TRABALHO FINAL – parte 3: implementação do analisador sintático

Implementar o **analisador sintático** de forma que indique quais programas escritos na linguagem 2024.2 estão sintaticamente corretos, seguindo as orientações abaixo:
1º passo: efetue correções, se for o caso, na especificação da gramática, conforme solicitado/indicado na avaliação do trabalho no.3.

2º 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, conforme explicado em aula.

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 e detalhado a seguir. Assim, tem-se alguns exemplos:

  Erro na linha 1 – **encontrado** `EOF` **esperado** `expressão`
  Erro na linha 1 – **encontrado** `i_area` **esperado** `(`

  Observa-se que:
  - quando for encontrado uma constante_`string` , como por exemplo `"olá mundo"`, a mensagem deve ser: **`encontrado`** `constante_string` **`esperado ...`**
  - quando for encontrado `$`, a mensagem deve ser: **`encontrado`** `EOF` **`esperado ...`**
  - para qualquer uma das demais classes de *tokens* encontradas, deve ser apresentado o lexema na mensagem de erro, como em: **`encontrado`** `i_area` **`esperado ...`** , **`encontrado`** `123` **`esperado ...`** , **`encontrado`** `;` **`esperado ...`**
  - quando for esperado fim de programa, a mensagem deve ser: **`encontrado ... esperado`** `EOF`
  - quando for esperado identificador, constante_`int`, constante_`float` ou constante_`string`, a mensagem deve ser, respectivamente: **`encontrado ... esperado`** `identificador`, **`encontrado ... esperado`** `constante_int`, **`encontrado ... esperado`** `constante_float`, **`encontrado ... esperado`** `constante_string`
  - as palavras reservadas e os símbolos especiais devem ser escritos nas mensagens de erro tal como definido na linguagem, ou seja, exatamente como o programador deve escrevê-los nos programas
  - para o não-terminal `<lista_de_expressoes>`, ou com outro nome, usado para definir essa estrutura sintática especificada no trabalho no.3, a mensagem deve ser: **`encontrado ... esperado`** `expressao`
  - para os não-terminais `<expressao>`, `<expressao1>`, `<elemento>`, `<relacional>`, `<relacional1>`, `<aritmética>`, `<aritmética1>`, `<termo>`, `<termo1>` e `<fator>`, a mensagem deve ser: **`encontrado ... esperado`** `expressão`
  - para os demais não-terminais, a mensagem deve ser: **`encontrado ... esperado`** `símbolo`$_1$`, símbolo`$_2$`, ... símbolo`$_n$, conforme tabela de análise sintática (exemplificado a seguir);
  - são exemplos de mensagens de erro **inadequadas**: `<lista_comandos> inválido`, `Era esperado cte_int` ou `$ inesperado`
  - todas as mensagens de erro geradas pelo GALS devem ser mantidas (em **comentário de linha**), MAS devem ser alteradas, seguindo as orientações dadas em aula.

Por exemplo, considerando o seguinte "trecho" da tabela de análise sintática (menu `Documentação > Tabela de Análise Sintática`):

	main	id	read	write	writeln	if	repeat	","	";"	"="
<programa>	0	-	-	-	-	-	-	-	-	-
<lista_instrucoes>	-	1	1	1	1	1	1	-	-	-
<lista_identificadores>	-	17	-	-	-	-	-	-	-	-
<lista_identificadores_>	-	-	-	-	-	-	-	19	18	18

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

- para o não-terminal `<programa>`: **encontrado** ... **esperado** `main`
- para o não-terminal `<lista_instrucoes>`: **encontrado** ... **esperado** `identificador read write writeln if repeat`
- para o não-terminal `<lista_identificadores>`: **encontrado** ... **esperado** `identificador`
- para o não-terminal `<lista_identificadores_>`: **encontrado** ... **esperado** `, ; =`
- para o não-terminal `<lista_de_expressoes>`: **encontrado** ... **esperado** `expressao`
- para o não-terminal `<expressao>`: **encontrado** ... **esperado** `expressao`

- 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 tarefa **"parte 3 - analisador sintático"**, aba **COMPILADOR**. Deve ser disponibilizado um **arquivo compactado** (com o nome: `sintatico`, seguido do número da equipe), contendo: o projeto completo, incluindo **código fonte**, **executável** (ou **.jar**), **arquivo com as especificações léxica e sintática** (no GALS, arquivo com extensão `.gals`) e demais recursos. Basta um integrante da equipe postar o trabalho. Caso não sejam postados os arquivos solicitados, será atribuída nota 0.0 (zero).
- 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; o uso apropriado de ferramentas para construção de compiladores. Observa-se que <u>todas</u> as mensagens de erro sintático geradas pelo GALS devem ser alteradas conforme especificado anteriormente.

DATA: entregar o trabalho até às 23h do dia 20/10/2024 (domingo).

EXEMPLOS DE ENTRADA / SAÍDA

EXEMPLO 1: com erro léxico

ENTRADA	SAÍDA (na área de mensagens)
linha **1:** main **2:** i_lado, i_area; **3:** read ("digite um valor para lado: , i_lado; **4:** i_area = i_lado * i_lado; **5:** writeln (i_area); **6:** end	Erro na linha 3 - constante_string inválida

EXEMPLO 2: com erro sintático

ENTRADA	SAÍDA (na área de mensagens)
linha **1:** main **2:** i_lado, i_area; **3:** read ("digite um valor para lado:", i_lado; **4:** i_area = i_lado * i_lado; **5:** writeln (i_area); **6:** end	Erro na linha 3 - encontrado ; esperado ,)

EXEMPLO 3: sem erro

ENTRADA	SAÍDA (na área de mensagens)
linha **1:** main **2:** i_lado, i_area; **3:** read ("digite um valor para lado:", i_lado); **4:** i_area = i_lado * i_lado; **5:** writeln (i_area); **6:** end	programa compilado com sucesso