**C# Tuple**

Думата Tuple означава „структура от данни, която се състои от множество части“. Така че кортежът е структура от данни, която ви дава най-лесния начин да представите набор от данни, който има множество стойности, които може/може да не са свързани една с друга. Въведен е в .NET Framework 4.0. В tuple можете да добавяте елементи от 1 до 8. Ако се опитате да добавите елементи, по-големи от осем, компилаторът ще изведе грешка. Кортежите обикновено се използват, когато искате да създадете структура от данни, която съдържа обекти с техните свойства и не искате да създавате отделен тип за това.

Характеристики на кортежи:

Това ни позволява да представим множество данни в един набор от данни.

Тя ни позволява да създаваме, манипулираме и осъществяваме достъп до набор от данни.

Той връща множество стойности от метод, без да използва изходящ параметър.

Може също да съхранява дублирани елементи.

Позволява ни да предадем множество стойности към метод с помощта на единични параметри.

Каква е нуждата от C# Tuples?

Преди кортежите имаме три начина да върнем повече от една стойност от метода в C#, които са с използване на типове Class или Struct, изходящи параметри и анонимни типове, които се връщат чрез тип динамично връщане. Но след Tuples става лесно да се представи единичен набор от данни.

По друга причина просто си представете случай, в който искате да съхраните подробностите за служител само в един обект, като име, EmpID, кръвна група, номер за контакт. Сега най-често срещаният начин, който идва на ум, е да се създаде структура от данни, която да заема задължителните полета. Това е мястото, където Кортежите влизат в игра. С Tuples няма нужда да създавате отделна структура от данни. Вместо това за този случай можете да използвате Tuple<T1, T2, T3, T4>.

Най-често срещаните структури от данни като масив, списък и т.н. са само от определен тип и могат да съхраняват безкрайни елементи. Но кортежите могат да съхраняват ограничен брой елементи, т.е. 8 и могат да бъдат от всякакъв тип.

Създаване на кортеж

В C# има основно 2 начина за създаване на кортеж, които са както следва:

Използване на конструктор на клас Tuple: Можете да създадете кортеж, като използвате конструктора, предоставен от клас Tuple<T>. Къде можете да съхранявате елементи, започващи от едно до осем с техния тип. Но не ви е позволено да съхранявате елементи, по-големи от осем в кортеж. Ако се опитате да го направите, компилаторът ще изведе грешка.

Синтаксис:

// Конструктор за единични елементи

Кортеж <T1>(T1)

// Конструктор за два елемента

Кортеж <T1, T2>(T1, T2)

.

.

.

// Конструктор за осем елемента

Кортеж <T1, T2, T3, T4, T5, T6, T7, TRest>(T1, T2, T3, T4, T5, T6, T7, TRest)

Пример:

// C# program to create tuple using tuple constructor.

using System;

public class GFG{

// Main method

static public void Main (){

// Tuple with one element

Tuple<string>My\_Tuple1 = new Tuple<string>("GeeksforGeeks");

// Tuple with three elements

Tuple<string, string, int>My\_Tuple2 = new Tuple<string, string, int>("Romil", "Python", 29);

// Tuple with eight elements

Tuple<int, int, int, int, int, int, int, Tuple<int>>My\_Tuple3 = new Tuple<int, int, int, int, int, int, int, Tuple<int>>(1,2,3,4,5,6,7, new Tuple<int>(8));

}

}

Използване на метода Create:

Когато използваме конструктора на кортеж, за да създадем кортеж, трябва да предоставим типа на всеки елемент, съхраняван в кортежа, което прави кода ви тромав. И така, C# предоставя друг клас, който е клас Tuple, който съдържа статичните методи за създаване на обект на кортеж, без да предоставя типа на всеки елемент.

Синтаксис:

// Method for 1-tuple

Create(T1)

// Method for 2-tuple

Create(T1, T2)

.

.

.

// Method for 8-tuple

Create(T1, T2, T3, T4, T5, T6, T7, T8)

**Пример:**

// C# program to create tuple

// using Create Method

using System;

public class GFG {

// Main method

static public void Main()

{

// Creating 1-tuple

// Using Create Method

var My\_Tuple1 = Tuple.Create("GeeksforGeeks");

// Creating 4-tuple

// Using Create Method

var My\_Tuple2 = Tuple.Create(12, 30, 40, 50);

// Creating 8-tuple

// Using Create Method

var My\_Tuple3 = Tuple.Create(13, "Geeks", 67,

89.90, 'g', 39939, "geek", 10);

}

}

Достъп до кортеж

Можете да получите достъп до елементите на кортеж, като използвате свойството Item<elementNumber>, тук elementNumber е от 1 до 7 като Item1, Item 2, Item3, Item4, Item5, Item6, Item 7 и т.н. и последния елемент от 8-tuple е достъпен чрез използване на Rest property. Както е показано в примера по-долу:

Пример:

// C# program to access the tuple

// using Item and Rest property

using System;

public class GFG {

// Main method

static public void Main()

{

// Creating 1-tuple

// Using Create Method

var My\_Tuple1 = Tuple.Create("GeeksforGeeks");

// Accessing the element of Tuple

// Using Item property

Console.WriteLine("Element of My\_Tuple1: " + My\_Tuple1.Item1);

Console.WriteLine();

// Creating 4-tuple

// Using Create Method

var My\_Tuple2 = Tuple.Create(12, 30, 40, 50);

// Accessing the element of Tuple

// Using Item property

Console.WriteLine("Element of My\_Tuple2: " + My\_Tuple2.Item1);

Console.WriteLine("Element of My\_Tuple2: " + My\_Tuple2.Item2);

Console.WriteLine("Element of My\_Tuple2: " + My\_Tuple2.Item3);

Console.WriteLine("Element of My\_Tuple2: " + My\_Tuple2.Item4);

Console.WriteLine();

// Creating 8-tuple

// Using Create Method

var My\_Tuple3 = Tuple.Create(13, "Geeks",

67, 89.90, 'g', 39939, "geek", 10);

// Accessing the element of Tuple

// Using Item property

// And print the 8th element of tuple

// using Rest property

Console.WriteLine("Element of My\_Tuple3: " + My\_Tuple3.Item1);

Console.WriteLine("Element of My\_Tuple3: " + My\_Tuple3.Item2);

Console.WriteLine("Element of My\_Tuple3: " + My\_Tuple3.Item3);

Console.WriteLine("Element of My\_Tuple3: " + My\_Tuple3.Item4);

Console.WriteLine("Element of My\_Tuple3: " + My\_Tuple3.Item5);

Console.WriteLine("Element of My\_Tuple3: " + My\_Tuple3.Item6);

Console.WriteLine("Element of My\_Tuple3: " + My\_Tuple3.Item7);

Console.WriteLine("Element of My\_Tuple3: " + My\_Tuple3.Rest);

}

}

**Резултат:**

Element of My\_Tuple1: GeeksforGeeks

Element of My\_Tuple2: 12

Element of My\_Tuple2: 30

Element of My\_Tuple2: 40

Element of My\_Tuple2: 50

Element of My\_Tuple3: 13

Element of My\_Tuple3: Geeks

Element of My\_Tuple3: 67

Element of My\_Tuple3: 89.9

Element of My\_Tuple3: g

Element of My\_Tuple3: 39939

Element of My\_Tuple3: geek

Element of My\_Tuple3: (10)

Вложени кортежи

В C# имате право да създавате кортеж в друг кортеж. Можете да използвате вложени кортежи, когато искате да добавите повече от осем елемента в един и същи кортеж. Вложеният кортеж е достъпен чрез използване на свойството Rest, както е показано в Пример 1. Имате право да добавяте вложен кортеж навсякъде в последователността, но се препоръчва да можете да поставите вложен кортеж в края на последователността, така че да могат лесно достъп от имота Rest. Ако поставите вложен кортеж, различен от последното място, тогава кортежът е достъпен според свойството Item<elementNumber>, както е показано в пример 2.

**Пример 1:**

// C# program to illustrate nested tuple

using System;

public class GFG{

// Main method

static public void Main ()

{

// Nested Tuple

var My\_Tuple = Tuple.Create(13, "Geeks", 67, 89.90,

'g', 39939, "geek", Tuple.Create(12, 30, 40, 50));

// Accessing the element of Tuple

// Using Item property

// And accessing the elements of nested tuple

// Using Rest property

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item1);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item2);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item3);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item4);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item5);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item6);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item7);

Console.WriteLine("Element of Nested tuple: "+My\_Tuple.Rest);

Console.WriteLine("Element of Nested tuple: "+My\_Tuple.Rest.Item1.Item1);

Console.WriteLine("Element of Nested tuple: "+My\_Tuple.Rest.Item1.Item2);

Console.WriteLine("Element of Nested tuple: "+My\_Tuple.Rest.Item1.Item3);

Console.WriteLine("Element of Nested tuple: "+My\_Tuple.Rest.Item1.Item4);

}

}

**Резултат:**

Element of My\_Tuple: 13

Element of My\_Tuple: Geeks

Element of My\_Tuple: 67

Element of My\_Tuple: 89.9

Element of My\_Tuple: g

Element of My\_Tuple: 39939

Element of My\_Tuple: geek

Element of Nested tuple: ((12, 30, 40, 50))

Element of Nested tuple: 12

Element of Nested tuple: 30

Element of Nested tuple: 40

Element of Nested tuple: 50

**Пример 2:**

// C# program to illustrate nested tuple

using System;

public class GFG{

// Main method

static public void Main ()

{

// Nested Tuple

// Here nested tuple is present

// at the place of 2nd element

var My\_Tuple = Tuple.Create(13, Tuple.Create(12, 30, 40,

50),67, 89.90, 'g', 39939, 123, "geeks");

// Accessing the element of Tuple

// Using Item property

// And accessing the elements of

// nested tuple Using Rest property

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item1);

Console.WriteLine("Element of Nested Tuple: "+My\_Tuple.Item2.Item1);

Console.WriteLine("Element of Nested Tuple: "+My\_Tuple.Item2.Item2);

Console.WriteLine("Element of Nested Tuple: "+My\_Tuple.Item2.Item3);

Console.WriteLine("Element of Nested Tuple: "+My\_Tuple.Item2.Item4);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item3);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item4);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item5);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item6);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Item7);

Console.WriteLine("Element of My\_Tuple: "+My\_Tuple.Rest);

}

}

**Output:**

Element of My\_Tuple: 13

Element of Nested Tuple: 12

Element of Nested Tuple: 30

Element of Nested Tuple: 40

Element of Nested Tuple: 50

Element of My\_Tuple: 67

Element of My\_Tuple: 89.9

Element of My\_Tuple: g

Element of My\_Tuple: 39939

Element of My\_Tuple: 123

Element of My\_Tuple: (geeks)

Кортежът като параметър на метода

В C# имате право да предавате кортеж като параметър на метод, както е показано в примера по-долу. Тук предаваме кортеж с име mytuple в метода PrintTheTuple() и елементите на кортежа са достъпни чрез използване на свойството Item<elementNumber>.

Пример:

// C# program to illustrate the

// tuple as a method parameter.

using System;

public class GFG{

// Main method

static public void Main ()

{

// Creating a tuple

var mytuple = Tuple.Create("GeeksforGeeks", 123, 90.8);

// Pass the tuple in the

// PrintTheTuple method

PrintTheTuple(mytuple);

}

static void PrintTheTuple(Tuple<string, int, double>mytuple)

{

Console.WriteLine("Element: "+mytuple.Item1);

Console.WriteLine("Element: "+mytuple.Item2);

Console.WriteLine("Element: "+mytuple.Item3);

}

}

**Резултат:**

Element: GeeksforGeeks

Element: 123

Element: 90.8

Кортеж като връщащ тип

В C# методите имат право да използват tuple като тип на връщане. Или с други думи, метод може да върне кортеж, както е показано в примера по-долу:

Пример:

// C# program to illustrate

// how a method return tuple

using System;

public class GFG{

// Main Method

static public void Main ()

{

// Return tuple is stored in mytuple

var mytuple = PrintTuple();

Console.WriteLine(mytuple.Item1);

Console.WriteLine(mytuple.Item2);

Console.WriteLine(mytuple.Item3);

}

// PrintTuple method return a tuple

static Tuple<string, string, string>PrintTuple()

{

return Tuple.Create("Geeks", "For", "Geeks");

}

}

**Резултат:**

Geeks

For

Geeks

Ограничения на Tuple:

Той е от референтен тип, а не от стойностен тип.

Ограничен е до осем елемента. Означава, че не можете да съхранявате повече от осем елемента без вложен кортеж.

Те са достъпни само чрез използване на свойството Item<elementNumber>.