Visual Basic Variables

In Visual Basic, Variables will represent storage locations and each variable will have a particular data type to determine the type of values the variable can hold.

Visual Basic is a Strongly Typed programming language so before we perform any operation on variables, it’s mandatory to define the variable with a required data type to indicate that the type of data the variable can hold in our application.

Syntax of Visual Basic Variables Declaration

Following is the syntax of declaring and initializing variables in visual basic.

Dim [Variable Name] As [Data Type]

Dim [Variable Name] As [Data Type] = [Value]

If you observe the above variable declarations, we added a required data type after the variable name to tell the compiler about the type of data the variable can hold.

Item Description

Dim It is useful to declare and allocate the storage space for one or more variables.

[Variable Name] It’s the name of the variable to hold the values in our application.

As The As clause in the declaration statement allows you to define the data type.

[Data Type] It’s a type of data the variable can hold such as integer, string, decimal, etc.

[Value] Assigning a required value to the variable.

Now, we will see how to declare and initialize the values to the variables in visual basic applications with examples.

Visual Basic Variables Example

Following is the example of using the variables in visual basic.

Module Module1

Sub Main()

Dim id As Integer



Dim name As String = "BCA"

Dim percentage As Double = 10.23

Dim gender As Char = "M"

Dim isVerified As Boolean

id = 10

isVerified = True

Console.WriteLine("Id:{0}", id)

Console.WriteLine("Name:{0}", name)

Console.WriteLine("Percentage:{0}", percentage)

Console.WriteLine("Gender:{0}", gender)

Console.WriteLine("Verfied:{0}", isVerified)

Console.ReadLine()

End Sub

End Module

If you observe the above example, we defined multiple variables with different data types and assigned the values based on our requirements.

When we execute the above example, we will get the result as shown below.

Id:10

Name:BCA

Percentage:10.23

Gender:M

Verfied:True

This is how we can declare and initialize the variables in Visual Basic applications based on our requirements.

**Visual Basic Data Types**

In Visual Basic, Data Types are useful to define a type of data the variable can hold such as integer, float, string, etc. in our application.



Visual Basic is a Strongly Typed programming language so before we perform any operation on a variable, it’s mandatory to define a variable with a required data type to indicate what type of data the variable can hold in our application.



Syntax of Defining Visual Basic Data Types

Following is the syntax of defining data types in visual basic.

Dim [Variable Name] As [Data Type]

Dim [Variable Name] As [Data Type] = [Value]

If you observe the above syntax, we added a required data type after the variable name to tell the compiler about the type of data the variable can hold.

Item Description

Dim It is useful to declare and allocate the storage space for one or more variables.

[Variable Name] It’s the name of the variable to hold the values in our application.

As The As clause in the declaration statement allows you to define the data type.

[Data Type] It’s a type of data the variable can hold such as integer, string, decimal, etc.

[Value] Assigning a required value to the variable.

Data Types in Visual Basic

The following table shows the list of available data types in a visual basic programming language with memory size and range of values.

Data Type Size Range

Boolean It depends on the Platform. True or False

Byte 1 byte 0 to 255

Char 2 bytes 0 to 65535

Date 8 bytes 0:00:00am 1/1/01 to 11:59:59pm 12/31/9999

Decimal 16 bytes (+ or -)1.0 x 10e-28 to 7.9 x 10e28

Double 8 bytes -1.79769313486232e308 to 1.79769313486232e308

Integer 4 bytes -2,147,483,648 to 2,147,483,647

Long 8 bytes -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Object 4 bytes on a 32-bit platform, 8 bytes on a 64-bit platform

Any type can be stored in a variable of type Object

SByte 1 byte -128 to 127

Short 2 bytes -32,768 to 32,767

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 Platform 0 to approximately 2 billion Unicode characters

UInteger 4 bytes 0 to 4,294,967,295

ULong 8 bytes 0 to 18,446,744,073,709,551,615 (1.8...E+19 †)

UShort 2 bytes 0 to 65,535

Now, we will see how to use the Data Types in our visual basic applications with examples.

Visual Basic Data Types Example

Following is the example of using the data types in visual basic.

Module Module1

Sub Main()

Dim id As Integer

Dim name As String = "BCA"

Dim percentage As Double = 10.23

Dim gender As Char = "M"c

Dim isVerified As Boolean

id = 10

isVerified = True

Console.WriteLine("Id:{0}", id)

Console.WriteLine("Name:{0}", name)

Console.WriteLine("Percentage:{0}", percentage)

Console.WriteLine("Gender:{0}", gender)

Console.WriteLine("Verfied:{0}", isVerified)

Console.ReadLine()

End Sub

End Module

If you observe the above example, we defined multiple variables with different data types and assigned the values based on our requirements.

When we execute the above example, we will get the result as shown below.

Id:10

Name:BCA

Percentage:10.23

Gender:M

Verfied:True

This is how we can use the data types in Visual Basic applications based on our requirements.

**Common Integer Math Operations**

‘define n…

Dim n As Integer

‘try adding numbers…

n=16

n=n+8

messageBox.Show(“Addition test…” &n, “Integer Math”)

‘try subtracting numbers…

n=24

n=n-2

messageBox.Show(“Subtraction test…” &n, “Integer Math”)

‘try multiplying numbers…

n=6

n=n\*10

messageBox.Show(“Multiplication test…” &n, “Integer Math”)

‘try dividing numbers…

n=12

n=n/6

messageBox.Show(“Division test…” &n, “Integer Math”)

**Integer Math Shorthand**

‘define n…

Dim n As Integer

‘try adding numbers…

n=16

n+=8

messageBox.Show(“Addition test…” &n, “Integer Math”)

‘try subtracting numbers…

n=24

n-=2

messageBox.Show(“Subtraction test…” &n, “Integer Math”)

‘try multiplying numbers…

n=6

n\*=10

messageBox.Show(“Multiplication test…” &n, “Integer Math”)

‘try dividing numbers…

n=12

n/=6

messageBox.Show(“Division test…” &n, “Integer Math”)

**Floating Point Math Operations**

‘define n…

Dim n As Double

‘try multiplying numbers…

n=45.34

n\*=4.333

MessageBox.Show(“Multiplication test…”&n,”Floating Points”)

‘try dividing numbers…

n=12

n/=7

MessageBox.Show(“Division test…”&n,”Floating Points”)

**Single-Precision Floating-Point Numbers**

Dim n As Double tells Visual Basic .NET that we want to create a variable that holds a double-precision floating-point number.

A double-precision floating-point number can hold any value between -1.7×10308and +1.7×10308 and A single-precision floating-point number can hold any value between -3.4×1038and +3.4×1038. If we want to use a single -precision number, use As Single rather than As Double

Dim n As Single.

**String**

A String is simply a collection of characters and use double quotes to mark its beginning and end. We need to use double quotes because these double quotes tell the compiler not to compile the text that is contained within the string. If we didn’t include the quotes, VB.NET Compiler will treat the value to be stored in the variable as part of the program’s code.

‘define a string…

Dim s As String

S=”Hello, world!”

‘Display the result

MessageBox.Show(s,”Strings”)

**Date**

Another common datatype that we will often use is Date. This data type holds a date value.

‘get the current date and time…

Dim theDate As Date

theDate=Date.Now()

‘display it…

MessageBox.Show(theDate,“Date Demo”)

**Boolean**

A Boolean variable can be either True or False. Boolean values are important to make decisions in our programs.

‘define a Boolean…

Dim myfirstboolean As Boolean

myfirstboolean=True

‘define another Boolean…

Dim mysecondboolean As Boolean

mysecondboolean=False

‘display the value

MessageBox.Show(“First:”& myfirstboolean,”Boolean Demo”)

MessageBox.Show(“Second:”& mysecondboolean,”Boolean Demo”)

**Keywords**

A keyword is a word that has a special meaning in Visual Basic .NET that is used for things such as commands.

Dim n as Integer

Dim tells VB that what follows is a variable definition. Dim means Dimension.

**Constants**

A constant is a meaningful name that takes the place of a number or string that does not change. Constants store values that, as the name implies, remain constant throughout the execution of an application. Constants we create ourself are described as user-defined.

Const conPi = 3.14159265358979

Const conPi2 = conPi \* 2

**Comments**

In Visual Basic .NET, we begin our comments with an apostrophe (‘) and anything on the same line following that apostrophe is our comment.

‘ define a value for n….

Dim n As Integer

n=50

‘ Add 1 to the value of n ..

n=n+1

‘ display the new value of n to the user..

MessageBox.Show(“Value of n+1 =” &n)