Estructuras de Tipo Registro

1.

1.1. Introducción

Una estructura de tipo registro se define por la lista de campos que la componen de esta manera:

```
<nombreDelCampo> = <valorDelCampo>
```

separados por coma (,) y encerrados entre llaves ({ y }).

El `<nombreDelCampo>` es el identificador del campo.

El tipo de dato de la estructura también se denota entre llaves, con la lista de sus elementos separados por coma de la siguiente manera:

```
<nombreDelCampo>: <tipoDeValorDelCampo>
```

Ejemplo: *Diseñemos una estructura que pemita representar estudiantes. Los campos necesarios son:*

1. *Un entero ID, que representa el "número de estudiante"*
2. *Una cadena NOMBRE, el nombre del estudiante*
3. *Una lista de cadenas que se llamará CURSOS, que representa los cursos a los cuales el estudiante está inscripto.*

```
- val Estudiante1 = {
      ID = 123,
      NOMBRE = "José Perez",
      CURSOS = ["AM1", "ALG2", "INF"]
};

val Estudiante1 = {
      CURSOS = ["AM1", "ALG2", "INF"],
      ID = 123,
      NOMBRE = "José Perez"
} : { CURSOS: string list, ID: int, NOMBRE: string}
```

Notas:

- ML reordena los campos de acuerdo al orden lexicográfico (tiene en cuenta los códigos ASCII para su ordenamiento)
- Una declaración de variable equivalente sería:
  `- val Estudiante1 = { NOMBRE = "José Perez", CURSOS = ["AM1", "ALG2", "INF"], ID = 123 };`
  El orden de los campos NO importa.

1.2. Valores de los campos

El operador `#` toma el nombre de un campo y devuelve el valor asociado a dicho campo en una estructura. Se suele escribir así:

```
#<nombreDelCampo>(<estructura>)
```

Sin embargo, los paréntesis no son necesarios y un espacio luego del # está permitido.

Ejemplo:

```
- #NOMBRE (Estudiante1);
val it = "José Perez": string
```

1.3. TUPLAS: casos especiales de estructuras de registro

Una tupla

```
(<valor1>, ..., <valorn>)
```

es una abreviación de la estructura:

```
{ 1 = <valor1>, ..., n = <valorn>}
```

Ejemplo:

La tupla

```
(3, "cuatro")
```

es una abreviación de

```
{ 1= 3, 2= "cuatro"}
```

Notas:

- Los paréntesis sólo pueden ser usados en tuplas.
- Las llaves se usan en las estructuras y las nombres de los campos son obligatorios.

1.4. Patrones para las estructuras

Es posible utilizar la notación de estructura para definir los patrones que necesita la definición de una función. Entre llaves (**{** y **}**) se encuentra una lista de campos separados por coma:

```
<nombreDelCampo> = <patron>
```

Ejemplo:

```
- exception NoEncontrado                                              (1)
exception NoEncontrado

- fun getID(nombre, nil) = raise NoEncontrado                         (2)
    | getID(nombre, (x as {NOMBRE=nom, ... })::XS) =                  (3)
          if nom=nombre then                                          (4)
               #ID(x:{NOMBRE:string, ID:int, CURSOS:string list})     (5)
          else getID(nombre, XS);                                     (6)
```

En la línea (3)

```
getID(nombre, (x as {NOMBRE=nom, ... })::XS) =
```

los ... son llamados *"wild cards"*, permiten hacen el matcheo de los otros campos. Si usamos este tipo de escritura es importante que demos el tipo de estructura completo en algún otro lado de la función, como se hace por ejemplo en la línea (5)

```
#ID(x:{NOMBRE:string, ID:int, CURSOS:string list})
```

Podríamos haber reemplazado la línea (3) por:

```
getID(nombre, (x as {NOMBRE=nom, ID=i, CURSOS=_})::XS) =
```

entonces en la línea (5) podríamos usar i en lugar de #ID(x) y podríamos evitar de declarar el tipo de la estructura. Otra opción para reemplazar la línea (3) sería

```
getID(nombre, (x as {NOMBRE=nom, ID=i, ...})::XS) =
```

nuevamente aquí podríamos usar i en lugar de #ID(x) pero deberíamos dar el tipo de estructura completo en otro lado (ejemplo, como hecho en línea (5)), ya que ML no podría deducir los campos abreviados al usar ...

Ejemplo: Suponga queremos trabajar con estructuras de tipo estudiante y calcular el costo de matrícula:

1. *Un estudiante que no se ha inscripto a cursos paga $1000.-*
2. *Un estudiante inscripto a un solo curso paga $2000.-*
3. *Un estudiante inscripto a dos o más cursos paga $4000.- si se trata de un estudiante de grado y $5000 si se trata de un estudiante de posgrado. A los estudiantes de posgrado se los identifica con un ID* ≥ 10000

```
- fun matricula({NOMBRE=_, ID=_, CURSOS=nil}) = 1000                    (1)
    | matricula({CURSOS=[_],...}) = 2000                                (2)
    | matricula({ID=i,...}) =                                           (3)
          if (i>=100000) then 5000                                      (4)
          else 4000;                                                    (5)

val matricula = fn: {CURSOS:'a list, ID:int, NOMBRE: 'b}->int

- matricula(Estudiante1);
val it = 4000

- matricula({NOMBRE="Pamela Rodriguez", ID = 54320, CURSOS=["MAT1"]});
val it = 2000

- matricula({NOMBRE="Martina Gonzalez", ID = 200000, CURSOS=["ALG3","MAT3"]});
val it = 5000

- matricula({NOMBRE="Marta Gomez", ID = 200570, CURSOS=[]});
val it = 1000
```

Como patrón, en vez de usar

```
<nombreDelCampo> = <patron>
```

podríamos usar

```
<nombreDelCampo>
```

Ejemplo: Podríamos reemplazar las líneas (3) y (4) de la siguiente manera:

```
    | matricula({ID,...}) =
          if (ID>=100000) then 5000
```