Seconde-session blanc 2021-2022: Programmation avancée

Axel Viala <axel.viala@darnuria.eu>

4 mai 2022

Nom et Prénom : Numéro étudiant :

Objectifs : La clarté des réponses sera appréciée, veillez à écrire soigneusement. Les questions portent sur le langage Rust. Notes de cours autorisé. Réponse sur une copie a part encouragée. Ce sujet blanc ne comporte pas la question de composition et les questions sont "un peu" plus dures que sur le sujet prévu.

1 Généralités

1. Par défaut doit t'on toujours écrire les types explicitement partout en Rust
   - ◯ Non dans le corps des fonctions ne n'est pas neccessaire
   - ◯ Il faut écrire les types partout aucune inférence
2. En Rust une référence peut-elle pointer sur rien ? :
   - ◯ Une référence peut pointer sur rien temporairement
   - ◯ Les références doivent toujours référencer quelque chose
3. Dans ce code, `m` est passé comment ?
   - ◯ Par deplacement *move*
   - ◯ Par référence *borrow* (mut/immutable)
   - ◯ Par copie *copy*

```rust
struct Point { x: i32, y:i32 }
fn mystere(m: Point) -> Point {
    let mut m = m;
    m.x += 1;
    m.y += 42;
    m
}
```

4. Anatomie d'un code Rust : associez les termes suivants au code suivant :

— Opérateur d'addition
— Nom de variable
— Mot clef de déclaration de variable
— Argument de fonction
— Type
— Mot clef de déclaration de fonction
— Opérateur de d'enchaînement d'instruction
— Bloc du corps de la fonction
— Nom de fonction
— Appel de fonction associée à un type
— Argument de fonction passé en appel
— Expression du if

```rust
fn inconnue(a: &[i32], i: usize) -> Option<i32> {
    if a.len() < i {
        Some(a[i] + 50)
    } else {
        None
    }
}
```

5. Que signifie `T` dans la signature de la fonction `fn mystere<T>(a: T)`.

6. Ce code peut-il compiler ? Justifiez.

```rust
fn mystere(a: &mut [i32], b: &i32) {
    a[0] += *b;
}

fn main() {
    let mut a = [1, 2];
    mystere(&mut a, &a[0]);
}
```

7. Expliquez ce qu'est un `trait` en Rust.

8. A quoi sert le type `Option<T>` et le type `Result<T, E>`

9. Proposez votre implementation de `MyOption<T>` qui fait comme `Option`, Et implementer la fonction `MyOption::map` qui dois faire comme la documentation de `Option::map` de la lib standard.

10. `u32` est il passé par copie ou par move par défaut.

11. Ce code comporte une erreur, laquelle justifiez. Que fait `?`.

```
fn mystere(a: Option<u32>) -> Option<u32> {
    let a = a?;
    Some(a + 1);
}
```

12. Pour un point realisez une implementation du trait `Display` et `Add` entre deux `Point`.

```
struct Point {
    x: f32,
    y: f32,
}
```