Programmation 3 : feuille 1

Premiers pas - Expressions et fonctions

Exercice 1 .1 *Symboles auto-évaluants*
Évaluer des entiers, des flottants, des rationnels, des complexes, des vecteurs, des caractères, des chaînes de caractères, les booléens.

Exercice 1 .2 *Premières évaluations d'expressions*
Évaluer :

1. $3 + 4$.
2. $3.2 * 4.5$.
3. $| - 5.3|/cos(4.3)$.
4. $3/4 - 4/5$.
5. $(1 + i3) * (2 + i4)$.

Exercice 1 .3 *Arguments facultatifs et/ou nombre arbitraire d'arguments*
Évaluer, et cela après en avoir vérifié la définition par :

- la documentation "on-line" associée aux symboles
  exemple : `(documentation '+ 'function)` ou `C-c C-d d` en donnant `+` comme argument)
- par la documentation `Hyperspec` :

1. $1 + 2 + 3 + 4$.
2. $maximum(1, 2, 3, 4, 5, 6, 7, 8, 9, 8, 7, 6, 5, 4, 3, 2, 1)$.
3. $log_3(4)$.

Deviner le résultat et tester les expressions suivantes :

1. `(+)`
2. `(*)`
3. `(/ 3)`
4. `(- 3)`

Exercice 1 .4 *Typage dynamique*
En Common Lisp, le typage des arguments est souvent dynamique : le type des arguments d'une opération est vérifié à l'exécution. Les vérifications sont faites à l'aide de prédicats prédéfinis de la forme <*type*>`p` (exemples : `stringp` pour une chaîne, `integerp` pour un entier).

Exercice 1 .5 *Définition de nouvelles fonctions*

1. Implémenter :
   (a) $x + 3$.
   (b) x^2.

(c) La valeur absolue.

(d) La moyenne de deux nombres.

(e) $\sum_{i=n}^{p} i$.

(f) Factorielle (version naïve récursive).

(g) PGCD (version récursive, méthode des soustractions successives).

(h) Fibonacci.

2. Provoquer des erreurs en transmettant des arguments non valides aux fonctions.
3. Utiliser les macros `trace`/`untrace`, pour tracer les appels récursifs de vos fonctions.

Exercice 1 .6 *Récursivité terminale*
Écrire une version récursive terminale de la fonction factorielle. Comparer ses limites avec celles de la version naïve.

Exercice 1 .7 *Documentation des fonctions*
Ajouter de la documentation aux fonctions.