1)Чем класс отличается от структуры?

Структуры синтаксически очень похожи на классы, но существует принципиальное отличие, которое заключается в том, что **класс – является ссылочным** типом (reference type), а **структуры – значимый класс** (value type). А следовательно, *классы* всегда создаются в, так называемой, “**куче”** (heap), а структуры создаются в стеке (**stack**).

2)Что может и чего не может быть в структуре?

Как и у классов, у каждой структуры имеются свои члены: методы, поля, индексаторы, свойства, операторные методы и события. В структурах допускается также определять конструкторы, **но не деструкторы**. В то же время для структуры **нельзя определить конструктор, используемый по умолчанию (т.е. конструктор без параметров).** Дело в том, что конструктор, вызываемый по умолчанию, определяется для всех структур автоматически и не подлежит изменению. Такой конструктор инициализирует поля структуры значениями, задаваемыми по умолчанию. А поскольку структуры не поддерживают наследование, то их члены нельзя указывать как abstract, virtual или protected.

3)Что такое перечисление? Приведите пример определения и использования перечисления

Перечисление (или enum) - это тип данных значения в C#. Он в основном используется для присвоения имен или строковых значений константам, которые облегчают чтение и обслуживание программы.

Синтаксис: enum nameOfEnum {str1, str2…} (где str1 == 0, str2 ==1…)

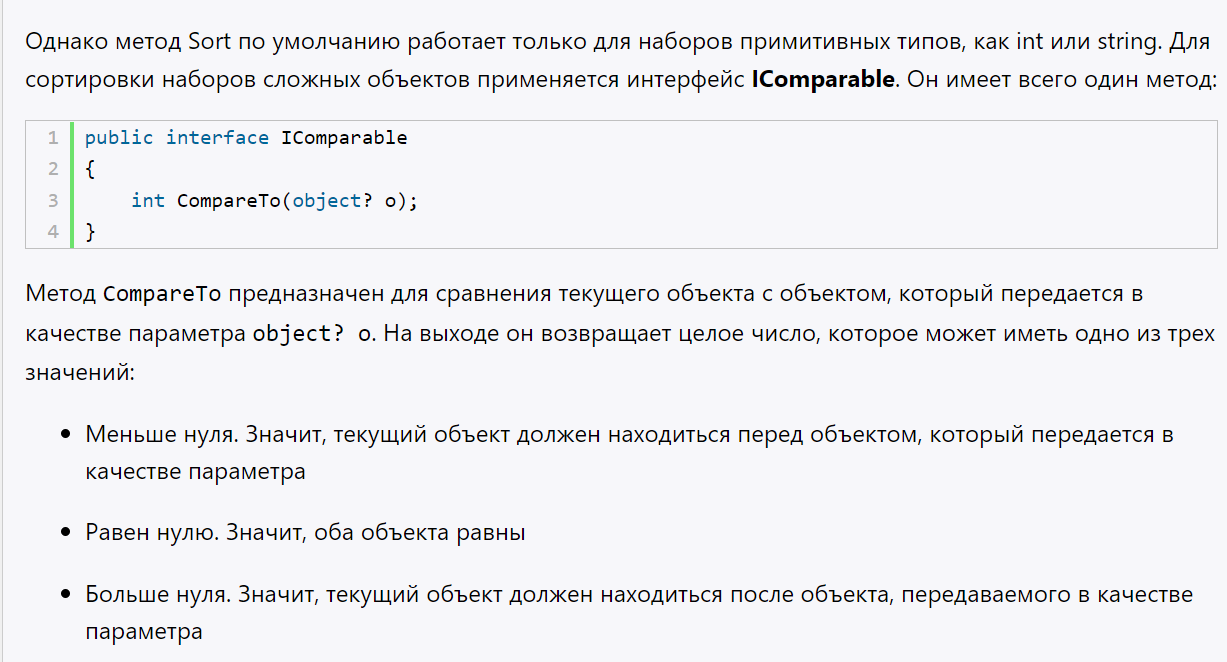
4)Перечислите и поясните стандартные интерфейсы .Net?

В библиотеке .NET определено множество стандартных интерфейсов, задающих желаемое поведение объектов. Например, интерфейс **IComparable** задает метод сравнения объектов по принципу больше или меньше, что позволяет выполнять их сортировку. Реализация интерфейсов **IEnumerable** и IEnumerator дает возможность просматривать содержимое объекта с помощью конструкции foreach, а реализация интерфейса **ICloneable** – клонировать объекты.

Стандартные интерфейсы поддерживаются многими стандартными классами библиотеки. Например, работа с массивами с помощью цикла foreach возможна именно потому, что тип Array реализует интерфейсы IEnumerable и IEnumerator. Можно создавать и собственные классы, поддерживающие стандартные интерфейсы, что позволит использовать объекты этих классов стандартными способами.

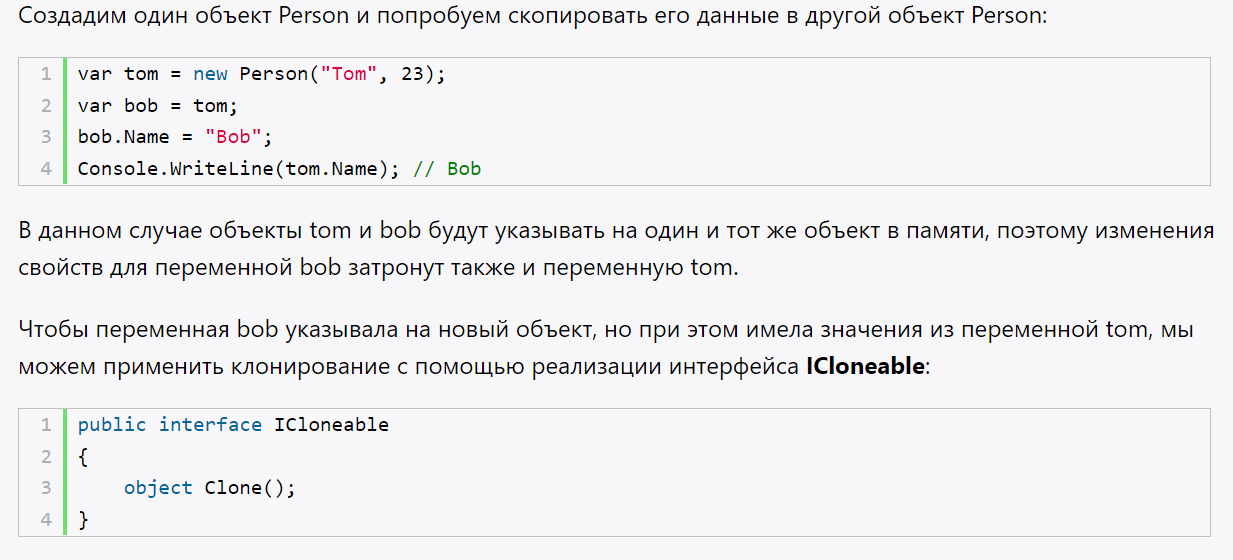
5)Как используется интерфейс IComparable?

Интерфейс IComparable задает метод сравнения объектов по принципу больше или меньше, что позволяет выполнять их сортировку.



6)Как используется интерфейс ICloneable?

Клонирует объекты

7)Что такое полиморфизм? Перечислите его формы. Приведите примеры.

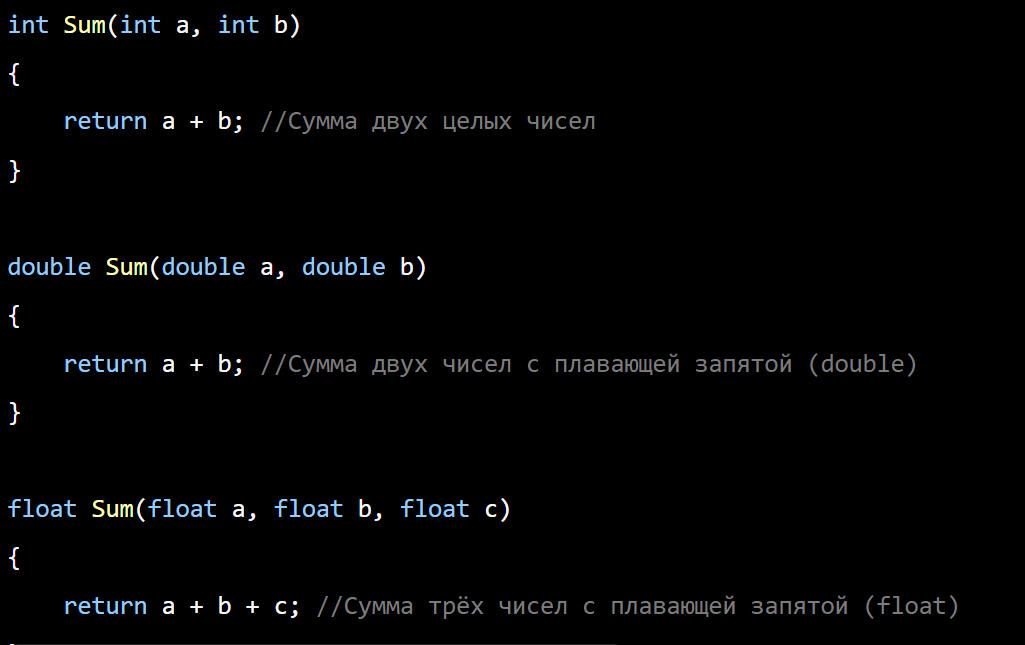
Полиморфизм (от греч. poly — много и morphe — форма) — один из главных столпов объектно-ориентированного программирования. Его суть заключается в том, что один фрагмент кода может работать с разными типами данных. В C# это реализуется с помощью перегрузок (**overloading**).

Бенджамин Пирс в “[Типы в языках программирования](https://www.ozon.ru/context/detail/id/7410082/)”:

*Термин “полиморфизм” обозначает семейство различных механизмов, позволяющих использовать один и тот же участок программы с различными типами в различных контекстах.*

Формы палиморфизма могут быть различные: от определения аргументов в функциях для разных типов данных до объектов, которые могут менять значение текущего типа на значение другого типа. Или даже переменные, которые меняют свое значение на значение другого типа.

Примеры:



8)Зачем в классе определяют виртуальные методы?

Виртуальные методы базового класса определяют интерфейс всей иерархии классов. Это значит, что в любом производном классе, который не является прямым наследником от базового класса, можно переопределить виртуальные методы.

Возможность переопределять методы в классах наследниках.

9)Как сделать запрет переопределения методов?

Также можно запретить переопределение методов и свойств. В этом случае их надо объявлять с модификатором **sealed**.