TRABAJO PRÁCTICO INTEGRADOR – Análisis de extracto de tarjetas de crédito

1) Objetivo

El objetivo de este T.P. consiste en desarrollar un aplicativo de consola con comandos en línea de órdenes y escrito en lenguaje ANSI C, que permita realizar un análisis de transacciones de tarjeta de crédito de un extracto bancario.

2) Alcance

Mediante el presente T.P. se busca que el alumno adquiera y aplique conocimientos sobre los siguientes temas:

- programas en modo consola;
- argumentos en línea de órdenes (*CLA*);
- modularización;
- *makefile;*
- archivos de texto y binarios;
- memoria dinámica;
- punteros a función;
- tipo de dato abstracto (T.D.A. Vector, T.D.A. *ad hoc*);
- estructura básica de un archivo CSV;
- manejo de fechas;

3) Desarrollo

En este T.P. se pide escribir un programa ejecutable que procese a partir de un extracto bancario las transacciones de débito y crédito de dinero de los distintos usuarios en un rango de tiempo determinado.

El programa ejecutable, denominado "`analisis_extracto.exe`" (WinXX) o "`analisis_extracto`" (Unix), debe ser invocado de la siguiente forma:

WinXX:
```
analisis_extracto –fmt <formato> -out <salida> -in <salida> -ti
<tiempo_inicial> -tf <tiempo_final>
```

UNIX:
```
./analisis_extracto –fmt <formato> -out <salida> -in <salida> -ti
<tiempo_inicial> -tf <tiempo_final>
```

Los comandos en línea de órdenes utilizados por la aplicación son los indicados a continuación.

Formato del índice a generar

Comando	Descripción	Valor	Tipo de dato
-fmt	Formato del índice a generar	"csv"	Cadena de caracteres
		"xml"	Cadena de caracteres

Archivo de salida <salida>

Es la ruta del archivo con la información analizada.

Archivos de entrada <entrada>

Es la ruta del archivo de texto que contiene la información a analizar.

Tiempo inicial <tiempo_inicial>

Tiempo inicial a partir del que se debe realiza el análisis en segundos como tiempo Unix .

Tiempo inicial <tiempo_final>

Tiempo final hasta el que se debe realizar el análisis en segundos como tiempo Unix .

Nota: Se puede asumir que los comandos en línea de órdenes estarán en cualquier orden (en pares), si esta estrategia simplifica el desarrollo de la presente aplicación, pero se debe consignar la decisión en el informe.

Formato de entrada

El archivo del extracto bancario a procesar tiene un formato CSV con las siguientes columnas:

ID_TRANSACCION, ID_USUARIO, FECHA, MONTO, NÚMERO DE TARJETA, DESCRIPCION
123412, 1 , 05/11/2011 10:00:00 , -10 , 4916288217067475, *Compra supermercado*
123413 , 2 , 05/11/2011 10:00:01, -100, 5031755734530604, Compra supermercado
123414 , 1 , 05/11/2011 10:00:02 , 200, 4916288217067475, *Pago de tarjeta*
123415 , 3 , 05/11/2011 10:00:03, -100, 4509953566233704, Extracción cajero

Nombre	Ejemplo	Descripción
Id Transacción	12345	Identificación de la transacción [unsigned int]
Id Usuario	2153	Identificación del usuario [unsigned int]
Fecha	*05/11/2011 10:00:00*	Fecha en la que ocurre la transacción en formato dd/mm/yyyy hh:mm:ss
	05-11-2011 10:00:00	Fecha en la que ocurre la transacción en formato dd-mm-yyyy hh:mm:ss
	05.11.2011 10:00:00	Fecha en la que ocurre la transacción en formato dd.mm.yyyy hh:mm:ss
Monto	100	Monto de la transacción [int]
Número de tarjeta de crédito	4916288217067475	Número de la tarjeta de crédito [char[]]
Descripción	Transferencia	Descripción de la operación [char[]]

Notas:
El identificador del usuario tiene un valor máximo desconocido pero se sabe que es suficientemente chico como para que el programa pueda ser procesado por un ordenador hogareño, por lo tanto el manejo del arreglo donde se compacten los datos deberá utilizar memoria dinámica.
Es importante observar que el formato de entrada puede tener las fechas expresadas en distintos formatos en el mismo archivo y por lo tanto el mismo deberá poder manejar los distintos tipos de fechas.

Operación

El programa debe analizar el archivo de entrada descartando aquellas líneas que se encuentran fuera del rango de tiempo dado. El objetivo es poder acumular los débitos y créditos de cada usuario. Los créditos estarán dados por los montos positivos y los débitos por los montos negativos.

Formatos de salida

La salida del programa debe generar un archivo con los montos transaccionados por cada usuario, separando créditos de débitos, ordenados de mayor a menor por la cantidad de créditos.

a) Formato CSV

El formato del documento CSV de salida a generar es:

```
<Id Usuario>,<Créditos>,<Débitos>
…
<Id Usuario>,<Créditos>,<Débitos>
```

Ejemplo de archivo de salida

```
ID_USUARIO, CREDITOS, DEBITOS
3           ,   100,       0
2           ,   100,       0
1           ,    20,     200
```

b) Formato XML

Ejemplo equivalente de salida en XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <row>
    <user>3</user>
    <credits>0</credits>
    <debits>100</debits>
  </row>
  <row>
    <user>2</user>
    <credits>0</credits>
    <debits>100</debits>
  </row>
  <row>
    <user>3</user>
    <credits>200</credits>
    <debits>10</debits>
  </row>
</root>
```

Opcional

Cómo punto opcional deberán validarse los números de tarjeta de crédito asociados a cada transacción. Para la validación de los números de tarjeta de crédito se deberá utilizar el algoritmo de Luhn. Este algoritmo detecta cualquier error de un único dígito, así como casi todos las transposiciones

de dígitos adyacentes. Fue desarrollado por Hans Peter Luhn en 1954 trabajando para IBM, dando lugar a la patente U.S. Patent No. 2,950,048. En la actualidad su especificación se encuentra en la norma ISO/IEC 7812-1.

El algoritmo tiene tres pasos:

1) De izquierda a derecha se deben duplicar las posiciones impares. Si el resultado de cada número duplicado es mayor a 10, se deben sumar los dígitos.
2) Sumar todos los números
3) Si el resultado es múltiplo de 10 entonces el número es válido

Aquellas transacciones inválidas deberán ser informadas por el flujo de estándar errores y excluidas del cálculo final de débitos y créditos. Sin embargo, no impedirán el procesamiento completo del archivo.

4) Restricciones

La realización de este programa está sujeta a las siguientes restricciones:

- Se debe recurrir al uso de punteros a función a fin de parametrizar la impresión de los archivos de salida.
- Se deben utilizar funciones y una adecuada modularización.
- Se debe construir un proyecto mediante la utilización de *makefile*.
- Hay otras cuestiones que no han sido especificadas intencionalmente en este requerimiento, para darle al desarrollador la libertad de elegir implementaciones que, según su criterio, resulten más favorables en determinadas situaciones. Por lo tanto, se debe explicitar cada una de las decisiones adoptadas y el o los fundamentos considerados para ellas.

5) Entrega del Trabajo Práctico

Se debe presentar la correspondiente documentación de desarrollo. Ésta se debe entregar en forma impresa y encarpetada, de acuerdo con la numeración siguiente:

1) Carátula. Incluir una dirección de correo electrónico. Nombre y Padrón de los integrantes.
2) Enunciado (el presente documento).
3) Diagrama de flujo básico y simplificado del programa (1 carilla A4).
4) Estructura funcional del programa desarrollado (diagrama de funciones, 1 carilla A4).
5) Explicación de las alternativas consideradas y las estrategias adoptadas.
6) Resultados de la ejecución (corridas) del programa, en condiciones normales e inesperadas de entrada.
7) Reseña de los problemas encontrados en el desarrollo del programa y las soluciones implementadas para subsanarlos.
8) Conclusiones.
9) *Script* de compilación.
10) Código fuente.

6) Bibliografía

Se debe incluir la referencia a toda bibliografía consultada para la realización del trabajo: libros, artículos, etc.

7) Fecha de entrega:

22/07 Primer entrega formal
29/07 Reentrega final