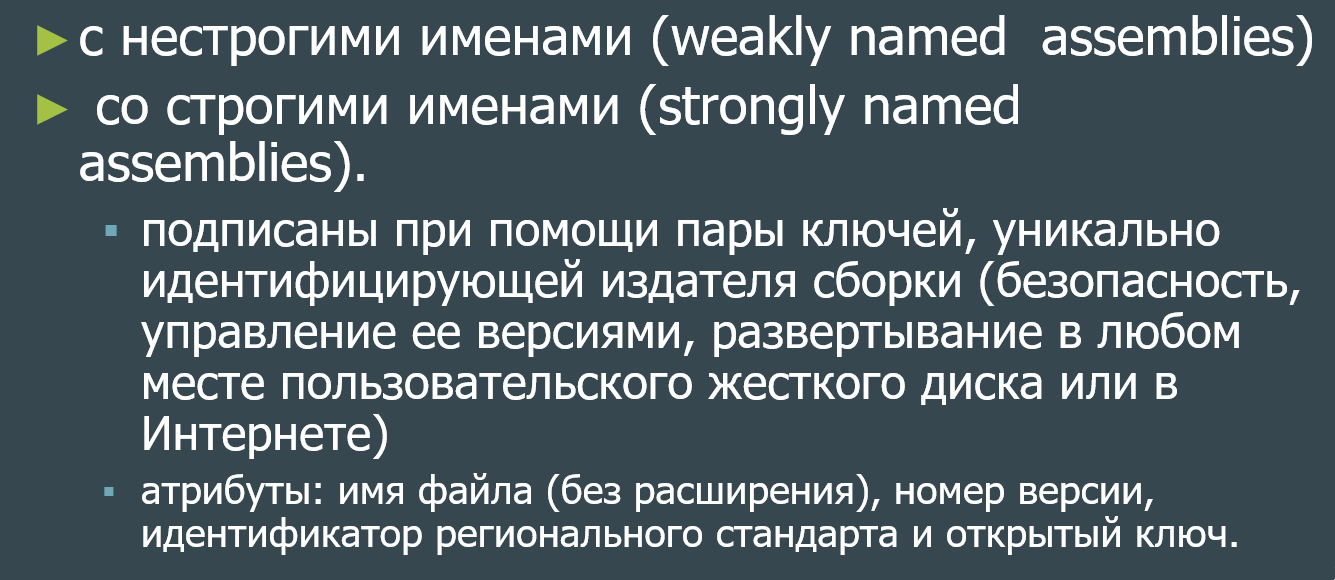
- \*\*Ресурсов\*\*: Включает изображения, файлы и другие ресурсы, используемые сборкой.

### 8. Какие виды сборок существуют?

- \*\*Private Assembly\*\*: Используется только в рамках одного приложения.

- \*\*Shared Assembly\*\*: Хранится в Global Assembly Cache (GAC) и доступна для нескольких приложений.

- \*\*Satellite Assembly\*\*: Содержит ресурсы, локализованные для различных языков.



### 9. Что такое assembly manifest?

\*\*Assembly Manifest\*\* — это часть сборки, содержащая метаданные, такие как название сборки, версия, культура и список зависимостей. Он также описывает, какие файлы содержатся в сборке.

### 10. Что такое GAC?

GAC, или **Global Assembly Cache** (Глобальный Кэш Сборок), – это специальное хранилище в операционной системе Windows, предназначенное для хранения сборок .NET, которые могут использоваться несколькими приложениями на одном компьютере.

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

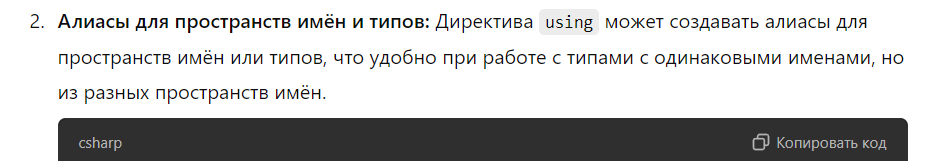
\*\*Managed Code\*\* — это код, который выполняется под управлением CLR, что обеспечивает управление памятью, безопасность и исключения. \*\*Unmanaged Code\*\* выполняется напрямую на операционной системе, не имея этих преимуществ.

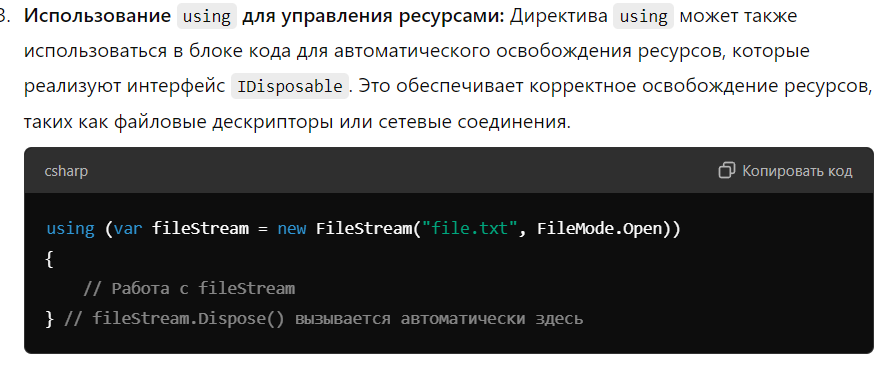
### 12. Как и для чего определен метод Main?

Метод `Main` является точкой входа в приложение на C#. Он должен быть объявлен как `static` и может возвращать `void` или `int`, и принимать массив строк `string[]` в качестве параметра. Он определяет, с чего начинается выполнение программы.

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

Импорт пространств имён



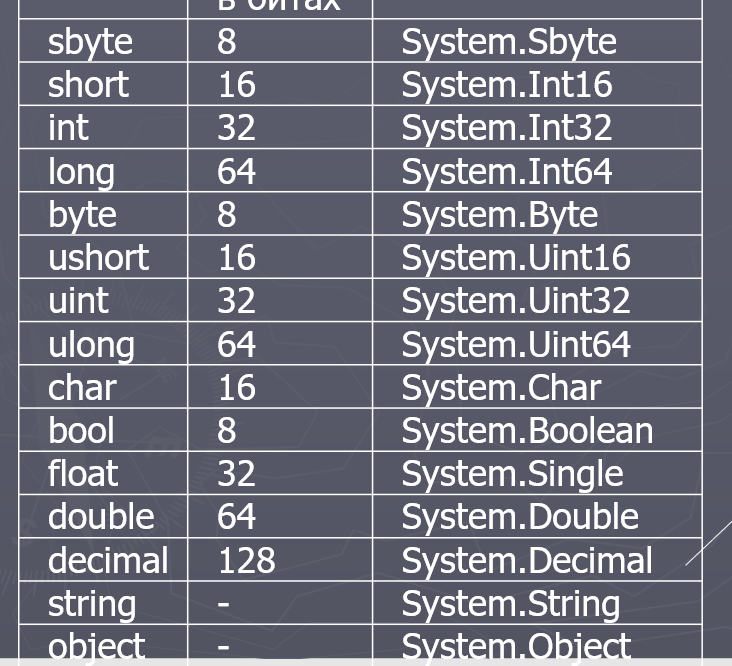


### 14. Как связаны между собой сборки и пространства имен?

Пространства имен используются для организации кода внутри сборки и предотвращения конфликтов имен. Сборка может содержать несколько пространств имен.

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

Примитивные типы данных в C# включают:



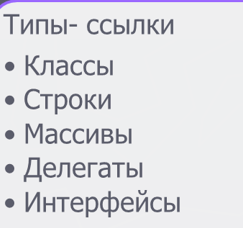
### 16. Что такое ссылочные типы? Какие типы относятся к ним?

\*\*Ссылочные типы\*\* хранят ссылки на объекты в памяти. Примеры:

- `string`

- `object`

- Классы (`class`)



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

\*\*Типы-значения\*\* хранят данные непосредственно. Примеры:

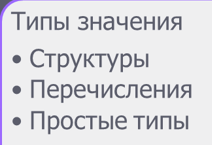
- `int`

- `float`

- `bool`

- `char`

- `struct`



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

- \*\*Ссылочные типы\*\* хранят ссылку на объект в памяти и могут быть `null`. Изменения в объекте, на который ссылается переменная, затрагивают все переменные, ссылающиеся на этот объект.

- \*\*Типы-значения\*\* хранят данные непосредственно и не могут быть `null`. Изменение значения в одной переменной не затрагивает другие переменные.

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

- \*\*Упаковка\*\* (`boxing`) — это процесс преобразования типа-значения в тип-объект (например, из `int` в `object`).

- \*\*Распаковка\*\* (`unboxing`) — это преобразование объекта обратно в тип-значение (например, из `object` в `int`).

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

`int` и `System.Int32` (и аналогично `double` и `System.Double`) являются синонимами в C#. `int` — это псевдоним для `System.Int32`, и они взаимозаменяемы. Эти псевдонимы используются для удобства и совместимости.

### 21. Для чего используется тип dynamic?

\*\*dynamic\*\* позволяет работать с объектами, тип которых неизвестен на этапе компиляции. Это позволяет выполнять операции, которые будут проверены только во время выполнения, например, динамическое взаимодействие с COM-объектами или вызовы методов, которые не существуют на этапе компиляции.

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

- \*\*var\*\* — это ключевое слово для неявного типизирования. Тип переменной определяется на этапе компиляции на основе присвоенного значения.

- \*\*dynamic\*\* — это ключевое слово для динамической типизации. Тип переменной определяется на этапе выполнения, что позволяет избежать проверок на этапе компиляции.

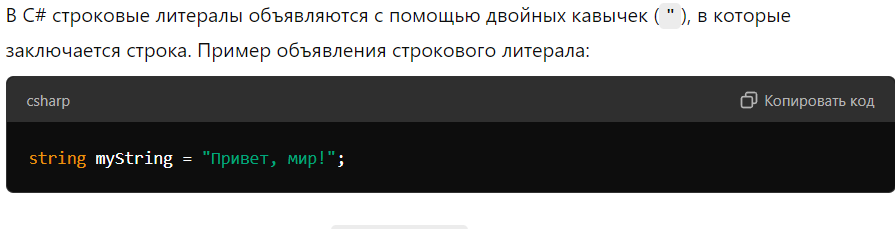
### 23. Что такое неявно типизированная переменная?

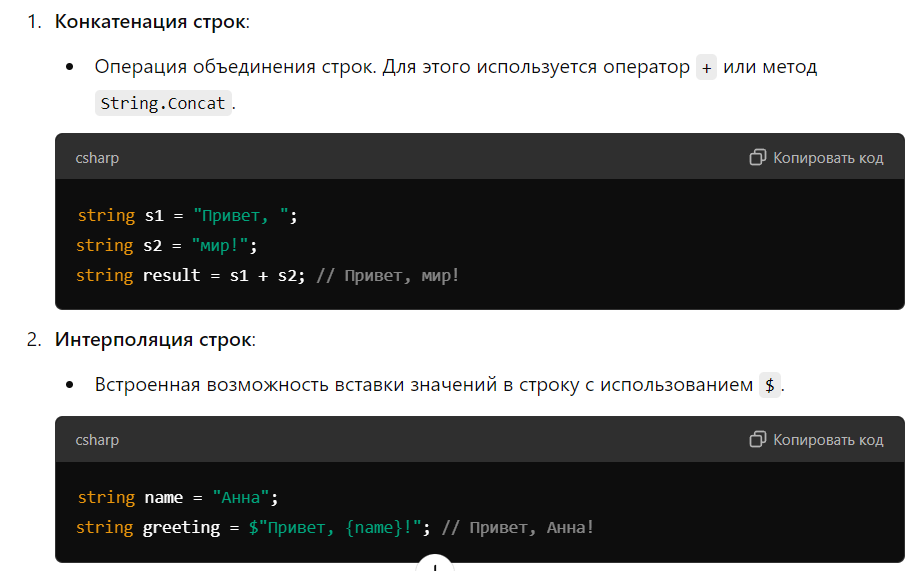
\*\*Неявно типизированная переменная\*\* — это переменная, тип которой определяется на основе присвоенного значения с помощью ключевого слова `var`. Тип переменной определится только на этапе компиляции.

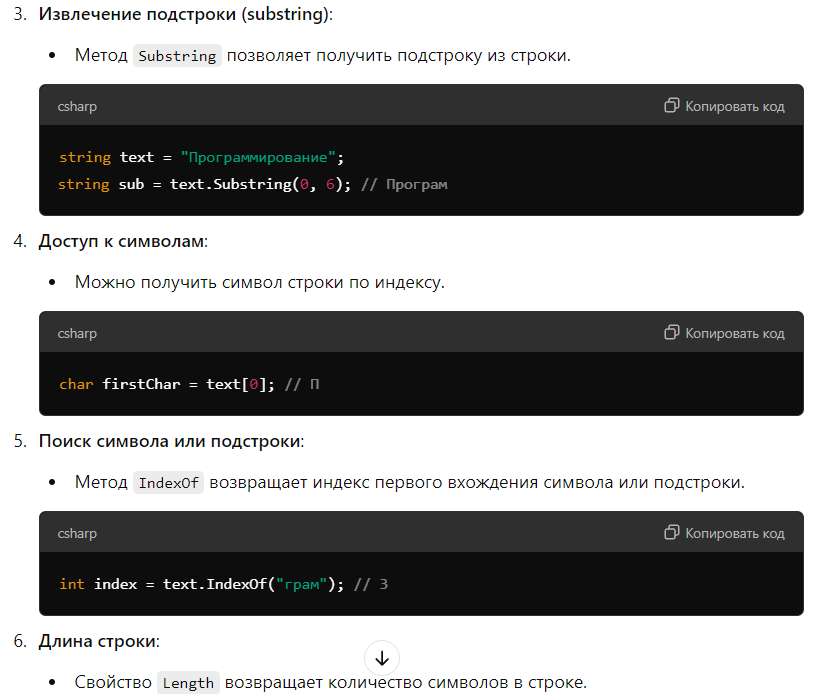
### 24. Для чего используют Nullable тип?

\*\*Nullable типы\*\* (например, `int?`, `double?`) позволяют переменным типа-значения принимать значение `null`. Это полезно для работы с базами данных или когда требуется обозначить отсутствие значения.

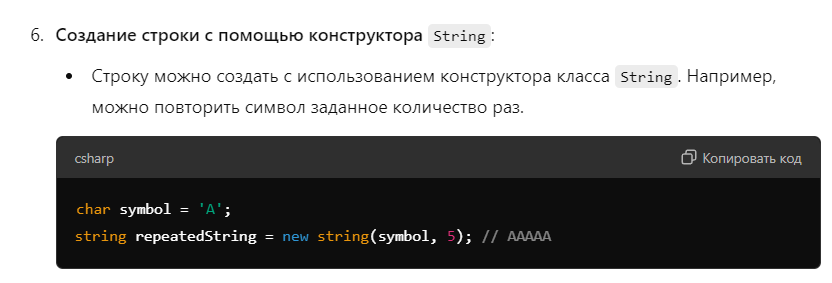
### 25. Как объявить строковый литерал? Какие операции можно выполнять со строкой?

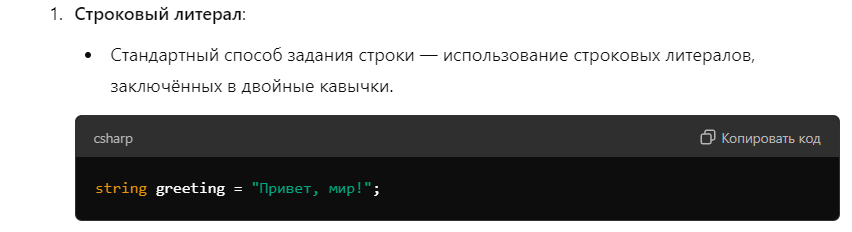


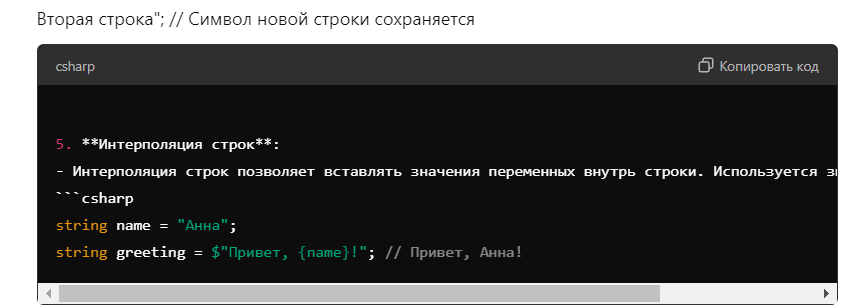




### 26. Какие есть способы для задания и инициализации строк?

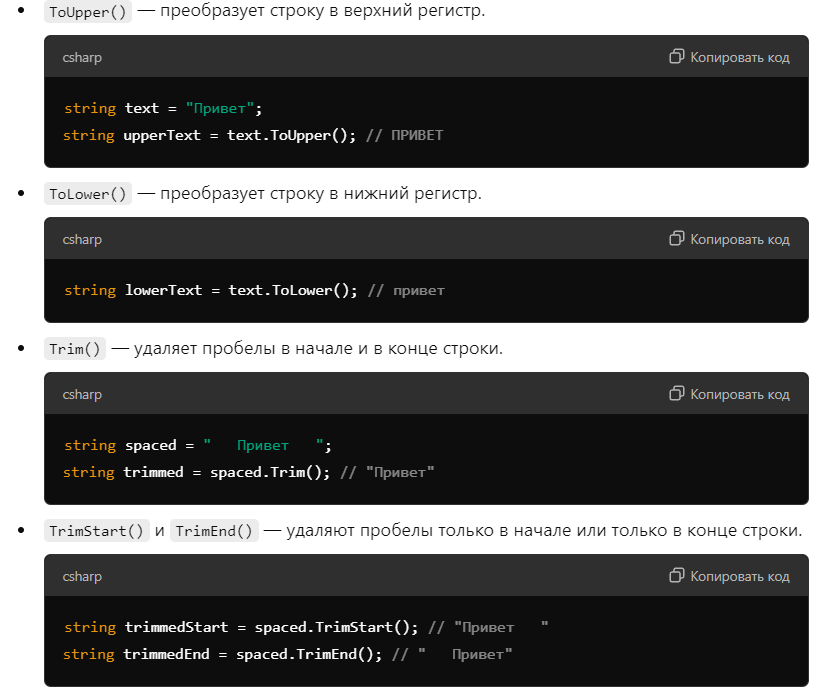


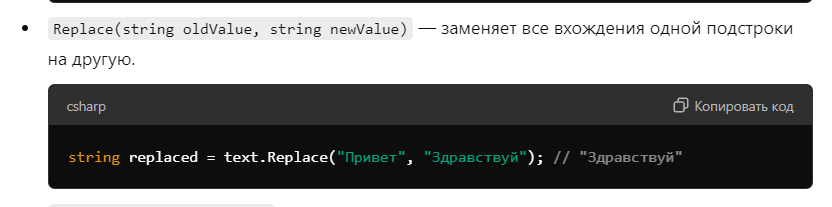


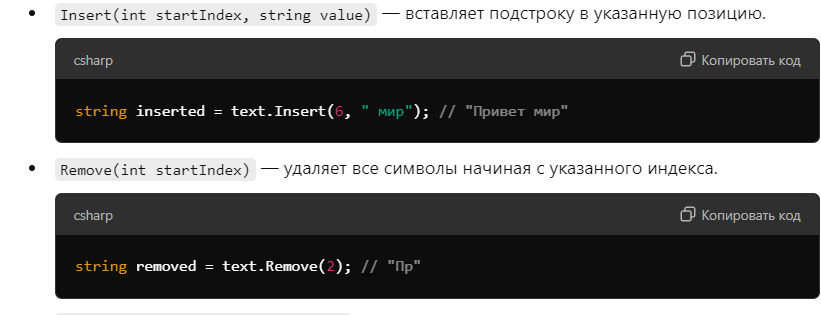


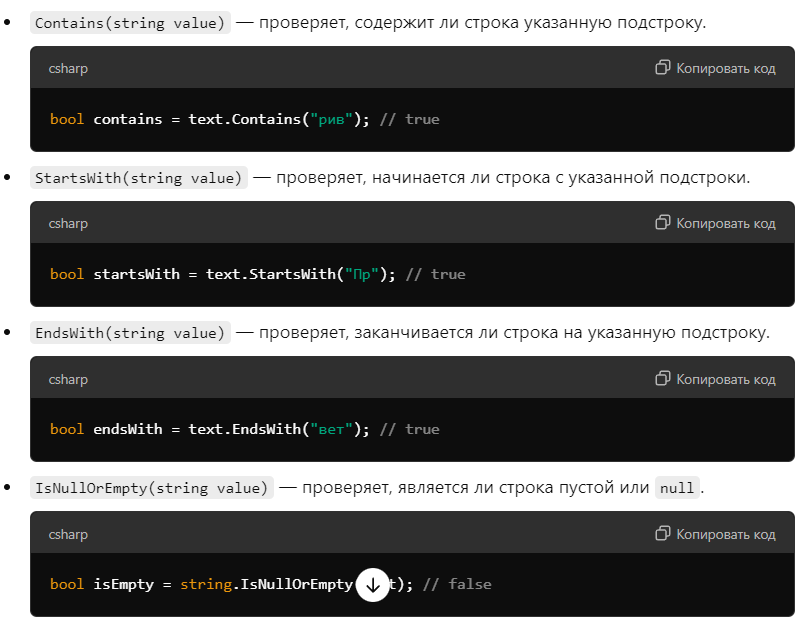
### 27. Какие методы есть у типа String?

Методы типа `String` включают:







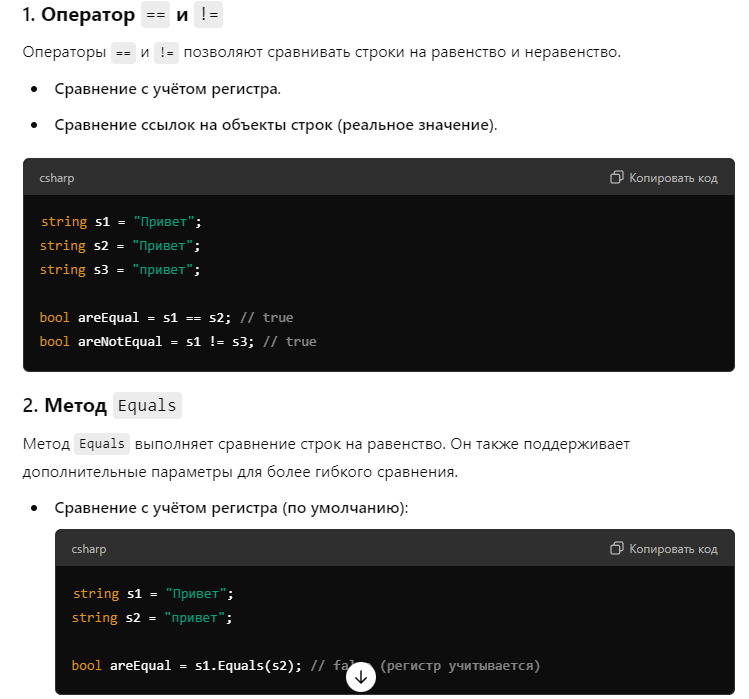


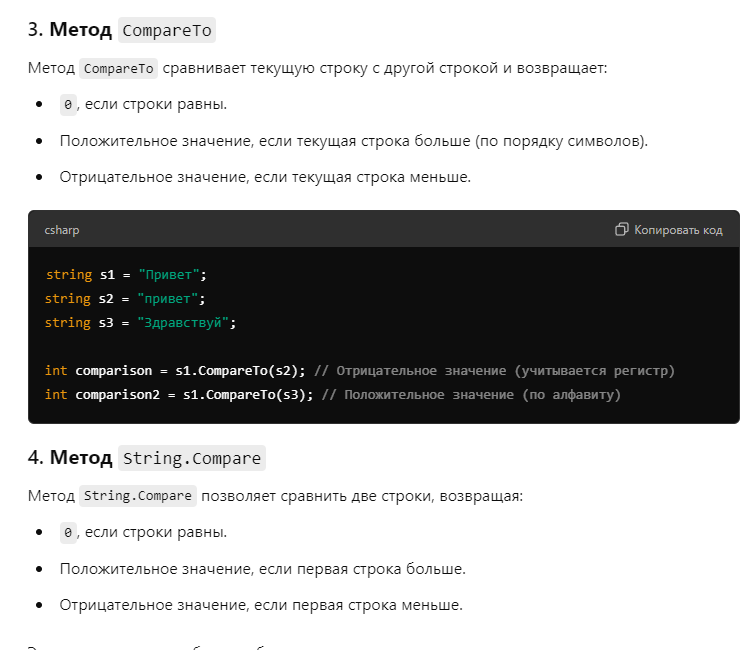
### 28. В чем отличие пустой и null строки?

- \*\*Пустая строка\*\* (`""`) — это строка с нулевой длиной, но она все еще существует.

- \*\*Null строка\*\* (`null`) — это отсутствие строки, что означает, что переменная не ссылается на какой-либо объект.

### 29. Как можно выполнить сравнение строк?





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

- \*\*String\*\* — это неизменяемый тип. Любые изменения создают новый объект строки.

- \*\*StringBuilder\*\* — это изменяемый тип, предназначенный для эффективной работы с большим количеством изменений в строке.

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

Класс `Convert` предоставляет методы для явного преобразования типов, такие как:

- `Convert.ToInt32(value)`

- `Convert.ToDouble(value)`

- `Convert.ToString(value)`

Эти методы выполняют преобразования с учетом возможных исключений и специфик типов.

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

- \*\*Ввод\*\*: `string input = Console.ReadLine();`

- \*\*Вывод\*\*: `Console.WriteLine("Hello, world!");`

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

- \*\*Одномерный массив\*\*:

int[] numbers = new int[] { 1, 2, 3, 4, 5 };

- \*\*Двумерный массив\*\*:

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

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

\*\*Ступенчатый массив\*\* (или невыравненный массив) — это массив массивов, где каждый подмассив может иметь разную длину.

```csharp

double[][] jaggedArray = new double[3][];

jaggedArray[0] = new double[2];

jaggedArray[1] = new double[3];

jaggedArray[2] = new double[4];

```

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

массивы, списки, словари, строки и другие коллекции.

В `foreach` можно использовать коллекции, массивы и любые другие объекты, которые реализуют интерфейс `IEnumerable`:

```csharp

foreach (int number in numbers)

{

Console.WriteLine(number);

}

```

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

\*\*Кортеж\*\* — это структура данных, которая может содержать несколько элементов различных типов. Он используется для группировки нескольких значений в один объект.

```csharp

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

Console.WriteLine(person.Name);

```

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

\*\*Локальная функция\*\* — это функция, определенная внутри другой функции. Ее область видимости ограничена родительской функцией.

```csharp

void Main()

{

void LocalFunction()

{

Console.WriteLine("Hello from local function");

}

LocalFunction();

}

```

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

- \*\*checked\*\*: Переполнение целочисленных операций приводит к исключению `OverflowException`.

- \*\*unchecked\*\*: Переполнение игнорируется, и результат может быть некорректным, но не вызывает исключений.

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

По умолчанию, если не указан контекст, применяется `unchecked` (без проверки переполнения). Для переопределения поведения можно использовать блоки `checked` или `unchecked`.

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

\*\*fixed\*\* используется для фиксации объекта в памяти, что позволяет работать с указателями. Это необходимо, чтобы избежать перемещения объекта сборщиком мусора.

```csharp

unsafe

{

int[] array = new int[10];

fixed (int\* p = array)

{

// работа с указателем p

}

}

```

Использование ключевого слова `fixed` возможно только в небезопасном контексте (`unsafe`).