第3回分レポート解答例

おおいわ

May 8, 2001

1 タートル

解答例

```
 1  type turtle = { mutable x : float; mutable y : float; mutable angle : float}
 2
 3  let pi = 4.0 *. atan 1.0
 4
 5  let new_turtle () = { x = 0.0; y = 0.0; angle = 0.0 }
 6
 7  let advance t s =
 8    let sx, sy = s *. cos t.angle, s *. sin t.angle in
 9    t.x <- t.x +. sx; t.y <- t.y +. sy
10
11  let rotate t r =
12    t.angle <- t.angle +. r *. pi /. 180.0
13
14  let locate t = t.x, t.y
```

new_turtle で新しいタートルを生成し、advance と rotate は．<- 構文を使ってタートルの変更可能フィールドを更新します。角度のとり方はいくつか考えられますが、ここではフィールドを更新する際に弧度法に変換してみました。

2 スタック

解答例

```
 1  type 'a stack = { mutable c : 'a list }
 2
 3  exception EmptyStack
 4
 5  let new_stack () = { c = [] }
 6
 7  let push s v = s.c <- v :: s.c
 8
 9  let pop s = match s.c with
10    [] -> raise EmptyStack
11  | h::t -> s.c <- t; h
```

データ型の定義は単純で、実際のところほとんど Reference と等価ですので、もっと単純に

```
type 'a stack = 'a list ref
```

として、他の関数を適宜 := などで書き換えても OK です。pop の ; の使い方がキーでしょうか。

3 キュー

解答例

```
1   type 'a cell = { v : 'a ; mutable next : 'a cell option }
2   type 'a queue = { mutable head : 'a cell option;
3                     mutable tail : 'a cell option }
4   exception EmptyQueue
5
6   let new_queue () = { head = None; tail = None }
7
8   let add q v =
9     let new_cell = { v = v; next = None }
10    in
11    match q.tail with
12      None -> q.head <- Some new_cell; q.tail <- Some new_cell
13    | Some n -> n.next <- Some new_cell; q.tail <- Some new_cell
14
15  let take q =
16    match q.head with
17      None ->
18        (* no element *)
19        raise EmptyQueue
20    | Some { v = v; next = None } ->
21        (* last one element *)
22        q.head <- None; q.tail <- None; v
23    | Some { v = v; next = n } ->
24        (* two or more elements *)
25        q.head <- n; v
```

Stack のプログラムの :: を @ に変えたようなプログラムがかなりありました。問題の意図としては、「Queue である以上は要素の追加は定数時間オーダで行なう」ことを期待していましたので、レジュメにも「難しいかも……」と書いたのでした。@ を末尾 1 要素の追加に使うと、前のリストを全部コピーして最後の部分だけを書き換えるので (第 2 回の append の実装を思い出してください)、操作に現在の要素数に比例した時間がかかってしまいます。

上のプログラムは、cell としてリストの尾部を書き換え可能にしたデータ構造を用意して、その先頭と末尾を queue のフィールドに保持することで、先頭の要素からの取り出しと、末尾への追加の両方を定数時間オーダで可能にしています。queue が空である時には head と tail の両方に None が格納されており、要素がある場合は head と tail にそれぞれ先頭と末尾の cell が Some *x* の形で格納されています。この性質を常に保つため、空の queue に対する追加操作と、最後の 1 要素を取り出す操作が特別扱いされています。

別解としては、双方向リストや円環リストの利用も考えられます。@ を使った解答を提出してくれた人で余裕のある人は、これらのデータ構造を使った別実装にトライしてみてください。