Ocaml est un langage relativement récent, développé par l'INRIA. C'est un langage fonctionnel. Ce premier TD a pour but de vous familiariser avec les notions de base du langage Ocaml, et vous permettre dans un premier temps de créer des programmes simples.

Un programme Ocaml est une suite d'expressions. Un programme effectue le calcul d'une fonction *f* dont l'ensemble de départ *D* est un ensemble de données, et l'ensemble d'arrivée *R* est un ensemble de résultats :

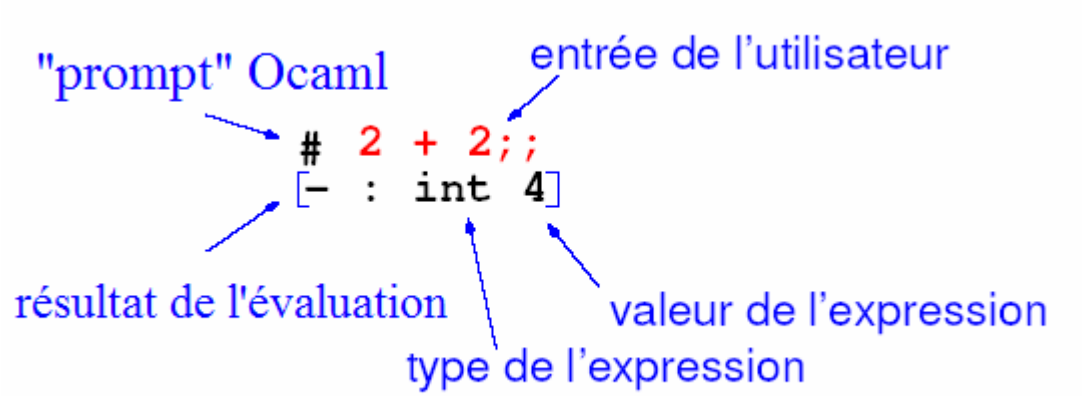
$$f : D \rightarrow R$$

Nous verrons dans les TDs les différentes expressions Ocaml pouvant constituer un programme fonctionnel.

1- Rappels

1.1 Exemple d'une expression Ocaml

Evaluation d'une expression.



Voici l'exemple d'un programme réalisant une addition de deux nombres entiers donnés. Le résultat est 4 de type **int** qui correspond à un entier. Ce programme est composé d'une seule expression dont le type est déduit de l'expression elle-même à l'aide des règles d'inférence de type. Cette expression est de type **int** car l'opérateur + est un opérateur sur les entiers et n'est pas liée à un nom (symbole -).

1.2 Déclarations globales et locales de variables Ocaml.

Une déclaration globale correspond à la liaison d'un symbole avec une valeur. La forme d'une déclaration globale est la suivante :

```
# let a=2008 ;;
val a :int=2008
# let b=a+1;;
val  b:int=2009    /* a et b sont 2 variables globales */
```

```
# let a=1 and b=2 in a*b ;;  /* a et b sont 2 variables locales  */
- : int = 2
```

Attention :
1 - Les noms de variables et de fonctions doivent commencer par une minuscule.
2 - Les déclarations locales permettent d'introduire des liaisons temporaires.

1.3 Calcul de la surface d'un rectangle

On considère un rectangle de longueur 6m et de largeur 3m. ***Il est demandé d'exprimer le calcul de la surface du rectangle par deux expressions Ocaml : l'une utilisant les données déclarées globalement et l'autre localement.***

2- Différents types de données

2.1 Booléens : Utilisés dans les expressions logiques

— Le type booléen (syntaxe Ocaml : bool) est composé des deux éléments true et false. A ce type de donnée, on associe des opérateurs logiques. Les principaux sont : non, et, ou qui se notent en Ocaml not, &&, ||.

a) Compléter la table de vérité suivante (en automatique, c'est la même chose) :

A	B	not(A)	not(B)	A && B	A || B
true	true				
true	false				
false	true				
false	false				

b) La table de vérité du ou exclusif (XOR) est :

A	B	XOR
true	true	false
true	false	true
false	true	true
false	false	false

Ecrire l'expression booléenne représentant l'opérateur XOR de A et B en utilisant les opérateurs booléens d'Ocaml.

c) Evaluer l'expression suivante:

```
#  let a= 4 and b=6 in not  (a>b) || (a<3)  &&  (b=3);;
```

2.2 Entiers relatifs, décimaux

Le type d'une expression en Ocaml caractérise l'ensemble de ses valeurs, l'ensemble des opérations que l'on peut faire sur ces valeurs et est représenté par ce qu'on appelle une expression type.
Voici un exemple sur les nombres entiers relatifs et les nombres décimaux.

Type du nombre	Entiers relatifs	Décimaux
Exemple	-2, 3, 0, 8, …	-1.0, 2.3, 0.0, 8.0, ..
Opérateurs arithmétiques	+, -, *, /	+., -., *., /.
Opérateurs logiques	<, <=, >, >=, =, <>	<, <=, >, >=, =, <>
Syntaxe en Ocaml	int	float

Attention : pour le type float, bien noter le **.** (point) après chaque opérateur arithmétique.

Les opérateurs logiques sont bien souvent utilisés lors d'expressions dites conditionnelles, c'est à dire lorsqu'on effectue une comparaison entre deux valeurs.

Voici un tableau récapitulatif :

- < : inférieur à,
- <= : inférieur ou égal à,
- > supérieur à,
- >= supérieur ou égal à,
- = égal à,
- <> différent de.

} Pour le type : bool

} Pour les types : int, float, string, char

Exercice 1 : Ecrire les expressions Ocaml qui permettent de calculer les expressions mathématiques suivantes, et donner les résultats qu'afficherait Ocaml.

1. faire la somme des nombres suivants : 2, 4.8, 3, 0.2
2. 4 * 5 + 2.5 * 2.5
3. 4.6/2 + 5.7 * 3

Exercice 2 : Vérifiez le typage des expressions Ocaml suivantes et si possible, fournir le résultat de l'évaluation produit par Ocaml.

```
# 4 + 5 * 9 ;;          # 7 / 3 + 9 / 2 ;;     # false &&(true || true) ;;
# (7/3 + 9) / 2 ;;      # 140 / 10 / 2 ;;      # false && true || true ;;
# 6 + (6.7 +. 3.3) ;;   # 140 / (10 / 2) ;;
```

3- Expressions conditionnelles

Syntaxe Ocaml : **if** condition **then** exp1 **else** exp2 **;;**

1- Evaluation de condition
2- Si condition est vraie alors évaluation de exp1
sinon évaluation de exp2

L'expression `"if ... then ... else"` renvoie un résultat de même type pour chacune des deux clauses : celle qui suit le `then` et celle qui suit le `else`.

Exercice 1 : Donner le résultat de l'évaluation des expressions suivantes**:**

(0) # if 2>3 then 3. else 0;;
This expression has type int but is here used with type float

(1)
```
# let a = if 3=4 then 0 else 5;;
```

(2)
```
# (if 3=4 then 0 else 5) + 8;;
```

(3)
```
# if (3=4) or (5<6) then 0 else 5;;
```

Exercice 2 :
a) Montrer que l'expression **if x then true else false** peut se simplifier en **x** (comparer l'évaluation de ces deux expressions).

Rq : dans la suite, on évitera d'écrire de telles expressions.

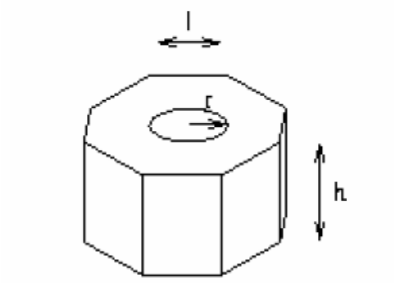
b) Réécrire **if a then b else false** par une expression booléenne équivalente.

Exercice 3 :
Vérifier la syntaxe, le typage et le cas échéant donner le résultat de l'évaluation.

```
# 34 > 12 ;;
# 3 < 5 <7 ;;
# (4. < 5.6) or (5.2 < 1.4) ;;
# 3 = 5-1 ;;
# 10 = 10.0 ;;
# if true then 23 else 0 ;
# if 4>5 then 3*5 else 3/2 ;;
```

4- Algorithmes

Exercice 1 : Calcul d'un volume (à préparer pour le TP1)
Calculer le volume d'un écrou à 8 pans de longueur *l (la base du triangle)*, évidé par un cylindre de rayon *r* et dont la hauteur est *h*.



Il faut décomposer ce problème en sous-problèmes faciles à résoudre :

volume de l'écrou = surface de l'écrou * hauteur de l'écrou
surface de l'écrou = surface de l'octogone – surface du cercle
surface de l'octogone = 8 * surface du triangle
surface du triangle = base du triangle * (1/2) * hauteur du triangle
hauteur du triangle = (base du triangle/2)/tangente (pi/8)

Traduire cette décomposition en un programme Ocaml.
(définir pi = 4*atan(1.) (arctangente) , tan (x) est la fonction tangente où x est de type float)

Exercice 2 : Conversion
- Convertir un temps donné en heures, minutes et secondes en secondes.
- Convertir un nombre de secondes en heures, minutes et secondes. Par exemple, si l'on a 4000s le résultat doit aboutir à ceci 1h, 6 mn et 40s (l'opérateur ***mod*** donne le reste de la division entière : 45 mod 7 retourne 3).