TRABALHO FINAL – parte 3: implementação do analisador sintático

Implementar o **analisador sintático** de forma que indique quais programas escritos na linguagem 2013.1 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, se for o caso, na gramática da linguagem conforme 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: 1ª situação: programa compilado com sucesso ✓ **mensagem** (*programa compilado com sucesso*), na área reservada para mensagens, indicando que o programa não apresenta erros. A **lista de *tokens*** não deve mais ser mostrada na área reservada para mensagens. 2ª situação: programa apresenta erros ✓ **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. 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 e o(s) símbolo(s) esperado(s). Assim, tem-se alguns exemplos:

  Erro na linha 1 – **encontrado** `fim de programa` **esperado** `)`
  Erro na linha 1 – **encontrado** `identificador (i_valor)` **esperado** `isFalseDo ou isTrueDo`

  Observa-se que:
  - caso o símbolo encontrado seja um identificador (de `int` ou de `float` ou de `string` ou de `bool`), uma constante `int`, uma constante `float` ou uma constante `string`, devem ser mostrados a classe e o lexema entre parênteses, como em: `identificador (i_valor)`;
  - quando for esperado um identificador (de `int` ou de `float` ou de `string` ou de `bool`), a mensagem deve ser do tipo: **encontrado** ... **esperado** `identificador`;
  - para os não-terminais alcançados a partir de `<expressão>`, inclusive, a mensagem deve ser do tipo: **encontrado** ... **esperada** `expressão`;
  - para os não-terminais cujo primeiro símbolo de alguma de suas regras seja `<expressão>`, como por exemplo `<lista_de_expressão>`, a mensagem também deve ser do tipo: **encontrado** ... **esperada** `expressão`;
  - para os demais não-terminais, a mensagem deve ser do tipo: **encontrado** ... **esperado** $símbolo_1$, $símbolo_2$, ... $símbolo_n$;
  - são exemplos de mensagens de erro inadequadas: `<lista_de_comandos>` inválido, esperado `cteInteira` ou `$ inesperado`;
  - deve ser mantida (em comentário) a mensagem de erro gerada pelo GALS.
- 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, no **repositório 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.

DATA: entregar o trabalho até às 23h do dia 29/05/2013 (quarta-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 3 - constante literal não finalizada
1	main module [	
2		
3	out ("digite o valor do lado:);	
4	in (i_lado)	
5	i_area <- i_lado * i_lado;	
6	out (i_area);	
7	]	

EXEMPLO 2: com erro sintático

ENTRADA		SAÍDA (na área de mensagens)
linha		Erro na linha 5 - encontrado identificador (i_area) esperado ;
1	main module [	
2		
3	out ("digite o valor do lado:");	
4	in (i_lado)	
5	i_area <- i_lado * i_lado;	
6	out (i_area);	
7	]	

EXEMPLO 3: sem erro

ENTRADA		SAÍDA (na área de mensagens)
linha		programa compilado com sucesso
1	main module [	
2		
3	out ("digite o valor do lado:");	
4	in (i_lado);	
5	i_area <- i_lado * i_lado;	
6	out (i_area);	
7	]	