Analisador sintático e léxico para a linguagem C- -

Demonstrando o software

Victor E. Almeida Marco A. G. Pedroso

UNIOESTE

11 de abril de 2022

Conteúdo

1. Introdução
2. Bison
3. Analisador sintático
4. Conclusão

Atividades realizadas

- **Analisador sintático**: Victor E.;
- **Analisador léxico**: Victor E.;
- **funções auxiliares como log**: Victor E.;
- **Geração da árvore sintática**: Marco G.;
- **Geração dos slides**: Marco G.;
- **Geração do README**: Marco G.;
- **Geração dos arquivos fonte para entrada**: Victor E.;

Softwares utilizados

- **Sistema para compilar**: GNU make e gcc,
- **Linguagem de programação**: C11,
- **Gerador de analisador léxico**: GNU flex;
- **Gerador de analisador sintático**: GNU bison.

Exemplo mínimo do bison

```
1   %{
2       definicoes C
3   %}
4   %token algum_token
5   %start simbolo_inicial
6
7   %%
8   simbolo_inicial: algum_token ';'
9   %%
10
11  int main() {
12      return yyparse();
13  }
```

Exemplo de regra do bison

```
1  keyword:
2      TOKEN_KEYWORD_IF
3      | TOKEN_KEYWORD_ELSE
4      | TOKEN_KEYWORD_CONST
5      | TOKEN_KEYWORD_FOR
6      | TOKEN_KEYWORD_WHILE
7      | TOKEN_KEYWORD_RETURN
8      | TOKEN_KEYWORD_STRUCT
9      ;
```

Integrando Flex e bison

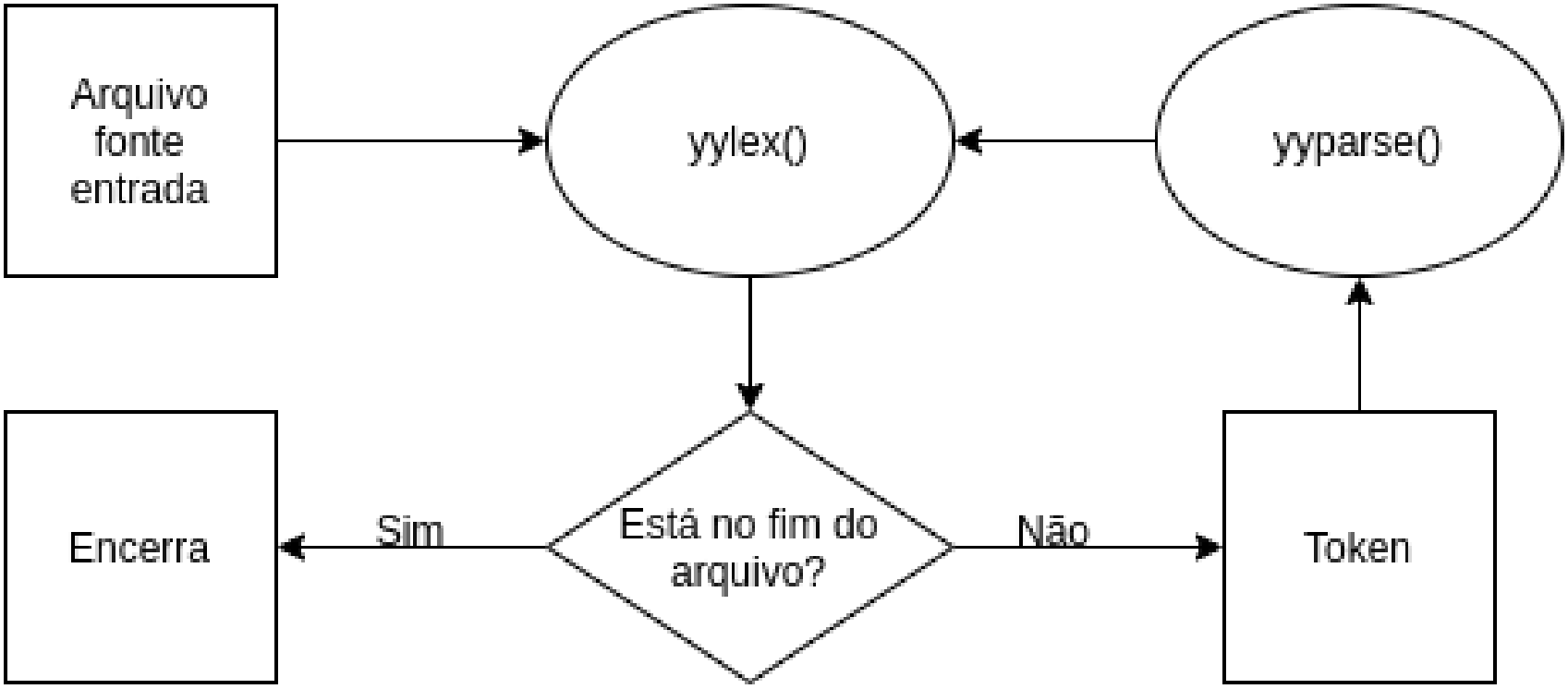

Figura 1: Fluxograma do compilador

Lista de algumas regras gramaticais I

- <programa> ::=
  TOKEN_PREPROCESSOR_COMMAND <programa> |
  <definition> <programa> | λ
- <keyword>::= TOKEN_KEYWORD_IF |
  TOKEN_KEYWORD_ELSE |
  TOKEN_KEYWORD_CONST |
  TOKEN_KEYWORD_FOR |
  TOKEN_KEYWORD_WHILE |
  TOKEN_KEYWORD_RETURN |
  TOKEN_KEYWORD_STRUCT

Lista de algumas regras gramaticais II

- <literal>::= TOKEN_INTEGER_LITERAL | TOKEN_FLOAT_LITERAL | TOKEN_CHAR_LITERAL | TOKEN_STRING_LITERAL
- <value >::=TOKEN_ID | TOKEN_INTEGER_LITERAL | TOKEN_FLOAT_LITERAL
- <exp>::= <complete_variable> <exp> | <ifel> <exp> | <atribuition> <exp> | <for> <exp> | <while> <exp> | λ

Mão na massa!!

Agradecimentos

Perguntas?

Obrigado pela atenção