Design and Implementation*

• Objective: To design and implement a program for a relatively small yet reasonably complicated problem. To introduce and review a variety of implementation languages and to have students review the pros and cons of different implementation choices and languages.

• “Show me your flowcharts and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won’t usually need your flowcharts; they’ll be obvious”

  – Frederick P. Brooks, The Mythical Man Month

*The examples in these slides come from Brian W. Kernighan and Rob Pike, “The Practice of Programming”, Addison-Wesley, 1999.
Themes

• “Once the data structures are laid out, the algorithms tend to fall into place, and the coding is comparatively easy”

• The choice of programming language is relatively unimportant to the overall design.

• Comparing implementations demonstrates how languages can help or hinder, and ways in which they are unimportant.
Topics

• Problem: Generate random English text that reads well.
• Program: some data comes in, some data goes out, and the processing depends on a little ingenuity.
• Implementations: C, C++, Java, Perl
Failed Attempts

- Generate random letters (even with proper frequency).
- Generate random words chosen from a dictionary
- Need a statistical model with more structure (frequency of whole phrases)
Markov Algorithm

• set $w_1$ and $w_2$ to the first two words in the text
• print $w_1$ and $w_2$
• loop:
  – randomly choose $w_3$, one of the successors of $w_1$ and $w_2$ (in sample text)
  – print $w_3$
  – replace $w_1$ and $w_2$ by $w_2$ and $w_3$
Sample Markov Algorithm

• Input Prefix:
  – show your
  – your flowcharts
  – flowcharts and
  – flowcharts will
  – your tables
  – will be
  – be mystified
  – be obvious

• Suffix words that follow
  – flowcharts tables
  – and will
  – conceal
  – be
  – and and
  – mystified. obvious.
  – Show
  – (end)
Hash Table to Find Suffix Following Two Word Prefix

A state:

```
| pref[0] | pref[1] | suf | next |
```

"show"

"your"

A suffix:

`word
next`

Another suffix:

`word
next`

Another state:

```
| pref[0] | pref[1] | suf | next |
```

"tables"

"flowcharts"
Implementation

• See lecture outline for links to 4 different implementations
  – C (see Makefile)
  – C++
  – Java
  – Perl

• What are the pros and cons of the different implementations?