1. Перечислите свойства внутренних и вложенных классов?

**Внутренний класс** — это класс с модификатором доступа internal, который определяет доступность класса только внутри той же сборки. Класс с модификатором internal доступен только для классов и методов внутри той же самой сборки (например, одного проекта).

**Вложенный класс** — это класс, который определен внутри другого класса. Вложенный класс существует в рамках внешнего класса и может быть использован для представления логически связанной функциональности.  Вложенный класс имеет доступ к полям и методам внешнего класса, даже если они имеют модификатор доступа private.

1. Что такое статический класс и какие у него свойства?

Статические классы предназначены для хранения статических членов и не могут быть инстанцированы. Их используют для группировки методов и свойств, которые не требуют создания экземпляра класса.

прямой потомок System.Object

экземпляры такого класса создавать запрещено

не должен реализовывать никаких интерфейсов (не вызвать)

нельзя использовать в качестве поля, параметра метода или локальной переменной

от него запрещено наследовать

все элементы такого класса должны явным образом объявляться с модификатором static

может иметь статический конструктор

Компилятор не создает автоматически конструктор по умолчанию

1. Каково назначение перегрузки операторов?

Перегрузка операторов в C# позволяет определить, как операторы (например, +, -, ==, >) должны вести себя при работе с пользовательскими типами.

1. Как используется ключевое слово operator?

Ключевое слово operator используется для определения перегруженного оператора. Оно указывает, что метод описывает, как конкретный оператор должен вести себя для типа.

public static ReturnType operator OperatorSymbol(Type parameter)

{

// Логика оператора

}

1. Какие операции нельзя перегружать в C#?

В C# нельзя перегружать следующие операции:

* **Условные логические операторы**: &&, || (логическое И и ИЛИ).
* **Операторы присваивания**: =, +=, -=, и т. д.
* **Оператор typeof**: для получения типа.
* **Оператор sizeof**: для получения размера типа.
* **Оператор .** (точка): доступ к членам класса или структуры.
* **Оператор ?:** (тернарный оператор).
* **Оператор ??** (null-объединение).
* **Оператор =>** (лямбда-выражение).

1. Можно ли перегрузкой отменить очередность выполнения операции?

Нет, **перегрузка не может изменить приоритет или ассоциативность операторов**. Порядок выполнения операторов в выражениях остается таким же, каким он установлен в языке C#. Перегрузка изменяет только реализацию операции для пользовательского типа, но не её очередность выполнения.

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

Да, это утверждение **истинно**. Операция >= может быть перегружена. Однако в C# при перегрузке оператора >= также требуется перегрузить оператор <= для того же типа.

1. Сколько аргументов требуется для определения перегруженной унарной операции?

Перегрузка унарных операторов требует указания **одного параметра**, который представляет собой операнд, для которого оператор применяется. В методе перегрузки оператор принимает один параметр — это объект, к которому применяется операция.

1. Можно ли перегружать операцию []?

В C# **операцию индексирования** нельзя перегружать в традиционном смысле, как это делается для других операторов (например, +, -, \* и т.д.). Однако, вы можете **создавать индексаторы** в своих классах или структурах, что позволит вам использовать синтаксис [] для доступа к элементам этого класса или структуры.

1. Можно ли перегружать операцию ->?

Нет, **операцию -> в C# перегружать нельзя**, так как она в этом языке не используется. Этот оператор встречается в C++ для доступа к членам объекта через указатель. В C# для обращения к членам класса и структуры применяется оператор . (точка).

1. Приведите пример оператора приведения типа

**Оператор приведения** — это специальный метод, который позволяет преобразовать значение из одного типа в другой. В C# можно создавать явные (explicit) и неявные (implicit) операторы приведения.

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

**Метод расширения** — это специальный статический метод, который позволяет добавлять новые функции к существующим типам (классам, структурам) без изменения их кода или создания производных классов. Это особенно полезно для добавления функциональности к типам, которые вы не можете изменять, например, встроенным типам или типам из сторонних библиотек.

13. Пусть дан фрагмент кода определения оператора преобразования типа. Определить форму преобразования.

public static implicit operator Point2D(Point3D a)

{/\* код\*/;}

В этом коде определено **неявное (implicit) преобразование** типа Point3D в тип Point2D. Использование implicit позволяет преобразовать Point3D в Point2D автоматически, без явного приведения

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

 **Получать доступ к public членам расширяемого класса**  
**Верно.** Методы расширения могут получить доступ к всем публичным членам класса, который они расширяют.

 **Получать доступ к protected членам расширяемого класса**  
**Неверно.** Методы расширения не могут получить доступ к protected членам, поскольку они не являются частью иерархии наследования.

 **Получать доступ к internal членам расширяемого класса**  
**Неверно.** Методы расширения не могут получить доступ к internal членам, если они не находятся в одной сборке с классом.

 **Быть объявлен в любом классе**  
**Верно.** Методы расширения могут быть объявлены в любом статическом классе.

 **Быть без параметров**  
**Неверно.** Методы расширения должны принимать как минимум один параметр, помеченный ключевым словом this, который определяет тип, который вы расширяете.

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

 **Префиксные операции ++ и – – перегружаются парами**  
**Верно.** Если вы перегружаете префиксный оператор ++, вы должны перегрузить и префиксный оператор --.

 **Операции сравнения перегружаются парами: == и !=; < и >; <= и >=**  
**Верно.** Перегрузка операций сравнения должна следовать парной логике.

 **Перегруженные операции обязаны возвращать значения**  
**Верно.** Операторы должны возвращать значение, например, логическое или числовое.

 **Должны объявляться как protected**  
**Неверно.** Операторы перегрузки могут быть объявлены как public, protected или private, в зависимости от нужд класса. Это утверждение является неверным, потому что нет обязательства использовать уровень доступа protected.

 **True и false можно перегружать**  
**Верно.** В C# вы можете перегружать операторы true и false для определения, является ли объект истинным или ложным.