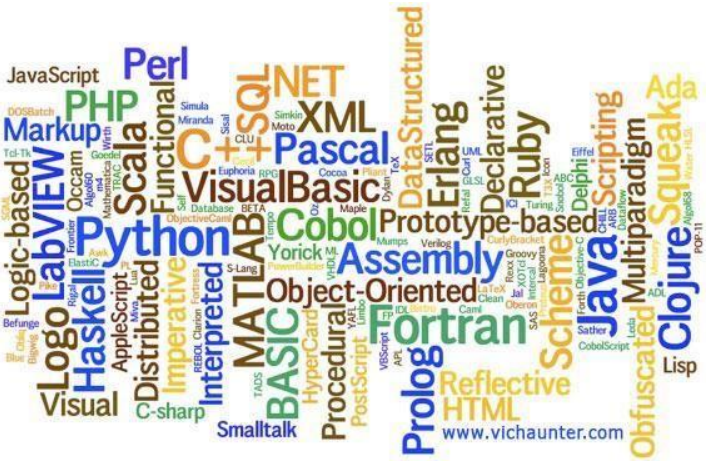
Tecnológico de Costa Rica
Escuela de Ingeniería en Computación



Tarea I “Gramática BNF”

Compiladores e intérpretes

Profesor:
Rodríguez Dávila, Allan
Presentado por:
Bonilla Espinoza, Alina [2016248502]
Lara Oses, Froylan [2018276191]

Setiembre, 2023

Descripción del problema

Un grupo de desarrolladores desea crear un nuevo lenguaje imperativo, ligero, fuertemente tipado, que le permita realizar operaciones básicas para la configuración de chips, ya que esta es una industria que sigue creciendo constantemente, y cada vez estos chips necesitan ser configurados por lenguajes más ligeros y potentes.

Es por esto que este grupo de desarrolladores requiere desarrollar su propio lenguaje para el desarrollo de sistemas empotrados, y como primer paso necesitan desarrollar una gramática simple y poderosa.

Diseño del programa

A continuación, se detalla la lista de terminales, no terminales y la gramática BNF del lenguaje en producción:

1. **Lista de terminales:**

Terminales			
;	bool	--	if
#	char	<	else
=	string	<=	switch
:	-	>	while
,	+	>=	forRange
(-?[1-9][0-9]*) | 0	*	==	main
<numeroInt>\.[0-9][1-9]*	/	!=	break
True	%	(	return
False	^	)	case
\'[^']\'	&	{	default
\" (\" | [^"])+\"	|	}	//.*"\n"
int	!	read	/* .*[\s.]* "*/"
float	++	write	

2. **Lista de no terminales:**

No terminales		
identificador	reservEscribir	creaAsigVarString
endLine	reservIf	variablesString
separador	reservElse	variables
igual	inicio	funcLeerInt
dosPuntos	reservSwitch	funcLeerFloat
coma	reservWhile	funcLeer
litInt	reservForRange	funcEscribirInt
litFloat	reservMain	funcEscribirFloat
litTrue	reservBreak	funcEscribirString
litFalse	reservReturn	funcEscribirId
litChar	reservCase	funcEscribir
litString	reservDefault	operacionBool
litBool	comentarioSimple	operandosRelaInt
tipoInt	comentarioMultiple	operacionRelaInt
tipoFloat	comentarios	operandosRelaFloat
tipoBool	sentReturn	operacionRelaFloat
tipoChar	sentBreak	operandosLogicos
tipoString	operaUnaria	operacionLogicas
signoResta	operacionUnaria	condiciones
signoSuma	valoresInt	sentencia
signoMulti	operacionInt	estructControl
signoDiv	varInt	valoresBloqueCodigo
signoMod	creaVarInt	bloqueCodigo
signoPot	creaAsigVarInt	estructElse
operadoresAritme	variablesInt	estructIf
signoConjuncion	valoresFloat	paramForRangeUno
signoDisyuncion	operacionFloat	paramForRangeDos
signoNegacion	varFloat	paramForRangeTres
operadoresLogicos	creaVarFloat	paramForRange
sigAumen	creaAsigVarFloat	estructForRange
sigDiminu	variablesFloat	estructWhile
operadoresUnarios	valoresBool	literal
signoMenor	varBool	case
signoMenorIgual	creaVarBool	default
signoMayor	creaAsigVarBool	estructSwitch
signoMayorIgual	variablesBool	tiposRetornoFunc
signoIgual	valoresChar	tiposParam
signoDiferente	varChar	paramFunciones
operadoresRela	creaVarChar	funcMain
sigAbreParent	creaAsigVarChar	creaFunc
sigCierraParent	variablesChar	paramLlamarFunc
sigAbreLlaves	valoresString	llamarFunc
sigCierraLlaves	varString	sentLlamarFunc

reservLeer	creaVarString	globalVar
valoresPrograma	programa	

3. **Símbolo inicial:**
   El símbolo inicial del lenguaje es: “inicio”, cuya producción corresponde a “inicio ::= <programa>”

4. **Producciones:**
   A continuación, se detalla la gramática BNF del lenguaje en producción:
   identificador ::= [a-zA-Z][a-zA-Z0-9_]*

   endLine ::= \;
   separador ::= \#
   igual ::= \=
   dosPuntos ::= \:
   coma ::= \,

   litInt ::= (-?[1-9][0-9]*) | 0
   litFloat ::= <numeroInt>\.[0-9][1-9]*
   litTrue ::= "True"
   litFalse ::= "False"
   litChar ::= \'[^']\'
   litString ::= \" (\" | [^"])+\"
   litBool ::= <litTrue> | <litFalse>

   tipoInt ::= "int"
   tipoFloat ::= "float"
   tipoBool ::= "bool"
   tipoChar ::= "char"
   tipoString ::= "string"

   signoResta ::= \-
   signoSuma ::= \+
   signoMulti ::= *
   signoDiv ::= \/

```
signoMod ::= \%
signoPot ::= \^
operadoresAritme ::= <signoResta> | <signoSuma> | <signoMulti> | <signoDiv> | <signoMod> | <signoPot>

signoConjuncion ::= \&
signoDisyuncion ::= \|
signoNegacion ::= \!
operadoresLogicos ::= <signoConjuncion> | <signoDisyuncion>

sigAumen ::= "++"
sigDiminu ::= "--"
operadoresUnarios ::= <sigAumen> | <sigDiminu>

signoMenor ::= "<"
signoMenorIgual ::= "<="
signoMayor ::= ">"
signoMayorIgual ::= ">="
signoIgual ::= "=="
signoDiferente ::= "!="
operadoresRela ::= <signoMenor> | <signoMenorIgual> | <signoMayor> | <signoMayorIgual> | <signoIgual> |
<signoDiferente>

sigAbreParent ::= \(
sigCierraParent ::= \)
sigAbreLlaves ::= \{
sigCierraLlaves ::= \}

reservLeer ::= "read"
```

```
reservEscribir ::= "write"

reservIf ::= "if"
reservElse ::= "else"
reservSwitch ::= "switch"
reservWhile ::= "while"
reservForRange ::= "forRange"
reservMain ::= "main"
reservBreak ::= "break"
reservReturn ::= "return"
reservCase ::= "case"
reservDefault ::= "default"

comentarioSimple ::= "//".*"\n"
comentarioMultiple ::= "/*" .*[\s.]* "*/"
comentarios ::= <comentarioSimple> | <comentarioMultiple>

sentReturn ::= <reservReturn> <endLine>
sentBreak ::= <reservBreak> <endLine>

operaUnaria ::= <identificador> <operadoresUnarios>
operacionUnaria ::= <operaUnaria> <endLine>

valoresInt ::= <litInt> | <identificador> | <llamarFunc>
operacionInt ::= <valoresInt>
operacionInt ::= <sigAbreParent> <valoresInt> <operadoresAritme> <valoresInt> <sigCierraParent> | <valoresInt>
        <operadoresAritme> <valoresInt>
```

varInt ::= <identificador> <igual> <operacionInt> <endLine>
creaVarInt ::= <tipoInt> <separador> <identificador> <endLine>
creaAsigVarInt ::= <tipoInt> <separador> <identificador> <igual> <operacionInt> <endLine>
variablesInt ::= <varInt> | <creaVarInt> | <creaAsigVarInt>

valoresFloat ::= <litFloat> | <identificador> | <llamarFunc>
operacionFloat ::= <valoresFloat>
operacionFloat ::= <sigAbreParent> <valoresFloat> <operadoresAritme> <valoresFloat> <sigCierraParent> | <valoresFloat>
<operadoresAritme> <valoresFloat>
varFloat ::= <identificador> <igual> <operacionFloat> <endLine>
creaVarFloat ::= <tipoFloat> <separador> <identificador> <endLine>
creaAsigVarFloat ::= <tipoFloat> <separador> <identificador> <igual> <operacionFloat> <endLine>
variablesFloat ::= <varFloat> | <creaVarFloat> | <creaAsigVarFloat>

valoresBool ::= <litBool> | <identificador> | <llamarFunc> | <operacionRelaInt> | <operacionRelaFloat> |
<operacionLogicas>
varBool ::= <identificador> <igual> <valoresBool> <endLine>
creaVarBool ::= <tipoBool> <separador> <identificador> <endLine>
creaAsigVarBool ::= <tipoBool> <separador> <identificador> <igual> (<signoNegacion>)? <valoresBool> <endLine>
variablesBool ::= <varBool> | <creaVarBool> | <creaAsigVarBool>

valoresChar ::= <litChar> | <identificador>
varChar ::= <identificador> <igual> <valoresChar> <endLine>
creaVarChar ::= <tipoChar> <separador> <identificador> <endLine>
creaAsigVarChar ::= <tipoChar> <separador> <identificador> <igual> <valoresChar> <endLine>
variablesChar ::= <varChar> | <creaVarChar> | <creaAsigVarChar>

valoresString ::= <litString> | <identificador>

varString ::= <identificador> <igual> <valoresString> <endLine>
creaVarString ::= <tipoString> <separador> <identificador> <endLine>
creaAsigVarString ::= <tipoString> <separador> <identificador> <igual> <valoresString> <endLine>
variablesString ::= <varString | <creaVarString> | <creaAsigVarString>

variables ::= <variablesInt> | <variablesFloat> | <variablesBool> | <variablesChar> | <variablesString> | <operacionUnaria>

funcLeerInt ::= <reservLeer> <sigAbreParent> <identificador> <sigCierraParent> <endLine>
funcLeerFloat ::= <reservLeer> <sigAbreParent> <identificador> <sigCierraParent> <endLine>
funcLeer ::= <funcLeerInt> | <funcLeerFloat>

funcEscribirInt ::= <reservEscribir> <sigAbreParent> <LitInt> <sigCierraParent> <endLine>
funcEscribirFloat ::= <reservEscribir> <sigAbreParent> <LitFloat> <sigCierraParent> <endLine>
funcEscribirString ::= <reservEscribir> <sigAbreParent> <litString> <sigCierraParent> <endLine>
funcEscribirId ::= <reservEscribir> <sigAbreParent> <identificador> <sigCierraParent> <endLine>
funcEscribir ::= <funcEscribirInt> | <funcEscribirFloat> | <funcEscribirString> | <funcEscribirId>

operacionBool ::= (<litBool> | <identificador>) (<signoIgual> | <signoDiferente>) (<litBool> | <identificador>)

operandosRelaInt ::= <valoresInt> | <operacionInt> | <operaUnaria>
operandosRelaInt ::= <sigAbreParent> <operandosRelaInt> <sigCierraParent>
operacionRelaInt ::= <operacionBool> | <operandosRelaInt> <operadoresRela> <operandosRelaInt>
operacionRelaInt ::= <sigAbreParent> <operacionRelaInt> <sigCierraParent>

operandosRelaFloat ::= <valoresFloat> | <operacionFloat> | <operaUnaria>
operandosRelaFloat ::= <sigAbreParent> <operandosRelaFloat> <sigCierraParent>
operacionRelaFloat ::= <operacionBool> | <operandosRelaFloat> <operadoresRela> <operandosRelaFloat>
operacionRelaFloat ::= <sigAbreParent> <operacionRelaFloat> <sigCierraParent>

operandosLogicos ::= <operacionRelaInt> | <operacionRelaFloat> | <litBool> | <identificador> | <operacionLogicas>
operacionLogicas ::= <operandosLogicos> <operadoresLogicos> <operandosLogicos>
operacionLogicas ::= <signoNegacion> <operandosLogicos>
operacionLogicas ::= <sigAbreParent> <operacionLogicas> <sigCierraParent>
condiciones ::= <sigAbreParent> <operacionLogicas> <sigCierraParent>

sentencia ::= <variables> | <funcLeer> | <funcEscribir> |
estructControl ::= <estructIf> | <estructSwitch> | <estructForRange> | <estructWhile>
valoresBloqueCodigo ::= <sentencia> | <estructControl> | <sentBreak> | <sentReturn> | <sentLlamarFunc>
bloqueCodigo ::= <valoresBloqueCodigo>*

estructElse ::= <reservElse> <sigAbreLlaves> <bloqueCodigo> <sigCierraLlaves>
estructIf ::= <reservIf> <condiciones> <sigAbreLlaves> <bloqueCodigo> <sigCierraLlaves> <estructElse>?

paramForRangeUno ::= <litInt>
paramForRangeDos ::= <litInt> <endLine> <litInt>
paramForRangeTres ::= <litInt> <endLine> <litInt> <endLine> <litInt>
paramForRange ::= <sigAbreParent> (<paramForRangeUno> | <paramForRangeDos> | <paramForRangeTres>)
<sigCierraParent>
estructForRange ::= <reservForRange> <paramForRange> <sigAbreLlaves> <bloqueCodigo> <sigCierraLlaves>

estructWhile ::= <reservWhile> <condiciones> <sigAbreLlaves> <bloqueCodigo> <sigCierraLlaves> <estructElse>

literal ::= <litInt> | <litFloat> | <litChar> | <litString>
case ::= <reservCase> <separador> <literal> <dosPuntos> <bloqueCodigo> <sentBreak>
default ::= <reservDefault> <dosPuntos> <bloqueCodigo> <sentBreak>

estructSwitch ::= <reservSwitch> <sigAbreParent> <identificador> <sigCierraParent> <sigAbreLlaves> (<case>)+ <default>
<sigCierraLlaves>

tiposRetornoFunc ::= <tipoInt> | <tipoFloat> | <tipoBool>
tiposParam ::= <tiposRetornoFunc> | <tipoChar>
paramFunciones ::= <sigAbreParent> (<tiposParam> <identificador>(<coma> <tiposParam> <identificador>)*)?
<sigCierraParent>
funcMain ::= <tipoInt> <reservMain> <paramFunciones> <sigAbreLlaves> <bloqueCodigo> <sigCierraLlaves>

creaFunc ::= <tiposRetornoFunc> <identificador> <paramFunciones> <sigAbreLlaves> <bloqueCodigo> <sigCierraLlaves>

paramLlamarFunc ::= <litInt> | <litFloat> | <litChar> | <litBool> | <identificador>
llamarFunc ::= <identificador> <sigAbreParent>(<paramLlamarFunc>(<coma> <paramLlamarFunc>)*)? <sigCierraParent>
sentLlamarFunc ::= <llamarFunc> <endLine>

globalVar ::= <creaVarInt> | <creaAsigVarInt> | <creaVarFloat> | <creaAsigVarFloat> | <creaVarBool> | <creaAsigVarBool> |
<creaVarChar> | <creaAsigVarChar> | <creaVarString> | <creaAsigVarString>
valoresPrograma ::= <creaFunc> | <globalVar> | <comentarios>
programa ::= <valoresPrograma>* <funcMain> <valoresPrograma>*
inicio ::= <programa>

Análisis de resultados

Lecciones aprendidas

Gracias a la realización de esta tarea, los responsables pusieron en práctica su conocimiento obteniendo con respecto a los temas de: expresiones regulares, terminales y no terminales. Así mismo se perfeccionaron habilidades de trabajo en equipo como: comunicación asertiva, trabajar de forma ordenada, la mayoría de los avances se realizaron de forma sincrónica para facilitar la comunicación, ya que muchas partes del trabajo dependen de otras.

Objetivos alcanzados

- Todos los objetivos solicitados en la tarea fueron desarrollados (del inciso a hasta el q), se realizaron con éxito.

Objetivos no logrados

- Todos los objetivos solicitados fueron alcanzados

Bitácora

Seguidamente, se muestra la bitácora recaudada del archivo compartido en la plataforma de GitHub de los integrantes: Froylan Lara (Udue11) y Alina Bonilla (Alina-bonilla).