Universidad Rafael Landívar
Facultad de Ingeniería
Compiladores
Sección: 02
Ing. Moises Alonso

PROYECTO
FASE 1

Ana Sofía Asturias To, 1078123
Dereck Alexander Cabrera Ng, 1177223
Jorge Francisco Muñoz Portela, 1177523

Guatemala de la Asunción, 3 de marzo 2025

GRAMÁTICA

Gramática:

T = {int, float, string, bool, if, else, while, Read, Write, return, +, -, *, /, >, <, =, !=, (,), {, }, [,], ;, ',', ->, ID, STRING, INT, FLOAT, BOOL}

NT = {<PG>, <SL>, <S>, <D>, <E>, <E'>, <T>, <T'>, <F>, <DS>, <DS'>, <IO>, <CE>, <IFD>, <ED>, <ED'>, <FD>, <RT>, <PL>, <PL'>, <TY>, <TY'>}

$ = {<PG>}

P =
{
<PG> ::= <FD> BeginProgram <SL>
<SL> ::= <S> <SL> | ε
<S> ::= <D> | <CE> | <IO>
<D> ::= int ID = <E>; | float ID = <E>; | string ID = STRING; | bool ID = <DS>;
<E> ::= <T> <E'>
<E'> ::= + <T> <E'> | - <T> <E'> | ε
<T> ::= <F> <T'>
<T´> ::= * <F> <T´> | / <F> <T´> | ε
<F> ::= (<E>) | INT | FLOAT
<DS> ::= true | false | <E> <DS'> <E> | STRING == STRING | STRING != STRING
<DS'> ::= > | < | >= | <= | == | !=
<IO> ::= Read(ID); | Write(STRING);
<CE> ::= <IFD> | while (<DS>) { <SL> }
<IFD> ::= if (<DS>) { <SL> } <ED>
<ED> ::= else <ED'> | ε
<ED'> ::= { <SL> } | <IFD>
<FD> ::= fn ID(<PL>) <RT> <FD> | ε
<RT> ::= -> <TY> { <SL> return <TY'> } | { <SL> }
<PL> ::= <TY> ID <PL'> | ε
<PL'> ::= , <TY> ID <PL'> | ε
<TY> ::= int | float | string | bool
<TY'> ::= INT | FLOAT | STRING | BOOL
}

Expresiones Regulares:

ID = [a-zA-Z][a-zA-Z1-9_]*
INT = [0-9]+
FLOAT = f[0-9]+\.[0-9]+
STRING = "([^"\\]*(\\.[^"\\]*)*)"
BOOL = true|false
KEYWORD = int | float | string | bool | if | else | while | Read | Write | return
SPECIALCHAR = + | - | * | / | > | < | >= | <= | = | != | (|) | { | } | [|] | ; | , | ' | "

EXPLICACIÓN DE LA GRAMÁTICA

ANÁLISIS GRAMATICAL

- El nombre de los No Terminales se definieron como abreviaciones de las palabras originales para menor confusión (PG = program, E = expression, etc).

Desglose de las Producciones:

- En <PG> se le añadió “BeginProgram” para que cuando el programa identifique esta cadena, se termine la declaración de funciones y comience la declaración de sentencias.

- En <D> se realizó de esa manera para evitar que se declaren variables de tipo int e ingresarle una cadena de caracteres o cosas similares.

- Se declaró el No Terminal <IFD> de manera que la función “if” solo pudiera comparar valores o cadenas de caracteres, seguida de un “else” opcional. Esto para evitar declarar funciones dentro de otras funciones o dentro del “if” o “while” y que el programa no quedara demasiado largo.

- Para declarar las funciones se decidió agregar “fn” antes para ubicarlas fácilmente.

- Se declararon los Terminales INT, FLOAT, STRING, BOOL en mayúscula para dar a entender que serán los valores, es decir, los números o cadenas de caracteres. Por otro lado, se declararon también en minúsculas para dar a entender el tipo de variable que se está declarando, es decir, lo que va antes del ID.

- La gramática utiliza No Terminales como <E>, <T> y <F> para definir la precedencia de los operadores aritméticos (multiplicación/división antes que suma/resta).

EXPRESIONES REGULARES

- Hicimos estas expresiones regulares específicas para que la gramática no quedara demasiado larga.

- Se declaró la ER STRING de esta manera ya que puede aceptar cualquier carácter luego de las comillas (“), las cuales marcan el inicio de la cadena.