TP0x0: Hello Rust, Hi Cargo

Florent Becker

1 La pluie et le beau temps

Cet exercice utilise le code contenu dans le dossier `meteo` de l'archive Célène.

Question 1

Quelle commande commande vous permet de lancer le code fourni par votre enseignant? Où est situé le point d'entrée du programme qui vous est fourni?

Question 2

Quelle commande `cargo` vous permet de compiler le projet sans le lancer?

Question 3

Où est placé l'exécutable produit par la compilation?

Question 4

Quelle commande `cargo` permet de lancer les tests? Combien y a-t-il de tests dans ce projet, où sont-ils situés? Quels sont les tests qui échouent, pourquoi?

Question 5

Quelle commande `cargo` permet de générer la documentation de ce code? Où cette documentation est elle placée?

Question 6

Quels sont les types déclarés dans le fichier `src/main.rs`, avec quels mots clés sont-ils déclarés. Quelle est la différence entre une déclaration `struct MonType` et une déclaration `enum MonAutreType`?

Question 7

Dans la fonction `main`, créer les instances du type `Meteo` correspondant à la météo d'aujourd'hui et de demain à Orléans et à Perpignan.

Question 8

Ajouter les cas manquants à la fonction `que_porter` de sorte à faire passer le test associé.

Question 9

De même, faire passer le tests associé à la conversion en degrés Farenheit.

Question 10

Pourquoi utilise-t-on un attribut `alerte: Option<Alerte>` plutôt qu'un attribut `alerte: Alerte` dans la structure `Meteo`?

Question 11

Pour les jours de pluie ou de neige, ajouter un affichage de la hauteur de précipitations attendue, avec docmuentation et test.

Pour générer des nombres aléatoires, on utilise en Rust la bibliothèque (ou *crate*) `rand`.

Question 12

Ajouter la bibliothèque `rand` aux dépendances du projet (dans `Cargo.toml`).

Question 13

Implémenter une fonction `meteo_aleatoire() -> Meteo` qui renvoie une instance de `Meteo` tirée au hasard.

2 Le jugement majoritaire

Dans le repertoire `sondage` de l'archive Célène, vous trouverez le squelette d'une application web de sondages qui implémente le jugement majoritaire. Vous pouvez en trouver une explication détaillée sur https://fr.wikipedia.org/wiki/Jugement_majoritaire.

L'application a déjà été écrite par votre enseignant, elle est accessible via l'url https://tausendblum.site/jugement_maj. Chaque sondage consiste en deux pages, l'une pour voter: https://tausendblum.site/jugement_maj/exemple/voter et l'autre pour afficher les résultats: https://tausendblum.site/jugement_maj/exemple/resultats. La partie que vous allez *ré*-implémenter est celle qui détermine les résultats du sondage à partir des réponses des participants. Ces résultats consistent en:

- la note médiane de chaque proposition
- l'identité de la proposition ayant recueilli la meilleure note médiane.

Ces fonctions sont situées dans le module `traitements.rs`.

Attention, une option pour laquelle personne n'a encore donné d'avis n'a **pas** de note médiane.

De même, tant que personne n'a répondu au sondage, il n'y a **pas** d'option gagnante.

Les scores obtenus par chacune des réponses du sondage sont de la forme «3 réponses très bien, 2 bien et 1 médiocre». Ils sont représentés par le type `Histogramme`. Comme il y a 7 notes possibles, on les représente par un tableau de 7 entiers non signés (`[u32; 7]`). Chaque note correspond donc à un *indice* dans ce tableau.

Question 14

Quel est le type (arguments et valeur de retour) de la fonction `traitements::mediane()`?

Question 15

Implémenter `traitements::mediane`

Vous allez avoir besoin de faire une boucle sur les fréquences de l'histogrammes en conservant l'indice. Vous pouvez:

- soit garder une variable `indice_courant` incrémentée à chaque tour de boucle. Il faut alors que cette variable soit déclarée avec une instruction `let mut indice_courant = 0` pour pouvoir être modifiée ensuite.
- soit utiliser une boucle sur `h.frequences.iter().enumerate()` qui vous permet d'itérer sur des couples `(indice, valeur)`, comme `enumerate` en python.

Question 16

De même, implémenter `meilleure_option`

3 Le Top 10

On veut réaliser une application de Top 10.

Question 17

Initialiser un nouveau projet `cargo`.

Question 18

Créer un type `Chose` pour représenter les objets à classer: ceux-ci ont au minimum un titre et une note (un flottant, qui sera compris entre 0 et 100).

Question 19

Quel type de données permet de représenter un tableau de `Chose` dans lequel on peut insérer et retirer des objets? Comment en créer un vide? Comment y insérer des éléments? Comment le créer avec ses éléments?

Question 20

Implémenter le votre algorithme de tri favori; créer une petite application qui affiche le top 10 d'une liste de `Choses` définie en dur dans votre programme.

Il est possible que certaines `Choses` n'aient pas de note définie; dans ce cas, pendant le déroulement du tri, on va demander de choisir interactivement laquelle de deux `Choses` est la préférée.

Question 21

Modifier la structure de donnée `Chose` pour que la note puisse être absente `None`; modifier le jeu de données d'entrée pour que certaines `Choses` n'aient pas de note déterminée au départ.

Question 22

Adapter votre algorithme de tri pour fonctionner avec ces nouvelles `Choses`. Quand les deux `Choses` à comparer n'ont pas la même note, on doit demander interactivement laquelle est préférée.

Vous pouvez récupérer une réponse depuis l'entrée standard avec le code suivant:

```
let stdin = std::io::stdin;
let reponse : String;
println!("Une question");
stdin.readline(&mut reponse).unwrap();
```

Après l'exécution de ces lignes, la variable `reponse` contient la réponse donnée. On peut la débarrasser des espaces / retour lignes initiaux et finaux en utilisant `reponse.trim()`.

Il peut arriver que cette version de l'algorithme pose des questions inutiles, comme dans le scénario suivant:

— Lequel préférez-vous: l'éléphant (note: 7) ou l'hippopotame (note inconnue)?
— L'hippopotame !
— Lequel préférez-vous: l'écureuil (note: 6.5) ou l'hippopotame (note inconnue)?
— À ton avis? L'hippopotame!

Pour éviter celà, il va falloir mettre à jour la note à chaque question. Ainsi, une `Chose` peut avoir une note qui est soit:

- connue, avec une valeur flottante,
- inconnue, sans valeur associée,
- minorée, avec une valeur flottante: on sait que la note est au-dessus de cette valeur
- majorée, avec une valeur flottante: on sait que la note est en-dessous de cette valeur
- bornée, avec deux valeurs flottantes: on sait que la note est entre ces deux valeurs.

Question 23

Implémenter cette nouvelle version de l'application