

# KWIC Exercise

- D. Parnas. "On the Criteria to be Used in Decomposing Systems into Modules." *Communications of the ACM*, 15(12):1053-1058, December 1972.

# Key Word in Context (KWIC)

The KWIC index system accepts as input an ordered set of lines, each line is an ordered set of words, and each word is an ordered set of characters. Any line may be “circularly shifted” by repeatedly removing the first word and appending it at the end of the line. The KWIC index system outputs a listing of all circular shifts of all lines in alphabetical order of the keyword used to shift the line

- Common (*stop*) words may be omitted

# Circular Shifts

- **Original title**
  - Gone with the Wind
- **Circular shifts (key words underlined)**
  - Gone with the Wind
  - with the Wind Gone
  - the Wind Gone with
  - Wind Gone with the
- **Stop word removal**
  - Gone with the Wind
  - Wind Gone with the

# Example With Multiple Titles

- Gone with the Wind
- War and Remembrances
- The Winds of War

# KWIC Example

	<u>Gone</u>	with the Wind	
and	<u>Remembrances</u>		War
Winds of	<u>War</u>		The
	<u>War</u>	and Remembrances	
with the	<u>Wind</u>		Gone
The	<u>Winds</u>	of War	

# EXERCISE

(Parnas)

- Assume that you had to implement KWIC. Decide how you would break it into approximately six pieces (even if you don't think it is big enough to have pieces)
  - Use a box for each piece and give it a label
  - Ignore stop-word removal
- Decide how the pieces communicate
  - Use a line between two boxes to indicate some form of communication
  - Label the line to indicate the type of communication
- **Come up with at least two solutions**
  - For each, give circumstances when that solution would be advantageous

# 1. Shared Data Decomposition

- Break into components based on the functions computed
  - Also Known As: *Functional Decomposition*
- All components share access to data in memory
- Usually contain a *master-controller* routine
  - Dependencies realized with function calls

# Components

- **Input:** reads titles, stores in memory
- **Circular shifter:** constructs array of pairs <line index; word index>
- **Alphabetizer:** batch sorting of circular shifts
- **Output**
- **Master controller**

# Shared Data

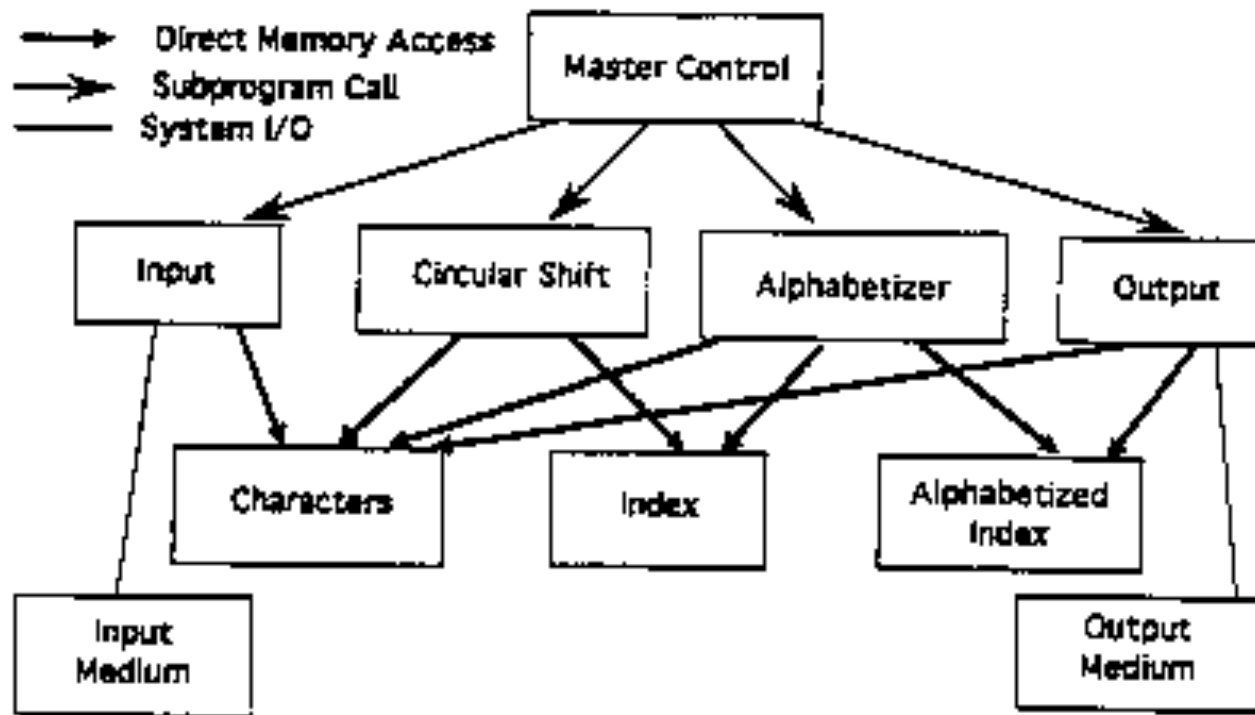


Figure 6: KWIC - Shared Data Solution

## 2. Abstract Data Types (ADT) Decomposition

- Organization based on data structures
- Representation hiding
- Modules holding data structures also provide services
- Precursor to object orientation

# Components

- **Line:** ADT
- **Input:** reads titles, hands to Line
- **Characters:** ADT
- **Circular shifter:** ADT
- **Alphabetic Shifts:** ADT
  - l-th circular shift
- **Output**
- **Master controller**

# Abstract Data Type

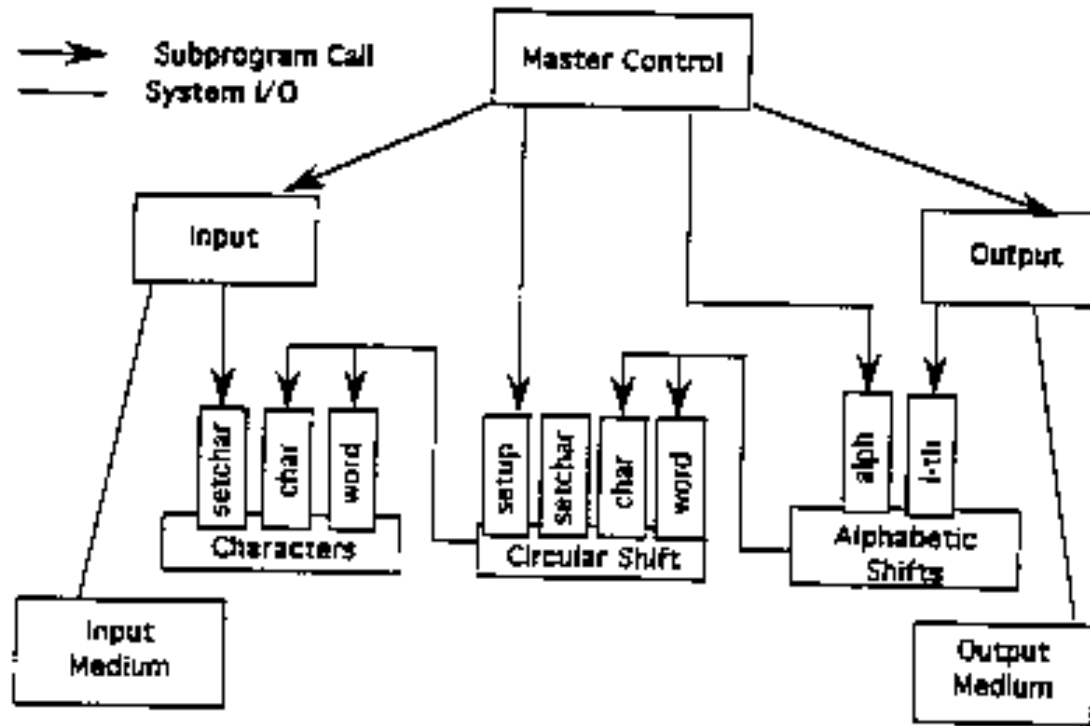


Figure 7: KWIC - Abstract Data Type Solution

# 3. Implicit Invocation

- Client modules express interest in state changes in server modules (*registration*)
- Server announces changes to all registered listeners (*broadcast*)
- Server does not know identity of clients
- Unit of notification is the *event*

# Components

- Input of new line
- Update line data structure
- Invoke circular shifter
- Update (second) line data structure
- Invoke alphabetizer

# Implicit Invocation

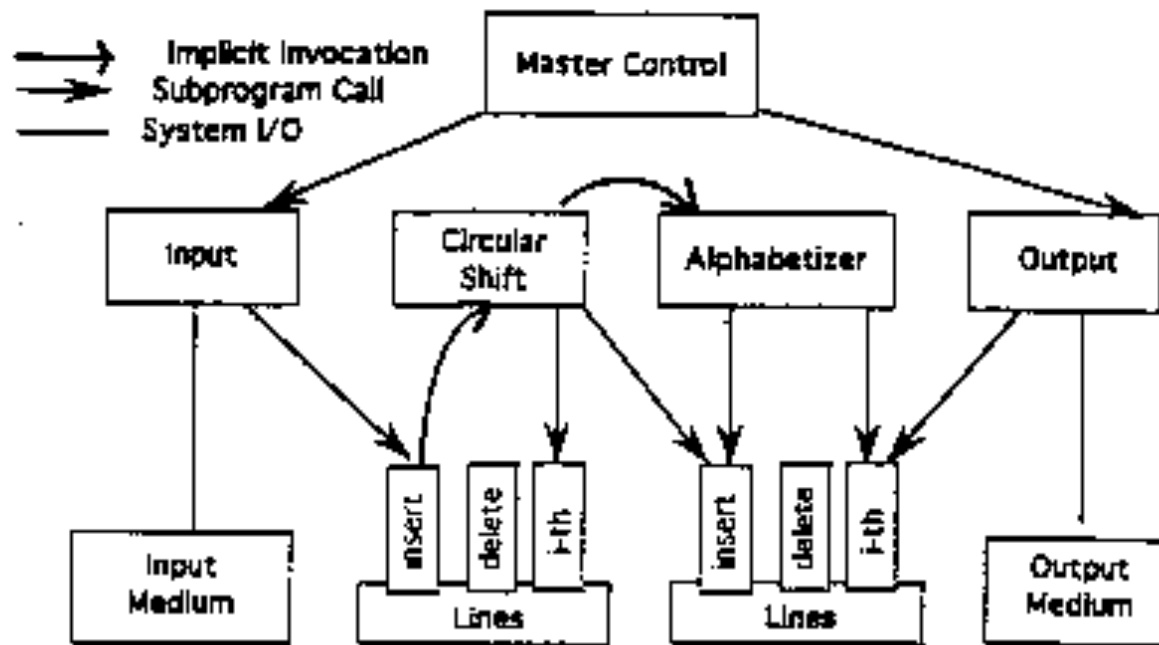


Figure 8: KWIC - Implicit Invocation Solution

# 4. Pipe And Filter

- System organized into independent programs (*filters*)
- Each filter takes in input (`stdin`) and produces output (`stdout`)
- Filters are connected together via FIFO queues (*pipes*)
- Common assumption of sequential files containing lines of ASCII characters
  - Can you think of a non-ASCII example?

# Components

- Filters for circular shift and alphabetizer
- Pipe connections with files of FIFOs
- No common data storage

# Pipe And Filter

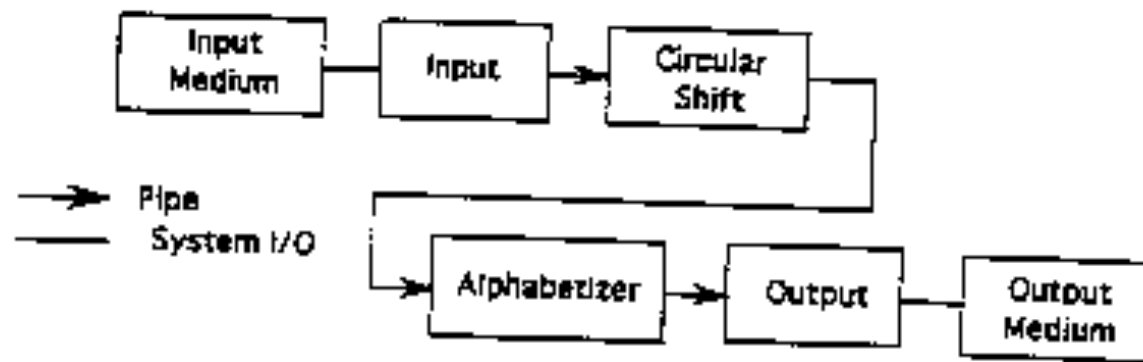


Figure 9: KWIC - Pipe and Filter Solution

# Evaluation

- Shared data
  - + Intuitive, efficient access
  - Brittle to changes in representation
- ADT (with no data sharing)
  - + Maintainability, reuse
  - Performance
- Implicit invocation (active data)
  - + Enhancements, data representation changes, reuse
  - Difficult to control/think about, inefficient
- Pipe and filter
  - + Intuitive, reuse
  - Interactivity, space efficiency

# Exercise Continued

- List at least three possible enhancements / improvements that could be made to KWIC
- For each of the four styles (shared data, ADT, implicit invocation, pipe and filter) indicate how well it would cope with each of your changes
  - Use the categories (*brittle*, *moderate*, *flexible*) to label your choices

# KWIC Changes

- Input format
- Reuse
- Processing algorithm
  - Shift each line as it is read
  - Shift lines after all are read
  - Shift lines on demand
  - Incremental versus batch sort

- New functionality
  - Stop-word elimination
  - Interactive line deletion
  - Use of external storage
  - Performance optimization
- Data representation
  - Line storage
  - Explicit circular shifts
  - Index/offset circular shift

# Lessons

- Hide design decisions, particularly where representation decisions are concerned:  
*information hiding*
- Organize components around data