A picture containing text

Description automatically generated

17 de marzo de 2023

Jean HUNT, PRISCILLA cHACÓN

COMPILADORES E INTERPRETES

Ingeniería en computación

Tarea 1

GRAMATICA BNF

Tabla de contenidos

[***1.*** ***Descripción del problema*** 1](#_Toc129881726)

[***2.*** ***Diseño del programa*** 1](#_Toc129881727)

[**2.1. *Terminales*** 1](#_Toc129881728)

[**2.2. *No terminales*** 2](#_Toc129881729)

[***2.3.*** ***Símbolo inicial*** 2](#_Toc129881730)

[***2.4.*** ***Producciones*** 2](#_Toc129881731)

[***3.*** ***Análisis de resultados*** 5](#_Toc129881732)

Indice de tablas

[Tabla 1:Lista de terminales 1](#_Toc129881733)

[Tabla 2:Lista de no terminales 2](#_Toc129881734)

[Tabla 3:Objetivos 5](#_Toc129881735)

1. ***Descripción del problema***

Un grupo de desarrolladores desea crear un nuevo lenguaje imperativo, ligero, 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.

**Nota**: La descripción anterior fue extraída de la especificación de la tarea.

Se deberá de desarrollar una gramática libre de contexto que permita gran cantidad de funcionalidades comunes en lenguajes de programación imperativos. Entre están funcionalidades están: creación de funciones, asignación de tipos, evaluación de expresiones lógicas y aritméticas, creación de variables, bloques de código, entre otras.

1. ***Diseño del programa***

A continuación, se detallarán los diferentes componentes de la gramática tales como la lista de terminales, no terminales, el símbolo inicial y las distintas producciones:

Tabla 1:Lista de terminales

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2.1. *Terminales*** | | | | |
| "<=" | "main" | "elif" | "false" | identificador |
| ">=" | "int" | "while" | "++" | intLiteral |
| "==" | "float" | "do" | "--" | floatLiteral |
| "!=" | "string" | "for" | "^" | stringLiteral |
| "(" | "char" | "return" | "#" | charLiteral |
| ")" | "bool" | "break" | "'" | boolLiteral |
| "{" | "function" | "leer" | """" |  |
| "}" | "array" | "escribir" | "@" |  |
| "," | "if" | "not" | "/\_" |  |
| ";" | "else" | "true" | "\_/" |  |

Tabla 2:Lista de no terminales

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2.2. *No terminales*** | | | | |
| programa | opSuma | if | sentencia | lectura |
| numero | opMult | elif | bloque | numerico |
| letra | operRelacional | doWhile | parametros | escritura |
| operador | expresionRelacional | for | declaraFuncion | comentario |
| identificador | operadorUnario | estructuraControl | tipo |  |
| operando | operadorLogico | return | declaraArray |  |
| expresion | expresionLogica | break | declaraArrayAux |  |
| termino | terminoLogico | argumentos | declaraVar |  |
| factor | array | llamaFuncion | getValorArray |  |
| valor | while | asignacion | setValorArray |  |

* 1. ***Símbolo inicial***

El símbolo inicial es el no terminal ´**programa**´

* 1. ***Producciones***

Las distintas producciones que componen la gramática se encuentran a continuación en formato consola para facilitar su lectura, además, se adjunta un archivo txt con las mismas y una serie de comentarios relacionadas a estas.

programa ::= "main" "("")" bloque

numero ::= [0-9]+

letra ::= [a-zA-Z]

operador ::= "+" | "-" | "\*" | "/" | "<" | ">" | "<=" | ">=" | "==" | "!="

identificador ::= letra (letra | numero | "")\*

operando ::= identificador

    | numero

    | llamaFuncion

expresion ::= termino (operadorSum termino)\*

termino ::= factor (operadorMult factor)\*

factor ::= opUnario? valor

valor ::= intLiteral

    | floatLiteral

    | stringLiteral

    | charLiteral

    | boolLiteral

    | identificador array?

    | llamaFuncion

opSuma ::= "+" | "-"

opMult ::= "\*" | "\*\*" | "/" | "~"

opRelacional ::= "<" | "<=" | ">" | ">=" | "==" boolLiteral? | "!="  boolLiteral?

opUnario ::= "-" | "++" | "--"

opLogico ::= "^" | "#" | "!" | "not"

expresionRelacional ::= numerico opRelacional numerico

expresionLogica ::= terminoLogico opLogico terminoLogico | terminoLogico

terminoLogico  ::= boolLiteral | "("expresionLogica")"

intLiteral ::= numero

floatLiteral ::= numero "." numero

stringLiteral ::= """letra\*"""

charLiteral ::= "'" letra "'"

boolLiteral ::= "true" | "false"

array ::= expresion ("," expresion)\*

if ::= "if" "("expresion")" bloque ["else" if | "else" sentencia]

elif ::= "elif" "("expresion")" bloque ["else" if | "else" sentencia]

while ::= "while" "("expresion")" bloque

doWhile ::= "do" bloque "while" "("expresion")"

for ::= "for" "(" declaraVar expresion "$" expresion ")" bloque

estructuraControl ::=  if

    | while

    | doWhile

    | for

return ::= "return" expresion

break ::= "break"

argumentos ::= expresion {"," expresion}

llamaFuncion ::= identificador "(" [argumentos] ")" "$"

asignacion ::= identificador "=" expresion "$"

sentencia ::= asignacion

    | llamaFuncion "$"

    | estructuraControl

    | bloque

    | expresion "$"

    | declaraVar "$"

    | return "$"

    | break "$"

bloque ::= "{" {sentencia} "}"

parametros ::= tipo identificador {"," tipo identificador}

declaraFuncion ::= "function" tipo identificador "("[parametros]")" bloque

tipo ::= "int"

    | "float"

    | "string"

    | "char"

    | "array"

    | "bool"

declaraArray ::= ("int"|"char") identificador declaraArrayAux "$"

declaraArrayAux ::= "[" numero "]" | "["numero"]" declaraArrayAux

declaraVar ::= tipo identificador declaraArrayAux? "=" expresion "$"

getValorArray ::= identificador declaraArrayAux "$"?

setValorArray ::= getValorArray "=" expresion

lectura ::= leer "(" numerico expresion")" "$"

numerico ::= "int" | "float"

escritura ::= escribir "(" expresion ")" "$"

comentario ::= "@" char\* | "/\_" char\* "\_/"

1. ***Análisis de resultados***

En la siguiente tabla se muestra una lista de objetivos para los cuales se evaluará si fueron cumplidos o no, en caso de no cumplirse también se indicará la razón del porqué:

Tabla 3:Objetivos

|  |  |  |
| --- | --- | --- |
| ***Objetivo*** | ***Alcanzado*** | ***Razón*** |
| Realizar una investigación sobre la redacción y estructura de las gramáticas libres de contexto | Si |  |
| Cumplir con las reglas y funcionalidades solicitadas en la especificación de la tarea | Si |  |
| Mantener uniformidad a la hora de nombrar las producciones | Si |  |
| Obtener una legibilidad simple en la gramática | Si |  |
| Documentar de manera interna la gramática redactada | Si |  |
| Documentar de manera externa la tarea realizada | Si |  |

En el siguiente enlace se puede encontrar el repositorio en GitHub que contiene la tarea: <https://github.com/JPHuntV/T1-Compiladores-IS-2023.git>