Linguagens de Programação

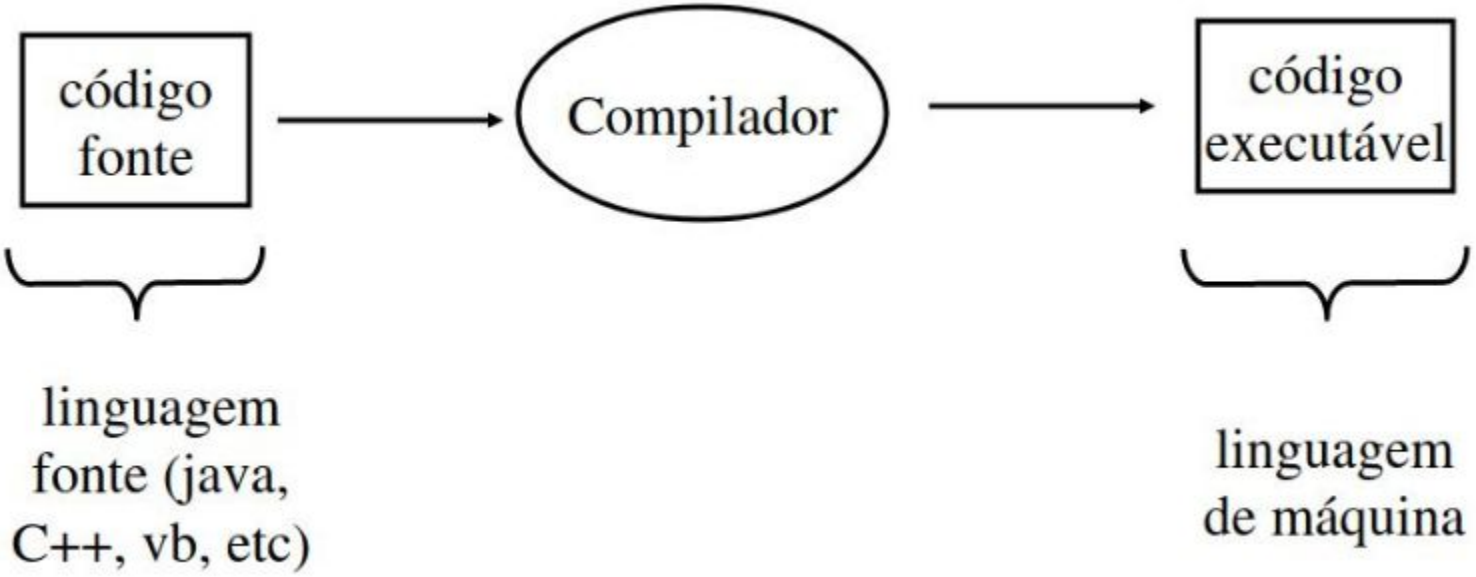
Análise Léxica e Sintática

Samuel da Silva Feitosa

Aula 16

Análise Léxica e Sintática

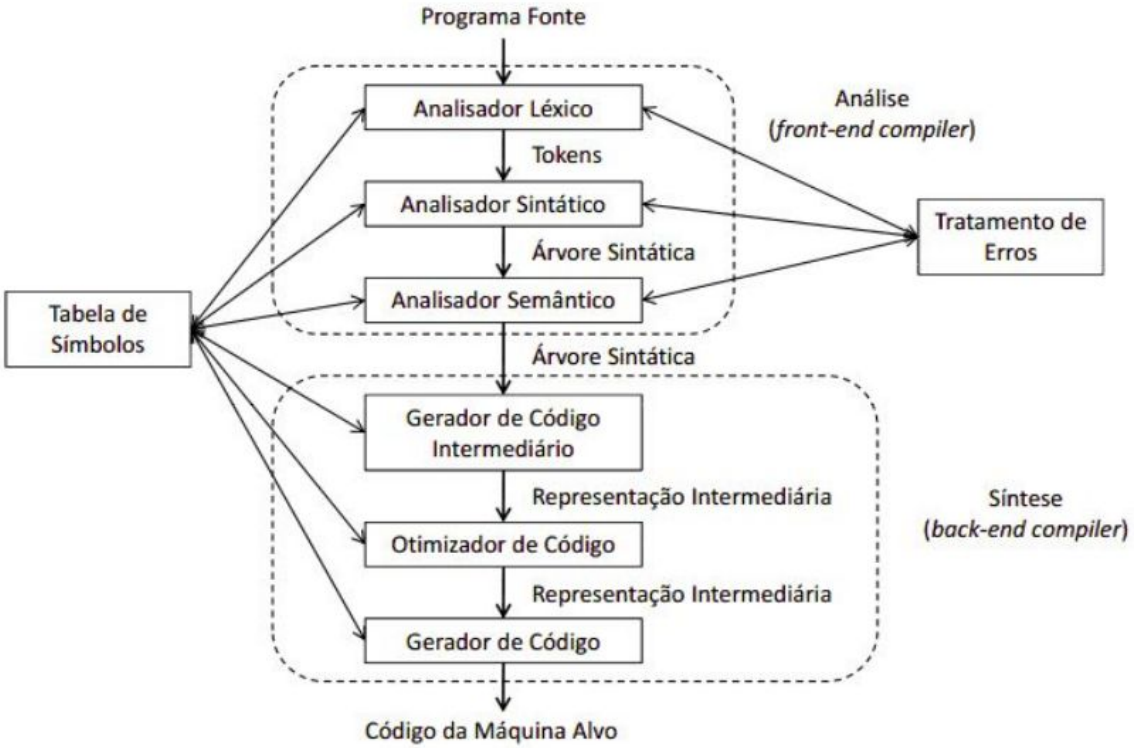
Processo de Compilação



O processo de compilação

- A “compilação” de um programa tipicamente envolve o seguinte:
  - Análise Léxica
  - Análise Sintática
  - Análise Semântica
  - Tradução / Interpretação

O processo de compilação



Análise Léxica

- A análise léxica é responsável por:
  - Ler o texto fonte, caracter por caracter.
  - Identificar os “elementos léxicos” da linguagem como identificadores, palavras reservadas, constantes, operadores.
  - Ignorar comentários, espaços em branco, tables, etc.
  - Processar “includes” (se for o caso).

ANÁLISE LÉXICA: X:=Y-50

ID	COD_ATRIB	ID,Y	COD_MENOS	CTENUM
X	:=	y	-	50

Análise Sintática

- A análise sintática é responsável por identificar, na sequência de elementos léxicos, as construções da linguagem.

Exemplo : para a sequência abaixo, em Pascal,

`“ if  a > b then a:=a-b else b:=b-a “`

A análise sintática deve idenficá-la como um ***comando condicional***, para a qual a ***condição*** é “`a > b`” e ao qual estão associados os comandos “`a := a-b`” e “`b := b-a`”.

Análise Sintática

> *"the way in wich words are put together to form phrases, clauses or sentences."*
>
> *Webster´s Dictionary*

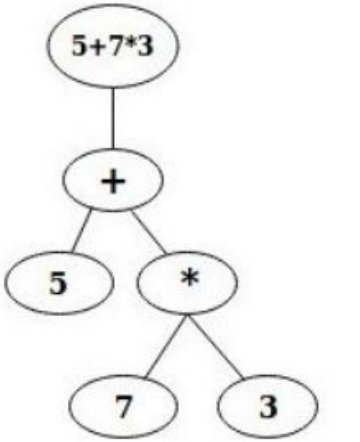
- As construções seguintes são válidas?

```
int y = 0,k = 0;
int x = y+++k;
```

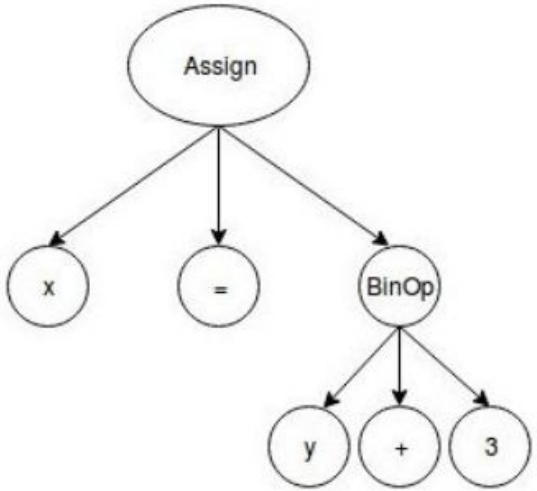
- Responsável por verificar quando uma sentença faz parte da **gramática** da linguagem.
- Produz ao final uma **árvore de sintaxe abstrata**.

Árvore de Sintaxe Abstrata (AST)

- Uma AST produz uma estrutura de árvore para o código fonte.



AST tree for expression: 5+7*3



Análise Léxica x Análise Sintática

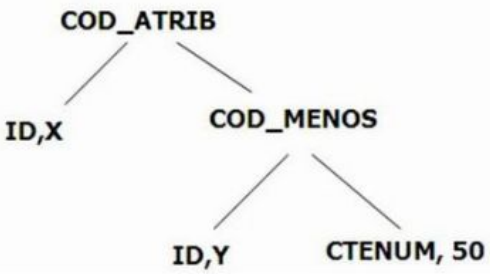
- A análise léxica e a análise sintática têm papéis que se complementam, já que as duas tratam de sequências de “símbolos”.
- A separação das duas em dois níveis de descrição e tratamento simplifica não só a descrição da linguagem como também a implementação do compilador.

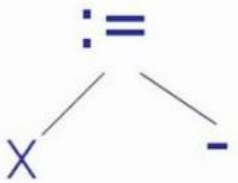
Análise Léxica x Análise Sintática

ANÁLISE LÉXICA: X:=Y-50

ID	COD_ATRIB	ID,Y	COD_MENOS	CTENUM
X	:=	y	-	50

ANÁLISE SINTÁTICA:





Como descrever uma linguagem?

- Gramáticas livres de contexto são utilizadas para **descrever** linguagens de programação.
  - Símbolo inicial
  - Produções
  - Símbolos terminais
  - Símbolos não-terminais

Uma gramática para o Português

- Exemplo de formação de sentenças simples em português.
  - **Sentença:** sintagma nominal + verbo + sintagma nominal — `<S> ::= <NP> <V> <NP>`
  - **Sintagma nominal:** artigo + substantivo — `<NP> ::= <A> <N>`
  - **Verbos:** `<V> ::= adora | odeia | come`
  - **Artigos:** `<A> ::= o | a | um | uma`
  - **Substantivos:** `<N> ::= cachorro | gato | rato`

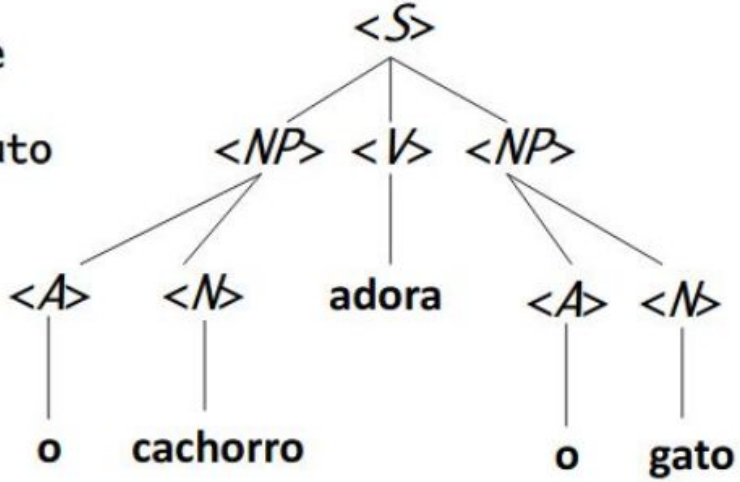
Funcionamento da Gramática

- A gramática especifica um **conjunto de regras** que diz como se deve construir uma **árvore de derivação** (*parse tree*).
- O **símbolo inicial** (neste exemplo <S>) deve ser colocado na raiz da árvore.
- As **regras** da gramática dizem como os nós podem ser expandidos (derivação) em qualquer ponto da árvore.
  - **Ex.:** a regra <S> ::= <NP> <V> <NP> diz que se pode adicionar os nós <NP>, <V> e <NP>, nessa ordem, como filhos de <S>

Árvore de Derivação

- Exemplo de derivação para a gramática anterior para o texto: **o cachorro adora o gato**.

```
<S>  ::= <NP> <V> <NP>
<NP> ::= <A> <N>
<V>  ::= adora | odeia | come
<A>  ::= o | a | um | uma
<N>  ::= cachorro | gato | rato
```



Uma gramática para uma Linguagem de Programação

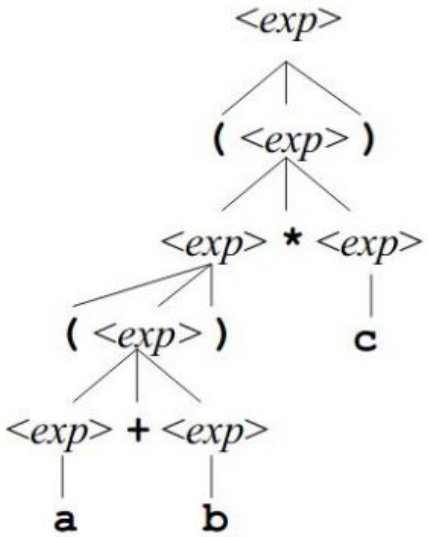
- Exemplo de definição de uma expressão de uma linguagem de programação, onde uma expressão pode ser:
  - Uma soma de duas expressões.
  - Um produto de duas expressões.
  - Uma subexpressão com parênteses.
  - Uma das variáveis **a**, **b** ou **c**.

```
<exp> ::= <exp> + <exp>
<exp> ::= <exp> * <exp>
<exp> ::= ( <exp> )
<exp> ::= a | b | c
```

Árvore de Derivação

- Exemplo de derivação para a gramática de expressões de uma linguagem de programação para a expressão: **((a + b) * c)**

```
<exp> ::= <exp> + <exp>
<exp> ::= <exp> * <exp>
<exp> ::= ( <exp> )
<exp> ::= a | b | c
```



Considerações Finais

- Nesta aula estudamos dois conceitos que integram o processo de compilação de um programa.
  - Análise léxica: responsável por separar *tokens*.
  - Análise sintática: responsável por gerar a *AST*.
- Também vimos brevemente como criar uma gramática que especifica uma linguagem de programação.
  - Aceita apenas construções válidas da linguagem.