Assignment 1: KWIC

Code Repository URL:	https://github.com/TIC3001/tic3001-ay2122s2-15		
Team number:	15		
Name:	Wang Wei	Name:	Chan Khai Wen
Student Number:	A0211395J	Student Number:	A0211493L

1. Introduction

p to 1 paragraph about the assignment and the approach that you took in your requirements specification, design and implementation. Specify who in the pair has done which implementation.Up to 1 paragraph about the assignment and the approach that you took in your requirements specification, design and implementation. Specify who in the pair has done which implementation.

Key-Word-In-Context (KWIC), after reading the case study we learnt there are 4 types of solution to implement it (Main Program/Subroutine with Shared Data, Abstract Data Types, Implicit Invocation and Pipes and Filters). After reading the explanation and diagram we have decided to implement solution 2 (Abstract Data Types, done by Chan Khai Wen) and solution 4 (Pipes and Filters, done by Wang Wei). Due to time constraints we have chosen the solutions that we felt most comfortable and easier to implement compared to other 2.

2. Requirements Analysis

Specify functional and non-functional requirements against which you design and implement. You can use bullet points or tabular format. You can record requirements as user stories or feature lists or combination of both. Refer to lecture material to get information on requirements specification.

Functional Requirement index	Description
1	Allow user to use text file to store data
2	Able to read the text file
3	Able to perform KWIC actions
4	Able to output a text file with results
5	The output must be alphabetical order, 1 - 9, a - z and ignore case sensitive (the sample output is ordered this way)

Non-Functional Requirement index	Description
1	Allow user to enter which file want to be processed

2	Allow user to enter output file name
3	Able to handle exceptions like file not exist in the location
4	Performance of the software (Duration needed to generate output file)

3. Architectural Design

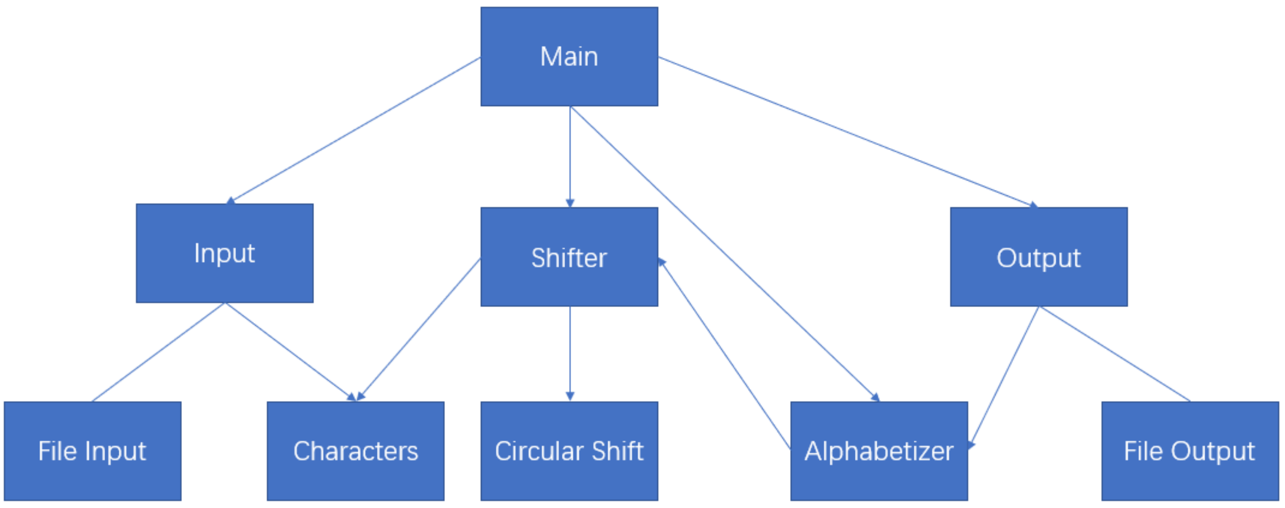

Figure 1: KWIC - Abstract Data Types solution

Abstract Data Types (ADT) solution is based on data abstraction and object-oriented organization and data representations. The associated operations are highly encapsulated in abstract data type/object. In this design, each object provides interfaces for other objects to access/communicate data only (information hiding). In the shared data solution, any changes on algorithms/data representations will impact the system heavily. Meanwhile ADT allows you to change individual modules without much inconvenience.

As per above figure, Main will pass control to Input, Input will be reading the data lines from text file (or any input medium) and store the lines by invoking “SetChar” of “Characters”. In this stage, the operation will create/access/delete characters, words and lines. The “Char” and “Word” will be used by the Shifter object to shift the words of each line. Next, the Shifter object will be repeating the process by removing the first word and adding it back at the end of line until a circular is archived. After lines are reconstructed, the Alphabetizer object will sort the lines in alphabetical order(case insensitive order) and return to output to create a text file.

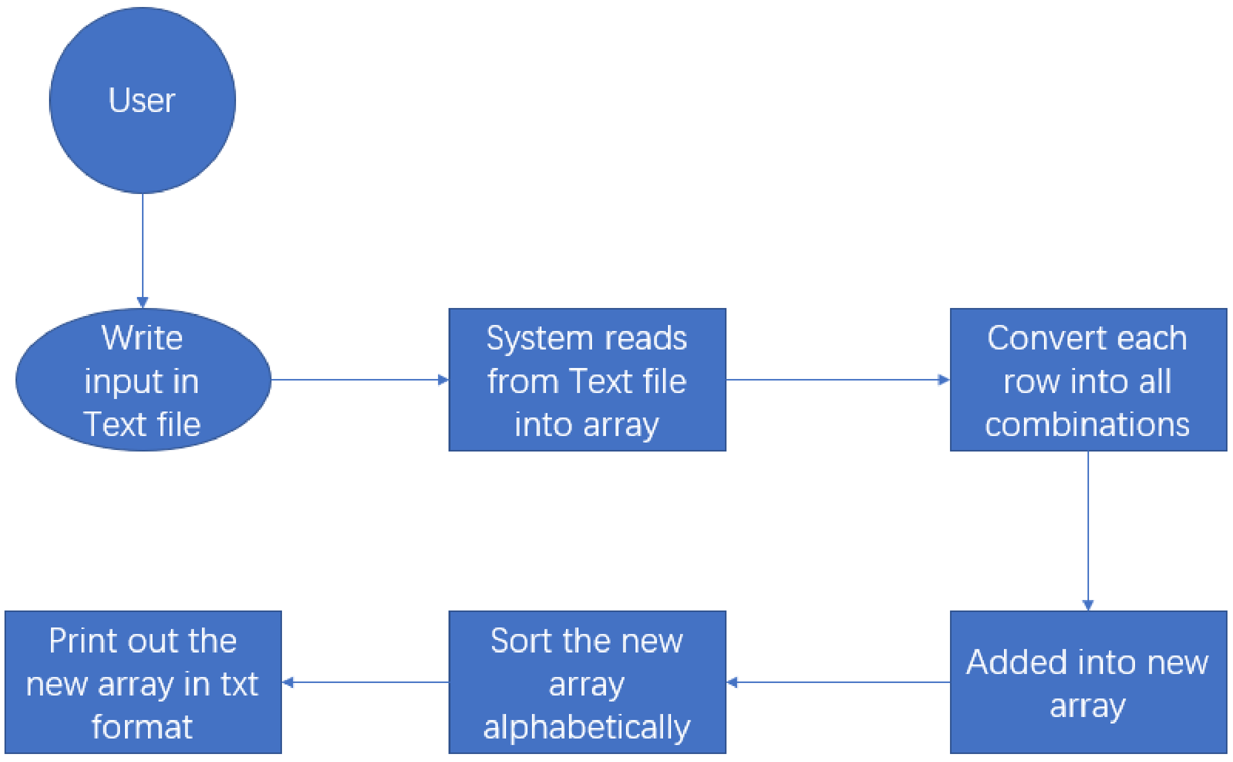
User
Write input in Text file
System reads from Text file into array
Convert each row into all combinations
Added into new array
Sort the new array alphabetically
Print out the new array in txt format

Figure 2 : KWIC - Pipes and Filters solution

Pipes and Filters solution uses pipeline solution. As the actions from one step to another step is already fixed. And there are not many changes to the workflow.

Once the user has written input into the text file, the user can start the program to process the file just written. System will be able to scan the items inside the file and stored in an array list. It will loop each of the items inside the array and use a circular shift method to write out all combinations. The program now will add all combinations into a new array list. Currently this list is not sorted, now when the sorting module receives the data, it will sort the list according to alphabetical order (1 - 10 first, and a - z with ignorance of case sensitivity). Lastly the program will print all the data in this new list into a text file for the user to view the output.

In comparison of both designs, ADT may be the best option for performance and reuse. However, our team feels that the Pipes and Filters solution makes more sense for us as it is easier to understand and it mostly follows the natural modeling technique. As the problem is described as a series/flow of operation on input and output the result eventually, so Pipes and Filters also fit the problem closely. Besides, if any changes in algorithm or function, the Pipes and Filters solution provides better for changes than ADT. However, in terms of data space and performance speed, ADT will be the better option as the Pipes and Filters solution generally needs to wait from one input to another. In a nutshell, we will need to sacrifice some performance issues if we use the Pipes and Filters solution.

4. Limitation & Benefits of Selected Designs

The Abstract Data Types design the performance is better compared to Pipes and Filters. Which means Abstract Data Types can complete the same job faster than Pipes and Filters.

As Abstract Data Types some modules are able to work at the same time while for Pipes and Filters it has to wait for the previous process to be completed and passed the data over to the next function.

Abstract Data Types

Limitation:

1. Hard to add new functions to it
2. If data representation is changed, circular shift and alphabetizer must be updated accordingly.
3. If want to add new function have to compromising the simplicity and integrity
4. Add new functions will affect the program performance as well

Benefits:

1. Easily change 1 module without affecting other modules
2. Can be easily reuse

Pipes and Filters

Limitation:

1. Cannot be modified to use on an interactive system
2. Storage usage is inefficient as each process they have to copy the entire data

Benefits:

1. It has an intuitive flow
2. It can be reuse as each is functioning individually
3. Easy to add new functions as long added to the correct sequence
4. Easy to modify each function since they work in isolation

When I started to code this implementation, the first thing I would like to achieve is to be able to read the file where the specific file name is given in the program. This will be the very first function in the solution.

After that I can add new functions to the program to achieve the goal. And this will not affect the first function I have already implemented.

During testing, I am able to add extra functions to output additional data to tell myself what is the data stored in the variable. This helps me to isolate the issue.

5. Any other information <Optional>

Perform Unit Test when designing Pipes and Filters. Add a print function to each of the variables to verify if it is containing the desired data. If the data is wrong, I will know roughly where it went wrong. This can help me to isolate the problem faster.

For test cases, beside the 3 same test cases, we have added 2 more test cases. We are using the given 3 sample cases first to confirm our output data is having the same order of the sample output. Once this is completed, we will test with the 2 test cases we have made. For the new test cases we will compare the output made by both solutions and see are there any differences like the order of the rows.

6. Contributions

We have decided to choose 1 solution each and work on it individually due to the time constraint. And scheduled meetings during weekends to discuss our codes and the report writing.