Instituto Tecnológico de Costa Rica
Centro Académico de Alajuela
Ingeniería en Computación
IC-5701. Compiladores e Intérpretes
Prof. Emmanuel Ramírez Segura
Semestre I – 2020
Valor: 15%
Fecha de Asignación: 09/06/2020
Fecha de Entrega: 30/06/2020

Proyecto #2:

Creación de un Analizador Sintáctico – Java Tropicalizado al Español

➢ Objetivos General

Aplicar principios, modelos y técnicas para el diseño y la construcción de procesadores de lenguajes de programación de alto nivel, con énfasis en compiladores e intérpretes, para este proyecto, enfocándose en el diseño de un Analizador Sintáctico.

➢ Objetivos Específicos

1. Identificar los principales problemas relacionados con la implementación de lenguajes de programación.

2. Comprender principios y métodos para implementar lenguajes de programación.

3. Comprender modelos abstractos de lenguajes y máquinas formales (gramáticas, expresiones regulares y autómatas) y su aplicabilidad para la implementación de lenguajes de programación.

4. Diseñar un analizador Sintáctico que procese el lenguaje a utilizar.

- **Especificación del Proyecto**

El proyecto consistirá en implementar un analizador sintáctico sobre el subconjunto del lenguaje Java con el que trabajó en el proyecto #1.

- **Requerimientos de Diseño de Software**

1. El lenguaje fuente sobre el que construirá el analizador sintáctico será Java o Python, deberá continuar con el mismo lenguaje con el que inició la fase léxica.

2. La construcción de esta fase se denominará “faseSintáctica”, la cual depende de la “faseLéxica”, es necesario que, cualquier error de la “faseLéxica” sea depurado para seguir con la generación de la “faseSintáctica”.

3. La gramática sobre la cual se construirá la fase sintáctica, corresponde a la misma gramática traducida al español como parte del proyecto #1.

4. El programa a entregar deberá utilizarse desde la línea de comandos.

5. Se deberá entregar al finalizar el analizador sintáctico con el siguiente software:

    a. El código fuente desarrollado, el mismo debe encontrarse con comentarios generales indicando lo que hacen los procedimientos, funciones que se utilizan.
    b. Un archivo README.txt que indique cómo ejecutar su programa en la línea de comandos.
    c. Dos archivos de prueba: prueba1.txt y prueba2.txt que permitan ser utilizados para corroborar el funcionamiento de su analizador sintáctico.

6. Sobre la manera de ejecutar el programa del analizador sintáctico:

[NOMBRE DEL PROGRAMA] [ARCHIVO DE ENTRADA] [ARCHIVO SALIDA]

Descripción:

[NOMBRE DEL PROGRAMA]: Se debe utilizar el nombre **analizadorSintactico**.
[ARCHIVO DE ENTRADA]: Es el archivo a ser analizado. Por ejemplo: prueba1.txt.
[ARCHIVO DE SALIDA]: El archivo de salida deberá contener el análisis sintáctico, siempre y cuando no existan errores en la faseLéxica. Si no hay errores léxicos pero si hay errores sintácticos el archivo de salida será:

Por ejemplo, el [ARCHIVO DE SALIDA CONTENDRÁ]:

Línea 1: Sintácticamente Correcta
Línea 2: Con error sintáctico, cercano al token XXX

Nota: Apenas se detecte un error sintáctico, el programa se detendrá. Para el ejemplo de arriba se detuvo en la línea 2.

7. Puesto que el analizador sintáctico **debe** reportar errores, en esta parte sólo será necesario indicar el número de línea que contiene el error y cerca de cual token está, por ejemplo***: "La línea 2, Con error sintáctico, cercano al toquen XXX".***

8. **No se requiere para esta entregar implementar técnicas de corrección de errores**, es decir, ante un error, el programa desplegará lo contenido en el punto 7 y finalizará la ejecución del programa.

9. Se podrá hacer uso de Generadores Sintácticos automáticos **siempre y cuando la generación del código de salida cumpla con lo descrito en el punto 10**, de no cumplirse la entrega no será revisada.

10. El analizador sintáctico a construir debe ser del tipo: **Descendente Recursivo**. Para que su analizador sintáctico descendente recursivo no tenga inconvenientes, deben recordar dos puntos importantes:

    a. Eliminar la recursividad por la izquierda en las producciones.
    b. Factorizar por la izquierda las producciones que así lo requieran.

    Adicionalmente y no menos importante, su analizador sintáctico debe generar una construcción de AST (Abstract Syntatic Tree), para comprender de mejor manera esto, revise el libro "***Programming Language Processors in Java***", específicamente las páginas 114-118 (Sección 4.4.2), complementariamente revise el material de las páginas 89-114.

- **Requerimientos del Reporte**

Se deberá entregar un documento con las siguientes secciones, cada una en **una hoja independiente**.

PORTADA (incluir, entre lo usual el número de grupo, carné de los estudiantes, nombre de los estudiantes, fecha de entrega, semestre, año).

INDICE (generado automáticamente y con la numeración de páginas correcta).

GRAMATICA (la gramática en español generada por su grupo (la misma solicitada en el reporte del proyecto #1)).

GRAMATICA MODIFICADA (Es la GRAMATICA por usted entregada en el proyecto 1, con la salvedad de que debe venir **sin recursividad por la izquierda y factorizada por la izquierda**).

EJEMPLOS DE ARCHIVOS PROCESADOS (Debe aportar al menos 4 evidencias del funcionamiento de su programa, para ello, genere a drede errores sintácticos que puedan ser detectados por su analizador, coloque dos evidencias con errores y dos evidencias sin errores, sea explícito y detalle por qué hay error y por qué no desde el punto de vista de las producciones de la gramática).

LOGROS, ERRORES O PROBLEMAS
-Describir brevemente qué se logró, que no se implementó **justificando objetivamente** el por qué.

-Colocar imágenes de ejemplo de análisis sintácticos realizados con su proyecto.
-En caso de existir, describir qué problemas o errores persisten, si no hay errores o problemas de igual manera indicarlo.

- **Consideraciones a tomar en cuenta en general**

1. Los diversos grupos pueden aportar ideas en la solución de sus proyectos entre sí, **pero se prohíbe compartir códigos o implementaciones entre los grupos**.

2. **La solución de los proyectos debe ser algo inédito**, se permitirá utilizar únicamente segmentos de códigos derivados del libro de consulta recomendado "*Programming Language Processors in Java*".

3. El profesor bajo ninguna circunstancia atenderá dudas originadas a problemas de diseño o implementación de los proyectos, **únicamente atenderá dudas en cuanto al entendimiento del proyecto si hubieren**.

- **Aspectos Evaluativos**

Aspecto a evaluar	Porcentaje
Requerimientos del Diseño del Software	**80%**
Requerimientos del Reporte	**20%**
--------	**100%**

Sobre cada Aspecto se evaluará:	Porcentaje
Excelente (cumple con lo solicitado)	**[90% al 100%]**
Muy Bueno	**[80% a 89%]**
Bueno	**[70% - 79%]**
Regular	**[50% - 69%]**
Malo	**[1% al 49%]**
No hay entrega	**0%**

- **Aspectos de entrega**

1. La fecha de entrega de este proyecto será única.

   Fecha de Entrega: Antes del 30/06/2020 – 11:59pm GMT-6

2. **Comodín a favor del estudiante**: Dada la situación generada a raíz del COVID-19, se concede a cada grupo, la posibilidad de aplazar la fecha de entrega una semana. Aquellos estudiantes que logren entregar el proyecto en la fecha según corresponde (y que este sea funcional, completo y sin errores) se le reconocerá 10% EXTRA en la nota global del mismo.

3. **Superpoder del profesor**: El profesor puede solicitar avances de los proyectos a los estudiantes, a partir de la semana #1 posterior a la fecha de asignación de este proyecto, los mismos no tendrán nota, pero se tomarán para evidenciar el trabajo y avance de los grupos.

El medio de entrega digital a la dirección de correo: entregasITCR@gmail.com debe de adjuntar un archivo .ZIP con:

a) Código Fuente.
b) Archivo README.txt
c) Archivos de Pruebas 1 y 2.

IMPORTANTE:

Para la entrega, coloque en el **Asunto** del correo: Proyecto N2. Análisis Sintáctico Grupo X
El Adjunto deberá indicar: ProyectoN2_AnalisisSintáctico_GrupoX.zip

- **Libro de Consulta Recomendado**

1. David Watt y Deryck Brown, “Programming Language Processors in Java”.