



ML 演習 第 3 回

おおいわ
April 24, 2001



資料について

- ● ● ●
- 演習ホームページ

<http://www.yl/~oiwa/lecture/ocaml/> に

- 毎回のレジュメの PDF
- 課題提出状況一覧

を載せました



今回の内容

- 例外の処理
- 副作用のサポート
 - reference
 - 変更可能フィールド
 - 複文

例外 (1)

■ 例1: ベクトルの正規化

```
# let normalize (x1, x2) =  
  let n = sqrt (x1 *. x1 +. x2 *. x2)  
  in (x1 /. n, x2 /. n);;
```

```
val normalize : float * float -> float * float = <fun>
```

```
# normalize (3.0, 4.0);;
```

```
- : float * float = 0.600000, 0.800000
```

→ (0, 0) が与えられたとき?

例外 (2)

■ 2つのベクトルのなす角

```
# let angle v1 v2 =  
  let ((x,y),(x',y')) = (normalize v1, normalize v2)  
  in acos (x *. x' +. y *. y');;  
val angle : float * float -> float * float -> float = <fun>  
# let degree_of_radian x = x *. 180.0 /. 3.1415926535897;;  
val degree_of_radian = float -> float = <fun>  
# let angleD v1 v2 = degree_of_radian (angle v1 v2);;  
val angleD : float * float -> float * float -> float = <fun>  
# angleD (1.0, 0.0) (0.0, 0.5);;  
- : float = 90.000000
```

例外 (3)

■ 方法1: エラーを示す値を決めておく

```
# type 'a option = None | Some of 'a;; (* 実は組み込み型 *)
```

```
type 'a option = None | Some of 'a
```

```
# let normalize (x1, x2) =
```

```
    let n = sqrt (x1 *. x1 +. x2 *. x2) in
```

```
    if n = 0.0 then None else Some(x1 /. n, x2 /. n);;
```

```
val normalize : float * float -> (float * float) option = <fun>
```

```
# normalize (0.0, 0.0);;
```

```
- : (float * float) option = None
```

例外 (4)

■ この方法の欠点: 使いづらい!!

```
# let angle v1 v2 =
```

```
  match (normalize v1, normalize v2) with
```

```
    (None, _) | (_, None) -> None
```

```
  | (Some(x,y), Some(x',y')) ->
```

```
      Some(acos(x *. x' +. y *. y'));;
```

```
val angle : float * float -> float * float -> float option = <fun>
```

```
# let angleD v1 v2 = match angle v1 v2 with
```

```
  None -> None | Some x -> Some (degree_of_radian x);;
```

例外の送出 (1)

■ 例外の定義と送出 (1)

```
# exception ZeroVector;;  
exception ZeroVector  
# raise ZeroVector;;  
Uncaught exception: ZeroVector.
```

```
# exception BadArg of float;;  
exception BadArg of float  
# raise (BadArg 5.0);;  
Uncaught exception: BadArg 5.000000.
```


例外の送出 (2)

■ 例外の利用

```
# let normalize (x1, x2) =  
  let n = sqrt (x1 *. x1 +. x2 *. x2) in  
  if n = 0.0 then raise ZeroVector else (x1 /. n, x2 /. n);;  
val normalize : float * float -> float * float = <fun>  
# normalize (3.0, 4.0);;  
- : float * float = 0.600000, 0.800000  
# normalize (0.0, 0.0);;  
Uncaught exception: ZeroVector.
```

例外の送出 (3)

```
# let angle v1 v2 =  
  let ((x,y),(x',y')) = (normalize v1, normalize v2)  
  in acos (x *. x' +. y *. y');;  
val angle : float * float -> float * float -> float = <fun>  
# let angleD v1 v2 = degree_of_radian (angle v1 v2);;  
val angleD : float * float -> float * float -> float = <fun>  
  
# angleD (1.0, 0.0) (0.0, 0.5);;  
- : float = 90.000000  
# angleD (0.0, 0.0) (0.0, 0.5);;  
Uncaught exception: ZeroVector.
```

例外の処理 (1)

■ 発生した例外を処理する

```
# let angle_str v1 v2 = try
  "Angle is " ^ string_of_float (angleD v1 v2)
with ZeroVector -> "Not defined.";;

val angle_str : float * float -> float * float -> string = <fun>

# angle_str (1.0, 0.5) (2.0, 3.0);;
- : string = "Angle is 29.7448812969"
# angle_str (1.0, 0.5) (0.0, 0.0);;
- : string - "Not defined."
```

例外の処理 (2)

■ 応用例

```
# exception Zero;;  
# let prod l =  
    let rec f = function [] -> 1  
      | hd::tl -> if hd = 0 then raise Zero else hd * f tl  
    in try f l with Zero -> 0;;  
val prod : int list -> list  
# prod [1;2;3;4;5;0;7;8;0];;  
- : int = 0
```



Imperative Features

- 副作用に依存したプログラミングのサポート
 - reference (破壊的代入)
 - 変更可能フィールド
 - 複文

Reference

■ 変更可能なセル

```
# val a = ref 0;;
```

```
val a : int ref = {contents=0}
```

```
# !a;;
```

```
- : int = 0
```

```
# a := 5;;
```

```
- : unit = ()
```

```
# !a;;
```

```
- : int = 5
```

変更可能なレコード

```
# type mutable_point = { mutable x:int; mutable y:int };;  
type mutable_point = { mutable x : int; mutable y : int; }  
# let p1 = { x = 5; y = 3; };;  
val p1 : mutable_point = {x=5; y=3}  
# p1.x <- 6;;  
- : unit = ()  
# p1;;  
- : mutable_point = {x=6; y=3}
```

(cf.) type 'a ref = { mutable contents : 'a }

複文

```
# let increment x a = (x := !x + a ; !x);;
val increment : int ref -> int -> int = <fun>
# let a = ref 0;;
val a : int ref = {contents=0}
# increment a 5;;
- : int = 5
# increment a 5;;
- : int = 10
```


unit 型

■ () が唯一の値

();;

- : unit = ()

■ 用途

- 副作用以外に意味のない関数の返り値
- 引数の不要な関数に与えるダミー値
 - C++ の void 型に相当

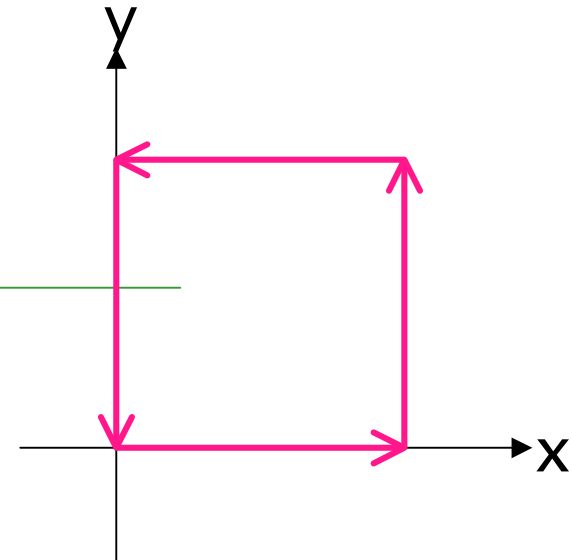
課題1

- “Turtle” のモデルとして、現在の位置と頭の向きを記憶するデータ型 turtle を作り、次の4つの動作を実現する関数群を定義せよ。
 - (0,0) に新しい turtle を生成する
 - turtle を n 歩前に進める
 - turtle を左に k 度回転させる
 - 現在の turtle の位置を返す

課題1 (仕様)

- `type turtle = (any implementation)`
- `val new_turtle = unit -> turtle`
 - 新しいタートルを生成
- `val advance = turtle -> float -> unit`
- `val rotate = turtle -> float -> unit`
 - タートルの位置を移動させる
- `val locate = turtle -> float * float`
 - 現在の位置を (x, y) のペアで報告

課題1 (例)



```
# let t1 = new_turtle ();  
val t1 : turtle = { ... }  
# advance t1 1.0; locate t1;;  
- : float * float = 1, 0  
# rotate t1 90.0; advance t1 1.0; locate t1;;  
- : float * float = 1, 1  
# rotate t1 90.0; advance t1 1.0; locate t1;;  
- : float * float = 0, 1  
# rotate t1 90.0; advance t1 1.0; locate t1;;  
- : float * float = 0, 0
```

(*計算誤差で値が正確に0にならないのは気にしなくて良い*)

課題1 (ヒント)

- 三角関数
 - $\sin, \cos, \dots : \text{float} \rightarrow \text{float}$ (弧度法)
 - $\pi = \text{atan } 1.0 * .4.0$ (必要なら...)
- データ型
 - 変更可能なレコードが最適か?

課題2

- Stack のデータ構造を表現する多相型を定義し、次の操作を実装せよ。
 - new_stack: 新しい stack の作成
 - new_stack の返り値の型は明示する (次回以降説明)
 - push: 要素を頭に追加
 - pop: 先頭要素を取り出す
 - 空 stack に対する pop は例外を送出

課題2 (例)

```
# let s = ( new_stack () : int stack);;
val t1 : int stack = .....
# push s 1;;
- : unit = ()
# push s 2;;
- : unit = ()
# pop s;;
- : int = 2
# pop s;;
- : int = 1
# pop s;;
Uncaught exception: EmptyStack.
```

課題2 (ヒント)

- データ型の実際の定義は?
 - 実はシンプルに

```
type 'a stack = { mutable c : 'a list }
```

でよい...
 - push: リストの先頭に要素を追加
 - pop: リストの head を取り出す、
取り出した後の stack はリストの tail

課題3 (optional)

- 課題2と同様に Queue を作れ。
 - new_queue: 新しい queue の作成
 - add: 新しい要素を末尾に追加
 - take: 先頭の要素を取り出す
- Stack と似てますが、ずっと難しいです。



提出方法

- 〆切: 2001年5月7日 (月) 24:00
- 提出先: ml-report@yl.is.s.u-tokyo.ac.jp
- 題名: “Report 3 (学生証番号)”