Travaux Pratiques 2 - Variables numériques & Récursivité
Programmation Logique sous l'environnement SWI-Prolog

Présentation des variables numériques :

En Prolog, l'écriture : **X=Y** peut avoir différents résultats :

- Si X et Y liées, répondra Yes si elles sont égales, No sinon.
- Si X libre et Y lié, proposera comme solution X valant la valeur de Y.
- Si X lié et Y libre, proposera comme solution Y valant la valeur de X.
- Si X et Y libres, erreur (car infinité de solutions)

Par contre l'écriture X=Y+1 (qui est équivalent à Y+1=X) va poser quelques problèmes. Il est tout d'abord obligatoire, dans cette écriture, qu'Y soit lié (Prolog ne sait pas inverser une équation, qu'elle soit simple comme ici ou plus compliquée). Mais ce n'est pas tout : = va simplement recopier l'équation (en ayant instancié les variables) :

Y=23, X=Y+1, write(X). affichera 23+1, et pas 24.

C'est pourquoi deux autres opérateurs sont définis :

- **is** calcule (on dit aussi "évalue") la valeur à droite (toujours liée) et instancie le résultat à la variable (libre) à gauche, ou la compare à la valeur liée de gauche. Par exemple X is 10+15 met 25 dans X, ou dit Yes si X vaut 25 (parce que avant on a dit X is 30-5 ou même X=25). Certains Prolog utilisent aussi l'opérateur :=
- **=:=** évalue les expressions à droite et à gauche et regarde si les résultats sont égaux (les deux arguments doivent être liés)
- Pour les autres comparaisons, les opérateurs (qui évaluent à droite et à gauche) sont :
  - =:= =\= (différent) < > =< >= ,
  - les opérateurs arithmétiques : + - * / mod ** , sin, cos, log,....

Prédicats prédéfinis :

Prolog définit plusieurs prédicats pour manipuler des variables numériques et réaliser des opérations arithmétiques. Parmi ces prédicats, on cite :

var(X) : pour tester si X est non liée,

nonvar(X) : pour tester si X est liée

plus(X,Y,Z) : est vrai ssi X, Y, Z entiers relatifs vérifient Z = X+Y

fois (X, Y, Z) : est vrai ssi X, Y, Z entiers relatifs vérifient Z = X*Y

div(X, Y, Z) : est vrai ssi X, Y, Z entiers relatifs vérifient Z = X/Y

between(X, Y, Z) : est vrai ssi X, Y, Z entiers relatifs vérifient X<=Z <=Y

Exercice 1 :

1. Ecrire un prédicat **mention(M)** qui affiche la mention d'un étudiant selon sa moyenne M donnée. (Note éliminatoire pour M=0, Médiocre pour M<10, Passable pour M<12, Assez bien pour M<14, Bien pour M<16, et Très bien pour M>=16)
2. Ecrire un programme qui affiche les solutions (X1 et X2) d'une équation de 2ème degré ($A*X^2 + B*X +C = 0$) en entrant les valeurs des variables A, B et C.
3. Écrire un programme qui pour un but **affiche1(N)** qui affiche les nombres de N à 0 dans l'ordre décroissant.
4. Écrire un programme qui pour un but **affiche(N, Ordre)** qui affiche les nombres de N à 0 dans l'ordre défini par le paramètre Ordre (croissant ou décroissant).
5. Ecrire un prédicat **puissance(X, N, P)** qui calcule X à la puissance de N et met le résultat dans P. ce prédicat accepte les 2ème et le 3ème paramètres comme des variables libres.
   Exemples :
   ?- puissance(2, 4, 16) → true
   ?- puissance(2, 4, P) → P=16
   ?- puissance(2, X, 16) → X=4
6. Ecrire un programme qui pour un but **fact(N, X)** qui affiche les factoriels de 0 à N et enregistre la factoriel de N dans X.

   Ex : ? - fact(3, X).
   0 ! = 1
   1 ! = 1
   2 ! = 2
   3 ! = 6
   X = 6

Exercice 2 :

L'objectif de cet exercice est de redéfinir des prédicats permettant de réaliser des opérations arithmétiques

- **plus(A,B,C)** qui réussit si C est égal à la somme de A et de B. il accepte au maximum une variable libre.

  ?-plus(10, 20, X). → X=30 ?-plus(10, X, 30). → X=20

?-plus(X, 20, 30). → X=10

?-plus(10, 20, 30). → true ?-plus(10, 15, 30). → false

- **fois(A,B,C)** qui réussit si C est égal au produit de A et B. il accepte au maximum une variable libre. Définir deux versions :
  - version 1 : en utilisant tous les opérateurs arithmétiques (+, -, *, /, //)
  - version 2 : en utilisant que le prédicat plus

?-fois(2, 3, X). → X=6 ?-fois(2, X, 6). → X=3

?-fois(X, 3, 6). → X=2

?-fois(2, 3, 6). → true ?-fois(2, 3, 10). → false

- **div(A, B, C)** qui réussit si C est égal au quotient de la division de A par B. il accepte seulement C comme variable libre. Définir ce prédicat en utilisant que le prédicat plus.

?-div(2, 3, X). → X=0 ?-div(10, 3, X). → X=1

?-div(2, 3, 1). → false

- **between(A,B,C)** qui réussit si C est compris entre les valeurs de A et B. il accepte seulement C comme variable libre.

?- between (2, 6, X). → X=2 ; X=3 ; X=4 ; X=5 ; X=6.

?- between (2, 6, 4). → true ?-between (2, 6, 10). → false