情報科学演習及び実験1（OCaml抽象データ型）

IS2 6316047 出納 光

平成29年6月16日

1　課題1

課題内容

多相性のヴァリアントで定義したリスト構造を用いて，以下の処理を行うモジュールを定義しなさい．
ただし，リストの方向は「無し」（Nil）を参照している要素を末尾とする．
余裕のある人は，下記以外の処理も自分で定義して作成しなさい．

create: 空のリストをつくる
unshift x: リストの先頭にデータを追加する
shift: リストの先頭の要素を削除する
push x: リストの末尾にデータを追加する
pop: リストの末尾を削除する
size: リストの要素数を返す
max: 最大値を返す
min: 最小値を返す
get x: x番目の要素を参照する
indexOf x: 要素がxと同じ場合にインデックスを返す．xがない場合には-1を返す
set x y: 指定された要素を，指定された要素で置き換える（複数ある場合はすべて）
remove x: 指定された要素を，削除する（複数ある場合はすべて）
concat: 二つのリストを接続する

1．アルゴリズムの説明
2．プログラムの説明

create
1．内容が単純であり、アルゴリズムを考える必要がなかった。
2．あらかじめ type によって定義しておいたコンストラクタ Nil を返す。

unshift
1．内容が単純であり、アルゴリズムを考える必要がなかった。
2．あらかじめ type によって定義しておいたコンストラクタ Cell の先頭に引数を与える。

shift
1．内容が単純であり、アルゴリズムを考える必要がなかった。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest を返す。

push
1．再帰させリスト構造から 1 つずつ要素を外していき、最後の Nil の直前で引数を返す。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest が Nil になるまで再帰させ、
その時点で、Nil の手前に引数を返す。

pop
1．再帰させリスト構造から 1 つずつ要素を外していき、最後の Nil の直前の 1 つを取り除く。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest が Nil になるまで再帰させ、
その時の n を除外して、Nil を返す。

size
1．再帰させリスト構造から 1 つずつ要素を外していき、その再帰回数をカウントする。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest が Nil になるまで再帰させ、
その回数だけ、+1 する。

get
1．再帰させリスト構造から 1 つずつ要素を外していき、最後の値を返す。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest が Nil

になるまで、
また、num が 1 になるまで再帰させ、そのとき直後の 1 つの値を返す。

set
1．再帰させリスト構造から 1 つずつ要素を外していき、全ての要素を処理する。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest が Nil になるまで、
与えられた引数と照合、置き換えを繰り返す。

remove
1．再帰させリスト構造から 1 つずつ要素を外していき、全ての要素を処理する。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest が Nil になるまで、
与えられた引数と照合、削除を繰り返す。

concat
1．再帰させ前者のリストの Nil の直前までキープし、後者のリストの先頭に付ける。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest が Nil になるまで、
それらの要素をキープし、最後に後者のリストを与える。

max
1．再帰させリスト構造から 1 つずつ要素を外していき、大小関係を比較し大きい方を残す。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest が Nil になるまで、
大小比較し、大きい方を残し、それを返す。

min
1．再帰させリスト構造から 1 つずつ要素を外していき、大小関係を比較し小さい方を残す。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest が Nil になるまで、
大小比較し、小さい方を残し、それを返す。

indexOf

1．再帰させリスト構造から 1 つずつ要素を外していき、再帰回数もカウントしておく（ただし、-1)、
対象が見つかれば、そのカウントを返す。
2．パターンマッチングにより、Cell(n, rest) と分け、残った rest が Nil になるまで、
照合し、再帰回数もカウントしておく（ただし、-1)、見つかった時点で 0 を返す。

実行結果

```
module List :
  sig
    exception Empty
    type 'a list = Nil | Cell of 'a * 'a list
    val create : 'a list
    val unshift : 'a -> 'a list -> 'a list
    val shift : 'a list -> 'a list
    val push : 'a -> 'a list -> 'a list
    val pop : 'a list -> 'a list
    val size : 'a list -> int
    val get : int -> 'a list -> 'a
    val set : 'a -> 'a -> 'a list -> 'a list
    val remove : 'a -> 'a list -> 'a list
    val concat : 'a list -> 'a list -> 'a list
    val max : 'a list -> 'a
    val min : 'a list -> 'a
    val indexOf : 'a -> 'a list -> int
  end
```

考察

基本的には再帰関数を用いることで、あらゆるメソッドが実装できる。
適宜、条件分岐も使い分け、少しでも重くない処理などに置き換えていくことが、大切だと思う。

2　課題 2

課題内容

多相性のヴァリアントで定義した二分木構造を用いて，以下の処理を行うモジュールを定義しなさい．
コンストラクタや以下に指定してある関数の内部構造は隠蔽すること．
（隠蔽している場合はデータ構造は見えません）

create: 空の二分木をつくる
insert: 新しいデータを，木の該当する部分に追加する
search: 該当するデータがあるか判定する
search_min: 最小値を返す（隠蔽）
delete_min: 最小値を削除する（隠蔽）
delete: 該当するデータを，木から削除を行い，場合によって木を変形させる

1．アルゴリズムの説明
2．プログラムの説明

create
1．内容が単純であり、アルゴリズムを考える必要がなかった。
2．あらかじめ type によって定義しておいたコンストラクタ Nil を返す。

search_min
1．内容が単純であり、アルゴリズムを考える必要がなかった。
2．再帰させ、2 つ目の要素が Nil になるまで続け、そのときの 1 つ目の要素を返す。

delete_min
1．内容が単純であり、アルゴリズムを考える必要がなかった。
2．再帰させ、2 つ目の要素が Nil になるまで続け、そのときの 3 つ目の要素を返す。

実行結果

```
module Separate :
  sig
    type 'a tree = Nil | Node of 'a * 'a tree * 'a tree
    val create : 'a tree
```

```
  exception Empty
  val insert : 'a -> 'a tree -> 'a tree
  val delete : 'a -> 'a tree -> 'a tree
  val search : 'a -> 'a tree -> bool
end
```

考察

今回、内部構造の隠蔽に留意して、プログラムを実装したが、
そのデータ構造を隠蔽する必要があるかどうかというのを考えながら、
今後もプログラムを実装すべきであると思う。

また、検証に十分な時間が得られなかった。