Visual Basic .NET

Visual Basic .NET (**VB.NET**) is a multi-paradigm, object-oriented programming language, implemented on the .NET Framework. Microsoft launched VB.NET in 2002 as the successor to its original Visual Basic language. Although the ".NET" portion of the name was dropped in 2005, this article uses "Visual Basic [.NET]" to refer to all Visual Basic languages released since 2002, in order to distinguish between them and the classic Visual Basic. Along with Visual C#, it is one of the two main languages targeting the .NET framework.

Microsoft's integrated development environment (IDE) for developing in Visual Basic .NET language is Visual Studio. Most Visual Studio editions are commercial; the only exceptions are Visual Studio Express and Visual Studio Community, which are freeware. In addition, the .NET Framework SDK includes a freeware command-line compiler called vbc.exe. Mono also includes a command-line VB.NET compiler.

Contents

Visual Basic .NET

Paradigm	Structured, imperative, object-oriented, declarative, generic, reflective and event-driven
Designed by	Microsoft
Developer	Microsoft
First appeared	2001
Stable release	2019 (16.0) / July 24, 2019
Typing discipline	Static, both strong and weak,[1] both safe and unsafe,[1] nominative
Platform	.NET Framework, Mono, .NET Core 3[2]
OS	Chiefly Windows Also on Android, BSD, iOS, Linux, macOS, Solaris and Unix
License	Roslyn compiler: Apache License 2.0[3]
Filename extensions	`.vb`
Website	docs.microsoft.com/dotnet

	/visual-basic/ (https://docs.microsoft.com/dotnet/visual-basic/)
Major implementations	
.NET Framework SDK, Roslyn Compiler and Mono	
Dialects	
Microsoft Visual Basic	
Influenced by	
Visual Basic	
Influenced	
Small Basic	

Syntax

VB.NET uses statements to specify actions. The most common statement is an expression statement, consisting of an expression to be evaluated, on a single line. As part of that evaluation, functions or subroutines may be called and variables may be assigned new values. To modify the normal sequential execution of statements, VB.NET provides several control-flow statements identified by reserved keywords. Structured programming is supported by several constructs including two conditional execution constructs (`If` … `Then` … `Else` … `End If` and `Select Case` ... `Case` ... `End Select`) and three iterative execution (loop) constructs (`Do` … `Loop`, `For` … `To`, and `For Each`) . The `For` … `To` statement has separate initialisation and testing sections, both of which must be present. (See examples below.) The `For Each` statement steps through each value in a list.

In addition, in Visual Basic .NET:

- There is no unified way of defining blocks of statements. Instead, certain keywords, such as "If … Then" or "Sub" are interpreted as starters of sub-blocks of code and have matching termination keywords such as "End If" or "End Sub".
- Statements are terminated either with a colon (":") or with the end of line. Multiple-line statements in Visual Basic .NET are enabled with " _" at the end of each such line. The need for the underscore continuation character was largely removed in version 10 and later versions.[4]
- The equals sign ("=") is used in both assigning values to variables and in comparison.
- Round brackets (parentheses) are used with arrays, both to declare them and to get a value at a given index in one of them. Visual Basic .NET uses round brackets to define the parameters of subroutines or functions.
- A single quotation mark (') or the keyword `REM`, placed at the beginning of a line or after any number of space or tab characters at the beginning of a line, or after other code on a line, indicates that the (remainder of the) line is a comment.

Simple example

The following is a very simple VB.NET program, a version of the classic "Hello, World!" example created as a console application:

```
Module Module1

    Sub Main()
        ' The classic "Hello, World!" demonstration program
        Console.WriteLine("Hello, World!")
    End Sub
```

```
End Module
```

It prints "*Hello, World!*" on a command-line window. Each line serves a specific purpose, as follows:

```
Module Module1
```

This is a module definition. Modules are a division of code, which can contain any kind of object, like constants or variables, functions or methods, or classes, but can't be instantiated as objects like classes and cannot inherit from other modules. Modules serve as containers of code that can be referenced from other parts of a program.[5]
It is common practice for a module and the code file which contains it to have the same name. However, this is not required, as a single code file may contain more than one module and/or class.

```
Sub Main()
```

This line defines a subroutine called "Main". "Main" is the entry point, where the program begins execution.[6]

```
Console.WriteLine("Hello, world!")
```

This line performs the actual task of writing the output. *Console* is a system object, representing a command-line interface (also known as a "console") and granting programmatic access to the operating system's standard streams. The program calls the *Console* method *WriteLine*, which causes the string passed to it to be displayed on the console.

Instead of Console.WriteLine, one could use MsgBox, which prints the message in a dialog box instead of a command-line window.[7]

Complex example

This piece of code outputs Floyd's Triangle to the console:

```
Imports System.Console

Module Program

    Sub Main()
        Dim rows As Integer

        ' Input validation.
        Do Until Integer.TryParse(ReadLine("Enter a value for how many rows to be displayed: " &
vbcrlf), rows) AndAlso rows >= 1
            WriteLine("Allowed range is 1 and {0}", Integer.MaxValue)
        Loop

        ' Output of Floyd's Triangle
        Dim current As Integer = 1
        Dim row As Integer
        Dim column As Integer
        For row = 1 To rows
            For column = 1 To row
                Write("{0,-2} ", current)
                current += 1
            Next

            WriteLine()
        Next
    End Sub

    ''' <summary>
```

```
    ''' Like Console.ReadLine but takes a prompt string.
    ''' </summary>
    Function ReadLine(Optional prompt As String = Nothing) As String
        If prompt IsNot Nothing Then
            Write(prompt)
        End If

        Return Console.ReadLine()
    End Function

End Module
```

Comparison with the classic Visual Basic

Whether Visual Basic .NET should be considered as just another version of Visual Basic or a completely different language is a topic of debate. There are new additions to support new features, such as structured exception handling and short-circuited expressions. Also, two important data-type changes occurred with the move to VB.NET: compared to Visual Basic 6, the `Integer` data type has been doubled in length from 16 bits to 32 bits, and the `Long` data type has been doubled in length from 32 bits to 64 bits. This is true for all versions of VB.NET. A 16-bit integer in all versions of VB.NET is now known as a `Short`. Similarly, the Windows Forms editor is very similar in style and function to the Visual Basic form editor.

The things that *have* changed significantly are the semantics—from those of an object-based programming language running on a deterministic, reference-counted engine based on COM to a fully object-oriented language backed by the .NET Framework, which consists of a combination of the Common Language Runtime (a virtual machine using generational garbage collection and a just-in-time compilation engine) and a far larger class library. The increased breadth of the latter is also a problem that VB developers have to deal with when coming to the language, although this is somewhat addressed by the *My* feature in Visual Studio 2005.

The changes have altered many underlying assumptions about the "right" thing to do with respect to performance and maintainability. Some functions and libraries no longer exist; others are available, but not as efficient as the "native" .NET alternatives. Even if they compile, most converted Visual Basic 6 applications will require some level of refactoring to take full advantage of the new language. Documentation is available to cover changes in the syntax, debugging applications, deployment and terminology.[8]

Comparative examples

The following simple examples compare VB and VB.NET syntax. They assume that the developer has created a form, placed a button on it and has associated the subroutines demonstrated in each example with the click event handler of the mentioned button. Each example creates a "Hello, World" message box after the button on the form is clicked.

Visual Basic 6:

```
Private Sub Command1_Click()
    MsgBox "Hello, World"
End Sub
```

VB.NET (MsgBox or MessageBox class can be used):

```
Private Sub Button1_Click(sender As object, e As EventArgs) Handles Button1.Click
    MsgBox("Hello, World")
End Sub
```

- Both Visual Basic 6 and Visual Basic .NET automatically generate the `Sub` and `End Sub` statements when the corresponding button is double-clicked in design view. Visual Basic .NET will also generate the necessary `Class` and `End Class` statements. The developer need only add the statement to display the "Hello, World" message box.
- All procedure calls must be made with parentheses in VB.NET, whereas in Visual Basic 6 there were different conventions for functions (parentheses required) and subs (no parentheses allowed, unless called using the keyword `Call`).
- The names `Command1` and `Button1` are not obligatory. However, these are default names for a command button in Visual Basic 6 and VB.NET respectively.
- In VB.NET, the `Handles` keyword is used to make the sub `Button1_Click` a handler for the `Click` event of the object `Button1`. In Visual Basic 6, event handler subs must have a specific name consisting of the object's name ("Command1"), an underscore ("_"), and the event's name ("Click", hence "Command1_Click").
- There is a function called `MessageBox.Show` in the `Microsoft.VisualBasic` namespace which can be used (instead of `MsgBox`) similarly to the corresponding function in Visual Basic 6. There is a controversy[9] about which function to use as a best practice (not only restricted to showing message boxes but also regarding other features of the `Microsoft.VisualBasic` namespace). Some programmers prefer to do things "the .NET way", since the Framework classes have more features and are less language-specific. Others argue that using language-specific features makes code more readable (for example, using `int` (C#) or `Integer` (VB.NET) instead of `System.Int32`).
- In Visual Basic 2008, the inclusion of `ByVal sender as Object, ByVal e as EventArgs` has become optional.

The following example demonstrates a difference between Visual Basic 6 and VB.NET. Both examples close the active window.

Visual Basic 6:

```
Sub cmdClose_Click()
    Unload Me
End Sub
```

VB.NET:

```
Sub btnClose_Click(sender As Object, e As EventArgs) Handles btnClose.Click
    Close()
End Sub
```

The 'cmd' prefix is replaced by the 'btn' prefix, conforming to the new convention previously mentioned.

Visual Basic 6 did not provide common operator shortcuts. The following are equivalent:

Visual Basic 6:

```
Sub Timer1_Timer()
    'Reduces Form Height by one pixel per tick
    Me.Height = Me.Height - 1
End Sub
```

VB.NET:

```
Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    Me.Height -= 1
End Sub
```

Comparison with C#

C# and Visual Basic .NET are Microsoft's first languages made to program on the .NET Framework (later adding F# and more; others have also added languages). Though C# and VB.NET are syntactically different, that is where the differences mostly end. Microsoft developed both of these languages to be part of the same .NET Framework development platform. They are both developed, managed, and supported by the same language development team at Microsoft.[10] They compile to the same intermediate language (IL), which runs against the same .NET Framework runtime libraries.[11] Although there are some differences in the programming constructs, their differences are primarily syntactic and, assuming one avoids the Visual Basic "Compatibility" libraries provided by Microsoft to aid conversion from Visual Basic 6, almost every feature in VB has an equivalent feature in C# and vice versa. Lastly, both languages reference the same Base Classes of the .NET Framework to extend their functionality. As a result, with few exceptions, a program written in either language can be run through a simple syntax converter to translate to the other. There are many open source and commercially available products for this task.

Examples

Hello World!

Windows Form Application

Requires a button called Button1.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        MsgBox("Hello world!", MsgBoxStyle.Information, "Hello world!") ' Show a message that
says "Hello world!".
    End Sub
End Class
```

Console Application

```
Module Module1

    Sub Main()
        Console.WriteLine("Hello world!") ' Write in the
console "Hello world!" and start a new line.
        Console.ReadKey() ' The user must press any key
before the application ends.
    End Sub
End Module
```

Hello world! window

Speaking

Windows Form Application

Requires a TextBox titled 'TextBox1' and a button called Button1.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        CreateObject("Sapi.Spvoice").Speak(TextBox1.Text)
    End Sub
End Class
```

Console Application

```
Module Module1
    Private Voice = CreateObject("Sapi.Spvoice")
    Private Text As String

    Sub Main()
        Console.Write("Enter the text to speak: ") ' Say "Enter the text to speak: "
        Text = Console.ReadLine() ' The user must enter the text to speak.
        Voice.Speak(Text) ' Speak the text the user has entered.
    End Sub
End Module
```

Version history

Succeeding the classic Visual Basic version 6.0, the first version of Visual Basic .NET debuted in 2002. As of 2020, ten versions of Visual Basic .NET are released.

2002 (VB 7.0)

The first version, Visual Basic .NET, relies on .NET Framework 1.0. The most important feature is managed code, which contrasts with the classic Visual Basic.

2003 (VB 7.1)

Visual Basic .NET 2003 was released with .NET Framework 1.1. New features included support for the .NET Compact Framework and a better VB upgrade wizard. Improvements were also made to the performance and reliability of .NET IDE (particularly the background compiler (http://msdn.microsoft.com/msdnmag/issues/05/06/AdvancedBasics/default.aspx)) and runtime. In addition, Visual Basic .NET 2003 was available in the Visual Studio.NET Academic Edition, distributed to a certain number of scholars from each country without cost.

2005 (VB 8.0)

After Visual Basic .NET 2003, Microsoft dropped ".NET" from the name of the product, calling the next version Visual Basic 2005.

For this release, Microsoft added many features intended to reinforce Visual Basic .NET's focus as a rapid application development platform and further differentiate it from C#., including:

- *Edit and Continue* feature
- Design-time expression evaluation
- A pseudo-namespace called "My", which provides:[12][13]

    - Easy access to certain areas of the .NET Framework that otherwise require significant code to access like using `My.Form2.Text = " MainForm "` rather than `System.WindowsApplication1.Forms.Form2.text = " MainForm "`
    - Dynamically generated classes (e.g. *My.Forms*)
- Improved VB-to-VB.NET converter[14]
- A "using" keyword, simplifying the use of objects that require the Dispose pattern to free resources
- *Just My Code* feature, which hides (steps over) boilerplate code written by the Visual Studio .NET IDE and system library code during debugging
- Data Source binding, easing database client/server development

To bridge the gaps between itself and other .NET languages, this version added:

- Generics[15]
- Partial classes, a method of defining some parts of a class in one file and then adding more definitions later; particularly useful for integrating user code with auto-generated code
- Operator overloading and nullable types[16]
- Support for unsigned integer data types commonly used in other languages

Visual Basic 2005 introduced the `IsNot` operator that makes `'If X IsNot Y'` equivalent to `'If Not X Is Y'`. It gained notoriety[17] when it was found to be the subject of a Microsoft patent application.[18][19]

2008 (VB 9.0)

Visual Basic 9.0 was released along with .NET Framework 3.5 on November 19, 2007.

For this release, Microsoft added many features, including:

- A true conditional operator, "If(condition as boolean, truepart, falsepart)", to replace the "IIf" function.
- Anonymous types
- Support for LINQ
- Lambda expressions
- XML Literals
- Type Inference
- Extension methods

2010 (VB 10.0)

In April 2010, Microsoft released Visual Basic 2010. Microsoft had planned to use Dynamic Language Runtime (DLR) for that release[20] but shifted to a co-evolution strategy between Visual Basic and sister language C# to bring both languages into closer parity with one another. Visual Basic's innate ability to interact dynamically with CLR and COM objects has been enhanced to work with dynamic languages built on the DLR such as IronPython and IronRuby.[21] The Visual Basic compiler was improved to infer line continuation in a set of common contexts, in many cases removing the need for the " _" line continuation characters. Also, existing support of inline Functions was complemented with support for inline Subs as well as multi-line versions of both Sub and Function lambdas.[22]

2012 (VB 11.0)

Visual Basic 2012 was released along .NET Framework 4.5. Major features introduced in this version include:

- Asynchronous programming with "async" and "await" statements
- Iterators
- Call hierarchy
- Caller information
- "Global" keyword in "namespace" statements

2013 (VB 12.0)

Visual Basic 2013 was released along .NET Framework 4.5.1 with Visual Studio 2013. Can also build .NET Framework 4.5.2 applications by installing Developer Pack.[23]

2015 (VB 14.0)

Visual Basic 2015 (code named VB "14.0") has been released with Visual Studio 2015. Language features include a new "?." operator to perform inline null checks, and a new string interpolation feature is included to format strings inline.[24]

2017 (VB 15.x)

Visual Basic 2017 (code named VB "15.0") has been released with Visual Studio 2017. Extends support for new Visual Basic 15 language features with revision 2017, 15.3, 15.5, 15.8. Introduces new refactorings that allow organizing source code with one action.[25][26]

2019 (VB 16.0)

Visual Basic 2019 (code named VB "16.0") has been released with Visual Studio 2019. [27] The first version of Visual Basic focused on .NET Core.[28]

Cross-platform and open-source development

The official VB.NET compiler is written in VB.NET and is available on GitHub as a part of the .NET Compiler platform.[29] The creation of open-source tools for VB.NET development has been slow compared to C#, although the Mono development platform provides an implementation of VB.NET-specific libraries and a VB.NET 8.0 compatible compiler written in VB.NET,[30] as well as standard framework libraries such as Windows Forms GUI library.

SharpDevelop and MonoDevelop are open-source alternative IDEs. The Gambas environment is also similar but distinct from Visual Basic.

See also

- Microsoft Visual Studio Express
- List of .NET libraries and frameworks

- Comparison of C# and Visual Basic .NET
- Visual Basic for Applications
- Microsoft Small Basic
- Comparison of programming languages

References

1. "Option Explicit and Option Strict in Visual Basic .NET and in Visual Basic" (https://support.microsoft.com/en-us/kb/311329). *Support*. Microsoft. March 19, 2008. Retrieved August 22, 2013.
2. Dollard, Kathleen. "Visual Basic in .NET Core 3.0" (https://blogs.msdn.microsoft.com/vbteam/2018/11/12/visual-basic-in-net-core-3-0/). *blogs.msdn.microsoft.com*.
3. https://github.com/dotnet/roslyn/blob/1ff27b046b5c03abb38bfeda44eb82da0b8df9de/License.tx
4. "New Features in Visual Basic 10" (https://msdn.microsoft.com/en-us/library/ff637436.aspx).
5. "Module Statement" (http://msdn.microsoft.com/en-us/library/aaxss7da(VS.80).aspx). MSDN – Developer Center. Retrieved January 20, 2010.
6. "Main Procedure in Visual Basic" (http://msdn.microsoft.com/en-us/library/ms235406(VS.80).aspx). MSDN – Developer Center. Retrieved January 20, 2010.
7. "Visual Basic Version of Hello, World" (http://msdn.microsoft.com/en-us/library/3cf7t4xt(VS.80).aspx). MSDN – Developer Center. Retrieved January 20, 2010.
8. "Microsoft Visual Basic 6.0 Migration Resource Center" (http://msdn.microsoft.com/en-us/vstudio/ms788233). *MSDN*. Microsoft. Retrieved November 9, 2014.
9. "Visual Studio 2003 Retired Technical documentation" (https://www.microsoft.com/en-us/download/details.aspx?id=55979). *Microsoft Download Center*.
10. Krill, Paul (February 27, 2009). "Microsoft converging programming languages | Developer World" (https://archive.is/20130126074556/http://www.infoworld.com/article/09/02/27/Microsoft_converging_programming_languages_1.html?R=printThis&A=/article/09/02/27/Microsoft_converging_programming_languages_1.html). InfoWorld. Archived from the original (http://www.infoworld.com/article/09/02/27/Microsoft_converging_programming_languages_1.html?R=printThis&A=/article/09/02/27/Microsoft_converging_programming_languages_1.html) on January 26, 2013. Retrieved August 18, 2013.
11. "Microsoft Intermediate Language" (http://www.dotnet-guide.com/msintermediate.html). Dotnet-guide.com. Retrieved August 18, 2013.
12. Mackenzie, Duncan (2006). "Navigate The .NET Framework And Your Projects With The My Namespace" (http://msdn.microsoft.com/en-us/magazine/cc163680.aspx). *MSDN Magazine Visual Studio 2005 Guided Tour 2006*. Microsoft.
13. Whitney, Tyler (November 2005). "My.Internals: Examining the Visual Basic My Feature" (http://msdn.microsoft.com/en-us/library/ms379610.aspx). *MSDN*. Microsoft.
14. "What's New with the Visual Basic Upgrade Wizard in Visual Basic 2005" (http://msdn2.microsoft.com/en-us/library/ms379614.aspx). *msdn2.microsoft.com*.
15. "Defining and Using Generics in Visual Basic 2005" (http://msdn2.microsoft.com/en-us/library/ms379608.aspx). *msdn2.microsoft.com*.
16. "Operator Overloading in Visual Basic 2005" (http://msdn2.microsoft.com/en-us/library/ms379613.aspx). *msdn2.microsoft.com*.
17. Sherriff, Lucy (February 22, 2005). "Real Software slams MS IsNot patent application" (https://www.theregister.co.uk/2005/02/22/real_slams_ms_patent/). The Register. Retrieved April 6, 2009.
18. Taft, Darryl K. (February 21, 2005). "Real Software Slams Microsofts Patent Effort" (http://www.eweek.com/article2/0,1759,1766949,00.asp). eWeek. Retrieved April 6, 2009.

19. Vick, Paul A. Jr.; Barsan, Costica Corneliu; Silver, Amanda K. (May 14, 2003). "United States Patent Application: 20040230959" (http://appft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.html&r=1&f=G&l=50&s1=%2220040230959%22.PGNR.&OS=DN/20040230959&RS=DN/20040230959). *Patent Application Full Text and Image Database*. US Patent & Trademark Office. Retrieved April 6, 2009.
20. "What the heck is "VBx"?" (http://www.panopticoncentral.net/archive/2007/05/01/20383.aspx). May 1, 2007. Retrieved August 12, 2009. "*With the new DLR, we have support for IronPython, IronRuby, Javascript, and the new dynamic VBx compile*"
21. "What is New in Visual Basic 2010" (http://msdn.microsoft.com/en-us/library/we86c8x2%28VS.100%29.aspx). Microsoft. 2009. Retrieved August 12, 2009. "*Visual Basic binds to objects from dynamic languages such as IronPython and IronRuby*"
22. "What's New in Visual Basic 2010" (http://msdn.microsoft.com/en-us/vbasic/ee336123.aspx). Microsoft. 2010. Retrieved August 1, 2010.
23. Download Microsoft .NET Framework 4.5.2 Developer Pack for Windows Vista SP2, Windows 7 SP1, Windows 8, Windows 8.1, Windows Server 2008 SP2 Windows Server 2008 R2 SP1, Windows Server 2012 and Windows Server 2012 R2 from Official Microsoft Download Center (http://www.microsoft.com/en-us/download/details.aspx?id=42637)
24. "New Language Features in Visual Basic 14" (http://blogs.msdn.com/b/vbteam/archive/2014/12/09/new-language-features-in-visual-basic-14.aspx). *msdn.com*.
25. reshmim. "Visual Studio 2017 Release Notes" (https://www.visualstudio.com/en-us/news/releasenotes/vs2017-relnotes). *www.visualstudio.com*.
26. reshmim. "What's new for Visual Basic 2017,15.3,15.5,15.8" (https://docs.microsoft.com/en-us/dotnet/visual-basic/getting-started/whats-new#visual-basic-158). *www.visualstudio.com*.
27. reshmim. "Visual Studio 2019 Release Notes" (https://docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes). *www.visualstudio.com*.
28. reshmim. "What's new for Visual Basic 16.0" (https://docs.microsoft.com/en-us/dotnet/visual-basic/getting-started/whats-new#visual-basic-160). *www.visualstudio.com*.
29. *Roslyn* (https://github.com/dotnet/roslyn), .NET Foundation, April 13, 2019, retrieved April 14, 2019
30. "Redirecting…" (http://www.mono-project.com/VisualBasic.NET_support). *www.mono-project.com*.

Further reading

1. "Visual Basic Language Specification 8.0" (http://www.microsoft.com/downloads/en/details.aspx?FamilyId=6D50D709-EAA4-44D7-8AF3-E14280403E6E&displaylang=en). Microsoft Corporation. November 15, 2005. Retrieved December 10, 2010.
2. "Visual Basic Language Specification 9.0" (http://www.microsoft.com/download/en/details.aspx?id=995). Microsoft Corporation. December 19, 2007. Retrieved September 28, 2011.
3. "Visual Basic Language Specification 11.0" (http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=15039). Microsoft Corporation. June 7, 2013. Retrieved September 22, 2013.

External links

- Official website (https://docs.microsoft.com/dotnet/visual-basic/)
- The Visual Basic Team Blog (http://blogs.msdn.com/b/vbteam/)