A seguinte gramática de atributos define um algoritmo de verificação de tipos para programas similares aos de Pascal:

P	→	**program** D **begin** C **end.**	{C.env := D.t; P.ck := C.ck}
D	→	**var id :** T	{D.t := S*etOf*(**id**.txt, T.t)}
D	→	D **;** D	{D_0.t := D_1.t ∪ D_2.t}
T	→	**integer**	{T.t := *inteiro* }
T	→	**char**	{T.t := *caractere* }
T	→	**bool**	{T.t := *booleano* }
T	→	**array [num] of** T	{T_0.t := *array*(**num**.val, T_1.t) }
C	→	**id** := E	{E.env := C.env; C.ck := (lookup(**id**, E.env) = E.tp) }
C	→	**if** E **then** C **else** C	{E.env := C_0.env; C_1.env := C_0.env; C_2.env := C_0.env; C_0.ck := (E.tp = *booleano*) ∧ C_1.ck ∧ C_2.ck }
C	→	**while** E **do** C	{E.env := C_0.env; C_1.env := C_0.env; C_0.ck := (E.tp = *booleano*) ∧ C_1.ck }
C	→	**read(id)**	{C.ck := *verdadeiro* }
C	→	**write(** E **)**	{E.env := C.env; C.ck := (E.tp = *booleano* ∨ E.tp = *caractere* ∨ E.tp = *inteiro*) }
E	→	**id**	{E.tp = lookup(**id**, E.env); }
E	→	**id[** E **]**	{E_1.env := E_0.env; ***if*** E_1.tp = *inteiro* ***then*** E_0.tp := lookup(**id**, E.env); ***else*** E_0.tp := *erro* }
E	→	**num**	{ E.tp := *inteiro* }
E	→	**truth-value**	{ E.tp := *booleano* }
E	→	**character**	{ E.tp := *caractere* }
E	→	**(** E **)**	{E_1.env := E_0.env; E_0.tp := E_1.tp}
E	→	E + E	{E_1.env := E_0.env; E_2.env := E_0.env; ***if*** E_1.tp = *inteiro* ∧ E_2.tp = *inteiro* ***then*** E_0.tp := *inteiro* ***else*** E_0.tp := *erro* }
E	→	E **or** E	{E_1.env := E_0.env; E_2.env := E_0.env; ***if*** E_1.tp = *booleano* ∧ E_2.tp = *booleano* ***then*** E_0.tp := *booleano* ***else*** E_0.tp := *erro* }
E	→	**not** E	{E_1.env := E_0.env; ***if*** E_1.tp = *booleano* ***then*** E_0.tp := *booleano* ***else*** E_0.tp := *erro* }

Na gramática acima são usados os seguintes atributos:

Símbolo	***Atributo***	***Observações***
P	ck	*Sintetizado*. Booleano, É verdadeiro se não existem erros de tipos no programa.
D	t	*Sintetizado*. Conjunto de pares (identificador, tipo). Os tipos permitidos são: *inteiro*, *caractere*, *booleano*, *array*(*n*, *tipo*) e *erro*.
T	t	*Sintetizado*. Um tipo, a ser usado em declarações (nunca será *erro*).
C	env	*Herdado*. Conjunto de pares (identificador, tipo). Os tipos permitidos são: *inteiro*, *caractere*, *booleano*, *array*(*n*, *tipo*) e *erro*.
C	ck	*Sintetizado*. Booleano, que é verdadeiro se e somente se não existem erros de tipos no comando.
E	env	*Herdado*. Conjunto de pares (identificador, tipo). Os tipos permitidos são: *inteiro*, *caractere*, *booleano*, *array*(*n*, *tipo*) e *erro.*
E	tp	*Sintetizado*. O tipo da expressão (pode ser *erro*).
id	txt	*Sintetizado*. O texto do identificador.
num	val	*Sintetizado*. O valor correspondente ao número.

A operacao “lookup(**id**, E.env)” devolve o tipo associado ao identificador “**id**” no ambiente “E.env”.