Taller de Rust, sesión 5

- [x] enum Option
- [x] match + enum
- [x] struct con derive, derivation
- [x] collections → HashMap

Ejercicio propuesto

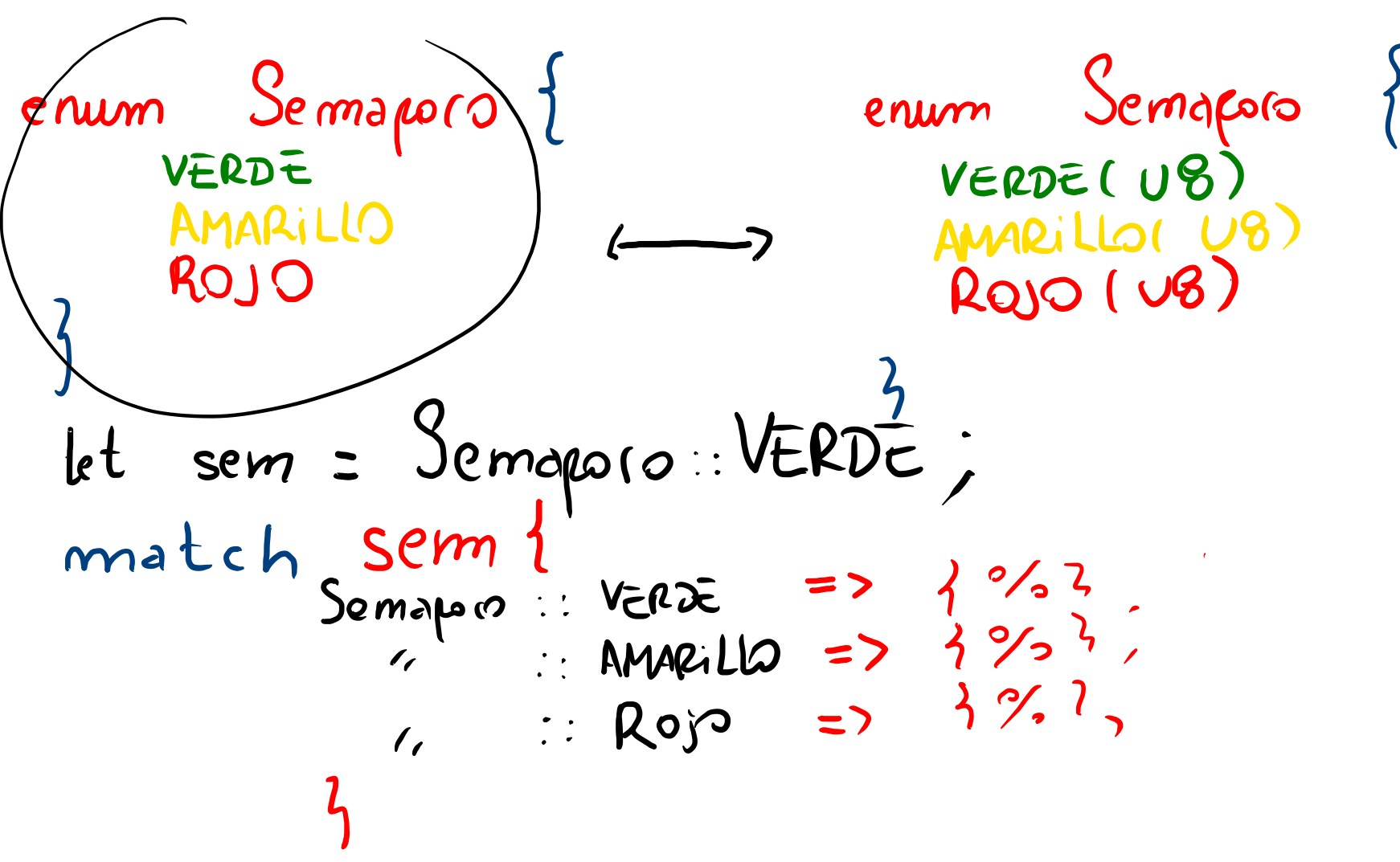
Result < Ok , Err >
T T

- campos opcionales

valor o None

Option { Some (T)
None

fn jugar (valor : Option (String)) { }




```
let sem = Semaforo::VERDE;
match sem {
    Semaforo::VERDE    => {%},
    ''     ::AMARILLO  => {%},
    ''     ::Rojo      => {%},
}
```

struct ⟷ clases

```
struct Calle {
    nombre: String,
    comuna: String,
    p0 : Point
    p1 : Point
}
```

GIS ~ OpenStreetMap.

Calle ∝ Semaforo.

```
match sem {
    Verde    => println("La calle {} está en verde", calle),
    Amarillo =>
    Rojo     =>
}
```

```
println!("La calle {} está en verde", calle);
```

↳ trait Display

```
println!("La calle {:?} está en verde", calle);
```

↳ trait Debug

std :: ~~Libreria~~ estandar

Biblioteca estándar

se declaran

∃ elementos llamados trait

'rasgos' o 'características' → comportamientos

Estructuras

struct

enum

union (unsafe)

← se implementan

bib. estándar

Display, Debug, Eq, PartialEq
Ord, Hash

Ofrecen una forma por DEFECTO
que se implementa en el objeto

(enum o struct)

mediante derive

↳ `#[derive(<trait>, ...)]`

ejemplo ::

```
trait Electronico {
    fn encender() {}
    fn apagar() {}
}
```

Dada una estructura ⇒ implementacion trait en estructura

---

patron × depeto

macro derive

```
impl Electronico por Semaforo {
        fn encender() { print("Esta encendido"); }
        fn apagar() { print("Se apaga"); }
}
```

¿Podemos usar derive en 'Electronico'?

No ⟶

derive "patrón x defecto" → se puede usar

```
#[derive(Debug)]
struct Calle {
    :
}
println!("{:?}", calle);
```

→ derivative → con + traits de std.
(crate)

std :: collections

vector → vec![...];

Diccionario → (llave, valor)

Diccionario → HashMap

llave → Hashable → trait Hash → generador de nº Hash

valor → T

generador nº Hash

"Perro" → 100
"Gato" → 150
⋮
"Vaca" → 300

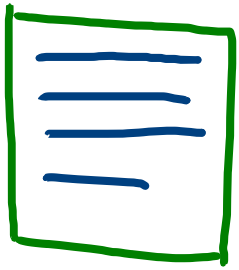
minimizar la colisión ↕ (nº = pasa valor de llave ≠)

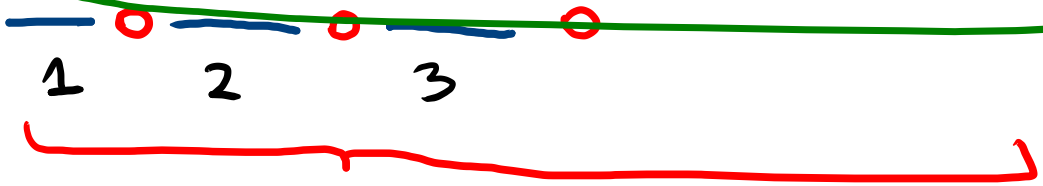
fn Hash

Revisión ejercicio propuesto

CANONICA

comando -F -C 1,2,3 2,3 0 campos 1,2,3 2,3 toda la pista (1)





AWK.

stream

comando <ops> < archivo \ (2)

accion 1 | accion 2 | comando <ops> (3)

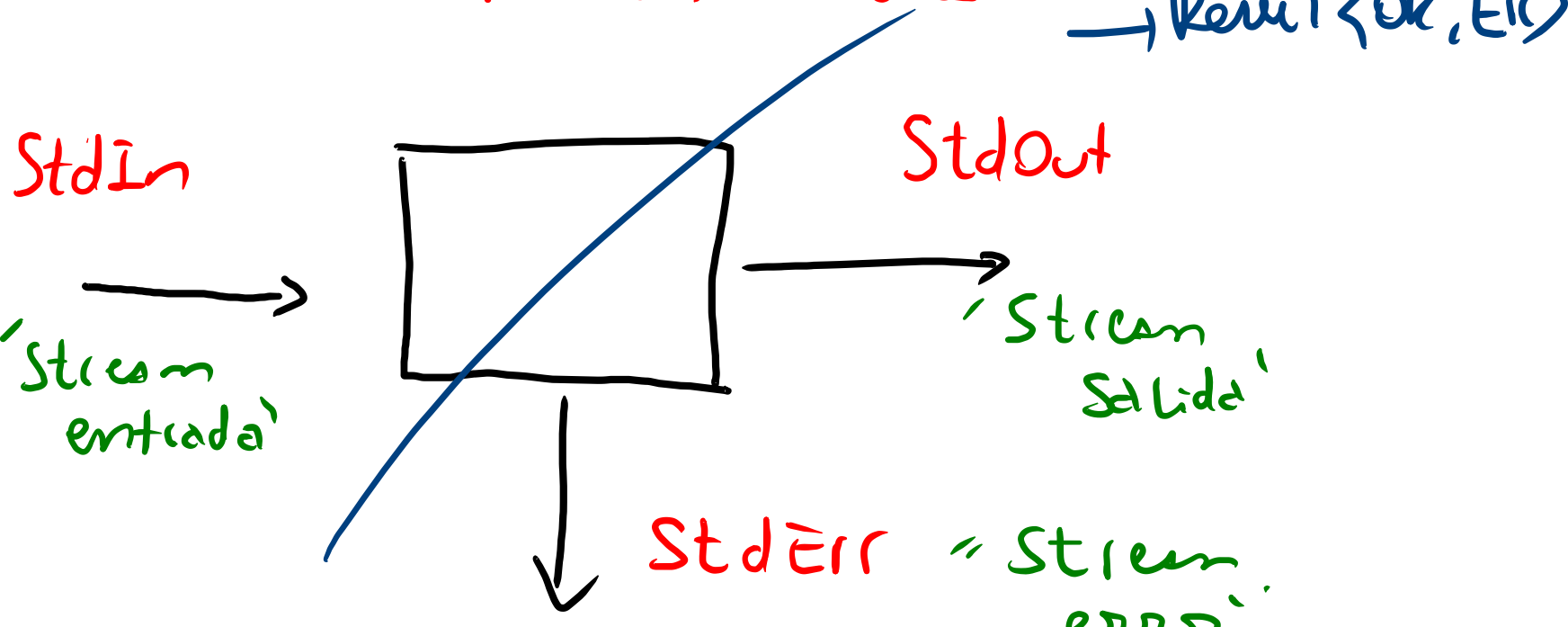
---

awk -F';' '{ print $1, $2, $3 }' archivo

Posix

comando <opciones> ARCHIVO

<opciones> StdIn → Result<Ok, Err>



comando