CIS166 - Programming in Perl

Week One

Introduction

Writing Code

Writing Programs

First Things First

- We Lied (Sort-a)
  - This is CIS166 - Programming in Perl
  - Not CIS166 - Web Scripting/Programming
  - We will focus on the language, not the application.

Introduction

- We are here to learn how to program by learning to program in the Perl scripting language
- Instructor - Mark Pease
  - Software Engineer at Freescale Semiconductor
  - 30+ years of experience
  - 15+ years of Perl programming and teaching

Todays Topics of Discussion

- Cover the course Syllabus
- Introduction
  - What is Perl
- Writing Code
  - What is “Code”
- Writing Programs
  - A first program

Course Syllabus

- ITI Standard Syllabus

Course Number:	CIS166	Course Name:	Programming in Perl
Section Number:	5687	Prerequisite:	CIS105, CIS150
Semester:	Fall 2004	Class Time:	Saturdays 8:30 am - 11:10 am
		Text Book:	Learning Perl 3rd Edition
Instructor's Office:	None	Instructor:	Mark Pease
Instructor's E-mail:	Mark.Pease@cgcmail.maricopa.edu		

- When sending email always start the Subject: line with
  - [CIS166]
- It might not get read very quickly if you don't

Course Syllabus

- Instructor's Lab Hours:
  - Right after class
- Instructor's Office Hours:
  - By appointment
  - Please use E-mail!
    - Remember to prefix the Subject: with [CIS166]

Course Syllabus

- Grading

Attendance/Participation	50
Midterm Exam	100
Final Exam	100
2 Other Exams (50 each)	100
Exercises	100
Project Planning	50
Project	100

  - Lowest scoring exam(s) will be excluded (100 points)
  - 500 total points possible

Course Syllabus

- Grading Scale

Scale	Letter Grade	Points Needed
86-100%	A	430
71-85%	B	355
56-70%	C	280
41-55%	D	205
<40%	F	0

Exams

- Each Exam will be 25 questions covering all course material covered up to the point of the exam
- The Midterm Exam will be on 10/16
- The Final will be 12/11
- There will be two “Other” exams sometime
- You will write a program during the Midterm and Final

Exercises

- Each exercise is worth 10 points
- You get 5 points for turning in the assignment, as long as it made a substantial attempt at completing each exercise
- If you don't understand, or you can't find a "bug", E-mail the instructor for help (early!)
  - Remember to prefix the Subject: with [CIS166]

Project

- Details about the project will be given on 9/18
- Be thinking about what you would like to do
  - Simple task, not a lot of features
  - Examples:
    - CD Library Card Index that you can enter, search, and print
    - Address list
    - Account management

Project

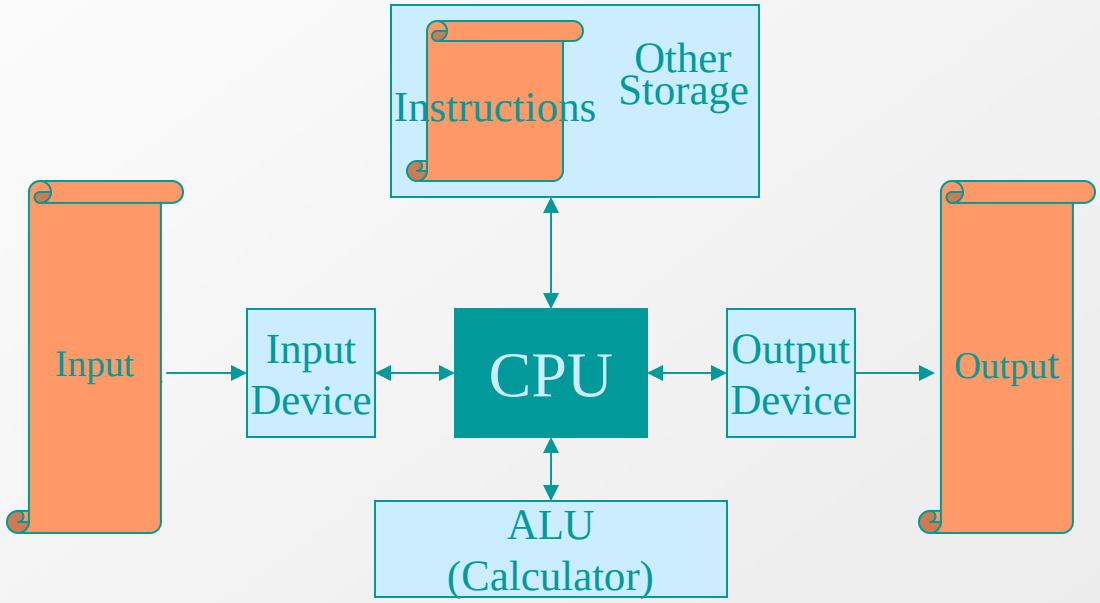
- For the Project, you will:
  - Write a simple Proposal
  - Have Code reviews with the instructor
  - Give a short presentation when the project is completed (12/4)
- Proposal/Reviews are worth 50 total points
- Completed Project/Presentation are worth total 100 points

Introduction

- On Programming
  - You have been programming for a long time
  - When you write a story, tell a joke, give someone directions to your house, your programming
  - Programming is about solving problems

On Programming

- Computers are mindless devices capable only of doing what they are told



On Programming

- Because Computers a mindless devices, programming (solving the problem) is developing step-by-step instructions to solve the problem
- The step-by-step instructions are called an algorithm and they can be specified in any language

On Programming

- Computers are really programmed in “machine language”
  - Just a bunch of zero and ones (binary code)
  - We have trouble remembering what all the binary codes mean, so we developed ways to attach symbols to the code to make it easier to program

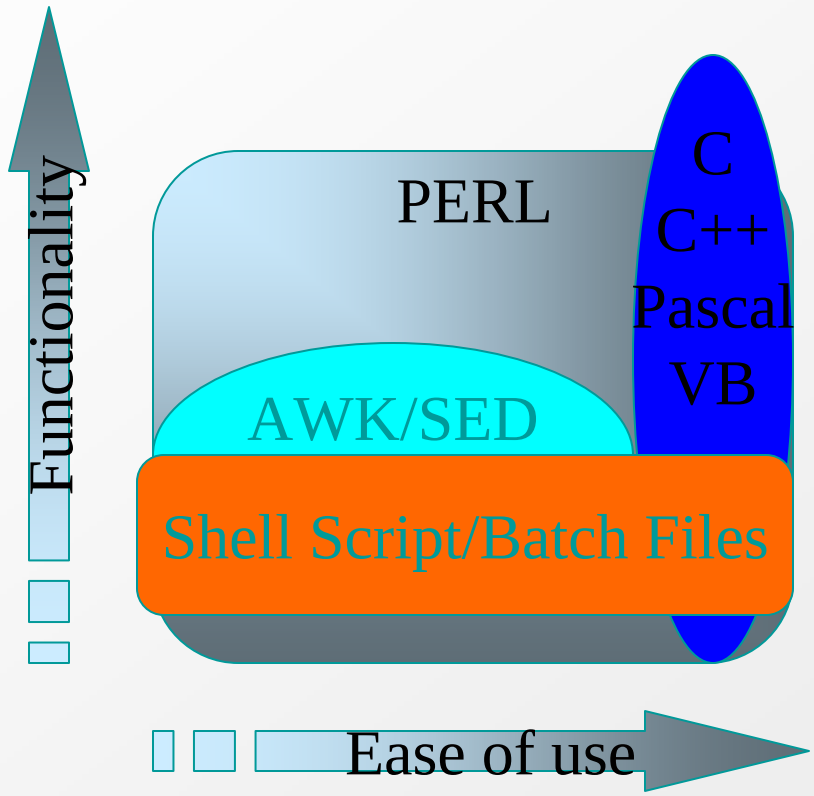
On Programming

Machine Code	Assembler Code	High Level Code
8B45FC	movl a, %eax	
0345F8	addl b, %eax	c = a + b
8945F4	movl %eax, c	

On Perl

- History of Perl
  - Larry Wall created Perl in late 1987 to solve a text processing problem that he could not solve using standard UNIX tools
  - Instead of “filling the gaps” in the standard tools, he created a general tool that makes it easer to solve text processing problems
  - http://history.perl.org/PerlTimeline.html

On Perl


Functionality
PERL
C
C++
Pascal
VB
AWK/SED
Shell Script/Batch Files
Ease of use

On Perl

- Some Applications where Perl is used:
  - Data Manipulation
    - data converters, report generators
  - Automation
    - system administration, data gathering, etc.
  - Portable Applications
    - UNIX, MSDOS, MacOS, others
  - Rapid Prototypes
  - Web Automation
    - CGI, Database Interface

On Perl

- Perl Culture
  - Perl is free
    - Perl is one of the first large “Open Source” development efforts
    - Perl is a copyrighted work under the GNU “Copyleft” or Perl’s Artistic License
    - The source, and some binaries, are available on CPAN

On Perl

- Perl Culture
  - Perl is developed by a large group of volunteers
    - Mailing lists: Perl5-Porters, Perl6 lists
    - New releases are controlled by Larry Wall and the “pumpkin” holders
    - New releases are split into two types Maintenance and Developmen
      - Maintenance are even sub numbers: `5.8.4`
      - Development are odd sub numbers: `5.9.2`

On Perl

- Perl Culture
  - Perl is supported by volunteers
    - “Bugs” are reported using the perlbug utility, and tracked using the Perl “Bug-a-tron” `http://bugs.perl.org`
    - Usage questions are handled on newsgroups and mailing lists.
      - comp.lang.perl.misc for General Perl questions
      - comp.lang.perl.announce for General Perl Announcements

On Perl

- Perl Culture
  - Perl Self-help
    - The Perl Reference Manual and the Perl FAQ, are available on `http://www.perldoc.com`
    - Programming Perl, Learning Perl, The Perl Cookbook, etc.
    - Lots and lots of other books on perl!
  - The Perl Journal magazine (TPJ)
  - `www.perl.com` - News and articles from O'Reilly
  - `www.perl.org` - The Perl Mongers
  - `use.perl.org` - Perl news and discussion

On Perl

- The Perl Philosophy:
  - “There’s more than one way to do it!” (TMTOWTDI, pronounced “tim toady”)
  - Perl provides several different way to do things
    - It's up to you to pick the "best" way
    - There is no "correct" way to do anything

On Perl

- What does Perl look like?
  - A Perl program is a ASCII text file called a “script”
  - Perl is an interpreted language— no need for the edit-compile-test development loop like you use with “C” or “C++”
  - When perl is invoked on a script, it is compiled to an internal form, then executed

On Perl

- What Does Perl Look Like?
  - Perl's syntax is somewhat "C" like, "AWK" like, "Lisp" like (and a little Basic, too)
  - Whitespace is, for the most part, insignificant
  - Comments are a "#" to the end of the line— there are no multi-line comments

On Perl

- What Does Perl Look Like?
  - Perl is dynamically typed language
    - A simple “scalar” variable can take on any type of value: integer, floating point, string
    - Variables pop into existence as needed
- A line of Perl, a “Statement”, always end with a semicolon (“;”)

On Perl

- Some Perl Examples
  - “Classic” Hello, World!

```
#!/usr/local/bin/perl
print "Hello, World!\n";
```

  - A more complex example

```
#!/usr/local/bin/perl
@a=split(/X*/," ,Jpacehklnorstu");
print@a[2,15,13,14,0,4,10,11,14,7,6,12,0,3,6,12,9,0,7,4,5,8,6,12,1];
```

Running Perl

- Perl is located in `d:\perl\bin`
- Using textpad, create a file called `hello.pl`

```
print "Hello, World!\n";
```

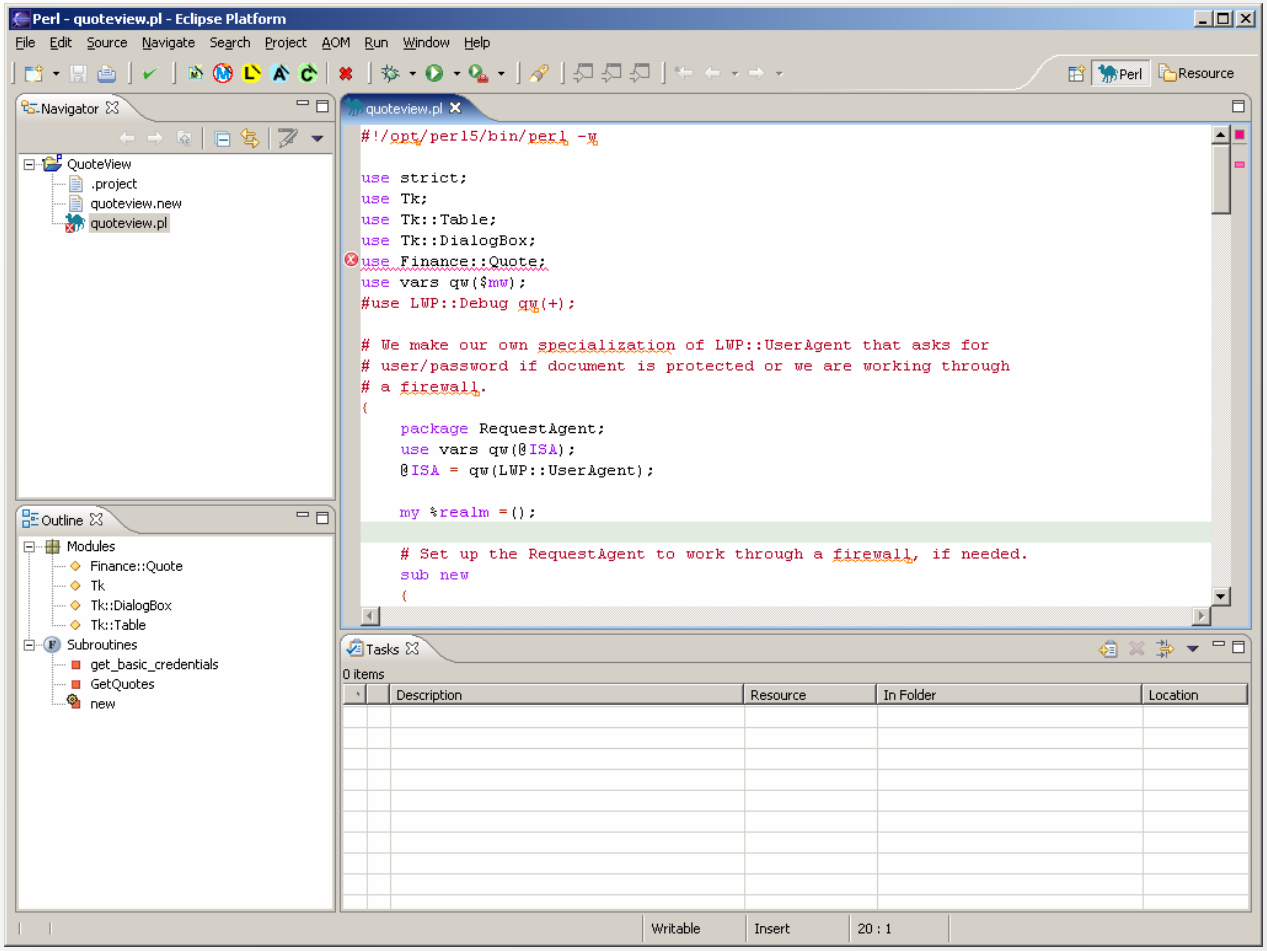
- In a MSDOS Command Window type:

```
perl hello.pl
```

Getting Perl for Yourself

- Pre-compiled versions
  - Windows: http://www.activestate.com
    - An IDE- Komodo - is available from http://aspn.activestate.com
  - Mac: Included with Mac OS X
  - Source for "rolling you own": http://www.cpan.org/src/stable.tar.gz

Eclipse with e-p-i-c



Eclipse with e-p-i-c

- Eclipse is an open source "generic" IDE written in Java
  - `http://www.eclipse.org`
- Epic is an Eclipse "plugin" for working with Perl
  - `http://e-p-i-c.sf.net`

Chapter 2- Writing Code

- Writing Code is not programming- just part of the process.
  - It's the act of translating your thoughts about what you program will do, into its implementation in a programming language.
  - How well you write your code affects how easy it is to read, which in turn affect how easy it is to fix, upgrade, and add features.

Writing Code

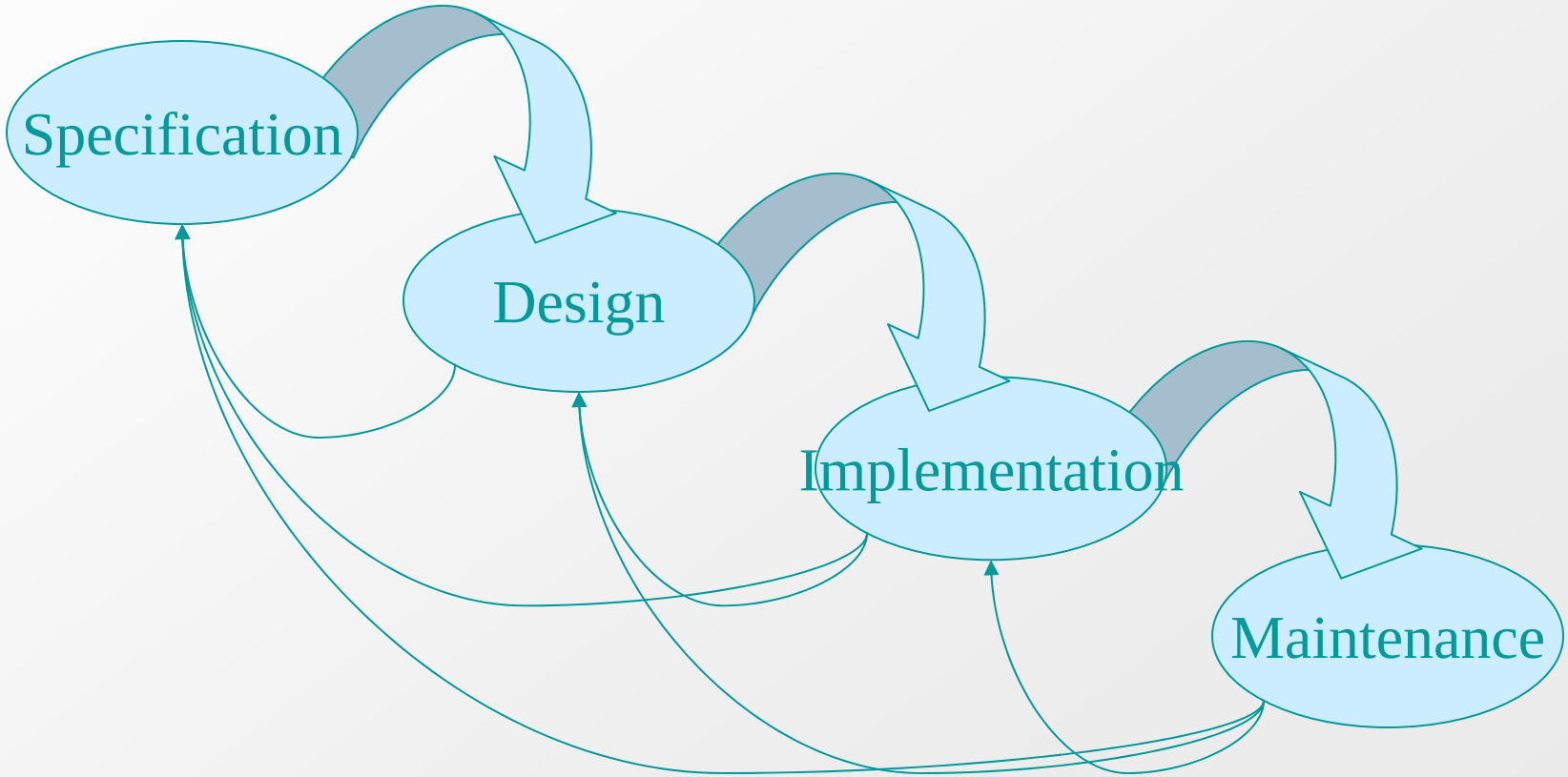
- Perl is sometimes call a “write only” language
  - It’s use of “funky symbols”
  - Short cuts
- But it’s not true!
  - Mostly it just bad coding style, and you can write bad in any language

Writing Programs

- Programming is more than writing code
- It’s a process with specific phases
  - Specification - The program is defined
  - Design - The algorithms are chosen
  - Implementation - We code, test, debug the program
  - Maintenance - The program is maintained

Writing Programs

- All of the phases overlap and loop back on each other
  - The development cycle



Writing Programs

- Specification
  - Must be well-defined
  - Everyone must agree that the specification is what will be implemented
  - An example is a good way to show what will be implemented

Writing Programs

- Design
  - Break down the tasks into a list
  - Decide on how you will approach the tasks
    - Top-down design- start at the top and work your way down
    - Bottom-up design- start at the bottom and work your way up
    - Top-down/Bottom-up- start at the top and work your way down, but implement the obvious drivers and interfaces
  - Keep doing that until you have something that you can implement
  - Write task in a “pseudo-code” that is structured English
    - Start writing out, in progressively more detail, what you want a section of a program to do
    - Be consistent on how you write out the routines
    - Perl can be thought of as "Structured English"
  - Also plan on how you will test the program

Writing Programs

- Implementation
  - Implement what was planned in the design phase ("Coding")
  - Once you know what you want to implement, and how it will work, it should be simple to translate into your target language
  - After you have you have implemented, in code, a routine, you should test what you have written to see if it does what you planned
    - You should start testing early
      - For example, if you just code a driver that reads and writes data to/from a file, write a simple program that exercises the routines to make sure they work
    - Continuing testing as you develop your code, until the complete program is written and tested

Writing Programs

- Maintenance
  - Bug fixes - You may have missed something in testing or implemented something wrong
    - You need to plan on how you will handle bug reports
  - Enhancements - Once the program is in use, new features may need to be implemented
    - You need to evaluate the change, or bug, to see if it is cost effective to implement
  - End Of Life - You must know when you should just give up on the program!

Home Work - Week One

- Quickly read Chapters 1, 2, 3, 5, and 10 of Learning Perl
- Do Exercise One
- E-mail the results to mark.pease@cgcmail.maricopa.edu
  - Attach the answer as a file named <lastname>_3.2.txt
  - Use the subject line: [CIS166] Week One Homework: <lastname>

Exercise One

- 1. Install Perl on your system
- 2. Type perl –v in a command window, and copy the results into an email to me

Blackboard

- The slides will be available on blackboard.cgc.maricopa.edu
  - Use your school login
    - Login ID: <4digitsOfLast><FirstofFirst><first3OfID>
      - peasm543
    - Password: <Last4OfID>
      - 1234