Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Proyecto 2: MINOR-C

Manual Usuario

Organización de Lenguajes y Compiladores II

M.Sc Luis Fernando Espino Barrios

Aux. Pavel Vásquez

Aux. Juan Carlos Maeda

Jorge Daniel Juarez Aldana

201807022

VALORES LEXICOS:

Token	Nombre	Patron
1	INT	int
2	DOUBLE	double
3	CHAR	char
4	GOTO	goto
5	MAIN	main
6	IF	if
7	ELSE	else
8	VOID	void
9	WHILE	while
10	BREAK	break
11	CONTINUE	continue
12	DO	do
13	FOR	for
14	STRUCT	struct
15	SWITCH	switch
16	CASE	case
17	DEFAULT	default
18	RETURN	return
19	ES_IGUAL	==
20	NO_IGUAL	!=
21	SUMA_IGUAL	+=
22	RESTA_IGUAL	-=
23	MULT_IGUAL	*=
24	DIV_IGUAL	/=
25	MOD_IGUAL	%=
26	SHIFT_IZQ_IGUAL	<<=
27	SHIFT_DER_IGUAL	>>=
28	AND_BIT_IGUAL	&=
29	OR_BIT_IGUAL	|=
30	XOR_BIT_IGUAL	^=
31	MAYOR_IGUAL	>=
32	MENOR_IGUAL	<=
33	INTERROGACION	?
34	SHIFT_DER	>>
35	SHIFT_IZQ	<<
36	MAYOR	>
37	MENOR	<
38	XOR_BIT	^
39	OR	||
40	AND	&&
41	NOT_BIT	~
42	AND_BIT	&
43	OR_BIT	|
44	NOT	!
45	SUMA	+
46	SUMA_SUMA	++
47	RESTA	-
48	RESTA_RESTA	--
49	MULT	*
50	DIV	/
51	MOD	%
52	PAR_IZQ	(
53	PAR_DER	)
54	COR_IZQ	[
55	COR_DER	]
56	BRA_IZQ	{
57	BRA_DER	}
58	IGUAL	=
59	DOS_PUNTOS	:
60	PUNTO_COMA	;
61	COMA	,
62	PUNTO	.
63	ID	[a-zA-Z][a-zA-Z0-9_]*
64	COMENTARIO	//.*\n
65	COMENTARIO_MULTILINEA	/*(.|\n)**/
66	ENTERO	\d+
67	CARACTER	['].[']
68	CADENA	\"([^\"]|\\.")*\"
69	DECIMAL	\d+\.\d+

GRAMATICA ASCENDENTE:

init → program

init → ε

program → main

program → declars main

declars → declars declar

declars → declar

declar → primitive_type function

declar → VOID function

declar → declaration

declar → assignation

declar → struct

main → MAIN PAR_IZQ PAR_DER function_block

function → function_type ID param function_block

param → PAR_IZQ PAR_DER

param → PAR_IZQ parameter_list PAR_DER

parameter_list → parameter_list COMA parameter

parameter_list → parameter

parameter → primitive_type ID

primitive_type → INT

primitive_type → DOUBLE

primitive_type → CHAR

primitive_type → STRUCT ID

struct → STRUCT ID struct_block

struct_block → BRA_IZQ declarations BRA_DER

function_block → BRA_IZQ instructions BRA_DER

function_block → BRA_IZQ BRA_DER

instructions → instructions instruction

instructions → instruction

instruction → declaration PUNTO_COMA

instruction → assignation PUNTO_COMA

instruction → for

instruction → while

instruction → do PUNTO_COMA

instruction → if

instruction → switch

instruction → label

instruction → goto PUNTO_COMA

instruction → break PUNTO_COMA

instruction → continue PUNTO_COMA

instruction → return PUNTO_COMA

instruction → expression PUNTO_COMA

instruction → PUNTO_COMA

declarations → declarations declaration PUNTO_COMA

declarations → declaration PUNTO_COMA

declaration → primitive_type declaration_list

declaration → ID declaration_list

declaration_list → declaration_list COMA ID IGUAL expression

declaration_list → declaration_list COMA ID brackets IGUAL BRA_IZQ expression_list BRA_DER

declaration_list → declaration_list COMA ID

declaration_list → declaration_list COMA ID brackets

declaration_list → declaration_list COMA ID COR_IZQ COR_DER IGUAL CADENA

declaration_list → ID IGUAL expression

declaration_list → ID brackets IGUAL BRA_IZQ expression_list BRA_DER

declaration_list → ID

declaration_list → ID brackets

declaration_list → ID COR_IZQ COR_DER IGUAL CADENA

identifier → ID

identifier → ID bracket

brackets → bracket

brackets → COR_IZQ COR_DER

bracket → bracket COR_IZQ expression COR_DER

bracket → COR_IZQ expression COR_DER

assignation → identifier equals expression

assignation → identifier PUNTO ID equals expression

equals → IGUAL

equals → SUMA_IGUAL

equals → RESTA_IGUAL

equals → MULT_IGUAL

equals → DIV_IGUAL

equals → MOD_IGUAL

equals → SHIFT_IZQ_IGUAL

equals → SHIFT_DER_IGUAL

equals → AND_BIT_IGUAL

equals → XOR_BIT_IGUAL

equals → OR_BIT_IGUAL

if → IF PAR_IZQ expression PAR_DER function_block

if → IF PAR_IZQ expression PAR_DER function_block ELSE if

if → IF PAR_IZQ expression PAR_DER function_block ELSE function_block

while → WHILE PAR_IZQ expression PAR_IZQ function_block

do → DO function_block WHILE PAR_IZQ expression PAR_DER

for → FOR PAR_IZQ for_declaration PUNTO_COMA expression PUNTO_COMA step PAR_DER function_block

for_declaration → declaration

for_declaration → assignation

step → assignation

step → increase

step → decrease

switch → SWITCH PAR_IZQ expression PAR_DER BRA_IZQ cases BRA_DER

cases → cases case

cases → cases default

cases → case

case → CASE expression DOS_PUNTOS instructions

default → DEFAULT DOS_PUNTOS instructions

label → ID DOS_PUNTOS

goto → GOTO ID

break → BREAK

continue → CONTINUE

return → RETURN expression

return → RETURN

function_call → ID param_val

param_val → PAR_IZQ PAR_DER

param_val → PAR_IZQ expression_list PAR_DER

expression_list → expression_list COMA expression

expression_list → expression

expression → expression SUMA expression

expression → expression RESTA expression

expression → expression MULT expression

expression → expression DIV expression

expression → expression MOD expression

expression → expression AND expression

expression → expression OR expression

expression → expression ES_IGUAL expression

expression → expression NO_IGUAL expression

expression → expression MENOR_IGUAL expression

expression → expression MAYOR_IGUAL expression

expression → expression MENOR expression

expression → expression MAYOR expression

expression → expression AND_BIT expression

expression → expression OR_BIT expression

expression → expression XOR_BIT expression

expression → expression SHIFT_IZQ expression

expression → expression SHIFT_DER expression

expression → expression INTERROGACION expression DOS_PUNTOS expression

expression → RESTA expression

expression → AND_BIT expression

expression → NOT expression

expression → NOT_BIT expression

expression → PAR_IZQ expression PAR_DER

expression → increase

expression → decrease

expression → ENTERO

expression → DECIMAL

expression → CARACTER

expression → CADENA

expression → function_call

expression → conversion

expression → identifier

expression → identifier PUNTO ID

conversion → PAR_IZQ INT PAR_DER expression

conversion → PAR_IZQ DOUBLE PAR_DER expression

conversion → PAR_IZQ CHAR PAR_DER expression

increase → SUMA_SUMA expression

increase → expression SUMA_SUMA

decrease → RESTA_RESTA expression

decrease → expression RESTA_RESTA

ESQUEMA DE TRADUCCION:

El codigo de 3 direcciones fue generado a partir de la gramática mediante el uso de marcadores, por medio de los cuales se fue capaz de generar instrucciones para el contro de los bloques básicos de ejecución. Ademas de eso cabe resaltar que se hizo uso del método de backpatch para la traducción de saltos condicionales. Las optimizaciones aplicadas al codigo fueron mayormente aritméticas, aunque por medio de optimizaciones de flujo de control fueron eliminadas algunas partes del código muerto.