Trabajo Práctico
Teoría de Lenguajes
"Generador de JSON de ejemplo"

Primer cuatrimestre 2019

1. Introducción

El lenguaje Go permite definir tipos de datos como estructuras usando tipos básicos, arreglos u otras estructuras.

En este trabajo práctico construiremos un programa que permitirá leer definiciones de un subconjunto simplificado de tipos en Go y generar textos JSON válidos con datos aleatorios que respeten esas definiciones.

2. Definición de la entrada

El siguiente ejemplo muestra el formato de tipos en Go que se deben aceptar:

```
type persona struct {
    nombre  string
    edad    int
    nacionalidad pais
    ventas  []float64
    activo  bool
}

type pais struct {
   nombre   string
   codigo   struct {
        prefijo string
        sufijo  string
   }
}
```

Las estructuras pueden anidarse sintácticamente, como en el caso de pais/codigo, o por nombre, como en el caso de persona/pais. Es un error si hay referencias circulares (por ejemplo si pais quisiera incluir a su vez a persona).

Todos los identificadores deben comenzar por una letra minúscula.

Los tipos basicos soportados son *string, int, float64 y bool*. Los arreglos se indican con [] delante del tipo.

La entrada debe constar de los todos structs necesarios para no tener dependencias externas; no hay directivas de paquete o funciones. No es un error si hay tipos definidos y no usados por el tipo principal (directa o indirectamente).

En caso de que la entrada no sea válida, el programa debe informar en lo posible la ubicación del error.

3. Salida

Dada una definicion de tipo, y tomando como tipo principal al primero que aparece en el archivo, se debe generar una cadena JSON que sea un ejemplo de la definición, usando datos aleatorios para los tipos básicos. La cantidad de elementos en los arreglos tambien debe ser aleatoria (entre 0 y 5). Para generar los strings deben usarse sólo caracteres entre a y z.

Se puede consultar la definición del formato JSON en `http://www.json.org/`

Para la entrada de la sección anterior un ejemplo de salida esperado sería:

```
 {
   "nombre":"fdskhs",
   "edad":23,
   "nacionalidad":{
      "nombre":"kjhkjdf",
      "codigo":{
         "prefijo":"gd",
         "sufijo":"asd"
      }
   },
   "ventas":[
      32.5,
      223.3,
      999.0
   ],
   "activo":false
}
```

4. Detalles de la entrega

Pueden implementar su solución en C, Java, C++, Go, C# o Python. Si quieren usar otro lenguaje, consúltenlo con el JTP.

Toda la entrada y salida debe manejarse por stdin y stdout. Opcionalmente, podrá hacerse *también* a través de archivos. El programa deberá tener interfaz de línea de comandos.

La entrega será unicamente por mail a la dirección tptleng@gmail.com.

La entrega debe incluir:

- Un programa que cumpla con lo solicitado
- El código fuente del programa.
- Informe conteniendo:
  - El código de la solución. Si se usaron herramientas generadoras de código, incluir la fuente ingresada a la herramienta, no el código generado.
  - Descripción de cómo se implementó la solución.
  - Información y requerimientos de software para ejecutar y recompilar el TP (versiones de compiladores, herramientas, plataforma, parámetros, etc).
  - Casos de prueba con expresiones válidas e inválidas, resultados obtenidos y conclusiones.