Organización de Lenguajes y Compiladores 2
Gramática C3D

JAVIER ALBERTO CABRERA PUENTE

Introducción

A continuación se encuentran todas las especificaciones acerca de la gramática que define el lenguaje de C3D que deriva de "J#". Las especificaciones incluyen los detalles pertenecientes al análisis léxico y sintáctico, es decir, los símbolos terminales, no terminales y sus reglas de definición.

Análisis Léxico y Símbolos Terminales

El lenguaje cuenta con un total de 39 símbolos terminales, definidos a continuación según su expresión regular, las expresiones regulares siguen las reglas de regex de javascript.

No.	Terminal	Regex
1	espacios en blanco	\s+
2	comentarios simples	## . *
3	comentarios compuestos	[#][*][^*]*[*]+([^#*][^*]*[*]+)*[#]
4	temp	"t"[0-9]+
5	label	"L"[0-9]+
6	varKW	"var"
7	beginKW	"begin"
8	ifKW	"if"

9	stackKW	“stack”
10	endKW	“end”
11	heapKW	“heap”
12	callKW	“call”
13	gotoKW	“goto”
14	printKW	“print”
15	procKW	“proc”
16	p	“p”
17	h	“h”
18	equals	“==”
19	greaterEquals	“>=”
20	asignment	“=”
21	notEquals	“<>”
22	lessEquals	“<=“
23	comma	“,”
24	semicolon	“;”
24	colon	“:”
25	leftP	“(”
26	rightP	“)”
27	leftS	“[”
28	rightS	“]”
29	plusOp	“+”
30	minusOp	“-”
31	timesOp	“*”
32	divOp	“/”
33	modOp	“%”

34	greaterThan	“>”
35	lessThan	“<”
36	id	([a-zA-Z0-9])[a-zA-Z0-9_]*
37	float	[0-9]+”.”[0-9]\b
38	int	[0-9]+\b
39	EOF	<<EOF>>
40	printChar	”%c”
41	printDouble	“%d”
42	printInt	“%i”

Precedencia Utilizada

No se utilizó precedencia, pues el lenguaje no lo requería.

Símbolos No Terminales

En el presente documento los símbolos no terminales se representan con nombres cuyos caracteres son todos mayúsculas, a continuación se encuentran numerados, junto con una explicación corta de su función en la gramática del lenguaje:

No.	No Terminal	Descripción
1	**INIT**	Utilizado únicamente por convención para tener un acceso limpio a la raíz del árbol de análisis sintáctico.
2	**TDC**	Encapsula el encabezado y el cuerpo del código.
3	**HEADER**	Encapsula las instrucciones de encabezado.

4	**BODY**	Encapsula las instrucciones del cuerpo del lenguaje.
5	**TEMPS_DECL**	Define la forma de declarar variables temporales.
6	**P_DECL**	Define la forma de declarar a los punteros especiales p y h.
7	**HEAP_DECL**	Define la regla para crear el heap.
8	**STACK_DECL**	Define la regla para crear el stack
9	**TEMPS_LIST**	Define una lista de temporales
10	**PARAMETERS_LIST**	Crea una lista de parámetros
11	**PARAMS_VALUE**	Encapsula todas las distintas formas que un parámetro puede tomar.
12	**INSTRUCTIONS**	Encapsula una lista de instrucciones que puede ser utilizadas en el cuerpo del lenguaje.
13	**INSTRUCTION**	Encapsula todas los diferentes no terminales que pueden ser calificados como instrucciones.
14	**ASIGNMENT**	Define la forma de realizar una asignacion en el lenguaje
15	**DESTINATION**	Define la forma de agregar una etiqueta
16	**JUMP**	Define un salto sin condición a una etiqueta
17	**CONDITIONAL_JUMP**	Define un salto que únicamente se realiza si se cumple una condición
18	**PROCEDURE**	Define la estructura de un procedimiento
19	**CALL**	Define la manera de llamar un procedimiento
20	**PRINT**	Define la manera de realizar una impresion
21	**A_OPT**	Define las opciones que pueden ser asignadas
22	**EXPRESSION**	Define las formas que puede tomar una asignación
23	**OPERATOR**	Define los operadores permitidos en una expresión
24	**E_OPT**	Define los valores que pueden ser operados
25	**CONDITION**	Define una condición para un salto

26	**C_OPT**	Define los valores que pueden ser comparados
27	**COND_OP**	Define los operadores que pueden ser parte de una condición
28	**PRINT_OPT**	Define las formas de imprimir
29	**PRINT_VALUES**	Define los valores que pueden ser impresos

Definición de la Gramática

A continuación se definen las reglas gramaticales de cada símbolo no terminal, para fines de formato y claridad se define una tabla por cada símbolo no terminal con sus reglas y acciones semánticas para poder formar el árbol de análisis sintáctico. Para cada símbolo la variable de retorno será denominada retorno, también se utiliza pseudocódigo para definir los métodos de programación utilizados. Los símbolos no terminales se encuentran en mayúsculas y negritas, mientras que los terminales se encuentran en minúsculas, sin negrita. La regla sintáctica se encuentra a la izquierda, las reglas semánticas se encuentran a la derecha.

INIT :=

SCRIPT eof

TDC :=

HEADER BODY

HEADER :=

TEMPS_DECL P_DECL HEAP_DECL STACK_DCL

TEPS_DECL :=

varKW **TEMPS_LIST** semicolon

TEMPS_LIST :=

TEMPS_LIST comma temp
temp

P_DECL :=

varKW p comma h semicolon

HEAP_DECL :=

varKW heapKW leftS rightS semicolon

STACK_DECL :=

varKW stackKW leftS rightS semicolon

BODY :=

INSTRUCTIONS

INSTRUCTIONS :=

INSTRUCTIONS INSTRUCTION
INSTRUCTION

INSTRUCTION :=

ASIGNMENT
DESTINATION
JUMP
CONDITIONAL_JUMP
PROCEDURE
CALL
PRINT

ASIGNMENT :=

A_OPT asignment **EXPRESSION** semicolon

A_OPT:=

temp
p
h
stackKW leftS temp rightS
stackKW leftS int rightS
stackKW leftS p rightS
heapKW leftS temp rightS
heapKW leftS int rightS
heapKW leftS h rightS

EXPRESSION :=

minusOp **E_OPT**
E_OPT OPERATOR E_OPT
E_OPT
stackKW leftS temp rightS
stackKW leftS int rightS
stackKW leftS p rightS
heapKW leftS temp rightS

heapKW leftS int rightS
heapKW leftS h rightS

E_OPT :=

temp
int
float
p
h

OPERATOR :=

plusOp
minusOp
timesOp
divOp
modOp

DESTINATION :=

label colon

JUMP :=

gotoKW label semicolon

CONDITIONAL_JUMP :=

ifKW leftP **CONDITION** rightP gotoKW label semicolon

CONDITION :=

C_OPT COND_OP C_OPT

C_OPT :=

int
float
temp

COND_OP :=

notEquals
equals
lessThan

lessEquals
greaterThan
greaterEquals

PROCEDURE :=

procKW id beginKW **INSTRUCTIONS** endKW

CALL :=

call id semicolon

PRINT :=

printKW left **PRINT_OPT** comma **PRINT_VALUES** rightP semicolon

PRINT_OPT :=

printChar
printDouble
printInt

PRINT_VALUES :=

temp
int
float
h
p