Trabajo práctico especial: Diseño e implementación de un lenguaje

Beade, Gonzalo (61223)
gbeade@itba.edu.ar

Di Toro, Camila (62576)
cditoro@itba.edu.ar

Castagnino, Salvador (60590)
scastagnino@itba.edu.ar

Autómatas, teoría de lenguajes y compiladores - Primer cuatrimestre 2022

Repositorio

https://github.com/camilaDiToro/Flex-Bison-Compiler

Idea

Diseñar un programa que permita generar código HTML a partir de código JSON.

Prestaciones

1. **Tags:** Se podrán crear tags HTML a partir de objetos JSON, el atributo *type* de cada objeto se corresponderá con el tipo de etiqueta HTML. Las etiquetas HTML no serán validadas, el usuario es responsable de ingresar etiquetas válidas. Esto parte de la base de que en HTML las etiquetas tampoco se validan. Además, podrían salir nuevas etiquetas en nuevas versiones de HTML y el compilador quedaría desactualizado.
2. **Etiquetas anidadas:**
   (a) Cada objeto JSON tendrá un atributo *content* que puede ser texto, otro objeto JSON o un arreglo que puede contener ambos.
   (b) El atributo *content* es optativo.
   (c) Un JSON dentro de la etiqueta *content* dará como resultado una etiqueta anidada en HTML.
   (d) Los arreglos se indicarán con corchetes y permitirán anidar mas de una etiqueta HTML, así también como agregar texto si es que hay etiquetas anidadas. En caso de haber más de una posición del arreglo que contenga texto, el mismo será concatenado.
   (e) La definición del lenguaje es recursiva. El caso base para el lenguaje son los scripts JSON con el tag "type" bien definido y sin una etiqueta "content" presente o con una etiqueta "content" asociada a un string.
   (f) Los atributos y valores reservados del lenguaje comenzaran con el simbolo @ para distinguirlos de los atributos y equiquetas utilizadas por html.
3. **Atributos:** Si dentro de un objeto JSON existe una etiqueta distinta de *content* o *type*, la misma será considerada un atributo HTML.
4. **Estructuras de control:** Se proveerán estructuras de tipo IF-THEN-ELSE y FOR. Las mismas serán representadas con atributos particulares dentro del objeto JSON, como se detalla en los ejemplos.
5. **Variables en las estructuras de control:** Se dispondrá de una variable que permitirá iterar a partir de lo indicado en las estructuras de control.

6. **Operaciones:** Se podrá realizar operaciones aritméticas básicas como +, -, * y /, sobre las variables de control definidas en el punto anterior. Solo tendrán validez semántica dentro de ${} en un string. Caso contrario, se considerarán texto.

7. **Lectura de entrada estándar:** Si se indica en el campo *type* que esa etiqueta será de tipo *read*, se podrá leer de entrada estándar.

8. **Manejo de los argumentos del programa:** Se dispondrá de una variable *args* que permitirá acceder a los argumentos del prorgama. Esta variable podrá ser manipulada como un vector.

Ejemplos

```
{
    "@type": "body",
    "style": "bg-red",
    "@content": [
    {
        "@type": "h1",
        "@content": "Hello world!"
    },
    {
        "@type": "@for",
        "@var": "x",
        "@in": [2, 9],
        "@content": {
                        "@type": "p",
                        "@content": "I am tag ${x}"
                    }
    },
    {
        "@type": "@read",
        "@var": "x"
        "@content":  {
            "@type": "h3",
            "@content": "${x}"
        }
    }]
}
```

```
<!DOCTYPE html>
<html>
  <body style="bg-red">
    <h1>Hello World!<\h1>
    <p>I am tag 2<\p>
    <p>I am tag 3<\p>
    <p>I am tag 4<\p>

    <p>I am tag 9<\p>
    <h3>Bye Gonzalo!<\h3>        <!-- Leido de la variable x -->
  </body>
</html>
```