Tokens

- if
- then
- else
- while
- do
- let
- in
- begin
- end
- const
- var
- (
-)
- +
- -
- *
- /
- >
- <
- =
- ~
- :
- :=

- identifier

  pois não é possível inserir separadores entre os caracteres de um identifier;
- integer-literal

  idem

Portanto:

01 conjunto finito +
01 conjunto infinito +
01 conjunto infinito

- A Gramática Léxica deve representar o conjunto de todos os símbolos úteis (por meio do não-terminal Token) e também os símbolos inúteis (por meio do não-terminal Separator).
- A Gramática Léxica serve de base para construir o Analisador Léxico;
- A linguagem gerada pela Gramática Léxica é o alfabeto da Gramática Sintática;
- A Gramática Léxica descreve os símbolos (úteis e inúteis da linguagem);
- A Gramática Sintática descreve a ordem em que os símbolos podem ser usados.

Gramática Léxica

- Token ::=
  if | then | else | while | do |let | in | begin | end | const | var | (|) |
  + | - | * | / | > | < | = | ~ | : | := | Identifier |Integer-literal
- Identifier ::=
  Letter |
  Identifier Letter |
  Identifier Digit
- Letter ::=
  a | b | c | ... | z
- Digit ::=
  0 | 1 | 2 | ... | 9
- Integer-Literal ::=
  Digit |
  Integer-Literal Digit

- Separator ::=
  □ | \t | \n | Comment
- Comment ::=
  ! CommentLine EOL
- CommentLine ::=
  Graphic CommentLine
  | Graphic
- EOL ::=
  end-of-line character
- Graphic ::=
  any printable character or space

Suposições:
□ representa um espaço em branco
\t representa uma tabulação
\n representa uma mudança de linha

Gramática Sintática

Program ::=

single-Command

single-Command ::=

V-name **:=** Expression

| Identifier **(** Expression **)**

| **if** Expression **then** single-Command

else single-Command

| **while** Expression **do** single-Command

| **let** Declaration **in** single-Command

| **begin** Command **end**

Command ::=

single-Command

| Command **;** single-Command

Expression ::=
primary-Expression
| Expression Operator primary-Expression
primary-Expression ::=
Integer-Literal
| V-name
| Operator primary-Expression
| **(** Expression **)**
V-name ::=
Identifier
Operator ::=
+ | **-** | ***** | **/** | **<** | **>** | **=**
Declaration ::=
single-Declaration
| Declaration ; single-Declaration
single-Declaration ::=
const Identifier **~** Expression
| **var** Identifier **:** Type-denoter
Type-denoter ::=
Identifier