codeguru™

C# 4.0 Cheat Sheet

Operators

Basic (~ = bit complement)	+ - * / % ~
Increment/decrement	++x x++ --x x--
Comparison	== != > >= < <=
Logical (^ = xor, ! = not)	& | ^ !
Assignment	= += -= *= /= %= &= |= ^=
Conditional AND & OR	&& ||
Shifting	>> >>= << <<=
Type testing	is as
Conditional ?:	x == 5 ? true : false
Null coalescing ?? (nullable type testing)	int? x = 5; int y = x + 1 ?? -1
Lambda	=>

Data types (all value types unless specified)

Signed integral types	sbyte (8 bits, -128..127) short (16 bits) int (32 bits) long (64 bits)
Unsigned integral types	byte (8 bits, 0..255) ushort (16 bits) uint (32 bits, non-CLS) ulong (64 bits)
Floating-point types (IEEE)	float (32 bits) double (64 bits)
High-precision	decimal (128 bits)

Real value literals (default = double; lowercase or uppercase OK)	float f = 123.45f;
	double d = 123.45d;
	decimal m = 123.45m;

Overflow checking (default is on)	z = checked(x+y); z = unchecked(x+y);
Boolean	bool
Character and strings	char string
Structure	struct
Enumeration	enum
Reference types	class, interface, delegate, object, string

C# type aliases

C# alias	.NET class
bool	System.Boolean
byte	System.Byte
char	System.Char
decimal	System.Decimal
double	System.Double
float	System.Single
int	System.Int32
long	System.Int64
object	System.Object
sbyte	System.SByte
short	System.Int16
string	System.String
uint	System.UInt32
ulong	System.UInt64
ushort	System.UInt16

Dynamic and Object

In C# 4.0, the dynamic keyword allows you work with objects whose type is only known at run-time. In certain cases, it works differently than the general object type.

Operation	Example	Notes
Definitions 1:	*dynamic d = 123;* *object o = 123;*	assignment to a type
Types at runtime	MessageBox.Show(d.GetType().FullName); MessageBox.Show(o.GetType().FullName);	returns System.Int32 returns System.Int32
Working with types	int i = d + 10; int i = (int)o + 10;	 must cast to an int
Definitions 2:	*private void Test(int param)* *{* *MessageBox.Show(param.ToString());* *}*	
Passing as parameters	Test(d); Test((int)o);	 must cast to an int
Calling methods, compile-time behaviour	d.MyMethod(); o.MyMethod();	compiles fails at compile-time
Calling methods, run-time behaviour	d.MyMethod(); o.MyMethod();	runtime exception n/a

Formatting strings

Format string	Example	Result
Definitions for numeric values:	*double d = 123.45;* *int i = 123;*	*Culture: "en-US"*
c or C (currency)	string.Format("{0:c}", d); string.Format("{0:c}", i));	$123.45 $123.00
d or D (decimal; optional precision can be given as in "d8")	string.Format("{0:d8}", d); string.Format("{0:d8}", i));	*(not valid for floats)* 00000123
e or E (scientific [exponential])	string.Format("{0:e}", d); string.Format("{0:e}", i));	1.234500e+002 1.230000e+002
f or F (fixed-point; optional precision can be given as in "f1")	string.Format("{0:f1}", d); string.Format("{0:f1}", i));	123.5 123.0
g or G (general; optional precision can be given as in "g5")	string.Format("{0:g}", d); string.Format("{0:g}", i));	123.45 123
n or N (number)	string.Format("{0:n}", d); string.Format("{0:n}", i));	123.45 123.00

x or X (hexadecimal; x = lowercase, X = uppercase)	"string.Format("{0:x}", d);
	string.Format("{0:x}", i));
	string.Format("{0:X}", i));
		(not valid for floats)
		7b
		7B

Miscellaneous

Verbatim strings with the @ character	string path = "C:\\Path\\File.txt""; string path = @"C:\Path\File.txt";

Regular Expressions

Class used	*System.Text.RegularExpressions.Regex (in System.dll)*
Test if input string matches	string input = "ABC123"; string pattern = "[A-Z]{3}\d{3}"; Regex.IsMatch(input, pattern); // match
Replace unwanted characters	string input = " 329A 83"; string invalid = "[^0-9]"; string safe = Regex.Replace(input, invalid, ""); // safe becomes "32983"
Common character escapes	\t Tab (\u0009) \r Carriage return (\u000D) \n New line (\u000A) \x4F ASCII character code in hex \u0020 Unicode character code in hex
Common character classes	[aeiou] vowels [^aeiou] non-vowels [a-z] character range . period, matches any character \d decimal digits \w word characters \W non-word characters
Common quantifiers	+ one or more occurrences * zero or more occurrences ? zero or one occurrence {n} exactly n occurrences {n,} at least n occurrences {n,m} n to m occurrences

Operator Precedence

C# operators have equal precedence within a group.

Operator group	Operator examples

Primary	x.y // member access
	x(y) // method access
	x++ // post increment
	typeof(x) // type retrieval
	new X() // object creation

Unary	+x -x !x ~x ++x
Multiplicative	x*y x/y x%y
Additive	x+y x-y
Shift	x<<y x>>y
Relational and type testing	x<y x>y x is y x as y
Equality	x==y x!=y
Logical AND	x & y
Logical XOR	x ^ y
Logical OR	x | y
Conditional AND	x && y
Conditional OR	x || y
Null coalescing	x ?? y
Conditional	x ? y : z
Assignment or anonymous function	x = y x += y x *= y x => y // lambda

Pre-processing Directives

Declaration directives	#define symbol #undef symbol
Conditional compilation directives	#if #else #elif #endif
Source code line directives	#line number “filename” #line default #line hidden
Diagnostic directives	#warning #error
Region directives	#region #endregion
Pragma directives	#pragma warning #pragma warning no(s) #pragma warning disable #pragma warning restore

Language Keywords

The following keywords have been defined for the C# language. An identifier can have the name of a keyword if it is prefixed with the @ sign: @readonly.

Group	Words
Keywords	abstract as base bool break byte case catch char checked class const continue decimal default delegate do double else enum event explicit extern false finally fixed float for foreach goto if implicit in int interface internal is lock long namespace new null object operator out override params private protected public readonly ref return sbyte sealed short sizeof stackalloc static string struct switch this throw true try typeof uint ulong unchecked unsafe ushort using virtual void volatile while
Contextual keywords	get set var

Explicit Conversions requiring Type Cast

From	To
sbyte	byte, ushort, uint, ulong, or char
byte	sbyte and char
short	sbyte, byte, ushort, uint, ulong, or char
ushort	sbyte, byte, short, or char
int	sbyte, byte, short, ushort, uint, ulong, or char
uint	sbyte, byte, short, ushort, int, or char
long	sbyte, byte, short, ushort, int, uint, ulong, or char
ulong	sbyte, byte, short, ushort, int, uint, long, or char
char	sbyte, byte, or short
float	sbyte, byte, short, ushort, int, uint, long, ulong, char, or decimal
double	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, or decimal
decimal	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, or double
Enumeration type	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, double, or decimal
sbyte, byte, short, ushort, int, uint, long, ulong, char, float, double, or decimal	Any enumeration type
Object (class instance) or dynamic	Any reference type
Class T1	Class T2, if T2 descends from T1
Class T1	Interface I2, if T1 is not sealed and T1 does not implement I2

LINQ to Entities Inner Join

Inner Join Query	var innerJoinQuery = from category in categories join prod in products on category.ID equals prod.CategoryID select new { ProductName = prod.Name, Category = category.Name };

codeguru™

C# 4.0 Cheat Sheet version 1.0 by
Jani Järvinen <janij@dystopia.fi>

Character Sets and Conversions

Available encoding classes in the System.Text namespace	ASCIIEncoding UTF7Encoding UTF8Encoding UnicodeEncoding UTF32Encoding
Code pages	UTF-7 = code page 65000 UTF-8 = 65001 Unicode, little endian = 1200 Unicode, big endian = 1201 UTF-32, little endian = 65005 UTF-32, big endian = 65006
Accessing Encoding class instances	using System.Text; … Encoding.ASCII Encoding.UTF7 Encoding.UTF8 Encoding.Unicode Encoding.UTF32
Getting the bytes from a string in selected encoding	string s = "Some data: åäö-αβγδ"; byte[] bytes1 = Encoding.UTF8.GetBytes(s); byte[] bytes2 = Encoding.UTF32.GetBytes(s); …
Convert a byte array to a string based on a certain encoding	byte[] bytes = ...; string s = Encoding.UTF7.GetString(bytes);
Get BOM (*Byte Order Mark*) or preamble for a given encoding	byte[] bom1 = Encoding. UTF8.GetPreamble(); byte[] bom2 = Encoding. UTF32.GetPreamble();

Strings and Readers

Convert string to a TextReader by using a StringReader (*StringReader descends from the TextReader abstract class*)	using System.IO;
	...
	string s = “ABCD...”;
	StringReader reader =
	new StringReader(s);
	
	// example use:
	using System.Xml.XPath;
	...
	XPathDocument doc =
	new XPathDocument(reader);

Read contents of a file into a string	using System.IO; string filename = @”C:\Path\File.dat”; string data = File.ReadAllText(filename);