**Core C# and .NET**

Quick Reference

|  |  |  |
| --- | --- | --- |
| **1. Data Types** |  |  |
|  |  |  |
| **Primitive** | **Size** | **Example** |
| **string** | 2 bytes/char | s = “reference”; |
| **boo**l |  | b = true; |
| **char** | 2 bytes | ch = ‘a’; |
| **byte** | 1 byte | b = 0x78; |
| **short** | 2 bytes | Ival = 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*, *star*t, *num*) |
|  | *val* – string to search for |
|  | *star*t – 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); |

**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; // +,-,>,<,==, !=

|  |  |  |
| --- | --- | --- |
| **6. C# Language Fundamentals** | | |
|  |  |  |
|  | **Control Flow Statements** | |
|  | **switch** (*expression*) | switch (genre) |
|  | { case *expression*: | { |
|  | // statements | case “vhs”: |
|  | break / goto / return() | price= 10.00M; |
|  | case ... | break; |
|  | default: | case “dvd”: |
|  | // statements | price=16.00M; |
|  | break / goto / return() | break; |
|  | } | default: |
|  | *expression* may be | price=12.00M: |
|  | integer, string, or enum. | break; |
|  |  | } |
|  |  |  |
|  | **if** (condition) { | if (genre==”vhs”) |
|  | // statements | price=10.00M; |
|  | } else { | else if (genre==”dvd”) |
|  | // statements | price=16.00M; |
|  | } | else price=12.00M; |
|  |  |  |
|  | **Loop Constructs** |  |
|  | **while** (condition) | while ( ct < 8) |
|  | { body } | { tot += ct; ct++; } |
|  | **do** { body } | do { tot += ct; ct++;} |
|  | while (condition); | while (ct < 8); |
|  |  |  |
|  | **for** (initializer; | for (int i=0;i<8;i++) |
|  | termination condition; | { |
|  | iteration;) | tot += i; |
|  | { // statements } | } |
|  | **foreach** (type identifier in | int[] ages = {27, 33, 44}; |
|  | *collection*) | foreach(int age in ages) |
|  | { // statements } | { tot += age; } |