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: 14/07/2020
Fecha de Entrega: 11/08/2020

Proyecto #3:

Desarrollando un Mini-Compilador de Expresiones Algebraicas

➢ 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 compilador simple.

➢ 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 varias fases de un compilador simple.

- **Especificación del Lenguaje del Mini Compilador**

El proyecto consistirá en implementar un mini compilador de varias pasadas con la siguiente especificación de lenguaje:

```
programa      → declaraciones
              ;
declaraciones→ asignacion
              |  asignacion ';' declaraciones
              ;
asignacion    → ID '='  expresion;
expresion     → expresion '+' term
              | expresion '-' term
              | termino
              ;
termino       → termino '*' factor
              | termino '/' factor
              | factor
              ;
factor        → '(' expresion ')'
              | ID
              | numero
              ;
digito        → 0|1|2 ... |9
numero        → digito | digito numero
imprimir      → 'imprime' ID
              ;
```

Donde:

1. **ID**: es el nombre de un identificador conformado únicamente por una letra minúscula, de la “a” a la “z”.
2. Todos los elementos marcados dentro de la especificación del lenguaje con **letra negrita** constituyen a símbolos terminales del lenguaje.

Ejemplo de un programa construido con la sintáxis anterior:

```
x = 100 ;
z = 300 + 10 * ( x – 50 ) ;
w = z / x ;
imprime w ;
```

NOTAS IMPORTANTES:

1. Tomar en cuenta que las operaciones **siempre serán números enteros** de un máximo de 16 bits (1 palabra).
2. La separación entre elementos: identificadores (IDs), números, puntos y coma, operadores como =, +, -, /, * o el imprime siempre será un espacio esto para facilitar la identificación de los lexemas.
3. El punto y coma validará siempre el fin de una línea.
4. Las operaciones siempre deberán dar como resultado números enteros (omita resultados decimales).

Requerimientos de Diseño de Software

1. El lenguaje fuente sobre el que construirá el analizador sintáctico será Java o Python, utilizando las bondades de la programación Orientada a Objetos.

2. El proyecto deberá ser elaborado por etapas claramente definidas como archivos independientes (similar al programa/compilador TRIANGLE del libro "Programming Language Processors in Java"). Se deberán identificar etapas independientes como:

    a. **Análisis Léxico**
        i. En esta etapa se espera ver el reconocimiento de los lexemas y su clasificación correcta en tokens.
        ii. Se espera ver en caso de error, se despliegue la línea que tiene el error y se identifique el token inválido.
    b. **Análisis Sintáctico**
        i. En esta etapa se espera ver la generación de los métodos **Parse** para determinar si la estructura sintáctica del programa está correcta o no.
        ii. Se espera la generación del árbol AST correspondiente.
        iii. Se espera ver en caso de error, se despliegue la línea que tiene el error.
    c. **Análisis Semántico**
        i. Se espera tener métodos visitantes que recorran el árbol AST y lo decoren.
        ii. Se espera identificar errores semánticos, como, por ejemplo:

Error semántico: x se usa pero no se asigna:

```
z = 300 + 10 * x - 50 ;
w = z / x ;
imprime w ;
```

        iii. El orden de precedencia de operadores es:

-Suma y Resta tienen la misma precedencia.
-Multiplicación y División tienen la misma precedencia, pero tienen mayor precedencia que la Suma y la Resta.
-Paréntesis abierto y paréntesis cerrado tienen la mayor precedencia.

    d. **Generación de Código**
        i. Se espera ver el código en ensamblador (en Arquitectura Intel) del código resultante y su correcta compilación y resultado.

Queda a libertad el nombre y la cantidad de los archivos a utilizar en la implementación de cada etapa (al menos un archivo por etapa, trate de modularizar su código, tome como ejemplo el programa Triángulo).

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

4. Se deberá entregar al finalizar el proyecto 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 compilador.

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

[NOMBRE DEL PROGRAMA] [ARCHIVO DE ENTRADA]

Deberá desplegar al finalizar cada etapa si pasó correctamente o hay algún error, en caso de error identificar a qué se debe el error, no debe corregirlo. En caso de error el programa termina.

Posible ejemplo de salida (se evidencia error en la Etapa Sintáctica):

```
Etapa: Análisis Léxico: Completada correctamente.
Etapa: Análisis Sintáctico: Error sintáctico en línea 7, se finaliza el programa.
```

6. El analizador sintáctico a construir debe ser del tipo: **Descendente Recursivo**.

7. Se recomienda tomar ideas del compilador TRIANGLE del libro "***Programming Language Processors in Java***" como mecanismo de guía para la realización de este compilador.

8. Dados los inconvenientes tenidos con anterioridad con los generadores automáticos, **se prohíbe la utilización de los mismos como parte de esta entrega**.

9. La generación de código se realizará mediante ensamblador de la arquitectura Intel.

- **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**).

EVIDENCIAS (Debe aportar al menos 2 ejemplos para cada etapa realizada, una con errores y otra sin errores, que permita evidenciar las salidas de las diferentes fases de su programa ante archivos de código fuente de entrada).

CODIGO DEL PROGRAMA

Mencionar brevemente los archivos implementados, a qué etapa pertenecen y qué realizan.

NOTA EVALUATIVA POR PARTE DE LOS COMPAÑEROS

Se deberá incluir la nota de los compañeros a cada miembro del grupo, la nota tendrá un impacto sobre la nota de este proyecto para cada miembro. Se apela a la ética, a la objetividad y al respeto cuando se evalúen a los miembros de su grupo.

Integrante a evaluar: NOMBRE APELLIDO1 APELLIDO2

	Malo (0pts)	**Regular (5pts)**	**Bueno (10pts)**
Aporte de Ideas	El compañero no aporta ideas, tampoco busca soluciones efectivas a problemas del proyecto.	El compañero a veces aporta ideas y en ocasiones busca soluciones efectivas a problemas del proyecto.	El compañero siempre aporta ideas y en siempre busca soluciones efectivas a problemas del proyecto.
Cumplimiento de labores asignadas	El compañero no cumple con las labores asignadas dentro del grupo.	El compañero en ocasiones cumple con las labores asignadas dentro del grupo.	El compañero siempre cumple con las labores asignadas del grupo.
Calidad en el trabajo entregado	El compañero no entrega sus avances en la forma solicitada en el tiempo asignado.	El compañero a veces entrega sus avances en la forma solicitada en el tiempo asignado.	El compañero siempre entrega sus avances en la forma solicitada y el tiempo asignado.
Trabajo Equitativo	El compañero no realiza un trabajo equitativo en el grupo.	El compañero a menudo realiza un trabajo equitativo en el grupo.	El compañero siempre realiza un trabajo equitativo en el grupo.
Evalúa: Nombre Integrante 1			
Evalúa: Nombre Integrante 2			
Evalúa: Nombre Integrante 3			
Evalúa: Nombre Integrante 4 (si aplica)			
TOTAL			

LECCIONES APRENDIDAS: Posterior al diseño de su compilador, especificar:

Qué mejoras propondría a su compilador?

Qué etapa fue la más compleja de implementar y por qué lo considera de esa manera?

- **Evaluación**

Requerimientos de Diseño de Software	**80%**
Requerimientos del Reporte y Entrevista	**20%**
Total	**100%**

- **Evaluación Desglosada (Llenado por el profesor)**

Requerimientos de Diseño de Software

Rubro / Puntaje	**No Implementado o No funcional (0pts)**	**Funcional con algunos errores encontrados (Mitad de los puntos)**	**Funcional y sin errores (Todos los puntos)**
Análisis Léxico / 20 puntos			
Análisis Sintáctico / 40 puntos			
Análisis Semántico / 40 puntos			
Generación de Código / 10 puntos			
TOTAL			

Requerimientos del Reporte y Entrevista

Rubro / Puntaje	**No se presenta (0 puntos)**	**Se presenta con omisiones (Mitad de los puntos)**	**Se presenta según indicaciones (Todos los puntos)**
Reporte Escrito / 50			
Nota Evaluativa por parte de los compañeros / 20			
Entrevista / 30			
TOTAL			

- **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 ya que se considerará como plagio.**

2. Cualquier proyecto entregado posterior a la fecha/hora de entrega límite tendrá calificación cero.

3. **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*".

4. 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 de entrega**

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

   Fecha de Entrega: Antes del 11/08/2020 – 1:00pm GMT-6

2. 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.
   d) Reporte en formato PDF

3. Se hará una revisión vía ZOOM únicamente el día Martes **11/08/2020 de 3:30pm a 8pm**, para realizar una entrevista a cada grupo por aparte para revisar su proyecto. La asistencia de la **revisión es obligatoria,** si no media justificación se castigará al grupo con la mitad del puntaje de la entrevista.

IMPORTANTE:
Para la entrega, coloque en el **Asunto** del correo: Proyecto N3. Mini Compilador Grupo X
El Adjunto deberá indicar: ProyectoN3_MiniCompilador_GrupoX.zip

- **Libro de Consulta Recomendado**

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