Cheatography

General C# Cheat Sheet

by Veyleria via cheatography.com/99177/cs/21007/

Data Types

byte	8-bit unsigned integer	0 to 255	`byte value = 255;`
int	32-bit signed integer	-2,147,483,648 to 2,147,483,647	`int value = 3;`
float	32-bit Single-precision floating point type	-3.402823e38 to 3.402823e38	`float value = 6.3F;`
char	16-bit single Unicode character	Any valid character, e.g. a,*, \x0058 (hex), or\u0058 (Unicode)	`char value = 'H';`
bool	8-bit logical true/false value	True or false.	`bool value = true;`
string	A sequence of Unicode characters	Combination of characters.	`string value = " Hel lo ";`

Type Conversion Methods

```
Conver t.T oBo ole an( var iable);
Conver t.T oBy te( var iable);
Conver t.T oCh ar( var iable);
Conver t.T oDa teT ime (va ria ble);
Conver t.T oIn t32 (va ria ble);
Conver t.T oSt rin g(v ari able(;
```

Naming Conversions

Class	MyClass
Method	MyMethod
Local variable	myLocalVariable
Private variable	_myPrivateVariable
Constant	MyConstant

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	`foreach (int item in array) {...}`
while	`while (true) {...}`
do-while	`do {...} while (true);`
try-catch-finally	`try {...} catch (Exception e) {...} catch {...} finally {...}`

By **Veyleria**
cheatography.com/veyleria/

Not published yet.
Last updated 3rd November, 2019.

Cheatography

General C# Cheat Sheet

by Veyleria via cheatography.com/99177/cs/21007/

Arrays and Methods

```
int[] array = new int[] {1, 2, 3};
```

```
int[] array = {1, 2, 3};
```

```
var array = new int[] {1, 2, 3};
```

```
int[] array = new int[3];
```

```
int[,] array2D = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
```

```
int[,] array2Da = new int[4, 2] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
```

```
int[, ,] array3D = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } }, { { 7, 8, 9 }, { 10, 11, 12 } } };
```

```
int[, ,] array3Da = new int[2, 2, 3] { { { 1, 2, 3 }, { 4, 5, 6 } }, { { 7, 8, 9 }, { 10, 11, 12 } } };
```

```
array.G et Len gth (int32)
```

Classes

Class	`public class Animal {...}`	Makes a new class named Animal.
Inheritance	`public class Dog:Animal {...}`	Inherits from a class. Example every animal has a size, but not every animal is the same size.
Constructor (no parameters)	`public Dog() {...}`	Method in a class that activates when the class is instanciated.
Constructor (one parameter)	`public Dog (string var) {...}`	Method in a class that activates when the class is instanciated with parameters.
Deconstructor (cannot have parameters)	`~Dog () {...}`	Method in a class that activates when the class is destroyed.
Call method	`Method Name();`	Calls a custom or already existing method.

Lists and Methods

`List<T ype> listName = new List<T >( );`	Declares a new list.
`listNa me.C ount`	Gets the number of elements contained in the List<T>.
`listNa me.A dd(T);`	Adds an object to the end of the List<T>.
`listNa me.C le ar();`	Removes all elements from the List<T>.
`listNa me.C on tai ns(T);`	Determines whether an element is in the List<T>.
`listNa me.E qu als (Ob ject);`	Determines whether the specified object is equal to the current object.
`listNa me.I nd exO f(T);`	Searches for the specified object and returns the zero-based index of the first occurrence within the entire List<T>.
`listNa me.R em ove(T);`	Removes the first occurrence of a specific object from the List<T>.
`listNa me.R em ove At( Int32);`	Removes the element at the specified index of the List<T>.

Access Modifiers

public	Accessible by any other code in the same assembly or another assembly that references it.	`public int ...;`
private	Only accessible by code in the same class or struct.	`private int ...;`
protected	Only accessible by code in the same class or struct, or in a derived class	`protected int ...;`

Other Modifiers

abstract	Indicates that a class is intended only to be a base class of other classes.	`abstract class Shape { ... }`
async	Indicates that the modified method, lambda expression, or anonymous method is asynchronous. (This is used if a function needs to have an delay or await)	`private async void Task() { ... }`
const	Specifies that the value of the field or the local variable cannot be modified. (You cannot say X = 1; later in the program if it's a const)	`const int X = 0;`
event	Declares an event. Mostly used in combination with an delegate.	`public event Sample Eve ntH andler Sample Event;`
delegate	Declares a delegate. Mostly used in combination with an event.	`public delegate void Sample Eve ntH and ler (object sender, Sample Eve nt Args e;`
new	The new operator creates a new instance of a type.	`public Random random = new Random();`

Other Modifiers (cont)		
override	Provides a new implementation of a virtual member inherited from a base class.	`public override void ToString() { ... }`
readonly	Declares a field that can only be assigned values as part of the declaration or in a constructor in the same class. (Same as const, you cannot change the value later)	`private readonly int value = 6;`
static	Declares a member that belongs to the type itself instead of to a specific object.	`static int = 7;`

Assigment Operators	
=	Simple assignment.
+=	Addition assignment.
-=	Subtraction assignment.
*=	Multiplication assignment.
/=	Division assignment.
%=	Remainder assignment.
&=	AND assignment.
|=	OR 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.
/	Devide numbers.
%	Compute remainder of division of numbers.
++	Increases integer value by 1.
--	Decreases integer value by 1.

Logical Operators

&&	Logical AND.
||	Logical OR.
!	Logical NOT.

Other Operators

&	Returns the address of a variable.
*	Pointer to a variable.
? :	Conditional expression. `is this condition true ? yes : no;`
is	Determines whether an object is of a specific type.
as	Cast without raising an exception if the cast fails.

Console

`Consol e.C lear();`	Clears the console buffer and corresponding console window of display information.
`Consol e.R ead Key() ;`	Obtains the next character or function key pressed by the user. The pressed key is displayed in the console window.
`Consol e.R ead Line( );`	Reads the next line of characters from the standard input stream.
`Consol e.W rit eLi n e();`	Writes the current line terminator to the standard output stream.

Misc

//	Adds a comment.
#region RegionName - #endregion	Makes a region (for code colapsing) and ends it with endregion.