**TECNOLÓGICO DE COSTA RICA II SEMESTRE 2020**

**INGENIERÍA EN COMPUTACIÓN**

**IC-4700 LENGUAJES DE PROGRAMACIÓN**

**TAREA PROGRAMADA 3**

Este proyecto consiste en realizar una aplicación en Eiffel que permita hacer lo siguiente:

* convertir archivos CSV en estructuras en memoria que siguen la representación JSON
* realizar operaciones con esas estructuras JSON para generar nuevas estructuras JSON
* almacenar la información de estructuras JSON en archivos de texto; tanto en formato JSON con en formato CSV.

**Archivos CSV**

Los archivos CSV (comma separated values) son archivos de texto muy sencillos que se usan para representar datos tabulares, en los que las columnas se separan por medio de algún caracter especial (coma, punto y coma, salto de tabulación, etc.). El separador se suele escoger para evitar conflictos con los caracteres que vienen en los campos; por ejemplo si se usa punto como separador de decimales en números, entonces el separador puede ser coma; por otro lado, si los número usan comas decimales, se usa punto y coma como separador. Algunas veces se usan comillas para delimitar valores que contienen caracteres separadores.

Los archivos CSV que la aplicación debe manejar tienen las siguientes características:

* los caracteres vienen codificados en ANSI
* el caracter separador es el punto y coma ‘;’
* se usará coma en los números: 3,14
* las líneas vendrán separadas por los caracteres CR/LF
* la primera línea del archivo contendrá los nombres de las columnas; todos los nombres deben aparecer y ser distintos entre sí
* la segunda línea del archivo especifica el tipo de dato de cada columna; una “X” indica que la columna correspondiente almacena valores de tiras de caracteres; un “9” indica que la columna correspondiente almacena valores numéricos; una “B” indica que la columna correspondiente almacena valores booleanos, donde ‘T’ y ‘S’ representan true, ‘F’ y ‘N’ representan false.
* exceptuando las dos primeras filas, es posible omitir valores; en ese caso se pone una tira nula en el lugar en que debería estar el dato
* ejemplo:

**equipo;pais;alojamiento;mejor marca**

**X;X;X;N**

**A;CRI;Edificio A1;3,08**

**B;COL;Edificio A2;2,40**

**C;CUB;Edificio A2;2,20**

**D;DOM;;3,15**

**E;ECU;Edificio A1;2,75**

**JSON**

JSON (JavaScript Object Notation) es una representación pequeña y simple que permite almacenar y transmitir datos estructurados. Se usa con frecuencia para transferir datos de un servicio web a una aplicación JavaScript que ejecuta en un navegador web.

JSON usa cuatro tipos de datos primitivos: tiras de caracteres, números, booleanos y el valor especial llamado **null**. A continuación se muestran unos posibles valores para esos tipos de datos:  
 "una tira" 12345 true null.

JSON provee dos tipos de datos compuestos:  
 arreglos: secuencia ordenada de ***valores***  
 objetos: colección no ordenada de pares (***nombre***, ***valor***).  
Los ***nombres*** son siempre tiras de caracteres; los ***valores*** en arreglos u objetos pueden ser de cualquiera de los tipos JSON. A continuación se muestran un par de posibles valores compuestos de JSON:  
 **[ -3.14, true, null, "una tira" ]  
 { "numeros" : [1,2,3,4,5], "util" : false }**

Eiffel dispone de una biblioteca **JSON** que define clases para manejar los diferentes valores de JSON: **JSON\_ARRAY**, **JSON\_BOOLEAN**, J**SON\_NULL**, **JSON\_NUMBER**, **JSON\_OBJECT**, **JSON\_STRING**, **JSON\_VALUE**. Dichas clases permiten construir objetos JSON a partir de valores de entrada, así como serializarlos en una tira de caracteres y realizar otras conversiones.

**Aplicación**

La aplicación consiste de un ciclo interactivo que lee los comandos que a continuación se describen, y producen el resultado especificado. Algunos comandos almacenan los resultados internamente en memoria, mientras que otros producen archivos de salida. Los resultados que se almacenan internamente se identifican por medio de un nombre, el cual consiste de una tira de caracteres alfanuméricos cuyo primer caracter debe ser una letra.

**load *Nombre* *RutaEntrada1.csv***

Carga el archivo csv indicado por ***RutaEntrada1.csv*** en una estructura JSON que es identificada con ***Nombre*** para usos posteriores.  
Si ya existe una estructura con ese ***Nombre***, se rechaza el comando.

**save *Nombre* *RutaSalida.json***

Guarda la estructura JSON identificada con ***Nombre*** en el archivo ***RutaSalida.json***.  
La salida sigue el formato JSON.

**savecsv *Nombre* *RutaSalida.csv***

Guarda la estructura JSON identificada con ***Nombre*** en el archivo ***RutaSalida.csv***.  
La salida sigue el formato CSV; debe cumplir con las especificaciones de entrada: separador punto y coma, segunda línea con los tipos X, 9 y B, etc.

**select *Nombre1 Nombre2 Atributo* = *Valor***

Selecciona de la estructura JSON identificada con ***Nombre1*** aquellas filas (objetos) que cumplan con que ***Atributo* = *Valor***. Esto crea una estructura JSON nueva, ***Nombre2***. Se considera que ***Valor*** empieza un caracter después del **=** y continúa hasta el fin de línea.

**project *Nombre1 Nombre2 Atributo1 ... Atributo5***

Extrae de la estructura JSON identificada con ***Nombre1*** las columnas ***Atributo1 ... Atributo5***. Esto crea una estructura JSON nueva, ***Nombre2***. Se pueden especificar menos de 5 atributos, pero debe haber por lo menos uno.

**exit**

Sale del ciclo iterativo y terminar la ejecución.

**Formato de salida**

Para efectos de legibilidad, cuando se pide que se escriba una estructura JSON a un archivo, se debe tener una línea de salida por cada elemento del arreglo que contiene la estructura JSON. Por ejemplo, una salida sería:

[

{"A":10, "B": "hola mundo", "C":true, "D":1.2, "E": "e1"},

{"A":20, "B": "adios muchachos", "C":false, "D":4.3, "E": "e2"},

{"A":30, "B": "arena y sol", "C":false, "D":0.7, "E": "e1"},

{"A":40, "B": "asi es la vida", "C":true, "D"2.1, "E": "e2"}

]

**Consideraciones finales**

Para efectos de simplificar los comandos, se supondrá que los nombres de las columnas (atributos) son alfanuméricos, sin espacios en blanco u otros caracteres; además su primer caracter debe ser una letra.

El programa desarrollado debe seguir los principios del paradigma de la programación orientada a objetos: abstracción, encapsulamiento, modularidad, jerarquía, paso de mensajes, polimorfismo.

La tarea es individual y la fecha de entrega es lunes 24 de enero de 2021 al mediodía.

**Ejemplos de operaciones**

Se tiene el siguiente archivo, **Equipos.csv**:

**equipo;pais;alojamiento;marca**

**X;X;X;N**

**A;CRI;Edificio A1;3.08**

**B;COL;Edificio A2;2.40**

**C;CUB;Edificio A2;2.20**

**D;DOM;Edificio A1;3.15**

**E;ECU;Edificio A1;2.75**

Se tiene el siguiente archivo, **Integrantes.csv**:

**equipo;nombre;apellidos;fechaNac;primera**

**X;X;X;X;B**

**A;Berta;Bermúdez;02/02/1997;S**

**D;Jimena;Jiménez;08/08/1996;S**

**B;Dora;Domec;04/04/1995;N**

**C;Emilia;Elcano;05/05/1999;S**

**B;Cora;Carmona;03/03/1996;S**

**E;Rita;Mora;12/12/1998;N**

**D;Gabriela;Gomez;07/07/1997;S**

**A;Ana;Abarca;01/01/1998;N**

**C;Franci;Franco;06/06/1998;S**

**E;Marta;Murillo;10/10/1999;N**

**>> load eq Equipos.csv**

*Produce una estructura JSON en memoria, llamada* ***eq****, con el contenido del archivo* ***Equipos.csv****.*

**>> save eq eq.json**

*Guarda en el archivo de texto* ***eq.json*** *el contenido de la estructura* ***eq*** *siguiendo el formato JSON:*

[

{"equipo":"A", "pais":"CRI", "alojamiento":"Edificio A1", "marca":3.08},

{"equipo":"B", "pais":"COL", "alojamiento":"Edificio A2", "marca":2.40},

{"equipo":"C", "pais":"CUB", "alojamiento":"Edificio A2", "marca":2.20},

{"equipo":"D", "pais":"DOM", "alojamiento":"Edificio A1", "marca":3.15},

{"equipo":"E", "pais":"ECU", "alojamiento":"Edificio A1", "marca":2.75}

]

**>> select eq eq2 alojamiento = Edificio A1**

*Extrae de una estructura JSON, llamada* ***eq****,otra estructura, llamada* ***eq2****,con aquellas “filas” que cumplen con* ***alojamiento=”Edificio A1”****:*

**>> savecsv eq2 eq2.csv**

*Guarda en el archivo de texto* ***eq2.csv*** *el contenido de la estructura* ***eq2*** *siguiendo el formato CSV:*

equipo;pais;alojamiento;marca

X;X;X;N

A;CRI;Edificio A1;3.08

D;DOM;Edificio A1;3.15

E;ECU;Edificio A1;2.75

**>> project eq eq3 equipo pais**

*Extraer de la estructura JSON, llamada* ***eq,*** *otra estructura, llamada* ***eq3****, conteniendo solo los atributos especificados:*

**>> save eq3 eq3.json**

*Guarda en el archivo de texto* ***eq3.json*** *el contenido de la estructura* ***eq3*** *siguiendo el formato JSON:*

[

{"equipo":"A", "pais":"CRI"},

{"equipo":"B", "pais":"COL"},

{"equipo":"C", "pais":"CUB"},

{"equipo":"D", "pais":"DOM"},

{"equipo":"E", "pais":"ECU"}

]

**>> load int** **Integrantes.csv**

*Produce una estructura JSON en memoria, llamada* ***int****, con el contenido del archivo* ***Integrantes.csv****.*

**>> save int int.json**

*Guarda en el archivo de texto* ***int.json*** *el contenido de la estructura* ***int*** *siguiendo el formato JSON:*

[

{"equipo":"A", "nombre":"Berta", "apellidos":"Bermúdez", "fechaNac":"02/02/1997", "primera":True},

{"equipo":"D", "nombre":"Jimena", "apellidos":"Jiménez", "fechaNac":"08/08/1996", "primera":True},

{"equipo":"B", "nombre":"Dora", "apellidos":"Domec", "fechaNac":"04/04/1995", "primera":False},

{"equipo":"C", "nombre":"Emilia", "apellidos":"Elcano", "fechaNac":"05/05/1999", "primera":True},

{"equipo":"B", "nombre":"Cora", "apellidos":"Carmona", "fechaNac":"03/03/1996", "primera":True},

{"equipo":"E", "nombre":"Mirela", "apellidos":"Mora", "fechaNac":"12/12/1998", "primera":False},

{"equipo":"D", "nombre":"Gabriela", "apellidos":"Gomez", "fechaNac":"07/07/1997", "primera":True},

{"equipo":"A", "nombre":"Ana", "apellidos":"Abarca", "fechaNac":"01/01/1998", "primera":False},

{"equipo":"C", "nombre":"Franci", "apellidos":"Franco", "fechaNac":"06/06/1998", "primera":True},

{"equipo":"E", "nombre":"Marta", "apellidos":"Murillo", "fechaNac":"10/10/1999", "primera":False}

]

**>> select int int2 primera = True**

*Extrae de una estructura JSON, llamada* ***int****,otra estructura, llamada* ***int2****,con aquellas “filas” que cumplen con* ***primera = True****:*

**>> project int2 int3 nombre apellidos fechaNac**

*Extrae de una estructura JSON, llamada* ***int2,*** *otra estructura, llamada* ***int3****, conteniendo solo los atributos especificados:*

**>> save int3 int3.json**

*Guarda en el archivo de texto* ***int3.json*** *el contenido de la estructura* ***int3*** *siguiendo el formato JSON:*

[

{"nombre":"Berta", "apellido":"Bermúdez", "fechaNac":"02/02/1997"},

{"nombre":"Jimena", "apellido":"Jiménez", "fechaNac":"08/08/1996"},

{"nombre":"Emilia", "apellido":"Elcano", "fechaNac":"05/05/1999"},

{"nombre":"Cora", "apellido":"Carmona", "fechaNac":"03/03/1996"},

{"nombre":"Gabriela", "apellido":"Gomez", "fechaNac":"07/07/1997"},

{"nombre":"Franci", "apellido":"Franco", "fechaNac":"06/06/1998"}

]

**>> savecsv int3 int3.csv**

*Guarda en el archivo de texto* ***int3.csv*** *el contenido de la estructura* ***int3*** *siguiendo el formato CSV:*

nombre;apellidos;fechaNac

X;X;X

Berta;Bermúdez;02/02/1997

Jimena;Jiménez;08/08/1996

Emilia;Elcano;05/05/1999

Cora;Carmona;03/03/1996

Gabriela;Gomez;07/07/1997

Franci;Franco;06/06/1998