FREE eBook

LEARNING Visual Basic 6

#vb6

Table of Contents

About

Chapter 1: Getting started with Visual Basic 6

Remarks

This section provides an overview of what vb6 is, and why a developer might want to use it.

It should also mention any large subjects within vb6, and link out to the related topics. Since the Documentation for vb6 is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting vb6 set up or installed.

Hello world

```
' A "Hello, World!" program in Visual Basic.
Module Hello
  Sub Main()
      MsgBox("Hello, World!") ' Display message on computer screen.
  End Sub
End Module
```

Read Getting started with Visual Basic 6 online: https://riptutorial.com/vb6/topic/3439/getting-started-with-visual-basic-6

Chapter 2: Basic Syntax

Examples

if / else statement

```
If condition Then
    code to execute if true
ElseIf condition Then
    code
Else
    code to execute if conditions are both false
End If
```

for loop

```
For I as Integer = 1 To 10 Step 1
    code to execute
Next
```

Step is optional and Step 1 is the default. Step tells it how to count, so -1 would have it subtract 1 each time and Step 5 would have it add 5 each time thru the loop.

In case the loop need to be stopped, then the `Exit For` statement can be used, as in the below example;

```
Dim iIndex as integer

For I as Integer = 1 To 10 Step 1

Debug.Print  I
iIndex = I * 10

If iIndex > 90 Then
    Exit For
End If

Loop
```

Here, instead of printing 1 to 10, it will stop at 9, since the condition told the process to stop when iIndex reaches 90.

Do Loop

Another common type of loop in Visual Basic is the `DO loop`, which would run a piece of code continuously until it is told to stop. On the contrary of some other loops whereby indexes are used to stop the process, in this particular loop, it should be told to stop.

A simple example illustrating the loop is as follows

```
Dim iIndex1 As Integer
iIndex1 = 1

Do
   Debug.Print iIndex1
   iIndex1 = iIndex1 + 1

   If iIndex1 = 10 Then
      Exit Do
   End If
Loop
```

The above piece of code will take an Index, initialized to 1, and increment it. A `Debug.Print` will help print the index to rack the loop. On each loop, the code will verify if the index has reached 10 and if and only if the condition is true, the `Exit Do` will be executed, which will stop the loop.

Select Case Statement

```
    Dim number As Integer = 8
    Select Case number
        Case 1 To 5
            Debug.WriteLine("Between 1 and 5, inclusive")
            ' The following is the only Case clause that evaluates to True.
        Case 6, 7, 8
            Debug.WriteLine("Between 6 and 8, inclusive")
        Case 9 To 10
            Debug.WriteLine("Equal to 9 or 10")
        Case Else
            Debug.WriteLine("Not between 1 and 10, inclusive")
    End Select
```

Read Basic Syntax online: https://riptutorial.com/vb6/topic/9389/basic-syntax

Chapter 3: Function Procedures

Introduction

Function is a series of statements enclosed by "Function" and "End Function" statements.

The Function performs an activity and returns control to the caller. When it returns control, it also returns a value to the calling code.

You can define a Function in a Class, Structure & Module. By default It is Public. It means, you can call it from anywhere in your application that has access to the class, Structure or Module in which you defined it.

Syntax

- [Modifiers] Function Name_Of_The_Function [(Arg_List)] As Return_Type
- [Statements]
- End Function

Remarks

- The two Function Modifiers used in this examples are Public & Private. This Modifiers define the scope of the Function.
- Functions with a Private scope can only be called from the source file from where they were defined. In our case it can be called with in the Module. And cannot be called outside the Module.
- Functions with Public scope can be called both outside and inside the Module. Simply we can say as "We can call it any where in the program".
- Default Modifier of the Function is Public.
- By default, the function arguments are passed by reference (In a separate topic, this will be explained in detail).

Examples

Creating & Calling a Function

This Example using Standard EXE Project With addition of a Module File.

- Create New "Standard EXE" Project. So here, a Form will get added to the Project by default.
- Add a Module File to the Project
- Place a Command Button on the Form
- Create Command Button Click Event.

Module code

Created two Functions in the Module. One is a Public Function (FnAdd). It takes two Integer arguments val_1 & val_2. It returns an Integer. This Function add the two arguments and return the value to the caller. Before the addition, the two arguments undergo a process in another Function. Which is a Private Function. Characteristic/Rules of the Public & Private Function explained in the Remarks section.

```
Public Function FnAdd(val_1 As Integer, val_2 As Integer) As Integer

'Calling private function
val_1 = FnMultiplyBy5(val_1)

'Calling private function
val_2 = FnMultiplyBy5(val_2)

'Function return statement
FnAdd = val_1 + val_2

End Function
```

Below is the Private function in the Module. It takes one integer arguments val. It returns an integer. This function multiply a value 5 with the argument and return the result to the caller.

```
Private Function FnMultiplyBy5(Val As Integer) As Integer

'Function return statement
FnMultiplyBy5 = Val * 5

End Function
```

Form Code

In the Command Button click Event. Here we are calling the Module Public function "FnAdd"

```
Private Sub Command1_Click()
Debug.Print FnAdd(3, 7)
End Sub
```

Result in the Immediate Window

```
50
```

Read Function Procedures online: https://riptutorial.com/vb6/topic/9227/function-procedures

Chapter 4: Installing VB6 on Windows 10

Examples

Installation Wizard

Visual Studio 6.0 Installer wizard

By default, the following packages do not install properly under Windows 10:

- Visual Studio 6 Enterprise
- Visual Studio 6 Professional
- Visual Basic 6 Enterprise
- Visual Basic 6 Professional

To install the above packages, you'll either need to make numerous adjustments and registry hacks, or use the fantastic Visual Studio 6.0 Installer wizard by Giorgio Brausi.

You'll need the following items before starting:

- Your original Visual Studio/Basic Program CDs and keys
- Your original MSDN CDs
- Visual Studio Service Pack 6
- Visual Studio 6.0 Installer wizard
- On Windows 10 build 1511 or later, you'll require Admin Rights.

The Wizard will take you through the necessary steps for a successful installation of Visual Basic 6.

Note that the installation of the Server Applications is not currently possible.

Read Installing VB6 on Windows 10 online: https://riptutorial.com/vb6/topic/4230/installing-vb6-on-windows-10

Chapter 5: Variables

Examples

Variable types

There are different variable types for different purposes. In Visual Basic 6 the following variable types are available:

- Array
- Boolean
- Byte
- Currency
- Date
- Double
- Integer
- Long
- Single
- String
- Variant

You declare a variable by using the `Dim` keyword:

```
Dim RandomNumber As Integer
```

If you do not specify a variable type the variable will default to `Variant`:

```
Dim Foo
```

is equivalent to

```
Dim Foo As Variant
```

Boolean

Boolean is the simplest variable type as it can contain only one of two values: True or False.

```
Foo = True
Bar = False
```

Booleans can be used to control the flow of code:

```
Dim Foo as Boolean
Foo = True
```

```
If Foo Then
  MsgBox "True"
Else
  MsgBox "False"
End If
```

Integer

An integer is a numeric data type and can contain a 16-bit signed value (-32768 to +32767). If you know that a variable will only contain whole numbers (such as 9) and not fractional numbers (such as 5.43), declare it as an integer (or long) datatype.

```
Dim RandomNumber As Integer
RandomNumber = 9
```

Integers are commonly used as counters in `For...Next` loops:

```
Dim Counter As Integer

For Counter = 0 to 2
  MsgBox Counter
Next Counter
```

Trying to assign a value less than -32768 or greater than 32767 to an integer will result in a run-time error:

```
Dim MyNumber As Integer
MyNumber = 40000  'Run-time error '6': Overflow
```

String

A string variable can contain an empty text, a character, a word or a text of variable length. The string value must be contained in quotation marks (`"`).

```
Dim Fruit as String
Fruit = "Banana"
```

If you need quotation marks inside a string literal you use two subsequent quotation marks (`""`).

```
Dim Quote as String
Quote = "Bill says: ""Learn VB!"""
```

Read Variables online: https://riptutorial.com/vb6/topic/7511/variables

Credits

S. No	Chapters	Contributors
1	Getting started with Visual Basic 6	Community, Scheffer
2	Basic Syntax	Nadeem_MK, Talal Abdoh, user7491506
3	Function Procedures	Jeet
4	Installing VB6 on Windows 10	ThunderFrame
5	Variables	BlueEel