Entrada	Salida			
## Entrada sintacticamente valida para	{	;	*	{
## los analizadores de Tiny	<type></type>	<pre><pre><pre><pre>u</pre></pre></pre></pre>	x ;	} <else></else>
	^	(	<u>@</u>	{
cype  ^^struct{	<struct></struct>	<int></int>	х	}
int x,	{	V	= 5	; <while></while>
bool y,	X	<int></int>	}	5
real z,	,	[	}	{
struct {	<bool></bool>	98	<else></else>	}
^int[20][30] x,	у	] z	{	} <eof></eof>
int y	<real></real>	)	у	(2017
} k	z	{	(	
}[20] tRaro;	, <struct></struct>	} &&	) {	
ype ^^ tRaro[5][10] tRarito; ^^tRarito[10][10] x;	{	@	}	
roc subprog(int x, ^^tRarito y, struct{int x,	^	k	&&	
tRarito y} z) {	<int></int>	c c	@ ×	
^^tRarito k;	20	^	=	
proc u(int v, int[98] z) {}	]	•	2	
&&	[ 30	X	*	
@ k.c^.x[e+e[x.c]].y^[r] % 2;	]	[ e	y ;	
call u(5, 5 - 5)	x	+	<call></call>	
&	,,,,,,	e r	y	
$f(x == 34) \{$	<int></int>	[ x	( 2	
$if(x < 56)$ {	}		*	
$(0 \times 1)^{-1}$	k	c	x	
while (x < 20) {	}		+ 80	
int x	20		,	
&&	]	у	45	
$0 \times = 2 \times x;$	tRaro	^	-	
@ x = 5	; <type></type>	[ r	x )	
}	^	]	}	
} else {	^	%	}	
proc y() {}	tRaro	2 ;	; {	
&& &&	5	<call></call>	<int></int>	
$0 \times 2 \times y;$	]	u	х	
call y(2*x + 80, 45 - x)	[ 10	5	&&	
}	10		@ ×	
;	tRarito	5	;	
	;	-	<read></read>	
nt x	^	5	x	
& 	^	j j	x	
@ x; read x.x;	tRarito	&&	;	
write x + 67;	[ 10	<if></if>	<write></write>	
delete x.x[x];	]	( x	+	
new u.v[x];	į	==	67	
nl	10	34	;	
;	] x	) {	<delete></delete>	
"una cadena!!!" + null;	ĵ,	<if></if>	¨.	
f 5 {};	<proc></proc>	(	x	
f 5 {} else {}; hile 5 {}	subprog (	×	[ ×	
niie 5 {}	<int></int>	56	ĵ	
	x	)	;	
	^	{	<new></new>	
	^	@ ×	u	
	tRarito	=	v	
	у	×	[	
	, <struct></struct>	1	x ]	
	{	;	;	
	<int></int>	<while></while>	<nl></nl>	
	x	( x	}	
	^	× <	; @	
	tRarito	20	"una	
	y	)	cadena!!!"	
	}	{	+ znulls	
	, z	<int></int>	<null></null>	
	{	&&	<if></if>	
	^	@	5	
	^ tRarito	x =	{ }	
	k	2	;	

Sirven también los casos 1 y 3 para Tiny(0). El caso 2 tiene una salida diferente, ya que un identificador sirve también como nombre de tipo:

Entrada	Salida
{     ## error: falta @     x = 5 }	{     x     ERROR_SINTACTICO