Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Algoritmos y Estructuras de Datos
Curso: K 1042

TRABAJO PRÁCTICO INTEGRADOR

Profesor: Ing. Pablo D. Mendez
Integrantes:

García Felipe Alejo
- Legajo: 2036540
- Mail: felgarcia@frba.utn.edu.ar

Gómez Taboada Tomás
- Legajo: 2036903
- Mail: tgomeztaboada@frba.utn.edu.ar

Hidalgo Mora Sofía
- Legajo: 2037233
- Mail: mohidalgo@frba.utn.edu.ar

Descripción de la solución con hipótesis

Antes de la corrección, nos planteamos como primer problema, la definición de las estructuras tanto de conductores como de infracciones. Se nos ocurrió utilizar arrays para ambas pero se nos complicaba la dimensión de los mismos, para esto pusimos diferentes opciones a prueba y nos quedamos con dimensionarlo con una variable que sirviera como tope del array y que al llegarse al mismo se autoincrementa, guardando así los creados y pudiendo ingresar nuevos sin problema.
Luego de la corrección decidimos implementar listas para poder tener un mejor manejo sobre la memoria dinámica. Para esto, tuvimos que duplicar la estructura de los conductores y definir a la original como Archivo(la cual va a ser grabada) y a la otra como Nodo, la cual va a servir para hacer uso de la memoria dinámica. También creamos 2 nodos globales que utilizamos como punteros para el inicio y final de la lista.

Luego de solucionar las dificultades planteadas realizamos el programa que, en primera instancia se preguntará por pantalla la fecha del día actual, luego de introducirla se pasa al menú, dando al usuario la opción de elegir la función que desee ejecutar. Este menú consta de 10 opciones:

- levantar archivo (1);
- cargar nuevo conductor (2);
- desactivar conductor (3);
- leer conductores con infracción en una provincia determinada (4);
- procesar lote de infracciones (5);
- mostrar por pantalla el informe de conductores con detalle de infracciones (6);
- Mostrar todos los conductores con registro vencido entre fechas establecidas (en formato HTML o CSV) (7 y 8);
- finalizar la jornada (9);
- e ingresar una infracción (10)
- Además de estas funciones, se le dará la opción al usuario de salir del programa (0) sin necesidad de grabar el archivo de conductores en disco.

Cabe destacar que el programa siempre libera la memoria al finalizar, ya sea por la opción (9) o la (0).

- Al optar por la opción uno (1), se procederá a abrir un archivo llamado “Conductores.bin“ y se leerá su contenido el cual valorizará la lista conductores. También se imprimirá sólo a los fines de control su contenido en pantalla, en éste se encuentran los conductores que se ingresaron previamente o, de ser la primera vez abierto, en el que se ingresarán los conductores durante la ejecución.

- En la opción dos (2), se le preguntará al usuario que ingrese los datos del conductor a ingresar: ID, fecha de vencimiento y su eMail. La solución que propusimos corrobora que el ID que se ingresa no coincida con la identificación de otro conductor previamente ingresado. Para esto utilizamos un subprograma “BuscarConductor” que recorre la estructura lista de conductores y devuelve un puntero al nodo que contiene a la información de dicho conductor en caso de existir.

De repetirse el ID, el programa anunciará por pantalla que el conductor ya ha sido ingresado. Además en caso de ingresar una fecha vencida, se alertará al usuario y se procederá a crear un nuevo nodo para guardar en memoria este conductor.

- En la tercera opción (3), se le pide al usuario que introduzca el ID del conductor que desea desactivar. Una vez seleccionado, se invocará el subprograma "BuscarConductor" para buscar al conductor que coincida con dicho identificador. Si se encuentra un conductor cuyo ID coincida con el seleccionado, se cambiará el estado del conductor para desactivarlo. Si no se encuentra un conductor que coincida, se mostrará por pantalla que hubo un error ya que no hay un conductor que coincida con el dato ingresado. Y en caso de que el conductor ya se encontrará desactivado, lo comunicará.

- Si se desea leer los conductores que cometieron una infracción en una provincia determinada, el usuario deberá seleccionar la cuarta opción (4). Si esto sucede, se pedirá por pantalla que se ingrese el número de provincia deseado, teniendo en cuenta que el número límite es 23.
  Para esto, decidimos hacer una copia de la estructura de Nodo con la diferencia que esta no es global y contiene únicamente 3 variables (ID, eMail y un puntero al siguiente Nodo), la llamamos rListado. Decidimos crear esta nueva estructura para poder ir guardando los datos de los conductores, para luego imprimirlos por pantalla sin que estos se repitan.
  A continuación, se abrirá el archivo procesados.bin para: 1ero, poder verificar que existan infractores en dicha provincia, 2do si existen, leer la infracción y 3ero, en caso de que el conductor no haya sido registrado en rListado, se guarda el ID y el eMail del mismo en un nuevo nodo de rListado.
  Por último, se imprime por pantalla la información de cada infractor en la provincia seleccionada y se libera la memoria de la lista auxiliar utilizada.

- Para la quinta opción (5), procesar lote de infracciones, se abrirá el archivo "infracciones.bin", donde se encuentran las nuevas infracciones que no han sido procesadas, y "procesados.bin", donde se incorporarán las nuevas infracciones sin procesar. Se corrobora que el ID de las infracciones coincida con un conductor existente y se procesarán las infracciones. Una vez terminado este proceso, se vacía el archivo de "infracciones.bin" y se muestra por pantalla la cantidad de casos leídos y de casos inexistentes. Durante este proceso se incrementa la cantidad de infracciones del conductor para mantener consistente la información.

- En la siguiente opción (6), se mostrará por pantalla el informe de conductores con detalle de infracciones. Si hay datos cargados, se listaran los datos de los conductores seguidos y se entrará a un subprograma (ImprimirDetalleInfracciones) que abrirá el archivo de "procesados.bin" en modo lectura y mostrarán por pantalla los detalles de las infracciones que posea el conductor: fecha, monto y provincia.

- Al seleccionar la opción siete (7), mostrará en formato HTML los registros vencidos entre fechas establecidas por el usuario. Esta opción generará una tabla con el ID del conductor, la fecha de vencimiento de su registro y el e-mail del conductor. Para lograrlo se utiliza un puntero a la lista de conductores, y, al recorrerla, se busca si hay algún conductor con el registro vencido (esto se sabe gracias a la fecha del día introducida por el usuario al principio del programa). Si es así, se verifica que la fecha de vencimiento se encuentre entre las ingresadas por el usuario y se genera el archivo HTML con la tabla de datos. Esta opción debe utilizarse antes de terminar la jornada con la opción nueve (9), de lo contrario se eliminarán todos los

registros vencidos. La opción ocho (8) funciona de la misma manera que la siete (7) pero en vez de generar un archivo HTML con la tabla, lo genera en CSV utilizando los mismos datos.

- Si el usuario ingresa la opción nueve (9), se finalizará la jornada. Esto significa que se procederá a leer la lista almacenada en memoria y se grabará el archivo “conductores.bin“, que se ha modificado durante la ejecución del programa y, se verificará en los datos si están activos y dentro del período de vencimiento. Cabe destacar que la consistencia del total de infracciones se produce al incrementar dicho totalizador únicamente con la opción (5) del menú (y no cuando se ingresan nuevas infracciones con la opción 10). El procedimiento para finalizar la jornada, además, tomará la cuenta de los conductores actualizados, inactivos y con fechas vencidas para luego informar las cantidades por pantalla.

- Por último, el usuario cuenta también con la opción de ingresar infracciones(10). La misma, le permite grabar al usuario de a una infracción por vez, para luego guardar los datos anexándolos en el archivo “infracciones.bin“. Previamente se corroborará que el ID del conductor coincida con alguno existente utilizando el subprograma “BuscarConductor” mencionado anteriormente.

- En la solución propuesta, el programa sólo se cerrará optando por la opción nueve(9) o la cero (0) ofrecida en el menú. Si el usuario selecciona esta última opción, no se realizará ninguna acción, más que la de finalizar el programa liberando la memoria.

División de tareas

Luego de recibir, leer y analizar las consignas del trabajo, los 3 integrantes tuvimos una reunión en la cual definimos lo necesario para poder iniciar a diagramar cada uno de los diferentes subprogramas necesarios. En dicha reunión diseñamos la estructura global de nuestro programa. Lo primero fue definir el menú, después, optamos por utilizar un “switch” con múltiples “case” que a su vez invoquen diferentes subprogramas para realizar las opciones del menú. También en esa primera reunión nos propusimos resolver la forma de las estructuras de conductores e infracciones. Luego de la corrección nos reunimos nuevamente para implementar un nuevo método más efectivo para resolver los problemas.
Una vez aclarado todo, nos dividimos de forma equitativa los diferentes subprogramas a realizar, intentando que los que estén relacionados los hiciera la misma persona. En el proceso surgió por necesidad, crear otras 2 opciones al menú para poder llevar a cabo una solución eficaz de lo pedido.
Por último para realizar el informe y hacer un testeo completo del programa nos reunimos nuevamente los 3.

Diagrama de bloques