Actividad de Aprendizaje – Proyecto final (parte 1)

Nombre de la Actividad: Implementación del Analizador Léxico de un Lenguaje.

Modalidad de trabajo: No presencial

Resultados de aprendizaje:

- Diseña algoritmos computacionales que reconocen los elementos básicos de un lenguaje de programación, con base en los modelos formales de la teoría computacional.

Proceso:

Por parejas, implementar un programa para reconocer código fuente escrito en un lenguaje de programación básico que incluye estructuras y sentencias de control para representar operaciones aritméticas usando variables y números en base octal. El proceso de interpretación constará de dos fases:

1. Análisis Léxico. Consiste en identificar las unidades léxicas (tokens) del código para determinar si son correctas. Genera una secuencia de tokens y una tabla de símbolos para los **identificadores**, las **literales numéricas** y **literales de texto**. El análisis léxico reportará las unidades léxicas incorrectas y la línea del código en donde se encuentran.
2. Análisis Sintáctico. Consiste en identificar si la secuencia de tokens obtenida del analizador léxico corresponde a estructuras gramaticales correctas del lenguaje de programación. El análisis sintáctico reportará si la compilación del programa es correcta o incorrecta y los errores encontrados.

Analizador léxico.

Este programa leerá el texto del código fuente, escrito en el lenguaje de programación MIO (“programa.mio”), e identificará las unidades léxicas del lenguaje; en caso de que una unidad léxica no pertenezca al lenguaje, se imprimirá un mensaje de error que incluya el número de línea del archivo del código de entrada en donde se encontró el error. La secuencia de tokens identificados se escribirá en un archivo de texto (“programa.lex”), un token por línea. Los elementos del lenguaje de programación MIO son:

Palabras reservadas

- PROGRAMA
- FINPROG
- SI
- ENTONCES
- SINO
- FINSI
- REPITE
- VECES
- FINREP
- IMPRIME
- LEE

Operadores relacionales

- Mayor que (>)
- Menor que (<)
- Igual a (==)

Asignación (=)
Comentario (# al inicio de la línea hasta encontrar el fin de línea)

Operadores aritméticos

- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)

Identificadores. Son cadenas de caracteres alfanuméricos que se usan para identificar variables y programas, tienen una longitud de hasta 16 caracteres. Para los identificadores de longitud mayor a 16 caracteres se runcarán y solo se conservarán los primeros 16. Inician siempre con un carácter alfabético y son sensibles a mayúsculas y minúsculas. Por ejemplo: DiaDeLaSemana1, variable03, X2.

Literales de texto. Son cadenas de caracteres alfanuméricos encerrados entre comillas. Por ejemplo: “caracteres alf4num3ric05”. No tienen restricción en su longitud.

Literales numéricas. Son cadenas de caracteres numéricos que representan valores numéricos en base octal. Se utiliza la numeración posicional en base 8 y se utilizan los dígitos del 0 al 7 para los elementos básicos.

La tabla de símbolos constará de tres secciones: la primera es una matriz que almacena los identificadores (IDS[]), la segunda almacena las literales de texto (TXT[]), y la tercera almacena las literales numéricas (VAL[]) Cada fila de una matriz contiene un token detectado dependiendo de su tipo. Las tres secciones de la tabla de símbolos se escribirán en un archivo (“programa.sim”), una después de otra, en el orden en que se definieron los token en el código de entrada.

Las matrices IDS y TXT tendrán dos columnas: la primera contiene el valor del token detectado, y la segunda contiene una clave alfanumérica consecutiva (inicia en ID01 para IDS y en TX01 para TXT). La matriz de literales numéricas contiene también dos columnas: la primera almacena el token en su representación octal, y la segunda, el valor decimal que corresponda.

La gramática del lenguaje MIO es la siguiente:

Símbolo inicial: <PROG>
Reglas:
<PROG> → PROGRAMA [id] <SENTS> FINPROG
<SENTS> → <SENT> <SENTS>
<SENTS> → <SENT>
<SENT> → [id] = <ELEM> [op_ar] <ELEM>
<SENT> → [id] = <ELEM>
<SENT> → SI <COMPARA> ENTONCES <SENTS> SINO <SENTS> FINSI
<SENT> → SI <COMPARA> ENTONCES <SENTS> FINSI
<SENT> → REPITE <ELEM> VECES <SENTS> FINREP
<SENT> → IMPRIME <ELEM>
<SENT> → IMPRIME [txt]
<SENT> → LEE [id]
<SENT> → # [comentario]
<ELEM> → [id]
<ELEM> → [val]
<COMPARA> → [id] [op_rel] <ELEM>

Ejemplo. El programa fuente llamado “factorial.mio” contiene las siguientes sentencias:

```
# Programa que calcula el factorial de un número
PROGRAMA factorial
# VarX acumula los productos por iteración
VarX = 1
# VarY contiene el iterador del factor
VarY = 0
LEE Num
#  Aplica Num! = 1 * 2 * 3 * … * Num
REPITE Num VECES
   VarY = VarY + 1
   VarX = VarX * VarY
FINREP
IMPRIME “Factorial de ”
IMPRIME Num
IMPRIME “ es “
IMPRIME VarX
FINPROG
```

El analizador léxico se deberá ejecutar desde la línea de comandos de la siguiente manera:

```
$ analex factorial.mio
```

y generará los archivos “factorial.lex” con la siguiente secuencia de tokens:

```
PROGRAMA
[id] ID01
[id] ID02
=
[val]
[id] ID03
=
[val]
LEE
[id] ID04
REPITE
[id] ID04
VECES
[id] ID03
=
[id] ID03
[op_ar]
[val]
[id] ID02
=
```

[id] ID02
[op_ar]
[id] ID03
FINREP
IMPRIME
[txt] TX01
IMPRIME
[id] ID04
IMPRIME
[txt] TX02
IMPRIME
[id] ID02
FINPROG

y el archivo “factorial.sim” con los siguientes arreglos de símbolos:
IDS
factorial, ID01
Varx, ID02
VarY, ID03
Num, ID04

TXT
“Factorial de ”, TX01
“ es “, TX02

VAL
1, 1
0, 0

Valor de la Actividad: 100 puntos

Fecha de entrega: por definir.

Formato de entrega:

a) Entrega por equipo: En plataforma, entregar en un archivo comprimido el código fuente y manual del usuario. Incluir en la portada de la documentación el nombre completo de los integrantes del equipo.

b) Entrega individual: Formato de autoevaluación.

Recursos y referencias:

1. Martin. J.C. (2010). Introduction to languages and the theory of computation (4th Ed.). McGraw Hill. USA.
2. Brena, R. (2003). Lenguajes Formales y Autómatas. Centro de Inteligencia Artificial, Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Monterrey.

Instrumento de evaluación: La calificación del producto se asignará de acuerdo a la siguiente rúbrica:

Aspectos a evaluar	Excelente -100%	Bien -85%	Regular -70%	Deficiente -60%
Analizador léxico 40 puntos	El programa lee correctamente el código fuente y clasifica los tokens de acuerdo a la especificación	El programa lee correctamente el código fuente y tiene a lo más 2 errores en la clasificación de tipos de tokens	El programa lee correctamente el código fuente y tiene a lo más 4 errores en la clasificación de tipos de tokens	El programa no lee el código fuente correctamente ni identifica los tokens de acuerdo la especificación
Interpretación de valores octales 20 puntos	El programa identifica e interpreta correctamente el valor decimal de los números octales y registra ambos elementos en la tabla de símbolos.	El programa identifica correctamente el números octales, calcula correctamente su valor decimal pero no los registra en la tabla de símbolos	El programa identifica correctamente los números octales pero no interpreta correctamente el valor decimal correspondiente	El programa no identifica ni interpreta correctamente el valor decimal de los números octales.
Generación de archivos temporales 20 puntos	Los archivos temporales .lex y .sim contienen la información correcta y completa	El archivo .lex contiene información correcta y completa, pero el archivo .sim presenta errores.	Los archivos .lex y .sim presentan a lo más 5 errores en su información en conjunto.	Los archivos .lex y .sim presentan al menos 6 errores en su información en conjunto.
Manual del usuario 20 puntos	El manual del usuario describe correctamente la operación del programa e incluye casos de uso. Menos de 5 errores ortográficos y gramaticales	El manual del usuario describe correctamente la operación del programa e incluye casos de uso. Entre 6 y 10 errores ortográficos y gramaticales	El manual del usuario describe correctamente la operación del programa, pero no incluye casos de uso. Entre 11 y 15 errores ortográficos y gramaticales.	No entrega el manual del usuario; o éste no describe correctamente la operación del programa y no incluye casos de uso. Más de 15 errores ortográficos y gramaticales.

Autoevaluación personal: **20 puntos**.