# Tipos de datos

Por valor

1. Caracteristicas
   1. Se guardan en el stack y se libera la memoria ni bien se sale del alcance
   2. Heredan de la clase ValueType
   3. Cuando definimos la variable esta ya tiene un valor por defecto
2. Tipos
   1. bool
   2. byte
   3. char
   4. decimal
   5. double
   6. float
   7. int
   8. long
   9. short
   10. Struct

Por referencia

1. Caracteristicas
   1. Se guardan en el heap y en el stack, el stack se libera ni bien se sale del alcance pero el heap necesita que lo libere el GC
   2. Cuando definimos la variable tiene valor null hasta que la instanciamos
2. Tipos
   1. class
   2. interface
   3. delegate
   4. dynamic
   5. object
   6. string

# Estructuras de control

1. Tipos
   1. If
   2. While
   3. For
   4. Foreach – en este caso no podemos modificar la lista que estamos recorriendo
   5. Switch

# Namespace

Sirven para ordenar el código, son similares a los package de java

Modificadores de visibilidad

1. Public – lo ven todos
2. Protected – solo se ve en la misma clase y derivadas
3. Internal – solo se ve en el mismo proyecto
4. Protected Internal – solo lo ven dentro de la misma clase o derivadas dentro del mismo proyecto
5. Private – solo se ve dentro de la misma clase

# Clases

Son las unidades de desarrollo en un sistema orientado a objetos, pueden tener métodos, propiedades, variables

Caracteristicas

1. This hace referencia a la instancia actual de la clase
2. Base hace referencia a la clase de la que hereda
3. Las clases pueden heredar de otras clases, pero solamente de 1, c# no admite herencia multiple
4. Las clases pueden implementar interfaces sin limite
5. Para impedir que una clase se pueda instanciar la definimos como abstracta
6. Para impedir que una clase sea heredada la definimos como sellada
7. Para impedir que de una clase se creen más de una instancia la definimos estática

## Definicion

Ej.

Private class persona{

Public persona(){

}

}

## Métodos

Los métodos son las acciones de una clase, puden recibir parámetros sin limite, y van a retornar un resultado o ninguno.

Caracteristicas

1. Los métodos se pueden definir como
   1. virtuales para que puedan ser sobrescritos
   2. abstractos para que tengan que ser sobrescritos
   3. estáticos para que no necesiten una instancia de la clase para ser ejecutados
2. Parametros
   1. Por defectos se reciben por valor, esto quiere decir que es una copia del valor original y cualquier cambio que hagamos solo es visible dentro del método.
   2. Por referencia, en este caso recibimos el valor original y los cambios realizados en el método son permanentes
   3. De salida, estos parámetros no necesariamente traen un valor, pero deben ser completados dentro del cuerpo del método

## Conversion de datos

Para convertir valores tenemos 3 opciones

1. As cuando conocemos el tipo de datos concreto y es diferente al de la variable podemos utilizar el operador “as“ para convertirlo
2. Podemos utilizar los métodos parse y tryparse del tipo de datos destino
3. Podemos utilizar la clase convert

## Manejo de errores

En .net todas las excepciones derivan de la clase Exception.

Para capturar un error usamos la estructura try/catch/finally

Ej.

Try{

// código que podría generar el error

}catch(Exception er){

}finally{

}

El segmento finally se va a ejecutar siempre ocurra o no un error, por lo general se utiliza para liberar recursos utilizados dentro del segmento try, aunque tengamos un return dentro del segmento try el finally se va a ejecutar igual.

No se deben utilizar las excepciones como respuesta habitual de la lógica propia de la aplicación ya que tienen un costo extra en performance.

Puedo tener todos los segmentos catch que necesite si tengo mas de un tipo de excepción, solo deben ir desde el mas especifico hacia el mas genérico.

## Collecciones

En .net existen varios tipos de colecciones

* System.Collection
* System.Collection.Generics
* System.Collections.Specialized
* System.Collections.Concurrent
* System.Collections.Immutable

Las colecciones implementa interfaces vamos nombrar a las mas importantes

* IEnumerable<T> - Nos permite usar el foreach
* ICollection<T> - Contiene la propiedad Count y los métodos CopyTo, Add, Remove y Clear
* IList<T> - Nos permite acceder a los ítems por su posición
* ISet<T> -
* IDictionary<TKey, TValue>
* ILookup<TKey, TValue>
* IComparer<T>
* IEqualityComparer<T>

Tipos de colecciones

|  |  |  |
| --- | --- | --- |
| Tipo | Acceso directo | Ordenada |
| Dictionary | Key | No |
| SortedDictionary | Key | Si |
| SortedList | Key | Si |
| List | Index | Si |
| LinkedList | No | Si |
| HashSet | Key | No |
| SortedSet | Key | Si |
| Stack | Top | LIFO |
| Queue | Front | FIFO |

Metodos comunes sobre las listas

Para agregar utilisamos el método Add, la clase List nos ofrece también AddRange para agregar varios ítems y el método Insert para poder agregar en una posicion determinada, en la clase Queue utilizamos el método Enqueue y en la clase Stack utilizamos Push