TD 4 - Javascript, Rust - First-class functions, Closures

Hugo Bolloré[1]

[1]hugo.bollore@uvsq.fr

Dans ce TD nous utiliserons plusieurs langages, le Javascript et le Rust.
Pour le langage Javascript, pour rappel vous avez 2 choix :

1. Dans votre navigateur, allez sur le site https://playcode.io/javascript
2. Téléchargez le package *nodejs*, écrivez le code dans un fichier script_name.js et le lancer avec :

   ```
   node script_name.js
   ```

Pour langage Rust :

1. Dans votre navigateur, allez sur le site https://play.rust-lang.org
2. Suivez les indications sur cette page pour installer un environnement de développement pour Rust https://www.rust-lang.org/learn/get-started

Exercice 4.1 First-class

Question 4.1.1 First-class citizens

En Rust, les fonctions sont des objet de première classe, modifier le code suivant pour que la fonction *example* soit affectée à une variable puis appelée via celle-ci plutôt que directement.

```rust
/* "fn" keyword for functions
- example is the name of the function,
- n is the first parameter name, its type is set using
  - ":" keyword followed by the type: i32 for integer 32 bits
- "->" keyword allows to specify return data type*/
fn example(n: i32) -> i32 {
    /* "return" keyword is optional:
    if missing the last statement is considered as the return
    statement*/
    n+42 // return n + 42
}

fn main() {
    /* Let is the keyword to declare a variable, by default they
    are immutable*/
    let ie_7 = example(7);

    /* println operates in the same manner as C printf except you
    don't have to specify the parameter type for standard
    transformations
    - "{}" are replaced by parameters in the order given
    - "{var_name}" can be used to directly refence a variable*/
    println!("{} < {} < {ie_7}", example(5), example(6));
}
```

Question 4.1.2 First-class functions

En reprenant le code précédent, créez une fonction *mul_by_two* qui permet de multiplier par 2 un entier et passez cette fonction en paramètre de la fonction *example*. La fonction *example* devra être mise à jour pour appeler cette fonction sur la variable n avant de retourner le résultat additionné de 42. La syntaxe pour passer en paramètre une fonction en Rust dépend de plusieurs paramètres :

- Si vous souhaitez passer un pointeur de fonction :

  ```
  <param_name>: fn(i32) -> i32
  ```

- Si vous souhaitez passer une fonction, vous devez spécifier deux choses :

  1. Le type de fermeture :
     - Le mot-clé *FnOnce* s'applique aux fermetures qui ne peuvent être appelées qu'une seule fois,

       ```
       <param_name>: FnOnce(i32) -> i32
       ```

     - Le mot-clé *FnMut* s'applique aux fermetures qui peuvent modifier les valeurs capturées,

       ```
       <param_name>: FnMut(i32) -> i32
       ```

     - Le mot-clé *Fn* s'applique aux fermetures qui ne modifient pas les valeurs capturées ou qui ne capturent aucune valeur.

       ```
       <param_name>: Fn(i32) -> i32
       ```

  2. Le type de dispatch :
     - Statique :

       ```
       <param_name>: impl Fn(i32) -> i32
       ```

     - Dynamique :

       ```
       <param_name>: &dyn FnMut(i32) -> i32
       ```

Question 4.1.3 ??

Après cette modification, comment qualifieriez vous la fonction *example* ?

Question 4.1.4 Anonymous functions

Commentez la fonction *mul_by_two* et faites la fonction anonyme équivalente lors des appels à la fonction *example*. La syntaxe en Rust pour déclarer une fonction anonyme est la suivante :

```
toto(18,|n: i32| n * n * n);
```

Les paramètres de la fonction anonyme sont contenus entre les | et le corps de la fonction est défini juste après.

Exercice 4.2 Closures en Javascript

Dans le langage Javascript, une fermeture est automatiquement créée quand une fonction en retourne une autre, par exemple :

```
cookBanana=function(preparation, cuisson){
    return function(ingredients){
        return preparation + ", " + ingredients + ", " + cuisson;
    }
}

recette=cookBanana("éplucher banane", "aucune cuisson");
console.log(recette("ajouter chocolat fondu et crème chantilly"));
console.log(recette("ajouter miel et amandes grillées"));
```

Ici la variable cookBanana contiendra une fermeture qui est utilisée par la suite pour présenter plusieurs recettes.

Question 4.2.1 Génération de code html

En vous inspirant de l'exemple précédent et d'un exemple de table en html (link to an html table example), écrivez un code Javascript qui permet de générer une table html de 2 colonnes et 3 lignes et qui utilise exclusivement des fermetures pour ajouter les balises html aux éléments du tableau.

Exercice 4.3 Closures en Rust

Dans le langage Rust, la création d'une fermeture peut se faire en dehors d'une fonction imbriquée et la syntaxe est la même que celle utilisée pour les fonctions anonymes vues précédemment.

```
let print_text = |x| println!("Text: {x}");
```

Vous pouvez également utiliser un format multi lignes :

```
let print_text = |x| {
    println!("Text: {x}");
};
```

Question 4.3.1 Closure sans capture

Écrivez une fermeture qui multiplie deux valeurs entières et retourne le résultat.

Question 4.3.2 Closure avec capture

Écrivez une fermeture qui affiche la valeur d'une variable string définie en-dehors de la fermeture.

```
let my_string = String::from("I am your father!");
```

Question 4.3.3 Closure avec capture qui modifie une variable

Écrivez une fermeture qui modifie puis affiche la valeur d'une variable string définie en-dehors de la fermeture. Vous aurez besoin du mot-clé *mut* qui permet de spécifier qu'une variable pourra être modifiée :

```
let mut my_string = String::from("I am your father!");
```

Question 4.3.4 Closure avec capture qui déplace une variable

Écrivez une fermeture qui affecte la valeur d'une variable string définie en-dehors de la fermeture à une variable locale à la fermeture puis qui affiche la valeur de cette variable locale.

En dehors, et après la définition de la closure, essayez d'afficher la variable string d'origine. Que se passe t-il ? Regardez les explications du compilateur, consultez la page d'erreur qu'il vous indique, expliquez quelle est la fonctionnalité de Rust qui cause ce comportement et mettez à jour le code pour éviter ce problème.