

**TegoSoft Inc.**

Copyright © 1994-1996 TegoSoft Inc. ® All Rights Reserved

*P.O.Box 389, Bellmore, NY 11710*

Web Site: <http://www.tegosoft.com>

 e-mail: [tegosoft@msn.com](mailto:tegosoft@msn.com)

# Lesson 14 - Decision Maker Statements

In this lesson you'll learn about the decision maker statements of Visual Basic.


## Decision Maker Statements


Every programming language needs decision maker statements. As implied by the name, a decision maker statement is a statement that makes a decision.


## Creating the Directory of the Project and Saving the Files of the Project

As usual, you'll start by creating the directory of the project, and saving the files of the project.


 Create the **C:\VBMyProg\Lesson14** directory. You'll save the files of this lesson to this directory.

 Select **New Project** from the **File** menu of Visual Basic.

 Make the window of Form1 the selected window, select **Save File As** from the **File** menu of Visual Basic, and save the file as **Decision.FRM** inside the **C:\VBMyProg\Lesson14** directory.

 Select **Save Project As** from the **File** menu of Visual Basic, and save the project as **Decision.VBP** inside the **C:\VBMyProg\Lesson14** directory.

## Always Use Option Explicit

 Inside the general declarations section of Form1 type the following code:

**Option Explicit**

## Ask the User to Type a Number

You'll now write code that asks the user to type a number:

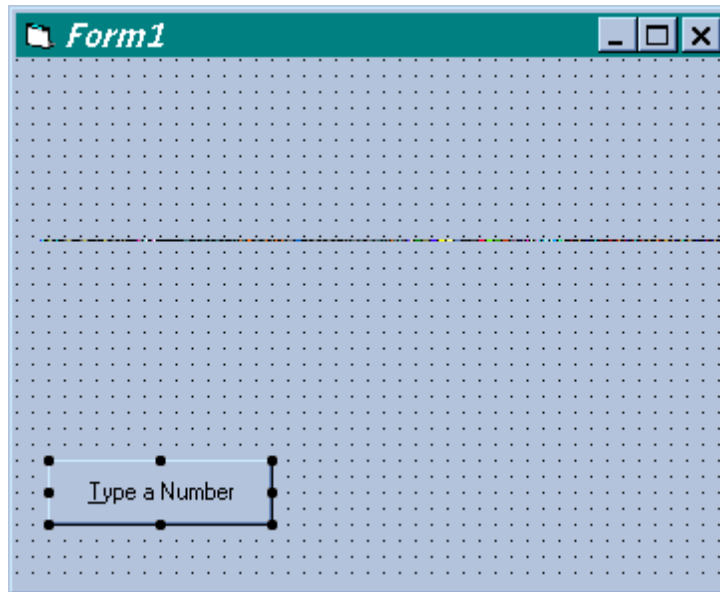
 Place a **CommandButton** inside Form1.

 Set the properties of the **CommandButton** as follows:

**Name:** cmdTypeANumber

**Caption:** Type a Number


 Size Form1 and drag the **cmdTypeANumber** button so that Form1 will look as shown in Figure 14.1.



**Figure 14.1.** Form1 with the Type a Number button in it.

## Attaching Code to the Click Event of the Type a Number Button

You'll now attach code to the **Click** event of the **Type a Number** button.

 Type the following code inside the **cmdTypeANumber\_Click()** procedure:

```
Private Sub cmdTypeANumber_Click()  
Dim Result  
Result = InputBox("Type the Number 1, 2, or 3", _  
                  "The Decision Program", _  
                  1)  
  
Result = Val(Result)  
  
If Result = 1 Then  
    MsgBox "You typed 1"
```

```
End If
```

```
If Result = 2 Then  
    MsgBox "You typed 2"  
End If
```

```
If Result = 3 Then  
    MsgBox "You typed 3"  
End If
```

```
End Sub
```

You declared the variable **Result**:

```
Dim Result
```

You then display an Input dialog box and assigned the returned value of the Input dialog box to the **Result** variable:

```
Result = InputBox("Type the Number 1, 2, or 3", _  
                 "The Decision Program", _  
                 1)
```

You then execute the **Val()** function to convert the string that was returned from the **InputBox()** function to a number:

```
Result = Val(Result)
```

The you use a series of three **If** statements to examine the value of **Result**:

```
If Result = 1 Then  
    MsgBox "You typed 1"  
End If
```

```
=====
```

```
If Result = 2 Then
    MsgBox "You typed 2"
End If
```

```
If Result = 3 Then
    MsgBox "You typed 3"
End If
```


If **Result** is equal to **1**, the first **If** statement is satisfied, the message box displays the message: **You typed 1**

Similarly, if **Result** is equal to **2**, the second **If** statement is satisfied, and if **Result** is equal to **3**, the third **If** statement is executed.

## Executing the Decision Program


Let's see your code in action:

 Select **Save Project** from the **File** menu of Visual Basic to save your Work.


 Select **Start** from the **Run** menu of Visual Basic to execute the Decision program.

 Click the **Type a Number** button.

The Decision program responds by displaying the Input dialog box.

 Type **2** inside the Input dialog box and then click the OK button of the Input dialog box.

The Decision program displays a message, telling you that you typed 2.

 Experiment with the Decision program. Type **1,2**, or **3** inside the Input dialog box, and verify that the message box reports the number that you typed inside the Input dialog box.

 Terminate the Decision program.

## Exiting from a Procedure

Currently, if you typed **4** for example inside the Input dialog box, you will not get an message box. Why? Because you did not type code that takes care of a number other than 1, 2, or 3. you typed three **If** statements inside the **cmdTypeANumber\_Click()** procedure. One **If** statement takes care of the case when you typed **1** inside the Input dialog box, the second **If** statement takes care of the number **2**, and the third **If** statement takes care of the **number 3**. You did **not** type any code that takes care of other numbers.

It would be nice if when the user types a number other than 1, 2 or 3, the Decision program will display a message telling the user that a number other than 1,2 or 3 was typed. There are a few ways you can implement this. For the sake of learning new topics in Visual Basic, you'll now implement this in a few ways.

 Add code and modify the **cmdTypeANumber\_Click()** procedure so that it will look as follows:

```
Private Sub cmdTypeANumber_Click()  
Dim Result  
  
Result = InputBox("Type the Number 1, 2, or 3", _  
                  "The Decision Program", _  
                  1)  
  
Result = Val(Result)
```

```
If Result = 1 Then
    MsgBox "You typed 1"
    Exit Sub
End If

If Result = 2 Then
    MsgBox "You typed 2"
    Exit Sub
End If

If Result = 3 Then
    MsgBox "You typed 3"
    Exit Sub
End If

MsgBox "You did not type 1,2 or 3"

End Sub
```

Take a look at the first **If** statement:

```
If Result = 1 Then
    MsgBox "You typed 1"
    Exit Sub
End If
```

Under the **If** you have the statement:

```
Exit Sub
```

So if the user typed **1** inside the Input dialog box, the message box is displayed, and then the **Exit Sub** statement is executed. **Exit Sub** causes the procedure to

=====

terminate immediately. So all the code that appears after the first **If** statement is **not** executed.

Similarly, if the user typed **2** inside the Input dialog box, the code under the second **If** statement is executed, and because the **Exit Sub** statement appears under the second **If** statement, the procedure terminates.

Similarly, if the user typed **3** inside the Input dialog box, the code under the third **If** statement is executed, and because the **Exit Sub** statement appears under the third **If** statement, the procedure terminates.


If the user did not type 1, 2 or 3 inside the Input dialog box, none of the **If** statement is satisfied, and the last statement in the procedure is executed:


```
MsgBox "You did not type 1,2 or 3"
```

So if the user did not type 1, 2 or 3 inside the Input dialog box, the preceding message box will be displayed.

Let's see your code in action:

 Select **Save Project** from the File menu of Visual Basic to save your work.

 Execute the Decision program, and verify that when you type 1, 2, or 3 inside the input dialog box, you get a message that tells you which number you typed. If you type a number other than 1, 2 or 3, you get the message: **You did not type 1,2 or 3**

 Experiment with the Decision program, then terminate the program.

## Using If...Else If Statements

In the preceding sections you implemented a code that takes care of the case when the user did not type 1, 2 or 3. Actually, this situation where your program examines various cases is a situation that you'll encounter a lot during your

=====

future programming projects. So the designers of Visual Basic incorporated a more elegant way of implementing this mechanism. This will now be demonstrated.

 Modify the code inside the **cmdTypeANumber\_Click()** procedure so that it will look as follows:

```
Private Sub cmdTypeANumber_Click()  
Dim Result  
  
Result = InputBox("Type the Number 1, 2, or 3", _  
                  "The Decision Program", _  
                  1)  
  
Result = Val(Result)  
  
If Result = 1 Then  
    MsgBox "You typed 1"  
ElseIf Result = 2 Then  
    MsgBox "You typed 2"  
ElseIf Result = 3 Then  
    MsgBox "You typed 3"  
Else  
    MsgBox "You did not type 1,2 or 3"  
End If  
  
End Sub
```

The code that you typed inside the **cmdTypeANumber\_Click()** procedure uses a series of **If...Elseif** statements. It works as follows:

The first **If** condition is examined. If the **If** condition is satisfied, the code under the **If** is executed. The first **If** statement is satisfied if the user typed **1** inside the Input dialog box.

=====


If the user typed **2** inside the Input dialog box, the first **Elseif** is satisfied.


If the user typed **3** inside the Input dialog box, the second **Elseif** is satisfied.


If the user typed a number other than 1, 2 or 3, the **If** condition is not satisfied, and none of the **Elseif** conditions are satisfied. In this case, the code under the **Else** will be executed.

As you can see, this way of implementing the program is shorter and more elegant.

Let's see your code in action:

 Select **Save Project** from the **File** menu of Visual Basic.

 Execute the Decision program, and verify that it works in the same manner as before.

 Experiment with the Decision program, then terminate the Decision program.

## Another Elegant Way of Implementing the Same Thing...

As stated, many of your future programs will examine several cases, so the designers of Visual Basic incorporated an additional way of making decisions from within your program. In particular, the **Select Case** statement is yet another way of making decisions in Visual Basic programs.

 Modify the code inside the **cmdTypeANumber\_Click()** procedure so that it will look as follows:

=====

```

Private Sub cmdTypeANumber_Click()

Dim Result

Result = InputBox("Type the Number 1, 2, or 3", _
                  "The Decision Program", _
                  1)

Result = Val(Result)

Select Case Result
    Case 1
        MsgBox "You typed 1"
    Case 2
        MsgBox "You typed 2"
    Case 3
        MsgBox "You typed 3"
    Case Else
        MsgBox "You did not type 1,2 or 3"
End Select

End Sub

```

You typed the following decision maker statements:

```

Select Case Result
    Case 1
        MsgBox "You typed 1"
    Case 2
        MsgBox "You typed 2"
    Case 3
        MsgBox "You typed 3"
    Case Else

```

=====

```
        MsgBox "You did not type 1,2 or 3"  
End Select
```

The **Select Case** statements starts as follows:

```
Select Case Result
```

In the preceding, the variable **Result** is typed at the end of the **Select Case** statement. This means that the **Select Case** statement will examine the value of **Result**.

The last statement in the **Select Case** block is **End Select**:

```
Select Case Result  
    ...  
    ...  
    ...  
End Select
```

In between the **Select Case Result** and the **End Select** you type code that corresponds to the various cases. For example, the first case is as follows:

```
    Case 1  
        MsgBox "You typed 1"
```

So when **Result** is equal to **1**, the code under the **Case 1** is executed.

Similarly, when **Result** is equal to **2**, the case under the **Case 2** statement is executed, and when **Result** is equal to **3**, the case under the **Case 3** is executed.

If **Result** is not equal to 1, 2 or 3, the case under the **Case Else** is executed.

=====



# Frequently Asked Questions



**Q1.** Which decision make statement should I use? A regular **If...End If** statements, **If...Elseif...EndIf** statements, **Select Case**?

**A1.** Depending on the particular task that you are trying to implement, sometimes a regular **If...EndIf** statement will do the job. If you have more than one case to examine, use the **If...Elseif...EndIf** statements or the **Select Case**.

The **Elseif** statement are more powerful than the **Select Case** statement because the **Select Case** statement can only check if a certain variable is equal or not equal to a certain value.

On the other hand, the **Elseif** statement give you additional checking capabilities.

For example, the following **If...Elseif** statement check if the value of the **MyVariable** variable is in a certain range:

```
If Result > 0 And Result < 3 Then
    MsgBox "Value is between 1 and 3"
ElseIf Result > 4 And Result < 8 Then
    MsgBox "Value is between 4 and 8"
ElseIf Result > 10 Then
    MsgBox "Value is larger than 10"
End If
```

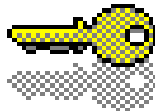
## Exam



1. Which of the following Case will be executed?

```
Dim MyVariable
MyVariable = -1
Select Case MyVariable
    Case -1
        MsgBox "Case -1 is executed"
        MyVariable = -2
    Case -2
        MsgBox "Case -2 is executed"
End Select
```

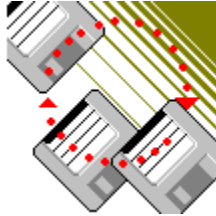
## Answers to Exam




1. Only the code under **Case -1** will be executed.

Under **Case -1**, the value of **MyVariable** is set to **-2**. So at first glance, it looks as if also **Case -2** will be executed, because when the program will go over the cases, **Case -2** is satisfied. However, this is **not** the case. Why? Because the **Select Case** statement checks the value of **MyVariable** only once. Once one of the cases is satisfied, the program aborts the **Select Case** block. So after **Case -1** is satisfied, the other cases are not checked.

# Project



Modify the Decision program so that when the user types the text **Visual Basic** inside the Input box, a message box will confirm that this is what the user typed.

 Modify the code inside the **cmdTypeANumber\_Click()** procedure so that the procedure will look as follows:

```
Private Sub cmdTypeANumber_Click()  
  
Dim Result  
  
Result = InputBox("Type the Number 1, 2, or 3", _  
                  "The Decision Program", _  
                  1)  
  
If Result = "Visual Basic" Then  
    MsgBox "You typed Visual Basic"  
    Exit Sub  
End If  
  
Result = Val(Result)  
  
Select Case Result  
    Case 1  
        MsgBox "You typed 1"  
    Case 2  
        MsgBox "You typed 2"  
    Case 3
```

```
    MsgBox "You typed 3"  
    Case Else  
        MsgBox "You did not type 1,2 or 3"  
End Select
```

**End Sub**


The code that you typed examines the value of the **Result** variable as follows:


```
If Result = "Visual Basic" Then  
    MsgBox "You typed Visual Basic"  
    Exit Sub  
End If
```


So if the user types **Visual Basic**, the **If** statement is satisfied, and the message box is displayed. Because you also typed the **Exit Sub** statement under the preceding **If** statement, the **cmdTypeANumber\_Click()** procedure is terminated after the message box is executed (so the rest of the code in the procedure are **not** executed).

 Select **Save Project** from the **File** menu of Visual Basic to save your work.

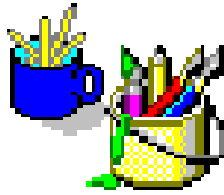
 Execute the Decision program.

 Inside the Input dialog box type a number such as 1 , 2 or 3 and verify that the corresponding message box is displayed.

 Inside the Input dialog box type the text **Visual Basic** and verify that the corresponding message box is displayed.


 Experiment with the Decision program, and then terminate the program.

# Cosmetic Considerations



Typically, when implementing programs that let the user type text, you have to decide whether you want to force your user to type the text as case sensitive text or as a non case sensitive text. For example, in the Project section of this lesson the first **If** statement inside the **cmdTypeANumber\_Click()** procedure is satisfied provided that the user typed the text as **Visual Basic** with the **B** and **V** as capital letters, and the rest of the letters in lower case letters.

Cosmetically speaking, in many cases you want your program to be liberal, and you want the **If** condition to be satisfied regardless of the capitalization. You can modify the Decision program so that if the user typed **visual basic**, **visUAL bASIC**, **ViSuAl bAsIc**, or any other combination of capitalization, the first **If** condition inside the **cmdTypeANumber\_Click()** procedure will be satisfied. Here is how you accomplish this:

 Modify the code inside the **cmdTypeANumber\_Click()** procedure so that the procedure will look as follows:

```
Private Sub cmdTypeANumber_Click()  
  
Dim Result  
  
Result = InputBox("Type the Number 1, 2, or 3", _  
                  "The Decision Program", _  
                  1)  
  
If UCase(Result) = "VISUAL BASIC" Then  
    MsgBox "You typed Visual Basic"  
    Exit Sub  
End If
```

=====

```
...  
...  
...  
End Sub
```

You modified the **If** statement so that it will look as follows:


```
If UCase(Result) = "VISUAL BASIC" Then  
    MsgBox "You typed Visual Basic"  
    Exit Sub  
End If
```


The **UCase()** function has one parameter. You supply a string to the **UCase()** function, and the **UCase()** function converts the string that you supply to a string that is all capital case letters. For example, if you supplied the string "**abc**" as the parameter of the **UCase()** function, the **UCase()** function returns the string **ABC**.

In our case, if the user types the string **visual basic** inside the Input dialog box, the **If** statement checks if the upper case of the string **visual basic** (which is **VISUAL BASIC**) is equal to the string **VISUAL BASIC**.

 Select **Save Project** from the **File** menu of Visual Basic to save your work.

 Execute the Decision program.

 Inside the Input dialog box type the text **ViSuAl BaSiC** and verify that the corresponding **If** statement is satisfied.

 Experiment with the Decision program, and then terminate the program.

# How To Contact TegoSoft



You can contact TegoSoft Inc. by any one of the following methods:

- Use TegoSoft Internet Web site:

<http://www.tegosoft.com>

- Send TegoSoft an e-mail:

[tegosoft@msn.com](mailto:tegosoft@msn.com)

- Send TegoSoft a letter:

*TegoSoft Inc.*

*P.O.Box 389*

*Bellmore, NY 11710*

*USA*

## Technical Support



If you have a technical question, you can post the question to the TegoSoft Technical Support staff, and they will try to answer your question.

The **best** way to post a technical question is by sending TegoSoft an e-mail.

- The e-mail of TegoSoft is:

[tegosoft@msn.com](mailto:tegosoft@msn.com)

**When sending TegoSoft an e-mail with a technical question, please follow the following format:**

=====

Date: \_\_\_\_\_  
Your name: \_\_\_\_\_  
Company (if applicable): \_\_\_\_\_  
Your phone number: \_\_\_\_\_  
Your e-mail: \_\_\_\_\_  
Country (if not USA): \_\_\_\_\_  
State (if inside USA): \_\_\_\_\_

**Operating System used:** \_\_\_\_\_  
**Programming language and version :** \_\_\_\_\_

**My technical question is:**

## **Copyright © and Notices**

Copyright © 1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved

### **TegoSoft Self Study Tutorials & Software**

#### **Copyright ©1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved**

Although precaution was taken in the preparation of this document, TegoSoft assumes no responsibility for errors or omissions. TegoSoft is not liable for damages resulting from the use of the information contained in this document. Use this document at your own risk. This document is copyright protected. This means that you should treat this document like any other copyright material. No part of this document should be copied in any way. You are not allowed to use this document or any part of this document for any purpose other than read it. You are not allowed to publish this document or any part of this document in any way. You are not allowed to sell this document or any part of this document, you are not allowed to incorporate this document or any part of this document in any book, magazine, Web Site, Electronic forums, disks, CDs, or any other media. The only thing that you are allowed to do with this document is read it for the sole purpose of studying the material that is presented in this document. If this document includes software, the software must be treated in the exact same way that this document is treated. You are not allowed to distribute the software in any way. The sole purpose of

=====

supplying the accompanying software is to enable you to experience with it. No part of this document should be modified. If accompanying software is included with this document, you are not allowed to modify the software.

Rev. 031796-1