Week 1

Readings

Browse Chapter 1 if you'd like
Chapter 2
Chapter 3

Key notes to keep in mind (AKA: how Perl differs)

- "There's More Than One Way To Do It"
- "No unnecessary limits"
- **#** begins a comment for the rest of the line
- always does floating point division (top of page 22)
- scalar variable always contains leading **$** (page 27 – "Scalar Variables")
- unless they change the meaning, parenthesis are not required, as in print function (page 29)
- curly brackets enclosing a block within a control structure are required (top of page 33)
- no boolean data type although you can still do boolean comparisons (page 34 – "Boolean Values")
    - 0 whether treated as a number or string, empty string, and **undef** are false
    - everything else is true
    - see page 36 – "The undef Value" for information on **undef**
- **undef** is treated as 0 or empty string depending on context, and can be used as either (Page 36)
    - no need to declare or assign a variable before it is needed.
    - note that it may throw warnings but will not cause a runtime error
    - perl handles garbage collection, so there is usually no need to assign **undef** to a variable (page 38)
- an array can hold strings, numbers, and **undef** all together in one list (page 39-40)
- arrays can grow and shrink dynamically, no need to pre-declare the size
- scalars, arrays, and hashes (discussed later) have separate namespaces (page 40)
- bookmark pages 42-43 on discussions of quoting with **qw** you will often see it in other code
- array contains leading **@** (page 44)
- keep in mind Perl's **$_** default variable as it is often implicitly used in others' code (page 48 – "Perl's Favorite Default: $_")
- "Perl always calculates the value being assigned (on the right) before it begins the actual assignment." (page 48)
    - This allows for easier swap statements without an intermediate variable (page 43)
    - For example: **($a, $b) = ($b, $a);**

Typos in the Reading

- Page 20 “Floating-Point Literals”
  - **-1.2E-23** should read **-1.2E-24**
  - It really makes no difference whether **e** is uppercase or lowercase
- Page 27 “Scalar Variables”
  - “Scalar variables in Perl are always referenced with the leading $&.” Should read “Scalar variables in Perl are always referenced with the leading $.”
  - Ignore the ampersand

Exercises

Book exercises from chapters 2 and 3.

Write a program to accept a list of names on separate lines until end-of-input. Then prompt the user to issue one command (listed below) and perform the appropriate action printed to the terminal. Thank the user and end the program.

Commands to implement: **sort**, **reverse**, and **concatenate**

Some caveats:

- **concatenate** should print the list of names without separation
- **sort** and **reverse** should print the list of names separated by a new-line
- you may look ahead to page 153 for **elsif** clauses or simply nest **if else** statements