疑似コードとリスト処理関数について

1　はじめに

教科書として用いている「人工知能の基礎」(西田豊明著) では，アルゴリズムを説明するために PASCAL 風の擬似コードを用いている．これはある特定のプログラミング言語を使うことで，その言語に不慣れな読者が困ることがないようにという配慮である．このコードを実際にコンパイルする，もしくは解釈実行する処理系はない．この疑似コードでは，記述が煩雑になることを避けるために，部分的に自然言語で説明してあったり，数式をもちいている．自然言語による説明は “<” および “>” でくくられている．

以下では，この疑似コードの予約語や規則，そしていくつかのリスト処理関数に関して説明をする．以下ではカバーしきれてない部分もあるかも知れないが，C 言語によるプログラミングの経験のある人なら常識的に解釈できるであろう．

2　代入

$n \leftarrow n + 1$ は代入を意味する．
C 言語の $n = n + 1$ と等価である．

3　手続き宣言

「**proc** foo(仮引数: 型)」は，手続き foo の宣言であり括弧の中は仮引数および引数の型の並びである．

4　関数宣言

「**function** bar(仮引数) **returns** 戻り値」は，関数 bar の宣言である．関数は戻り値を持つ．

5　関数呼出し

n ← pop(L) は，リスト L に pop という関数を適用して，その結果 (戻り値) を n に代入することを意味する．

6　条件式

if, then, else による条件式は C 言語のそれから容易に推察できるであろう．

7　複文

「**begin** 文 1; 文 2; ....; 文 n **end**」は複文を表す．

8　繰り返し

8.1　repeat 文

「**repeat** 文 1; 文 2; ...; 文 n **until** 条件」は，条件が成り立つまで，文 1 から文 n までを順次繰り返すことを意味する．

8.2　while 文

「**while** 条件 **do** 文」は，条件が成り立つ間，文を繰り返すことを意味する．複数の文を繰り返す時は複文としてまとめればよい．

8.3　for 文

「**for** 変数 ← 式 1 **to** 式 2 **do** 文」は，変数の値を式 1 の値 (初期値) から式 2 の値 (終値) まで (整数の場合は一つずつ) 順に増やしながら文を繰り返して実行することを意味する．

8.4　exit 文

exit を実行すると，一番内側の repeat, while, もしくは for の制御構造から抜出す．

9 case 文

case 文は C 言語の **switch** 文とほぼ同様な制御を行う.

case 式 **of**
　選択肢の組 1: 文 1;
　選択肢の組 2: 文 2;
　.....
　選択肢の組 n: 文 n
end

10 リスト処理関数

10.1 定数

まず，**定数** (constant) を定義する．定数とは，**特定の事物**や**特定の関係**の名前，もしくは**数**である．定数は**アトム** (atom, 原子記号) とも呼ばれる．アトムは単語のような物だと思えばよい.

10.2 リスト

リストとは以下のような構造を持つデータである．以下の例では a,b,c,d はすべてアトムとする．nil は空リスト (つまり要素のないリスト) を意味する.

```
.-.-.-.-nil
| | | |
a b c d
```

便宜上，上記のリストを $[a, b, c, d]$ と表記することにする (注意: 教科書ではリストの表記は $\{a, b, c, d\}$ となっている)．この表記を用いると，以下のリストは $[a, b, c, [x, y]]$ となる (リストの要素はアトムでもリストでもよい).

```
.-.-.-.-nil
| | | |
a b c .-.-nil
```

```
| |
x y
```

リストはそれを頭部 (head) と尾部 (tail) に分割することで処理できる．リスト $[a, b, c, d]$ の頭部はアトム a で，尾部はリスト $[b, c, d]$ となる (尾部はリストであることに注意)．さらにリスト $[a]$ の頭部はアトム a で尾部は nil (空のリスト，つまり [] のこと) である．

10.3 リスト分解関数

リストを分解する関数として，$car()$ および $cdr()$ を定義する．

リスト操作関数 $car()$ は一つのリストを引数としてとり，与えられたリストの頭部を返す．例えば，$car([a, b, c])$ の値は a である．また，$car([a])$ の値も a である．

リスト操作関数 $cdr()$ は一つのリストを引数としてとり，与えられたリストの尾部を返す．例えば，$cdr([a, b, c])$ の値は $[b, c]$ である．また，$cdr([a])$ の値は [ ]$(null)$ となる．

10.4 pop 関数

教科書に出てくる $pop()$ という関数は，おおよそ以下の処理を行う．

function pop(*list*)
　　returns a member
　head ← car(list);
　list ← cdr(list);
　exit (*head*);

つまり，$pop()$ は引数のリストの頭部を返し，引数であったリストを，元のリストの尾部に変更してしまう (副作用があることに注意)．

10.5 append 関数

教科書で使われる関数に $append()$ がある．$append()$ は引数として二つのリストをとり，リストを値として返す．例により $append()$ の動きをみてみよう．

$append([a,b],[c,d])$ の値は $[a,b,c,d]$ となる．つまり $append()$ は，引数として与えられた二つのリストを連結し新しいリストを生成し，それを値として返す．$append$ は引数にリストを要求するので，もしも a がアトムなら $append(a,[b,c,d])$ はエラーとなる．

10.6 list 関数

もう一つ教科書で使われる関数に $list()$ がある．$list()$ は引数として任意の個数のアトムもしくはリストをとり，リストを値として返す．例により $list()$ の動きをみてみよう．

$list(a,b,c,d)$ の値は $[a,b,c,d]$ となる．また，$list(a,[b,c],d)$ の値は $[a,[b,c],d]$ となる．

つまり，$list()$ は任意個の引数 (アトムでもリストでもよい) をとり，それらの引数を要素として連ねたリストを生成し，それを返す．

10.7 リスト処理関数が組み込まれたプログラミング言語

Prolog, Lisp, Java などのプログラミング言語には，最初からいくつかのリスト処理関数が組み込まれている．