Learn VB.NET in 1 Day

By Krishna Rungta

Table Of Content

Chapter 1: What is VB.Net? Introduction, History, Features, Advantages, Disadvantages

1. What is VB.Net?
2. History of VB.NET
3. VB.NET Features
4. Advantages of VB.NET
5. Disadvantages of VB.NET

Chapter 2: VB.Net Program Structure, Module, Classes: Hello World Example

1. Modules
2. Hello World Program
3. Classes
4. Structures
5. Using Microsoft Visual Studio IDE

Chapter 3: VB.Net Data Types and Variable Declaration with DIM

1. What are Data Types?
2. Type Conversion Functions
3. Variable Declaration
4. Variable Initialization
5. Accepting User Values
6. Lvalues and Rvalues

Chapter 4: VB.Net Operators: Arithmetic, Comparison, Logical with EXAMPLES

1. What is an operator?
2. Arithmetic Operators
3. Comparison Operators
4. Logical/Bitwise Operators
5. Bit Shift Operators
6. Assignment Operators
7. Miscellaneous Operators

Chapter 5: VB.Net Arrays: String, Dynamic with EXAMPLES

1. What is an Array?
2. How to Declare and Initialize an Array
3. Fixed-Size Arrays
4. Dynamic Arrays
5. Retrieving the Contents of an Array
6. Adding New Elements to an Array
7. Deleting an Array
8. The Split Function
9. The Join Function

Chapter 6: VB.NET Substring Method with EXAMPLE

1. What is a SubString?
2. Syntax of Substring
3. Examples
4. One Argument
5. Middle Characters
6. One Char

Chapter 7: For Each...Next, Exit, Continue Statement in VB.net with EXAMPLE

1. What is For Each?
2. Syntax of For Each… Next
3. Examples
4. Nested For Loops
5. Exit For and Continue For

Chapter 8: Select...Case Statement in VB.Net with Example

1. What is a Select Case?
2. Syntax of Select Case
3. Examples
4. Case Sensitive: ToLower, ToUpper

Chapter 9: Try...Catch...Finally, Throws, User Defined exception in VB.Net

1. What is an Exception?
2. What is Exception Handling?
3. Syntax of Try/Catch
4. Example to Handle Exceptions
5. User-Defined Exceptions
6. Throwing Objects

Chapter 10: VB.Net ComboBox Control with EXAMPLE

1. What is Combobox Control?
2. Creating a Combobox
3. Adding Items to Combobox

4. Selecting Combobox Items
5. Retrieving Combobox Values
6. Removing Combobox Items
7. Binding DataSource
8. SelectedIndexChanged event

Chapter 11: VB.Net TEXTBOX Control Tutorial: Properties with Example

1. What is TextBox Control?
2. TextBox Properties
3. Textbox Events
4. How to Create a TextBox
5. Password character
6. Newline in TextBox
7. Retrieving Integer Values
8. ReadOnly TextBox
9. max length

Chapter 1: What is VB.Net? Introduction, History, Features, Advantages, Disadvantages

What is VB.Net?

VB.NET stands for Visual Basic.NET, and it is a computer programming language developed by Microsoft. It was first released in 2002 to replace Visual Basic 6. VB.NET is an object-oriented programming language. This means that it supports the features of object-oriented programming which include encapsulation, polymorphism, abstraction, and inheritance.

Visual Basic .ASP NET runs on the .NET framework, which means that it has full access to the .NET libraries. It is a very productive tool for rapid creation of a wide range of Web, Windows, Office, and Mobile applications that have been built on the .NET framework.

The language was designed in such a way that it is easy to understand to both novice and advanced programmers. Since VB.NET relies on the .NET framework, programs written in the language run with much reliability and scalability. With VB.NET, you can create applications that are fully object-oriented, similar to the ones created in other languages like C++, Java, or C#. Programs written in VB.NET can also interoperate well with programs written in Visual C++, Visual C#, and Visual J#. VB.NET treats everything as an object.

It is true that VB.NET is an evolved version of Visual Basic 6, but it's not compatible with it. If you write your code in Visual Basic 6, you cannot compile it under VB.NET.

History of VB.NET

Microsoft®
VB.net

- VB.NET is a multi-paradigm programming language developed by Microsoft on the .NET framework. It was launched in 2002 as a successor to the Visual Basic language. This was the first version of VB.NET (VB.NET 7.0) and it relied on .NET version 1.0.
- In 2003, the second version of VB.NET, VB.NET 7.1, was released. This one relied on .NET version 1.1. This version came with a number of improvements including support for .NET Compact Framework and an improved reliability and performance of the
  .NET IDE. VB.NET 2003 was also made available in the academic edition of Visual Studio.NET and distributed to various scholars from different countries for free.
- In 2005, VB.NET 8.0 was released. The .NET core portion was dropped from its name so as to distinguish it from the classical Visual Basic language. This version was named Visual Basic 2005. This version came with many features since Microsoft wanted this

language to be used for rapid application developers. They also wanted to make it different from C# language. Some of the features introduced by this version of VB.NET included partial classes, generics, nullable types, operator overloading, and unsigned integer support. This version also saw the introduction of the IsNot operator.

- In 2008, VB 9.0 was introduced. This was released together with .NET 3.5. Some of the features added to this release of VB.NET included anonymous types, true conditional operator, LINQ support, XML literals, Lambda expressions, extension methods, and type inference.
- In 2010, Microsoft released VB 2010 (code 10.0). They wanted to use a Dynamic Language Runtime for this release, but they opted for co-evolution strategy shared between VB.NET and C# to bring these languages closer to each other.
- In 2012, VB 2012 (code 11.0) was release together with .NET 4.5. Its features included call hierarchy, iterators, caller data, asynchronous programming with "await" and "async" statements and the "Global" keyword in the "namespace" statements.
- In 2015, VB 2015 (code 14.0) was released alongside Visual Studio 2015. The "?." operator was introduced to do inline null checks. A string interpolation feature was also introduced to help in formatting strings inline.
- In 2017, VB 2017 (code 15.0) was introduced alongside Visual Studio 2017. A better way of organizing source code in just a single action was introduced.

VB.NET Features

VB.NET comes loaded with numerous features that have made it a popular programming language amongst programmers worldwide. These features include the following:

- VB.NET is not case sensitive like other languages such as C++ and Java.
- It is an object-oriented programming language. It treats everything as an object.
- Automatic code formatting, XML designer, improved object browser etc.
- Garbage collection is automated.
- Support for Boolean conditions for decision making.
- Simple multithreading, allowing your apps to deal with multiple tasks simultaneously.
- Simple generics.
- A standard library.
- Events management.
- References. You should reference an external object that is to be used in a VB.NET application.
- Attributes, which are tags for providing additional information regarding elements that have been defined within a program.
- Windows Forms- you can inherit your form from an already existing form.

Advantages of VB.NET

The following are the pros/benefits you will enjoy for coding in VB.NET:

- Your code will be formatted automatically.
- You will use object-oriented constructs to create an enterprise- class code.

- You can create web applications with modern features like performance counters, event logs, and file system.
- You can create your web forms with much ease through the visual forms designer. You will also enjoy drag and drop capability to replace any elements that you may need.
- You can connect your applications to other applications created in languages that run on the .NET framework.
- You will enjoy features like docking, automatic control anchoring, and in-place menu editor all good for developing web applications.

Disadvantages of VB.NET

Below are some of the drawbacks/cons associated with VB.NET:

- VB.NET cannot handle pointers directly. This is a significant disadvantage since pointers are much necessary for programming. Any additional coding will lead to many CPU cycles, requiring more processing time. Your application will become slow.
- VB.NET is easy to learn. This has led to a large talent pool. Hence, it may be challenging to secure a job as a VB.NET programmer.

Summary:

- VB.NET was developed by Microsoft. It is
- an object-oriented language.
- The language is not case sensitive.
- VB.NET programs run on the .NET framework.
- In VB.NET, the garbage collection process has been automated.

- The language provides windows forms from which you can inherit your own forms.
- VB.NET allows you to enjoy the drag and drop feature when creating a user interface.

Chapter 2: VB.Net Program Structure, Module, Classes: Hello World Example

Modules

A VB.NET program consists of the following:

- Namespace declaration
- One or more procedures
- A class or module
- Variables
- The Main procedure
- Comments
- Statements & Expressions

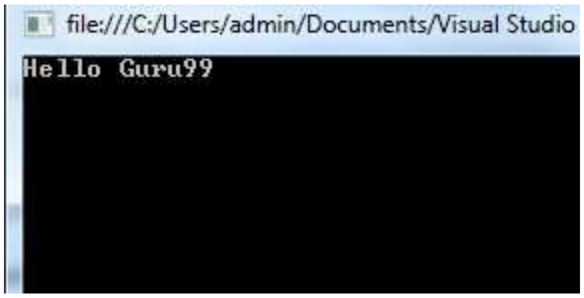
Hello World Program

Step 1) Create a new console application.

Step 2) Add the following code:

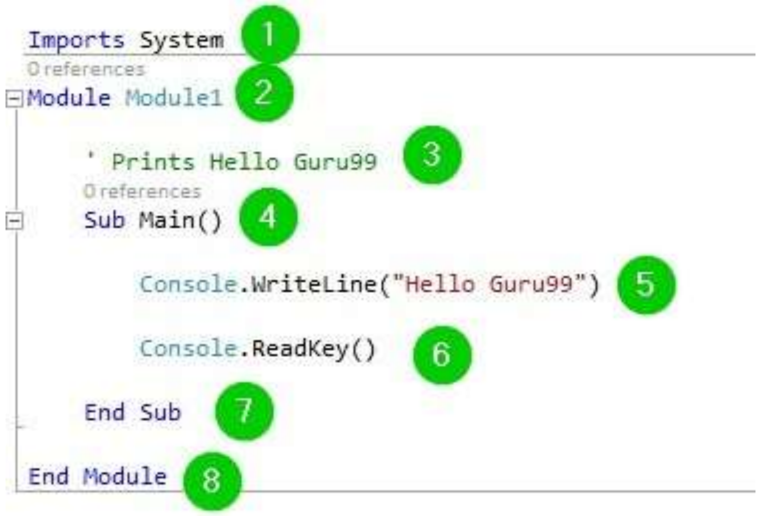
```
Imports System
Module Module1

    'Prints Hello Guru99
    Sub Main()

        Console.WriteLine("Hello Guru99")
        Console.ReadKey()
    End Sub
End Module
```

Step 3) Click the Start button from the toolbar to run it. It should print the following on the console:



```
Hello Guru99
```

Let us discuss the various parts of the above program:



```
Imports System
Module Module1

    ' Prints Hello Guru99
    Sub Main()

        Console.WriteLine("Hello Guru99")

        Console.ReadKey()

    End Sub

End Module
```

Explanation of Code:

1. This is called the namespace declaration. What we are doing is that we are including a namespace with the name System into our programming structure. After that, we will be able to access allthe methods that have been defined in that namespace without getting an error.

2. This is called a module declaration. Here, we have declared a module named Module1. VB.NET is an object-oriented language. Hence we must have a class module in every program. It is inside this module that you will be able to define the data and methods to be used by your program.
3. This is a comment. To mark it as a comment, we added a single quote (') to the beginning of the sentence. The VB.NET compiler will not process this part. The purpose of comments is to improve the readability of the code. Use them to explain the meaning of various statements in your code. Anyone reading through your code will find it easy to understand.
4. A VB.NET module or class can have more than one procedures. It is inside procedures where you should define your executable code. This means that the procedure will define the class behavior. A procedure can be a Function, Sub, Get, Set, AddHandler, Operator, RemoveHandler, or RaiseEvent. In this line, we defined the Main sub-procedure. This marks the entry point in all VB.NET programs. It defines what the module will do when it is executed.
5. This is where we have specified the behavior of the primary method. The WriteLine method belongs to the Console class, and it is defined inside the System namespace. Remember this was imported into the code. This statement makes the program print the text Hello Guru99 on the console when executed.
6. This line will prevent the screen from closing or exiting soon after the program has been executed. The screen will pause and wait for the user to perform an action to close it.
7. Closing the main sub-procedure.
8. Ending the module.

Classes

In VB.NET, we use classes to define a blueprint for a data type. It does not mean that a class definition is a data definition, but it describes what an object of that class will be made of and the operations that we can perform on such an object.

An object is an instance of a class. The class members are the methods and variables defined within the class.

To define a class, we use the Class keyword, which should be followed by the name of the class, the class body, and the End Class statement. This is described in the following syntax:

```
[ <attributelist> ] [ accessmodifier ] _
Class name
   [ Inherits classname ]
   [ statements ]
End Class
```

Here,

- The attributeList denotes a list of attributes that are to be applied to the class.
- The accessModifier is the access level of the defined class. It is an optional parameter and can take values like Public, Protected, Protected Friend, Friend, and Private.
- The Inherits denotes any parent class that it inherits.

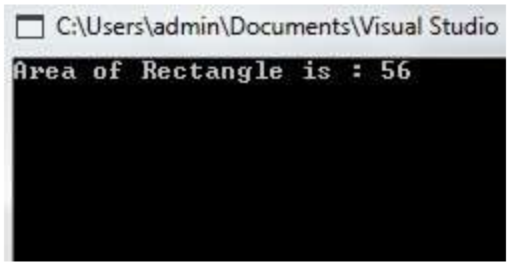
Following is example code to create a class in VB.NET -

Step 1) Create a new console application.

Step 2) Add the following code:

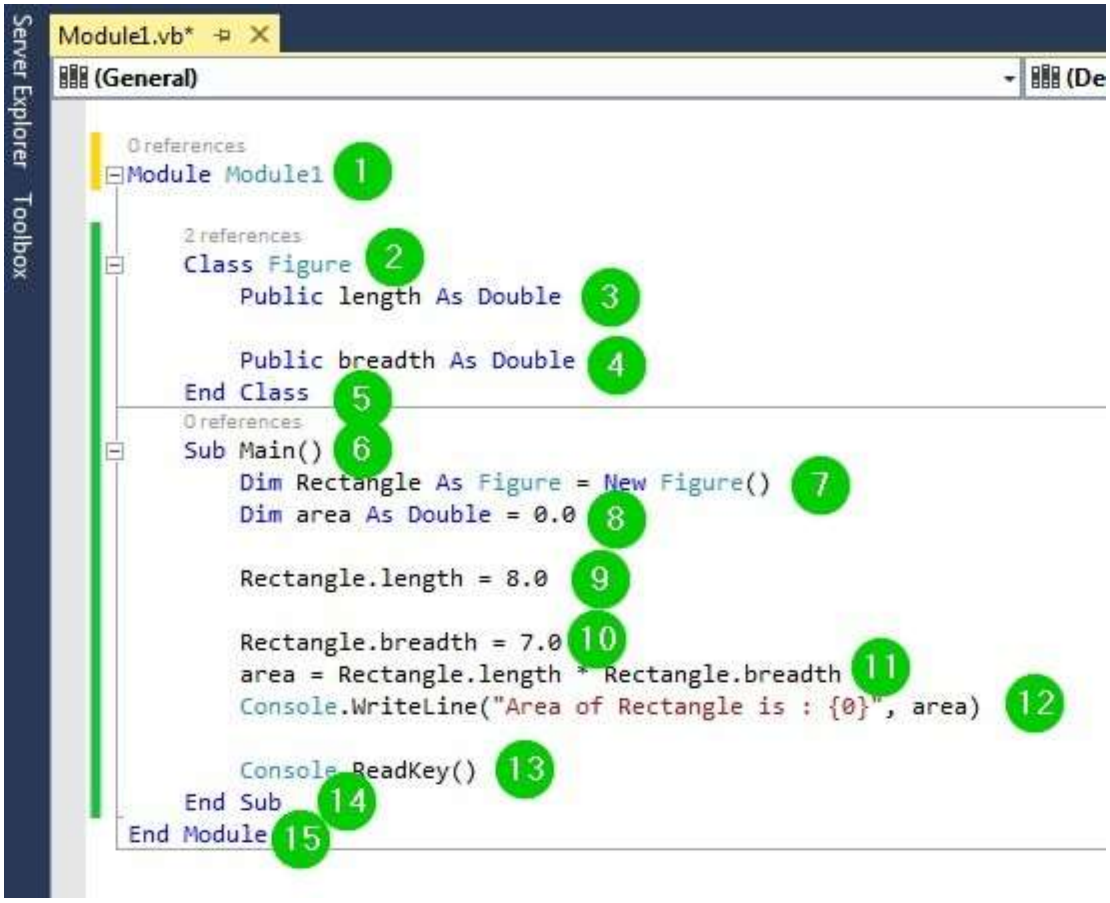
```
Imports System
Module Module1

    Class Figure
        Public length As Double

        Public breadth As Double
    End Class
    Sub Main()
        Dim Rectangle As Figure = New Figure()
        Dim area As Double = 0.0

        Rectangle.length = 8.0

        Rectangle.breadth = 7.0
        area = Rectangle.length * Rectangle.breadth
        Console.WriteLine("Area of Rectangle is : {0}", area)

        Console.ReadKey()
    End Sub
End Module
```

Step 3) Run the code by clicking the Start button from the toolbar. You should get the following window:



```
Area of Rectangle is : 56
```

We have used the following code:



```
Module Module1
    Class Figure
        Public length As Double

        Public breadth As Double
    End Class
    Sub Main()
        Dim Rectangle As Figure = New Figure()
        Dim area As Double = 0.0

        Rectangle.length = 8.0

        Rectangle.breadth = 7.0
        area = Rectangle.length * Rectangle.breadth
        Console.WriteLine("Area of Rectangle is : {0}", area)

        Console.ReadKey()
    End Sub
End Module
```

Explanation of Code:

1. Creating a module named Module1.
2. Creating a class named Figure.
3. Creating a class member named length of type Double. Its access level has been set to public meaning that it will be accessed publicly.
4. Creating a class member named breadth of type Double. Its access level has been set to public meaning that it will be accessed publicly.
5. Ending the class.
6. Creating the main sub-procedure.
7. Creating an object named Rectangle. This object will be of type figure, meaning that it will be capable of accessing all the

members defined inside the Figure class.

8. Defining a variable named area of type Double and initializing its value to 0.0.
9. Accessing the length property defined in the Figure class and initializing its value to 8.0.
10. Accessing the breadth property defined in the Figure class and initialize its value to 7.0.
11. Calculating the area of the rectangle by multiplying the values of length and breadth. The result of this calculation will be assigned to the area variable.
12. Printing some text and the area of the rectangle on the console.
13. Pausing the console waiting for a user to take action to close it.
14. Ending the sub-procedure.
15. Ending the class.

Structures

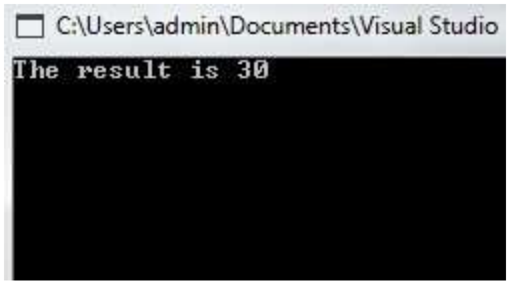
A structure is a user-defined data type. Structures provide us with a way of packaging data of different types together. A structure is declared using the structure keyword. Example to create a structure in VB.NET:

Step 1) Create a new console application.

Step 2) Add the following code:

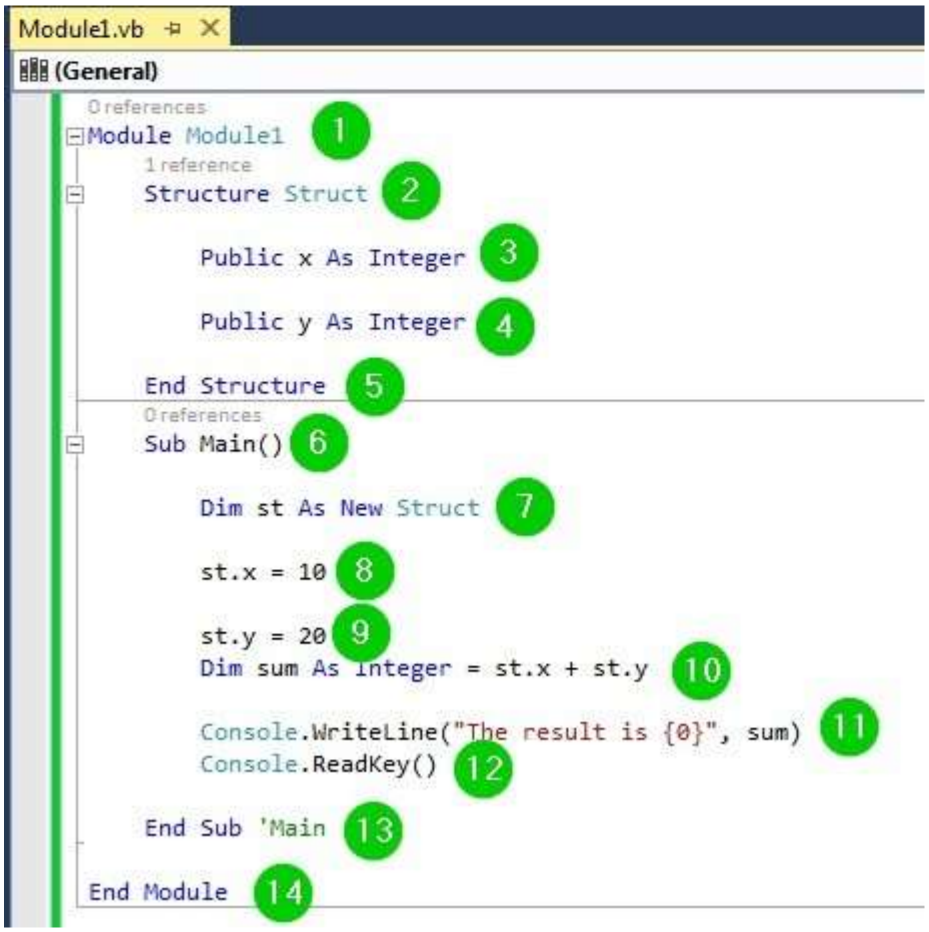
```
Module Module1
    Structure Struct
        Public x As Integer
        Public y As Integer
    End Structure
    Sub Main()
        Dim st As New Struct
        st.x = 10
        st.y = 20
        Dim sum As Integer = st.x + st.y
        Console.WriteLine("The result is {0}", sum)
        Console.ReadKey()

    End Sub
End Module
```

Step 3) Run the code by clicking the Start button from the toolbar. You should get the following window:



```
The result is 30
```

We have used the following code:



```
Module Module1
    Structure Struct

        Public x As Integer

        Public y As Integer

    End Structure
    Sub Main()

        Dim st As New Struct

        st.x = 10

        st.y = 20
        Dim sum As Integer = st.x + st.y

        Console.WriteLine("The result is {0}", sum)
        Console.ReadKey()

    End Sub 'Main

End Module
```

Explanation of Code:

1. Creating a module named Module1.
2. Creating a structure named Struct.
3. Creating a variable x of type integer. Its access level has been set to Public to make it publicly accessible.
4. Creating a variable y of type integer. Its access level has been set to Public to make it publicly accessible.
5. End of the structure.
6. Creating the main sub-procedure.
7. Creating an object named st of type Struct. This means that it will be capable of accessing all the properties defined within the structure named Struct.
8. Accessing the variable x defined within the structure Struct and initializing its value to 10.
9. Accessing the variable y defined within the structure Struct and initializing its value to 20.
10. Defining the variable sum and initializing its value to the sum of the values of the above two variables.
11. Printing some text and the result of the above operation on the console.
12. Pausing the console window waiting for a user to take action to close it.
13. End of the main sub-procedure.
14. End of the module.