

---

# Interprétation et compilation

## TP 1 : Une calculatrice en OCaml

---

Dans ce TP :

- Commencer à se familiariser avec le langage de programmation OCaml, en écrivant un évaluateur d'expressions arithmétiques simples.

### Exercice 0.

Cas de base.

1. Dans le cas de base le plus simple, une expression arithmétique consiste simplement en un nombre (on se limitera aux entiers, `int` en OCaml).  
→ Définissez un type `expr` avec un seul constructeur `Num` qui gère ce cas de base.
2. On aura besoin d'une fonction d'affichage pour notre type `expr`.  
→ En utilisant la fonction `sprintf` du module `Printf`, écrivez une fonction `format : expr -> string`.  
On pourra alors l'utiliser en faisant par exemple `print_endline (format e)` dans le cas où `e` contient une valeur de type `expr`.
3. → Écrivez maintenant notre fonction `eval : expr -> int`.

### Exercice 1.

Somme.

1. À partir de deux expressions arithmétiques, on peut en construire une nouvelle en prenant par exemple la somme de ces deux expressions.  
→ Mettez à jour le type algébrique `expr` en ajoutant un constructeur `Add` qui permet d'exprimer la somme de deux expressions (indice : notre type `expr` devient récursif).
2. Si vous recompilez votre code, vous verrez qu'OCaml vous dit maintenant que vos fonctions `format` et `eval` sont invalides car le filtrage que vous y faites n'est plus exhaustif : de fait, il manque le cas du constructeur `Add` !  
→ Mettez à jour ces deux fonctions (indice : quand un type est récursif, les fonctions qui le manipule le sont généralement aussi).

### Exercice 2.

Différence, produit, quotient, modulo.

1. → Un à un, ajoutez ces différentes opérations à votre programme.
2. Est-ce que votre fonction `format` affiche correctement vos expressions ? Par exemple comment s'affiche `Mul (Num 3, Add (Num 9, Num 8))` ?  
→ Si il y a un soucis, corrigez votre fonction `format` pour ajouter les parenthèses manquantes.

### Exercice 3.

Bonus : éviter les parenthèses inutiles.

1. → Essayez de réécrire votre fonction `format` pour minimiser le nombre de parenthèses affichées.

Exemples à tester :

- `Add (Num 1, Add (Num 2, Num 3))` devrait s'écrire `1 + 2 + 3`;
- `Add (Num 1, Mul (Num 2, Num 3))` devrait s'écrire `1 + 2 * 3`;
- `Mul (Num 1, Add (Num 2, Sub (Num 3, Num 4)))` devrait s'écrire `1 * (2 + 3 - 4)`;