Reporte de Práctica 01
Problema 08: Simulación de TDA en aplicación para Android

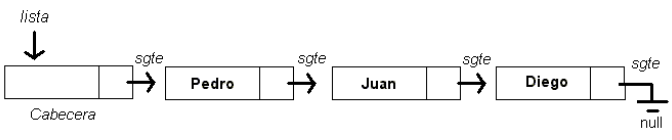
José Antonio Molina De la Fuente*
*Ingeniería en Tecnologías de la Información
Universidad Politécnica de Victoria

***Resumen*—Se reporta acerca del desarrollo de una aplicación movil para el S.O Android realizada en el lenguaje de programación Java, que permite simular un *Tipo de Dato Abstracto* tipo lista enlazada. La aplicación permite al usuario realizar operaciones como insertar al frente, insertar al final, eliminar al frente, eliminar atras y busqueda. Ademas, se visualiza en un TextView el estado de la lista actual. Despues de realizar algun cambio en la lista es guardada en un archivo de texto. Esta aplicación fue realizada utilizando controles y funciones del SDK de Android, ademas de la implementación de ciertos algoritmos para la manipulacion de la lista enlazada.**

I. Introducción

Un concepto muy importante que hay que explicar es el de estructura de datos, lo cual se podría decir que es una forma de representar información. Así como se usa una variable de tipo array para representar un número finito de elementos, se puede representar una lista en una estructura de datos de tipo lista enlazada (ejemplo).[**1**] Las estructuras de datos no solo representan la información, también tienen un comportamiento interno, y se rige por ciertas reglas/restricciones dadas por la forma en que esta construida internamente.[**2**] Un Tipo de dato abstracto (TDA) es un conjunto de datos u objetos al cual se le asocian operaciones. El TDA provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en como estén implementadas dichas operaciones. Esto quiere decir que un mismo TDA puede ser implementado utilizando distintas estructuras de datos y proveer la misma funcionalidad.[**3**]

En este caso, se enfoco en un TDA de tipo lista enlazada por lo que es necesario conocer lo que es una lista. Una lista se define como una serie de N elementos E1, E2, ..., EN, ordenados de manera consecutiva, es decir, el elemento Ek (que se denomina elemento k-ésimo) es previo al elemento Ek+1. Si la lista contiene 0 elementos se denomina como lista vacía.[**3**]



[**4**]
Representación de un TDA tipo lista enlazada.

La aplicación fue realizada en Android Studio el cual es el entorno de desarrollo integrado oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.[**5**]

Android Studio está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de Android.[**5**]

II. Desarrollo Experimental

En este trabajo se desarrolló una aplicación móvil para el S.O Android en el lenguaje de programación Java, el cual permite simular un TDA o lista enlazada, pudiendo realizar operaciones de inserción tanto al principio y al final de la lista, asi como elminación al principio y final de esta tambien por parte del usuario. Ademas de esto, el usuario puede realizar una búsqueda de un elemento y la aplicación muestra en que posiciones se encuentra el elemento que se ingreso en el criterio de busqueda, es decir si se ingresa como entrada el elemento **X**, la aplicación mostrara la posición o posiciones en las que se encuentra este elemento en la lista empezando obviamente desde la posición **0**.

Para la realización de la aplicación, se investigó acerca de sintaxis del lenguaje de programación Java, ademas de la instalación del JDK de Java y SDK de Android Studio. Ademas de esto, se tuvo que investigar acerca de los TDA y listas enlazadas para poder conocer como estan estructuradas, como funcionan y familiarizarse con las operaciones que se les pueden realizar a estas. Una vez investigados estos temas, se procedió a realizar un análisis detallado acerca de los requerimientos de la práctica y la entrada que recibirá la aplicación. En este caso en particular, la entrada por parte del usuario es un *elemento* de cualquier tipo de dato. Posteriormente, se colocaron botones en el Layout de la aplicación los cuales permiten; *Insertar al frente de la lista* (inserta el elemento al principio), *insertar al final de la lista* (inserta el elemento al final), *eliminar al frente de la lista* (elimina el primer elemento), *eliminar al final de la lista* (elimina el ultimo elemento) y finalmente *buscar*, ademas de dos controles de tipo *TextView*, en uno se visualiza el estado

de la lista actual y en el otro se visualiza el resultado de la búsqueda. Finalmente se tiene un control de tipo *EditText* el cual corresponde a la entrada de la aplicación, es decir el usuario ingresa la entrada en esa caja de texto.
Despues de realizar el diseño de la aplicación se procedió a colocar dos líneas de código las cuales corresponden a la petición de permisos de escritura y lectura de archivos en Android. Esto debido a que es necesario guardar la lista enlazada en un archivo con extensión **txt**. Las líneas son las siguientes:

⟨*uses-permissionandroid:name=”android.permission.WRITE_EXTERNAL_STORAGE”*⟩⟨*/uses-permission*⟩
⟨*uses-permissionandroid:name=”android.permission.READ_EXTERNAL_STORAGE”*⟩⟨*/uses-permission*⟩

Ademas de esto, se utilizaron algunas funciones para la peticion de permisos al usuario para lectura y escritura de archivos. Lo anterior dicho fue obtenido de un demo presentado en clase.

Cabe destacar que se utilizo el demo proporcionado en clase, en el cual se implementa la escritura y lectura de archivos de texto del almacenamiento interno del dispositivo. Por lo que se implemento parte de este código.
Sabiendo que la única entrada de texto por parte del usuario es un *elemento* elemento de cualquier tipo, se realizó una función de validación la cual consiste básicamente en verificar que el elemento no esté vacío.
Como paso inicial para la lectura y escritura de datos, primeramente, se hizo uso de una función que consiste en la creación del archivo de texto en el directorio en caso de que el este no exista. En dicho archivo se guardará la lista enlazada. Posteriormente, se procedió a diseñar y programar un algoritmo que, en base un nombre de archivo, realiza la lectura de este en el directorio señalado y posteriormente, lee cada línea del archivo de texto y almacena en un *ArrayList* cada linea que contiene este (cada linea corresponde a un elemento de la lista enlazada).

Posteriormente se procedió a programar las funciones encargadas de realizar operaciones a la lista enlazada. Primeramente se realizó la función *insertar al frente*, para ello se diseñó e implementó por cuenta propia, un algoritmo, el cual dado un *ArrayList* que contiene los elementos de la lista enlazada leídos desde el archivo de texto en el almacenamiento, se crea un nuevo *ArrayList* del tamaño de la lista anterior + 1, posteriormente se realiza la inserción del elemento que fue dado de entrada por parte del usuario en la primera posición del nuevo *ArrayList* y posteriormente se trasladan los elementos restantes de la lista original a la nueva lista mediante un ciclo for, actualizando las posiciones a *i + 1*, en donde *i* es la posición original del elemento de la lista antigua. Posteriormente se remplaza el contenido del archivo de texto que almacena la lista con la nueva lista creada de tal manera que cada elemento se coloca en una línea del archivo de texto y se procede a guardar el contenido del este.

La siguiente función que se realizó fue la de *insertar al final*, la cual consiste en insertar un elemento al final de la lista enlazada. Para realizar esto se diseño e implementó un algoritmo muy similar al anterior. De igual manera, se lee el archivo de texto del almacenamiento el cual contiene la lista enlazada despues de realizar alguna operación y se obtiene un *ArrayList* el cual contiene los elementos de la lista y posteriormente, se crea un nuevo *ArrayList* el cual corresponde a la nueva lista del tamaño de la lista anterior + 1. Despues de esto, se trasladan todos los elementos de la lista antigua a la lista nueva mediante un ciclo for, y finalmente, en la ultima posición de esta se inserta el elemento dado de entrada por el usuario. Posteriormente se procede a guardar la lista en el archivo de texto.

Despues de esto se realizó la función *eliminar al frente*, la cual consiste en borrar el primer elemento de la lista enlazada. Para realizar esto se diseño e implementó un algoritmo, de igual manera, se lee el archivo de texto del almacenamiento el cual contiene la lista enlazada despues de realizar alguna operación y se obtiene un *ArrayList* el cual contiene los elementos de la lista y posteriormente, se crea un nuevo *ArrayList* el cual corresponde a la nueva lista del tamaño de la lista anterior - 1. Despues de esto, se trasladan todos los elementos de la lista antigua a partir de la **posición 1** a la lista nueva mediante un ciclo for con inicio en *i = 1 hasta i menor o igual al tamaño del ArrayList antiguo*, de tal manera que se excluye el elemento en la posición 0. Posteriormente se procede a guardar la lista en el archivo de texto.

La siguiente función que se realizó fue *eliminar al final*, la cual consiste en borrar el ultimo elemento de la lista enlazada. Para realizar esto se diseño e implementó un algoritmo, de igual manera, se lee el archivo de texto del almacenamiento el cual contiene la lista enlazada despues de realizar alguna operación y se obtiene un *ArrayList* el cual contiene los elementos de la lista y posteriormente, se crea un nuevo *ArrayList* el cual corresponde a la nueva lista del tamaño de la lista anterior - 1. Despues de esto, se trasladan todos los elementos de la lista antigua a partir de la **posición 0** a la lista nueva mediante un ciclo for con inicio en *i = 0 hasta i menor al tamaño del ArrayList antiguo disminuido en una unidad*, de tal manera que se excluye el elemento en la ultima posición del *ArrayList* antiguo. Posteriormente se procede a guardar la lista en el archivo de texto.

Otra de las funciones que se realizaron fue la de *buscar elemento*, la cual consiste en dado un elemento de entrada, se procede a verificar los elementos de la lista, si alguno(os) coinciden con este, se guarda la/las posicion(es) en donde se encuentra. Para realizar esto se diseño e implementó un algoritmo, de igual manera, se lee el archivo de texto del almacenamiento el cual contiene la lista enlazada despues de realizar alguna operación y se obtiene un *ArrayList* el cual contiene los elementos de la lista y posteriormente, se realiza un ciclo for, el cual accedera a cada uno de los elementos

del ArrayList y verifica si es igual al elemento dado de entrada, de ser cierto, se guarda la posicion y asi con todos los elementos. Finalmente esta funcion muestra al usuario a traves del *EditText* de busqueda que posiciones o posicion hicieron match con el elemento ingresado.

Es importante mencionar que, se realizaron funciones auxiliares como lo es *showList* la cual lee el archivo de texto del directorio y muestra en pantalla la lista enlazada. Esta funcion es ejecutada cada vez que se realiza alguna operación a la lista, para que la visualizacion de esta sea actualizada.

De igual manera, al realizar alguna operación a la lista o alguna busqueda, se es guardado en otro archivo de texto en el directorio, el registro de las operaciones realizadas a la lista. Para ello, se tuvo que diseñar e implementar un algoritmo que, dada una operación realice una escritura en un archivo de texto dado un nombre, en este caso el nombre del archivo es log Por lo que cualquier usuario con una aplicación de administración de archivos puede localizar este y el archivo en el que se almacena la lista en el directorio siguiente: `Almacenamiento Interno / Problema8_CDM`, dentro de esta carpeta se encuentra el archivo `lista_enlazada.txt` y `log.txt`.

Para ver un panorama más general del proceso realizado, a continuación se muestra un diagrama de flujo:

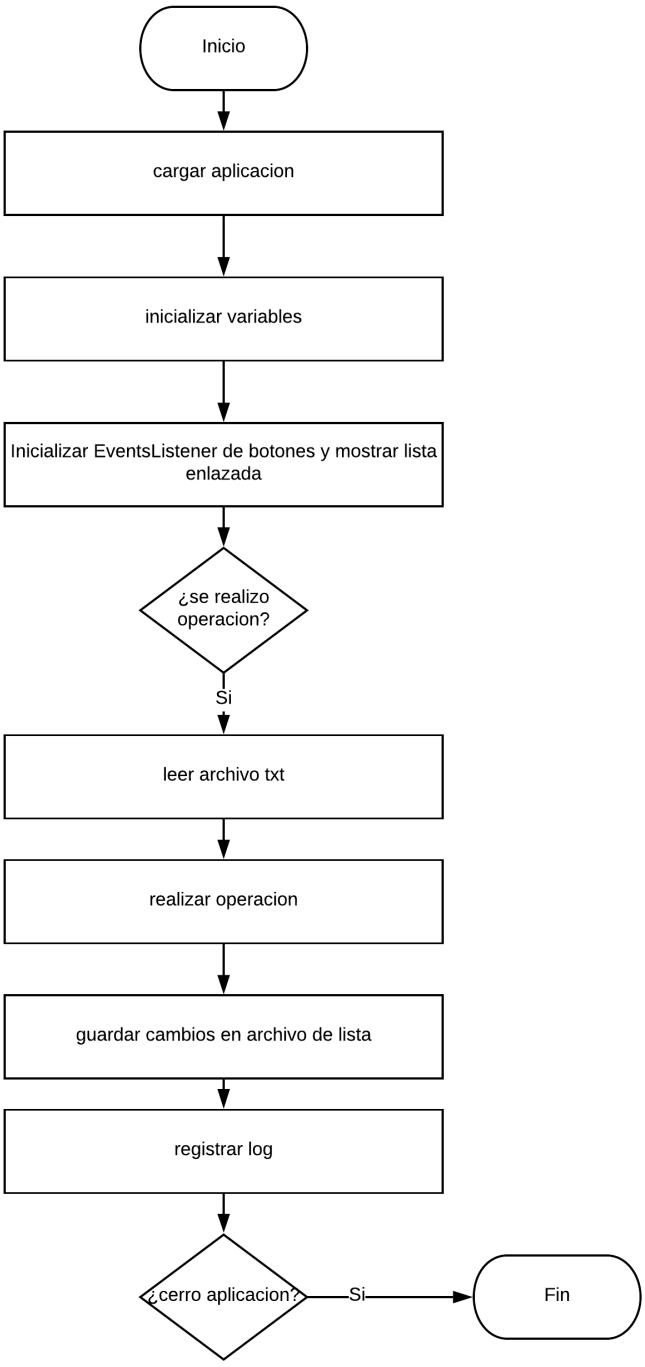


Diagrama de flujo de la aplicacion.

III. Resultados

El programa deberá compilarse de la siguiente manera:

Después de ejecutarlo aparecerá en consola lo siguiente, lo cual es información e instrucciones del uso del programa:

Abrimos la ventana de nuestro programa y nos encontraremos con lo siguiente:

En donde aparece la información del desarrollador y los controles, con la tecla + se podrá cambiar las medidas de los lados del cuadrado y con la tecla escape (ESC) se cierra el programa. Así como también, la pantalla nos muestra las

longitudes de lado de cada cuadrado(a longitud del cuadrado A y así sucesivamente). Además, nos encontramos con los dos lados de los cuadrados que podemos mover. Los puntos en color negro corresponden a vértices de los cuadrados, usted puede dar clic y moverlos a razón de radio de una circunferencia. A continuación se procede a intentar a unir (debido a que no en todos los casos será posible) para formar el triángulo y obtener el ángulo entre las dos semirrectas.

Una vez unidos los vértices (lados juntos) se colorea el triángulo formado con color azul y se muestra la medida del ángulo que en este caso fue de 60.94 grados.

Si queremos cambiar las medidas o reiniciar el simulador se presiona la tecla +. Finalmente presionamos la tecla escape para salir.

IV. Conclusión

En este trabajo se ha dado solución a un simulador de tres cuadrados mediante el cual, dada la medida de sus lados, se puede saber si pueden o no formar un triángulo con estos y calcular el ángulo comprendido entre dos de sus rectas. Para lograr esto se tuvo que realizar una investigación acerca de diversos temas relacionados con la trigonometría y fórmulas para calcular puntos o pendientes de rectas, así como también sobre el lenguaje C++. En el desarrollo de este programa se pusieron en práctica habilidades de investigación y análisis.

Referencias

[1] Triángulo,http://www.universoformulas.com/matematicas/geometria/triangulo/, visitado 08/10/2017

[2] Concepto de triángulo,http://conceptodefinicion.de/triangulo/, visitado 08/10/2017

[3] Ángulos,https://www.smartick.es/blog/matematicas/recursos-didacticos/angulos-i/, visitado 08/10/2017

[4] Funciones trigonométricas,https://sites.google.com/site/magiamatematica2015/funciones-trigonometricas, visitado 08/10/2017

[5] Imagen función trigonométrica,https://es.wikipedia.org/wiki/Funcion_trigonometrica, visitado 08/10/2017