TRABALHO FINAL – parte 2: implementação do analisador léxico

Implementar o **analisador léxico** de forma que identifique, nos programas escritos na linguagem 2025.2, os *tokens* corretos, levando em consideração as especificações/correções feitas no trabalho nº1, assim como as palavras reservadas e os símbolos especiais relacionados a seguir. Deve-se implementar também **tratamento de erros** léxicos, quais sejam: símbolos que não fazem parte da linguagem em questão, bem como sequências de símbolos que não obedecem às regras de formação dos *tokens* especificados.

Na implementação do analisador léxico pode ser utilizada qualquer ferramenta para geração de compiladores (GALS, JavaCC, etc.) que gere analisadores sintáticos do tipo descendente (recursivo ou preditivo tabular).

entrada	– A entrada para o analisador léxico é um conjunto de caracteres, isto é, o programa fonte do editor do compilador.
saída	– Caso o botão **compilar** seja pressionado (ou a tecla de atalho correspondente), a ação deve ser: executar a análise léxica 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 ✓ apagar o conteúdo da área reservada para mensagens; ✓ apresentar a **lista de *tokens***, na área reservada para mensagens, contendo, para cada *token* reconhecido, a linha onde se encontra, a sua classe (por extenso) e o lexema, nessa ordem, conforme exemplo; ✓ apresentar a **mensagem** (*programa compilado com sucesso*), na área reservada para mensagens, indicando que o programa não apresenta erros. As classes possíveis para os *tokens* são: símbolo especial, palavra reservada, identificador, constante_`int`, constante_`float` e constante_`string`. 2ª situação: programa apresenta erros ✓ apagar o conteúdo da área reservada para mensagens; ✓ apresentar a **mensagem**, na área reservada para mensagens, indicando que o programa apresenta erro. Neste caso, indicar a linha onde ocorreu o erro e a descrição do erro, emitindo uma mensagem adequada, conforme relacionado abaixo: • para símbolo inválido – apresentar: a mensagem (*símbolo inválido*), a linha onde ocorreu o erro e o símbolo não reconhecido; • para identificador inválido – apresentar: a mensagem (*identificador inválido*) e a linha onde ocorreu o erro; • constante `string` inválida – apresentar: a mensagem (*constante_`string` inválida*) e a linha onde ocorreu o erro; • para comentário de inválido – apresentar: a mensagem (*comentário inválido ou não finalizado*) e a linha onde ocorreu o erro, sendo que, nesse caso, deve ser apresentada a linha onde inicia o comentário. As mensagens geradas por ferramentas, como o GALS, devem ser alteradas conforme as orientações dadas em aula.

OBSERVAÇÕES:

- As palavras reservadas da linguagem 2025.2, escritas exatamente conforme segue, são: `add and begin bool count delete do elementOf else end false float if int list not or print read size string true until` As palavras reservadas devem ser especificadas como casos especiais de `identificador` (ou outro nome) definido no trabalho nº1, sendo que a linguagem é *case sensitive*. Observa-se que, se necessário, outras palavras reservadas podem ser incluídas, quando da especificação das regras gramaticais.
- Os símbolos especiais da linguagem 2025.2 são: `+ - * / == ~= < > = <- ( ) ; ,` Símbolos diferentes dos especificados nesse item constituem erro léxico. Observa-se que, se necessário, outros símbolos especiais podem ser incluídos, quando da especificação das regras gramaticais.
- Os comentários de linha, comentários de bloco e os caracteres de formatação (espaços em branco, quebra de linha, tabulação) devem ser reconhecidos, porém ignorados. Ou seja, não devem ser apresentados como saída do analisador léxico. Isso deve ser especificado na própria ferramenta que será usada para gerar o analisador léxico, no arquivo com as especificações léxicas (no caso do GALS, arquivo com extensão `.gals`). Comentários que não seguem o padrão de formação especificado no trabalho nº1 devem ser diagnosticados como erro léxico.
- As **mensagens de erro** devem ser conforme exemplos abaixo:
  linha 1: `@` símbolo inválido
  linha 1: identificador inválido
  linha 1: constante_`string` inválida
  linha 1: comentário inválido ou não finalizado

  No primeiro exemplo, o símbolo `@` não é um símbolo especial da linguagem, portanto caracteriza um erro léxico e deve ser apresentado na mensagem de erro, juntamente com a linha onde o erro foi detectado. Nos demais exemplos, deve ser apresentada a mensagem de erro, juntamente com a linha onde o erro foi detectado. Ou seja, a sequência não reconhecida não deve ser apresentada na mensagem de erro.
- As especificações feitas no trabalho nº1 (e já corrigidas) devem usadas para implementação do analisador léxico. Observa-se que: (1) as especificações feitas no trabalho nº1 devem ser adaptadas à notação da ferramenta que será utilizada para gerar o analisador léxico; (2) o analisador léxico desenvolvido usando especificações diferentes daquelas elaboradas pela equipe no trabalho nº1 receberá nota 0.0 (zero); (3) o trabalho nº1 deve ser devolvido para professora na aula de 28/08, caso contrário, o analisador léxico receberá nota 0.0 (zero).

- A implementação do analisador léxico, bem como da interface do compilador, deve ser disponibilizada no **AVA**, na aba **COMPILADOR**, na tarefa **"parte 2 - analisador léxico"**. Deve ser disponibilizado **um arquivo compactado** (com o nome: `lexico`, seguido do número da equipe) contendo: o projeto completo, incluindo **código fonte**, **executável** (ou **.jar**), **arquivo com as especificações léxicas** (no GALS, arquivo com extensão `.gals`) e demais recursos. Basta um integrante da equipe postar o trabalho. Caso não sejam postados todos os arquivos solicitados ou não seja possível rodar o **executável** (ou **.jar**), será atribuída nota 0.0 (zero) ao analisador léxico.
- Na avaliação do analisador léxico serão levadas em consideração: a correta adaptação dos *tokens*, definidos no trabalho no.1, à notação da ferramenta utilizada para gerar o analisador léxico; a qualidade das mensagens de erro, conforme descrito acima; o uso apropriado de ferramentas para construção de compiladores.

DATA: entregar o trabalho até às 23h do dia 26/09/2025 (sexta).

EXEMPLOS DE ENTRADA / SAÍDA

EXEMPLO 1: sem erro léxico

ENTRADA

```
linha
   1: {
   2: isso é um comentário de bloco
   3: }
   4:
   5: add area_1  (
   6:
   7: 1.00
```

SAÍDA (na área de mensagens)

```
linha   classe                  lexema
5       palavra reservada       add
5       identificador           area_1
5       símbolo especial        (
7       constante_float         1.0
7       constante_int           0

        programa compilado com sucesso
```

EXEMPLO 2: com erro léxico

ENTRADA

```
linha
   1: {
   2: isso é um comentário de bloco
   3: }
   4:
   5: add area_1  (@
   6:
   7: 1.00
```

SAÍDA (na área de mensagens)

```
linha 5: @ símbolo inválido
```