Programming in Perl

Week Seven

Using modules

Midterm review

Homework 6.1

```
my %hash = (a => 15, d => 42, c => 31, b => 25);
print join('|',values_by_key(%hash)),"\n";
sub values_by_key {
   my %hash = @_;
   my @values;
   foreach my $key (sort keys %hash){
       push @values, $hash{$key}
   }
   return @values;
}

# Same thing using map():

print join('|', map{$hash{$_}} sort keys %hash);
```

Homework 6.2

```
print join('|', grep{$_ >= 25} map{$hash{$_}} sort keys %hash);
```

What are Modules?

- Modules are about "code reuse"
  - The ability to take some existing code and easily re use it in your program
  - The idea is to publish an "interface" (i.e.: the way you use the code) so that you do not need to know the details of how the code works
- Perl comes with a large number of modules: the CORE Perl Modules
- Additional modules can be found on the Comprehensive Perl Archive Network (CPAN)
  - Perl comes with a module that allows you to access CPAN directly
  - For the ActiveState version of Perl, you can use the ppm program

Types of modules

- Modules are used to extend the functionality of Perl
  - You have been using modules, and you have not know it!
- There are two types of modules
  - Pure Perl modules
    - Modules written completely in Perl
    - Usable with out any special work
  - Extension modules
    - Modules written in another compiled language, usually in C, with a Perl wrapper
    - Must me compiled using the language compiler
- Perl modules use the file extension `.pm`

Installing Modules

- Once you have a module downloaded, you must install it into the Perl distribution
- Usually, this is in the `site_perl` directory
- You don't need to do anything special yourself, Perl knows how to do the install

```
perl Makefile.PL
make
make text
make install
```

- You can get a version of `make` for ActiveState from:

```
ftp://ftp.microsoft.com/Softlib/MSFILES/nmake.exe
```

Using Modules

- It's easy to use a module, once you have it installed;

```
use Module_name;
```

- Perl will search in all its standard places for the `.pm` file with the same name as the module, and load it
  - The `@INC` array contains all of the paths to the standard places
- You can create your own place to put modules, and tell Perl to look there, buy using the `file` module

```
use file 'path/to/other/libs';
```

File::Basename

- One of the things that you will find you will need to is to extract the file name from a path to the file
- The `File::Basename` module does this job for you

```
use File::Basename;

my $fullname = '/devel/sim/simpn2.c';
my $filename = basename($fullname);
my $dirname = dirname($fullname);

my ($filename, $dirname, $suffix) = fileparse($fullname,
  qw/ .c .pm /);
```

- Works file paths from most operating systems!

Command line options

- When we wanted to add command line options to programs, we need to scan the @ARGV array looking for strings that start with "-"
- Perl includes a standard module that does this:

```
use strict;
use Getopt::Std;
my %opts;
getopts('ab:c', \%opts);
foreach my $opt ( sort keys %opts) {
    print "$opt:$opts{$opt}\n";
}
print "remaining arguments: @ARGV\n";
```

Date::Manip

- There are different ways to represent dates
  - `Oct 10, 2000 October 10, 2000`
  - `10/10/00 10-10-2000`
  - Etc.
- You may want to do may different things with dates
  - Is one date later that another?
  - How many day between dates?
  - What was the date of the last Friday in January of 1998?
- You can use the `Date::Manip` module to do all of the is, and more
- Available on CPAN

Date::Manip

```
use Date::Manip
my $first= 'last Friday in Jan 1998';
my $second= 'Feb 20th 1998';
my $date1 = ParseDate($first);
my $date2 = ParseDate($second);

if ($date1 lt $date2) { … }
$delta = DateCalc($date1, $date2, $err, 1);
$delta = DeltaFormat($delta,2,"%dt");
print "$delta days from $first to $second\n";

$delta = DateCalc($date1, $date2, $err, 2);
$delta = DeltaFormat($delta,2,"%dt");
print "$delta business days from $first to $second\n";
```

LWP::Simple

- One of the main tasks that Perl is now used for is Web Programming
- There are several modules that aid you in this task
  - `lib-www` (LWP) - a package of several modules for working with the Web
  - `CGI.pm` - a module for writing CGI scripts
- `LWP::Simple` is a module in `lib-www`

```
use strict;
use LWP::Simple
print "Enter url to fetch: ";
chomp (my $url = <STDIN>);
getprint("http://$url"/);
```

Modules on CPAN

- Explore CPAN
  - `http://www.cpan.org`
  - `http://search.cpan.org`

Midterm Review

- Covers all reading in Learning Perl
- It is "Open Everything"
  - Questions will come out of Text
  - You can use other books or info from the internet
- You will be writing one or more simple programs
- Multiple choice questions
- Worth 100 points (i.e.: two letter grades)

What the Midterm Covers

- Writing code and programs
  - Why should you comment?
  - How should you name variables?
  - What for the steps for developing a program?
- Data types and variables
  - Scalars and scalar variables
  - Arrays and array variables
  - Hashes and Hash variables
  - Expressions and common Perl operations (`+`, `-`, `*`, `/`, `le`, `==`, etc.)
  - Contexts

What the Midterm Covers

- Control Structures
  - Bare blocks
  - `if () {} else {}`
  - `while` loops
  - `for` loops
  - `foreach` loops
  - Statement modifiers
- I/O
  - File and Directory handles
    - Opening and Closing
    - Commands that use them
  - Input operator
  - Print commands

What the Midterm Covers

- Functions
  - Scope
  - Global Variables
  - `my` Variables
  - User subroutines
    - Arguments and return values
- References and aggregate data structures
  - Creating and using references
  - Creating aggregate data structures
    - List of Lists
    - Complex structures

What the Midterm Covers

- Documentation
  - POD syntax
- Regular Expressions
  - Simple patterns
  - Character classes
  - Iterators
  - Anchors
  - Grouping
    - With and with out memory
  - Modifiers

What the Midterm covers

- Working with Text
  - The Match and Substitution operators
  - Working with sub strings
- Working with LISTs
  - Processing LISTs
  - Filtering LISTs
  - Sorting LISTs
- Modules
  - What are modules?
  - Where can you find modules?
  - How do you use modules?