TRABALHO FINAL – parte 2: implementação do analisador léxico

Implementar o **analisador léxico** de forma que identifique, nos programas escritos na linguagem 2023.2, os *tokens* corretos, levando em consideração as especificações/correções feitas no trabalho nº1. 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 ✓ **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; ✓ **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 ✓ **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. Tem-se que: • para símbolo inválido – apresentar: a mensagem (*símbolo inválido*), a linha onde ocorreu o erro e o símbolo não reconhecido; • para palavra reservada inválida – apresentar: a mensagem (*palavra reservada inválida*), a linha onde ocorreu o erro e a sequência não reconhecida; • para identificador inválido – apresentar: a mensagem (*identificador inválido*), a linha onde ocorreu o erro e a sequência não reconhecida; • constante `string` inválida – apresentar: a mensagem (*constante_`string` inválida*) e a linha onde ocorreu o erro; • para comentário de bloco inválido ou não finalizado – apresentar: a mensagem (*comentário de bloco 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.

OBSERVAÇÕES:

- As palavras reservadas da linguagem 2023.2 são: `do else false fun if in main out repeat true while` As palavras reservadas devem ser especificadas como casos especiais de `palavra_reservada` (ou com outro nome) definida no trabalho nº1, sendo que a linguagem é *case sensitive*. Palavras diferentes das especificadas nesse item constituem erro léxico.
- Os símbolos especiais da linguagem 2023.2 são: `& | ! , ; = : ( ) { } == != < > + - * /` Símbolos diferentes dos especificados nesse item constituem erro léxico.
- Os comentários (de bloco e de linha) 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 de bloco que não seguem o padrão de formação especificado devem ser diagnosticados como erro léxico.
- As **mensagens de erro** devem ser conforme exemplos abaixo:
  linha 1: `@` símbolo inválido
  linha 1: `end` palavra reservada inválida
  linha 1: `_i9` identificador inválido
  linha 1: constante_`string` inválida
  linha 1: comentário de bloco inválido ou não finalizado

  Nos três primeiros exemplos, (1) 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; (2) a sequência `end` não é uma palavra reservada da linguagem, portanto caracteriza um erro léxico e deve ser apresentada na mensagem de erro, juntamente com a linha onde o erro foi detectado; (3) a sequência `_i9` não é um identificador da linguagem, portanto caracteriza um erro léxico e deve ser apresentada na mensagem de erro, juntamente com a linha onde o erro foi detectado. Nos dois últimos 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 nº1devem ser adaptadas à notação da ferramenta que será utilizada para gerar o analisador léxico; (2) os trabalhos desenvolvidos usando especificações diferentes daquelas

elaboradas pela equipe no trabalho nº1 receberão nota 0.0 (zero), sendo que qualquer alteração nas especificações feitas no trabalho nº1 devem ser acordadas com a professora; (3) o trabalho nº1 deve ser devolvido para professora, caso contrário, o analisador léxico não será corrigido.

- A implementação do analisador léxico, bem como da interface do compilador, deve ser disponibilizada no **AVA** na tarefa **"parte 2 - analisador léxico"**, aba **COMPILADOR**. Deve ser disponibilizado **um arquivo compactado** (com o nome: `lexico`, seguido do número da equipe) contendo: o projeto completo, incluindo **código fonte**, **executável**, **arquivo com as especificações léxicas** (no GALS, arquivo com extensão `.gals`) e demais recursos. Basta um integrante da equipe postar o trabalho.
- Na avaliação do analisador léxico serão levadas em consideração: a correta especificação dos *tokens*, conforme trabalho nº1; 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 24/09/2023 (domingo).

EXEMPLOS DE ENTRADA / SAÍDA

EXEMPLO 1: sem erro léxico

ENTRADA

```
linha
   1: [
   2: isso é um comentário
   3: ]
   4:
   5: main  _iarea   (
   6:
   7: 1.00
```

SAÍDA (na área de mensagens)

```
linha    classe                 lexema
5        palavra reservada      main
5        identificador          _iarea
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
   3: ]
   4:
   5: main  _iarea   @
   6:
   7: 1.00
```

SAÍDA (na área de mensagens)

```
linha 5: @ símbolo inválido
```