Grupo 7

GABRIEL SERGIO FERREIRA
MARCOS DA COSTA NOVAES
PEDRO IVO TERRA BANDOLI

Introdução

Esse documento especifica o compilador que será implementado na disciplina.

Linguagem em que o compilador será implementado

Foi escolhida a linguagem **Python** para implementação do compilador pela facilidade de uso pelos membros do grupo.

Linguagem a ser implementada

Foi escolhida a linguagem **C-** a ser implementada pela familiaridade de uso com a linguagem **C** e talvez por ter um compilador mais simples.

Todos os integrantes do grupo já tiveram contato com a linguagem **C** nas disciplinas anteriores.

Foi suposto que um compilador em **C-** deve ser mais simples de ser implementado pois C é fortemente e estaticamente tipado.

BNF - Backus-Naur Form

Foi encontrado o BNF da linguagem a ser implementada no livro **Compiladores: Princípios E Práticas** do **Kenneth C. Louden**. Segue abaixo uma cópia do texto do livro.

1. *program* → *declaration-list*
2. *declaration-list* → *declaration-list declaration* | *declaration*
3. *declaration* → *var-declaration* | *fun-declaration*
4. *var-declaration* → *type-specifier* **ID** **;** | *type-specifier* **ID** **[** **NUM** **]** **;**
5. *type-specifier* → **int** | **void**
6. *fun-declaration* → *type-specifier* **ID** **(** *params* **)** *compound-stmt*
7. *params* → *param-list* | **void**
8. *param-list* → *param-list* **,** *param* | *param*
9. *param* → *type-specifier* **ID** | *type-specifier* **ID** **[** **]**
10. *compound-stmt* → **{** *local-declarations statement-list* **}**
11. *local-declarations* → *local-declarations var-declaration* | *empty*
12. *statement-list* → *statement-list statement* | *empty*
13. *statement* → *expression-stmt* | *compound-stmt* | *selection-stmt* | *iteration-stmt* | *return-stmt*
14. *expession-stmt* → *expression* **;** | **;**
15. *selection-stmt* → **if** **(** *expression* **)** *statement* | **if** **(** *expression* **)** *statement* **else** *statement*
16. *iteration-stmt* → **while** **(** *expression* **)** *statement*
17. *return-stmt* → **return** **;** | **return** *expression* **;**
18. *expression* → *var* **=** *expression* | *simple-expression*
19. *var* → **ID** | **ID** **[** *expression* **]**
20. *simple-expression* → *additive-expression relop additive-expression* | *additive-expression*
21. *relop* → **<=** | **<** | **>** | **>=** | **==** | **!=**
22. *additive-expression* → *additive-expression addop term* | *term*
23. *addop* → **+** | **-**
24. *term* → *term mulop factor* | *factor*
25. *mulop* → ***** | **/**
26. *factor* → **(** *expression* **)** | *var* | *call* | **NUM**
27. *call* → **ID** **(** *args* **)**
28. *args* → *arg-list* | *empty*
29. *arg-list* → *arg-list* **,** *expression* | *expression*

Keywords: **else if int return void while**

Special symbols: **+ - * / < <= > >= == != = ; , ( ) [ ] { } /* */**

ID = letter letter*
NUM = digit digit*
letter = a | .. | z | A | .. | Z
digit = 0 | .. | 9

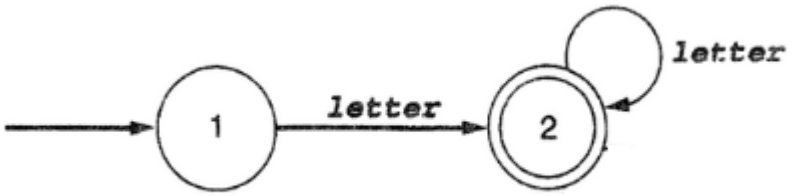
Comments: **/* ... */**

Expressões Regulares Mínimas

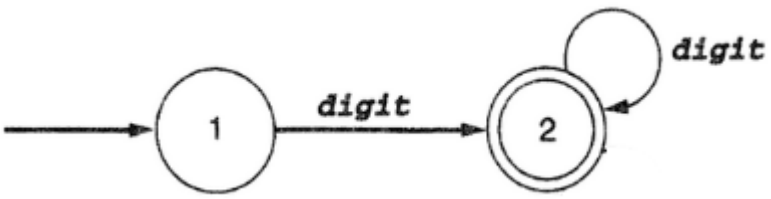
comentarios = \/*.**\/
digito = [0-9]
letra = [a-zA-Z]
numero = digito digito*
identificador = letra letra*
palavras reservadas = else if return void while
simbolos = + - * / < <= > >= == != = ; , ( ) [ ] { } /* */

Autômatos Finitos Determinísticos Mínimos

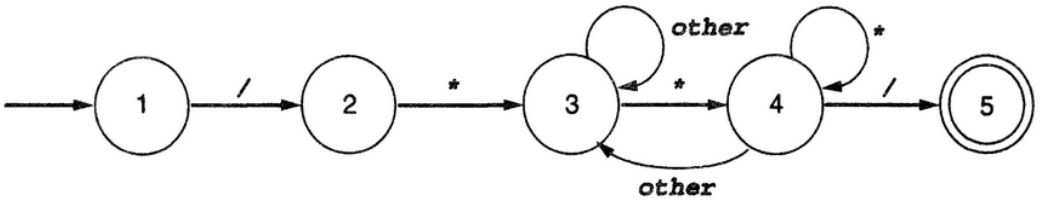
Identificador



Número



Comentário


other
*
1
/
2
*
3
*
4
/
5
other