LING/C SC/PSYC 438/538

Lecture 4
Sandiway Fong
Continuing with Perl

• Homework 3: first Perl homework
  – due Sunday by midnight
  – one PDF file, by email to me

• Learn Perl
  – Books...
  – Online resources
    • http://learn.perl.org/
    • we begin with ...
    • http://perldoc.perl.org/perlintro.html
Perl arrays and hashes

- **Scalars:**
  - strings, numbers (integers, floating point numbers), references

- **Arrays:**
  - Idea: list of scalars
  - Access by index: 0,1,2,...

- **Hash:**
  - Like an array except access not through a numeric index
  - Use user-specified keys

Different namespaces: $apple @apple %apple are different data structures and can co-exist simultaneously
Perl Arrays

• **Arrays:**
  – Idea: list of scalars
  – Access by index: 0,1,2,…

• **Notes on output**
  • print @a    zeroonetwothreefour
  • print “@a”  zero one two three four

  - negative indices count from the end -2..-1
  - $#array (index of last element)
  - @array (can have scalar interpretation)

controlled by system variable $”
**default value**: a space
Perl Arrays

Forgot to mention last time, a special array called @ARGV

• Getting user input into your program:

```perl
@ARGV
```

There are a couple of special arrays too, such as @ARGV (the command line arguments to your script) and @_ (the arguments passed to a subroutine). These are documented in perlvar.

- @ARGV

The array @ARGV contains the command-line arguments intended for the script. $#ARGV is generally the number of arguments minus one, because $ARGV[0] is the first argument, not the program's command name itself. See $0 for the command name.

```c
C programming language: int argc, char *argv[]
Shell script: $0  $1  $2  ..  ($#)
```
Perl Hashes

A hash represents a set of key/value pairs:

```
my %fruit_color = ("apple", "red", "banana", "yellow");
```

You can use whitespace and the `=>` operator to lay them out more nicely:

```
my %fruit_color = {
  apple => "red",
  banana => "yellow",
};
```

To get at hash elements:

```
$fruit_color{"apple"};  # gives "red"
```

You can get at lists of keys and values with `keys()` and `values()`:

```
my @fruits = keys %fruit_colors;
my @colors = values %fruit_colors;
```

Hashes have no particular internal order, though you can sort the keys and loop through them.

Just like special scalars and arrays, there are also special hashes. The most well known of these is `ENV` which contains environment variables. Read all about it (and other special variables) in `perldoc`. 
Perl Hashes

• Notes on arrays and hashes
  – **arrays** are indexed from 0,1,2,3...
  – **hashes** are like arrays with user-defined indexing
    *(aka associative array or hash table)*

– initialization
  (use list notation (or shortcut): *round brackets and commas*)
  • @a = ("zero", "one", "two", "three", "four");
  • %h = ("zero", 0, "one", 1, "two", 2, "three", 3, "four", 4);  *(key/value pairs)*

– access to individual elements
  *(square brackets vs. curly braces)*
  • $a[1]  "one"
  • $h{zero}  0

**Shortcut:**

```
1. qw(foo bar baz)
```

is semantically equivalent to the list:

```
1. "foo", "bar", "baz"
```
Perl Hashes

• Notes on arrays and hashes
  – output
    • `print @a` zeroonetwothreethreefour
    • `print "@a"` zero one two three four
    • `print %h` three3one1zero0two2four4
      (note: different order)
    • `print "%h"` %h (literal, no interpolation done)
More on Hash tables

• Example:

```perl
use strict;
my %fruitColor = ("apple", "red", "banana", "yellow", "kiwi", "green");
foreach my $i (keys %fruitColor) {
    print "$i => $fruitColor{$i}\n"
}
```

• Output:

```
dhcp-10-142-144-54:Desktop sandiway$ perl test.pl
banana => yellow
apple => red
kiwi => green
```

Unique key constraint:

```perl
my %fruitColor = ("apple", "red", "banana", "yellow", "kiwi", "green", "apple", "green");
```
More on Hash tables

• Last time:
  – Exercise: try to write the code for looking up keys based off a value
  – Anyone try it?

• An opportunity to use a loop
• Extras:
  last
  @ARGV
Perl arrays

• List operations:

1. `my @sorted = sort @animals;`
2. `my @backwards = reverse @numbers;`
Perl Arrays

- Last time:
  - by default sort works according to the ASCII chart, character by character

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>NUL (null)</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>NUL (null)</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>NUL (null)</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>NUL (null)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>SPACE (start of heading)</td>
<td>001</td>
<td>001</td>
<td>001</td>
<td>SPACE (start of heading)</td>
<td>001</td>
<td>001</td>
<td>001</td>
<td>SPACE (start of heading)</td>
<td>001</td>
<td>001</td>
<td>001</td>
<td>SPACE (start of heading)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>002</td>
<td>RET (raw text)</td>
<td>002</td>
<td>002</td>
<td>002</td>
<td>RET (raw text)</td>
<td>002</td>
<td>002</td>
<td>002</td>
<td>RET (raw text)</td>
<td>002</td>
<td>002</td>
<td>002</td>
<td>RET (raw text)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>003</td>
<td>EOT (end of text)</td>
<td>003</td>
<td>003</td>
<td>003</td>
<td>EOT (end of text)</td>
<td>003</td>
<td>003</td>
<td>003</td>
<td>EOT (end of text)</td>
<td>003</td>
<td>003</td>
<td>003</td>
<td>EOT (end of text)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>004</td>
<td>ENQ (acknowledgement)</td>
<td>004</td>
<td>004</td>
<td>004</td>
<td>ENQ (acknowledgement)</td>
<td>004</td>
<td>004</td>
<td>004</td>
<td>ENQ (acknowledgement)</td>
<td>004</td>
<td>004</td>
<td>004</td>
<td>ENQ (acknowledgement)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>005</td>
<td>ACK (acknowledgement)</td>
<td>005</td>
<td>005</td>
<td>005</td>
<td>ACK (acknowledgement)</td>
<td>005</td>
<td>005</td>
<td>005</td>
<td>ACK (acknowledgement)</td>
<td>005</td>
<td>005</td>
<td>005</td>
<td>ACK (acknowledgement)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>006</td>
<td>DLE (device control 1)</td>
<td>006</td>
<td>006</td>
<td>006</td>
<td>DLE (device control 1)</td>
<td>006</td>
<td>006</td>
<td>006</td>
<td>DLE (device control 1)</td>
<td>006</td>
<td>006</td>
<td>006</td>
<td>DLE (device control 1)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>007</td>
<td>DC1 (device control 1)</td>
<td>007</td>
<td>007</td>
<td>007</td>
<td>DC1 (device control 1)</td>
<td>007</td>
<td>007</td>
<td>007</td>
<td>DC1 (device control 1)</td>
<td>007</td>
<td>007</td>
<td>007</td>
<td>DC1 (device control 1)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>008</td>
<td>DC2 (device control 2)</td>
<td>008</td>
<td>008</td>
<td>008</td>
<td>DC2 (device control 2)</td>
<td>008</td>
<td>008</td>
<td>008</td>
<td>DC2 (device control 2)</td>
<td>008</td>
<td>008</td>
<td>008</td>
<td>DC2 (device control 2)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>009</td>
<td>DC3 (device control 3)</td>
<td>009</td>
<td>009</td>
<td>009</td>
<td>DC3 (device control 3)</td>
<td>009</td>
<td>009</td>
<td>009</td>
<td>DC3 (device control 3)</td>
<td>009</td>
<td>009</td>
<td>009</td>
<td>DC3 (device control 3)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>00A</td>
<td>DC4 (device control 4)</td>
<td>00A</td>
<td>00A</td>
<td>00A</td>
<td>DC4 (device control 4)</td>
<td>00A</td>
<td>00A</td>
<td>00A</td>
<td>DC4 (device control 4)</td>
<td>00A</td>
<td>00A</td>
<td>00A</td>
<td>DC4 (device control 4)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>00B</td>
<td>NAK (negative acknowledgement)</td>
<td>00B</td>
<td>00B</td>
<td>00B</td>
<td>NAK (negative acknowledgement)</td>
<td>00B</td>
<td>00B</td>
<td>00B</td>
<td>NAK (negative acknowledgement)</td>
<td>00B</td>
<td>00B</td>
<td>00B</td>
<td>NAK (negative acknowledgement)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>00C</td>
<td>SYN (synchronous idle)</td>
<td>00C</td>
<td>00C</td>
<td>00C</td>
<td>SYN (synchronous idle)</td>
<td>00C</td>
<td>00C</td>
<td>00C</td>
<td>SYN (synchronous idle)</td>
<td>00C</td>
<td>00C</td>
<td>00C</td>
<td>SYN (synchronous idle)</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>00D</td>
<td>ETB (end of transmission)</td>
<td>00D</td>
<td>00D</td>
<td>00D</td>
<td>ETB (end of transmission)</td>
<td>00D</td>
<td>00D</td>
<td>00D</td>
<td>ETB (end of transmission)</td>
<td>00D</td>
<td>00D</td>
<td>00D</td>
<td>ETB (end of transmission)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>00E</td>
<td>CAN (cancel)</td>
<td>00E</td>
<td>00E</td>
<td>00E</td>
<td>CAN (cancel)</td>
<td>00E</td>
<td>00E</td>
<td>00E</td>
<td>CAN (cancel)</td>
<td>00E</td>
<td>00E</td>
<td>00E</td>
<td>CAN (cancel)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>00F</td>
<td>SUB (substitute)</td>
<td>00F</td>
<td>00F</td>
<td>00F</td>
<td>SUB (substitute)</td>
<td>00F</td>
<td>00F</td>
<td>00F</td>
<td>SUB (substitute)</td>
<td>00F</td>
<td>00F</td>
<td>00F</td>
<td>SUB (substitute)</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>010</td>
<td>ESC (escape)</td>
<td>010</td>
<td>010</td>
<td>010</td>
<td>ESC (escape)</td>
<td>010</td>
<td>010</td>
<td>010</td>
<td>ESC (escape)</td>
<td>010</td>
<td>010</td>
<td>010</td>
<td>ESC (escape)</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>011</td>
<td>FS (file separator)</td>
<td>011</td>
<td>011</td>
<td>011</td>
<td>FS (file separator)</td>
<td>011</td>
<td>011</td>
<td>011</td>
<td>FS (file separator)</td>
<td>011</td>
<td>011</td>
<td>011</td>
<td>FS (file separator)</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>012</td>
<td>GS (group separator)</td>
<td>012</td>
<td>012</td>
<td>012</td>
<td>GS (group separator)</td>
<td>012</td>
<td>012</td>
<td>012</td>
<td>GS (group separator)</td>
<td>012</td>
<td>012</td>
<td>012</td>
<td>GS (group separator)</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>013</td>
<td>RS (record separator)</td>
<td>013</td>
<td>013</td>
<td>013</td>
<td>RS (record separator)</td>
<td>013</td>
<td>013</td>
<td>013</td>
<td>RS (record separator)</td>
<td>013</td>
<td>013</td>
<td>013</td>
<td>RS (record separator)</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>014</td>
<td>US (unit separator)</td>
<td>014</td>
<td>014</td>
<td>014</td>
<td>US (unit separator)</td>
<td>014</td>
<td>014</td>
<td>014</td>
<td>US (unit separator)</td>
<td>014</td>
<td>014</td>
<td>014</td>
<td>US (unit separator)</td>
<td></td>
</tr>
</tbody>
</table>

Source: [www.ascii-table.com](http://www.ascii-table.com)
Perl Arrays

Numeric sort?

- **Idea**: to pass an inline comparison function as parameter...
- **Note**: `fc` (fold case) from Perl 5.16 onwards only

```perl
use strict;
use v5.16;

# sort lexically
@articles = sort @files;

# same thing, but with explicit sort routine
@articles = sort { $a cmp $b } @files;

# now case-insensitively
@articles = sort { fc($a) cmp fc($b) } @files;

# same thing in reversed order
@articles = sort { $b cmp $a } @files;

# sort numerically ascending
@articles = sort { $a <=> $b } @files;

# sort numerically descending
@articles = sort { $b <=> $a } @files;
```

Binary "cmp" returns -1, 0, or 1 depending on whether the left argument is stringwise less than, equal to, or greater than the right argument.

Binary "<=>" returns -1, 0, or 1 depending on whether the left argument is numerically less than, equal to, or greater than the right argument.
Perl Hashes

• Sorting with a hash

```perl
# this sorts the %age hash by value instead of key # using an in-line function
@eldest = sort { $age{$b} <=> $age{$a} } keys %age;

# sort using explicit subroutine name
sub byage {
    $age{$a} <=> $age{$b}; # presuming numeric
}

sort byage keys %age
```

```perl
1 use strict;
2 my %fruitColor = ("apple", "red", "banana", "yellow", "kiwi", "green");

Let sort by fruit, by color, ascending, descending...
```
Implicit Coercion

Perl features implicit coercion of data types

• Example:
  – the following program prints \texttt{Equal!}
  – $\texttt{==}$ is the numeric equality comparison operator

\begin{verbatim}
my $one_string = "1";
my $one_number = 1;
if ($one_string == $one_number) {
    print "Equal!\n"
} else {
    print "Different!\n"
}
\end{verbatim}

• Example:
  – the following program prints \texttt{3 is my number}
  – \texttt{.} is the string concatenation operator

\begin{verbatim}
my @a = qw(one, two, three);
my $string = @a." is my number";
print "$string\n"
\end{verbatim}
Implicit Coercion

- print "4" * 4
  16
- (strange behavior if you’re coming from Python)
- print "4" x 4 ("x" is the repetition operator)
  4444
- @a = (4) x 4 (in list context)
  (4, 4, 4, 4)
Homework 3

Check your understanding...

Most frequent names (2000 US Census)

- Smith 2376207
- Johnson 1857160
- Williams 1534042
- Brown 1380145
- Jones 1362755
- Miller 1127803
- Davis 1072335
- Garcia 858289
- Rodriguez 804240
- Wilson 783051
- Martinez 775072
- Anderson 762394
- Taylor 720370
- Thomas 710696
- Hernandez 706372
- Moore 698671
- Martin 672711
- Jackson 666125
- Thompson 644368
- White 639515

1. Put them in a hash
2. Sort by name, sort by frequency
3. ascending and descending
4. Write code to count the number of names with frequency > some number, e.g. 1,000,000 – supplied on the command line
5. perl prnames.perl name|frequency ascending|descending
6. perl prnames.perl count number
   • i.e. computes item 4.
7. Print out the results (names...)