educación IT
CODE YOUR FUTURE

Introducción a C# .NET

Módulo 5

Estructuras

Estructuras

Las **estructuras** son un **conjunto de datos agrupados para formar un tipo de dato compuesto y personalizado**. Luego podemos definir variables con este tipo de dato personalizado; estos tipos de datos serán complejos. El lenguaje ofrece la posibilidad de implementar este tipo de dato definido por el usuario desarrollador (se los llama *User Defined Type* o *Tipo de dato definido por el usuario*).

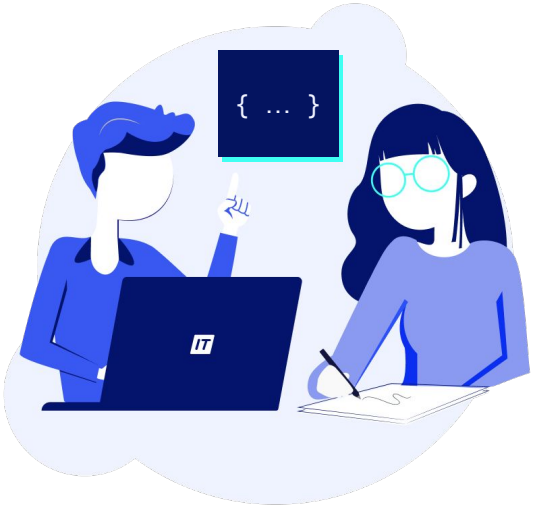
De esta manera, permite que un grupo de datos relacionados entre sí, pertenecientes a un objeto del mundo real, se puedan agrupar bajo un identificador único.

Por ejemplo, podríamos definir los siguientes tipos personalizados:

- **Cliente**: Código, Nombre, Apellido, CUIT, Domicilio.
- **Alumno**: Id, Nombre, Apellido, Mail, Teléfono, Curso.
- **Producto**: Código, Nombre, Presentación, Precio Unitario, Precio de Costo, Proveedor.

Dentro de una estructura, se llama ***miembros*** a las variables/campos que guardan los valores y a las acciones que estas estructuras pueden llevar a cabo.

Por ejemplo, se podría organizar la información de los datos bancarios de un cliente en una estructura formada por su número de cuenta, tipo de cuenta bancaria, nombre del titular, CBU, saldo, etc., y las operaciones que este cliente podrá realizar. Este agrupamiento permitirá administrar toda esa información simplificando la forma en que se accede a ella.



Cómo definir una estructura

Para crear una estructura se utilizará la palabra clave **`struct`** junto al nombre de la estructura, modelando a continuación los miembros de la estructura:

```
visibilidad struct nombre{
      visibilidad tipo campo1;
      ...
      visibilidad tipo campoN;
}
```

La visibilidad indica si la estructura puede verse por fuera del ámbito donde ha sido definida o no.

Si bien existen más, se podría utilizar un modificador de visibilidad **`public`** o **`private`**. En este ejemplo, la estructura se definirá de la siguiente manera:

```
public struct DatoCliente{
      public string NumeroCuenta;
      public string TipoCuenta;
      public string Titular;
      public string CBU;
      public double Saldo;
}
```

Cómo crear una variable de tipo estructura

Una vez definida la estructura, la forma de usarla desde el código consistirá en declarar una variable del tipo correspondiente y manipular sus miembros de forma similar a cuando lo hacemos con variables.

```
DatoCliente cliente;
```

A partir de ahora, en esta variable `cliente`, se tendrán disponibles los campos que permitirán administrar la información de un cliente. También será posible modificar esos campos cuando sea necesario. Al igual que cuando se declaran variables de tipos de datos nativos, se puede declarar más de una variable de este nuevo tipo en la misma sentencia.

```
DatoCliente cliente1, cliente2, cliente3;
```

En caso de necesitar utilizar más de una variable del mismo tipo, se puede crear un arreglo con elementos del tipo de la estructura. Al igual que cualquier arreglo, podrá dimensionarse con la cantidad de elementos necesarios. Cada elemento del `array` contendrá los datos correspondientes a un cliente específico.

```
DatoCliente[] clientes = new DatoCliente[5];
```

Creación de variables de tipo estructura

Se puede definir una variable de la estructura en la forma tradicional, pero también utilizar la palabra reservada **`new`**. De esta manera, se puede codificar un *constructor**, dentro de la estructura, que permita simplificar la creación de variables de este tipo. Si las declaramos de la forma tradicional tendremos que inicializar cada campo de la estructura por separado.

*Constructor, es un concepto de la POO, que se profundizará en el siguiente curso de C#.

Cómo acceder a los campos de una variable de tipo estructura

El acceso a los campos permitirá guardar, leer o modificar los datos, de la siguiente manera:

```
variableEstructura.Campo
```

Una vez identificados la variable y el campo con el que necesitamos trabajar, se accederá mediante el operador **punto** y, a continuación, el nombre del campo a acceder:

```
cliente1.Titular = “Pedro Gomez”;
```

Se pueden utilizar los campos en cualquier tipo de expresión válida, del mismo modo que con una variable:

```
if(ciente2.Saldo <= 0)
...
disponible = cliente2.Saldo - importeExtraccion;
...
Console.WriteLine("El saldo es {0}", cliente2.Saldo);
```

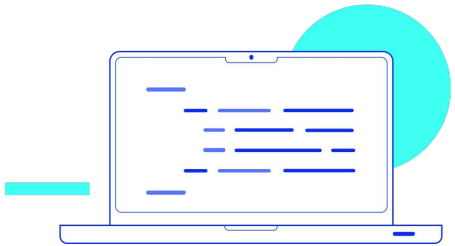
Ejemplo

```
DatoCliente[] clientes = new DatoCliente[3];
clientes[1].NumeroCuenta = "001874572";
clientes[1].tipoCuenta = "Caja de Ahorro en Pesos";
clientes[1].CBU = "01101234567845782348O1";
```

Codificación de un constructor para la estructura

Si bien no es estrictamente necesario, la codificación de un constructor dentro de la estructura tiene como objetivo simplificar la creación de variables de tipo estructura y la inicialización de los valores de sus campos.

Dentro del constructor, podemos colocar cualquier código válido de C#, aunque, en esta etapa, se escribe solamente el código necesario para la inicialización de los campos. El constructor irá dentro del bloque de código de la estructura y tendrá el mismo nombre. Veamos el ejemplo de la próxima slide.



```
public DatoCliente(String pNroCuenta, String TipoCuenta, String pTitular, string pCBU, double pSaldo)
{
// Asignamos los valores de los parámetros del constructor a los campos de la estructura
       NumeroCuenta = pNrocuenta;
       TipoCuenta = pTipoCuenta;
       Titular = pTitular;
       CBU = pCBU;
       Saldo = pSaldo;
}
```

A partir de ahora, también se podrá realizar la declaración de variables de tipo estructura de la siguiente manera:

```
DatoCliente cliente1 = new DatoCliente(“00156777”, ”Caja de Ahorro en Pesos”, ”Pedro Gomez”,
                                       ”0110123456784578234801”, 150000);
```

Esta forma de declarar variables de tipo estructura permite que la nueva variable sea creada con los valores de todos los campos, a diferencia de cuando se crea la variable de la forma tradicional, y se deben asignar los valores a sus campos por separado.

De acuerdo a distintos escenarios, en algunos casos, surgirá la necesidad de crear variables de tipo estructura con sus campos vacíos para luego cargar información en ellos, y en otros casos las variables ya podrán ser creadas con los valores correspondientes a sus campos.

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!