Values, types and variables

Contents

Overview

Variables and values of different types are one of the basic building blocks of a programming language. Variables may have different types or a different scope. The scope or visibility desribes the parts of the source code where the variable can be used.
HBasic knows four kinds of variable scope.

- global variables which may be used in the whole program.
- variables which may be used in one source (or form) module. They will be called *formlocal* .
- variables which may be used in a class scope (from class .. end class). They will be called classlocal.
- variables local in a subroutine (or method). They will be called *sublocal*.

Before using variables in expressions you normally use a Dim or Public statement in the following way:

Dim v As Integer within a subroutine defines a sublocal variable.
Dim v as Integer within a class (outside a method) definition defines a classlocal variable.
Dim v As Integer outside a subroutine defines a formlocal variable.
Public v As Integer outside a subroutine defines a global variable.

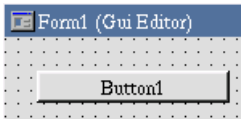
Currently you may declare variables of the following predefined types:

- Byte
- Short
- Integer
- Double
- String
- Object (replaces Variant)

Other possible types for a variable may be Enums, user defined types, predefined components or user defined class names. The types Date and File for example are implemented as components within a package.

Simple values

The first example shows how values of type integer, double or string may be used in the *Print statement*. The Print statement displays the value of the expression following the print token in the current form window. The following examples will all connect the start of the example code to a button_clicked event of a button component. This means you can start the method execution with a mouseclick of the left mouse button on the button widget that will be displayed in the form.




```
' Print some values
'

Sub btn_start_clicked()
   Print "Program started"
   Print "Integer value"
   Print 2223333

   Print "Double value"
   Print 1111.2222

   ' Show more than one value
   Print 11111, 2222
```

```
   Print "Show text"
End Sub
```

Example ex_simple_types.bas: Print some values of different types.

Variable access

The variable example defines some global, formlocal and sublocal variables. These variables will be assigned initial values with an assignment statement and the result will be printed with the *Print* statement. The interpreter and compiler of HBasic will create simple read and write access functions to the variable memory from this code.

```
' Define variables and read or write values

' global variables (public)

Public gl_var1 As Integer
Public gl_var2 As String

' local in module / form

Dim fl_var1 As Integer
Dim fl_var2 As Double

Sub btn_start_clicked()
   ' local in subroutine

   Dim sl_var1 As Integer
   Dim sl_var2 As Double

   gl_var1 = 888
   gl_var2 = "Global var"
   Print gl_var1, gl_var2

   fl_var1 = 333444555
   fl_var2 = 12.3456
   Print fl_var1, fl_var2

   sl_var1 = 3344
   sl_var2 = 456.789
   Print sl_var1
   Print sl_var2
End Sub
```

Example ex_var_types.bas: Write and read variable contents

Variables of type Object

Object variables replace the old type *Variant* in HBasic. Variables of type object may currently store values of type integer, double or string. If you assign a value to a variable which hasn't been defined before in a Dim statement it will get the type *object* automatically (Variable i in the following example). You may also define a variable of type object in a Dim statement (*variable v* in the example).

```
Sub button1_clicked()
   i = 1234
   Print i

   i = "Hello object variable"
   Print i

   Dim v As Object

   v = "Predefined variablt of type object"
   Print v
End Sub
```

Example ex_object.bas: Create and use object variable

Operator example

One or more variables and values may be combined in an expression with operators. Operators normally need matching types for the operand on the left and right side to work correctly. If the operands are not of the same type HBasic will try to convert them. The following conversions will be started automatically:

- Object values will be converted to normal values of type int, double or string.
- Integer values will be converted to double if the second operand is of type double.
- Integer or double values will be converted to a string if the second operand is of type string.

The following examples shows expressions with different type of operators.

```
' Use operators for simple types

Sub btn_start_clicked()
   Dim v1, v2, v3 As Integer

   v1 = 2
   v2 = 3
   v3 = 4

   Print "Add v1 + v2 = "
   Print v1 + v2

   Print "v1 * v2 = "
   Print v1 * v2

   Print "Operator priority"
   Print "v1 + v2 * v3 = "
   Print v1 + v2 * v3

   Print "(v1 + v2) * v3"
   Print (v1 + v2) * v3

   Print "Concatenating strings aaa + bbb"

   Dim s1, s2, s3 As String
   s1 = "Hello "
   s2 = "HBasic "
   s3 = "world!"

   Print s1 + s2 + s3
End Sub
```

Example ex_operator.bas: Example of some operators in expressions.

Predefined functions

HBasic provides some predefined functions/methods to change values of variables. You can can find a list of the available functions in the function list. The following examples calls some functions that are build into HBasic.

```
' Using predefined functions

Public g1 As Integer
Public g2 As Double
Public g3 As String

Const pi = 3.1415926535

Sub btn_start_clicked()
   g2 = pi

   Print "sin, cos ..."
   Print sin( pi )
   Print cos( g2 )

   Print "String methods"
   g3 = "aaabbb"
   Print left( g3, 3)

   msgbox( "Title", "Message box" )

   g1 = -2
   Print abs( g1 )

   Print "Overloading methods"
   Print "Integer str()" + str( 111 )
   Print "double str()" + str( 111.222 )
End Sub
```

Example ex_predef_functions.bas: Call predefined HBasic functions.

Enum definition

With the *Enum* statement you can define a new type of enumerated values. After you have defined a new Enum type you can declare variables of this type. The following example shows an Enum statement used to define a list of weekdays.

```
' Define and use Enum values

Enum days
   monday
   tuesday
   thursday = 4
   friday = 11
   saturday
   sunday
End Enum
```

```
Public var1 As days

Sub btn_start_clicked()
   Print "Enum values"

   var1 = tuesday
   Print "Tuesday = ", var1

   var1 = friday
   Print "Friday = ", var1

   var1 = sunday
   Print "Sunday = ", var1
End Sub
```

Example ex_enum.bas: Define Enum type and use it.

User defined types / structures

The *Type* keyword is used to declare a user defined data structure. The type statement describes only the information about the structure. You have to use the Dim statement to actually create variables of the new type.

The following examples shows how to declare a data structure with the type statement and use a variable declared of this type.

```
' Use of user defined types

Type smalldef
  i1 As Integer
  i2 As Integer
End Type

Type mydef
  ival As Integer
  sval As smalldef
  dval As Double
  aval(10) As Integer
End Type

Public var1 As mydef

Sub btn_start_clicked()
   var1.ival = 111
   var1.dval = 333.444555

   ' Use of array within structure

   var1.aval(4) = 222
   var1.aval(6) = var1.aval(4) + 333

   ' Use of structure within other structure

   var1.sval.i1 = 1234
   var1.sval.i2 = 4444

   Print "Read type values"

   Print var1.ival
   Print var1.dval

   Print var1.aval(4)
   Print var1.aval(6)

   Print var1.sval.i1
   Print var1.sval.i2
End Sub
```

Example ex_usertype.bas: Create user defined type and access values in it.

Constant definitions

You can use constant definitions to represent better readable values within your source code. A constant is a way to associate a value with a name which is easier to remember. This is helpful when trying to write better readable programs. Beside predefined constants of the HBasic program you may define your own constants by using the "CONST" keyword.

```
Const maxnum = 1234
Const pi = 3.141592

Sub btn_start_clicked()
   Const ctext = "Hello"
   Print "Const integer = ", maxnum
   Print "Const double = ", pi
   Print "Const string = ", ctext

End Sub
```

Example ex_const.bas: Defining and using constants

The scope of a constant definition is the whole program. You may define and use a constant anywhere in your source code before or behind the line defininig the value of the constant. Currently constant definitions may only be used for the following types of values:

Integer values	1, 555, 12345
Double values	3.1415, 123.456E10
Strings	"Test", "Hello world"

The syntax of a constant definition is

CONST <const-idf> = <const_value>

<const-idf> is the identifier which represents the constant in the rest of the program.
<const_value> is a constant value of type Integer, Double or string.

Array access

This examples shows how an array can be defined within a Dim statement and how you may access the components of an array in an expression. Currently you can only use arrays with predefined size. This means the size of the array cannot be changed dynamically at run time.

The following example shows how to define and access an array.

```
' Array access

Public a1( 10,10 ) As Integer
Public a2( 20 ) As Double

Sub btn_start_clicked()
   Print "Program started"
   a1(2,4) = 111
   a1(3,6) = 222

   a2( 4 ) = 111.111
   a2( 6 ) = 44.4444
      Print "Array1 type integer, two dimensions"
   Print a1(2,4), a1(3,6)
      Print "Array2 type double, one dimension"
   Print a2(4 ), a2(6)
End Sub
```

Example ex_array.bas: Use one and twodimensional array.