**1. Перечислите свойства внутренних и вложенных классов?**

Если вложенный класс объявлен как не private-класс, то создание экземпляра этого класса имеет следующий вид:

Outer.Inner objInner = new Outer.Inner(); где

Outer – имя внешнего класса, в котором объявляется вложенный класс Inner;

Inner – имя вложенного класса, который объявлен в пределах фигурных скобок класса Outer;

objInner – имя объекта (экземпляра) класса, который создается;

Outer.Inner() – имя конструктора по умолчанию, который вызывается для создания объекта класса. Если в классе Inner реализованы другие параметризированные конструкторы, то этот вызов может быть другим.

Вложенные классы обладают некоторыми специальными возможностями, которые удобны, когда нужен вспомогательный класс, работающий внутри содержащего его класса.

Например, контейнерный класс может содержать коллекцию объектов. Предположим, что требуется некоторое средство для выполнения итерации по всем содержащимся объектам, чтобы позволить внешним пользователям, выполняющим итерацию, поддерживать маркер, или некую разновидность курсора, который запоминает свое текущее место во время итерации.

Это распространенный подход в проектировании. Избавление пользователей от необходимости хранить прямые ссылки на содержащиеся в коллекции объекты обеспечивает большую гибкость в отношении изменения внутреннего поведения контейнерного класса без разрушения кода, использующего этот контейнерный класс.

Вложенные классы по нескольким причинам предоставляют отличное решение такой проблемы.

Вложенные классы имеют доступ ко всем членам, видимым содержащему их классу, даже если эти члены являются приватными.

Внутренний класс ведет себя как обычный класс за тем исключением, что его объекты могут быть созданы только внутри внешнего класса.

Внутренний класс имеет доступ ко всем полям внешнего класса, в том числе закрытым с помощью модификатора private. Аналогично внешний класс имеет доступ ко всем членам внутреннего класса, в том числе к полям и методам с модификатором private.

Объекты внутренних классов могут быть созданы только в том классе, в котором внутренние классы опеределены. В других внешних классах объекты внутреннего класса создать нельзя.

Существует четыре категории вложенных классов:

Статические вложенные классы и не статические вложенные классы. Вложенные классы, объявленные статически, называются вложенными статическими классами.

Внутренние классы — когда объект внутреннего класса связан с объектом обрамляющего класса. Не статические вложенные классы называются внутренними классами, если они связанны с внешним классом.

Локальные классы — объявленные внутри блока кода и не являющиеся членом обрамляющего класса. В этом случае можно рассматривать класс как локальную переменную типа класс.

Анонимные классы – наследуемые, от какого либо класса, классы в которых при объявлении не задано имя класса.

Причины использования вложенных классов такие. Если класс полезен только для одного другого класса, то вполне логично встроить его в этот класс и хранить их вместе. Использование вложенных классов увеличивает инкапсуляцию.

**2. Что такое статический класс и какие у него свойства?**

Класс считается статическим, если при его создании, перед ключевым словом class указывается ключевое слово static. Определение класса статическим, означает то, что этот класс является набором только статических сущностей (полей, методов, свойств и т.п.), и, внимание, создавать объекты такого класс нельзя!

Кроме этого, на статический класс, накладываются как минимум следующие ограничения:

статический класс не может служить базовым для других классов, а так же, сам не может быть наследником какого-либо произвольного класса;

статический класс должен содержать только статические члены.

статический класс не может содержать сущности с атрибутами доступа protected и protected internal (да они и не нужны, все равно наследование запрещено);

класс не может иметь перегруженных операций.

**3. Каково назначение перегрузки операторов?**

Перегрузка операторов заключается в определении в классе, для объектов которого мы хотим определить оператор, специального метода

(Способ объявления новых операторов для типа)

**4. Как используется ключевое слово operator?**

Определяет операторный метод, который, в свою очередь, определяет действие оператора относительно своего класса

Затем вместо названия метода идет ключевое слово operator и собственно сам оператор. И далее в скобках перечисляются параметры. Бинарные операторы принимают два параметра, унарные - один параметр. И в любом случае один из параметров должен представлять тот тип - класс или структуру, в котором определяется оператор.

**5. Какие операции нельзя перегружать в C#?**

[] (но есть индексатор)

() (можно определить новые операторы преобразования)

► +=, -=, \*=, /=, %=, &=, |=, ^=, <<=, >>= (но

получаем автоматически в случае перегрузки бинарной операции)

&&, ||

►=, ., ?:, ??, ->, =>, f(x), as, checked,

unchecked, default, delegate, is, new, sizeof, typeof

**6. Можно ли перегрузкой отменить очередность выполнения операции?**

да?????

перегрузка операторов не изменяет порядок выполнения операций и их приоритет;

**7. Истинно ли следующее утверждение: операция >= может быть перегружена.**

да

**8. Сколько аргументов требуется для определения перегруженной унарной операции?**

1

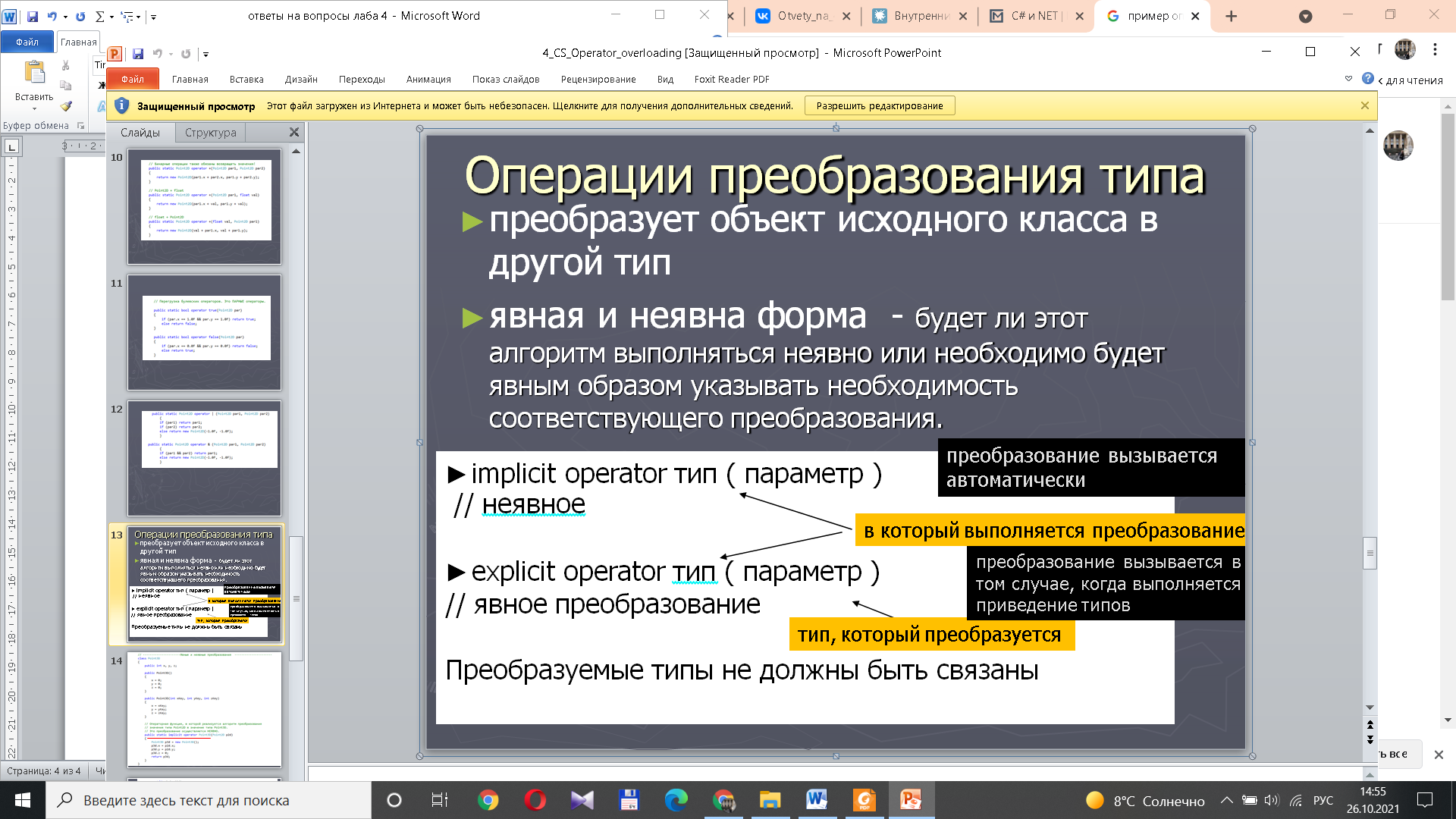
**9. Можно ли перегружать операцию []?**

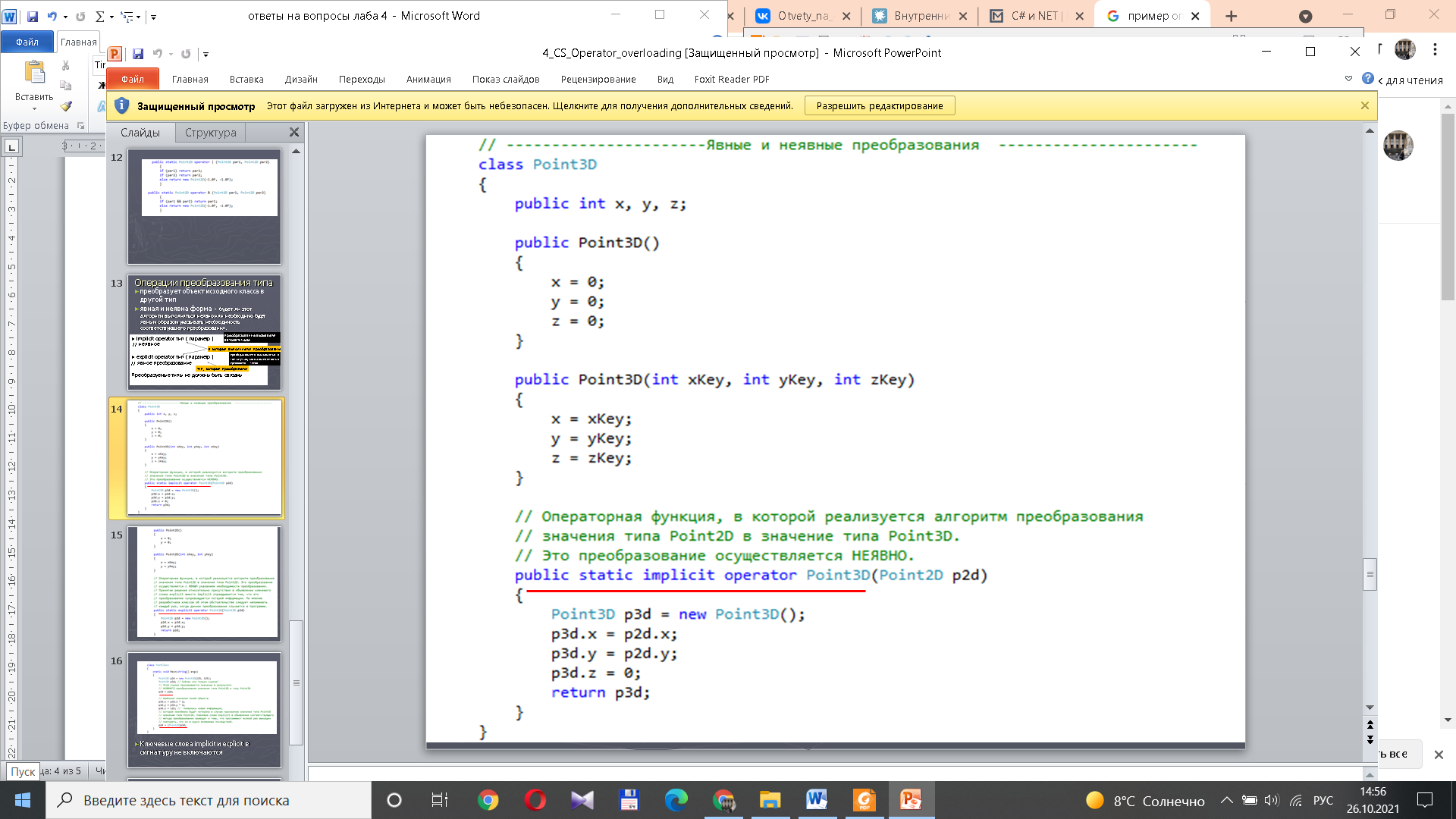
нет, но есть индексатор

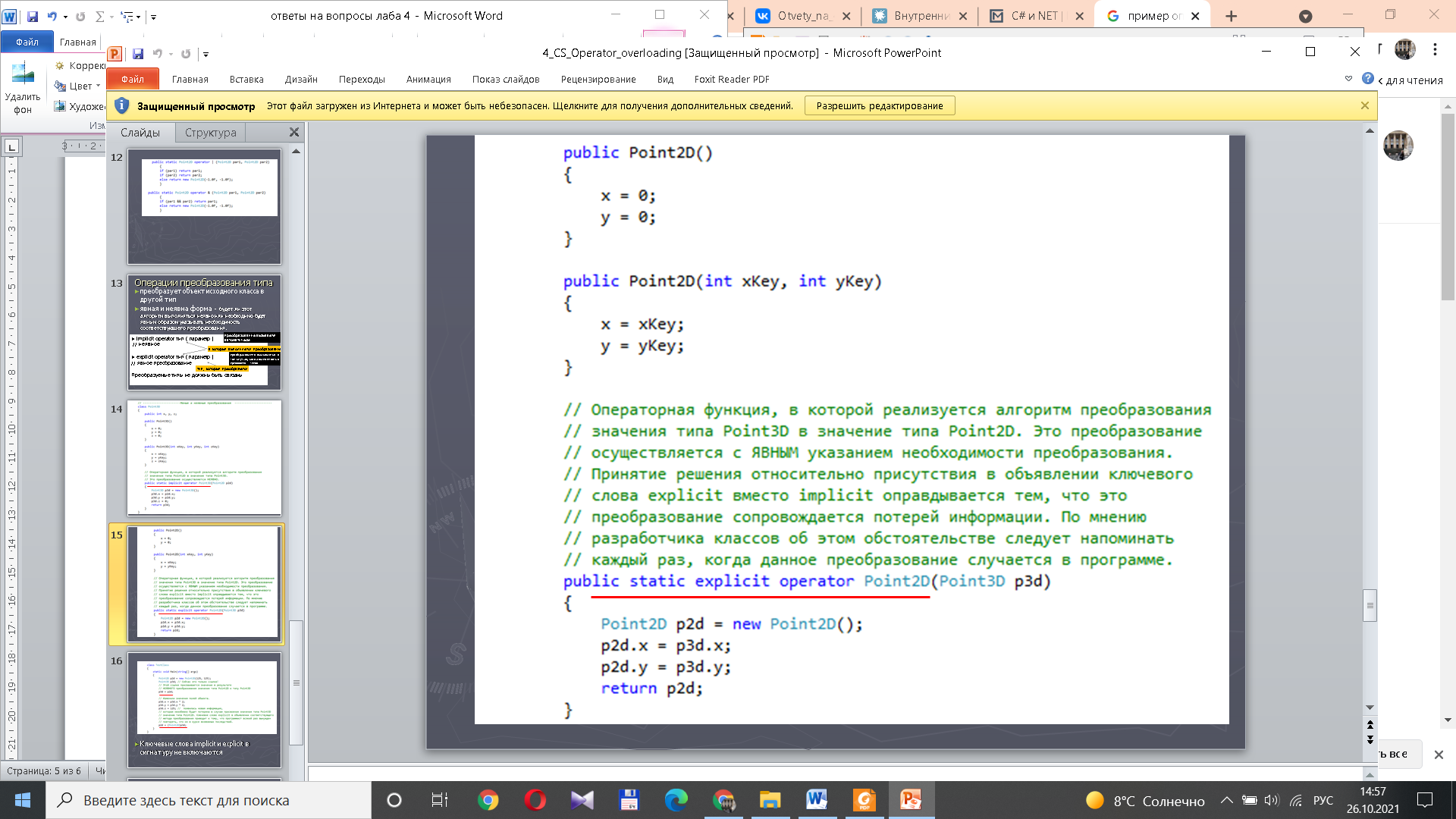
**10.Можно ли перегружать операцию ->?**

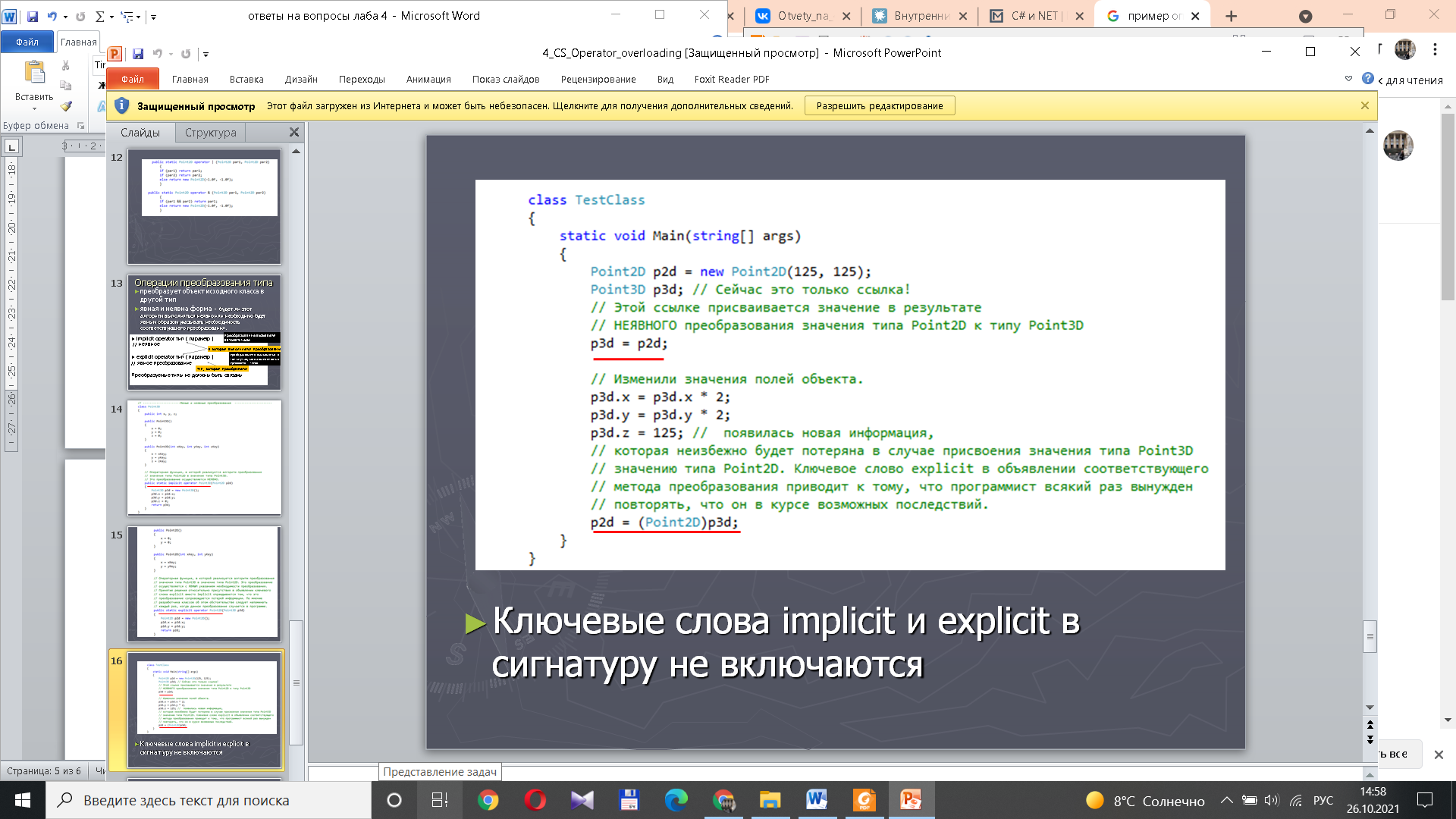
нет

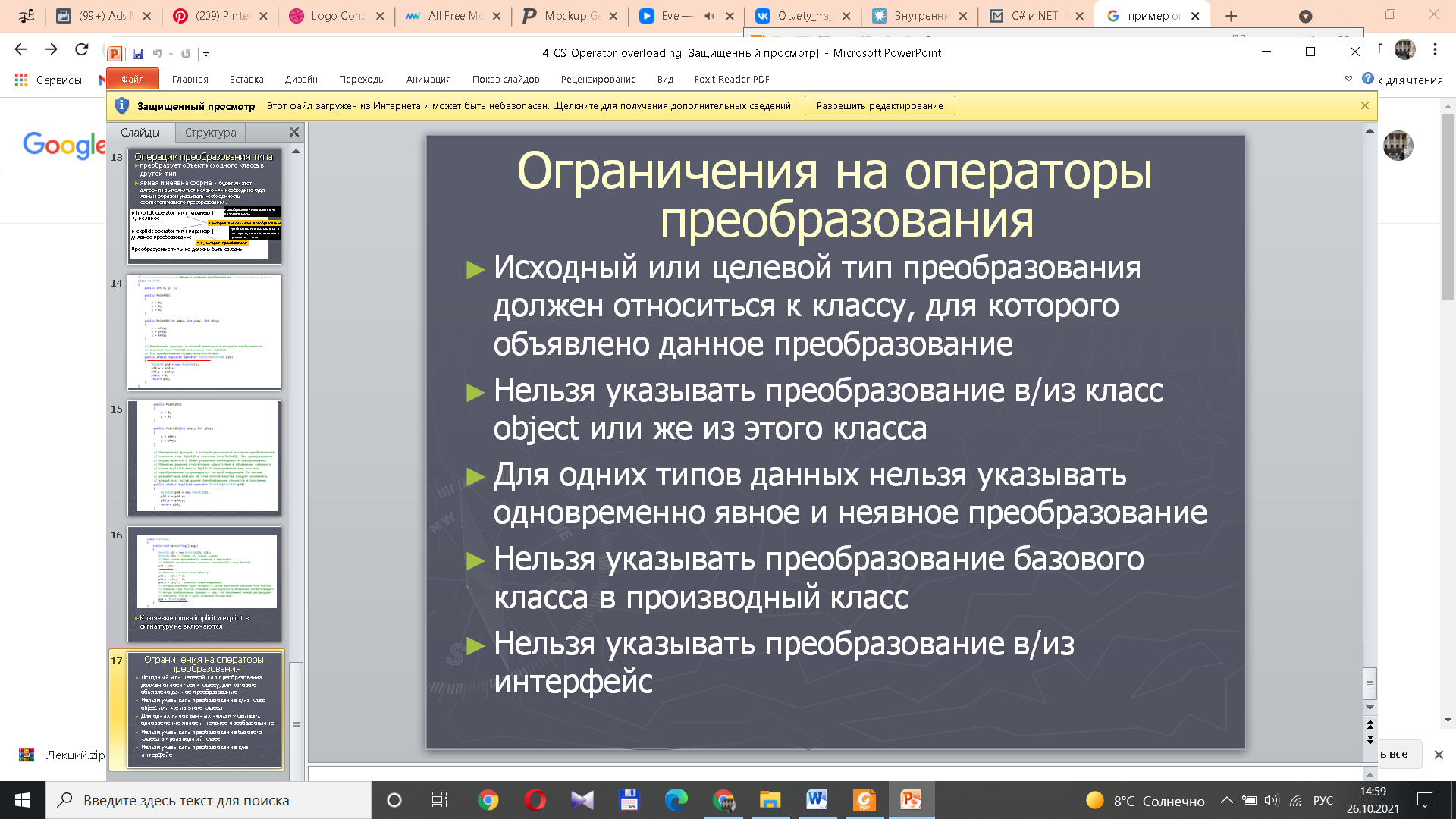
**11.Приведите пример оператора приведения типа**











**12.Что такое метод расширения? Как и где его можно использовать?**

Методы расширения (extension methods) позволяют добавлять новые методы в уже существующие типы без создания нового производного класса.

Эта функциональность бывает особенно полезна, когда нам хочется добавить в некоторый тип новый метод, но сам тип (класс или структуру) мы изменить не можем, поскольку у нас нет доступа к исходному коду. Либо если мы не можем использовать стандартный механизм наследования, например, если классы определенны с модификатором sealed.

Для того, чтобы создать метод расширения, вначале надо создать статический класс, который и будет содержать этот метод. Затем объявляем статический метод. Суть нашего метода расширения - подсчет количества определенных символов в строке.

Собственно метод расширения - это обычный статический метод, который в качестве первого параметра всегда принимает такую конструкцию: this имя\_типа название\_параметра. Так как наш метод будет относиться к типу string, то мы и используем данный тип.

Затем у всех строк мы можем вызвать данный метод: int i = s.CharCount(c);. Причем нам уже не надо указывать первый параметр. Значения для остальных параметров передаются в обычном порядке.

**13.Пусть дан фрагмент кода определения оператора преобразования типа.**

**Определить форму преобразования.**

**public static implicit operator Point2D(Point3D a)**

**{/\* код\*/;}**

неявное(смотреть пример выше)

**14.Выберите верное утверждение. Метод расширения может:**

1) получать доступ к public членам расширяемого класса

2) получать доступ к protected членам расширяемого класса

3) получать доступ к internal членам расширяемого класса

4) быть объявлен в любом классе

5) быть без параметров

**15.Выберите из списка неверное правило перегрузки операторов для C#.**

1) префиксные операции ++ и – – перегружаются парами

2) операции сравнения перегружаются парами: == и != ; < и >;<= и >=

3) перегруженные операции обязаны возвращать значения

4) должны объявляться как protected

5) true и false можно перегружать