Pseudocode, programming languages and tools

PSU CS 300 Lecture 5-1

Bart Massey
Assoc Prof Computer Science
Portland State University
<bart@cs.pdx.edu>

Pseudocode

- **Input: Last stage of detailed design**
- **Output: Program described as English text**
- **Rationale: Programming is hard; language is irrelevant**

Principles of pseudocode

- **Roughly one pseudocode statement for every 1-10 lines of code**
- **Easily translatable to code**
- **Use abstract data types, esp sets, graphs**
- **Pick your degree of formalism**

Example: Zipf's Law

- **Zipf's Law: Graph of $\langle R(w_i)^{-1}, C(w_i) \rangle$ roughly linear increasing**
- **Task: Emit, in ascending *x* order, ‹*x*, *y*› pairs for all words in input text**
- **Ties?**

Design

- **Read input text, generating *frequency table***
- **Sort frequency table by decreasing rank**
- **Emit ‹*x*, *y*› pairs from table**
- **(External plotting program *gnuplot*)**

Pseudocode (1)

- Create new empty hash map *freq* from word to count
- Read characters from input
- Group characters into words

Pseudocode (2)

- For each word *w* in input
  - Normalize *w*
  - If *w* in *freq*, increment count
  - Otherwise, set count to 1
- Convert *freq* to list of ‹w_i, $C(w_i)$› pairs

Pseudocode (3)

- Sort *freq*' by decreasing $C(w_i)$
- For each $\langle w_i, C(w_i) \rangle$ in *freq*'
  - Print line w_i, 1.0 / i, $C(w_i)$

Implementation

- Pick a language
- Write code
- No problem!
- Problems
  - Coding issues
  - Bugs

Pick a language, any language

- Four basic idioms
  - Imperative (von Neumann)
  - Functional
  - Logic
  - Object Oriented

The C problem

- Horrible application lang
  - Inexpressive
  - Error-prone
    - with bad failure modes
  - Not modular
  - Not portable
- So bad, it encourages C++

You want these features

- “Higher-level” constructs
- Decent module system
- Automatic storage mgmt
- Maximal static checking
- Domain support
- Decent dev environment