

Variables in Yabasic 3 (some preliminary ideas)

Thomas Larsen

August 18, 2010

Hello all

- Here are some of my suggestions for how variables could work in Yabasic 3.
- Your comments and questions would be greatly appreciated! Just respond at the Yabasic blog: <http://yabasic.basicprogramming.org/blog>.
- Nothing has been concretely decided upon yet. All input is welcome.
- Please let me know if you think anything major is missing.

Variables

- Variables identify values.
- Variables are usually typeless:
 - ◆ variables with the suffix “\$” appended to their names can only identify character string values
 - ◆ variables without a “\$” can identify values of any type.
- Examples of variables:
`pi = 3.1415`
`name = "Bob"`
`item$ = "can of beans"`

Values

- Two types of values:
 - ◆ real numbers (for example, 0, 1, 3.14, -753)
 - ◆ character strings (for example, the empty string "", "z", "Hello, world!", "oranges").

Scalars

- Scalars are variables that can hold only one value:

```
town = "Tinyville"
```

```
population = 100
```

Arrays

- Arrays are variables that can hold more than one value:
town (1) = "Randomville"
population (1) = 242
town (2) = "Somewhereton"
population (2) = 583
- The position of an element in an array is specified using indices.
- Arrays can have any number of dimensions:
array_1D (2) = 1
array_2D (3, 4) = 2
array_3D (5, 6, 7) = 3
array_4D (8, 9, 10, 11) = 4
- Arrays expand and contract dynamically.

Associative arrays

- Array indices can be:
 - ◆ integers (positive or negative)
 - ◆ character strings
 - ◆ a combination of integers and character strings.
- Indices may be expressions that return these types of values.
- The order of array indices *does* matter.
- Examples of associative arrays:

```
crate ("fruit", "apples") = 93
population ("Random" + "ville") = 242
student_subject ("Alice", 1) = "CS101"
```

Records

- Records are implemented using associative arrays:
 - ◆ `a.b` is syntactical sugar for `a ("b")`
 - ◆ note that `a.b` is *not* equivalent to `a (b)`.
- Both `a` and `b` must be valid symbols for this to work.
- Examples of records:

```
crate.fruit ("oranges")
crate ("fruit").oranges
crate.fruit ("oranges")
```

All of these actually represent:

```
crate ("fruit", "oranges")
```

Initialisation

- Variables are dynamic.
- Neither scalar nor array variables need to be declared before use.
- However, an error will be reported if a program tries to access a variable that has not previously been assigned a value.
- The operator “defined” can be used to preempt this situation:

do

```
    if not (defined (iterator)) then
        iterator = 1           // initialise "iterator"
    else
        iterator = iterator + 1 // increment "iterator"
    end if
```

loop

Subroutine parameters

- Parameters passed to subroutines are stored in variables:
values (1, 2, 3) // output will be "1 2 3"

```
sub values (a, b, c)
  print a, b, c
end sub
```

- If too few parameters are passed to a subroutine, one or more variables will be left undefined.
- Scalars are always passed by value, arrays are always passed by reference.

Globals, locals, statics

- Variables initialised in the main body of a program (not in a subroutine) will always be globals.
- Global variables are accessible from any subroutine that does not have a local or static variable with the same name.
- Local, or static, variables may be declared in a subroutine:
`local a, b`
`static c`
- All subroutine parameters are local.
- A static variable is just a local variable that keeps its value (or values, if it is an array) between subroutine calls.