UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Buenos Aires

Algoritmos y Estructura de Datos

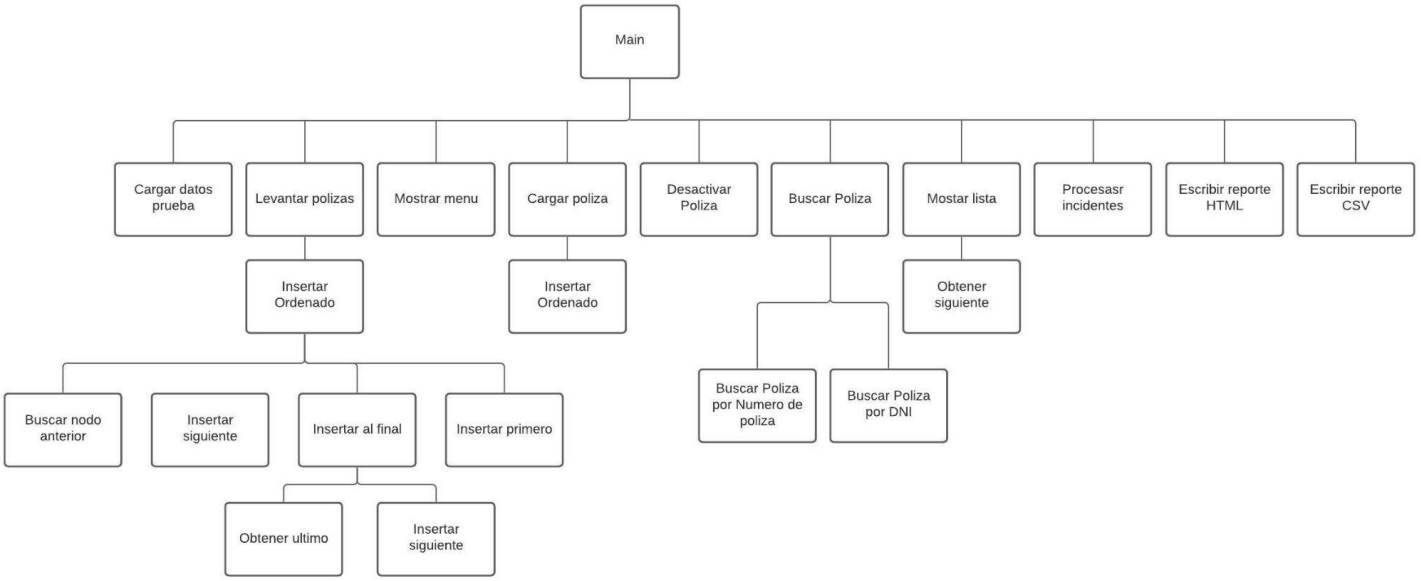
–2021–

Docente: Ing. Pablo Damián Méndez

Trabajo Práctico N.º 2

Localización: CAMPUS		**Curso: K1025**	
Integrantes del equipo: 4			
Legajo	**Apellido y Nombre // Correo Institucional**		
204 016-5	Roldan Gael // garoldan@frba.utn.edu.ar		
203 464-5	Canossa Joan Franco // jcanossa@frba.utn.edu.ar		
203 954-0	Pesce Franco // fpesce@frba.utn.edu.ar		
173 245-6	Giussani Ignacio Franco // igiussani@frba.utn.edu.ar		
Repositorio GitHub:	https://github.com/jfr4nc0/TpIntegrador.git		
Fecha de entrega	**27/10**		

Diagrama de bloque de subprogramas:



Informe:

Lo primero que se realizó tras haber comprendido bien lo pedido por el trabajo en cuestión, fue idear la estructura general que tendría el programa. La estructura se basó en 3 ejes principales:

1. **Funciones que deben cargar previamente al menú principal:** levantamiento de pólizas a una estructura de datos y mostrar el menú principal.

2. **Funciones que deben cargar en el menú principal:** acciones requeridas sobre la estructura de datos en memoria y la posibilidad de cargar casos de prueba.

3. **Funciones que deben cargar posteriormente al menú principal**: sobrescritura del archivo de pólizas con los cambios realizados por las acciones del menú principal.

Respecto al primer eje, se debían levantar las cuentas de las pólizas previamente existentes en el archivo “Asegurados.BAK” hacia alguna estructura de datos que se determinara. Para realizar este traslado de lo almacenado en el archivo a estructura de datos, se acordó utilizar listas. Esta decisión fue tomada principalmente porque se desconoce la cantidad de datos a almacenar, por consecuencia también se desconoce qué tamaño debería tener la estructura de datos seleccionada. La lista al ser una estructura dinámica, sacrifica algo de rendimiento a cambio de brindar la posibilidad de trabajar con una cantidad indefinida de datos, además de una mayor versatilidad al realizar operaciones. Todo el levantamiento se lleva a cabo en la función llamada “LevantarPolizas”, donde se insertan, de forma descendente respecto a la cantidad de incidentes con otra sub-funcion los datos almacenados en el archivo hacia una lista.

Como última función del primer eje se creó una función la cual despliega por pantalla un menú de opciones “MostrarMenu”, que integra todas las funciones desarrolladas en el segundo eje.

En cuanto al segundo eje, siendo la rama principal del desarrollo del programa. Se planificó, punto por punto, resolver las peticiones del sistema con su respectiva verificación en caso de mal funcionamiento. Las funciones son:

2) Cargar una nueva póliza

En este punto, el objetivo era que el usuario ingrese dato por dato una nueva póliza. Luego de obtener los datos, el programa insertará de forma ordenada los datos en la lista previamente creada.

3) Desactivar una póliza existente

Este punto, tuvo la finalidad de cambiar el valor de activación de una póliza. Por lo tanto el programa pide ingresar el número de póliza a desactivar, luego, mediante un nodo auxiliar, recorre la lista hasta encontrar el número de póliza ingresado. Posteriormente cambia el valor activado, de 1 a 0, siendo 1 “activado” y 0 “desactivado”.

4) Buscar una póliza

El accionar en este punto fue, permitir que el usuario decida de qué forma va a buscar la póliza, ya que puede ser por DNI o por Número de póliza. Una vez que el usuario elige de qué forma quiere buscarla, el subprograma llama al subprograma correspondiente a la búsqueda.

Si es por DNI, el programa solicitará el DNI y busca, nodo por nodo, hasta encontrar el solicitado. Una vez lo obtiene, muestra los datos del nodo por pantalla. Cuando encuentra los datos, sigue recorriendo la lista, ya que una persona puede tener más de una póliza. En caso de encontrar otra póliza con el número de DNI ingresado, vuelve a mostrar los datos correspondientes.

Si es por Número de póliza, el programa solicita el Número de póliza y busca, nodo por nodo, hasta encontrar el solicitado. Una vez que lo obtiene, muestra los datos del nodo por pantalla.

5) Listar pólizas activas ordenadas por saldo descendente:

Primero se verifica que las pólizas estén activas y se procede a mostrar los datos de los miembros correspondientes, en caso contrario se omite la póliza. Teniendo en cuenta que al principio del programa se “levantan” las pólizas de forma ordenada, también se posicionan en orden descendente.

6) Procesar un lote de incidentes:

Este subprograma hace lo siguiente, primero pide al usuario que elija qué lote de incidentes quiere procesar, una vez elegido abre el archivo del lote y, un nodo auxiliar, compara los datos de la lista con los datos del archivo hasta encontrar un número de póliza que coincida con el del incidente. Posterior a esto, se le aumenta en 1 la cantidad de incidentes a la póliza encontrada. **Si no hay ningún número de póliza que coincida con los datos de los incidentes, no se carga en el sistema.**

7) Mostrar todas las pólizas que no tengan la cuota al día, en formato HTML y CSV:

Se crea un reporte en ambos formatos, donde se selecciona las pólizas que no posean su cuota al día y posteriormente imprimiendo sus valores correspondientes listados en los archivos “ReportePoliza.html” y “ReportePoliza.csv”

8) Finalizar jornada, sobreescribir “Asegurados.BAK”:

Una vez que finalicen las acciones efectuadas ya mencionadas, esta opción sobrescribe todos los cambios realizados en el archivo Asegurados.BAK, dando fin al programa.

9) Procesamiento de prueba:

Se debían levantar las cuentas de las pólizas previamente existentes en el archivo “Asegurados.BAK”. Al realizar el programa por primera vez, este archivo no existía y por ende no contenía ninguna cuenta, por lo tanto se debió implementar casos de prueba. Para esto creamos una función que se encargará de crear el archivo “Asegurados.BAK” junto a algunas cuentas almacenadas. Además en esta función se realizó la creación de 2 lotes de incidentes, necesarios para el procesamiento posterior. La función involucrada fue llamada “CargarDatosPrueba”.

Casos de prueba

En cuanto a los casos de prueba, se carga 2 lotes de incidentes con todos los casos posibles:

- Del lote 1, hay 2 incidentes que no coinciden con el numero de poliza almacenado en el sistema, y 2 que sí coinciden
- Del lote 2, también hay 2 incidentes que no coinciden con el numero de poliza almacenado en el sistema, y 2 que sí coinciden

En cuanto al archivo de pólizas (Asegurados.BAK) hay 5 casos:

- Uno que tiene la póliza activa y con la cuota al dia
- Uno que tiene la póliza activa, pero no la cuota al dia
- Uno que no tiene la póliza activa, pero la cuota al dia
- Uno que no tiene la póliza activa y no tiene la cuota al dia
- Y uno que repita el DNI para el caso de buscar póliza por DNI

División de tareas del equipo:

- Roldan Gael se encargó de hacer los subprogramas de buscar póliza y mostrar lista
- Canossa Joan Franco se encargó de hacer los subprogramas de los reportes, tanto en HTML, como en CSV, y Mostrar lista
- Giussani Ignacio Franco se encargó de hacer los subprogramas de cargar póliza, desactivar póliza y sobreescribir los datos
- Pesce Franco se encargó de hacer los subprogramas de procesar incidentes y levantar pólizas
- Todos los miembros se encargaron de la corrección de Bugs.

Además, cada uno aportó con la escritura y decisión de la carga de datos de prueba, ya que eran muchos.