Trabajo Práctico Especial

1. Introducción

1.1. JSON

Javascript **O**bject **N**otation es un formato de intercambio de datos basado en texto (humanamente legible), similar a XML, pero de una estructura mucho más simple y flexible. JSON representa objetos que son agrupados mediante '{}' y contiene definiciones de claves-valor. Estos valores pueden ser cadenas, valores booleanos, valores númericos o arreglos de estos mismos. Por ejemplo:

```
{
"id" : 42,
"name" : "Bill Burke",
"married" : true,
"kids" : [ "Molly", "Abby" ]
}
```

Las claves-valores son separados por el símbolo de dos puntos ':' y son separados entre sí por comas ','. Los arreglos son agrupados entre corchetes '[]'.

1.2. JSON y Javascript

JSON surge originalmente como una manera de estructurar información dentro del lenguaje donde se originó Javascript, y fue creado por Douglas Crockford. Actualmente está reglado por dos especificaciones, el RFC 7159 (más general) y ECMA 404 (la gramática).

El código para crear objetos en Javascript es exactamente igual al producido para intercambiar datos en JSON. Esta posibilidad, que si bien ha sido explotada enormemente por bugs de seguridad, es lo que ha permitido que se transforme en un estándar de facto para el intercambio de datos asincrónicos entres servicios web.

1.3. Gramática

La gramática de JSON, según está especificada en ECMA-404:

```
object
    {}
    { members }
members
    pair
    pair , members
pair
    string : value
array
    []
    [ elements ]
elements
    value
    value , elements
value
    string
    number
    object
```

```
    array
    true
    false
    null


string
    ""
    " chars "
chars
    char
    char chars
char
    any-Unicode-character-
    except-"-or-\-or-
    control-character
    \"
    \\
    \/
    \b
    \f
    \n
    \r
    \t
    \u four-hex-digits
number
    int
    int frac
    int exp
    int frac exp
int
    digit
    digit1-9 digits
    - digit
    - digit1-9 digits
frac
    . digits
exp
    e digits
digits
    digit
    digit digits
e
    e
    e+
    e-
    E
    E+
    E-
```

2. Generador de Modelo de Datos para JSON

2.1. Objetivo

Desarrollar un compilador que en base a cualquier secuencia arbitraria de una palabra en JSON, verifique que gramaticalmente sea correcta, realice el análisis sintáctico y compile la entrada en una salida en código ejecutable (pueden ser los fuentes en un lenguaje que luego se compilen) con la estructura de datos que corresponde a la información contenida en el mensaje JSON.

Por ejemplo, si se recibe

```
[
   {
      "definition": "http://en.wiktionary.org/w/api.php",
      "status": "active",
      "translation": "http://glosbe.com/gapi/translate",
      "value": 150115,
      "word": "invigorant"
   },
   [
      {
         "definition": "http://en.wiktionary.org/w/api.php",
         "status": "active",
         "translation": "http://glosbe.com/gapi/translate",
         "value": 53,
         "word": "doing"
      }
   ]
]
```

Debería generar una estructura de objetos similar a:

```
Array array1 = new Array();

AnonymousObject first = array1.get(0);
first.status;    // "Active"
first.value;     // "150115"

Array2 array2 = array1.get(1);
AnonymousObject nested = array2.get(0);
nested.status;    // "Active"
nested.value;     // "53"
```

Tengan en cuenta que estas estructuras son generadas a partir del código JSON y la salida sería código en un lenguaje específico (Java, Ruby, Python, Perl, etc) o incluso en algún lenguaje de bajo nivel como bytecode Java o objeto.

Aclaraciones

- El modelo de objetos a generar en el lenguaje elegido, contienen los datos recibidos en la entrada, no su abstracción (no es el esquema en JSON, sino los datos particulares).
- Elegir un procedimiento adecuado para nombrar las estructuras anónimas en JSON.

Al final de la cursada vamos a correr los programas en los laboratorios de informática y aquel grupo que procese satisfactoriamente, y de manera más rápida, la salida del servicio publicado en **pampero** tendrá un 10.

Qué tienen disponible:

- Un Servicio RestLet en Java que provee datos de ejemplo: http://pampero.it.itba.edu.ar:8111/
- Pueden usar http://openweathermap.org/wiki/API/2.1/JSON_API
- Si quieren verificar si alguna estructura JSON en particular es válida: http://www.jsontest.com/
- Otro ejemplo: http://glosbe.com/gapi/translate?from=pol&dest=eng&format=json&phrase=witaj&pretty=true
- Otro más http://en.wiktionary.org/w/api.php?action=query&titles=core&prop=revisions &rvprop=content &rvgeneratexml=&format=json

Qué se espera del generador:

- Que pueda recibir cualquier entrada en JSON y generar el código fuente compilable que represente el modelo de datos en JSON.
- La entrada puede ser por línea de comandos, archivo de texto, estandard input.
- Que implemente un analizador sintáctico, cualquiera de los vistos en la cursada, que pueda parsear la entrada y generar el código de salida para modelar los datos.
- Entregar un informe bien escrito. Pueden hacerlo en inglés si lo desean (aconsejable). Es importante que detallen las decisiones que toman para solucionar el problema, las fuentes, referencias y librerías que usaron, los problemas que tuvieron.
- Permitir procesar entradas de gran tamaño.
- Hacerlo rapidamente (i.e. más rapidamente que el resto).
- $+\alpha$: Que pueda recibir una entrada en base a una URL, ejecutando un servicio HTTP (REST o no) y que obtenga la respuesta en JSON.

3. Sugerencias

- No es necesario que usen JSON schemas. Una de las principales ventajas de JSON es precisamente su dinamismo (que se limita con esquemas estáticos).
- Para acceder al servicio Restlet publicado sobre pampero, recuerden que pueden utilizar SSH tunneling.
- Como las entradas en JSON pueden ser muy grandes, especial cuidado con el manejo de memoria (i.e. memory-leaks, dangling pointers) y con el acceso a disco.

4. Material a entregar

Para aprobar el TPE, deben entregar:

- Códigos fuente, scripts de compilación y ejecución.
- Archivo ejecutable para Linux (Laboratorio).
- Un readme explicando cómo compilar y ejecutar el programa.
- Cinco(5) entradas de JSON que hayan obtenido de cualquier fuente.
- Cinco(5) programas que usen la salidas del procesamiento de esas entradas y demuestren la presencia de la estructura de datos correspondiente.
- Un informe que contenga, en este orden:
  - Carátula

- Índice
- Consideraciones realizadas (no previstas en el enunciado).
- Descripción del desarrollo del TP.
- Descripción del parser o Analizador sintáctico y el generador de código.
- Dificultades encontradas en el desarrollo del TP.
- Futuras extensiones, con una breve descripción de la complejidad de cada una.
- Referencias.

Importante: No hay ningún inconveniente en utilizar librerías públicas, soluciones similares públicas, soluciones de foros, etc., pero es necesario aclarar, y enumerar cada una de ellas en la sección Referencias. No se aceptan bloques de código públicos implementados verbatim sin ningún tipo de análisis. Tampoco implementaciones que resuelven problemas que no están detallados (e.g. implementa un garbage collector sin explicar cómo). Tampoco hay inconveniente en que interactuen con otros grupos pero tengan en cuenta que ellos son su competencia.

5. Grupos

El trabajo se realizará en grupos de no más de 3 integrantes.

6. Fecha de entrega

El material a entregar debe ser enviado por mail, en un archivo tipo .zip a la cuenta `rramele@itba.edu.ar`. Dentro de la carpeta .ZIP poner un directorio con los apellidos de los integrantes del grupo en camelCase: e.g. LopezGonzalezGoycochea.

No hay problema con que realicen la entrega desde bitbucket o github, pero tienen que enviar un email con la entrega formal, detallando la URL completa de un ZIP subido a esos repositorios listo para descargar.

El TP se debe entregar antes del día Domingo 22/6/2014 23:59 GMT -3. El Lunes 23/6/2014 se ejecutarán los programas con las instrucciones especificadas en cada entrega sobre los laboratorios.

7. Material de consulta

Se sugiere leer los documentos de las páginas:

1. "Compiladores, Principios, Técnicas y Herramientas",Aho, Sethi, Ullman, Addison Wesley.(El Libro del Dragón), capítulos 1,2 y 3.
2. ECMA-404 http://www.json.org/
3. OReilly.RESTful.Java.with.JAX-RS.2.0.2nd.Ed.Nov.2013