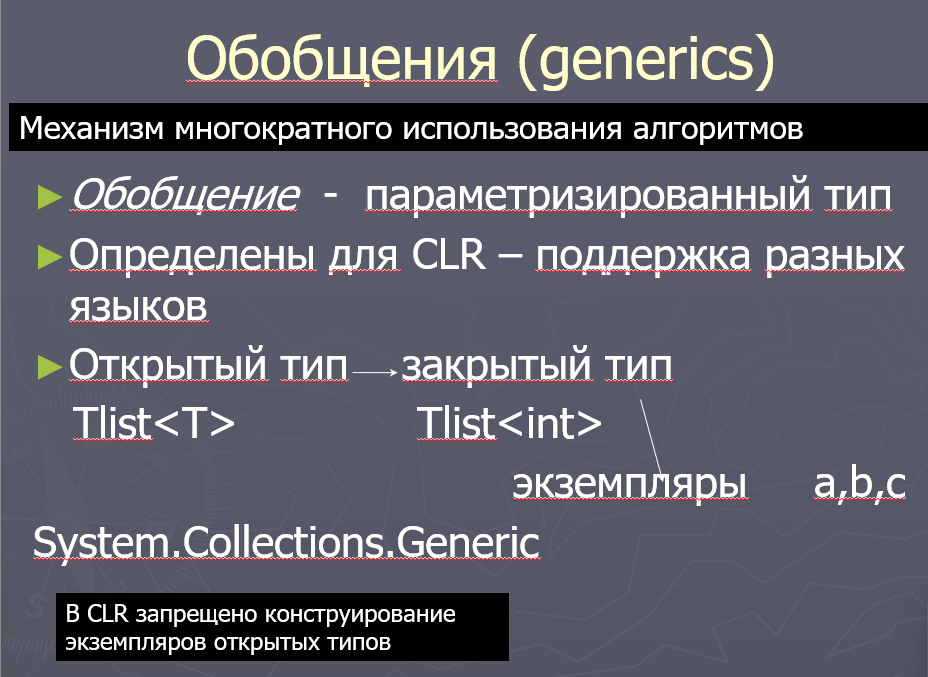
1. Что такое обобщение (generic)?

В C# обобщение (generic) представляет собой механизм языка, который позволяет создавать обобщенные классы, интерфейсы и методы. Обобщенные типы данных позволяют писать код, который может работать с различными типами данных, без необходимости повторного написания кода для каждого конкретного типа.



1. Пусть дан фрагмент листинга. В какой строчке содержится ошибка?

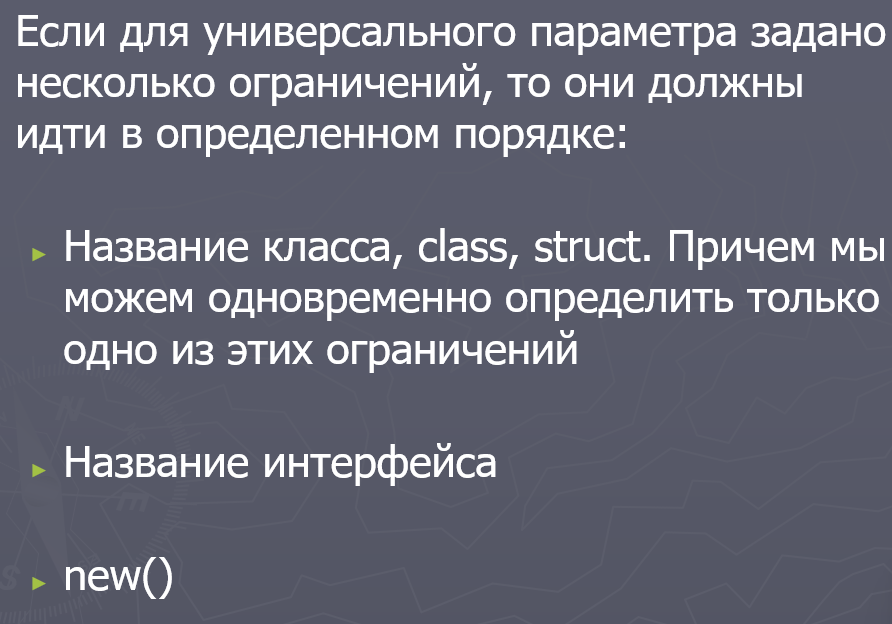
4

1. Как можно наложить определенное ограничение на параметр?

Для наложения определенного ограничения на параметр типа в обобщенном коде в C# можно использовать ключевое слово where (вэа). Ограничения позволяют указать, какие типы данных могут быть использованы в качестве аргументов для параметра типа.

1. Как можно наложить несколько ограничений на параметр?

Чтобы наложить несколько ограничений на параметр типа в C#, вы можете перечислить их, используя запятую.



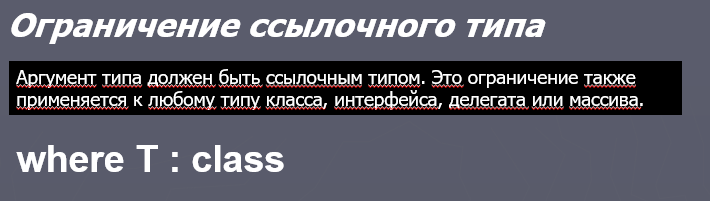
5. Перечислите все существующие ограничения на типы данных

обобщения?

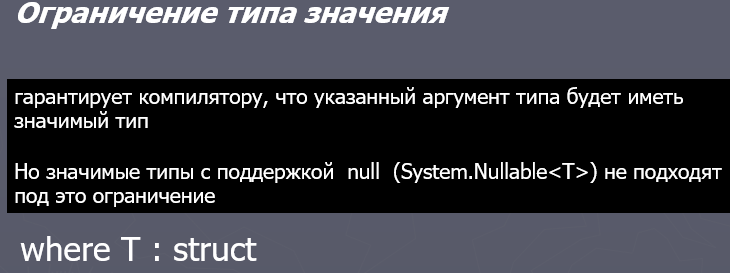
Классы

Интерфейсы

class - универсальный параметр должен представлять класс



struct - универсальный параметр должен представлять структуру



new() - универсальный параметр должен представлять тип, который имеет общедоступный (public) конструктор без параметров

6. Какое ограничение на тип задано в следующем фрагменте листинга?

class A { }class В : A { }

class С { }

class Test<T> where T : A { }

В следующем фрагменте листинга ограничение на тип T задано как where T : A. Это означает, что тип T, используемый в классе Test<T>, должен быть наследником или самим типом A или его производным классом.

7. Какое ограничение на тип задано в следующем фрагменте листинга?

interface A { }

class Test<T> where T : class { }

В следующем фрагменте листинга ограничение на тип T задано как where T : class. Это означает, что тип T, используемый в классе Test<T>, должен быть ссылочным типом данных (классом) или может быть null.

8. Какое ограничение на тип задано в следующем фрагменте листинга?

interface A { }

class Test<T> where T : struct { }

В следующем фрагменте листинга ограничение на тип T задано как where T : struct. Это означает, что тип T, используемый в классе Test<T>, должен быть значимым типом данных (структурой).

9. Приведите примеры, когда обобщенный класс может действовать как

базовый или производный класс.

Когда обобщенный класс выступает в роли контейнера для разных типов – базовый.

Производным если он помогает стилизовать и добавлять методы для конкретного типа данных

10.В каких случаях в обобщениях может использоваться оператор

default?

Инициализация переменных

Установка параметров типа по умолчанию

Условные операторы

**1. Инициализация переменной обобщенного типа**

Когда вы хотите инициализировать переменную обобщенного типа, но не знаете конкретного значения, вы можете использовать default для получения значения по умолчанию для типа:

public class Example<T>

{

public T Value { get; set; }

public Example()

{

Value = default(T); // Инициализируется значением по умолчанию для типа T

}

}

**2. Проверка на значение по умолчанию**

Вы можете использовать оператор default для проверки, является ли переменная обобщенного типа равной значению по умолчанию:

public bool IsDefaultValue(T value)

{

return EqualityComparer<T>.Default.Equals(value, default(T));

}

**3. В качестве параметра метода**

default может быть использован в качестве значения по умолчанию для параметра метода с обобщенным типом:

public void PrintValue<T>(T value = default(T))

{

Console.WriteLine(value);

}

**4. В условиях обобщенных методов**

Вы можете использовать default в обобщенных методах для работы с параметрами типов:

public void Process<T>(T item)

{

if (item == null || item.Equals(default(T)))

{

Console.WriteLine("Item is default value.");

}

else

{

Console.WriteLine("Item has a value.");

}

}

**5. В качестве возвращаемого значения**

Метод с обобщенным типом может возвращать значение по умолчанию с помощью default:

public T GetDefaultValue<T>()

{

return default(T); // Возвращает значение по умолчанию для типа T

}

11.Поясните как использовать статические переменные в обобщенных

классах.

И она является общей для всех экземпляров класса

Статические переменные в обобщенных классах в C# могут использоваться для хранения данных, которые общи для всех экземпляров класса, независимо от параметров типа.

12.Приведите пример обобщенного интерфейса.

public interface IGenericInterface<T>

{

void Process(T data);

T Retrieve();

}

13.В чем отличие обобщенных классов от обобщенных структур?

Обобщенные классы в C# используются для передачи типов данных по ссылке (reference types), тогда как обобщенные структуры используются для передачи значимых типов данных (value types).

Обобщенные классы могут быть унаследованы и использованы в качестве базовых классов, а обобщенные структуры - нет.

14.Какие классы для работы с файлами вы знаете? Приведите пример

пара классов File и FileInfo.

**1. System.IO.File**

Класс File предоставляет статические методы для создания, копирования, перемещения и удаления файлов. Он также позволяет читать и записывать текстовые и бинарные файлы.

**Пример:**

csharp

Copy

using System;

using System.IO;

class Program

{

static void Main()

{

string filePath = "example.txt";

// Запись текста в файл

File.WriteAllText(filePath, "Hello, World!");

// Чтение текста из файла

string content = File.ReadAllText(filePath);

Console.WriteLine(content);

}

}

**2. System.IO.FileInfo**

Класс FileInfo предоставляет методы и свойства для работы с файлами, но в отличие от класса File, он является не статическим и позволяет работать с конкретными файлами.

**Пример:**

csharp

Copy

using System;

using System.IO;

class Program

{

static void Main()

{

FileInfo fileInfo = new FileInfo("example.txt");

// Создание файла, если он не существует

if (!fileInfo.Exists)

{

fileInfo.Create().Close();

}

// Получение информации о файле

Console.WriteLine($"File Name: {fileInfo.Name}");

Console.WriteLine($"File Size: {fileInfo.Length} bytes");

}

}

**3. System.IO.Directory**

Класс Directory предоставляет статические методы для работы с директориями, включая создание, удаление и перечисление файлов и папок.

**Пример:**

csharp

Copy

using System;

using System.IO;

class Program

{

static void Main()

{

string directoryPath = "MyDirectory";

// Создание директории

Directory.CreateDirectory(directoryPath);

// Перечисление файлов в директории

string[] files = Directory.GetFiles(directoryPath);

Console.WriteLine("Files in directory:");

foreach (var file in files)

{

Console.WriteLine(file);

}

}

}

**4. System.IO.StreamReader и System.IO.StreamWriter**

Эти классы обеспечивают более гибкий способ чтения и записи текстовых файлов.

**Пример:**

csharp

Copy

using System;

using System.IO;

class Program

{

static void Main()

{

string filePath = "example.txt";

// Запись текста в файл с использованием StreamWriter

using (StreamWriter writer = new StreamWriter(filePath))

{

writer.WriteLine("Hello, World!");

}

// Чтение текста из файла с использованием StreamReader

using (StreamReader reader = new StreamReader(filePath))

{

string content = reader.ReadToEnd();

Console.WriteLine(content);

}

}

}