Ordering Information for Deitel Perl Products:

Perl How to Program, 1/e

The Complete Perl Training Course, 1/e (includes the interactive multimedia CD, the *Perl Cyber Classroom*, and the 1000+ page textbook, *Perl How to Program*)

- View *Perl How to Program's* **Table of Contents**
- Download the **Code Examples**

To view all the Deitel products and services available, visit the Deitel Kiosk on InformIT at **www.informIT.com/deitel**.

To follow the Deitel publishing program, sign-up now for the *DEITEL™ BUZZ ONLINE* e-mail newsletter at **www.deitel.com/newsletter/subscribeinformIT.html.**

To learn more about our **Perl programming courses** or any other Deitel instructor-led corporate training courses that can be delivered at your location, visit **www.deitel.com/training** or contact our Director of Corporate Training Programs at (978) 461-5880 or e-mail: **christi.kelsey@deitel.com.**

Note from the Authors: This DEITEL™ article is an excerpt from Chapter 9, Section 9.3 of *Perl How to Program* and introduces some basic concepts in string manipulation. Readers should be familiar with the basics of Perl syntax, including how to create and print strings. The code example included in this article shows readers an example of here documents, which allow the developer to display several lines of output easily. The code example teaching programming using the Deitel™ signature *LIVE-CODE™ Approach*, which presents all concepts in the context of complete working programs followed by the screen shots of the actual inputs and outputs.

9.3 "Here" Documents

Perl allows programmers to quote blocks of code. A *here document*, such as

```
<<identifier;
line of text
line of text
identifier
```

enables programmers to create or display a string over any number of lines of code. A programmer defines the beginning of a here document with **<<**, followed by an *identifier* and semicolon. Note that there is no space between the **<<** and the *identifier*. Perl quotes the subsequent lines and stops only when the interpreter reaches the *identifier* for a second time. This closing *identifier* is sometimes known as the here document's *terminating string*. For Perl to recognize the closing *identifier*, it must be unquoted and at the beginning of the line; no other code may be placed on this line, including the *end-of-file marker*, that is, a special marker that ends a file. If the closing *identifier* is on the last line of the program, there must be a carriage return. Note that, while the opening *identifier* may be quoted, the closing *identifier* may not be.

Common Programming Error 9.1

A space between the **<<** *and the identifier at the beginning of a here document is a logic error. The space causes the Perl interpreter to cease quoting at the first newline.*

Common Programming Error 9.2

When closing a here document, the identifier must be at the beginning of its own line of code. If there are any other characters on this line, including whitespace, the here document will not be closed.

An optional set of quotes surrounding the opening *identifier* results in different properties in the here document. Figure 9.3 demonstrates these properties. If the opening *identifier* is unquoted (line 10) or double quoted, the resulting string inherits the properties of a double-quoted string, including interpolation of escape sequences and variables.

```
1  #!/usr/bin/perl
2  # Fig. 9.3: fig09_03.pl
3  # Demonstration of here documents
4
5  use warnings;
6  use strict;
7
8  my $notInterpolated = "Interpolated!";
9
10 print <<DONE;
11 For here documents the type of quotes determines
12 what kind of interpolation is done.  No quotes means
13 double quoted interpolation.  $notInterpolated
14
15 DONE
16
```

Fig. 9.3 "Here" documents (part 1 of 2).

```
17  print <<'DONE';
18  Single quotes do not allow interpolation.
19  $notInterpolated.
20
21  DONE
22
23  my $variable = <<'DONE';
24  Here documents do not have to be used
25  with the print function.
26
27  DONE
28
29  print $variable;
30
31  ($variable = <<DONE) =~ s/^\s+//gm;
32      All the leading spaces will be
33         ignored so you can indent to
34            your heart's content.
35
36  DONE
37
38  print $variable;
```

```
For here documents the type of quotes determines
what kind of interpolation is done.  No quotes means
double quoted interpolation.  Interpolated!

Single quotes do not allow interpolation.
$notInterpolated.

Here documents do not have to be used
with the print function.

All the leading spaces will be
ignored so you can indent to
your heart's content.
```

Fig. 9.3 "Here" documents (part 2 of 2).

If the opening *identifier* is single quoted (line 17), the string inherits the properties of a single-quoted string. Lines 23–27 and 31–36 show how the here document does not necessarily have to be used with the **`print`** function. In this case, a here document is used to assign a multiline string to the scalar **`$variable`**.

Here documents are often used in CGI scripts as a convenient way to output large amounts of HTML without making multiple calls to **`print`**. This format can be used as an alternative to the **`CGI.pm`** functional shortcuts, thus:

```
print <<HTML;
<html><head><title>Here we are</title></head>
<body>
My web page here. The time is now $time.
</body></html>
HTML
```