1. Classification des termes

`var(` *?X*`)`
`nonvar(` *?X*`)`
`atom(` *?X*`)`
`integer(` *?X*`)`
`atomic(` *?X*`)` *X* est un atome *ou* un entier.

Cf. aussi `float/1`, `number/1`, `ground/1`...

2. Traitement des clauses comme des termes

La clause `Tete :- Queue.` est un terme prolog, que l'on pourrait écrire `:-(Tete,Queue)`.

`listing`, `listing(`*+Atome*`)`
`assert(` *:Clause*`)`
`clause(` *:Tete*, *?Corps*`)`
`retract(` *:Clause*`)`

Cf. aussi `asserta/1`, `assertz/1`, `retractall/0`, `abolish/2`...

3. (Dé)composition de structures

`functor(`*+Terme*, *?Nom*, *?Arité*`)`, `functor(` *?Terme*,*+Nom*,*+Arité*`)`
`arg(`*+ArgNo*,*+Terme*, *?Arg*`)`
+Terme `=..` *?Liste*, *?Terme* `=..` *+Liste*
`name(`*+Const*, *?ListeCar*`)`, `name(` *?Const*,*+ListeCar*`)`

Cf. aussi `atom_chars/2`, `number_chars/2`...

4. Comparaisons diverses

?X `=` *?Y*, *?X* `\=` *?Y*	Unification (ou non).
?Z `is` *X* (*X* expression arithmétique)	Évaluation arithmétique.
X `=:=` *Y*, *X* `=\=` *Y*	Égalité et différence *arithmétiques*.
Terme1 `==` *Terme2*, *Terme1* `\==` *Terme2*	Identité/différence *littérales*.

5. Modification du retour-arrière

`repeat`
`fail`