**В C++ время жизни объекта зависит от того, как он был создан и управлялся. Объекты, созданные с автоматическим сроком хранения (например, локальные переменные в функциях), имеют срок службы, который заканчивается при завершении блока, содержащего их объявление. Объекты, созданные с динамическим сроком хранения (с использованием оператора 'new'), будут продолжать существовать до тех пор, пока они не будут явно уничтожены с помощью оператора 'delete' или программа не завершится.**

**В C# временем жизни объектов управляет сборщик мусора, который периодически идентифицирует и удаляет объекты, которые больше не используются программой. Точное время уничтожения объекта не является детерминированным и может варьироваться в зависимости от конкретной реализации используемого сборщика мусора. Однако разработчики могут влиять на поведение сборщика мусора, используя такие методы, как доработка объекта и слабые ссылки.**

**a) Неявное копирование происходит при передаче объекта, как параметр функции по значению и возвращении объекта функцией.**

**b) Изначально создаются родительские объекты, объекты, которые присваиваются полям объекта-агрегата, а затем создаётся сам агрегат.**

**c) Интерфейс IDisposable предназначен для реализации метода Dispose, который высвобождает занимаемые объектом управляемые и неуправляемые ресурсы.**

**d) Dispose будет обязательно вызван для объекта, созданного в блоке using.**

**e) Нужно явно вызвать конструктор родительского объекта в конструкторе дочернего (MyChildClass(int x, int y) : MyParentClass(x, y) { … }).**

**f) В C# вызов конструктора предка происходит через команду base.**

**(MyChildClass(int x, int y) : base(x, y) { … }).**

**g) Нужно вызвать конструктор члена класса с параметрами в конструкторе класса-агрегата**

**h) Термин "ссылка" используется как в C++, так и в .NET, это понятие имеет разные значения в двух контекстах. В C++ ссылка - это тип переменной, в то время как в .NET ссылка - это способ ссылки на объект в управляемой куче.**

**i) template <class T> class MyNewTemplate: { … } — определение шаблона.MyTemplate<MyClass> template; — объявление переменной типа шаблона.**