Subroutine definitions and calls

Contents

Overview

HBasic currently uses only one type of syntax to define subroutines, functions or methods. It does not matter if you want to return a value from a function or not. The subroutine starts with the keyword "Sub" end ends with the keywords "End Sub". Subroutines are the normal entrypoints for all type of code in HBasic. Between normal used defined methods they are also used for event handling destinations or reading or setting properties in user defined classes (See the examples about classes). We already used methods for event handling in the preceeding examples. This document will show the following examples about subroutine syntax.

Simple subroutine call

This example shows how to set up a subroutine definition in HBasic and call it from another code position.

```
' Call function with or without return value

Sub func1()
   Print "Func1 started"
End Sub

Sub func2() As Long
   func2 = 1122
End Sub

Sub btn_start_clicked()
   ' subroutine call with empty parameter list
   func1()

   ' subroutine call without parameter list
   func1

   Print func2()
End Sub
```

Example ex_call_sub.bas: Call subroutine without and with return value.

The subroutine func2 also shows how to return a value from a subroutine. Assign a value to an identifier with the name of the subroutine. This value will be put on the value stack when returning from the subroutine. This example also shows that you may call a subroutine without parameters without any parantheses.

Remarks

- Following the name of the subroutine you can normally find a parameter list within parantheses. If there are no parameter you may also call a subroutine without a parameter list.
- A HBasic subroutine cannot be called with a "Call" keyword. You should directly use the subroutine name.

Subroutine with parameters and return value

When you define a subroutine you may also use a parameter list to pass some parameters at runtime. After executing the subroutine body the subroutine may return a result value. The following example shows the syntax for the parameter list and the return value.

```
' Call function with parameters

Sub add( p1 As Integer, p2 As Integer ) As Integer
   add = p1 + p2
End Sub

Sub btn_start_clicked()
   Print add( 1100, 22 )
End Sub
```

Example ex_call_sub2.bas: Call subroutine with parameters.

Remarks

There is no need to use the *Function* keyword for a subroutine returning a value (See following example).

Other syntax of subroutine definitions

If you like you may replace the keyword "Sub" for a subroutine definition with the keyword "Function" or "Method". This does not change anything to the semantics of the code.

```
' Call subroutine / method / function

Method func1()
   Print "Func1 started"
End Method

Function func2() As Long
   func2 = 1122
End Function

Sub btn_start_clicked()
   func1()

   Print func2()
End Sub
```

Example ex_call_sub3.bas: Define subroutines with other syntax.

Subroutine overloading

This example declares two subroutines with the same name but different paramter lists.

```
Project Project1

Source Form1

' Call overloaded functions

Method func2( p As Double )
   Print "Double value = "+str(p)
End Method

Function func2( p As Integer )
   Print "Integer value = "+str( p )
End Function

Sub btn_start_clicked()
   func2( 3344 )
   func2( 123.456 )
End Sub
```

Example ex_func_overload.bas: Call overloaded subroutine name.

Reference parameter

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

```
' Call function with or without return value

Public g As Integer

Sub func1( ByVal p As Integer )
   p = 3333
End Sub

Sub func2( ByRef p As Integer )
   p = 4444
End Sub

Sub btn_start_clicked()
   g = 1122
   func1( g )
   Print "After func1 "+str(g)
   func2( g )
   Print "After func2 "+str(g)
End Sub
```

Example ex_refpar.bas: Call subroutine with reference parameters.