

VB.NET - DATA TYPES

http://www.tutorialspoint.com/vb.net/vb.net_data_types.htm

Copyright © tutorialspoint.com

Data types refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

Data Types Available in VB.Net

VB.Net provides a wide range of data types. The following table shows all the data types available:

Data Type	Storage Allocation	Value Range
Boolean	Depends on implementing platform	True or False
Byte	1 byte	0 through 255 <i>unsigned</i>
Char	2 bytes	0 through 65535 <i>unsigned</i>
Date	8 bytes	0:00:00 <i>midnight</i> on January 1, 0001 through 11:59:59 PM on December 31, 9999
Decimal	16 bytes	0 through +/-79,228,162,514,264,337,593,543,950,335 +/- 7.9...E + 28 with no decimal point; 0 through +/- 7.9228162514264337593543950335 with 28 places to the right of the decimal
Double	8 bytes	-1.79769313486231570E+308 through - 4.94065645841246544E-324, for negative values 4.94065645841246544E-324 through 1.79769313486231570E+308, for positive values
Integer	4 bytes	-2,147,483,648 through 2,147,483,647 <i>signed</i>
Long	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807 <i>signed</i>
Object	4 bytes on 32-bit platform 8 bytes on 64-bit platform	Any type can be stored in a variable of type Object
SByte	1 byte	-128 through 127 <i>signed</i>
Short	2 bytes	-32,768 through 32,767 <i>signed</i>
Single	4 bytes	-3.4028235E+38 through -1.401298E-45 for negative values; 1.401298E-45 through 3.4028235E+38 for positive values
String	Depends on implementing platform	0 to approximately 2 billion Unicode characters

UInteger	4 bytes	0 through 4,294,967,295 <i>unsigned</i>
ULong	8 bytes	0 through 18,446,744,073,709,551,615 <i>unsigned</i>
User-Defined	Depends on implementing platform	Each member of the structure has a range determined by its data type and independent of the ranges of the other members
UShort	2 bytes	0 through 65,535 <i>unsigned</i>

Example

The following example demonstrates use of some of the types:

```
Module DataTypes
    Sub Main()
        Dim b As Byte
        Dim n As Integer
        Dim si As Single
        Dim d As Double
        Dim da As Date
        Dim c As Char
        Dim s As String
        Dim bl As Boolean
        b = 1
        n = 1234567
        si = 0.12345678901234566
        d = 0.12345678901234566
        da = Today
        c = "U"c
        s = "Me"
        If ScriptEngine = "VB" Then
            bl = True
        Else
            bl = False
        End If
        If bl Then
            'the oath taking
            Console.Write(c & " and," & s & vbCrLf)
            Console.WriteLine("declaring on the day of: {0}", da)
            Console.WriteLine("We will learn VB.Net seriously")
            Console.WriteLine("Lets see what happens to the floating point variables:")
            Console.WriteLine("The Single: {0}, The Double: {1}", si, d)
        End If
        Console.ReadKey()
    End Sub
End Module
```

When the above code is compiled and executed, it produces the following result:

```
U and, Me
declaring on the day of: 12/4/2012 12:00:00 PM
We will learn VB.Net seriously
Lets see what happens to the floating point variables:
The Single:0.1234568, The Double: 0.123456789012346
```

The Type Conversion Functions in VB.Net

VB.Net provides the following in-line type conversion functions:

S.N Functions & Description

CBool*expression*

Converts the expression to Boolean data type.

2

CByte*expression*

Converts the expression to Byte data type.

3

CChar*expression*

Converts the expression to Char data type.

4

CDate*expression*

Converts the expression to Date data type

5

CDbl*expression*

Converts the expression to Double data type.

6

CDec*expression*

Converts the expression to Decimal data type.

7

CInt*expression*

Converts the expression to Integer data type.

8

CLng*expression*

Converts the expression to Long data type.

9

CObj*expression*

Converts the expression to Object type.

10

SByte*expression*

Converts the expression to SByte data type.

11

Short*expression*

Converts the expression to Short data type.

12

Single*expression*

Converts the expression to Single data type.

- 13 **CStr***expression*
Converts the expression to String data type.
- 14 **CUInt***expression*
Converts the expression to UInt data type.
- 15 **CULng***expression*
Converts the expression to ULng data type.
- 16 **CUShort***expression*
Converts the expression to UShort data type.

Example:

The following example demonstrates some of these functions:

```
Module DataTypes
  Sub Main()
    Dim n As Integer
    Dim da As Date
    Dim bl As Boolean = True
    n = 1234567
    da = Today
    Console.WriteLine(bl)
    Console.WriteLine(CSByte(bl))
    Console.WriteLine(CStr(bl))
    Console.WriteLine(CStr(da))
    Console.WriteLine(CChar(CChar(CStr(n))))
    Console.WriteLine(CChar(CStr(da)))
    Console.ReadKey()
  End Sub
End Module
```

When the above code is compiled and executed, it produces the following result:

```
True
-1
True
12/4/2012
1
1
```

Loading [MathJax]/jax/output/HTML-CSS/jax.js