

Comparison of C#, Clarion# and VB.NET

This is a quick reference guide to highlight some key syntactical differences between C#, Clarion# and VB.NET (version 2).

NOTE: Clarion.NET and the Clarion# language are currently in alpha test, and various areas of the documentation are incomplete and in a state of flux. Therefore, it's very likely that some of the entries will change as new information becomes available.

Program Structure		
C#	Clarion#	VB.NET
<pre>using System; namespace Hello { public class HelloWorld { public static void Main(string[] args) { string name = "C#"; //See if an argument was passed from the command line if (args.Length == 1) name = args[0]; Console.WriteLine("Hello, " + name + "!"); } } }</pre>	<pre>PROGRAM NAMESPACE Hello USING System MAP END Name STRING CODE Name = 'Clarion#' !See if an argument was passed from the command line IF COMMAND('1') <> '' Name = COMMAND('1') END Console.WriteLine('Hello, ' & Name & '!')</pre>	<pre>Imports System Namespace Hello Class HelloWorld Overloads Shared Sub Main(ByVal args() As String) Dim name As String = "VB.NET" 'See if an argument was passed from the command line If args.Length = 1 Then name = args(0) Console.WriteLine("Hello, " & name & "!") End Sub End Class End Namespace</pre>
Comments		
C#	Clarion#	VB.NET
<pre>//Single line /* Multiple lines */ ///<summary>XML comments on single line</summary> /** <summary> XML comments on multiple lines </summary> */</pre>	<pre>!Single line !~ Multiple lines ~! !!!<summary>XML comments on single line</summary> !!!<summary> XML comments on multiple lines !!!</summary></pre>	<pre>'Single line only REM Single line only '''<summary>XML comments</summary></pre>

Data Types

C#	Clarion#	VB.NET
<pre> // Value Types bool byte, sbyte char short, ushort, int, uint, long, ulong float, double decimal DateTime //not a built-in C# type // Reference Types object string // Initializing bool correct = true; byte b = 0x2A; //hex object person = null; string name = "Mike"; char grade = 'B'; DateTime today = DateTime.Parse("12/31/2007 12:15:00"); decimal amount = 35.99m; float gpa = 2.9f; double pi = 3.14159265; long lTotal = 123456L; short sTotal = 123; ushort usTotal = 123; uint uiTotal = 123; ulong ulTotal = 123; // Type Information int x; Console.WriteLine(x.GetType()); //Prints System.Int32 Console.WriteLine(typeof(int)); //Prints System.Int32 Console.WriteLine(x.GetType().Name); //Prints Int32 // Type Conversion float d = 3.5f; int i = (int)d; //set to 3 (truncates decimal) </pre>	<pre> ! Value Types BOOL BYTE, SBYTE CHAR, CSTRING, PSTRING, CLASTRING SHORT, USHORT, SIGNED, UNSIGNED, LONG, ULONG, CLALONG SREAL, REAL, BFLOAT4, BFLOAT8 DECIMAL, PDECIMAL, CLADecimal DATE, TIME, CLADATE, CLATIME ! Reference Types OBJECT STRING &BYTE !References to Value Types require & ! Initializing Correct BOOL(True) H BYTE(02Ah) !hex O BYTE(052o) !octal B BYTE(01101b) !binary Person OBJECT Name STRING Name = 'Mike' Grade CHAR('B') Today DATE Today = DATE(12,31,2007) Amount DECIMAL(35.99) GPA SREAL(2.9) Pi REAL(3.14159265) lTotal LONG(123456) sTotal SHORT(123) usTotal USHORT(123) uiTotal UNSIGNED(123) ulTotal ULONG(123) ! Type Information X SIGNED Console.WriteLine(X.GetType()) !Prints System.Int32 Console.WriteLine(TYPEOF(SIGNED)) !Prints System.Int32 Console.WriteLine(X.GetType().Name) !Prints Int32 ! Type Conversion D SREAL(3.5) I SIGNED I = D !Implicitly truncate to 3 I = INT(D) !Explicitly truncate to 3 I = D TRYAS SIGNED !Explicitly truncate to 3 (may generate exception) </pre>	<pre> ' Value Types Boolean Byte, SByte Char Short, UShort, Integer, UInteger, Long, ULong Single, Double Decimal Date ' Reference Types Object String ' Initializing Dim correct As Boolean = True Dim b As Byte = &H2A 'hex Dim o As Byte = &O52 'octal Dim person As Object = Nothing Dim name As String = "Mike" Dim grade As Char = "B"c Dim today As Date = #12/31/2007 12:15:00 PM# Dim amount As Decimal = 35.99@ Dim gpa As Single = 2.9! Dim pi As Double = 3.14159265 Dim lTotal As Long = 123456L Dim sTotal As Short = 123S Dim usTotal As UShort = 123US Dim uiTotal As UInteger = 123UI Dim ulTotal As ULong = 123UL ' Type Information Dim x As Integer Console.WriteLine(x.GetType()) 'Prints System.Int32 Console.WriteLine(GetType(Integer)) 'Prints System.Int32 Console.WriteLine(TypeName(x)) 'Prints Integer ' Type Conversion Dim d As Single = 3.5 Dim i As Integer = CType(d, Integer) 'set to 4 (Banker's rounding) i = CInt(d) 'same result as CType i = Int(d) 'set to 3 (Int function truncates the decimal) </pre>

Constants

C#	Clarion#	VB.NET
<pre>const int MAX_STUDENTS = 25; // Can set to a const or var; may be initialized in a constructor readonly float MIN_DIAMETER = 4.93f;</pre>	<pre>! CONST and READONLY unsupported. ! Use EQUATEs or regular data types instead. MAX_STUDENTS EQUATE(25) !Instead of CONST; will auto-convert MIN_DIAMETER SREAL(4.93) !READONLY is unsupported</pre>	<pre>Const MAX_STUDENTS As Integer = 25 'Can set to a const or var; may be initialized in a constructor ReadOnly MIN_DIAMETER As Single = 4.93</pre>

Enumerations

C#	Clarion#	VB.NET
<pre>enum Action {Start, Stop, Rewind, Forward}; enum Status { Flunk = 50, Pass = 70, Excel = 90 }; Action a = Action.Stop; if (a != Action.Start) Console.WriteLine(a + " is " + (int) a); //Prints "Stop is 1" Console.WriteLine((int) Status.Pass); //Prints 70 Console.WriteLine(Status.Pass); //Prints Pass</pre>	<pre>Action ENUM Start ITEM Stop ITEM Rewind ITEM Forward ITEM END Status ENUM Flunk(50) Pass (70) Excel(90) END A Action(Action.Stop) IF (A <> Action.Start) Console.WriteLine(A.ToString() & 'is ' & A) !Prints "Stop is 1" END Console.WriteLine(Status.Pass + 0) !Prints 70 Console.WriteLine(Status.Pass) !Prints Pass</pre>	<pre>Enum Action Start [Stop] 'Stop is a reserved word Rewind Forward End Enum Enum Status Flunk = 50 Pass = 70 Excel = 90 End Enum Dim a As Action = Action.Stop If a <> Action.Start Then _ Console.WriteLine(a.ToString & " is " & a) 'Prints "Stop is 1" Console.WriteLine(Status.Pass) 'Prints 70 Console.WriteLine(Status.Pass.ToString()) 'Prints Pass</pre>

Operators

C#	Clarion#	VB.NET
<pre>// Comparison == < > <= >= != // Arithmetic + - * / % //mod / //integer division if both operands are ints Math.Pow(x, y) //raise to a power // Assignment = += -= *= /= %= &= = ^= <<= >>= ++ -- // Bitwise & ^ ~ << >></pre>	<pre>! Comparison = < > <= >= ~= <> &= ! Arithmetic + - * / % !mod / !integer division ^ !raise to a power ! Assignment = += -= *= /= %= &= ! Bitwise BAND(val,mask) BOR(val,mask) BXOR(val,mask) BSHIFT(val,count)</pre>	<pre>' Comparison = < > <= >= <> ' Arithmetic + - * / Mod \ 'integer division ^ 'raise to a power ' Assignment = += -= *= /= \= ^= <<= >>= &= ' Bitwise And Or Xor Not << >></pre>

```
// Logical
&& || & | ^ !
```

// Note: && and || perform short-circuit logical evaluations

```
// String Concatenation
+
```

```
! Logical
AND OR XOR NOT
```

! Note: AND and OR perform short-circuit logical evaluations

```
! String Concatenation
&
```

```
' Logical
AndAlso OrElse And Or Xor Not
```

' Note: AndAlso and OrElse perform short-circuit logical evaluations

```
' String Concatenation
&
```

Choices

C#	Clarion#	VB.NET
<pre>greeting = age < 20 ? "What's up?" : "Hello"; if (age < 20) greeting = "What's up?"; else greeting = "Hello"; // Semi-colon ";" is used to terminate each statement, // so no line continuation character is necessary. // Multiple statements must be enclosed in {} if (x != 100 && y < 5) { x *= 5; y *= 2; } if (x > 5) x *= y; else if (x == 5) x += y; else if (x < 10) x -= y; else x /= y;</pre>	<pre>Greeting = CHOOSE(Age < 20, 'What''s up?', 'Hello') ! One line requires THEN (or ;) IF Age < 20 THEN Greeting = 'What''s up?' END IF Age < 20; Greeting = 'What''s up?'ELSE Greeting = 'Hello' END ! Use semi-colon (;) to put two commands on same line. ! A period (.) may replace END in single line constructs, ! but it is discouraged for multi-line constructs. IF X <> 100 AND Y < 5 THEN X *= 5; Y *= 2. !Period is OK here ! Multi-line is more readable (THEN is optional on multi line) IF X <> 100 AND Y < 5 THEN X *= 5 Y *= 2 END IF X <> 100 AND Y < 5 X *= 5 Y *= 2 . !Period is hard to see here ! To break up any long single line use (pipe) IF WhenYouHaveAReally < LongLine AND ItNeedsToBeBrokenInto2 > Lines UseThePipe(CharToBreakItUp) END IF X > 5 X *= Y ELSIF X = 5 X += Y ELSIF X < 10 X -= Y ELSE X /= Y END</pre>	<pre>greeting = IIf(age < 20, "What's up?", "Hello") ! One line doesn't require End If If age < 20 Then greeting = "What's up?" If age < 20 Then greeting = "What's up?" Else greeting = "Hello" ! Use colon (;) to put two commands on same line If x <> 100 AndAlso y < 5 Then x *= 5 : y *= 2 ! Multi-line is more readable If x <> 100 AndAlso y < 5 Then x *= 5 y *= 2 End If ! To break up any long single line use _ If whenYouHaveAReally < longLine AndAlso _ itNeedsToBeBrokenInto2 > Lines Then _ UseTheUnderscore(charToBreakItUp) If x > 5 Then x *= y ElseIf x = 5 Then x += y ElseIf x < 10 Then x -= y Else x /= y End If</pre>

```
// Every case must end with break or goto case
switch (color) { //Must be integer or string
  case "pink" :
  case "red" : r++; break;
  case "blue" : b++; break;
  case "green": g++; break;
  default: other++; break; //break necessary on default
}
```

```
CASE Color !Any data type or expression
OF 'pink' OROF 'red'
  R += 1
OF 'blue'
  B += 1
OF 'green'
  G += 1
ELSE
  Other += 1
END

CASE Value
OF 0.00 TO 9.99; RangeName = 'Ones'
OF 10.00 TO 99.99; RangeName = 'Tens'
OF 100.00 TO 999.99; RangeName = 'Hundreds'
!etc.
ELSE ; RangeName = 'Zillions'
END

EXECUTE Stage !Integer value or expression
Stage1 !expression equals 1
Stage2 !expression equals 2
Stage3 !expression equals 3
ELSE
StageOther !expression equals some other value
END
```

```
Select Case color 'Must be a primitive data type'
Case "pink", "red"
  r += 1
Case "blue"
  b += 1
Case "green"
  g += 1
Case Else
  other += 1
End Select
```

Loops

C#	Clarion#	VB.NET
<pre>// Pre-test Loops while (c < 10) c++; // no "until" keyword for (c = 2; c <= 10; c += 2) Console.WriteLine(c); // Post-test Loop do c++; while (c < 10); // Untested Loop for (;;) { //break logic inside }</pre>	<pre>! Pre-test Loops LOOP WHILE C < 10 C += 1 END ! Post-test Loops LOOP C += 1 WHILE c < 10 ! Untested Loops LOOP //Break logic inside END LOOP UNTIL C = 10 C += 1 END LOOP C = 2 TO 10 BY 2 Console.WriteLine(C) END LOOP C += 1 UNTIL C = 10 LOOP 3 TIMES Console.WriteLine END</pre>	<pre>' Pre-test Loops While c < 10 c += 1 End While Do While c < 10 c += 1 Loop ' Post-test Loops Do c += 1 Loop While c < 10 ' Untested Loop Do //break logic inside Loop Do Until c = 10 c += 1 Loop For c = 2 To 10 Step 2 Console.WriteLine(c) Next Do c += 1 Loop Until c = 10</pre>

```
// Array or collection looping
string[] names = {"Fred", "Sue", "Barney"};
foreach (string s in names)
    Console.WriteLine(s);
```

// Breaking out of loops

```
int i = 0;
while (true) {
    if (i == 5)
        break;
    i++;
}
```

// Continue to next iteration

```
for (i = 0; i < 5; i++) {
    if (i < 4)
        continue;
    Console.WriteLine(i); //Only prints 4
}
```

! Array or collection looping

```
Names STRING,DIM(3)
S    STRING
CODE
Names[0] = 'Fred'; Names[1] = 'Sue'; Names[2] = 'Barney'
FOREACH S IN Names
    Console.WriteLine(S)
END
```

! Breaking out of loops

```
I    SHORT(0)
CODE
LOOP
    IF I = 5 THEN BREAK.
    I += 1
END
```

! Continue to next iteration

```
LOOP I = 0 TO 4
    IF I < 4 THEN CYCLE.
    Console.WriteLine(I)
END
```

' Array or collection looping

```
Dim names As String() = {"Fred", "Sue", "Barney"}
For Each s As String In names
    Console.WriteLine(s)
Next
```

' Breaking out of loops

```
Dim i As Integer = 0
While (True)
    If (i = 5) Then Exit While
    i += 1
End While
```

' Continue to next iteration

```
For i = 0 To 4
    If i < 4 Then Continue For
    Console.WriteLine(i) 'Only prints 4
Next
```

Arrays

C#	Clarion#	VB.NET
<pre>int[] nums = {1, 2, 3}; for (int i = 0; i < nums.Length; i++) Console.WriteLine(nums[i]);</pre> <p>// 5 is the size of the array</p> <pre>string[] names = new string[5]; names[0] = "David"; names[5] = "Bobby"; //Throws System.IndexOutOfRangeException</pre> <p>// C# can't dynamically resize arrays, so copy into new array</p> <pre>string[] names2 = new string[7]; Array.Copy(names, names2, names.Length); //or names.CopyTo(names2, 0);</pre> <pre>float[,] twoD = new float[rows, cols]; twoD[2,0] = 4.5f;</pre>	<pre>Nums SIGNED,DIM(3) I SIGNED CODE Nums[0] = 1; Nums[1] = 2; Nums[2] = 3 LOOP I = 1 TO Nums.Length Console.WriteLine(Nums[I]) END</pre> <p>! 5 is the size of the array</p> <pre>Names STRING,DIM(5) CODE Names[0] = 'David' Names[5] = 'Bobby' !Caught by compiler I = 5 Names[I] = 'Bobby' !Throws System.IndexOutOfRangeException</pre> <p>! Clarion# can't dynamically resize arrays, so copy into new array</p> <pre>Names2 STRING[] CODE Names2 = NEW STRING[7] Array.Copy(Names, Names2, Names.Length) !or Names.CopyTo(Names2, 0)</pre> <pre>TwoD SREAL[,] CODE TwoD = NEW SREAL[Rows, Cols] TwoD[2,0] = 4.5</pre>	<pre>Dim nums() As Integer = {1, 2, 3} For i As Integer = 0 To nums.Length - 1 Console.WriteLine(nums(i)) Next</pre> <p>' 4 is the index of the last element, so it holds 5 elements</p> <pre>Dim names(4) As String names(0) = "David" names(5) = "Bobby" 'Throws System.IndexOutOfRangeException</pre> <p>' Resize the array, keeping existing values (Preserve is optional)</p> <pre>ReDim Preserve names(6)</pre> <pre>Dim twoD(rows-1, cols-1) As Single twoD(2, 0) = 4.5</pre>

<pre>// Jagged arrays int[][] jagged = new int[3][] { new int[5], new int[2], new int[3] }; jagged[0][4] = 5;</pre>	<pre>! Jagged arrays unsupported</pre>	<pre>' Jagged arrays Dim jagged()() As Integer = { _ New Integer(4) {}, New Integer(1) {}, New Integer(2) {} } jagged(0)(4) = 5</pre>
---	---	---

Functions		
------------------	--	--

C#	Clarion#	VB.NET
<pre>// Pass by value(in,default), reference(in/out), and reference(out) void TestFunc(int x, ref int y, out int z) { x++; y++; z = 5; } int a = 1, b = 1, c; //c doesn't need initializing TestFunc(a, ref b, out c); Console.WriteLine("{0} {1} {2}", a, b, c); //1 2 5 // Accept variable number of arguments int Sum(params int[] nums) { int sum = 0; foreach (int i in nums) sum += i; return sum; } int total = Sum(4, 3, 2, 1); //returns 10 /* C# doesn't support optional arguments/parameters. Just create two different versions of the same function. */ void SayHello(string name, string prefix) { Console.WriteLine("Greetings, " + prefix + " " + name); } void SayHello(string name) { SayHello(name, ""); } }</pre>	<pre>! Pass by value(in,default), reference(in/out), and reference(out) TestFunc PROCEDURE(SIGNED X, *SIGNED Y, *SIGNED Z) CODE X += 1 Y += 1 Z = 5 RETURN <i>!Optional, if not returning a value</i> A UNSIGNED(1) B UNSIGNED(1) C UNSIGNED <i>!C doesn't need initializing</i> CODE TestFunc(A, B, C) Console.WriteLine('{0} {1} {2}', A, B, C) <i>!1 2 5</i> ! Accept variable number of arguments Sum PROCEDURE(PARAMS UNSIGNED[] Nums),UNSIGNED Result UNSIGNED(0) I UNSIGNED CODE FOREACH I IN Nums Result += I END RETURN Result Total# = Sum(4, 3, 2, 1) <i>!returns 10</i> ! Optional parameters without default value <i>! (When omitted, value parameters default to 0 or an empty string)</i> <i>! (Use OMITTED to detect the omission)</i> SayHello PROCEDURE(STRING Name, <STRING Prefix>) ! Optional parameters with default value <i>! (Valid only on simple numeric types)</i> <i>! (OMITTED will not detect the omission — the default is passed)</i> SayHello PROCEDURE(STRING Name, BYTE Age = 20)</pre>	<pre>' Pass by value(in,default), reference(in/out), and reference(out) Sub TestFunc(ByVal x As Integer, ByRef y As Integer, _ ByRef z As Integer) x += 1 y += 1 z = 5 End Sub Dim a = 1, b = 1, c As Integer 'c set to zero by default TestFunc(a, b, c) Console.WriteLine("{0} {1} {2}", a, b, c) '1 2 5 ' Accept variable number of arguments Function Sum(ByVal ParamArray nums As Integer()) As Integer Sum = 0 For Each i As Integer In nums Sum += i Next End Function 'Or use Return statement like C# Dim total As Integer = Sum(4, 3, 2, 1) 'returns 10 ' Optional parameters must be listed last and have a default value Sub SayHello(ByVal name As String, Optional ByVal prefix As String = "") Console.WriteLine("Greetings, " & prefix & " " & name) End Sub SayHello("Strangelove", "Dr.") SayHello("Madonna")</pre>

Strings

C#	Clarion#	VB.NET
<pre> // Escape sequences \r //carriage-return \n //line-feed \t //tab \\ //backslash \" //quote // String concatenation string school = "Harding\t"; school = school + "University"; // school is "Harding(tab)University" // Chars char letter = school[0]; <i>//letter is H</i> letter = Convert.ToChar(65); <i>//letter is A</i> letter = (char)65; <i>//same thing</i> char[] word = school.ToCharArray(); <i>//word holds Harding</i> // String literal string msg = @"File is c:\temp\x.dat"; <i>// same as</i> string msg = "File is c:\\temp\\x.dat"; // String comparison string mascot = "Bisons"; if (mascot == "Bisons") <i>//true</i> if (mascot.Equals("Bisons")) <i>//true</i> if (mascot.ToUpper().Equals("BISONS")) <i>//true</i> if (mascot.CompareTo("Bisons") == 0) <i>//true</i> // Substring Console.WriteLine(mascot.Substring(2, 3)); <i>//Prints "son"</i> </pre>	<pre> ! Special Characters <13> <i>!carriage-return</i> <10> <i>!line-feed</i> <9> <i>!tab</i> <n> <i>!character with the ASCII value=n (see above)</i> << <i>!less-than</i> {{ <i>!left-curly-brace</i> '' <i>!single-quote</i> {n} <i>!Repeat previous character "n" times</i> ! String concatenation School STRING Univ CLASTRING('University') <i>!Clarion string class</i> CODE School = 'Harding<9>' School = School & Univ <i>!School is "Harding(tab)University"</i> ! Chars Letter CHAR Word CHAR[] CODE Letter = School[1] <i>!Letter is H</i> Letter = Convert.ToChar(65) <i>!Letter is A</i> Letter = '<65>' <i>!Same thing</i> Word = School.ToCharArray <i>!Word holds Harding</i> ! No string literal operator Msg STRING CODE Msg = 'File is c:\temp\x.dat' ! String comparison Mascot STRING CODE Mascot = 'Bisons' IF Mascot = 'Bisons' <i>!true</i> IF Mascot.Equals('Bisons') <i>!true</i> IF Mascot.ToUpper().Equals('BISONS') <i>!true</i> IF UPPER(Mascot) = 'BISONS' <i>!true</i> IF Mascot.CompareTo('Bisons') = 0 <i>!true</i> ! Substring Console.WriteLine(Mascot.Substring(2, 3)) <i>!Prints "son"</i> Console.WriteLine(SUB(Mascot, 3, 3)) <i>!Prints "son"</i> Console.WriteLine(Mascot[3 : 6]) <i>!Prints "son"</i> </pre>	<pre> ' Special Character Constants vbCrLf, vbCr, vbLf, vbNewLine vbNullString vbTab vbBack vbFormFeed vbVerticalTab "" ' String concatenation (use & or +) Dim school As String = "Harding" & vbTab school = school & "University" <i>'school is "Harding(tab)University"</i> ' Chars Dim letter As Char = school.Chars(0) <i>'letter is H</i> letter = Convert.ToChar(65) <i>'letter is A</i> letter = Chr(65) <i>'same thing</i> Dim word() As Char = school.ToCharArray() <i>'word holds Harding</i> ' No string literal operator Dim msg As String = "File is c:\temp\x.dat" ' String comparison Dim mascot As String = "Bisons" If (mascot = "Bisons") Then <i>'true</i> If (mascot.Equals("Bisons")) Then <i>'true</i> If (mascot.ToUpper().Equals("BISONS")) Then <i>'true</i> If (mascot.CompareTo("Bisons") = 0) Then <i>'true</i> ' Substring Console.WriteLine(mascot.Substring(2, 3)) <i>'Prints "son"</i> </pre>

```
// String matching
// No exact equivalent to Like - use regular expressions
```

```
using System.Text.RegularExpressions;
Regex r = new Regex(@"Jo[hH]. \d:*");
if (r.Match("John 3:16").Success) //true
```

```
// My birthday: Sep 3, 1964
DateTime dt = new DateTime(1964, 9, 3);
string s = "My birthday: " + dt.ToString("MMM dd, yyyy");
```

```
// Mutable string
System.Text.StringBuilder buffer =
    new System.Text.StringBuilder("two ");
buffer.Append("three ");
buffer.Insert(0, "one ");
buffer.Replace("two", "TWO");
Console.WriteLine(buffer); //Prints "one TWO three"
```

```
! String matching
! No exact equivalent to Like - use regular expressions or MATCH
```

```
USING System.Text.RegularExpressions
R Regex
A STRING
B STRING
C STRING
CODE
R = NEW Regex('Jo[hH]. \d:*')
IF R.Match('John 3:16').Success !true
A = 'Richard'
B = 'RICHARD'
C = 'R*'
IF MATCH(A,B,MATCH:Simple+Match:NoCase) !true: case insensitive
IF MATCH(A,B,MATCH:Soundex) !true: soundex
IF MATCH(A,C) !true: wildcard (default)
IF MATCH('Fireworks on the fourth', '{{4|four}th', |
    MATCH:Regular+Match:NoCase) !true: RegEx
IF MATCH('July 4th fireworks', '{{4|four}th', |
    MATCH:Regular+Match:NoCase) !true: RegEx
```

```
! My birthday: Sep 3, 1964
DT DateTime
S STRING
CODE
DT = NEW DateTime(1964, 9, 3)
S = 'My birthday: ' & DT.ToString('MMM dd, yyyy')
```

```
! Mutable string
Buffer System.Text.StringBuilder('two ')
CODE
Buffer.Append('three ')
Buffer.Insert(0, 'one ')
Buffer.Replace('two', 'TWO')
Console.WriteLine(Buffer) !Prints "one TWO three"
```

```
' String matching
If ("John 3:16" Like "Jo[hH]? #:*") Then 'true

Imports System.Text.RegularExpressions 'More powerful than Like
Dim r As New Regex("Jo[hH]. \d:*")
If (r.Match("John 3:16").Success) Then 'true
```

```
' My birthday: Sep 3, 1964
Dim dt As New DateTime(1964, 9, 3)
Dim s As String = "My birthday: " & dt.ToString("MMM dd, yyyy")
```

```
' Mutable string
Dim buffer As New System.Text.StringBuilder("two ")
buffer.Append("three ")
buffer.Insert(0, "one ")
buffer.Replace("two", "TWO")
Console.WriteLine(buffer) 'Prints "one TWO three"
```

Exception Handling

C#

Clarion#

VB.NET

```
// Throw an exception
Exception ex = new Exception("Something is really wrong.");
throw ex;
```

```
! Throw an exception
Ex Exception('Something is really wrong.')
CODE
THROW Ex
```

```
' Throw an exception
Dim ex As New Exception("Something is really wrong.")
Throw ex
```

<pre> // Catch an exception try { y = 0; x = 10 / y; } catch (Exception ex) { //Argument is optional, no "When" keyword Console.WriteLine(ex.Message); } finally { //Requires reference to the Microsoft.VisualBasic.dll //assembly (pre .NET Framework v2.0) Microsoft.VisualBasic.Interaction.Beep(); } </pre>	<pre> ! Catch an exception TRY y = 0; x = 10 / y; CATCH(Exception Ex) !Argument is optional, no "When" keyword Console.WriteLine(Ex.Message); FINALLY BEEP END </pre>	<pre> ' Catch an exception Try y = 0 x = 10 / y Catch ex As Exception When y = 0 'Argument and When is optional Console.WriteLine(ex.Message) Finally Beep() End Try ' Deprecated unstructured error handling On Error GoTo MyErrorHandler ... MyErrorHandler: Console.WriteLine(Err.Description) </pre>
--	---	---

Namespaces

C#	Clarion#	VB.NET
<pre> namespace Harding.Compsci.Graphics { ... } //or namespace Harding { namespace Compsci { namespace Graphics { ... } } } using Harding.Compsci.Graphics; </pre>	<pre> NAMESPACE Harding.Compsci.Graphics ! Progressive, nested namespaces unsupported USING Harding.Compsci.Graphics </pre>	<pre> Namespace Harding.Compsci.Graphics ... End Namespace 'or Namespace Harding Namespace Compsci Namespace Graphics ... End Namespace End Namespace End Namespace Imports Harding.Compsci.Graphics </pre>

Classes / Interfaces

C#	Clarion#	VB.NET
<pre> // Accessibility keywords public private internal protected protected internal static // Inheritance class FootballGame : Competition { ... } </pre>	<pre> ! Accessibility keywords PUBLIC PRIVATE INTERNAL PROTECTED PROTECTED INTERNAL STATIC ! Inheritance FootballGame CLASS(Competition) ... END </pre>	<pre> ' Accessibility keywords Public Private Friend Protected Protected Friend Shared ' Inheritance Class FootballGame Inherits Competition ... End Class </pre>

```
// Interface definition
interface IAlarmClock {
    ...
}

// Extending an interface
interface IAlarmClock : IClock {
    ...
}

// Interface implementation
class WristWatch : IAlarmClock, ITimer {
    ...
}
```

```
! Interface definition
IAlarmClock INTERFACE
    ...
    END

! Extending an interface
IAlarmClock INTERFACE(IClock)
    ...
    END

! Interface implementation
WristWatch CLASS, IMPLEMENTS(IAlarmClock), IMPLEMENTS(ITimer)
    ...
    END
```

```
' Interface definition
Interface IAlarmClock
    ...
End Interface

' Extending an interface
Interface IAlarmClock
    Inherits IClock
    ...
End Interface

' Interface implementation
Class WristWatch
    Implements IAlarmClock, ITimer
    ...
End Class
```

Constructors / Destructors

C#	Clarion#	VB.NET
<pre>class SuperHero { private int _powerLevel; public SuperHero() { _powerLevel = 0; } public SuperHero(int powerLevel) { this._powerLevel= powerLevel; } ~SuperHero() { //Destructor code to free unmanaged resources. //Implicitly creates a Finalize method } }</pre>	<pre>SuperHero CLASS _PowerLevel UNSIGNED,PRIVATE Construct PROCEDURE,PUBLIC Construct PROCEDURE(UNSIGNED PowerLevel),PUBLIC Destruct PROCEDURE END SuperHero.Construct PROCEDURE CODE SELF._PowerLevel = 0 SuperHero.Construct PROCEDURE(UNSIGNED PowerLevel) CODE SELF._PowerLevel = PowerLevel SuperHero.Destruct PROCEDURE CODE !Destructor code to free unmanaged resources.</pre>	<pre>Class SuperHero Private _powerLevel As Integer Public Sub New() _powerLevel = 0 End Sub Public Sub New(ByVal powerLevel As Integer) Me._powerLevel = powerLevel End Sub Protected Overrides Sub Finalize() 'Destructor code to free unmanaged resources MyBase.Finalize() End Sub End Class</pre>

Using Objects

C#	Clarion#	VB.NET
<pre>SuperHero hero = new SuperHero(); // No "With" construct hero.Name = "SpamMan"; hero.PowerLevel = 3;</pre>	<pre>Hero SuperHero Hero2 SuperHero Obj Object CODE Hero = NEW SuperHero ! No "With" construct hero.Name = 'SpamMan' hero.PowerLevel = 3</pre>	<pre>Dim hero As SuperHero = New SuperHero 'or Dim hero As New SuperHero With hero .Name = "SpamMan" .PowerLevel = 3 End With</pre>

<pre> hero.Defend("Laura Jones"); SuperHero.Rest(); //Calling static method SuperHero hero2 = hero; //Both reference the same object hero2.Name = "WormWoman"; Console.WriteLine(hero.Name); //Prints WormWoman hero = null ; //Free the object if (hero == null) hero = new SuperHero(); Object obj = new SuperHero(); if (obj is SuperHero) Console.WriteLine("Is a SuperHero object."); // Mark object for quick disposal using (StreamReader reader = File.OpenText("test.txt")) { string line; while ((line = reader.ReadLine()) != null) Console.WriteLine(line); } </pre>	<pre> Hero.Defend('Laura Jones') SuperHero.Rest() !Calling static method Hero2 = Hero !Both reference the same object Hero2.Name = 'WormWoman' Console.WriteLine(Hero.Name) !Prints "WormWoman" Hero = NULL !Free the object IF Hero = NULL Hero = NEW SuperHero END Obj = NEW SuperHero(); IF Obj IS SuperHero Console.WriteLine('Is a SuperHero object.') END ! No equivalent to USING(Resource) to mark for quick disposal </pre>	<pre> hero.Defend("Laura Jones") hero.Rest() 'Calling Shared method 'or SuperHero.Rest() Dim hero2 As SuperHero = hero 'Both reference the same object hero2.Name = "WormWoman" Console.WriteLine(hero.Name) 'Prints WormWoman hero = Nothing 'Free the object If hero Is Nothing Then _ hero = New SuperHero Dim obj As Object = New SuperHero If TypeOf obj Is SuperHero Then _ Console.WriteLine("Is a SuperHero object.") ' Mark object for quick disposal Using reader As StreamReader = File.OpenText("test.txt") Dim line As String = reader.ReadLine() While Not line Is Nothing Console.WriteLine(line) line = reader.ReadLine() End While End Using </pre>
--	--	--

Structs

C#	Clarion#	VB.NET
<pre> struct StudentRecord { public string name; public float gpa; public StudentRecord(string name, float gpa) { this.name = name; this.gpa = gpa; } } StudentRecord stu = new StudentRecord("Bob", 3.5f); StudentRecord stu2 = stu; stu2.name = "Sue"; Console.WriteLine(stu.name); //Prints Bob Console.WriteLine(stu2.name); //Prints Sue </pre>	<pre> StudentRecord STRUCT Name STRING,PUBLIC GPA SREAL,PUBLIC Construct PROCEDURE(STRING Name,SREAL GPA),PUBLIC INLINE CODE SELF.Name = Name SELF.GPA = GPA END END Stu StudentRecord('Bob', 3.5) Stu2 StudentRecord CODE Stu2.Name = 'Sue' Console.WriteLine(Stu.Name) //Prints Bob Console.WriteLine(Stu2.Name) //Prints Sue </pre>	<pre> Structure StudentRecord Public name As String Public gpa As Single Public Sub New(ByVal name As String, ByVal gpa As Single) Me.name = name Me.gpa = gpa End Sub End Structure Dim stu As StudentRecord = New StudentRecord("Bob", 3.5) Dim stu2 As StudentRecord = stu stu2.name = "Sue" Console.WriteLine(stu.name) //Prints Bob Console.WriteLine(stu2.name) //Prints Sue </pre>

Properties

C#	Clarion#	VB.NET
<pre>private int _size; public int Size { get { return _size; } set { //parameter is always "value" if (value < 0) _size = 0; else _size = value; } } // Usage foo.Size++;</pre>	<pre>_Size UNSIGNED,PRIVATE Size PROPERTY,UNSIGNED,PUBLIC INLINE GETTER CODE RETURN SELF._Size SETTER !parameter is always "Value" CODE IF Value < 0 SELF._Size = 0 ELSE SELF._Size = Value END END ! Rather than use INLINE, you may define Get_PropertyName ! and Set_PropertyName methods separately. ClassName.Get_Size PROCEDURE CODE RETURN SELF._Size ClassName.Set_Size PROCEDURE(UNSIGNED pValue) CODE SELF._Size = CHOOSE(pValue < 0, 0, pValue) ! Usage Foo.Size += 1</pre>	<pre>Private _size As Integer Public Property Size() As Integer Get Return _size End Get Set (ByVal Value As Integer) If Value < 0 Then _size = 0 Else _size = Value End If End Set End Property ' Usage foo.Size += 1</pre>

Delegates / Events

C#	Clarion#	VB.NET
<pre>delegate void MsgArrivedEventHandler(string message); event MsgArrivedEventHandler MsgArrivedEvent; // Events must use explicitly-defined delegates in C# MsgArrivedEvent += new MsgArrivedEventHandler(My_MsgArrivedEventCallback); MsgArrivedEvent("Test message"); //Throws exception if obj is null MsgArrivedEvent -= new MsgArrivedEventHandler(My_MsgArrivedEventCallback);</pre>	<pre>MAP MsgArrivedEventHandler PROCEDURE(STRING Message),DELEGATE END MsgArrivedEvent EVENT,MsgArrivedEventHandler ! Events must use explicitly-defined delegates in Clarion# MsgArrivedEvent += My_MsgArrivedEventCallback MsgArrivedEvent('Test message') MsgArrivedEvent -= My_MsgArrivedEventCallback</pre>	<pre>Delegate Sub MsgArrivedEventHandler(ByVal message As String) Event MsgArrivedEvent As MsgArrivedEventHandler ' or to define an event which declares a delegate implicitly Event MsgArrivedEvent(ByVal message As String) AddHandler MsgArrivedEvent, AddressOf My_MsgArrivedCallback ' Won't throw an exception if obj is Nothing RaiseEvent MsgArrivedEvent("Test message") RemoveHandler MsgArrivedEvent, AddressOf My_MsgArrivedCallback</pre>

<pre>using System; using System.Windows.Forms; public class MyClass { Button MyButton; public void Init() { MyButton = new Button(); MyButton.Click += new EventHandler(MyButton_Click); } private void MyButton_Click(object sender, EventArgs e) { MessageBox.Show("Button was clicked", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information); } }</pre>	<pre>USING System USING System.Windows.Forms MyForm CLASS,TYPE,NETCLASS,PARTIAL MyButton Button Init PROCEDURE MyButton_Click PROCEDURE(Object Sender, EventArgs E) END MyForm.Init PROCEDURE CODE MyButton = NEW Button SELF.MyButton.Click += SELF.MyButton_Click MyForm.MyButton_Click PROCEDURE(Object Sender, EventArgs E) CODE MessageBox.Show('Button was clicked', 'Info', MessageBoxButtons.OK, MessageBoxIcon.Information)</pre>	<pre>Imports System Imports System.Windows.Forms Public Class MyForm Dim WithEvents MyButton As Button Public Sub Init MyButton = New Button End Sub Private Sub MyButton_Click(ByVal sender As Object, _ ByVal e As EventArgs) Handles MyButton.Click MessageBox.Show(Me, "Button was clicked", "Info", _ MessageBoxButtons.OK, MessageBoxIcon.Information) End Sub End Class</pre>
--	---	---

Console I/O

C#	Clarion#	VB.NET
<pre>Console.Write("What's your name? "); string name = Console.ReadLine(); Console.Write("How old are you? "); int age = Convert.ToInt32(Console.ReadLine()); Console.WriteLine("{0} is {1} years old.", name, age); //or Console.WriteLine(name + " is " + age + " years old."); int c = Console.Read(); //Read single char Console.WriteLine(c); //Prints 65 if user enters "A"</pre>	<pre>Name STRING Age UNSIGNED C UNSIGNED CODE Console.Write('What's your name? ') Name = Console.ReadLine() Console.Write('How old are you? ') Age = Convert.ToInt32(Console.ReadLine()) Console.WriteLine("{0} is {1} years old.", Name, Age); !or Console.WriteLine(Name + 'is ' + age + 'years old.');</pre> <pre>C = Console.Read() !Read single char Console.WriteLine(C) !Prints 65 if user enters "A" 0</pre>	<pre>Console.Write("What's your name? ") Dim name As String = Console.ReadLine() Console.Write("How old are you? ") Dim age As Integer = Val(Console.ReadLine()) Console.WriteLine("{0} is {1} years old.", name, age) 'or Console.WriteLine(name & " is " & age & " years old.") Dim c As Integer c = Console.Read() 'Read single char Console.WriteLine(c) 'Prints 65 if user enters "A"</pre>

File I/O

C#	Clarion#	VB.NET
<pre>using System.IO; // Write out to text file StreamWriter writer = File.CreateText("c:\\myfile.txt"); writer.WriteLine("Out to file."); writer.Close();</pre>	<pre>USING System.IO ! Write out to text file Writer StreamWriter CODE Writer = File.CreateText('c:\\myfile.txt') Writer.WriteLine('Out to file.') Writer.Close()</pre>	<pre>Imports System.IO ' Write out to text file Dim writer As StreamWriter = File.CreateText("c:\\myfile.txt") writer.WriteLine("Out to file.") writer.Close()</pre>

```
// Read all lines from text file
StreamReader reader = File.OpenText("c:\\myfile.txt");
string line = reader.ReadLine();
while (line != null) {
    Console.WriteLine(line);
    line = reader.ReadLine();
}
reader.Close();
```

```
// Write out to binary file
string str = "Text data";
int num = 123;
BinaryWriter binWriter =
    new BinaryWriter(File.OpenWrite("c:\\myfile.dat"));
binWriter.Write(str);
binWriter.Write(num);
binWriter.Close();
```

```
// Read from binary file
BinaryReader binReader =
    new BinaryReader(File.OpenRead("c:\\myfile.dat"));
str = binReader.ReadString();
num = binReader.ReadInt32();
binReader.Close();
```

```
! Read all lines from text file
Reader StreamReader
Line STRING
CODE
Reader = File.OpenText('c:\\myfile.txt')
Line = Reader.ReadLine()
LOOP WHILE NOT Line &= NULL
    Console.WriteLine(Line)
    Line = Reader.ReadLine()
END
Reader.Close()
```

```
! Write out to binary file
Str STRING
Num SIGNED(123)
BinWriter BinaryWriter
CODE
Str = 'Text data'
BinWriter = NEW BinaryWriter(File.OpenWrite('c:\\myfile.dat'))
BinWriter.Write(Str)
BinWriter.Write(Num)
BinWriter.Close()
```

```
! Read from binary file
BinReader BinaryReader
CODE
BinReader = NEW BinaryReader(File.OpenRead('c:\\myfile.dat'))
Str = BinReader.ReadString()
Num = BinReader.ReadInt32()
BinReader.Close()
```

```
' Read all lines from text file
Dim reader As StreamReader = File.OpenText("c:\\myfile.txt")
Dim line As String = reader.ReadLine()
While Not line Is Nothing
    Console.WriteLine(line)
    line = reader.ReadLine()
End While
reader.Close()
```

```
' Write out to binary file
Dim str As String = "Text data"
Dim num As Integer = 123
Dim binWriter As New BinaryWriter(File.OpenWrite("c:\\myfile.dat"))
binWriter.Write(str)
binWriter.Write(num)
binWriter.Close()
```

```
' Read from binary file
Dim binReader As New BinaryReader(File.OpenRead("c:\\myfile.dat"))
str = binReader.ReadString()
num = binReader.ReadInt32()
binReader.Close()
```

Based upon a document by Frank McCown (www.harding.edu/fmccown/vbnet_csharp_comparison.html). Licensed under a Creative Commons License (creativecommons.org/licenses/by-sa/2.0).

Clarion# examples provided by Mike Hanson (www.boxsoft.net). Last updated April 17, 2008.