Algoritmos y Estructuras de Datos (2023)

Trabajo Práctico 04: Gestión de Cabinas de Peaje [4.0]

Por hacer: Hacer un envío **Por hacer:** Recibir una calificación

Apertura: lunes, 25 de septiembre de 2023, 08:00
Cierre: domingo, 15 de octubre de 2023, 23:59

Aclaración importante 1: El desarrollo de este Trabajo Práctico está pensado para poder desarrollarse en base a las herramientas y temas que se vean hasta la Ficha 25 inclusive. Entre otros, los temas abarcados por esas fichas incluyen *arreglos de registros/objetos, ordenamiento, búsqueda, conteo/acumulación por acceso directo en un arreglo unidimensional o bidimensional, y gestión de archivos de texto y archivos binarios*. No deberían tener necesidad de utilizar temas y elementos de fichas posteriores.

Aclaración importante 2: Por favor, LEA TODO EL CONTENIDO DE ESTAS CONSIGNAS ANTES DE COMENZAR A TRABAJAR. No se aceptarán reclamos de ningún tipo por errores que pudiese cometer debido a no haberse tomado el trabajo de leer las consignas que se le fijaron.

a. Enunciado y Consignas[1].

(Click aquí para descargar el archivo de entrada en formato .csv).

La empresa *Peajes de Sudamérica* solicita ahora la versión final (o 4.0) del Sistema de Control de Peajes 2023, con la intención de ponerla en producción antes del cierre del año.

En esta versión 4.0 volveremos a tener un archivo de texto con cierto volumen de datos ya grabados allí, pero ese archivo ahora tendrá formato ***comma separated values (.csv)***: cada línea del archivo estará compuesta por los distintos valores de un ticket separados por una coma, en lugar de venir todos agrupados en una gran cadena de formato fijo.

A diferencia de la versión anterior, en la versión 4.0 los datos deberán ser leídos desde ese archivo de texto, y luego deberán ser almacenados en un un archivo binario de registros, sin generar en un vector de registros/objetos en ese momento (pero puede pedirse que generen un vector como parte de otros procesos). Toda la funcionalidad requerida debe ser programada directamente sobre el contenido de ese archivo binario (a menos que se especifique claramente otra cosa).

El programa debe ser gestionado a través de un menú de opciones. La corrección/calificación de este trabajo será realizada en forma manual por los profesores, así que no hay ahora restricciones en cuanto al formato y estilo de las salidas, ni hay tampoco exigencia en cuanto a cargar datos en cierto orden riguroso. Pero habrá un archivo de texto (con formato *.csv*) de entrada que deberán procesar (click aquí para descargar el archivo), y cuyos resultados serán conocidos de antemano para poder validar.

Cada registro del archivo binario a crear debe mantener los datos de una operación de cobro de ticket. Por cada ticket se asumen los mismos datos que para el TP3, pero quede claro que cada equipo de programadores puede decidir agregar nuevos atributos si los creen necesarios (lo que NO deben hacer, es **eliminar** algún atributo de los que son exigibles según la enumeración que sigue). Cada registro/objeto de tipo **Ticket** deberá contener (al menos) los datos siguientes:

- Código identificador del ticket (un número entero), que puede ser de cualquier longitud (y NO necesariamente de 10 dígitos de largo... Lo que NO debería pasar, es que este código sea cero ni negativo).
- Patente del vehículo al que se le cobra el ticket (una cadena) (que NO necesariamente es de 6 o 7 caracteres... Puede ser de cualquier longitud, y obviamente si no pertenece a ninguno de los países Mercosur ni Chile, pues será "Otro").
- Tipo de vehículo (un entero entre 0 y 2 - (0: motocicleta, 1: automóvil, 2: camión)).
- Forma de pago (un dígito 1 o 2 que indica la forma de pago (1: manual, 2 telepeaje)).
- País de la cabina (un dígito entre 0 y 4 que indica el país donde está la cabina que hizo el cobro (0: Argentina - 1: Bolivia - 2: Brasil - 3: Paraguay - 4: Uruguay)).
- Kilómetros recorridos desde la cabina anterior (un entero que puede ser un cero para indicar que la cabina actual es la primera que ese vehículo atraviesa).

Otra vez, el programa deberá procesar e identificar datos de vehículos cuyas patentes pueden ser de cualquiera de los países del Mercosur, más Chile, de acuerdo al mismo modelo de las versiones anteriores. Y pueden venir patentes de otros países (en cuyo caso deberán identificarse como "Otro" cuando sea requerido).

La forma de calcular los importes finales a cobrar por cada ticket, están especificadas en los enunciados/requerimientos de las versiones anteriores (y por supuesto deben volver a aplicarse aquí en la versión 4.0).

La versión 4.0 del programa deberá basarse también en un menú de opciones. Pero a diferencia del TP3, la opción 1 de esta nueva versión debe tomar los datos de todos los tickets desde en un *archivo de texto peajes-tp4.csv* que será provisto para su procesamiento con formato *.csv* como se explica a continuación.

El archivo de texto con los datos de entrada tendrá la primera línea de *timestamp* igual que en el TP2 y el TP3, y las líneas a partir de la segunda tendrán los datos de los tickets separados por una coma (terminando cada línea con un salto de línea). Estrictamente, la segunda línea tendrá los nombres o descriptores de los valores (y es una línea informativa que deberá ser ignorada), y a partir de la tercera vendrán los datos. El siguiente es un modelo de la forma que puede tener el archivo de entrada:

```
24 de mayo de 2023 - 13 hs 40 min - ES
codigo,patente,tipo_vehiculo,forma_pago,pais_cabina,distancia
231258082,PYV0922,0,2,0,541
37051001,XQY5183,0,2,2,145
613415725,PUF2144,2,1,2,297
337987378,JBD0091,1,1,1,543
361493731,FZ073NF,2,2,1,109
286072878,ZIVYW7,0,1,0,291
246674513,WGCX772,1,1,4,80
95827909,JMN7U20,1,1,4,0
```

Note que este formato elimina el problema de tener que compensar con ceros a la izquierda los valores numéricos, o con blancos (a la derecha o a la izquierda) las cadenas. La distancia entre cabinas (por ejemplo) no necesita expresarse con tres ceros si esa distancia era cero: un simple cero después de la última coma de una línea es suficiente (vea la última línea del modelo anterior). Además, con este formato el contenido del archivo está autodefinido (los nombres de los campos de la segunda línea expresan con cierta claridad qué es lo que contienen las líneas que siguen).

Se pide que esta versión del programa mantenga ***la clase* Ticket**, con los atributos indicados aquí (más los que el equipo de trabajo determine que pueda necesitar), y a través de las opciones del menú **genere un archivo binario de registros**, de forma que cada registro de ese archivo contenga los datos de un ticket. El programa debe incluir al menos los mismos dos módulos separados que en la versión anterior:

- En uno de esos módulos debe definirse la clase Ticket, incluyendo los métodos o funciones separadas que el grupo determine que son aplicables.
- En el otro módulo, debe desarrollarse el programa principal, con un menú de opciones para realizar cada una de las tareas solicitadas más abajo.

Los procesos que deben estar disponibles en el menú principal son los siguientes (note que en muchos casos se trata de adaptaciones de los que ya se pidieron para el TP3):

1. Crear el **archivo binario** de registros de forma que contenga todos los datos de todos los tickets guardados en el archivo de texto *peajes-tp4.csv* que se provee junto con este enunciado. Cada vez que se elija esta opción, el archivo binario debe ser creado de nuevo desde cero, perdiendo todos los registros que ya hubiese contenido. Asegúrese de que antes de eliminar el viejo archivo, se muestre en pantalla un mensaje de advertencia al usuario de forma que tenga la opción de cancelar la operación. Repetimos: **NO DEBE CREAR UN ARREGLO DE REGISTROS/OBJETOS**, sino directamente pasar del archivo de texto al archivo binario. Y salvo en el punto 7 de este listado de procesos, EN NINGÚN OTRO PUNTO DEBE CREAR TAL ARREGLO, sino trabajar directamente con los datos contenidos en el archivo binario.

2. Cargar por teclado los datos de un ticket, aplicando procesos de validación para cada campo, y agregar un registro con esos datos directamente al final del **archivo binario**. Cada vez que se elija esta opción, **el nuevo registro debe agregarse al final del archivo binario, sin perder ninguno de los registros que el archivo ya contenía**. Si el archivo no existiese, debe ser creado y luego agregar el registro cargado.

3. Mostrar **todos los datos de todos los registros del archivo binario**, tal como están grabados (sin ningún proceso de ordenamiento previo). Cada registro debe ocupar una sola línea en pantalla, **y debe mostrarse también el nombre del país al que pertenece cada patente**.

4. Mostrar todos los registros del **archivo binario** cuya patente sea igual a **p**, siendo **p** un valor que se carga por teclado. Al final del listado mostrar una línea adicional indicando cuántos registros se mostraron.

5. Buscar si existe en el **archivo binario** un registro cuyo código de ticket sea igual a **c**, siendo **c** un valor que se carga por teclado. Si existe mostrar el registro completo. Si no existe indicar con un mensaje. La búsqueda **debe** detenerse al encontrar el primer registro que coincida con el criterio pedido.

6. Determinar y mostrar la cantidad de vehículos de cada combinación posible entre tipo de vehículo y país de cabina en el **archivo binario**. Como son tres tipos de vehículos posibles (0, 1 y 2) y son cinco los países de cabinas posibles (entre 0 y 4), entonces se trata de 3 * 5 = 15 contadores, que obviamente deben ser gestionados en una matriz de conteo. Muestre solo los contadores cuyo valor final sea diferente de cero. Observación: ni siquiera se les ocurra plantear un esquema de 15 condiciones y 15 contadores separados... Esto se resuelve con una matriz de conteo o nada.

7. En base a la matriz que se pidió generar en el ítem anterior, muestre la cantidad total de vehículos contados por cada tipo de vehículo posible, y la cantidad total de vehículos contados por cada país de cabina posible. Es decir, se pide por un lado, totalizar las filas de esa matriz, y por otro, totalizar las columnas.

8. Calcular y mostrar la distancia promedio desde la última cabina recorrida entre todos los vehículos del **archivo binario**. Y ahora sí, generar en memoria un **arreglo de registros/objetos** con todos los tickets del archivo binario cuya distancia recorrida sea mayor al valor promedio que acaba de calcular. **Muestre el arreglo**, pero ordenado de menor a mayor de acuerdo a la distancia recorrida. En este punto, los programadores deben considerar que la cantidad de datos en el vector podría ser realmente un número grande o muy grande, y por lo tanto, no deberían aplicar un método de ordenamiento simple. Tienen al menos el **Shellsort** explicado en clases. El archivo de entrada en formato .csv tiene 100000 (cien mil) líneas de datos, por lo que muy posiblemente el tamaño del "arreglito" que les estamos pidiendo ronde los 50000 (cincuenta mil) objetos... Deduzcan lo que deben hacer...

(Click aquí para descargar el archivo de entrada en formato .csv).

b. Reglas de Entrega y Formación de Grupos.

1. El trabajo **debe** ser realizado en grupos de entre **tres y cinco personas**, y debe ser subido (o sea, entregado...) en la fecha indicada para ello en el aula virtual. Es suficiente con que UN alumno del grupo suba el trabajo, y con eso quedará entregado para todos los miembros del grupo. El práctico debe entregarse en un archivo comprimido (zip, rar o el tipo que dispongan los alumnos) cuyo nombre respete el siguiente formato: **2023_AED_TP4_Grupo_nnn_Apellido_Legajo[Curso]_Apellido_Legajo[Curso].zip** (en donde se debe reemplazar *Grupo_nnn por el número de grupo que entrega el trabajo, Apellido* por el apellido de cada integrante*, Legajo* por el número de legajo de cada uno, hasta completar al grupo, además de indicar a qué curso pertenece cada integrante. Obviamente, la extensión zip puede ser rar u otra que haya sido generada por un programa de compresión). Los grupos podrán estar formados por alumnos del mismo curso y/o de cursos diferentes de la Cátedra. Es muy importante, para facilitar el trabajo de corrección, que los alumnos indiquen claramente el curso al cual pertenecen en el nombre del archivo que suben.
2. Sólo podrán subir un trabajo los alumnos que formen parte de un grupo oficialmente formado y notificado por la Cátedra. Ningún estudiante podrá subir un trabajo sin pertenecer previamente a un grupo (no dispondrá de botones o links para subir su trabajo en el aula virtual si no pertenece a un grupo). Están avisados.
3. La nota que obtengan en este trabajo, será promediada en forma ponderada con el resto de las notas que se obtengan en la parte práctica de la materia, según fórmula indicada en el documento *Reglas del Juego* que figura en la Zona Cero del aula virtual del curso, para formar así la nota de trabajos prácticos que empezará a definir la regularidad.

c. Posibles Sanciones.

1. Si en los trabajos prácticos se detectan resoluciones duplicadas (dos o más grupos con trabajos duplicados o demasiado similares) se asumirá que fueron copiadas y TODOS los alumnos de todos los grupos que estén involucrados serán aplazados.
2. Como consecuencia, esos alumnos *perderán la posibilidad de promocionar o aprobar en forma directa la materia* y la condición de regularidad quedará sujeta a la decisión del docente adjunto y/o al coordinador de la cátedra.
3. Por lo tanto CADA GRUPO ES RESPONSABLE DE SU TRABAJO. No habrá disculpa incluso si lo comparten como ayuda a otros compañeros y estos traicionan su confianza copiándolo.

[1] El enunciado y planteo general de este Trabajo Práctico 4 fue desarrollado por el ingeniero *Valerio Frittelli*, y fue revisado por las ingenieras *Marcela Tartabini* y *Karina Ligorria*, y por el ing. *Jorge Harach*.

[β]

Agregar entrega

Estado de la entrega

Grupo	TP4-G054
Estado de la entrega	No se ha enviado nada en esta tarea
Estado de la calificación	No calificada
Tiempo restante	19 días 1 hora restante

◄ Cuestionario 25 [Temas: hasta Ficha 25]

Ir a...

Guía de Ejercicios Prácticos 25 ►