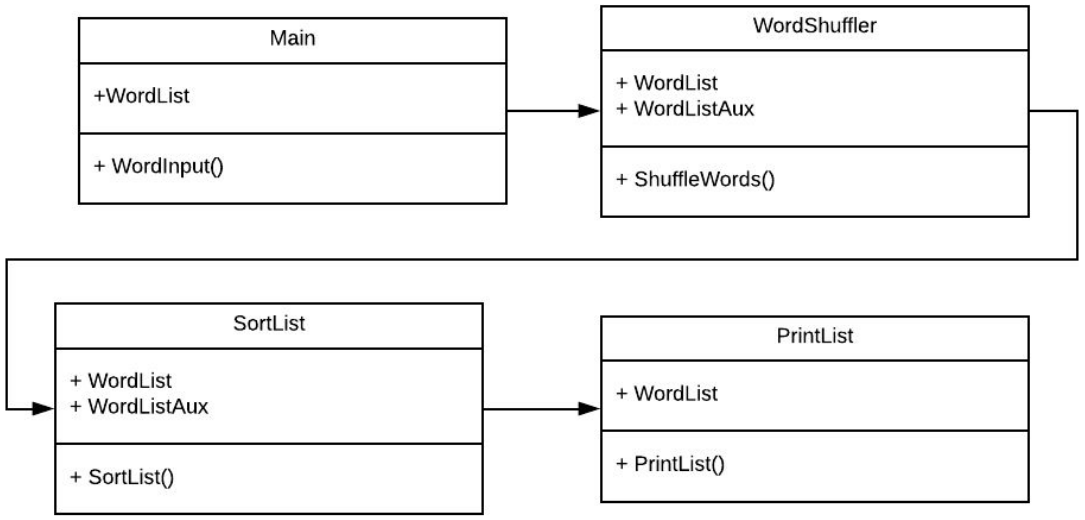
Arturo Cantú Cisneros A01196412
Juan Carlos De León A01233146
Marving Robles Angeles A01651377
Blanca Leticia Badillo Guzmán A00511262

# KWIC Con Pipes and Filters

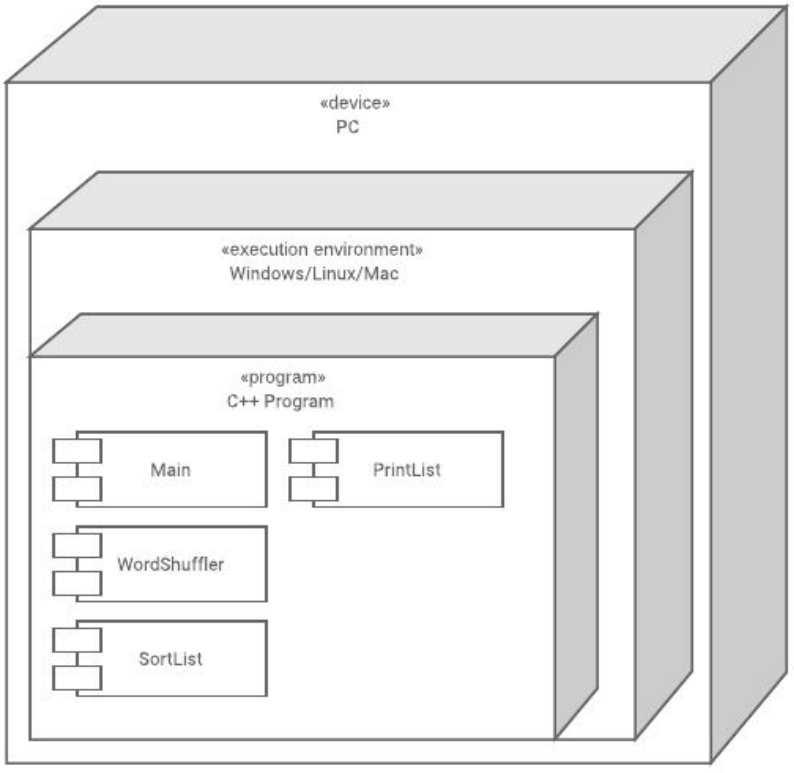
## Descripción de la arquitectura con el Modelo 4+1

### Vista Lógica



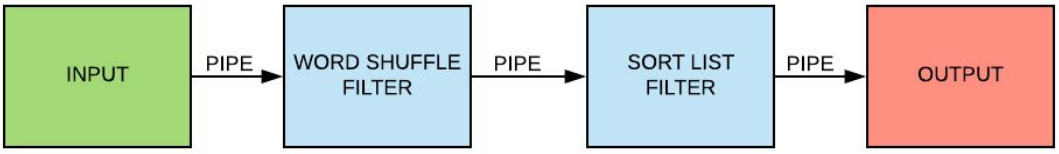

## Vista de Desarrollo

Para el funcionamiento del proceso es necesario un compilador de C++, en cualquier sistema operativo. El código fuente debe ser compilado para ejecutarse, ejemplo usando gcc: g++ kwik -o kwik.cpp



## Vista de Procesos

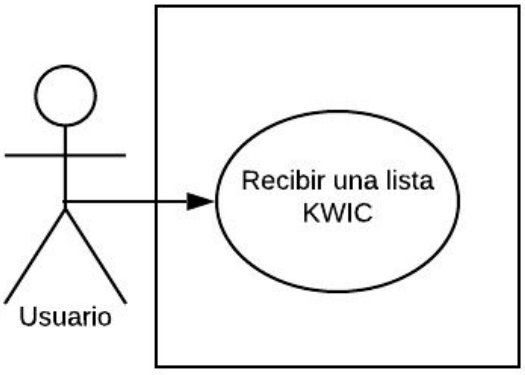
Utilizaremos un procesamiento secuencial, gracias al diseño de arquitectura que utilizaremos de Pipes and Filters.
Se recibe el input del usuario en el main y se manda esa información a la clase WordShuffler.
WordShuffler hace las combinaciones posibles con la frase o frases y manda esa información a SortList.
SortList Ordena la lista alfabéticamente y manda esa información a la clase PrintList.
PrintList Imprime esta información en terminal.




## Vista Física

Para la vista física no es necesario un procesador multi thread debido a la arquitectura de este sistema, la cual trabaja de manera secuencial, esperando a que termine el proceso anterior para comenzar a ejecutar el otro. Esto también es necesario ya que el siguiente filtro requiere de la información del filtro anterior para continuar la ejecución del programa, por lo que un approach multi thread no sería idóneo en este caso.

## Vista de Escenarios




ID: RF1
Titulo: Recibir una lista KWIC
Actores: Usuario
Objetivo: Recibir una lista ordenada con las variantes circularmente rotadas de las palabras o frases ingresadas.
Descripción: El usuario recibe una lista alfabéticamente ordenada (ascendente) con todas las variaciones circularmente rotadas de las palabras o frases que ingresó.
Precondiciones: El usuario ingresó una serie de palabras o frases como input.

Flujo Principal:

1. El usuario ingresa una serie de palabras o frases.
2. El sistema regresa una lista alfabéticamente ordenada (ascendente) con todas las variaciones circularmente rotadas de las palabras o frases que ingresó.

Flujo Alterno:

1. El usuario ingresa un input vacío.
2. El sistema regresa un error “Input vacío”.

Post Condición: Tener una lista alfabéticamente ordenada (ascendente) con todas las variaciones circularmente rotadas de las palabras o frases que ingresó.

# Introduction

The KWIC index system accepts 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 list of all circular shifts of all lines in alphabetical order.

# Installation and Usage

- Everyone can download the code here:
- You will also need an IDE capable of compiling C++ code or gcc compiler.
- GNU makefile functionality.
- Compile with `make all`.
- Run the kwic executable.
- Input the name of the file with the sentences to reorder and sort.

# Requirements

- Functional
    - The program should display all combinations for both sentences
    - The program sorts alphabetically the combinations
    - The program only needs two sentences as a input
- Non-Functional
    - The program has to be done in C, C++ or Java
    - Allows input from keyboard
    - The program follows  Pipes and Filters pattern
    - Each module should be unit-testable

# Architectural Design

## Pipe and Filter

This pattern provides a structure for situations where system process data flux. Each component performs a different step of computation, following a precise order of operations on ordered data.
The architecture consist of two main modules in which is performed a different activity.

**Main:** has the main role, receives the inputs and starts the process through the filters.
**WordShuffler:** In this class is received a stringstream with the sentences and concatenates all combinations, then sends through a pipe to the next filter.
**SortList:** This filter sorts the combinations and passes the data to the last filter
**PrintList:** This last filter receives the stringstream and separates it into sentences to print each one