Les fonctions (méthodes et closures)

> La différence entre une méthode et une fonction sera évoquée au chapitre sur les implémentations

Définition d'une fonction

```
fn ma_super_fonction() {
    // Cette fonction ne retourne rien
    println!("Hello World");
}

fn addition(a: i32, b: i32) -> i32 {
    // Cette fonction retourne un i32
    let result = a + b;

    // Il suffit d'indiquer le résultat sans terminer par un point virgule pour retourner une valeur
    result
}

// On peut aussi créer des fonctions génériques en utilisant <T> pour définir n'importe quel type
fn foobar<L,R>(left: L, right: R) -> (Vec<L>,Vec<R>) {
    (vec![left], vec![right])
}
```

Définition et utilisation des closures

Les closures (ou "fermeture" en français) sont des fonctions anonymes qui peuvent être sauvegardées dans une variable et peuvent être passée comme argument à d'autres fonctions.

Contrairement aux autres fonctions les closures peuvent également capturer des variables du contexte dans lequel elle est exécutée.

```
fn main() {
    let addition = |x, y| {
        println!("Une addition est en cours...");
        x + y
    };
    println!("1 + 1 = {}", addition(1, 1));
}
```

Certaines fonctions vous obligeront à utiliser des closures dans leurs arguments. Cela est représenté par `Fn`, `FnOnce` ou encore `FnMut`

Egalement quand on déplace des données vers une nouvelle tâche, on peut utiliser le mot clé `move` en face de la closure pour donner l'appartenance de toutes les variables qu'elle utilise à la fonction.

```
fn main() {
    let hello = String::from("Hello ");

    // Les {} sont facultatif dans la closure quand il n'y a qu'une seule instruction comme ici
    // L'appartenance de y passe à notre closure
    let concat = move |x| hello + x;

    // On ne peut donc plus utiliser y
    println!("Cette commande génère une erreur, commentez là pour supprimer l'erreur: {}", y);

    // Utilisons notre closure
    println!("{}", concat("World"));
}
```

Revision #1
Created 27 April 2023 06:10:34 by SnowCode
Updated 27 April 2023 06:12:49 by SnowCode