# テキスト処理 第7回 (2007-06-19)
# treefindレポート説明

田中哲
産業技術総合研究所
情報技術研究部門
akr@isc.senshu-u.ac.jp
http://staff.aist.go.jp/tanaka-akira/textprocess-2007/

# レポート

- ネストしているかもしれない配列を受け取り、各要素についてブロックを適用し、真を返した最初の要素を返す treefind を作れ。真を返す要素がなければ nil を返すものとする
- def treefind(ary) ... end
- 再帰をつかうこと。
- flatten の類は禁止
- ∠切 2007-06-19 16:20
- HIPLUS
- 拡張子が txt なプレインテキストで

# treefindの動作例

- treefind([1,2,3,4]) {|v| v % 3 == 2 } #=> 2
- treefind([[1,2],[3,[4,5]],6]) {|v| v % 3 == 2 } #=> 2
- treefind(
  ["ruby", "perl", "python", "php", "lisp"]) {|v| /^p/ =~ v }
  #=> "perl"

# 制限事項

- flatten の類は使わない
- 再帰する
- 例えば以下のような実装は禁止

```
def treefind(ary)
  a = ary.flatten.find_all {|v| yield v }
  a.empty? ? nil : a[0]
end
```

# ユニットテストを配布する

- 動作例で示したようなものを自動的に検査するための機構 (詳細はリファレンスマニュアル参照)
- test-treefind.rb
- test-treefind.rb に treefind の定義を書き加えて実行し、失敗が 0 になるまでがんばる

```
% ruby test-treefind.rb
Loaded suite test-treefind
Started
..
Finished in 0.000517 seconds.


2 tests, 4 assertions, 0 failures, 0 errors
```

# 解答例

```
def tree_each(obj, &b)
  if obj.respond_to? :each
    obj.each {|v|
      tree_each(v, &b)
    }
  else
    yield obj
  end
end
```

```
def treefind(ary)
  tree_each(ary) {|v|
    return v if yield v
  }
  nil
end
```

# ざっとみた結果

- flatten 相当の関数を書いたものがあった
  - でも要素を見付けた後の処理は無駄
- 関数をひとつで済ましたコードには無理がある
  - treefind は「見つかったかどうか」「見つかった場合にはその値」を呼び出し側に伝えなければならない
  - nil が見つかったらどうするか?
    - ブロックをもう一回呼び出して見つかった nil か見つからなかった nil かを調べる (無駄)
    - [見つかったかどうかの真偽, 見つかった値] という 2要素の配列を返す (treefind 以外の関数にわける必要がある)
- respond_to? には使用するメソッド名を指定する
  - compact を使わないのに compact を調べない