Control structures

Contents

Overview

In this document we will show some statements that provide a way to control the program flow that means to influence the sequence of source code statements that will be executed. Other methods to change the execution order like subroutine calls or event handling will be described in other documents.

If..Then..Else statement

Executes a group of statements conditionally. The result value of an expression defines if the sattements behind the **Then** or the statements behind the **Else** will be executed.

Syntax

```
If condition Then
   [statements1]
[ Else If
   [statements2]]
[Else
   [statements3]]
End If
```

where *condition* is a numeric expression that evaluates to **True** or **False.**
statements are one or more statements that are executed if the result of the condition is **True**.

If example

This example prints some text dpending on the value of the variable i.

```
 ' Event handler for button_click event

 Sub btn_start_clicked()
    Dim i As Integer

    i = 9
    Print "Value is "+str(i)

    If i < 5 Then
       Print "Less than 5"
    Else
       Print "Greater than 5"
    End If

    If i < 10 Then
       Print "Less than 10"
    Else
       Print "Greater then 10"
    End If
 End Sub
```

Example ex_if.bas: Example how to use the if statement.

See Also

Select...Case statement

While..Wend statement

Executes a group of statements while a condition is **True** .

Syntax

```
While condition [Do]
   [statements]
Wend
```

where *condition* is a numeric expression with value True or False.
statements are one or more statements executed while condition is True.

While loop

This example shows how to use the While..Wend statement.

```
' While..Do loop example

Sub btn_start_clicked()
   Dim i As Integer

   i = 1
   While i < 5
      Print i
      i = i + 1
   Wend

   Print "New loop"

   While i <= 10 Do
      Print i
      i = i + 1
   Wend
End Sub
```

Example ex_while.bas: Example of while loop statement

See Also

Do...Loop Statement

For..Next statement

Repeat a group of statements a specified number of times

Syntax

```
For counter = start_value To end_value [ Step step_value ]
   [statements]
Next counter
```

where *counter* is a numerical variable which may be defined in a Dim Statement
start_value is the initial value of the loop variable
end_value is the final value of the loop variable
step_value is the value the counter will be changed each time through the loop
statements are one or more statements that are executed a specified number of times.

Example

This example shows a simple for-loop.

```
' For loop

Sub btn_start_clicked()
   Dim i As Integer

   For i = 1 To 5
      Print "Value = "+str(i)
   Next i

   For i = 3 To 15 Step 3
      Print i
   Next i
End Sub
```

Example ex_for.bas: Example of For..Next statement.

See Also

Do...Loop Statement , **While...Wend** Statement

Do..Loop and Exit statement

The Do..Loop statement repeates a block of statements until the Exit statement will be called to terminate the loop.

```
' Do..Loop with exit termination

Sub btn_start_clicked()
   Dim i As Integer

   i = 1
   Do
      If i > 10 Then
         Exit Do
      End If

      Print i
      i = i + 1
   Loop
End Sub
```

Example ex_doloop.bas: Show of Do..Loop statement

Exit statement

Provides a way to exit a **Do...Loop** statement. Exit Do transfers control to the statement immediatly following the Loop statement. It can be used only inside a Do...Loop statement. When used within nested Do..Loop statements, Exit Do transfers control to the loop that is the one nested level above the loop where Exit Loop occurs.

See also

End Statement, Do...Loop While and Do..Loop Until Statement

Do..While / Until-Loop statement

The Do-Loop structure repeats a block of statements. The statements in the body of the structure may be repeated **while** a condition becomes tru or **until** a condition becomes true. Any number of **Exit Do** statements may be placed anywhere in the **Do...Loop** structure as an alternate way to exit. If the running program reaches the **Exit Loop** statement control is transfered to the statement immediately following the **Loop** .

When used within nested Do..Loop statements, Exit Do transfers control to the loop that is the one nested level above the loop where Exit Loop occurs.

Syntax

```
Do [{While | Until} condition ]
   [statements]
[Exit Do]
   [statements]
Loop
```

or

```
Do
   [statements]
[Exit Do]
   [statements]
Loop [{While | Until }] condition ]
```

where *condition* is a numeric expression with result True or False and
statements are one or more statements that are repeated while or until condition is True.

Example

This example shows how Do..Loop statements can be used. The loop counts from 1 to 10 and prints the current value in the form.

Do..While example

The select example selects all rows from the table tab1 and prints the value of column col1 for each row.

```
' While..Do..While loop example
```

```
Sub btn_start_clicked()
   Dim i As Integer

   i = 1
   Do While i <= 5
      Print i
      i = i + 1
   Loop

   i = i + 2
   Print "New loop"

   Do
      Print i
      i = i + 1
   Loop While i <= 10
End Sub
```

Example ex_while_do.bas: Example of while..do statement

Do..Until

The select example selects all rows from the table tab1 and prints the value of column col1 for each row.

```
' Until..Do..Until loop example

Sub btn_start_clicked()
   Dim i As Integer

   i = 1
   Do Until i > 5
      Print i
      i = i + 1
   Loop

   i = i + 2
   Print "New loop"

   Do
      Print i
      i = i + 1
   Loop Until i > 10
End Sub
```

Example ex_until_do.bas: Example of the Do..Until statement.

See also

Exit Statement, **For...Next** Statement, **While...Wend** Statement

Select..Case statement

Executes a group of statements depending on the value of an expression.

Syntax

```
Select Case sel_expression
[Case expression_1
   [statements_1]] ...
[Case expression_n
   [statements_n]]
[Case Else
   [statements_else]]
End Select
```

where *sel_expression* is a numeric expression which leads to a compare_value
expression_i is a numeric expression which is compared to the compare_value. If the values are eual the following statements will be executed.
statements_i are one or more statements that will be executed if the depending expression is equal to the compare_value.

Example

This example should show the text "select 2" selected by the value of 'i'.

```
' Select..case program example

Public i As Integer

Sub btn_start_clicked()
   i = 2
```

```
    Select Case i
    Case 1
       Print "select 1"

    Case 2
       Print "select 2"

    Case Else
       Print "Other value"
    End Select
 End Sub
```

Example ex_select_case.bas: Example of Select..Case statement

See Also

If...Then...Else Statement