Instituto Tecnológico de Costa Rica

Escuela de Computación

IC-5701 Compiladores e Intérpretes

**Proyecto 1**

**Fases de Análisis Léxico y Sintáctico**

Estudiantes:

Diaz Coto Johnny Andres 2020042119

Narvaez Carolina

Ocampo Nikolas

Profesor:

Ingnacio Trejos Zelaya

II Semestre, 2022

Contenido

[Su esquema para el manejo del texto fuente (lectura de archivos de entrada o de estructura de datos del IDE). Si no modifica lo existente, indique esto explícitamente. 3](#_Toc115427886)

[Modificaciones hechas al analizador de léxico (tokens, tipos, métodos, etc.). 3](#_Toc115427887)

[Cambios hechos a los tokens y a cualquier otra estructura de datos (por ejemplo, tabla de palabras reservadas) para incorporar las extensiones al lenguaje. 3](#_Toc115427888)

[Explicación su estrategia para generar la versión HTML del texto del programa fuente. 3](#_Toc115427889)

[Justificar explícitamente cualquier cambio realizado a las reglas sintácticas de xt, para lograr que tenga una gramática LL(1) equivalente a la extensión descrita arriba. 3](#_Toc115427890)

[Generar un apéndice con la macro-sintaxis (sintaxis) y la micro-sintaxis (léxico) de xt tal que satisfaga las propiedades LL(1). 3](#_Toc115427891)

[Nuevas rutinas de reconocimiento sintáctico, así como cualquier modificación a las existentes. 3](#_Toc115427892)

[Lista de nuevos errores sintácticos detectados, con los nuevos mensajes de error. 3](#_Toc115427893)

[Modelaje realizado para los árboles de sintaxis abstracta (ponga atención al modelaje de categorías sintácticas donde hay ítems repetidos14). 3](#_Toc115427894)

[Extensión realizada a los métodos que permiten visualizar los árboles de sintaxis abstracta (desplegados en una pestaña del IDE). 3](#_Toc115427895)

[Extensión realizada a los métodos que permiten representar los árboles de sintaxis abstracta como texto en XML. 3](#_Toc115427896)

[Pruebas 4](#_Toc115427897)

[Discusión y análisis de los resultados obtenidos 4](#_Toc115427898)

[Una reflexión sobre la experiencia de modificar fragmentos de un compilador/ambiente escrito por terceras personas. 4](#_Toc115427899)

[Descripción resumida de las tareas realizadas por cada miembro del grupo de trabajo. 4](#_Toc115427900)

[Indicar cómo compilar su programa. 4](#_Toc115427901)

[Indicar cómo ejecutar su programa. 4](#_Toc115427902)

# Su esquema para el manejo del texto fuente (lectura de archivos de entrada o de estructura de datos del IDE). Si no modifica lo existente, indique esto explícitamente.

Esta sección del proyecto no ha sido modificada.

# Modificaciones hechas al analizador de léxico (tokens, tipos, métodos, etc.).

En esta sección se han modificado distintos archivos para lograr el funcionamiento del proyecto, estos archivos han sido:

* Parser.java
* Scanner.java
* Toke.java

# Cambios hechos a los tokens y a cualquier otra estructura de datos (por ejemplo, tabla de palabras reservadas) para incorporar las extensiones al lenguaje.

En el archivo de token.java se agregaron las nuevas palabras reservadas dadas por el profesor en la especificación del proyecto, estas fueron: for, from, init, leave, local, loop, next, nil, rec, return, select, to, until, when. A esto también se agregó el símbolo pipe, “|” como un nuevo símbolo léxico. Todo esto fue agregado también a la tabla de palabras reservadas.

Cada agregación tiene su comentario respectivo.

Tabla

Descripción generada automáticamente

Ilustración 1 Cambios realizados a las palabras reservadas en token.java

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Ilustración 2 Pipe ha sido añadido a los símbolos léxicos.

# Explicación su estrategia para generar la versión HTML del texto del programa fuente.

# Justificar explícitamente cualquier cambio realizado a las reglas sintácticas de xt, para lograr que tenga una gramática LL(1) equivalente a la extensión descrita arriba.

# Generar un apéndice con la macro-sintaxis (sintaxis) y la micro-sintaxis (léxico) de xt tal que satisfaga las propiedades LL(1).

# Nuevas rutinas de reconocimiento sintáctico, así como cualquier modificación a las existentes.

# Lista de nuevos errores sintácticos detectados, con los nuevos mensajes de error.

# Modelaje realizado para los árboles de sintaxis abstracta (ponga atención al modelaje de categorías sintácticas donde hay ítems repetidos14).

# Extensión realizada a los métodos que permiten visualizar los árboles de sintaxis abstracta (desplegados en una pestaña del IDE).

# Extensión realizada a los métodos que permiten representar los árboles de sintaxis abstracta como texto en XML.

# Pruebas

Discusión y análisis de los resultados obtenidos.

# Una reflexión sobre la experiencia de modificar fragmentos de un compilador/ambiente escrito por terceras personas.

# Descripción resumida de las tareas realizadas por cada miembro del grupo de trabajo.

La elaboración el proyecto fue ejecutado en distintas secciones por los integrantes del grupo de trabajo, en un principio la idea era ir construyendo el proyecto en conjunto, sin embargo, distintas situaciones lo impidieron, así que por medio de mensajería se ha dividido el proyecto y organización, se puede ver la organización de una forma más visible con la siguiente matriz.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cambios | Realizados por: | Johnny D | Carolina N | Nikolas O |
| Eliminaciones | |  |  |  |
| e | | X |  |  |
| "begin" Command "end" | | X |  |  |
| "let" Declaration "in" single-Command | | X |  |  |
| "if" Expression "then" single-Command "else" single-Command | | X |  |  |
| "while" Expression "do" single-Command | | X |  |  |
| Agregación | |  |  |  |
| "nil" | | X |  |  |
| "let" Declaration "in" Command "end" | | X |  |  |
| "if" Expression "then" Command ("|" Expression "then" Command)\* "else" Command "end | | X | X |  |
| "loop" [Identifier] "while" Expression "do" Command "end" | | X | X |  |
| "loop" [Identifier] "until" Expression "do" Command "end" | |  | X |  |
| "loop" [Identifier] "do" Command "while" Expression "end" | |  | X |  |
| "loop" [Identifier] "do" Command "until" Expression "end" | |  | X |  |
| "loop" [Identifier] "for" Identifier "from" Expression "to" Expression "do" Command "end" | |  | X |  |
| "loop" [Identifier] "for" Identifier "from" Expression "to" Expression "while" Expression "do" Command "end" | |  | X |  |
| "loop" [Identifier] "for" Identifier "from" Expression "to" Expression "until" Expression "do" Command "end" | |  | X |  |
| "rec" Proc-Funcs "end" | | X |  |  |
| "local" Declaration "in" Declaration "end" | | X |  |  |
| "proc" Identifier "(" Formal-Parameter-Sequence ")" "~" Command "end" | | X |  |  |
| "func" Identifier "(" Formal-Parameter-Sequence ")" ":" Type-denoter "~" Expression | | X |  |  |
| Proc-Func ("|" Proc-Func)+ | | X |  |  |
| "var" Identifier "init" Expression | | X |  |  |
| Modificaciones | |  |  |  |
| "proc" Identifier "(" Formal-Parameter-Sequence ")" "~" Command "end" | | X |  |  |
| Declaration | | X |  |  |

# Indicar cómo compilar su programa.

# Indicar cómo ejecutar su programa.