

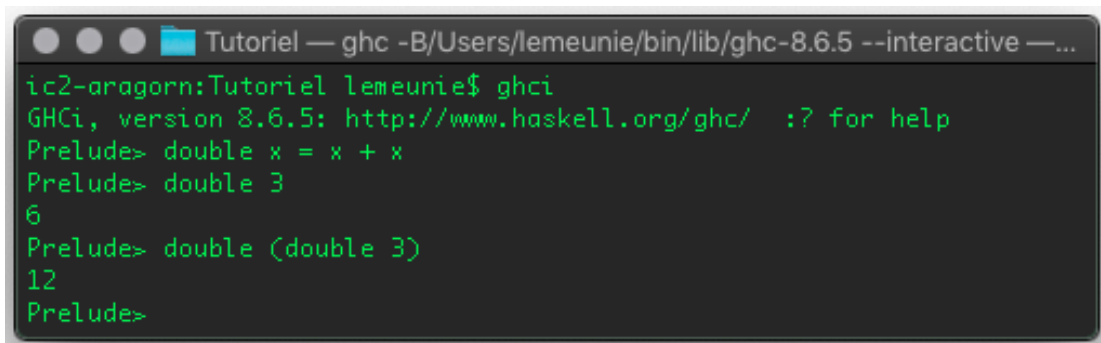
TP n°1

Objectifs du TP :

- Prendre en main l'interpréteur *ghci*
- Ecrire et appliquer des fonctions et des types de données simples

Exercice n°1 : prise en main de l'interpréteur

L'interpréteur Haskell disponible se nomme *ghci*. Il se lance depuis un terminal.



```
Tutoriel — ghci -B/Users/lemeunie/bin/lib/ghc-8.6.5 --interactive —...
ic2-aragorn:Tutoriel lemeunie$ ghci
GHCi, version 8.6.5: http://www.haskell.org/ghc/ :? for help
Prelude> double x = x + x
Prelude> double 3
6
Prelude> double (double 3)
12
Prelude>
```

Une aide en ligne est disponible dans l'interpréteur. Voici les principales commandes dont vous aurez besoin :

- **:?** => affiche une aide en ligne des commandes disponibles
- **:type *expr*** => affiche le type de l'expression *expr*
- **:info *expr*** => affiche des informations sur l'expression *expr*
- **:module +/- *nom*** => décharge ou charge un module dans l'interpréteur
- **:load *nom*** => charge un fichier ; le nom peut inclure un chemin
- **:quit ou ^D** => quitter l'interpréteur

L'interpréteur *ghci* se lance par défaut en chargeant le module Prelude qui définit un ensemble de types et de fonctions de base suffisant pour débiter.

Pour travailler durant les TP, il suffira de lancer l'interpréteur dans un terminal depuis un répertoire de travail contenant les fichiers que vous éditez. Vous pouvez éditer les fichiers dans n'importe quel éditeur Unix mais je recommande un bon éditeur comme par exemple Visual Studio Code.

Exercice n°2 : ma première fonction

- Editez dans un fichier *Qsort.hs* la fonction *qsort* de la diapositive 8 du cours.
- Chargez le fichier *Qsort.hs* dans l'interpréteur.

Rappel : pas de tabulation mais des espaces dans les fichiers sources

c) Vérifiez que les expressions de l'exemple suivant fonctionnent :

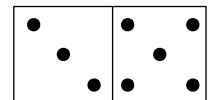
```
sources — ghc -B/Users/lemeunie/bin/lib/ghc-8.6.5 --interactive — bash
ic2-aragorn:sources lemeunie$
ic2-aragorn:sources lemeunie$ ghci
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
Prelude> :load Qsort.hs
[1 of 1] Compiling Main                ( Qsort.hs, interpreted )
Ok, one module loaded.
*Main> qsort [2,7,4,7]
[2,4,7,7]
*Main> qsort [2,3,4,5]
[2,3,4,5]
*Main> qsort []
[]
*Main> qsort [8,6,4,2]
[2,4,6,8]
*Main> 
```

Exercice n°3 : fonctions à 4 paramètres

- a) Définir une fonction qui, pour 4 nombres, renvoie True si les nombres sont égaux et False sinon.
- b) Définir une fonction qui, pour 4 nombres, retourne le plus grand des 4.

Exercice n°4 : poser des dominos

Tout le monde connaît le jeu des dominos : dans la variante classique du jeu, on forme des chaînes linéaires de dominos en réunissant des moitiés ayant la même valeur.



Vous utiliserez le type suivant : *type Domino = (Int, Int)*.

- a) Ecrivez une fonction qui détermine s'il est possible de former une chaîne linéaire avec deux dominos donnés. Par exemple avec 2-3 et 1-3 c'est possible, mais avec 2-3 et 1-4 c'est impossible.
- b) Ecrivez une fonction qui détermine s'il est possible de former une chaîne linéaire avec trois dominos donnés. Par exemple avec 2-3, 2-4 et 1-3 c'est possible mais avec 2-3, 2-4 et 1-2 c'est impossible.

Exercice n°5 : listes par compréhension

- a) Ecrivez une fonction *sumcarre* qui fait la somme des 100 premiers carrés d'entiers. On utilisera la fonction *sum :: [a] -> a* qui fait la somme des valeurs de la liste.
- b) Ecrivez une fonction *replic* qui retourne une liste contenant n répliques de x. Par exemples *replic 5 'a' = "aaaaa"* et *replic 5 1 = [1,1,1,1,1]*.

- c) Ecrivez une fonction *pyths* qui retourne la liste des triangles rectangles dont au moins un coté est de taille n donné.

Par exemples :

pyths 2 = []

pyths 5 = [(3,4,5),(3,5,4),(4,3,5),(4,5,3),(5,3,4),(5,4,3)]

pyths 10 = [(6,8,10),(6,10,8),(8,6,10),(8,10,6),(10,6,8),(10,8,6)]

Exercice n°6 : type de données algébriques simple

Dans cet exercice, on s'intéresse aux glaces à une boule, deux boules ou trois boules. Une boule correspond à un parfum : Chocolat, Vanille ou Framboise

- a) Ecrivez le type *Parfum*
- b) Ecrivez une fonction *prixParfum* qui donne le prix d'une boule selon le parfum : Chocolat à 1.5 €, Vanille à 1.2 € et Framboise à 1.4 €.
- c) Ecrivez le type *Glace*
- d) Ecrivez une fonction *prixGlace* qui donne le prix d'une glace selon sa composition. On utilisera la fonction *map :: (a->b) -> [a] -> [b]* qui applique la fonction sur chaque élément de la liste et retourne la listes des résultats de l'application.

Exercice n°7 : le palindrome et ma première fonction récursive

On considère qu'un palindrome doit être au moins constitué de deux caractères.

- a) Ecrivez une fonction *inverse* récursive qui inverse une liste. Par exemples *inverse [1,2,3] = [3,2,1]* et *inverse "toto" = "otot"*.
- b) Ecrivez une fonction *isPalindrome* qui test si une liste est un.

Par exemples : *isPalindrome "laval" = True*

isPalindrome "a" = False

isPalindrome [4,5,6,5] = False

isPalindrome [1,2,1] = True

- c) Ecrivez une fonction *doPalindrome* qui créer un palindrome à partir d'une liste. On utilisera la fonction *init :: [a] -> [a]* qui renvoie la liste sans son dernier élément.

Exemples : *doPalindrome [1,5] = [1,5,1]*

doPalindrome "lav" = "laval"

Références web

Site officiel Haskell

<https://www.haskell.org>

Site officiel des modules Haskell

<https://downloads.haskell.org/~ghc/latest/docs/html/libraries/>