Proyecto Diseño de compiladores

Curso: Diseño de compiladores

Sesiones: miércoles (3 horas)

Modalidad del entregable: Aplicación móvil, web o escritorio

Objetivo del proyecto

Construir, de forma incremental, un mini entorno de compilación para un lenguaje didáctico LL(1). El equipo deberá:

1. Identificar símbolos terminales y no terminales de una gramática dada.

2. Implementar un analizador léxico (tokenizador).

a. Deberá procesar un código fuente.

b. En caso de no existir error alguno, mostrar la tabla de tokens generada.

c. En caso de existir un error, detener la ejecución del compilador, mostrar en que línea del código fuente se encuentra el error y opcionalmente mostrar que token(s) esperaba el analizador.

3. Obtener el árbol de derivación de la gramática.

4. Implementar un analizador sintáctico.

5. Integrar una interfaz gráfica que permita lo siguiente:

a. Escribir, guardar y editar un código fuente.

b. Mostrar tabla de tokens.

c. Mostrar errores léxicos.

d. Mostrar si el análisis sintáctico tuvo algún error o si no, mostrar una alerta de que el código analizado paso ambos analizadores de forma exitosa.

6. Subir el proyecto a un repositorio de Git en donde se tienen que ver los hashes

de los commits.

Notas:

• Los analizadores deberán ignorar los comentarios del código, los cuales estarán dados por la siguiente expresión regular: //[A-Za-z0-9]+//.

• Los Id's estarán deberán comenzar por una letra (minúscula o mayúscula).

• Los string\_lit estarán dados de la siguiente forma: "Texto en string lit".

• Los num serán enteros, con punto decimal o notación científica ej. 12.2, 123

12e21, 13e-12.

Gramática

Program → program ID ';' DeclsOpt begin StmtList end '.' EOF

DeclsOpt → Decls

DeclsOpt → ε

Decls → Decl Decls

Decls → Decl

Decl → TypeSection

Decl → VarSection

Decl → SubprogDecl

TypeSection → type TypeDefList

TypeDefList → TypeDef TypeDefListTail

TypeDefListTail → ';' TypeDef TypeDefListTail

TypeDefListTail → ';'

TypeDef → ID '=' Type

VarSection → var VarDefList

VarDefList → VarDef VarDefListTail

VarDefListTail → ';' VarDef VarDefListTail

VarDefListTail → ';'

VarDef → IdList ':' Type

IdList → ID IdListTail

IdListTail → ',' ID IdListTail

IdListTail → ε

SubprogDecl → ProcedureDecl

SubprogDecl → FunctionDecl

ProcedureDecl → procedure ID '(' ParamSectionOpt ')' ';' Block ';'

FunctionDecl → function ID '(' ParamSectionOpt ')' ';' Type ';' Block ';'

ParamSectionOpt → ParamSection

ParamSectionOpt → ε

ParamSection → ParamGroup ParamSectionTail

ParamSectionTail → ';' ParamGroup ParamSectionTail

ParamSectionTail → ε

ParamGroup → IdList ':' Type

Type → SimpleType

Type → ArrayType

Type → NamedType

SimpleType → integer

SimpleType → boolean

SimpleType → string

ArrayType → array '[' NUM '..' NUM ']' of Type

Block → DeclSOpt begin StmtList end

StmtList → Stmt StmtListTail

StmtListTail → ';' Stmt StmtListTail

StmtListTail → ε

Stmt → AssignStmt

Stmt → WhileStmt

Stmt → CallStmt

Stmt → IfStmt

Stmt → Block

Stmt → ε

AssignStmt → Variable ':= ' Expr

Variable → ID VariableTail

VariableTail → '[' Expr ']' VariableTail

VariableTail → ε

CallStmt → ID '(' ArgListOpt ')'

ArgListOpt → ArgList

ArgListOpt → ε

ArgList → Expr ArgListTail

ArgListTail → ',' Expr ArgListTail

ArgListTail → ε

IfStmt → if Expr then Stmt ElseOpt

ElseOpt → else Stmt

ElseOpt → ε

WhileStmt → while Expr do Stmt

Bloque Común

Expr → Assign

Assign → Or AssignTail

AssignTail → '=' Assign

AssignTail → ε

Or → And OrTail

OrTail → '||' And OrTail

OrTail → ε

And → Eq AndTail

AndTail → '&&' Eq AndTail

AndTail → ε

Eq → Rel EqTail

EqTail → '==' Rel EqTail

EqTail → '!=' Rel EqTail

EqTail → ε

Rel → Add RelTail

RelTail → '<' Add RelTail

RelTail → '<=' Add RelTail

RelTail → '>' Add RelTail

RelTail → '>=' Add RelTail

RelTail → ε

Add → Mul AddTail

AddTail → '+' Mul AddTail

AddTail → '-' Mul AddTail

AddTail → ε

Mul → Unary MulTail

MulTail → '\*' Unary MulTail

MulTail → '/' Unary MulTail

MulTail → '%' Unary MulTail

MulTail → ε

Unary → '!' Unary

Unary → '-' Unary

Unary → Postfix

Postfix → Primary PostfixTail

PostfixTail → '(' ArgListOpt ')' PostfixTail

PostfixTail → '[' Expr ']' PostfixTail

PostfixTail → '.' ID PostfixTail

PostfixTail → ε

Primary → ID

Primary → NUM

Primary → STRING

Primary → 'true'

Primary → 'false'

Primary → '(' Expr ')'

ArgListOpt → ArgList

ArgListOpt → ε

ArgList → Expr ArgListTail

ArgListTail → ',' Expr ArgListTail

ArgListTail → ε