# TRABALHO FINAL – parte 3: implementação do analisador sintático

Implementar o **analisador sintático** de forma que indique quais programas escritos na linguagem 2018.1 estão sintaticamente corretos, seguindo as orientações abaixo:
1º passo: efetue correções, se for o caso, na especificação dos *tokens* da linguagem conforme indicado na avaliação do trabalho no.3.

2º passo: efetue correções na gramática da linguagem conforme solicitado/indicado na avaliação do trabalho no.3.

3º passo: implemente o analisador sintático, bem como o tratamento de erros sintáticos, conforme especificado abaixo.

| | |
|---|---|
| *Entrada* | – A entrada para o analisador sintático é um conjunto de caracteres, isto é, o programa fonte do editor do compilador. |
| *Saída* | – Caso o botão **compilar** seja pressionado, a ação deve ser: executar as análises léxica e sintática do programa fonte e apresentar a saída. Um programa pode ser compilado com sucesso ou apresentar erros. Em cada uma das situações a saída deve ser:<br>1ª situação: programa compilado com sucesso<br>✓ **mensagem** (*programa compilado com sucesso*), na área reservada para mensagens, indicando que o programa não apresenta erros.<br>A **lista de *tokens*** não deve mais ser mostrada na área reservada para mensagens.<br><br>2ª situação: programa apresenta erros<br>✓ **mensagem**, na área reservada para mensagens, indicando que o programa apresenta erro. O erro pode ser **léxico** ou **sintático**, cujas mensagens devem ser conforme descrito abaixo.<br>As mensagens geradas pelo GALS devem ser alteradas. |

**OBSERVAÇÕES:**

- O tipo do analisador sintático a ser gerado é **LL (1)**.
- As mensagens para os **erros léxicos** devem ser conforme especificado na parte 2 do trabalho final.
- As mensagens para os **erros sintáticos** devem indicar a linha onde ocorreu o erro, o *token* encontrado (lexema) e o(s) símbolo(s) esperado(s), conforme explicado em aula. Assim, tem-se alguns exemplos:

  Erro na linha 1 – **encontrado** `fim de programa` **esperado** `def`
  Erro na linha 1 – **encontrado** `constante int` **esperado** `identificador`

  Observa-se que:
  - quando for encontrado ou esperado `$`, a mensagem deve ser do tipo: **encontrado** `fim de programa` **esperado** ... ou **encontrado** ... **esperado** `fim de programa`, conforme o caso
  - quando for encontrado ou esperado `cte_int`, `cte_float` ou `cte_str`, a mensagem deve ser do tipo: **encontrado** `constante int` **esperado** ..., **encontrado** `constante float` **esperado** ..., **encontrado** `constante str` **esperado** ... ou vice-versa, conforme o caso
  - quando forem esperados `identificador`, `input`, `print`, `println` e `(`, ou seja, os cinco símbolos iniciais dos comandos da linguagem, conforme especificado no trabalho no.2, a mensagem deve ser do tipo: **encontrado** ... **esperado** `comando`
  - para o não-terminal `<lista_expressões>`, ou com outro nome, usado para definir essa estrutura sintática especificada no trabalho no.2, a mensagem deve ser do tipo: **encontrado** ... **esperada** `expressão`
  - para todos os não-terminais alcançados a partir de `<expressão>` (inclusive), a mensagem deve ser do tipo: **encontrado** ... **esperada** `expressão`
  - para os demais não-terminais, a mensagem deve ser do tipo: **encontrado** ... **esperado** $símbolo_1$ $símbolo_2$ ... $símbolo_n$, conforme tabela de análise sintática
  - todas as mensagens de erro geradas pelo GALS devem ser mantidas (em comentário), MAS devem ser alteradas, conforme especificado acima.

Assim, por exemplo, considerando o seguinte "trecho" da tabela de análise sintática:

| | $ | identificador | cte_int | cte_float | cte_str | bool | consts | def | end | execute | false | float | get | ifFalse | ifTrue | input | print | println | true | var | ( | ! | + | - |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| <programa> | – | – | – | – | – | – | – | 0 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| <def_constantes> | – | – | – | – | – | – | 17 | – | – | 16 | – | – | – | – | – | – | – | – | – | 16 | – | – | – | – |
| <lista_comandos> | – | 37 | – | – | – | – | – | – | – | – | – | – | – | – | – | 37 | 37 | 37 | – | – | 37 | – | – | – |
| <lista_comandos_> | 38 | 39 | – | – | – | – | – | – | 38 | – | – | – | – | 38 | – | 39 | 39 | 39 | – | – | 39 | – | – | – |
| <comando> | – | 40 | – | – | – | – | – | – | – | – | – | – | – | – | – | 41 | 42 | 42 | – | – | 43 | – | – | – |
| <entrada> | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | 46 | – | – | – | – | – | – | – | – |
| <lista_expressões> | – | 49 | 49 | 49 | 49 | – | – | – | – | – | 49 | – | – | – | – | – | – | – | 49 | – | 49 | 49 | 49 | 49 |
| <expressao> | – | 60 | 60 | 60 | 60 | – | – | – | – | – | 60 | – | – | – | – | – | – | – | 60 | – | 60 | 60 | 60 | 60 |

As mensagens de erro para os não-terminais relacionados devem ser:

- para o não-terminal `<programa>`: **encontrado** ... **esperado** `def`
- para o não-terminal `<def_constantes>`: **encontrado** ... **esperado** `consts, execute, var`
- para os não-terminais `<lista_comandos>` e `<comando>`: **encontrado** ... **esperado** `comando`
- para o não-terminal `<lista_comandos_>`: **encontrado** ... **esperado** `fim de programa, end, ifFalse, comando`
- para o não-terminal `<entrada>`: **encontrado** ... **esperado** `input`
- para o não-terminal `<lista_expressões>` e `<expressao>`: **encontrado** ... **esperada** `expressão`

- A gramática especificada no trabalho nº3 (com as devidas correções) deve ser usada para implementação do analisador sintático. Além disso, trabalhos desenvolvidos usando especificações diferentes daquelas elaboradas pela equipe no trabalho nº3 receberão nota 0.0 (zero).
- A implementação do analisador sintático, bem como da interface do compilador e do analisador léxico, deve ser disponibilizada no AVA, na **pasta da sua equipe**. Deve ser disponibilizado um **arquivo compactado** (com o nome: `sintatico`), contendo: o código fonte, o executável e o arquivo com as especificações léxica e sintática (no GALS, arquivo com extensão `.gals`).
- Na avaliação do analisador sintático serão levadas em consideração: a correta especificação da gramática, conforme trabalho nº3, a qualidade das mensagens de erro, conforme descrito acima e o uso apropriado de ferramentas para construção de compiladores. Observa-se que todas as mensagens de erro sintático geradas pelo GALS devem ser alteradas conforme especificado anteriormente.

**DATA:** entregar o trabalho até às 23h do dia 15/06/2018 (sexta-feira). Não serão aceitos trabalhos após data e hora determinados.

---

# EXEMPLOS DE ENTRADA / SAÍDA

**EXEMPLO 1:** com erro léxico

| ENTRADA | | SAÍDA (na área de mensagens) |
|---|---|---|
| linha | | `Erro na linha 4 - á símbolo inválido` |
| 1 | `def` | |
| 2 | `execute` | |
| 3 | `  input (lado)` | |
| 4 | `  área  lado * lado` | |
| 5 | `  print (area)` | |

**EXEMPLO 2:** com erro sintático

| ENTRADA | | SAÍDA (na área de mensagens) |
|---|---|---|
| linha | | `Erro na linha 4 - encontrado lado esperado set, :=` |
| 1 | `def` | |
| 2 | `execute` | |
| 3 | `  input (lado)` | |
| 4 | `  area  lado * lado` | |
| 5 | `  print (area)` | |

**EXEMPLO 3:** sem erro

| ENTRADA | | SAÍDA (na área de mensagens) |
|---|---|---|
| linha | | `programa compilado com sucesso` |
| 1 | `def` | |
| 2 | `execute` | |
| 3 | `  input (lado)` | |
| 4 | `  area := lado * lado` | |
| 5 | `  print (area)` | |