Tipos genéricos

en C#

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

14 de septiembre de 2023

Tipos genéricos

Temas

Genéricos

- Proveen un mecanismo para crear métodos y estructuras que funcionan sobre tipos distintos sin necesidad de utilizar conversión explícita (*casting*).
- En C#los tipos genéricos funcionan almacenan metadatos en cada ensamble (*assembly*), de modo que el ambiente de ejecución cuenta con la información del tipo dinámico de cada variable.
- Los genéricos forman parte de la plataforma .NET, por lo que son comunes y existen en tiempo de ejecución para todos los lenguajes que se ejecutan sobre ella. (*Generics in .NET* 2023)

Ejemplo de uso

Antes

```
1 Lista l = new Lista();
2 l.Agrega(new Número());
3 object o = l.Lee(0);
4 Número n = (Número) l.Lee(0);
```

vs

Ahora

```
1 Lista<Número> l = new Lista<Número>();
2 l.Agrega(new Número());
3 Número n = l.Lee(0);
```

Ejemplo de declaración

```
1  public class Lista <E>
2  {
3      private E _dato;
4      private Lista<E>? _sig;
5
6      public Lista(E dato)
7      {
8          this._dato = dato;
9      }
10
11     public void Agrega(E dato)
12     {
13         if(_sig != null) _sig.Agrega(dato);
14         else {
15             _sig = new Lista<E>(dato);
16         }
17     }
18 }
```

Ejemplo uso

```
1  public class UsoLista
2  {
3      public static void Main()
4      {
5          Lista<string?> l = new Lista<string?>("Primero");
6          l.Agrega("Hi");
7          l.Agrega(null);
8      }
9  }
```

Convensiones de nombrado

- Se utiliza una sola letra mayúscula.
- Nombres comunes son:

E	elemento (almacenado en una estructura de datos).
K	llave en un mapa o diccionario (del inglés *key*).
N	número
T	tipo (cualquiera)
V	valor
S,U,V,...	cuando se requieren más variables.

Temas

Métodos genéricos

- Si requerimos utilizar el un tipo genérico sólo en un método es posible declarar una variable local:[1]

```
1  class Programa
2  {
3      static void Intercambia<T>(ref T a, ref T b)
4      {
5          T temp = a;     a = b;     b = temp;
6      }
7      public static void Main(string[] args)
8      {
9          int x = 10, y = 20;
10         Console.WriteLine($"Antes de intercambiar: x={x},y={y}");
11         Intercambia(ref x, ref y);
12         Console.WriteLine($"Después de intercambiar: x={x},y={y}");
13     }
14 }
```

[1] https://www.csharptutorial.net/csharp-tutorial/csharp-generics/

Restricciones

Temas

Genéricos restringidos

Código: Ejemplo

```
1 public class Base {}
2 public class Genérica<T> where T : Base {}
```

Es posible restringir a los tipos genéricos para sean de alguno de tipos siguientes:[2]

- `class`, `class?` (cualquier tipo referencia)
- `struct` (cualquier tipo por valor)
- `new()` (debe tener constructor sin parámetros)
- `notnull`, `default`
- `unmanaged`
- `<clase base>`, `<interfaz base>` (con o sin ?).
- `U` (otro genérico)

[2] *Constraints on type parameters (C# Programming Guide)* s.f.

Múltiples restricciones

Código: Ejemplo

```
class ListaDeCanciones<T>
    where T : Canción, IReproducible, System.IComparable<T>, new()
{
    // ...
}
```

Código: Ejemplo

```
class Base { }
class Test<T, U>
    where U : struct
    where T : Base, new()
{ }
```

Genéricos restringidos por genéricos

Código: Ejemplos

```
1 public class Lista<T>
2 {
3     public void Agrega<U>(Lista<U> artículos) where U : T {/*...*/}
4 }
5
6 public class ClaseYAscendiente<T, U, V> where T : V { }
```

Temas

Ejemplo: Caja

```
1  public class Caja<T> where T : notnull
2  {
3      private T _objeto;
4      public T Objeto {
5          get => _objeto;
6          [System.Diagnostics.CodeAnalysis.MemberNotNull(nameof(_objeto))]
7          set
8          {
9              if (value == null)
10             {
11                 throw new ArgumentNullException("Pon␣algo␣en␣la␣caja.");
12             }
13             else _objeto = value;
14         }
15     }
16
17     public Caja(T dato) => Objeto = dato;
18 }
```

```
1  public class UsoCaja
2  {
3      public static void Main()
4      {
5          Caja<int> cajaEntero = new Caja<int>(8);
6          Console.WriteLine($"En la caja hay un {cajaEntero.Objeto}");
7
8          Caja<string> cajaCadena = new Caja<string>("H");
9          Console.WriteLine($"En la caja hay un {cajaCadena.Objeto}");
10         cajaCadena.Objeto = "Hola";
11         Console.WriteLine($"En la caja hay un {cajaCadena.Objeto}");
12     }
13 }
```

Genéricos locales restringidos

```
1 static T Add<T>(T left, T right) where T : INumber<T>
2 {
3     return left + right;
4 }
```

Nota final

Existen otras restricciones para los genéricos, pero no estaremos utilizando estos tipos de datos, por lo que hasta aquí llega esta presentación. Para más detalles puede consultarse la documentación en la bibliografía.

Bibliografía

Bibliografía I

- *Constraints on type parameters (C# Programming Guide)* (s.f.). Microsoft. URL: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/constraints-on-type-parameters.
- *Generic classes and methods* (2023). Microsoft. URL: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/types/generics.
- *Generics in .NET* (feb. de 2023). Microsoft. URL: https://learn.microsoft.com/en-us/dotnet/standard/generics/.

Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual