計算機プログラミング'93　＃2

久野　靖*

1993.9.8

前回思ったより進まなかったので、前回資料のおしまいの所を再録します。が、その前に nemacs から Lisp を使うやり方も解説しておきましょう。

1　nemacs から Lisp を使うやり方

前回の講義で一応説明しましたが、復習しておきます。基本的には (1) nemacs の窓を 2 つに分割する。(2) それぞれの部分で Lisp の編集と KCL の実行をやる。(3) 編集したファイルは Lisp の load 関数で読み込む。ということでしたね。

- nemacs を引数なしで起動します。
- 窓を 2 分割するには^X2 と打ち込みます。
- 分割した窓の間を行ききするには^Xo と打ちます。
- ファイルを指定するには^X^F を打った後ファイル名を指定します。Lisp の場合ファイル名は.lsp で終るようにしておくのがよいです。
- 編集した後は^X^S でファイルを保存します。
- もう 1 方の窓で KCL を実行させるには [ESC]xrun-lisp[RET] と打ち込みます。
- KCL の方では「(load "ファイル名")」でそのファイルが読み込めます。
- 1 つのファイルに何個関数を書いても構いません。

2　再帰的関数

さて、分岐までできたけれど、まだこれだけでは大した計算はできない。そこで次に、Pascal や C ならループをやるのだけれど、Lisp ではその代わりに再帰をやる。再帰というのは要するに自分に渡された問題を「ちょっとだけ」簡単にして、残っている部分をもう 1 度自分自身 (のコピー) にたらいまわしにするというやり方である。例えば階乗の計算を考えてみる。5! = 5 * 4 * 3 * 2 * 1 = 5 * 4!ですよね。だから:

```
(defun factorial (x)
  (cond ((zerop x) 1)
        (t (* x (factorial (- x 1))))))
```

*筑波大学大学院経営システム科学専攻

つまり、ごく簡単な場合 (`x = 0`) は自分で解いてしまうが、そうでない場合はまず `(x-1)!` を求めて、それに `x` を掛け合わせることで `x!` を求めるわけである。なお、「簡単な場合は自分で解く」というのがないと無限にたらいまわししようとして止まらなくなるので注意すること。あと、再帰関数の虫取りにはトレース機能を使うと便利である。これは、

`(trace` 関数名 …`)`

とやると、以後この関数が呼ばれたときの引数と戻るときの返値を画面に表示してくれる。では練習。

練習 4 以下の関数を定義せよ。

- 非負整数 n を受け取り、$1+2+\ldots+n$ を返す関数 `nsum`
- 数 x、n(n は非負整数) を取り、x^n を返す関数 `power1`
- 数 x、n(n は任意整数) を取り、x^n を返す関数 `power`
- 数 x、n(n は自然数) を取り、$1+x+x^2+\ldots+x^n$ を返す関数 `crsum`
- 2 つの自然数 x、y を受け取り、その最大公約数を返す関数 `xgcd`

なお、最後のはちょっと説明を要する。とりあえず `x>=y` と仮定すると、もし `x=y` であれば最大公約数は `x` と同じである。そうでなければ、`(xgcd x y)` は `(xgcd y (- x y))` と同じである。これを利用して再帰呼び出しをすればよいわけだ。

3 本日の小課題 — 2 分法による求根

関数 $f(x)$ が区間 $[a,b]$ で連続かつ単調増大 (実は単調減少でも原理は同じ) であり、なおかつこの区間に $f(x)=0$ の根があるものとする。その時、任意の小ささの誤差 e を指定して、その誤差範囲内でこの根の値を求める方法がある。それには次のような関数 `(getroot` a b e`)` を書けばよい。

- $(b-a)<e$ ならば、$\frac{a+b}{2}$ が答え。
- そうでなくて、$f(\frac{a+b}{2})<0$ なら、`(getroot` $\frac{a+b}{2}$ b e`)` が答え。
- そうでなければ `(getroot` a $\frac{a+b}{2}$ e`)` が答え。

なぜそうか? よく考えてみよう。納得したら、この方法で $f(x)=x^2-2$ として、つまり 2 の平方根を求めるプログラムを作って動かせ。納得したら、別の関数の根もいろいろ試してみよ。また任意の数の平方根を求める関数 `sqroot` を作ってみよ (誤差は適当に決めてよい)。

結果をレポートとして提出する場合には、必ず A4 版の紙を使用し、次のような順で構成し、全体を綴じて提出すること。

- 表紙。課題名 (1993 年度計算機プログラミング課題＃ 1)、学籍番号、氏名、提出日付) のみを記すこと。
- 方針。どのような考え方で課題を解こうと思ったか記す。
- 回答。作成したプログラムとその解説、実行例など。
- 考察。課題をやった結果どんなことがわかったか、何が問題か、など。(感想も入れてほしいが、感想ばかりでは内容として不足。)

〆切は一応次回の授業の直前までとしますが、遅れても受理はします。最初に述べたように小課題の提出は義務ではありません。

4　リストとその操作 (1)

前回述べたように、Lisp ではリスト構造を持つ一般の S 式をデータとして扱うこともできる (というか、この方が単なる数値を扱うことよりもずっと多い)。そこで、今回はリスト操作について学ぶ。リストを扱う基本的な関数を挙げておく。まず、リストを分解する関数から。

- (car L): リスト L の先頭要素を取り出す。
- (cdr L): リスト L の先頭を取り除いた残りのリストを返す。

この 2 つを利用すれば、リストの任意の部分を取り出すことができる。例えば次の演習をやってみる。

練習 5 まず、KCL に向かって次のように打ち込め。

```
(setq x '(a b (c d) (e f (g h i))))
```

これで変数 x にリスト (入れ子になっていますが) が格納された。この状態で、x の中から次の要素を car と cdr を使って取り出せ。

- A
- (B (C D) (E F (G H I)))
- B
- (C D)
- (D)
- NIL
- (H I)

なお、この問題から分かるように car と cdr を連続して使うことが多いので、例えば

- (car (cdr X)) → (cadr X)
- (cdr (car (car X))) → (cdaar X)
- (car (cdr (car (car X)))) → (cadaar X)

のように、4 つまでの任意の組合せについて予め組み合わせた関数が用意されている。

5　リストとその操作 (2)

前節では分解する方だけだったが、組み立てたり様々な操作をする関数が予め用意されている。

- (cons X L): リスト L の先頭に X を追加したリストを返す。
- (list X_1 X_2 $\cdots$ X_n): X_1〜X_n から成るリストを返す。
- (append L_1 L_2): リスト L_1 と L_2 を連結したリストを返す。
- (reverse L): リスト L を逆順にしたリストを返す。
- (last L): リスト L の最後の要素のみから成るリストを返す。

練習 6 テキストがあるのだから、たまには参照しよう。テキストの練習問題 2.1〜2.12。

6　リストに対する述語と cdr 再帰

前回やった述語は数に関するものが中心だったが、今度は一般の S 式や記号に関するもの。なお、関数=は数値専用で、記号や S 式は eq、equal を使わなければならない。

- (atom X): X がアトムなら「はい」。
- (numberp X): X が数値アトムなら「はい」。
- (symbolp X): X が記号アトムなら「はい」。
- (eq X Y): X と Y が同じ記号なら「はい」。
- (listp X): X がリストなら「はい」。
- (consp X): X が空でないリストなら「はい」。
- (null X): X が空リストなら「はい」。(空のリストは nil と同じものだから、これは (eq X nil) と同じこと。)
- (equal X Y): X と Y が同じ内容の S 式なら「はい」。

リスト関係の述語を学んだので、これを使った基本的な再帰のパターン (cdr 再帰) をまず覚えてもらいたい。次の関数はリストの長さを返すものである。

```
(defun xlength (l)
  (cond ((null l) 0)
        (t (+ 1 (xlength (cdr l))))))
```

なぜこれでいいか考えてみてほしい。もう少し内容のある例として、数のリストを受け取ってその合計を返す関数も示す。

```
(defun listsum (l)
  (cond ((null l) 0)
        (t (+ (car l) (xlength (cdr l))))))
```

要するに、cdr 再帰というのは「リストの先頭要素は自分で処理し、残りの部分については自分自身を再帰的に呼び出すことで処理する」というパターンを指す。

練習 7 以下の関数を定義せよ。

- リスト l を受け取り、その中の数値のみの合計を返す関数 sumnum
- リスト l とアトム x を受け取り、l 中の x と同じものの個数を返す関数 countx
- リスト l とアトム x を受け取り、l 中に x と同じものがあるかどうか調べる述語 xmember
- リスト l を受け取り、その最後の要素を返す関数 saigo

7　cdr 再帰によるリストの組み立て

前節では 1 個の数値や特定の値を返すだけだったが、そうではなく cdr 再帰のパターンで新しいリストを組み立てることも容易である。要は、car と cdr で分解することの反対は cons でくっつけることだと理解していればよい。例えば次の例を見てみる。

```
(defun add1list (l)
  (cond ((null l) nil)
        (t (cons (+ (car l) 1) (add1list (cdr l))))))
```

練習 8 以下の関数を定義せよ。

- リスト l を受け取り、その各要素を () の中に入れたリストを返す関数 `enpar`
- リスト l を受け取り、その中の数値を取り除いたリストを返す関数 `removenum`