1. Отличия между классом и структурой:

- Способ хранения в памяти: Классы являются ссылочными типами и хранятся в управляемой куче (heap), тогда как структуры - значимые типы и хранятся в стеке или вложены в другие объекты.

- Наследование: Классы поддерживают наследование, структуры этого не поддерживают.

- Присвоение значений по умолчанию: Классы неявно имеют конструктор без параметров, который инициализирует объект значениями по умолчанию. Структуры требуют явного определения конструктора без параметров для этого.

- Производительность: В общем случае, структуры могут быть более производительными, так как они хранятся в стеке или внутри других объектов, что может уменьшить накладные расходы.

2. В структуре могут быть:

- Поля (fields)

- Свойства (properties)

- Методы (methods)

- Конструкторы (constructors)

- Индексаторы (indexers)

- События (events)

В структуре не могут быть:

- Наследование от других структур или классов (кроме наследования от `System.ValueType`, от которого наследуют все структуры по умолчанию).

- Виртуальные методы (но они могут реализовывать интерфейсы).

3. Перечисление (enum) в C# - это набор именованных целочисленных констант. Пример:

```csharp

enum Days { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };

// Использование перечисления

Days today = Days.Wednesday;

```

4. Стандартные интерфейсы .NET включают в себя:

- `IEnumerable`: Предоставляет перечислитель, поддерживающий простое перечисление элементов коллекции.

- `IComparable`: Определяет метод для сравнения объекта с другим объектом того же типа.

- `IEquatable`: Определяет метод для определения равенства объекта с другим объектом того же типа.

- `IDisposable`: Предоставляет метод для освобождения ресурсов.

- и другие.

5. Интерфейс `IComparable` используется для сравнения объектов. Реализуется методом `CompareTo`, который возвращает отрицательное число, если текущий объект меньше переданного, положительное - если больше, и ноль - если они равны.

6. Интерфейс `ICloneable` используется для создания копии объекта. Включает метод `Clone()`, который возвращает копию объекта.

7. Полиморфизм - это возможность объекта использовать методы с одинаковыми именами, но реализованными в разных классах. Формы полиморфизма:

- Перегрузка методов (Compile-time Polymorphism): Одно имя метода, но различные параметры.

- Переопределение методов (Runtime Polymorphism): Методы с одинаковыми именами и параметрами, но реализованные в разных классах (через наследование). Пример:

```csharp

class A

{

public virtual void Display()

{

Console.WriteLine("A");

}

}

class B : A

{

public override void Display()

{

Console.WriteLine("B");

}

}

A obj = new B();

obj.Display(); // Выведет "B"

```

8. Виртуальные методы в классе позволяют им быть переопределенными в производных классах. Это является основой для реализации полиморфизма через переопределение методов в наследуемых классах.

9. Запрет переопределения методов в C# достигается с помощью ключевого слова `sealed`. Например:

```csharp

class A

{

public sealed void Display()

{

Console.WriteLine("A");

}

}

class B : A

{

// Нельзя переопределить метод Display в классе B

}

```