* Unitary steps:
  + tokens types step : one of the following: literal, char, int, double, string, ident, keyword
  + symbol (one character between ‘ ‘)
  + token (text between “ “)
  + multi-symbols (% + a string containing all possible symbols) : match the first symbom
  + multi-tokens ( syntax : ‘%(tok1, tok1, ...)’) : match the first token
* Keyword : (syntax ‘~keyword) or multi keyword (syntax ‘~(key1, key2, …)’) to match keywords.
* If an associated keyword procedure is defined it will be automatically executed)
* A keyword step could be considered as an unitary steps if it’s not associated to a keyword procedure

* Complex steps :
  + optional step : (syntax : '( <onestep> )' ) : an optional step
  + conditional accept step :  (syntax : '( <condition\_step> ;  )' ) : accept the current procedure it the condition step is accepted
  + steps sequence :
    - steps sequence (syntax : '{ <stepslist> }' ) : execute a set of steps consecutively, a sequence can be stopped or re-executed (see sequence control steps).
    - loop sequence (syntax : '{ <stepslist> } \* (<condition\_step>)' ) a steps sequence that will be re-executed if the condition step is accepted
    - repetitive sequence (syntax : '{+ <stepslist> }' ) : a steps sequence that will be automatically re-executed until the first break or accept step
    - conditional steps sequence (syntax : '[ <condition\_step>  <stepslist> ]' ) : a step sequence that will only be executed if the condition step is accepted
    - conditional loop (syntax : '[+ <condition\_step>  <stepslist> ]' ) : a loop sequence that will only be executed while the condition step is accepted
  + sequence control steps (only allowed for steps sequence)
    - break step : (syntax : '( <condition\_step> , )' ) : break a step sequence
    - continue step : (syntax : '( <condition\_step> \*  )' ) : re-execute a step sequence
* NB : a condition step should be an unitary step