[5. Какие операции нельзя перегружать в C#?](#_5._Какие_операции)

[6. Можно ли перегрузкой отменить очередность выполнения операции?](#_6._Можно_ли)

[9. Можно ли перегружать операцию []?](#_9._Можно_ли)

[10.Можно ли перегружать операцию ->?](#_10._Можно_ли)

[12.Что такое метод расширения? Как и где его можно использовать?](#_12._Что_такое)

1. Перечислите свойства внутренних и вложенных классов?

Внутренний (Inner) класс:

1. Имеет доступ к закрытым членам внешнего класса.

2. Может быть создан внутри другого класса.

3. Обладает доступом ко всем членам внешнего класса.

Вложенный (Nested) класс:

1. Создается внутри другого класса.

2. Может быть закрытым (private), защищенным (protected), внутренним (internal) или открытым (public).

3. Имеет доступ ко всем членам внешнего класса, даже если они закрыты.

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

Статический (Static) класс:

1. Не может быть создан экземпляр класса.

2. Все члены класса должны быть статическими.

3. Используется для группировки связанных методов и свойств.

4. Методы и свойства вызываются через имя класса, без создания объекта.

5. Невозможно унаследовать от статического класса.

3. Каково назначение перегрузки операторов?

Назначение перегрузки операторов:

1. Позволяет определить поведение операторов для пользовательских типов данных.

2. Обеспечивает более удобный синтаксис для операций с пользовательскими типами.

3. Позволяет создавать более интуитивный и читаемый код.

### 4. Как используется ключевое слово `operator`?

Использование ключевого слова `operator`:

1. Определение перегрузки оператора происходит с использованием ключевого слова `operator` в теле класса.

2. Операторы, которые могут быть перегружены, включают `+`, `-`, `\*`, `/`, `==`, `!=`, `<`, `>`, `<=`, `>=`, и т. д.

# 5. Какие операции нельзя перегружать в C#?

Операции, которые нельзя перегружать в C#:

1. Присваивание (`=`)

2. Условный (тернарный) оператор (`? :`)

3. Вызов метода (`()`)

4. Член доступа (`.`)

5. Метод доступа к элементу массива (`[]`)

# 6. Можно ли перегрузкой отменить очередность выполнения операции?

Нет, перегрузка оператора не позволяет отменить очередность выполнения операции. Операции выполняются в том порядке, который определен языком программирования.

### 7. Истинно ли следующее утверждение: операция `>=` может быть перегружена?

Да, операция `>=` может быть перегружена.

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

Для определения перегруженной унарной операции требуется только один аргумент.

# 9. Можно ли перегружать операцию `[]`?

Да, операцию `[]` можно перегружать. Это позволяет пользовательским типам предоставлять доступ к своим элементам, аналогично массивам.

# 10. Можно ли перегружать операцию `->`?

Нет, в C# нельзя перегружать операцию `->`. Она используется для доступа к членам через указатель в некоторых языках, но в C# такой возможности нет.

### 11. Приведите пример оператора приведения типа.

```csharp

public class Temperature

{

private double degreesCelsius;

public Temperature(double celsius)

{

degreesCelsius = celsius;

}

// Оператор приведения типа

public static explicit operator double(Temperature t)

{

return t.degreesCelsius;

}

}

// Использование

Temperature temp = new Temperature(25.0);

double degrees = (double)temp; // Приведение типа

```

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

Метод расширения (Extension Method):

1. Это статический метод, который добавляется к существующему типу без изменения его кода.

2. Определяется в статическом классе.

3. Принимает объект типа, к которому применяется, в качестве первого параметра, используя ключевое слово `this`.

Пример:

```csharp

public static class StringExtensions

{

public static bool IsNullOrEmpty(this string str)

{

return string.IsNullOrEmpty(str);

}

}

// Использование

string myString = "Hello, World!";

bool result = myString.IsNullOrEmpty();

```

Где использовать:

1. Методы расширения могут использоваться в любом месте после включения пространства имен, в котором определен статический класс с методами расширения.

2. Обеспечивают более удобный и читаемый синтаксис для работы с типами, к которым добавляются методы.

3. Позволяют "расширять" функциональность сторонних библиотек без необходимости изменения их кода.