

Core C# and .NET Quick Reference

1. Data Types

Primitive	Size	Example
string	2 bytes/char	s = "reference";
bool		b = true;
char	2 bytes	ch = 'a';
byte	1 byte	b = 0x78;
short	2 bytes	lval = 54;
int	4 bytes	ival = 540;
long	8 bytes	ival = 5400;
float	4 bytes	val = 54.0F;
double	8 bytes	val = 54.0D;
decimal	16 bytes	val = 54.0M;

2. Arrays

Declaration

```
int[] numArray = {1903, 1907, 1910};
int[] numArray = new int[3];
// 3 rows and 2 columns
int[,] nums = {{1907, 1990}, {1904, 1986}, {1910, 1980}};
```

Array Operations

```
Array.Sort(numArray); // sort ascending
// Sort begins at element 4 and sorts 10 elements
Array.Sort(numArray, 4, 10);
// Use one array as a key and sort two arrays
string[] values = {"Cary", "Gary", "Barbara"};
string[] keys = {"Grant", "Cooper", "Stanwyck"};
Array.Sort(keys, values);
// Clear elements in array (array, 1st element, # elements)
Array.Clear(numArray, 0, numArray.Length);
// Copy elements from one array to another
Array.Copy(src, target, numelements);
```

3. String Operations

Method	Description
Compare	String.Compare(stra, strb, case, ci) bool case – true for case insensitive ci – new CultureInfo("en-US") returns: <0 if a<b, 0 if a=b, 1 if a>b
IndexOf	str.IndexOf(val, start, num) val – string to search for start – where to begin in string num – number of chars to search returns (-1) if no match.
LastIndexOf	Search from end of string.
Replace	newstr= oldstr.Replace("old", "new");
Split	Char[] delim= { ' ', ',' }; string w = "Kim, Joanna Leslie"; // create array with three names string[] names= w.Split(delim);

6. Formatting Numeric and Date Values

Format Item Syntax: {index[,alignment] [:format string]}

index – Specifies element in list of values to which format is applied.

alignment – Indicates minimum width (in characters) to display value.

format string – Contains the code that specifies the format of the displayed value.

Example: String.Format("Price is: {0:C2}", 49.95); // output: Price is: \$ 49.95

a. Numeric Formatting

Format Specifier	Pattern	Value	Description
C or c	{0:C2}, 1388.55	\$ 1388.55	Currency.
D or d	{0:D5}, 45	00045	Must be integer value.
E or e	{0,9:E2}, 1388.55	1.39+E003	Must be floating point.
F or f	{0,9:F2}, 1388.55	1388.55	Fixed Point representation.
N or n	{0,9:N1}, 1388.55	1,388.6	Insert commas
P or p	{0,9:P3}, .7865	78.650%	Converts to percent.
R or r	{0,9:R}, 3.14159	3.14159	Retains all decimal places.
X or x	{0,9:X4}, 31	001f	Converts to Hex

Example: CultureInfo ci = new CultureInfo("de-DE"); // German culture
string curdt = String.Format(ci, "{0:M}", DateTime.Now); // 29 Juni

b. DateTime Formatting: (January 19, 2005 16:05:20) en-US

Format	Value Displayed	Format	Value Displayed
d	1/19/2005	Y or y	January, 2005
D	Wednesday, January 19, 2005	t	4:05 PM
f	Wednesday, January 19, 2005 4:05:20 PM	T	4:05:20 PM
F	Wednesday, January 19, 2005 4:05 PM	s	2005-01-19T16:05:20
g	1/19/2005 4:05 PM	u	2005-01-19 16:05:20Z
G	1/19/2005 4:05:20 PM	U	Wednesday, January 19, 2005 21:05:20PM
M or m	January 19		

7. Using the System.Text.RegularExpressions.Regex class

```
string zipexp = @"^d{5}((-|s)?d{4})?$$";
string addr="W.44th St, New York, NY 10017-0233";
Match m = Regex.Match(addr, zipexp); // Static method
Regex zipRegex= new Regex(zipexp);
m= zipRegex.Match(addr); // Use Regex Object
Console.WriteLine(m.Value); // 10017-0233
```

Pattern	Description	Example
+	Match one or more occurrence	ab+c matches abc, abcc
*	Match zero or more occurrences	ab*c matches ac, abcc
?	Matches zero or one occurrence	ab?c matches ac, abc
\d \D	Match decimal digit or non-digit (\D)	\d\d matches 01, 55
\w \W	Match any word character or non-char	\w equals [a-zA-Z0-9_]
\s \S	Match whitespace or non-whitespace	\d*\s\d+ matches 246 98
[]	Match any character in set	[aeiou]n matches in, on
[^]	Match any character not in set	[^aeiou] matches r or 2
a b	Either a or b	jpg jpeg gif matches .jpg
\n \r \t	New line, carriage return, tab	

Method	Description
Substring	mystring.Substring(ndx, len) string alpha = "abcdef"; // returns "cdef" string s= alpha.Substring(2); // returns "de" s = alpha.Substring(3,2);
ToCharArray	Places selected characters in a string in a char array: String vowel = "aeiou"; // create array of 5 vowels char[] c = vowel.ToCharArray(); // create array of 'i' and 'o'. char[] c = vowel.ToCharArray(2,2);

4. System.Text.StringBuilder

Constructor

```
StringBuilder sb = new StringBuilder();
StringBuilder sb = new StringBuilder(mystring);
StringBuilder sb = new StringBuilder(mystring, capacity);
```

mystring – Initial value of StringBuilder object
capacity – Initial size (characters) of buffer.

Using StringBuilderMembers

```
decimal bmi = 22.2M;
int wt=168;
StringBuilder sb = new StringBuilder("My weight is ");
sb = sb.Append(wt); // can append number
sb= sb.Append(" and my bmi is ").Append(bmi);
// my weight is 168 and my bmi is 22.2
sb= sb.Replace("22.2", "22.4");
string s = sb.ToString();
// Clear and set to new value
sb.Length=0;
sb.Append("Xanadu");
```

5. DateTime and TimeSpan

DateTime Constructor

```
DateTime(yr, mo, day)
DateTime(yr, mo, day, hr, min, sec)
```

```
DateTime bday = new DateTime(1964, 12, 20, 11, 2, 0);
DateTime newyr= DateTime.Parse("1/1/2005");
DateTime currdt = DateTime.Now;
// also AddHours, AddMonths, AddYears
DateTime tomorrow = currdt.AddDays(1);
TimeSpan diff = currdt.Subtract(bday);
// 14795 days from 12/20/64 to 6/24/05
Console.WriteLine("{0}", diff.Days);
```

// TimeSpan(hrs, min, sec)

```
TimeSpan ts = new TimeSpan(6, 30, 10);
// also FromMinutes, FromHours, FromDays
TimeSpan ts = TimeSpan.FromSeconds(120);
TimeSpan ts = ts2 - ts1; // +, -, >, <, ==, !=
```

8. Using the C# Compiler at the Command Line

```
C:\>csc /t:library /out:reslib.dll mysource.cs
csc /t:winexe /r:ctls1.dll /r:ctls2.dll winapp.cs
csc /keyfile:strongkey.snk secure.cs
```

Option	Description
/addmodule	Import metadata from a file that does not contain a manifest.
/debug	Tells compiler to emit debugging info.
/doc	Specifies an XML documentation file to be created during compilation.
/keyfile	Specifies file containing key used to create a strong named assembly.
/lib	Specifies directory to search for external referenced assemblies.
/out	Name of compiled output file.
/reference (/r)	Reference to an external assembly.
/resource	Resource file to embed in output.
/target (/t)	/t:exe /t:library /t:module /t:winexe

9. C# Language Fundamentals

Control Flow Statements	
switch (<i>expression</i>) <pre>{ case <i>expression</i>: // statements break / goto / return() case ... default: // statements break / goto / return() }</pre> <i>expression</i> may be integer, string, or enum.	<pre>switch (genre) { case "vhs": price= 10.00M; break; case "dvd": price=16.00M; break; default: price=12.00M; break; }</pre>
if (condition) { // statements } else { // statements }	<pre>if (genre=="vhs") price=10.00M; else if (genre=="dvd") price=16.00M; else price=12.00M;</pre>
Loop Constructs	
while (condition) <pre>{ body }</pre>	<pre>while (ct < 8) { tot += ct; ct++; }</pre>
do { body } while (condition);	<pre>do { tot += ct; ct++; } while (ct < 8);</pre>

11. Delegates and Events

Delegates

[modifiers] **delegate** result-type *delegate name* ([parameter list]);

```
// (1) Define a delegate that calls method(s) having a single string parameter
public delegate void StringPrinter(string s);
// (2) Register methods to be called by delegate
StringPrinter prt = new StringPrinter(PrintLower);
prt += new StringPrinter(PrintUpper);
prt("Copyright was obtained in 2005"); // execute PrintLower and PrintUpper
```

Using Anonymous Methods with a Delegate

Rather than calling a method, a delegate encapsulates code that is executed:

```
prt = delegate(string s) { Console.WriteLine(s.ToLower()); };
prt += delegate(string s) { Console.WriteLine(s.ToUpper()); };
prt("Print this in lower and upper case.");
```

Events

```
// class.event += new delegate(event handler method);
Button Total = new Button();
Total.Click += new EventHandler(GetTotal);
// Event Handler method must have signature specified by delegate
private void GetTotal( object sender, EventArgs e) {
```

Commonly used Control Events

Event	Delegate
Click, MouseEnter, DoubleClick, MouseLeave	EventHandler (object sender, EventArgs e)
MouseDown, Mouseup, MouseMove	EventHandler (object sender, MouseEventArgs e) e.X, e.Y – x and y coordinates e.Button – MouseButton.Left, Middle, Right
KeyUp, KeyDown	EventHandler (object sndr, KeyEventArgs e) e.Handled – Indicates whether event is handled. e.KeyCode – Keys enumeration, e.g., Keys.V e.Modifiers – Indicates if Alt, Ctrl, or Shift key.
KeyPress	EventHandler (object sender, KeyPressEventArgs e)

12. struct

[attribute][modifier] **struct** name [:interfaces] { struct-body }

Differences from class:

- is a value type
- cannot inherit from a class or be inherited
- fields cannot have initializer
- explicit constructor must have a parameter

13. enum (Enumerated Type)

enum	enum Operations
<pre>enum Fabric: int { cotton = 1, silk = 2, wool = 4, rayon = 8 }</pre>	<pre>int cotNum = (int) Fabric.cotton; // 1 string cotName = Fabric.cotton.ToString(); // cotton string s = Enum.GetName(typeof(Fabric),2); // silk // Create instance of wool enum if it is valid if(Enum.IsDefined(typeof(Fabric), "wool") Fabric woolFab = (Fabric)Enum.Parse(typeof(Fabric),"wool");</pre>

Loop Constructs (Continued)

for (initializer; termination condition; iteration;) <pre>{ // statements }</pre>	<pre>for (int i=0;i<8;i++) { tot += i; }</pre>
foreach (type identifier in <i>collection</i>) <pre>{ // statements }</pre>	<pre>int[] ages = {27, 33, 44}; foreach(int age in ages) { tot += age; }</pre>

10. C# Class Definition

Class
<pre>[public protected internal private] [abstract sealed static] class <i>class name</i> [:class/interfaces inherited from]</pre>
Constructor
<pre>[access modifier] <i>class name</i> (parameters) [:initializer]</pre> <p>initializer – base calls constructor in base class. this calls constructor within class.</p> <pre>public class Shirt: Apparel { public Shirt(decimal p, string v) : base(p,v) { constructor body }</pre>
Method
<pre>[access modifier] [static virtual override new sealed abstract] <i>method name</i> (parameter list) { body }</pre> <p>virtual – method can be overridden in subclass. override – overrides virtual method in base class. new – hides non-virtual method in base class. sealed – prevents derived class from inheriting. abstract – must be implemented by subclass.</p> <p><i>Passing Parameters:</i> a. By default, parameters are passed by value. b. Passing by reference: ref and out modifiers</p> <pre>string id= "gm"; // caller initializes ref int weight; // called method initializes GetFactor(ref id, out weight); // ... other code here static void GetFactor(ref string id, out int wt) { if (id=="gm") wt = 454; else wt=1; return; }</pre>
Property
<pre>[modifier] <datatype> <i>property name</i> { public string VendorName { get { return vendorName; } set { vendorName = value; } // note <i>value</i> keyword }</pre>