Linguagens Formais e Compiladores

Aulas nº X: Analise Lexica

Prof. Luis Gustavo Araujo
2018

Objetivo

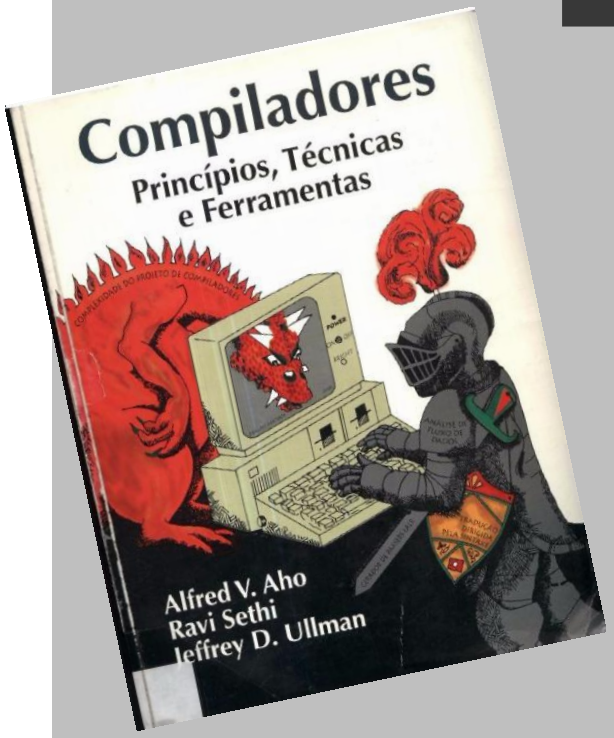
Compreender o objetivo e funcionamento da etapa de análise léxica.

Compiladores

Etapas

Análise Léxica

Compiladores



De forma mais simples, um compilador é um programa que lê um programa escrito numa linguagem – a linguagem fonte – e o traduz num programa equivalente numa outra linguagem – a linguagem alvo.

Alfred V. Aho

Compiladores: princípios, técnicas e ferramementas.

Fases da compilação

Fases da compilação

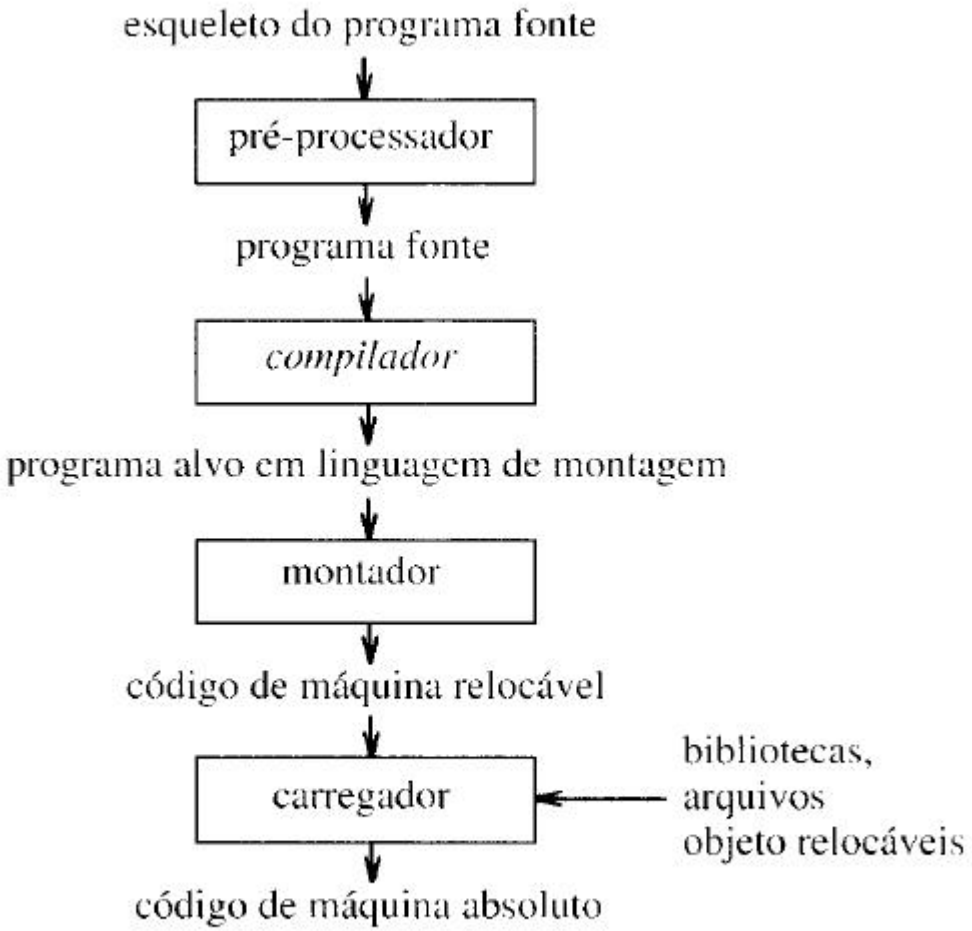


Fig. 1.3 Um sistema de processamento de linguagem.

Fases da compilação

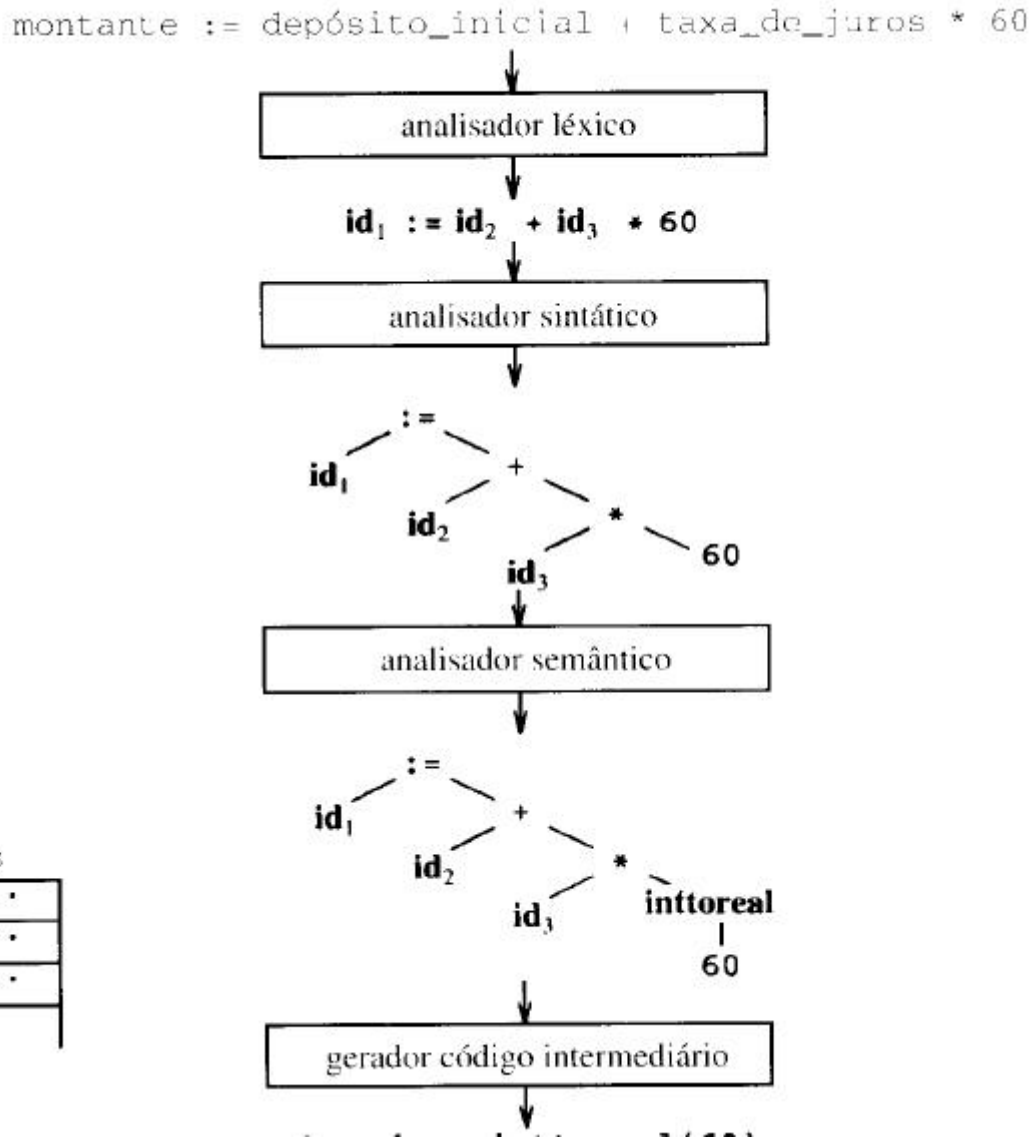


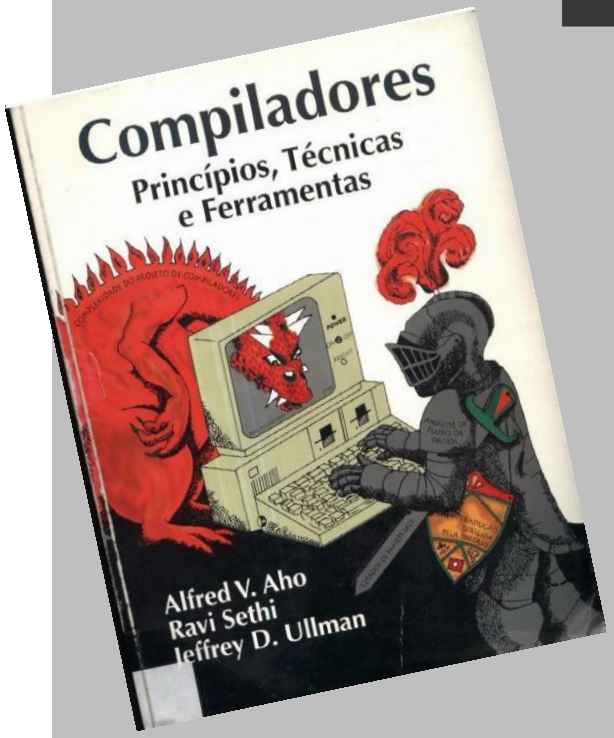
Tabela de Símbolos

1	montante	...
2	depósito_inicial	...
3	taxa_de_juros	...
4		

Fases da compilação

Fig. 1.10 Tradução de um enunciado.

Análise Léxica



O analisador léxico é a primeira fase de um compilador. Sua tarefa principal é a de ler os caracteres de entrada e produzir uma sequência de **tokens** *que o* ***parser*** *utiliza para a análise sintática.*

Alfred V. Aho

Compiladores: princípios, técnicas e ferramementas.

Análise Léxica

A AL é também chamada de **Scanner**, pois tem a função de ler o código fonte (*caracter a caracter*), buscando separar e identificar elementos componentes do programa (*tokens*).

Entrada: fluxo de caracteres;

Saída: fluxo de caracteres (*tokens*);

Exemplos: Lex e Flex

Tokens, Padrões e Lexemas

Em AL usamos alguns conceitos como Tokens, Padrões e Lexemas. A seguir uma descrição deles:

Tokens: Padrões de caracteres com um significado específico em um código fonte;

Padrões: são regras estipuladas para reconhecimento de tokens;

Lexemas: são ocorrências de tokens no código-fonte.

Tokens, Padrões e Lexemas

Em

```
1. int num  = 10;
```

int é um lexema de uma palavras reservada.;
num é um *lexema* para o *token* **identificador**;
= é um lexema para o token **operador**;
10 é um lexema para o **numero literal**, cujo valor é 10;
; é um lexema para o token do tipo **pontuação**.

Tokens, Padrões e Lexemas

TOKENS	LEXEMAS	DESCRIÇÃO INFORMAL DO PADRÃO
cosnt	const	const
if	if	if
relação	<, <=, =,<>,>,>=	< ou <= ou = ou<> ou > ou >=
id	pi, contador, D2	Letra seguida por letras e números
numeral	3.1416, 0, 6.02E23	Quaisquer constante numérica
string	“conteúdo de memória”	Quaisquer caracter em aspas, exceto as aspas

Tokens, Padrões e Lexemas

O padrão do **const**, na figura anterior, é justamente uma cadeia de caracter “const” e o padrão de **relação** é um conjunto dos operadores relacionais.

No entanto, para o padrão **id** e **num** é preciso usar um mecanismo de identificação mais rebuscado que uma simples comparação (=) , como por exemplo *expressões regulares.*

Atributos do Token

Do ponto de vista prático, o *token* só possui um atributo: um apontador para a tabela de símbolos na qual as infomações sobre os mesmos são mantidas.

```
E = M * C ** 2
```

são escritos abaixo como uma seqüência de pares:

<**id,** apontador para a entrada da tabela de símbolos para E>

<**operador_de_atribuição,**>

<**id,** apontador para a entrada da tabela de símbolos para M>

<**operador_de_multiplicação,**>

<**id,** apontador para a entrada da tabela de símbolos para C>

<**operador_de_exponenciação,**>

<**num,** valor inteiro 2>

Erros Léxicos

Vejamos o exemplo abaixo:

```
1. fi ( a == f(x)) . . .
```

O AL não é capaz de dizer que **fi** é a palavra-chave **if** incorretamente ou uma **função** não declarada anteriomente. Como **fi** é um identificador válido, a AL deve retornar o token e deixar alguma fase posterior do compilador tratar o erro.

Erros Léxicos

Suponha que o AL seja incapaz de prosseguir, por não ter identificado um padrão na entrada remanescente?

```
1. 1nome = “Luis”
```

O modo mais fácil é entrar na **modalidade pânico.** Removemos sucessivos caracteres da entrada até que o AL possa encontrar um token. Outras possibilidades são:

1. *Remover um caracter estranho;*
2. *Inserir um caracter ausente;*
3. *Substituir um caracter incorreto por um correto;*
4. *Transpor dois caracteres adjacentes.*

Atividade

Criar um analisador léxico, em qualquer linguagem, que identifique os *tokens* da linguagem Pascal (*not case sensitive*) descritos na tabela. O AL deve:

1 - Ter como entrada um arquivo do tipo texto;

2 - Escanear o arquivo, buscando identicar os *token*s;

3 - Gerar uma tabela de símbolos para armazenar os id e identificados;

4 - Imprimir os simbolos identificados.

Atividade

Elementos a serem identificados:

TOKENS	LEXEMAS	DESCRIÇÃO INFORMAL DO PADRÃO
string	string	Palavra “string”
int	int	Palavra “int”
operador_atribuição	:=	Dois pontos seguido de um sinal de igual
id	pi, contador, D2	Letra seguida por letras e números (que não seja reservado)
numeral	3.1416, 0, 6.02E23	Quaisquer constante numérica
string	“conteúdo de memória”	Quaisquer caracter em aspas, exceto as aspas
operador_soma	+	Simbolo +

Atividade

TOKENS	LEXEMAS	DESCRIÇÃO INFORMAL DO PADRÃO
operador_subtr açao	-	SÍmbolo -
operador_multip licação	*	Símbolo *
write	write	Palavra “write”
read	read	Palavra “read”
parentese	(,)	(ou)
ponto e vírgula	;	Símbolo ;

Atividade

No site da disciplina:

http://luisaraujo.github.io/aulas/unifacs/disciplinas/lin_comp/2018/aulas

Enviado por e-mail @unifacs

Referências Técnicas

AHO, Alfred V.; SETHI, Ravi; ULLMAN, Jeffrey D. **Compiladores: principios, técnicas e ferramentas**. LTC, Rio de Janeiro, 1995.

UNIFACS
LAUREATE INTERNATIONAL UNIVERSITIES

Linguagens Formais e Compiladores

Aulas nº X: Análise Léxica

Prof. Luis Gustavo Araujo
2018