CARONTE - RELATÓRIO DO PROJETO LÉXICO

Adriano Tosetto, Giulio Simão

15104099, 15100738
INE5426 - 06208

DESCRIÇÃO

Caronte é uma linguagem de programação procedural estruturada baseada nas linguagens Lua e C. Sua proposta é oferecer uma linguagem semelhante em termos de simplicidade a Lua, mas que oferecesse um estilo e alguns recursos mais familiares a programadores de C.

Caronte oferece alguns recursos da linguagem Lua, como atribuição de múltiplas variáveis simultaneamente e iteração baseada em listas. Outros recursos foram retirados do C, como desvio incondicional. Outros recursos serão adicionados, tais como estruturas e uniões. Esses recursos terão a mesma semântica que eles já têm na linguagem C.

RECURSOS DA LINGUAGEM

- Criação de funções
- Condicionais: if
- Laços: for, while, repeat
- Desvios incondicionais: goto
- Funções inline
- Atribuição múltipla
- Constantes
- auto (define automaticamente o tipo da variável na inicialização)
- Structs (tbi)
- Arrays (tbi)
- Unions (tbi)
- Funções com número variável de parâmetros (tbi)

ESPECIFICAÇÃO E GRAFOS DE SINTAXE

inicio ::= (bloco)+

bloco ::= trecho | nomedafuncao corpodafuncao | define Nome valores

trecho ::= (comando)+ (ultimocomando)? |

(comando)* ultimocomando

comando ::= (listavar '=')? listaexp ';' |

tipovar var ('=' exp)? ';' |

auto var '=' exp ';' |

chamadadefuncao ';' |

do trecho end |

while exp do trecho end |

repeat trecho until exp ';' |

if exp then trecho (elseif exp then trecho)* (else trecho)? end |

for '(' ((tipovar | auto) Nome '=' exp)? ';' (exp)? ';' (exp)? ')' do trecho end |

for '(' listadenomes in listaexp ')' do trecho end | goto Nome ';' | Nome ':' comando

ultimocomando ::= return (listaexp)? ';' | break ';'

nomedafuncao ::= ('inline')? tiporet Nome

tipovar ::= boolean | int | double | float | string

tiporet ::= tipovar | void

listavar ::= var (',' var)*

var ::= Nome

listadenomes ::= Nome (',' Nome)*

listaexp ::= (exp ',')* exp

exp ::= valores | '...' | expprefixo |

exp opbin exp | opunaria exp | '(' exp ')'

valores ::= null | false | true | Numero | String

Numero ::= [0-9]+(','[0-9]+)?

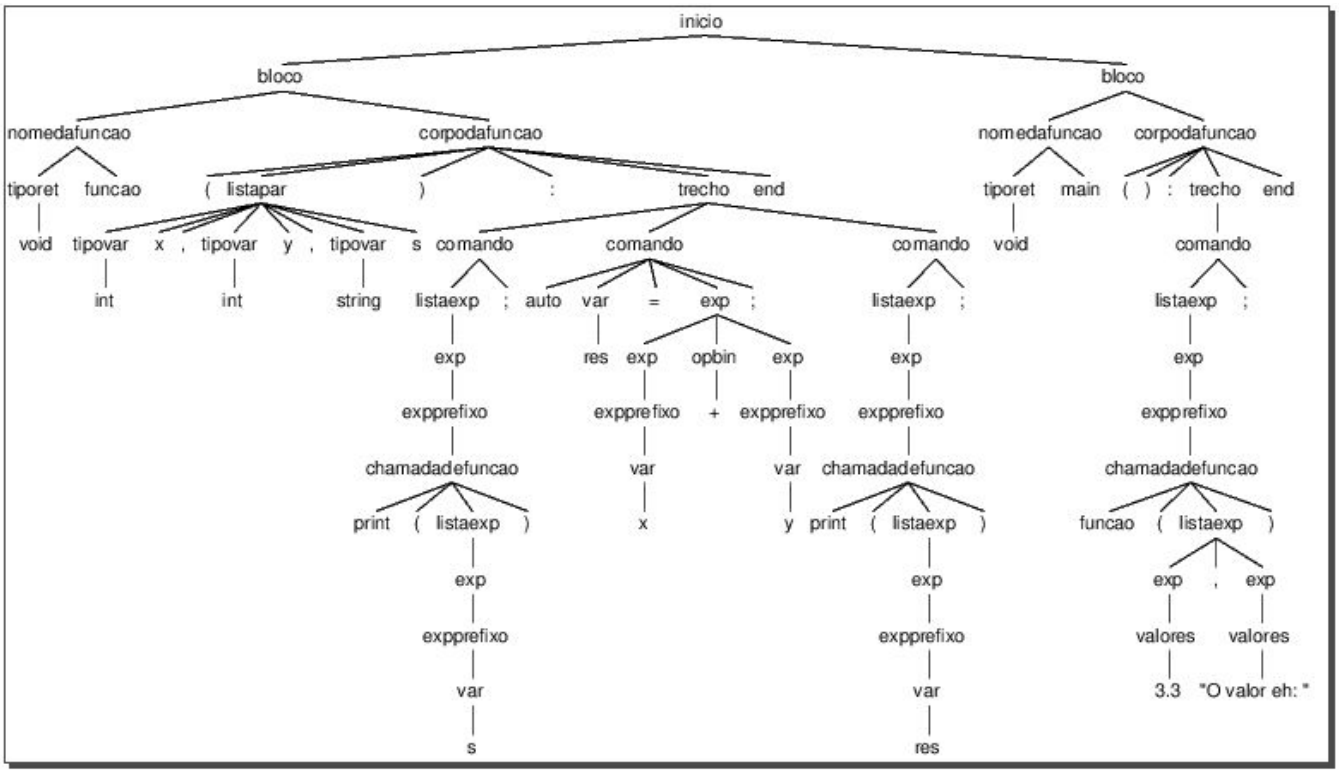
String ::= '"' [^"]* '"' | "'" [^']* "'"

```
Nome ::= [a-zA-Z_][a-zA-Z_0-9]*
expprefixo ::= var | chamadadefuncao
chamadadefuncao ::=  Nome '(' (listaexp)? ')'
corpodafuncao ::= '(' (listapar)? ')' ':' trecho end
listapar ::= (tipovar Nome ',')* tipovar Nome
opbin ::= '+' | '-' | '*' | '/' | '^' | '%' | '..' |
          '<' | '<=' | '>' | '>=' | '==' | '!=' |
          and | or | '+=' | '-=' | '*=' | '/=' | '^=' | '%='
opunaria ::= '-' | not | '#' | '++' | '--'
WS ::= #x9 | #xA | #xD | #x20
COMMENT ::= '/*' ( [^*] | '*'+ [^*/] )* '*'* '*/'
LINE_COMMENT ::= '//' ( [^*] | '*'+ [^*/] )* '*'* '\n'
```

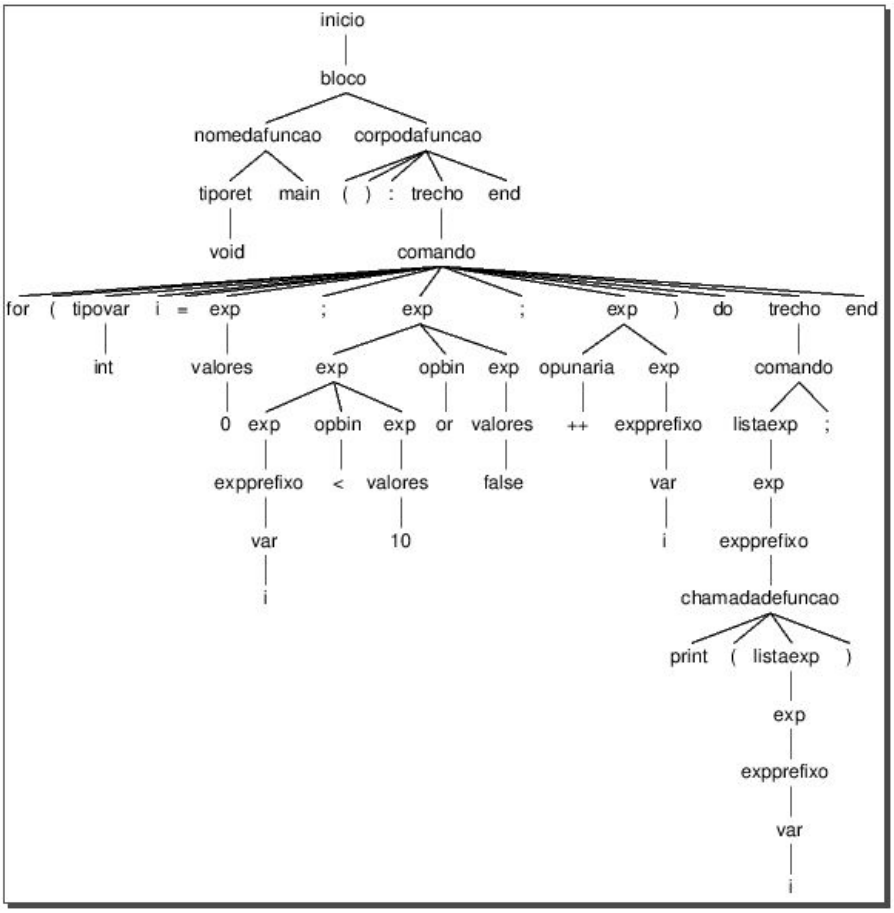
Os grafos de sintaxe estão presentes no zip da entrega.

EXEMPLOS E IDENTIFICAÇÃO DE TOKENS

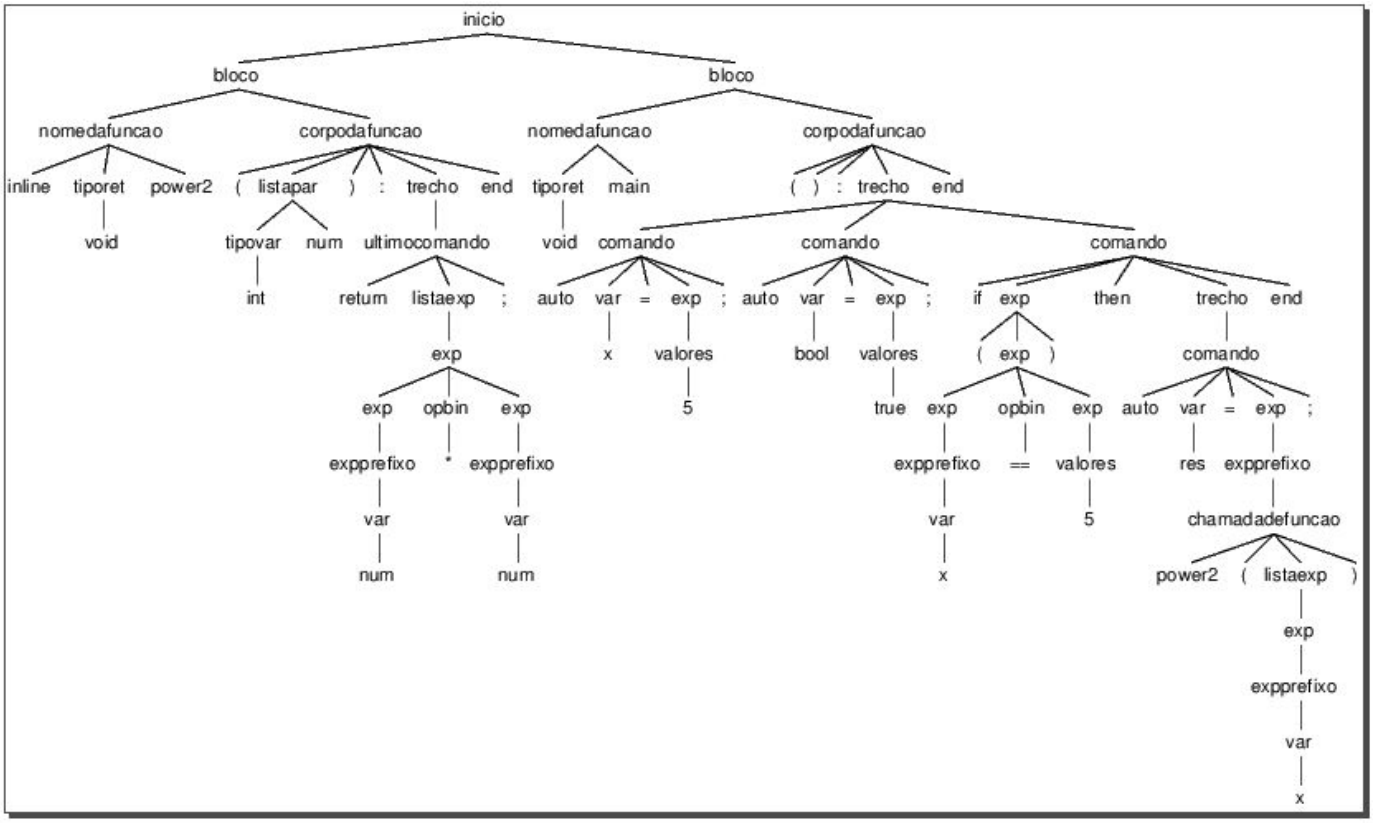
```
void funcao(int x, int y, string s):
    print(s);
    auto res = x+y;
    print(res);
end
void main():
    funcao(3.3, "O valor é: ");
end
```




```
void main():
    // este é um comentário de uma linha
    for (int i = 0;i<10 or false;++i) do
            print(i);
    end
    /*
            este é um comentário
            de múltiplas linhas
    */
end
```




```
inline void power2(int num):
    return num * num;
end
void main():
    auto x = 5;
    auto bool = true;
    if (x == 5) then
        auto res = power2(x);
    end
end
```

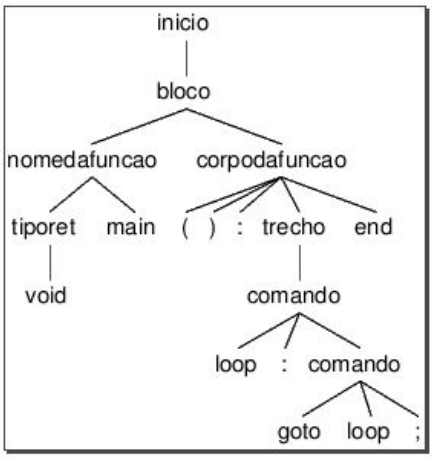


void main():

loop:

goto loop;

end



CÓDIGO-FONTE DO ANALISADOR LÉXICO

- Presente no zip da entrega