LINQ in C#
Filtering
var col = from o in Orders
where o.CustomerID == 84
select o;
var col2 = Orders
.Where ( o => o.Custo...

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Types**   |  |  | | --- | --- | | bool | Boolean value | | byte | 8-bit unsigned integer | | char | 16-bit Unicode character | | decimal | 128-bit precise decimal values with 28-29 signif­icant digits | | double | 64-bit double­-pr­ecision floating point | | float | 32-bit single­-pr­ecision floating point | | int | 32-bit signed integer | | long | 64-bit signed integer | | object | Base type for all other types | | sbyte | 8-bit signed integer | | short | 16-bit signed integer | | string | String value | | uint | 32-bit unsigned integer | | ulong | 64-bit unsigned integer | | ushort | 16-bit unsigned integer |   **Type Conversion Methods**   |  | | --- | | ToBoolean | | ToByte | | ToChar | | ToDateTime | | ToDecimal | | ToDouble | | ToInt16 | | ToInt32 | | ToInt64 | | ToSbyte | | ToSingle | | ToString | | ToType | | ToUInt16 | | ToUInt32 | | ToUInt64 |   **Naming Conven­tions**   |  |  | | --- | --- | | Class | MyClass | | Method | MyMethod | | Local variable | myLoca­lVa­riable | | Private variable | \_myPri­vat­eVa­riable | | Constant | MyConstant |   **Arrays**   |  | | --- | | int[] array = new int[] {1, 2, 3} | | int[] array = {1, 2, 3} | | var array = new int[] {1, 2, 3} | | int[] array = new int[3] |   **Statements**   |  |  | | --- | --- | | if-else | if (true) {...} else if (true) {...} else {...} | | switch | switch (var) { case 1: break; default: break; } | | for | for (int i =1; i < 5; i++) {...} | | foreach-in | foreach (int item in array) {...} | | while | while (true) {...} | | do... while | do {...} while (true); | | try-ca­tch­-fi­nally | try {...} catch (Exception e) {...} catch {...} finally {...} |   **Classes**   |  |  |  | | --- | --- | --- | | Class | public class Dog {...} | | | Inheri­tance | public class Dog: Pet {...} | | | Constr­uctor (no parame­ters) | public Dog () {...} | Constr­uctors can co-exist | | Constr­uctor (one parameter) | public Dog (string var) {...} | Constr­uctors can co-exist | | Field | public string name | | | Static Class | public static class Dog {...} | Must only have static members | | Static Member | public static int = 1 | | | Finalizer (destr­uctor) | ~Dog () {...} | Cannot have modifiers or parameters | |  | **Access Modifiers**   |  |  | | --- | --- | | public | Accessible by any other code in the same assembly or another assembly that references it | | private | Only accessible by code in the same class or struct | | protected | Only accessible by code in the same class or struct, or in a derived class | | internal | Accessible by any code in the same assembly, but not from another assembly | | protected internal | Accessible by any code in the same assembly, or by any derived class in another assembly |   **Other Modifiers**   |  |  | | --- | --- | | abstract | Indicates that a class is intended only to be a base class of other classes | | async | Indicates that the modified method, lambda expres­sion, or anonymous method is asynch­ronous | | const | Specifies that the value of the field or the local variable cannot be modified | | event | Declares an event | | extern | Indicates that the method is implem­ented externally | | new | Explicitly hides a member inherited from a base class | | override | Provides a new implem­ent­ation of a virtual member inherited from a base class | | partial | Defines partial classes, structs and methods throughout the same assembly | | readonly | Declares a field that can only be assigned values as part of the declar­ation or in a constr­uctor in the same class | | sealed | Specifies that a class cannot be inherited | | static | Declares a member that belongs to the type itself instead of to a specific object | | unsafe | Declares an unsafe context | | virtual | Declares a method or an accessor whose implem­ent­ation can be changed by an overriding member in a derived class | | volatile | Indicates that a field can be modified in the program by something such as the operating system, the hardware, or a concur­rently executing thread |   **Assignment Operators**   |  |  | | --- | --- | | = | Simple assignment | | += | Addition assignment | | -= | Subtra­ction assignment | | \*= | Multip­lic­ation assignment | | /= | Division assignment | | %= | Remainder assignment | | &= | AND assignment | | |= | OR assignment | | ^ | XOR assignment | | <<= | Left-shift assignment | | >>= | Right-­shift assignment |   **Comparison Operators**   |  |  | | --- | --- | | < | Less than | | > | Greater than | | <= | Less than or equal to | | >= | Greater than or equal to | | == | Equal to | | != | Not equal to |   **Arithmetic Operators**   |  |  | | --- | --- | | + | Add numbers | | - | Subtract numbers | | \* | Multiply numbers | | / | Divide numbers | | % | Compute remainder of division of numbers | | ++ | Increases integer value by 1 | | -- | Decreases integer value by 1 |   **Logical and Bitwise Operators**   |  |  | | --- | --- | | && | Logical AND | | || | Logical OR | | ! | Logical NOT | | & | Binary AND | | | | Binary OR | | ^ | Binary XOR | | ~ | Binary Ones Complement | | << | Binary Left Shift | | >> | Binary Right Shift |   **Other Operators**   |  |  | | --- | --- | | sizeof() | Returns the size of a data type | | typeof() | Returns the type of a class | | & | Returns the address of a variable | | \* | Pointer to a variable | | ? : | Condit­ional expression | | is | Determines whether an object is of a specific type | | as | Cast without raising an exception if the cast fails | |







