Nery Oswaldo Jiménez Contreras Carné: 201700381
Organización de Lenguajes y Compiladores 1 Sección: C

GRAMÁTICA TypeWise

El sistema TypeWise es un emulador de un pseudo lenguaje de programación similar a javascript. Nace para ser una herramienta que ayude a los futuros programadores a familiarizarse con la lógica y sintaxis de programación.

Para poder ejecutar el código que se escriba dentro del sistema, se debe llamar a una función *main*, la cuál será la encargada de instanciar las demás funciones y métodos durante la ejecución.

El sistema TypeWise cuenta con las siguientes expresiones regulares:

Expresiones Regulares:

Comentario de una línea: "//".*
Comentario de varias líneas: [/][*][^*]*[*]+([^/*][^*]*[*]+)*[/]
Cadena: \"((\\\")|[^\"\n])*\"
Caracter: \'((\\\\)|(\\n)|(\\t)|(\\\")|(\\\')|[^\'\n])\'
Decimal: [0-9]+"."[0-9]+\b
Entero: [0-9]+\b
Identificador: ([a-zA-Z_])[a-zA-Z0-9_ñÑ]*

También tenemos las siguientes palabras reservadas:

Palabras reservadas

- int
- double
- boolean
- char
- string
- new
- list
- add
- if
- else
- switch
- case
- default
- break
- while
- for
- do
- continue
- return
- void
- print
- toLower
- toUpper
- length
- truncate
- round
- typeof
- toString
- toCharArray
- main

Además de los siguientes símbolos:

Símbolos

- \n
- \\
- \"
- \t
- \'
- + (suma)
- - (resta)
- * (multiplicación)
- / (división)
- ^ (potencia)
- % (módulo)
- == (igual)
- != (distinto)
- < (menor que)
- <= (menor o igual)
- > (mayor que)
- >= (mayor o igual)
- ?
- || (or)
- && (and)
- ! (not)
- : (dos puntos)
- ; (puntos y coma)
- ((parentesis abre)
-) (parentesis cierra)
- { (llave abre)
- } (llave cierra)
- = (asignación)
- ' (comilla simple)
- . (punto)
- [(corchete abre)
-] (corchete cierra)
- , (coma

Definimos la precedencia de operaciones
Izquierda UMINUS (negativos)
Derecha UCAST (casteos)
Izquierda '++', '--'
Izquierda '^'
Izquierda '*'. '/'. '%'
Izquierda '+', '-'
Izquierda '==', '!=', '>=', '<=', '>', '<'
Derecha '!'
Izquierda '&&'
Izquierda '||'

Inicio de la gramática

producción inicial: <init>

<init> -> globalBody **(sentencias globales)**

Sentencias globales

globalBody -> globalBosy global **(listado de sentencias)**
| global

Listado de sentencias
global -> statment **(definición de variables)**
| method **(definición de métodos)**
| assigment **(asignación de variables)**
| main **(función principal)**

Definición de métodos

method -> tipo identificador () { localBody **(sentencias locales)** }
| tipo identificador (parametros) localBody }
| void identificador ( ) { localBody }
| void identificador (parametros) { localBody }

Llamada de métodos

callmethod -> identificador (atributos)
| identificador ( )

Atributos

attributes -> attributes , expression
| expression

Parametros

parameters -> parameters , tipo identificador
| tipo identificador

Sentencias locales

localBody -> localBody local
| local

Listado de sentencias

local -> statment
| ciclos
| assigment
| callmethod;
| control
| print;

Declaración de variables

statment -> identificador = expression ;
| declararVector
| asignaciónLista
| agregarLista

Declarar Vector

declararVector -> tipo [ ] identificador = new tipo [expression]
| tipo [ ] identificador = { atributos }
| tipo [ ] identificador = toChar

Asignar Vector

asignarVector -> identificador [expression] = expression ;

Declarar Lista

declararLista -> list < tipo > identificador = new list < tipo >
| list < tipo > identificador = toChar

Agregar Lista

agregarLista -> identificador . add (expression) ;

Valores de datos

valorDato -> decimal
| entero
| falso
| verdadero
| caracter
| cadena
| identificador

Tipos de datos

tipoDato -> int
| double
| boolean
| char
| string

Incremento y decremento

incremento -> identificador ++
| identificador - -

Ciclo for

cicloFor -> for (asignacion ; expression ; modificacion) { localBody }

Ciclo while

cicloWhile -> while (expression) { localBody }

Ciclo do while

doWhile -> do { localBody } while (expression) ;

Condición switch

condicionSwitch -> switch (expression) { listaCasos }

Lista de casos

listaCasos -> case expression : localBody
| case expression:
| default : localBody
| default :

condición if

condicionIf -> if (expression) { localBody } listaElse
| if (expression) { localBody } listaElse

Lista else

listaElse -> else condicionIf
| else { localBody }
| else { }

Funciones Nativas

funcionNativa -> casteo
| toCharArray (expression)
| length (expression)
| toLower (expression)
| toUpper (expression)
| truncate (expression)
| typeOf (expression)
| toString (expression)

Lista de Ciclos

listaCiclos -> while
| if
| ifelse
| for
| switch
| doWhile
| ternario

Sentencias de control

sentenciaControl -> break;
| continue;
| return;
| return expression;

Impresión

impresion -> print (expression) ;
| print ( ) ;

Casteos

casteo -> (tipo) expression

Ternario

ternario -> expression ? expression : expression

Lista de expresiones

```
expression ->  expression  + expression
|              expression  - expression
|              expression  * expression
|              expression  / expression
|              expression  ^ expression
|              expression  % expression
|              - expression
|              ( expression )
|              expression  == expression
|              expression  != expression
|              expression  <= expression
|              expression  >= expression
|              expression  < expression
|              expression  > expression
|              expression  || expression
|              expression  && expression
|              ! expression
|              ternario
|              incremento
|              estructuras
|              funciones
|              llamadasFunciones
|              tipoValor
|              casteos
```

Función principal

```
funcionMain ->          main llamadaMetodo ;
```

Llamada Funciones

```
llamadaFunciones ->  identificador ( atributos )
|                    identificador ( )
```

Estructuras de datos

```
estructuras ->  identificador [ expression ]
|               identificador [ [ expression ] ]
```