Perl Coding Standards

UVic SEng 265

Daniel M. German
Department of Computer Science
University of Victoria

September 30, 2002 Version: 1.00

dmgerman@uvic.ca

Coding Standards

- Formatting:
  - 4 spaces indentation
  - Control statements (if, for, foreach, etc) according to notes.
  - **No line larger than 80 characters**

dmgerman@uvic.ca

Coding Standards

- Names: the name of a variable should tell the reader what its purpose is
  - Function names: should be capitalized, with _ separating words in the idenfifier, for example: `My_Function, Get_Line, Read_File`
  - Contants: should be all uppercase, separated by _: `$MAX_NUMBER, $DEFAULT_INPUT_FILE`
  - Variables: use camel style: first word in the name (in a multi word variable name) should be lowercase, but the other ones should be capitalized, and they should not use _. For example: `$localCounter, $inputFile, $bufferZone, $nextChar, $identifier`.

Function Comments

- Split your code in functional units (a crucial Soft. Eng. principle)
- Every function should have a comment preceding it
  - Name
  - Purpose

dmgerman@uvic.ca

Function Header Example

```
#
# Name:      Get_Line
#
# Purpose:   To read a line of at most maxSize characters from
#            standard input
sub Get_Line
{
   # code goes here
}
```


dmgerman@uvic.ca

File Header

- Add comment at the very top of **every** file:
  - Programmer's Name
  - Copyright notice
  - Purpose of the program
  - Date of creation
  - Log changes you make to it, if possible (one line at most)

File Header Comment

```
#
# Programmer: Daniel German
#
# Copyright 2002 Daniel German
#
# Purpose:    The purpose of this file is blah, blah, blah
#
# Log:
# 010920 Created this file
# 010922 Added Get_Line
# 010923 Fixed pesky bug that did not let me sleep
# 010924 Completed error handling
# 010925 Finished!
#
```


dmgerman@uvic.ca

Finally

- Comment your code!
- Code should be easy to read and understand
- Everything else: use common sense
- **Impress me!**

dmgerman@uvic.ca