

コンパイラ理論 13 Racc その8 補足

櫻井彰人

calc.y で文字列を扱うには

◆ intp.y の文字列部分をコピーすればよい

スキャナーの "NUMBER" のあとに

```
when /%A'(?:[^'¥¥st]+|¥¥.)*"/, /%A'(?:[^'¥¥st]+|¥¥.)*'/  
  @q.push [:STRING, eval($&)]
```

exp の定義中 ("NUMBER" のあとに)

```
| STRING
```

08Racc-03.pdf を参照のこと

◆ または、intp.y の関連部分を全てコピー

08Racc-03.pdf の最後のスライド等を参照のこと

Ruby の map

配列の各要素にブロックを実行し配列を作成する

collectメソッド、mapメソッドは配列の各要素に対してブロックを実行し、結果を配列で返却します。それぞれ、破壊的メソッドcollect!、map!が用意されています。

```
a = [10, 20, 30, 40, 50]  
p a.collect {|x| x*10}      #=> [100, 200, 300, 400, 500]  
p a #=> [10, 20, 30, 40, 50]  
p a.map {|x| x*10}         #=> [100, 200, 300, 400, 500]  
p a #=> [10, 20, 30, 40, 50]  
a.collect! {|x| x/10}     #=> [1, 2, 3, 4, 5]  
p a #=> [1, 2, 3, 4, 5]  
a = [["vine", 2, 500], ["orange", 3, 100], ["apple", 10, 500]]  
p a.map {|x| [x[0], x[1]*x[2]]} #=> [["vine", 1000], ["orange", 300], ["apple", 500]]
```

逆引きRuby 配列: <http://www.namarai.com/rubytips/>

Python の map

(サンプル)

```
# map built-in function  
def f(x):  
    return (x + '?')  
  
def f2(x, y):  
    return (str(x) + str(y))  
  
a = ['a', 'b', 'c']  
print map(f, a)  
b = ['A', 'B', 'C', 'D']  
print map(None, b)  
print map(f2, b, a)
```

(実行結果)

```
['a?', 'b?', 'c?']  
['A', 'B', 'C', 'D']  
['Aa', 'Bb', 'Cc', 'DNone']
```

<http://hp.vector.co.jp/authors/VA003670/python/Builtin/map.htm>

Lisp の map

Map関数は一つの関数を繰り返し呼び出すための関数の総称です。組み込みのMap関数には、mapcar、maplist、mapc、mapl、mapcan、mapconがあります。これらは、Listデータだけにしか使うことができませんが、(Map関数 関数 データ1 ... データN) というように第一引数に繰り返し呼び出したい関数を与え、第二引数以降にデータを渡します。

```
(mapcar 'list '(1 2 3))  
(1) (2) (3))  
(mapcar 'list '(1 2 3) '(a b c))  
(1 A) (2 B) (3 C))  
(mapcar #'(lambda (x) (* x x)) '(1 2 3))  
(1 4 9))  
(maplist 'list '(1 2 3))  
(((1 2 3)) ((2 3)) ((3)))  
(maplist 'list '(1 2 3) '(a b c))  
(((1 2 3) (A B C)) ((2 3) (B C)) ((3) (C)))  
(maplist #'(lambda (x) (* x x)) '(1 2 3))  
Error: (1 2 3) is an illegal argument to *  
[1] :reset  
(maplist #'(lambda (x) (length x)) '(1 2 3))  
(3 2 1))  
(mapcan 'list '(1 2 3))  
(1 2 3)  
(mapcon 'list '(1 2 3) '(a b c))  
(1 A 2 B 3 C))
```

mapcar、maplistは呼び出しごとの関数の値をリストにしますが、mapc、maplは、リストを作らずに呼び出しだけを行いたい時に使います。mapcanは、呼び出した関数の結果がリストだとして、すべての結果のリストを appendした結果を返します。mapconはappendではなくnconcします。

<http://www.jsk.t.u-tokyo.ac.jp/~inaba/soft3/soft3-f1-lisp/node59.html>