

Bio 595

Computers in

Biomedical Research

Class notes set 7

Fall 2003

The Debugger

The “debugger” in VB consists of various tools and options in the IDE (Integrated Development Environment) that allow program problems to be identified and fixed.

Kinds of Errors

Common bug: the 'syntax error', a misspelled VB keyword or a grammar error.

These errors are reported upon entry .

Sometimes the error reported and highlighted is misleading .

E.g. 'compile' errors refer to a variety of unintuitive mistakes.

Look in the immediate vicinity of a highlighted line to find the error.

Kinds of Errors

Another bug category: 'runtime error' detected only when the program is executed.

'Divide by zero' is such an error because the zero value taken by a variable isn't known until the calculation is attempted during program execution.

Runtime error dialog box gives four button choices:

continue (continue the program execution if possible; may be dimmed and inoperable with fatal errors)

End (stop program execution, return to design time mode).

Debug (enter the debugger window for help with details about the error type.

Selecting Debug can possibly allow a fix and a resumption of the program without restarting.

Most Difficult Kind of Error

“Logic” errors: flaws in program's design.

Best corrected by using the debugger to trace program execution.

Note VB Debug pull-down menu choices.

Debugger allows insertion of "breakpoints", temporary halts in program execution to allow the user to examine the program's status, or to change values of program variables.

Breakpoints

In 'break mode', your interrupted program retains all variable values at the moment of interruption, allowing you to examine data values from one line of code.

You often can determine the source of an error by comparing actual values of variables with the values you *thought* they would have.

Breakpoints

Set a breakpoint during runtime program execution using the control + break keys, or the Run pull-down menu 'break' feature during program execution.

In design mode or break mode, you can set a breakpoint on a particular line of code where you want execution to halt.

The Debug/Add watch command lets you specify an expression that will cause a break when the expression becomes true.

What to Do at a Breakpoint?

Move your mouse to a variable to get a tool-tip-like pop-up message giving the current value (at breakpoint) of that variable.

Or, select the Debug/Quick Watch option, a dialog box showing the breakpoint line, variable name, and current value.

Another option: select the Watch dialog box, to track multiple variables that are continuously updated as the program is executed (to the next occurrence of the breakpoint).

Another option: display the Debug toolbar, a floating toolbar (View/Toolbars/Debug).

Single Stepping through Code

Once a breakpoint is reached, you can execute a single line at a time with Debug/Step into command.

Step Over allows jumping over repeated calls to a subroutine.

Step Out exits a current procedure without single stepping.

In the Immediate Window, you can enter VB commands directly, and have them executed.

Print statements executed here causes specified variables to be printed to the immediate window.

Times and Dates

Date data type: day is basic unit of measurement

Hour = $1/24$ of a day

Second = $1/86,400$ of a day

Day 1 = December 31, 1899

**July 4, 1776 = -45103 (45,103 days before
December 30, 1899)**

So, 25001.5 = noon on June 12, 1968

2 PM on May 15, 1998 = 35390.58333

The Timer

Timer control is a clock that fires a programmable event at a specified interval.

Timer event: TimerName_Timer()

Assign value to Timer's "Interval" property

Interval property units: the millisecond.

So if Timer1.Interval=500, clock fires every 0.5 second.

Shortest possible interval is 55 msec because clock ticks only 18 times/sec.

Put the timer anywhere: it isn't visible at runtime.

Timer Properties

Name: default is Timer1, etc.

Enabled: turns timer on and off; True = on.

Interval: timer firing event (in milliseconds).

Three functions: Time, Date, Now

Get the time and convert it into a string:

```
TextTime.text=CStr(Time)
```

Get the date and convert it into a string:

```
TextDate.text=CStr(Date)
```

Get the date and time, convert them into a string:

```
TextNow.text=CStr(Now)
```

Format() Function with Timer

**strShowThis= Format(expression,format_string,
FirstDayofWeek,FirstWeekofYear) where**

strShowThis is the return value

expression is the string reporting a date or time

**Format_string is the template telling the function
how you'd like the output to appear**

**FirstDayofWeek is optional constant setting first
day of the week (default=Sunday)**

**FirstWeekofYear is optional constant (Jan 1
week is default)**

Format() for time and date

Format(36000,"Long Date" Friday, July 24, 1998

Format(36000,"Medium Date" 24-Jul-98

Format(36000,"Short Date" 7/24/98

Format(0.874,"Long Time") 8:58:34 p.m.

Format(0.874,"Medium Time") 08:58 p.m.

Format(0.874,"Short Time") 20:58

Subroutines

VB is a procedural language: it can execute blocks of code referred to by name; the block is a “subroutine”

Compare subs (subroutines) and functions: subs don't return values, while functions do.

Sub example:

```
Private Sub Draw_Click()  
(lines of code to execute)  
End Sub
```

Subs can be written into General Declarations area

Functions

A function is a procedure that executes lines of code and returns a value. The syntax is:

```
[Private or Public] Function FunctionName As DataType  
(lines of code)  
FunctionName=ReturnValue  
End Function
```

FunctionName is name you've assigned the function

DataType is type of data function will return

ReturnValue is value passed back from the function - assign it the function's name

Passing Arguments

An argument (or parameter) is a variable used to hold a value to be passed into a subroutine or a function.

Place these in the parentheses of the declaration statement of the sub or function.

Example:

PH(sngConc As Single, intTemp As Integer) As Single

Make sure argument types and order match up.

VB Math Functions

Abs Absolute value (eg $b = \text{Abs}(a)$)

Atn Returns arctangent of a number in radians
(eg $z = \text{atn}(x)$)

CInt Converts the argument to an integer (eg
 $\text{CInt}(x)$ returns 1 if $x = 1.25$)

Cos Returns the cosine of an angle specified in
radians

Exp Raise the base of the natural logarithm e to
the (argument) power (eg $y = \text{exp}(x)$)

Int Converts a number to the Integer data type
(eg $x \% = \text{Int}(y)$)

VB Math Functions

Log Returns the natural log of the argument (eg $y=\log(x)$)

Mod The modulo arithmetic operator, gives the remainder of a division (eg if $B=7$ and $C=2$, $B \text{ Mod } C$ yields 1.

Randomize Initializes the seed of the random number generator; use before **Rnd()** function

Rnd Function returns a Single precision number between 0 and 1

VB Math Functions

Sgn Returns the sign, + or -, of the number argument (+1, -1, or 0 if argument is 0)

Sin Returns the sine of an angle, expressed in radians

Sqr Returns the square root of the number argument. (eg $x = \text{sqr}(4)$ would yield $x = 2$)

Tan Returns the tangent of an angle expressed in radians.