BNF Bruna

Esta BNF foi baseada na BNF simplificada do C. As mudanças estão destacadas em vermelho.

```
<function-definition> ::= {<declaration-specifier>}* <declarator> {<declaration>}*
<compound-statement>

<declaration-specifier> ::= <type-qualifier>

<type-specifier> ::= void
                   | char
                   | short
                   | int
                   | long
                   | float
                   | double
                   | signed
                   | unsigned


<constant-expression> ::= <conditional-expression>

<conditional-expression> ::= <logical-or-expression>
                           | <logical-or-expression> ? <expression> colon <conditional-expression>

<logical-or-expression> ::= <logical-and-expression>
                          | <logical-or-expression> logic_or <logical-and-expression>

<logical-and-expression> ::= <inclusive-or-expression>
                           | <logical-and-expression> logic_and <inclusive-or-expression>

<inclusive-or-expression> ::= <exclusive-or-expression>
                            | <inclusive-or-expression> in_or <exclusive-or-expression>

<exclusive-or-expression> ::= <and-expression>
                            | <exclusive-or-expression> ex_or <and-expression>

<and-expression> ::= <equality-expression>
                   | <and-expression> and <equality-expression>

<equality-expression> ::= <relational-expression>
                        | <equality-expression> equal <relational-expression>
                        | <equality-expression> diff <relational-expression>
```

```
<additive-expression> ::= <multiplicative-expression>
              | <additive-expression> plus <multiplicative-expression>
              | <additive-expression> minus <multiplicative-expression>

<multiplicative-expression> ::= <cast-expression>
                  | <multiplicative-expression> mult <cast-expression>
                  | <multiplicative-expression> div <cast-expression>
                  | <multiplicative-expression> rest <cast-expression>

<cast-expression> ::= <unary-expression>
            | ( <type-name> ) <cast-expression>

<unary-expression> ::= <postfix-expression>
             | pp <unary-expression>
             | mm <unary-expression>
             | <unary-operator> <cast-expression>
             | sizeof <unary-expression>
             | sizeof <type-name>

<primary-expression> ::= <identifier>
              | <constant>
              | <string>
              | ( <expression> )

<constant> ::= <integer-constant>
        | <character-constant>
        | <floating-constant>
        | <enumeration-constant>

<expression> ::= <assignment-expression>
         | <expression> comma <assignment-expression>

<assignment-expression> ::= <conditional-expression>
               | <unary-expression> <assignment-operator> <assignment-expression>

<assignment-operator> ::= receive
              | mult_equal
              | div_equal
              | rest_equal
              | plus_equal
              | minus_equal
              | and_equal
              | inor_equal
              | or_equal
```

```
<unary-operator> ::= and
                 | mult
                 | plus
                 | minus
                 | not_bit
                 | not



<expression-statement> ::= {<expression>}? semicolon

<selection-statement> ::= if ( <expression> ) <statement>
                    | if ( <expression> ) <statement> else <statement>
                    | switch ( <expression> ) <statement>

<iteration-statement> ::= while ( <expression> ) <statement>
                    | do <statement> while ( <expression> ) semicolon
                    | for ( {<expression>}? semicolon {<expression>}? semicolon {<expression>}?
) <statement>
```

Caracteres especiais que foram evitados. As colunas destacadas em vermelho são os símbolos que diferem do teclado português para o americano.

pt	en	pt	en	pt	en	pt	en	pt	en	pt	en	pt	en	pt	en	pt	en
‘	`	#	#	&	&	-	-	´	[	ç	;	]	\	.	.	/	/
“	~	$	$	*	*	_	_	`	{	Ç	:	}	|	>	>	?	?
!	!	%	%	(	(	=	=	[	]	~	‘	,	,	;	/	\	\
@	@	¨	^	)	)	+	+	{	}	^	“	<	<	:	?	|	|