3. В C ++ конструкторы и деструкторы не наследуются. Однако они вызываются, когда дочерний класс инициализирует свой объект. Конструкторы вызываются один за другим иерархически, начиная с базового класса и заканчивая последним производным классом. Деструкторы вызываются в обратном порядке.

7. **Указателю** **на** **базовый** **класс** **можно** **присвоить** значение **адреса** **объекта** любого **производного** **класса**, например: monster \*p; // Описывается **указатель** **на** **базовый** **класс** p = new daemon; /\* **Указатель** ссылается на **объект** **производного** **класса** \*/. Вызов методов **объекта** происходит в соответствии с типом **указателя**, а не фактическим типом **объекта**, на который он ссылается, поэтому при выполнении оператора, например.

Вообще говоря, переменная ссылки на объект может ссылаться только на объект своего типа.

Но из этого принципа строгого соблюдения типов в C# имеется одно важное исключение: переменной ссылки на объект базового класса может быть присвоена ссылка на объект любого производного от него класса.

8. Итак, присваивать объекту базового класса *объект* производного класса можно, но вызываются для него только методы и свойства, определенные в базовом классе. Иными словами, возможность доступа к элементам класса определяется типом ссылки, а не типом объекта, на который она указывает.

Полиморфизм — одна из трех основных парадигм ООП. Если говорить кратко, полиморфизм — это способность обьекта использовать методы производного класса, который не существует на момент создания базового.

**Наследование** – это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым или родительским. Новый класс – потомком, наследником или производным классом.