TRABAJO PRÁCTICO N.º 2 – Validación de operaciones con 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 validar un archivo con transacciones con tarjetas de crédito y presentar un informe en diversos formatos.

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;
- métodos de ordenamiento y búsqueda;
- tipo de dato abstracto (T.D.A. Vector/Lista, T.D.A. *ad hoc*);
- estructura básica de un archivo CSV;
- estructura básica de un documento XML;
- algoritmo de Luhn para validación de números identificatorios;

3) Desarrollo

En este T.P. se pide escribir un programa ejecutable que procese y valide un archivo con operaciones con tarjeta de crédito de una cadena de supermercados utilizando el algoritmo de Luhn y genere archivos de salida en formato CSV y XML con las operaciones inválidas detectadas en cada sucursal del supermercado.

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

WinXX:
```
luhn_analyzer –fmt <formato> -out <salida> -in <entrada>
```

UNIX:
```
./luhn_analyzer –fmt <formato> -out <salida> -in <entrada>
```

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 los resultados del análisis que se desea generar.

Archivos de entrada <entrada>

Es la ruta del archivo de texto que contiene las operaciones con tarjeta de crédito en formato CSV.

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.

Operación

El programa debe analizar el archivo que contienen las transacciones con tarjeta de crédito, detectar las operaciones con números de tarjetas inválidos en cada sucursal y generar los archivos en los formatos de salida correspondientes.

Es importante aclarar que el programa debe considerar que la cadena de supermercados tiene una cantidad de sucursales variable y que puede crecer a lo largo del tiempo, con lo cual será necesario utilizar memoria dinámica para poder resolver el problema.

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 todas 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

Ejemplo con un número de tarjeta de crédito válido

4	9	1	6	2	8	8	2	1	7	0	6	7	4	7	5
8		2		4		**16**		2		0		**14**		**14**	
8	9	2	6	4	8	7	2	2	7	0	6	5	4	5	5

Suma 80

Si por ejemplo se invierten los números del final vemos que el número no es válido

4	9	1	6	2	8	8	2	1	7	0	6	7	4	5	7
8		2		4		**16**		2		0		**14**		**10**	
8	9	2	6	4	8	7	2	2	7	0	6	5	4	1	7

Suma 78

Formato de entrada

El archivo de entrada se encuentra en formato CSV y contiene la información tabulada de la siguiente manera:

Transacción, Número de tarjeta de crédito, Sucursal, Monto, Descripción

213412, 4916288217067475, 123412, 250.60, Shampo
184535, 4916288217067475, 532123, 100.50, Manaos 1Lt
123956, 4916288217067475, 123412, 50.12, Leche
783619, 4916288217067475, 543256, 70.54, Yogurt
938179, 4916288217067475, 123412, 200.18, Cerveza
891234, 4916288217067475, 123555, 80.40, Doritos

Nombre	Ejemplo	Descripción
Transacción	213412	Identificador de la transacción [unsigned int]
Número de tarjeta de crédito	4916288217067475	Número de la tarjeta de crédito [char[]]
Sucursal	123412	Identificador de la sucursal [unsigned int]
Monto	234.23	Precio del item [float]
Descripción	Shampoo	Descripción del ítem comprado [char[]]

Formato de salida

a) Formato CSV

El formato del documento CSV de salida a generar es:

```
<sucursal>|<transacción>|<monto>
…
<sucursal>|<transacción>|<monto>
```

Se trata de un archivo CSV sin encabezado, campos sin calificador y carácter separador “|”.

b) Formato XML

El formato del documento XML de salida a generar es:

```
<?xml version="1.0" ?>
<transactions>
    <transaction>
      <store_id>123412</store_id>
      <transaction_id>213412</transaction_id>
      <amount>234.23</amount>
    </transaction>
    <transaction>
      <store_id>123413</store_id>
      <transaction_id>213413</transaction_id>
      <amount>123.23</amount>
    </transaction>
    ...
</transactions>
```

Ambos formatos de salida deberán estar ordenados por identificador de sucursal creciente y monto de las transacciones decreciente de forma tal de poder identificar rápidamente las transacciones inválidas más importantes de cada sucursal.

4) Restricciones

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

− Se debe recurrir al uso de T.D.A. vector y/o lista para lograr almacenar las transacciones inválidas de cada sucursal.

− Se debe recurrir al uso de T.D.A. para almacenar los elementos de información correspondientes al mensaje de cada transacción.

− 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.
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:**