

Intermediate Perl

Packages, References, Objects, and Modules

an intermediate-level course with exercises
by Tom Phoenix and Randal Schwartz
Stonehenge Consulting Services, Inc.

Version 2.2.0 (18 Jan 2006) [E]

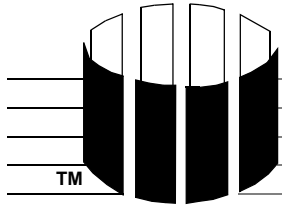


Table of Contents

Introduction

About this course	3
-------------------------	---

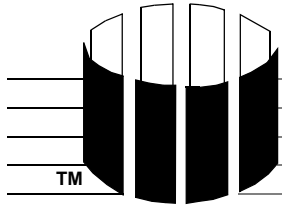
Perl Libraries

The cure for the common code	5
Good code, <code>eval</code> code	6
do the right thing	7
Checking out multiple libraries	8
Using <code>require</code>	9
How to find the library	10
Extending <code>@INC</code>	11
Namespace collisions	12
Using name prefixes	13
Using the <code>package</code> directive	14
About package names	15
Scope of a <code>package</code> directive	16
The bounding <code>main</code>	17
Package names and variables	18
Package names and filenames	19
Declaring global variables	20
The <code>our</code> directive	21
A complete library file	22
Outside the library	23

Exercises	24
Exercise answers	25

Introduction to References

Problem: Passing an array to a sub	28
Introducing PeGS—Perl Graphical Structures	29
References to arrays	30
Accessing an array through a reference	31
Dereferencing the reference	32
Using the array parameter	33
Additional ways to access an array	34
Hash references	35
Scalar references	36
The converted forms of a reference	37
Simplifying complex references	38
Modifying the array	39
Nested data structures	41
Using the nested data structure	42
References, piece by piece	43
Dereferencing with the small arrow	44
Omitting the little arrow	45
Exercises	46
Exercise answers	48



References and Scoping

References to arrays	50
One variable, many names	51
Going out of scope	52
Returning a ref from a sub	53
The anonymous array constructor	55
The anonymous hash constructor	56
A typical nested data structure	57
A hash of array references	58
Working with the hash of arrays	59
Auto-vivification	60
Auto-vivification and hashes	61
Displaying the multi-level data	63
Destruction	64
Implications of reference counting	65
Orphaned memory	66
Soft references	67
The <code>ref</code> function	68
Exercises	69
Exercise answers	70

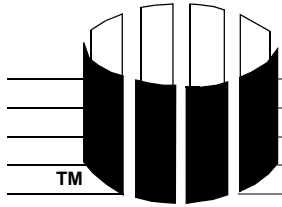
Manipulating Complex Data Structures

Perl's built-in debugger	73
The debugger's help system	74
Stepping through code	75
Watching auto-vivication happen	76

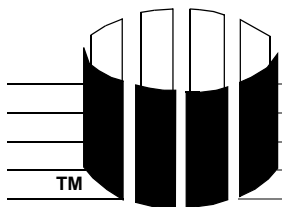
Debugger command history	77
Breakpoints	78
Getting lost, getting found	79
Executing Perl code	80
Review of <code>grep</code>	81
Review of <code>map</code>	82
Using a block with <code>grep</code> or <code>map</code>	83
Mapping one to many	84
Using <code>grep</code> or <code>map</code> with a nested data structure	85
Printing a structured variable for debugging	86
Saving and restoring complex data structures	87
Exercise	88
Exercise answer	89

Subroutine References

Coderefs	91
Having a conversation	92
Making a coderef	93
Using multiple coderefs	94
The gang's all here	95
Anonymous subroutines	97
Callbacks	98
Lexical variables in a surrounding scope	99
Closures	100
Returning a closure	101
Sharing variables among closures	103



Using the shared variables	104	Invoking the entire barnyard	132
Closure variables as inputs	105	The extra parameter of method invocation	133
Closures and BEGIN	106	Calling a second method to simplify things	134
END blocks	107	Inherit the windpipes	135
Exercise	108	What it takes for a Cow to speak	136
Exercise answer	110	Here @ISA note or two about @ISA	137
Practical Reference Tricks		Specialized ways of speaking	138
The problem: Sorting by a computed value	112	Avoiding code duplication	139
Using references to hold the criteria	113	A better way of going up	140
Sorting the references	114	A SUPER way of doing things	141
The Schwartzian Transform	115	Exercises	142
Make your own Schwartzian Transform	116	Exercise answers	143
Sorting by multiple keys	117	Objects with Data	
Recursive data structures	119	A Horse is a Horse, of course, of course (or is it?)	147
Generating the recursive data	120	Invoking an instance method	148
Designing the output	121	How to build a horse	149
Printing the recursive data	122	The revised Animal class	150
Exercises	123	Making a method work with classes or instances	151
Exercise answers	125	Adding parameters to a method	152
Introduction to Objects		What if an instance needs more data?	153
Introduction	128	The updated Animal class, with hashref instances	154
If we could talk to the animals	129	Changing and getting the color	155
Now out to pasture	130	Getting your deposit back	156
Introducing the method invocation arrow	131	Making the getter do double duty as the setter	157
		There's only one way to do it	158



You've got to have class	159
The croak and carp functions.....	160
Exercise	161
Exercise answer	162

Object Destruction

Beating a dead horse	166
Making a dead horse become useful	167
Alternate notation for class or instance method invocation.....	168
Additional instance variables for subclasses	169
Using class variables	172
Retrieving class variables	173
Dying on schedule	175
Weakening the argument	176
Exercise	178
Exercise answer	179

Some Advanced Object Topics

UNIVERSAL methods	182
When isa Horse not a Horse?	183
What can you do for me?	184
The final resort.....	185
Deferring compilation until necessary	186
Notes about deferring compilation	187
AUTOLOADing attribute accessors	188
Multiple inheritance.....	190

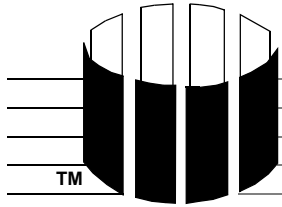
References to filehandles.....	191
Many filehandles at once.....	192
Exercise	194
Exercise answer	196

Using Modules

About modules.....	198
Function-oriented interface	199
Selecting items for import	200
Empty import lists	201
Object-oriented interface	202
What use is really doing.....	203
Inside the module.....	204
use and @INC: The right path at the right time.....	205
Importing with Exporter.....	206
Using @EXPORT and @EXPORT_OK	207
Exporting from OO Modules.....	208
Exercise	209
Exercise answer	210

Writing a Distribution

From module to distribution	214
Using h2xs to start writing your module	215
The MANIFEST template	216
The README template	217
The Changes template	218



The module template	219	Using the test harness	247
Export control	220	Using Test::More	248
Planning for forward-compatibility	221	The <code>ok</code> test function	249
Sacrificial versions	222	The <code>is</code> test function	250
Where to find the <code>\$VERSION</code>	223	The <code>cmp_ok</code> test function	251
The heart of the module	224	The <code>like</code> test function	252
Embedded documentation	225	Object-oriented test functions	253
Introduction to pod	229	Rarely-needed test functions	254
Pod text and directives	230	Planning ahead	255
Formatted lists in pod	231	Exercise	256
A few pod tags	232		
Pod translators	233	Contributing to CPAN	
The joy of <code>Makefile.PL</code>	234	The Comprehensive Perl Archive Network	258
Alternate installation locations	235	Getting started	259
Module prerequisites	236	Preparing the distribution	260
Testing, one, two, three	237	Uploading to PAUSE	261
The default test file	238	Announcing the distribution	262
Installing the module	239	The smoke test	263
Using the alternate library location	240		
Including foreign code	241	Conclusion	
Give away your code	242	Questions and answers	265
Exercises	243		
Essential Testing			
Writing a healthy test suite	245		
Sample tests for a simple function	246		