1. Тип char представляет собой символьный тип данных, который может хранить один символ Unicode. Допустимые значения для переменных типа char - символы Unicode от U+0000 до U+FFFF. Класс Char содержит методы и свойства для работы со значениями типа char, включая проверку на число, букву, символ пунктуации, регистр, преобразование к верхнему или нижнему регистру и т.д.
2. Тип string представляет собой последовательность символов Unicode. Строки могут быть созданы с помощью литералов (например, "hello world"), конструктора класса string (например, new string('a', 10) создает строку из 10 символов 'a') и статических методов класса string (например, string.Concat("hello", "world") объединяет две строки в одну).
3. Операции над строками типа string включают конкатенацию (соединение строк), сравнение, поиск подстрок, замену, выделение подстрок и т.д.
4. Класс String содержит множество методов и свойств для работы со строками, включая методы сравнения, поиска, замены, разделения, конкатенации, преобразования регистра и т.д.
5. Класс StringBuilder представляет собой изменяемую последовательность символов. Конструкторы класса StringBuilder позволяют создавать пустой объект StringBuilder или объект StringBuilder, содержащий заданное начальное количество символов.
6. Операции над строками StringBuilder включают добавление символов, удаление символов, вставку символов, замену символов и т.д. Основные методы класса StringBuilder включают Append, Insert, Remove, Replace и т.д.
7. Основная разница между классами System.String и System.Text.StringBuilder заключается в том, что объекты типа StringBuilder являются изменяемыми, то есть после создания и заполнения объекта его можно изменять, в то время как объекты типа String являются неизменяемыми, то есть после создания и заполнения строки ее нельзя изменить.
8. Регулярные выражения - это последовательности символов, которые используются для поиска и обработки текстовых данных. Регулярные выражения применяются для поиска, замены, валидации и разбора текста.
9. Задание регулярного выражения - это определение шаблона для поиска или замены строковых данных в тексте. Регулярные выражения используются для поиска подстрок, проверки строк на соответствие определенному формату, извлечения информации из текстовых данных и многого другого.

Поиск подстрок с помощью регулярного выражения осуществляется с помощью метода **Regex.Match** или **Regex.Matches**, которые возвращают объекты типа **Match** или **MatchCollection** соответственно.

1. Класс **Regex** является основным классом для работы с регулярными выражениями в C#. Он предоставляет методы для компиляции, поиска, замены и разбора регулярных выражений.

Некоторые из наиболее часто используемых методов класса **Regex** включают:

**Regex.Match** - ищет первое вхождение регулярного выражения в строке и возвращает объект типа **Match**.

**Regex.Matches** - находит все вхождения регулярного выражения в строке и возвращает объект типа **MatchCollection**.

**Regex.Replace** - заменяет все вхождения регулярного выражения в строке заданным значением.

**Regex.Split** - разбивает строку на подстроки, используя регулярное выражение в качестве разделителя.

1. Поиск первого вхождения образца в текст осуществляется с помощью метода **Regex.Match**, который принимает на вход строку для поиска и регулярное выражение в виде строки. Метод возвращает объект типа **Match**, который содержит информацию о первом вхождении образца в строку, такую как позиция начала и конца вхождения и само вхождение.
2. Поиск всех вхождений образца в текст осуществляется с помощью метода **Regex.Matches**, который работает аналогично методу **Regex.Match**, но возвращает объект типа **MatchCollection**, который содержит информацию о всех вхождениях образца в строку.
3. Замена образца в тексте осуществляется с помощью метода **Regex.Replace**, который принимает на вход строку для замены, регулярное выражение в виде строки и значение для замены. Метод заменяет все вхождения образца в строке заданным значением.
4. Разбор текста на лексемы (токены) – это процесс анализа текста и его разделения на отдельные элементы, называемые лексемами или токенами. Каждая лексема представляет собой последовательность символов с определенным значением или типом.

Для разбора текста на лексемы в C# можно использовать классы из пространства имен System.Text.RegularExpressions. Одним из основных классов, используемых для разбора текста на лексемы, является класс Match. Он представляет собой результат сопоставления регулярного выражения с заданной строкой.

1. Класс Match имеет ряд свойств, которые предоставляют информацию о найденной лексеме. Некоторые из наиболее часто используемых свойств:

Value – возвращает текст, соответствующий найденной лексеме.

Index – возвращает индекс первого символа найденной лексемы в исходной строке.

Length – возвращает длину найденной лексемы.

1. Класс **Match** представляет результат сопоставления регулярного выражения с заданной строкой. Этот класс содержит информацию о совпадении, включая совпадающий фрагмент строки, начальную и конечную позиции совпадения и другую полезную информацию.

**MatchCollection** представляет коллекцию объектов **Match**, которые соответствуют совпадениям, найденным в заданной строке. Эта коллекция может быть использована, чтобы получить все совпадения или перебрать их по одному.

Некоторые свойства класса **Match**:

**Success**: определяет, было ли найдено совпадение;

**Index**: начальная позиция совпадения в строке;

**Length**: длина совпадения;

**Value**: строковое представление совпадения;

**Groups**: коллекция объектов **Group**, каждый из которых представляет захватывающую группу в регулярном выражении.

Некоторые свойства класса **MatchCollection**:

**Count**: количество совпадений в коллекции;

**IsReadOnly**: указывает, доступна ли коллекция только для чтения;

**Item**: получает совпадение по индексу в коллекции.