PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC 2323 — Construcción de Compiladores

Proyecto 1/4: Analizador Léxico

Fecha de Entrega: Lunes 28-Marzo de 2016, 23:59
Composición: individual

En esta etapa el objetivo es escribir un analizador léxico (*lexer*) para Agú, un subconjunto de lenguaje Ogú. La descripción léxica del lenguaje se les entrega en el documento "Descripción Léxica".

Etapas y Fechas de Entrega

Cada etapa les permitirá avanzar en el desarrollo de un compilador para el lenguaje Agú. Las fechas y porcentaje respecto a la nota del proyecto son:

15 %	Etapa 1: Análisis Léxico	Lunes 28-Marzo-2016, 23:59
15 %	Etapa 2: Análisis Sintáctico	Lunes 18-Abril-2016, 23:59
35 %	Etapa 3: Análisis Semántico	Lunes 23-Mayo-2016, 23:59
35 %	Etapa 4: Generación de código	Lunes 27-Junio-2016, 23:59

Herramientas

El desarrollo será en lenguaje Java. Para el análisis léxico se utilizará el generador JFlex[1].

Requisitos

El objetivo es que su analizador léxico reciba como entrada un código fuente, y entregue como salida en pantalla (salida estándar) una lista de *tokens* de acuerdo a la especificación de Agú. Es su misión definir un conjunto apropiado de *tokens* que cumplan las siguientes condiciones:

- Los *tokens* deben poder ser desplegados en pantalla incluyendo toda su información relevante.
- Los *tokens* no deben perder información que pueda ser útil para las siguientes etapas de compilación. Información importante incluye: un valor (si es relevante para el *token*), línea y columna de detección (para un eventual mensaje de error).

El analizador léxico debe ser robusto. Esto significa que no debe fallar en caso de encontrar un lexema inválido en el código de entrada de acuerdo a la definición de Agú. Por el contrario, debe reportarlo como error (por ejemplo, agregando un *token* especial para representar el error) junto con un mensaje de error descriptivo que pueda ayudar al programador a detectar el error. Una vez detectado el error, el analizador léxico debe continuar leyendo la entrada, e intentar detectar la mayor cantidad de *tokens* posible.

Un analizador correcto no solamente procesa códigos bien escritos, sino que también códigos mal escritos. Se probará su analizador con códigos de entrada que cumplan y que no cumplan las especificaciones de Agú.

Las consultas deben ser efectuadas a través del foro del curso (en SIDIng), de manera de poder beneficiarse de las preguntas y respuestas de los compañeros.

Instrucciones de entrega

Se habilitará un cuestionario de SIDing donde deberán subir un archivo comprimido (`*.tar.gz` o `.zip`) con los archivos **fuente** necesarios para ejecutar esta etapa del proyecto (no suban ejecutables o binarios), y un archivo `README.txt` con las instrucciones para compilar, ejecutar y probar su código.

[1] http://www.jflex.de/

Evaluación

La siguiente es la ponderación de los distintos ítemes. Los subítemes indicados son puntos a evaluar, pero no todos tienen necesariamente la misma ponderación.

- 10 %. Respeto a las formalidades, condiciones de entrega correctas, y que el código compile.
- 30 %. Código del analizador para JFlex.
  - Todos los elementos importantes del lenguaje deben estar representados
  - Definición de conjunto de tokens apropiados
  - Información almacenada dentro de los *tokens*
- 60 %. Casos de prueba para el analizador.
  - Respuesta ante *tokens* válidos
  - Respuesta ante *tokens* inválidos: detección de errores
  - Capacidad de detectar la mayor cantidad de tokens posibles
  - Calidad de la información desplegada de *tokens*

Se verificará activamente eventuales casos de copia.