

Link al repositorio: https://github.com/JavierVernengoMaffei/TP1

Usuario GitHub: ATrossero

Integrantes

Correo Institucional

aagonzalez@frba.utn.edu.ar

atrossero@frba.utn.edu.ar

jvernengomaffei@frba.utn.edu.ar

2036939

2041182

1765656

Legajo

Javier Vernengo Maffei

Agustín Francisco Trossero

Aaron Isaac González

***Informe (“Hipótesis”)***

Primero que nada, se comienza creando las structs correspondientes a incidentes y pólizas. Además, creamos un menú con las diferentes operaciones que próximamente describiremos. Luego, cuando comenzamos a intentar imaginar el funcionamiento del programa, podemos reconocer que la cantidad de incidentes y de pólizas no es una constante, por lo cual, necesitamos utilizar listas. Ahí creamos las structs de los nodos de ambos y comenzamos con la primera consigna. Para “levantar” los archivos decidimos hacerlo automáticamente cuando comienza la ejecución del programa, ya que, todas las operaciones que vayamos a realizar son más accesibles con listas. Aquí abrimos los archivos “Asegurados.BAK” y “Lote1”, y a medida que los vamos leyendo, vamos insertando la información en nuestras listas hasta llegar al finalizar los archivos, momento en el que los cerramos. Una vez levantados los archivos a las listas comenzamos con la función cargarPoliza, la cual está diseñada para recibir los parámetros adecuados a la estructura ya establecida. Luego seguimos con la próxima consigna en la cual le pedimos al usuario el número de póliza (no pedimos el DNI porque una misma persona puede tener diferentes pólizas) para buscar en la lista el nodo correspondiente. Para ello vamos recorriendo los nodos de la lista hasta encontrar la coincidencia. Una vez encontrado el parámetro activa se lo establece como false, y si no se lo encuentra se muestra por pantalla la advertencia. Para la próxima consigna decidimos separarlas en dos búsquedas diferentes, ya que, como mencionamos previamente, una misma persona puede tener varias pólizas. El buscar póliza por número de póliza resulta más sencillo ya que solamente se recorre la lista una sola vez (solo puede haber una póliza con el mismo número), mientras que buscando por DNI se crea un vector de strucs en memoria dinámica para poder almacenar las diferentes pólizas a un solo nombre, mostrarlas por pantalla y luego borrar ese vector strucs. Para el listado de las pólizas activas por cantidad de accidentes descendientes, primero que nada, se aplica un ordenamiento a la lista en orden descendiente, para luego ir mostrando por pantalla las pólizas activas. Volvemos a crear todas las funciones necesarias para manejar una lista, pero en este caso correspondientes a la struct de incidentes. En este momento creamos las opciones de procesar el lote y la opción insertar un nuevo incidente. Para procesar el lote lo abrimos el archivo “Procesados.BAK”, en el cual los vamos a almacenar. En este procedimiento se van leyendo las listas de incidentes y pólizas. La lista de incidentes se va guardando en el archivo “Procesados.BAK” y eliminando, no sin antes verificar si en la lista de pólizas existe una coincidencia en el número de pólizas. En caso afirmativo, a esa póliza coincidente se le suma un incidente. Una vez terminado el procesamiento, se eliminan los incidentes de la lista. Para cargar un nuevo incidente se utiliza el mismo método que para cargar una nueva póliza a su lista correspondiente. A la hora de mostrar el reporte tanto por un archivo HTML como exportarlo a un EXCEL, debemos abrir el archivo, establecer la estructura correspondiente, e ir recorriendo la lista buscando pólizas que no tengan la cuota al día. Cada una de ellas que se encuentra, se las imprime en los archivos. Por último, la función de sobreescribirAsegurados no es nada más que sobrescribir el archivo ASEGURADOS.BAK con los datos actuales en el campo info de la lista de pólizas. Para finalizar se borran las listas de pólizas e incidentes que estaban ubicadas en la memoria dinámica.

***División de trabajo:***

Javier: Escritura de structs, datos correspondientes, Programa de CargarDatosPrueba, Diagrama de bloques de subprogramas.

Aaron: Subprogramas correspondientes a incidentes (punto 6), subprogramas creadores de archivos HTML y CSV.

Agustín: Programa main, con subprogramas correspondientes a los puntos 1 al 5, subprograma de finalizar jornada, Informe y casos de validación.

***Casos de validación (Capturas y explicación breve de lo que debía suceder)***

Ejecutamos el programa que carga los datos de prueba:

Ejecutamos el main y seleccionamos cargar póliza.

Texto

Descripción generada automáticamente

Mostramos las pólizas ordenadas (No aparece la desactivada).

Texto

Descripción generada automáticamente

Desactivamos a Juan José.

Texto

Descripción generada automáticamente

Procesamos el lote de incidentes, y vemos si se aumentaron la cantidad de incidentes (mostramos pólizas ordenadas nuevamente). Además se crea el archivo “Procesados.BAK”.

Texto

Descripción generada automáticamente

Se sumaron los incidentes, y se desactivo a Juan José (funcionó correctamente).

Cargamos nuevo incidente

Texto

Descripción generada automáticamente

Procesamos incidentes nuevamente (se actualiza “Procesados.BAK”) y exportamos los archivos a HTML y CSV (Debería aparecer solamente Messi con los incidentes actualizados: **11**).

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Finalizamos Jornada (Sobreescritura del archivo “Asegurados.BAK”).

***Diagrama***

Imagen que contiene texto, pizarrón

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Imagen que contiene pizarrón, plato, parado

Descripción generada automáticamente