

TRABALHO FINAL – parte 2: implementação do analisador léxico

Implementar o **analisador léxico** de forma que identifique, nos programas escritos na linguagem 2022.2, 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 | <p>– 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:</p> <p><u>1ª situação</u>: programa compilado com sucesso</p> <ul style="list-style-type: none">✓ lista de tokens, na área reservada para mensagens, contendo, para cada <i>token</i> reconhecido, a <u>linha</u> onde se encontra, a sua <u>classe</u> (por <u>extenso</u>) e o lexema, nessa ordem, conforme exemplo;✓ mensagem (<i>programa compilado com sucesso</i>), na área reservada para mensagens, indicando que o programa não apresenta erros. <p>As <u>classes</u> possíveis para os <i>tokens</i> são: palavra reservada, identificador, constante <code>int</code>, constante <code>float</code>, constante <code>char</code>, constante <code>string</code>, símbolo especial.</p> <p><u>2ª situação</u>: programa apresenta erros</p> <ul style="list-style-type: none">✓ mensagem, na área reservada para mensagens, indicando que o programa apresenta erro. Neste caso, indicar a <u>linha</u> onde ocorreu o erro e a <u>descrição</u> do erro, emitindo uma mensagem adequada. Tem-se que:• constante <code>char</code> inválida – apresentar: a mensagem (<i>constante char inválida</i>) e a linha onde ocorreu o erro;• constante <code>string</code> inválida – apresentar: a mensagem (<i>constante string inválida ou não finalizada</i>) e a linha onde ocorreu o erro;• para símbolo inválido – apresentar: a mensagem (<i>símbolo inválido</i>), a linha onde ocorreu o erro e o símbolo não reconhecido;• para comentário de bloco inválido ou não finalizado – apresentar: a mensagem (<i>comentário de bloco inválido ou não finalizado</i>) e a linha onde ocorreu o erro, sendo que, nesse caso, deve ser apresentada a linha onde <u>inicia</u> o comentário. <p>As mensagens geradas por ferramentas, como o GALS, devem ser alteradas.</p> |

OBSERVAÇÕES:

- As palavras reservadas da linguagem 2022.2 são: `boolean break char do else end false float fun if int main print println readln string true val var while`. As palavras reservadas devem ser especificadas como casos especiais de identificador definido no trabalho nº1.
- Os símbolos especiais da linguagem 2022.2 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 nos exemplos abaixo:
 - Erro na linha 1 – constante `string` inválida ou não finalizada
 - Erro na linha 1 – constante `char` inválida
 - Erro na linha 1 – @ símbolo inválido
 - Erro na linha 1 – comentário de bloco inválido ou não finalizado

No 3º exemplo, o símbolo @ não é um símbolo especial da linguagem, portanto caracteriza um erro léxico. Nesse caso, o símbolo deve ser apresentado na mensagem de erro. Nos 1º e 2º exemplos devem ser apresentadas a linha e a mensagem de erro. Ou seja, as sequências não reconhecidas não foram apresentadas na mensagem de erro. Observa-se que para comentários de bloco inválidos ou não finalizados deve ser apresentada a linha onde inicia o comentário.
- As especificações feitas no trabalho nº1 (e já corrigidas) devem usadas para implementação do analisador léxico. Observa-se que essas especificações devem ser adaptadas à notação da ferramenta que será utilizada para gerar o analisador léxico. Além disso, trabalhos desenvolvidos usando especificações diferentes daquelas elaboradas pela equipe no trabalho nº1 receberão nota 0.0 (zero). Qualquer alteração nas especificações feitas no trabalho nº1 devem ser acordadas com a professora.
- O trabalho nº1 deve ser devolvido para a professora. Caso contrário, será atribuída nota 0.0 (zero) ao analisador léxico.

- A implementação do analisador léxico, bem como da interface do compilador, deve ser disponibilizada no **AVA** (na aba COMPILADOR / analisador léxico), na “**pasta**” da sua equipe. 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.
- 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 23/09/2022 (sexta-feira).

EXEMPLOS DE ENTRADA / SAÍDA

EXEMPLO 1: sem erro léxico

| ENTRADA | | SAÍDA (na área de mensagens) | | |
|---------|----------------------|------------------------------|--------------------------------|--------|
| linha | | linha | classe | lexema |
| 1 | [| 5 | palavra reservada | break |
| 2 | isso é um comentário | 5 | identificador | area |
| 3 |] | 5 | símbolo especial | (|
| 4 | | 7 | constante float | 1.1d10 |
| 5 | break area (| | | |
| 6 | | | | |
| 7 | 1.1d10 | | | |
| | | | programa compilado com sucesso | |

EXEMPLO 2: com erro léxico

| ENTRADA | | SAÍDA (na área de mensagens) | | |
|---------|----------------------|--------------------------------------|--|--|
| linha | | Erro na linha 5 - @ símbolo inválido | | |
| 1 | [| | | |
| 2 | isso é um comentário | | | |
| 3 |] | | | |
| 4 | | | | |
| 5 | break area @ | | | |
| 6 | | | | |
| 7 | 1.1d10 | | | |