Compilador fase 1: Análise léxica e sintática

O objetivo desse trabalho é implementar um Compilador com as fases de análise léxica e sintática para uma linguagem baseada na linguagem Pascal, mas com palavras reservadas em português, ou seja, a linguagem **Portugol**.

O Compilador para Portugol restringe a linguagem Pascal para ter apenas tipos **inteiros** e **lógicos** (**booleanos**), comandos condicionais (**se**) e repetição (**while**), também não temos a possibilidade de escrevermos funções e procedimentos nessa linguagem.

Na implementação do Compilador o analisador léxico deve atender as necessidade do analisador sintático. A interação entre o analisador léxico e o analisador sintático se dará por meio da função **`consome()`** (do analisador sintático) que realizará chamadas à função **`obter_atomo()`** (do analisador léxico).

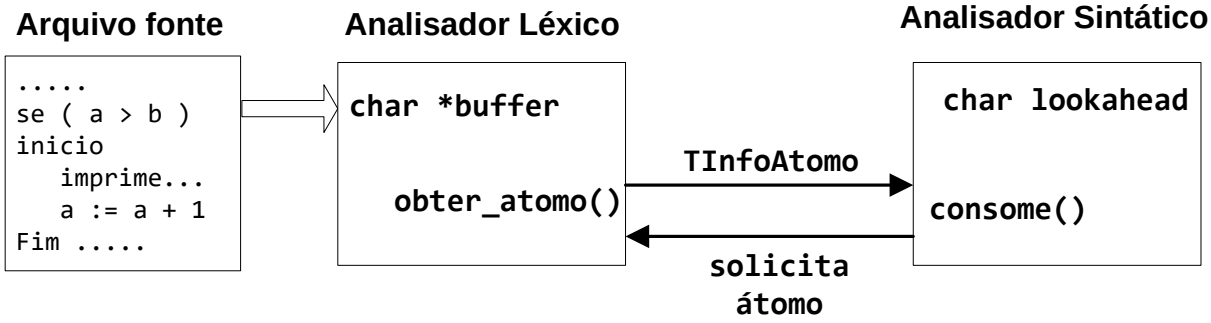

Figura 1: Interação entre Analisador Léxico e Sintático

Sintaxe da linguagem Portugol

A **sintaxe** da linguagem **Portugol** está descrita na notação **EBNF**, os `<não-terminais>` da gramática são nomes entre parênteses angulares **<** e **>** e os símbolos **`terminais`** (átomos do analisador léxico) estão em **`negrito`** ou entre aspas (Ex: **“;”**), a notação **{ α }⁺** denotará a repetição da cadeia α uma ou mais vezes **(α^+)**.

```
<programa>::= algoritmo identificador “;” <bloco> “.”

<bloco>::= [ <declaracao_de_variaveis> ] <comando_composto>

<declaracao_de_variaveis> ::= variavel {<lista_variavel> “:” <tipo> “;”}+

<lista_variavel> ::= identificador { “,” identificador }

<tipo> ::= inteiro | logico

<comando_composto> ::= inicio <comando> { “;” <comando>} fim

<comando> ::= <comando_atribuicao> |
              <comando_se>         |
              <comando_enquanto>   |
              <comando_entrada>    |
              <comando_saida>      |
              <comando_composto>

<comando_atribuicao> ::=  identificador “:=” <expressao>
```

```
<comando_se> ::= se “(” <expressao> “)” entao
                    <comando> [senao <comando>]

<comando_enquanto> ::= enquanto “(” <expressao> “)” faca <comando>

<comando_entrada> ::= leia “(“ <lista_variavel> “)”

<comando_saida> ::= escreva “(“ <expressao> { “,” <expressao> } “)”

<expressao> ::= <expressao_simples> [<relacional> <expressao_simples> ]

<relacional> ::= “<” | “<=” | “=” | “#” | “>” | “>=”

<expressao_simples> ::= [“+” | “–”] <termo> { (“+” | “–” | ou ) <termo> }

<termo> ::= <fator> { ( “*” | div | e )<fator> }

<fator> ::= identificador |
            numero        |
            verdadeiro    |
            falso         |
            “(” <expressao> “)”
```

Especificação Léxica

- **Caracteres Delimitadores** -> Os caracteres delimitadores: espaços em branco, quebra de linhas, tabulação e retorno de carro (`‘ ’`, `‘\n’`, `‘\t’`, `‘\r’`) deverão ser eliminados (ignorados) pelo analisador léxico, mas o controle de linha (contagem de linha) deverá ser mantido.

- **Comentários**: dois tipos de comentário, um começando com **`//`** e indo até o final da linha (1 linha) com o finalizador do comentário o caractere `‘\n’`. O outro começando com **`/*`** e terminando com **`*/`** (várias linhas), nesse comentário é importante que a contagem de linha seja mantida, além disso os comentários são repassados para o analisador sintático para serem reportados e descartados.

- **Palavras reservadas**: As palavras reservadas na linguagem **Portugol** são **strings em minúsculo**: **`algoritmo`**, **`variavel`**, **`inteiro`**, **`logico`**, **`inicio`**, **`fim`**, **`se`**, **`entao`**, **`enquanto`**, **`faca`**, **`leia`**, **`escreva`**, **`ou`**, **`div`**, **`e`**, **`verdadeiro`**, **`falso`**.
  Importante: Uma sugestão é que as palavras reservadas sejam reconhecidas na mesma função que reconhece os **identificadores** e deve ser retornado um **átomo específico para palavra reservada**.

- **Identificadores**: Os identificadores começam com uma letra minúscula ou maiúscula, seguido de zero ou mais letras minúsculas e/ou maiúsculas, dígitos ou caractere *underline* `‘_’`, limitados a 15 caracteres. Caso seja encontrado um identificador com mais de 15 caracteres deve ser retornado **ERRO** pelo analisador léxico. A seguir a definição regular para **identificadores**.
  `letra → a|b|...|z|A|B|...|Z`
  `digito → 0|1|...|9`
  `identificador` `→ letra(letra|digito|_)*`

- **Números**: No compilador teremos somente números inteiros, com seguinte definição regular abaixo:

  exponencial → (E|e)(+|-|λ)digito^{+}

  numero → digito^{+}(exponencial|λ)

Execução do Compilador

No Compilador quando for detectado um **erro sintático** ou **léxico**, o analisador deve-se emitir uma mensagem de erro explicativa e terminar a execução do programa. A mensagem explicativa deve informar a linha do erro, o tipo do erro (léxico ou sintático) e caso seja um erro sintático, deve-se informar a **linha do erro** e qual era o **átomo esperado** e qual foi o **átomo encontrado** pelo Compilador, por exemplo para o programa `exemplo1`:

Entrada compilador

```
1  algorimo exemplo1;
2       variavel maior:inteiro;
3  inicio
4      escreva(maior);
5  fim.
```

Saída do compilador:

```
#  4: Erro sintático: esperado [inicio] encontrado [end]
```

A seguir temos um outro programa em **Portugal** que lê uma dois números e encontra o maior, o programa a seguir está correto (léxico e sintático).

```
 1  /*
 2  programa le dois numeros
 3  inteiros e encontra o maior
 4  */
 5  algorimo exemplo2;
 6      variavel maior,n1,n2:inteiro;
 7  inicio
 8       leia(n1);
 9       leia(n2);
10       se( n1 > n2 ) então
11           maior := n1
12       senao
13           maior := n2;
14
15       escreva(maior) // imprime o maior valor
16  fim.
```

O para cada átomo reconhecido o compilador imprime as seguintes informações baseado nas informações contidas na estrutura **TInfoAtomo,** e ao final informa que a análise terminou com sucesso:

Saída do compilador:

```
#  1:comentario
#  5:algoritmo
#  5:identificador – atributo:exemplo2
#  5:ponto e virgula
#  6:variavel
#  6:identificador – atributo:maior
#  6:virgula
#  6:identificador – atributo:n1
..............
16 linhas analisadas, programa sintaticamente correto
```

Observações importantes:

O programa deve estar bem documentado e pode ser feito em grupo de até **2 alunos**, não esqueçam de colocar o **nome dos integrantes** do grupo no arquivo fonte do trabalho e sigam as **Orientações para Desenvolvimento de Trabalhos Práticos** disponível no **Moodle**.

O trabalho será avaliado de acordo com os seguintes critérios:

- Funcionamento do programa, caso programa apresentarem ***warning*** ao serem compilados serão penalizados. Após a execução o programa deve finalizar com **retorno igual a 0**;
- O trabalho deve ser desenvolvido na **linguagem C** e será testado usando o compilador do **CodeBlocks 17.12**.
- O quão fiel é o programa quanto à descrição do enunciado, principalmente ao formato de do **arquivo de entrada**;
- Clareza e organização, programas com código confuso (linhas longas, variáveis com nomes não-significativos, ...) e desorganizado (sem indentação, sem comentários, ...) também serão penalizados.