CS 5142
Scripting Languages

8/30/2013

End User Programming (VBA)

About VBA

- Visual Basic for Applications
  - Visual = create GUI by drag’n’drop
  - Basic = simple, for end users
  - for Applications = embedded in Word, Excel, Powerpoint, AutoCAD, Corel Draw, Acrobat
- Object-oriented, event-driven
- Aimed at end user programming
  - Automate task user does by hand otherwise
  - Same niche as Emacs Lisp, AppleScript

Related Languages

- VB6
  - Same language as VBA, different libraries
  - "End of life", not supported on new Microsoft platforms
- VBScript
  - Subset of VB 6.0, for server-side scripts
- VB.NET
  - Not backward compatible, based on CLR

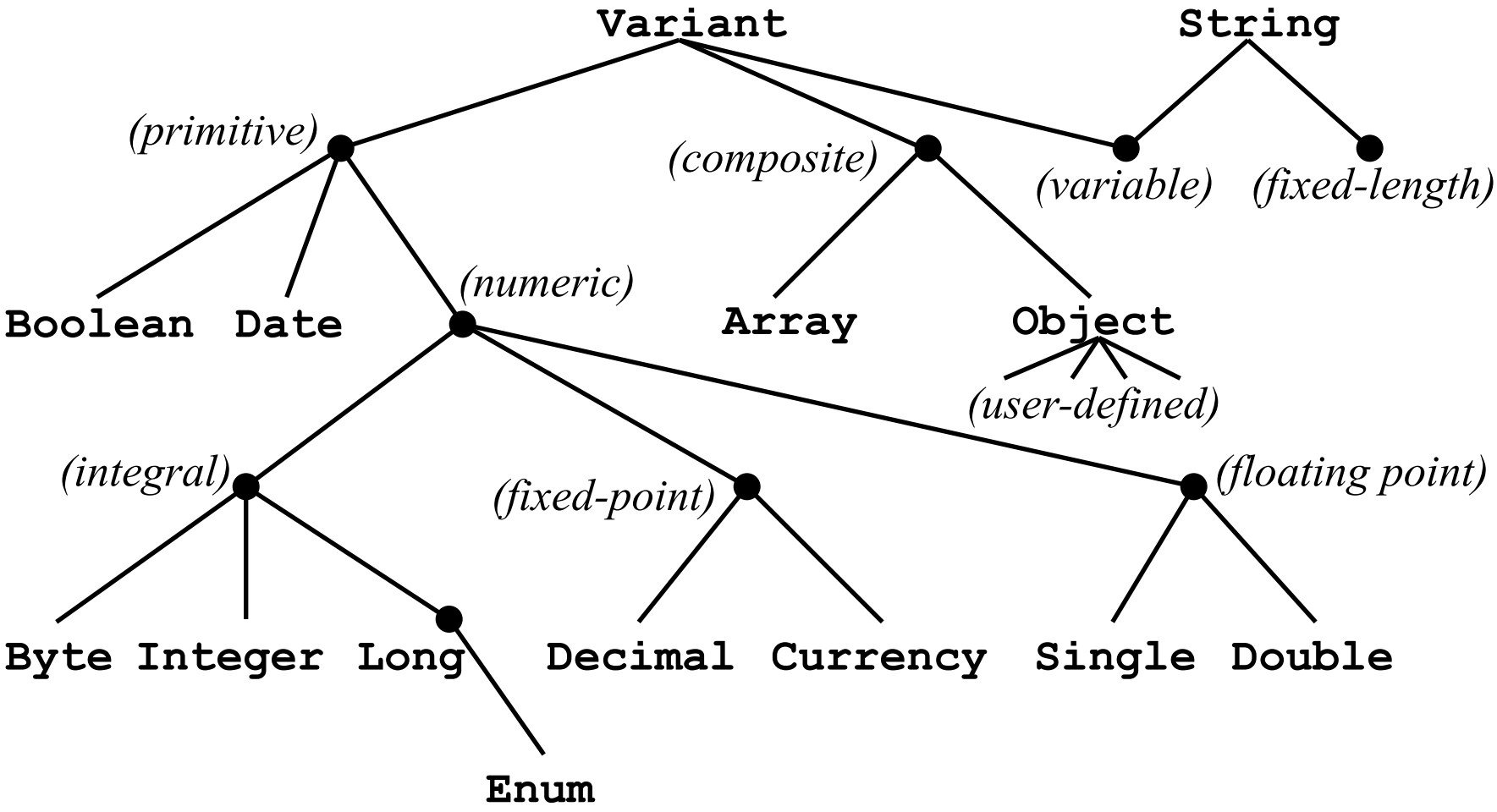
How to Write + Run Code

- Open Word, PowerPoint, or Excel
- Visual Basic Editor (Alt-F11)
  - Immediate window (edit-eval-print loop)
  - Code editor, including auto-completion
  - Debugger, object browser
  - Help system
- Macro Recorder
  - Recorded macros are useful code blue-print
- Customization
  - Add hand-written code to toolbars or menus

Lexical Peculiarities

- Case insensitive
- Line break sensitive
  - One statement, multiple lines: **`_`**
  - One line, multiple statements: **`:`**
  - Single-line comments: **`’`**, **`Rem`**
- Literals
  - Boolean: **`True`**, **`False`**
  - Date: **`#April 20, 2008#`**, **`#10:15pm#`**
  - Variant: **`Empty`**, **`Error`**, **`Nothing`**, **`Null`**
  - Character: **`vbCr`**, **`vbTab`**, **`vbLf`**, **`Chr(149)`**, …

Types


Variant
String
(primitive)
(composite)
(variable)
(fixed-length)
Boolean
Date
(numeric)
Array
Object
(user-defined)
(integral)
(fixed-point)
(floating point)
Byte
Integer
Long
Decimal
Currency
Single
Double
Enum

Variable Declarations

<table>
<tr><td>Implicit, untyped</td><td>fruit = "apple"</td><td rowspan="2">Type declaration characters:
% Integer, & Long,
@ Currency, ! Single,
# Double, $ String</td></tr>
<tr><td>Implicit, typed</td><td>fruit$ = "apple"</td></tr>
<tr><td>Explicit, untyped</td><td>Dim fruit</td><td rowspan="2">Explicit declaration keywords:
Dim, Private,
Public, Static
Enforce: Option Explicit</td></tr>
<tr><td>Explicit, typed</td><td>Dim f As String</td></tr>
<tr><td>Constant</td><td>Public Const pi_
As Byte = 3</td><td rowspan="5">Optional: As type</td></tr>
<tr><td rowspan="4">Array</td><td>Dim a1()</td></tr>
<tr><td>Dim a2(10)</td></tr>
<tr><td>Dim a3(5 To 10)</td></tr>
<tr><td>Dim a4(10, 5)</td></tr>
</table>

Gradual and Dynamic Typing

- *Gradual typing*: optional in declaration
- If not specified: **`Variant`**
  - *Dynamic typing*: check at the last moment whether runtime value legal for operation
  - E.g., same variable can hold an **`Integer`** at one time, and a **`String`** at another
- If specified:
  - *Static typing*: check at compile time
  - Compiler can optimize time+space

Arrays

- Indexing with round parentheses "**()**"
- Statements
  - **ReDim** [**Preserve**] *id* **(**[*new sizes/bounds*]**)**
  - "**Preserve**" keeps old values
  - **Erase** *id*
- Functions
  - **IsArray(***id***)**
  - **LBound(***id*[**,** *dim*]**)**, **UBound(***id*[**,** *dim*]**)**
- Options
  - "**Option Base 1**" overrides default base-0 indexing

BNF = Backus Naur Form

Construct	Description	Example
Terminal	concrete code	**`Courier Bold`**
Non-terminal	placeholder	*Times Italic*
Ellipsis	omitted code	…
Rule	non-terminal definition	*Lhs* ::= *Rhs*
Alternative	choose one	*Alt1* | *Alt2*
Optional	zero or one times	[Square brackets]
Repeat	zero or more times	Kleene star*
Repeat	one or more times	Kleene plus^{+}
Grouping	treat as unit	(parentheses)

Input and Output

- **`Debug.Print "Hello, world!"`**
- **`Application.StatusBar = "Hello, world!"`**
- **`MsgBox "Hello, world!"`**
  - **`MsgBox(`***prompt*[, *buttons*][, *title*]**`) As Long`**
  - Buttons: 0 vbOkOnly, 1 vbOkCancel, 2 vbAbortRetryIgnore, 3 vbYesNoCancel, 4 vbYesNo, 5 vbRetryCancel
  - Result: 1 vbOk, 2 vbCancel, 3 vbAbort, 4 vbRetry, 5 vbIgnore, 6 vbYes, 7 vbNo
- **`userName = InputBox("Who are you?")`**
  - **`InputBox(`***prompt*[,*title*][,*default*][,*xpos*][,*ypos*]**`) As String`**

Concepts

Operator Characterization

(…)			Call; Indexing
^	2	L	
+, -	1		
*, /	2	L	Multiplicative
\	2	L	Integer division
Mod	2	L	Modulu
+, -	2	L	Additive
&	2	L	String
<<, >>	2	L	Bit shi
=, <>, <, <=, >, >=, Is	2	L	Comparison
Not	1		Negation
And, Or, Xor, Eqv, Imp	2	L	Logic (not all same precedence)
[Set] … = …	2		Assignment statement

Arity:
1 = unary
2 = binary

Associativity:
L = left
R = right

Precedence:
from high
to low

Concepts

Arity, Precedence, Associativity

Arity	Number of operands	`-2` `2 - 2`	unary binary

Precedence	Binding strength	`2+2*2`
		~~`(2+2)*2`~~
		`2+(2*2)`
			`*` has higher precedence than `+`

Associativity	Grouping direction	`2/2/2`
		`(2/2)/2`
		~~`2/(2/2)`~~
			`/` is left-associative

Precedence and associativity in programming usually follows the conventions from math.

Operators

`(…)`			Subexpression; Call; Indexing
`^`	2	L	Exponentiation
`+`, `-`	1		Identity, negation
`*`, `/`	2	L	Multiplicative
`\`	2	L	Integer division
`Mod`	2	L	Modulus
`+`, `-`	2	L	Additive; String concatenation
`&`	2	L	String concatenation
`<<`, `>>`	2	L	Bit shift
`=`, `<>`, `<`, `<=`, `>`, `>=`, `Is`	2	L	Comparison
`Not`	1		Negation
`And`, `Or`, `Xor`, `Eqv`, `Imp`	2	L	Logic (not all same precedence)
`[Set]` … `=` …	2		Assignment statement

Control Statements

Conditional	**If** … **Then** … **ElseIf** … **Else** … **End If** **Select Case** *expr* … **Case Else** … **End Select**
Fixed-iteration loops	**For** *id* **=** *expr* **To** *expr* **Step** *expr* … **Next** *id* **For Each** *id* **In** *expr* … **Next** *id*
Indefinite loops	**Do** … **Loop** **While** *expr* … **Wend** **Do While** *expr* … **Loop** **Do** … **Loop While** *expr* **Do Until** *expr* … **Loop** **Do** … **Loop Until** *expr*
Unstructured control	**GoTo** *id / line number* **Exit Do** **Exit For** **Exit Function** **On Error GoTo** *id* **Err.Raise** *number* **Resume** *id / line numbe*

Writing Subroutines

- Declaration
  - *mods** **Sub** *id* [**(***arg**)**] … **End Sub**
  - *mods** **Function** *id* [**(***arg**)**] [**As** *type*] … **End Function**
  - To return a value, assign to function name, else default
- Function modifiers
  - **Public**, **Private**: visibility outside module
  - **Static**: make all locals static
- Arguments: *mods* id* [**()**] [**As** *type*] [**=** *expr*]
- Argument modifiers
  - **Optional**: after first, rest must also be **Optional**
  - **ByRef**, **ByVal**: default **ByRef**
  - **ParamArray**: must be last, allows for var-args

Concepts

Positional vs. Named Parameters

```
Sub bake(Optional Amount=1, Optional Dish="pizza") … End Sub
```

```
bake(          2,          "cakes")
bake(          2                  )
bake(            ,          "cakes")
```

positional

```
bake(Amount:=2, Dish:="cakes")
bake(Amount:=2                )
bake(                Dish:="cakes")
bake(Dish:="cakes", Amount:=2)
```

named

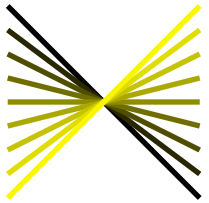
Call syntax:

- How to omit optional parameters
- Whether or not order matters
- Note: parentheses not required

Library Functions

- Simple data conversion: CBool, CByte, CCur, CDate, CDbl, CInt, CLng, CSng, CStr, CVar
- Complex data conversion: Asc, Chr, Format, Hex, Oct, RGB, QBColor, Str, Val
- String: InStr, InStrRev, LCase, UCase, Left, Len, LTrim, RTrim, Mid, Right, Space, StrComp, StrConv, StrReverse, Trim
- Math: Abs, Atn, Cos, Exp, Fix, Log, Rnd, Sgn, Sin, Sqr, Tan, IsNumeric
- And many more

Example, Revisited



```
Option Explicit

Sub LemonStar()
  Dim S As PowerPoint.Slide
  Set S = ActivePresentation.Slides( _
    ActivePresentation.Slides.Count)
  Dim I As Integer
  For I = 0 To 8
    Dim L As PowerPoint.Shape
    Const Dpi As Integer = 72 ' 72 dots per inch
    Set L = S.Shapes.AddLine( _
      BeginX:=Dpi*5, BeginY:=Dpi*3.75+I*Dpi/8, _
      EndX  :=Dpi*6, EndY  :=Dpi*4.75-I*Dpi/8)
    L.Line.ForeColor.RGB = RGB(I * 31, I * 31, 0)
  Next I
End Sub
```

Reference

VBA Documentation

- Included:
  - Macro recorder
  - Auto-completion
  - Help system
  - Object browser
- Book: Mastering VBA, 2nd edition.
  Guy Hart-Davis. Wiley, 2005.
- Online:
  MSDN Library→Development Tools+Languages
  →Visual Studio 6.0→VB 6.0

Soap-box

Evaluating VBA

Strengths

- Development environment
- Simplicity
- Availability
- Popularity
- Best tool for end users of MS Office

Weaknesses

- Single platform/vendor
- Binary format
- Security
- Missing features
  - Structured exceptions
  - Implementation inheritance
  - Regular expressions
  - Associative arrays

Administrative

Announcements

- No class on Monday
- Office hours on Wednesday after class

- We have a TA:
  Ozan Irsoy (oirsoy@cs.cornell.edu)

Administrative

Last Slide

- First homework is due on Friday, Sept. 6th
- Today's lecture
  - Dynamic typing
  - Precedence and associativity
  - Basics of VBA
- Next lecture
  - Putting the V and the A in VBA
  - Application extension
  - Properties
  - Call-backs