A seguinte gramática de atributos define um algoritmo de geracao de codigo para uma máquina de pilha simples, a partir de programas similares aos de Pascal. (A operação || indica concatenação de strings):

P → **program** D **begin** C **end.**	{ C.ah := D.as; P.cs := D.cs || C.cs ;}
D → **var id;**	{ k := install(**id**.txt); D.as := SetOf(**id**.txt, k); D.cs := "`reserve`"|| k ; }
D → D **;** D	{ D_0.as := union(D_1.as, D_2.as); D_0.cs := D_1.cs || D_2.cs }
C → **id** := E	{ E.ah := C.ah; k := lookup(**id**.txt, C.ah); C.cs := E.cs || "`push`" || k || "`attrib`";}
C → **if** E **then** C **else** C	{ E.ah := C_1.ah := C_2.ah := C_0.ah; l := newlabel(); l' := newlabel(); C_0.cs := E.cs || "`IfFalse`" || l || C_1.cs || "`goto`" || l' || l || "`:`" || C_2.cs || l' || "`:`" || "`nop`" ; }
C → **while** E **do** C	{ E.ah := C_1.ah := C_0.ah; l := newlabel(); l' := newlabel (); C_0.cs := l || "`:`" || E.cs || "`IfFalse`" || l' || C_1.cs || "`goto`" || l || l' || "`:`" || "`nop`" ; }
C → **read(id)**	{ k := lookup(**id**.txt, C.ah); C.cs := "`read`" || "`push`" || k || "`attrib`";}
C → **write(** E **)**	{ E.ah := C_0.ah; C.cs := E.cs || "`print`"}
E → **id**	{ k := lookup(**id**.txt, C.ah); E.cs := "`pushInd`" || k ;}
E → **num**	{ E.cs := "`push`" || **num**.val ;}
E → **truth-value**	{ E.cs := "`push`" || **truth-value**.val ;}
E → **character**	{ E.cs := "`push`" || **character**.val ;}
E → **(** E **)**	{ E_1.ah := E_0.ah; E_0.cs := E_1.cs ; }
E → E + E	{ E_1.ah := E_2.ah := E_0.ah; E_0.cs := E_1.cs || E_2.cs || "`add`" ;}
E → E **or** E	{ E_1.ah := E_2.ah := E_0.ah; E_0.cs := E_1.cs || E_2.cs || "`or`";}
E → **not** E	{ E_1.ah := E_0.ah; E_0.cs := E_1.cs || "`not`" ;}

Na gramática acima são usados os seguintes atributos:

Símbolo	***Atributo***	***Observações***
P	cs	*Sintetizado*. Codigo gerado para o programa.
D	as	*Sintetizado*. Conjunto de pares (identificador, endereco).
D	cs	*Sintetizado*. Codigo gerado para a declaracao.
C	ah	*Herdado*. Ambiente (identificador – endereco).
C	cs	*Sintetizado*. Codigo gerado para o comando.
E	ah	*Herdado*. Ambiente (identificador – endereco).
E	cs	*Sintetizado*. Codigo gerado para a expressao.
id	txt	*Sintetizado*. O texto do identificador.
num	val	*Sintetizado*. O valor correspondente ao número.
truth-value	val	*Sintetizado*. O valor correspondente ao número.
character	val	*Sintetizado*. O valor correspondente ao número.

As operações da máquina de pilha usada no programa gerado são:

Nome	***Argumentos***	***Operação realizada***
`push`	valor	empilha o valor
`reserve`	endereço	reserva uma palavra de memória, a partir do endereço dado.
`Attrib`	topo e sub-topo da pilha	atribui o valor no sub-topo da pilha no endereco dado no topo.
`IfFalse`	rótulo e topo da pilha	desvio de controle para o rótulo, caso o topo da pilha contenha o valor falso. Desempilha o valor do topo.
`goto`	rótulo	Desvio incondicional para o rótulo.
`nop`	-	Nenhuma operação.
`read`	-	Lê um valor da entrada e o empilha.
`print`	topo da pilha	Imprime o valor que aparece no topo da pilha. Desempilha o valor do topo.
`pushInd`	endereço	empilha o valor armazenado no endereço.
`add`	topo e sub-topo da pilha	Substitui os valores do topo e sub-topo da pilha pelo valor da sua soma.
`or`	topo e sub-topo da pilha	Substitui os valores do topo e sub-topo da pilha pelo valor da sua subtração.
`not`	topo da pilha	Substitui o valor do topo da pilha pela sua negação.

Outras operações usadas:

Operação	***Ação***
install(**id**.txt)	reserva e devolve um endereço de memória para o texto do identificador.
SetOf(x, y)	devolve o conjunto { (x, y) }.
union(X, Y)	devolve o conjunto formado pela união de X e Y.
lookup(t, X)	devolve a 2da componente do par que contém t como primeira componente, no conjunto X.
newlabel()	cria um rótulo novo, ainda nao usado.