RAPPORT DE PROJET :

RUST : BIG PROJECT

RÉALISÉ PAR :

BOUHADOUN Jugurtha

BARR Mohamed Anis

Master Intelligence Artificielle et Big Data

Année universitaire 2019-2020

1 Introduction

Ce projet s'inscrit dans le cadre de la validation du module RUST et afin de se familiariser avec le langage RUST . Les sujets traité sont :

- Traitements des images de type PPM
- Importation d'une libraires externe d'un autre langage "C"
- Utilisation des fonctions RUST dans un autre langage "PYTHON"

Le projet consiste à résoudre différents exercices à l'aide du langage RUST, il faut écrire du code et présenter un github.

1.1 Lvl 0 - Warmup

Pour ce niveau le but était de créer un nouveau projet, et rajouter nos nom dans authors de toml
<authors = ["AnisBarr <barrma@hotmail.fr> ", "JugurthaBouhadoun <jbouhadoun@gmail.com >"]>
et s'assurer que le cargo build fonctionner vraiment.
Pour cette partie on a pas vraiment rencontré de difficulté.

1.2 Lvl 1 - Struct

Pour ce niveau le but était de créer une structure Pixel nouveau projet, et l'implémentassions des fonctions de cette structure.

- fn new(red : u8, green : u8, blue : u8 -> Self
- Clone et Copy
- fn display(self) -> String
- fn invert(mut self)
- fn eq(self, other : Pixel) -> bool
- grayscale

Pour cette partie on a pas vraiment eu de problème pour le codage, on a fait quelques recherche pour Eq et display,On a rencontré une petite difficulté avec grayscale puisque on somme des U8 , et au moment du test si on passer un pixel dont la somme est supérieur à 255 la fonction cracher . donc on a caster à U32 notre somme puis on devise par 3 et on recaste à U8 et la ça marche .

1.3 Lvl 2 - Image manipulation

Pour ce niveau le but était de créer une structure Image en utilisant notre structure PIXEL crée auparavant, on a rajouté deux champs en plus de ceux qui sont demandés :

- pub header : String ,
- pub maxColor : u8,

et l'implémentassions des fonctions :

- fn newWithFile(filename : Path) -> Image
- fn save(filename : Path)

Pour cette partie la première difficulté qu'on a rencontré c'était de comment parser un fichier en RUST, après quelque recherche on a trouver une solution et on a finalement pu enregistré notre fichier dans une STRUCTRE IMAGE et dans cette partie là on a eu pas mal d'erreur de typage qu'on a pu résoudre petit à petit avec le débeuguer.

1.4 Lvl 3 - Benchmarks

Pour ce niveau le but était de créer de créer des test unitaires, vu qu'on a vu ça en cours et on avait déja fait des test unitaires en langages "C" dans d'ancien projet donc c'etait pas trop dur pour nous.

1.5 Lvl 4 - Unsafe - Optional C fun with Rust

pour cette partie le but c'était d'utiliser une librairie "C" déjà programmer pour effectuer des traitements sur les PPM. Donc on a choisi de suivre de stackOverflow, pour cela on a télécharger nos fichier ppmaio.c et ppmaio.c on a créer notre **libppmio.a** qu'on comme notre LIB C, on a crée un build.rs pour crée un lien entre notre lib C et notre fichier Rust. ensuite dans main.rs on déclarer les fonctions qu'on aller réutiliser dans <extern "C">.

- fn ppmawritetest (fileoutname : *const u8) ;
- fn ppmareadtest (inputname : *const u8) ;

1.6 Lvl 5 - Unsafe - Optional C fun with Rust

Pour cette partie on doit lancer des fichiers rust en python
lib

name = "src"

path = "src/main.rs"

crate-type = ["dylib"]

on ecrie notre fonction dans le fichier main.rs on rajoutant [no mangle]

juste avant la fonction

après avoir fait un build on trouve un fichier .so qu'on load ensuite dans pyhton avec from ctypes import cdll

l

ib = cdll.LoadLibrary("../target/debug/libsrc.so") puis il nous suffit juste de lancer la fonction souhaitée dans le python