CLU 言語入門　＃ 6

久野　靖

1993.6.29

1　おまけ 1: 自動的な型変換

次のようなプログラム辺で最後に代入される値は Pascal、C、PL/I、CLU でそれぞれどうなるか?　または、このような代入はコンパイラを通らないか?

```
(1) int i = 1; int j = i + "123";
(2) int i = 1; int j = i + "a";
(3) int i = 1, j = 2; int k = i = j;
(4) int i = 1; real a = i;
(5) real a = 3.14; int i = a;
```

2　おまけ 2: 型の同一性

以下の変数、や型定義のうち型が同一のものはどれとどれか。名前による同一性、構造による同一性の双方で考えよ。

```
var v1,v2: record a,b:integer end;
var v3: record a,b:integer end;
type r4 = record a,b:integer end; var v4:r4;
type r5 = record a,b:integer end; var v5:r5;
type r6 = record x,y:integer end;
type r7 = record x,y:1..1000 end;
type r8 = record y,x:1..1000 end;
type r9 = record y:1..1000; x:1..1000 end;
type r10 = record x:integer; n: ^r10 end;
type r11 = record x:integer; n: ^r10 end;
type r12 = record x:integer; n: ^r12 end;
type r13 = record x:integer; n: ^r12 end;
type r14 = record x:integer; n: ^r15 end;
type r15 = record x:integer; n: ^r14 end;
type r16 = record x:integer; n: ^r17 end;
type r17 = record x:integer; n: ^r16 end;
type r18 = record x:integer; n: ^r19 end;
type r19 = record x:integer; n: ^r20 end;
type r20 = record x:integer; n: ^r17 end;
```

3 CLU のクラスタ — 抽象データ型

秋まで待っているのも何ですから、そろそろクラスタをやってしまいましょう。要は、新しいデータ型を作り出すのがクラスタです。構文は次の通り。

```
クラスタ名 = cluster is 操作名, ....
  名前 = 型指定 ...        ←この中で必ず rep 型を定義
  操作名 = proc(引数...) returns(...) signals(...)
    文 ...
  end 操作名          ←もちろん、iter でもよい。
  ...                 ←操作を全部書く。局所操作もあってよい。
end クラスタ名
```

さっそく例。まずは外部仕様だけ考えるのがこつです。

```
nametable = cluster is new, store, fetch, entries
  new = proc() returns(nametable)
  fetch = proc(n:nametable, s:string) returns(int) signals(not_found)
  store = proc(n:nametable, s:string, i:int)
  entries = iter(n:nametable) yields(string,int)
end nametable
```

これが「あったとして」start_up を次のように用意する。

```
% sam15.clu -- cluster example

start_up = proc()
  pi:stream := stream$primary_input()
  po:stream := stream$primary_output()
  nt:nametable := nametable$new()
  while ~stream$empty(pi) do
    s:string := stream$getl(pi)
    i:int := int$parse(stream$getl(pi))
    if i ~= 0 then
      nt[s] := i
    else
      stream$putl(po, s||":"||int$unparse(nt[s]))
    end
      except when not_found: stream$putl(po, "?") end
  end
  for s:string,i:int in nametable$entries(nt) do
    stream$putright(po, int$unparse(i), 5)
    stream$putc(po, ' ')
    stream$putl(po, s)
  end
end start_up
```

ここまではよろしいでしょうか? ではクラスタ本体を作りましょう。

```
nametable = cluster is new, store, fetch, entries
  ent = record[s:string, i:int]
  rep = array[ent]
  new = proc() returns(cvt)
    return(rep$new())
  end new
  fetch = proc(r:cvt, s:string) returns(int) signals(not_found)
    for e:ent in rep$elements(r) do
      if e.s = s then return(e.i) end
    end
    signal not_found
  end fetch
  store = proc(r:cvt, s:string, i:int)
    for e:ent in rep$elements(r) do
      if e.s = s then e.i := i; return end
    end
    rep$addh(r, ent${s:s, i:i})
    for k:int in int$from_to_by(rep$high(r)-1, rep$low(r), -1) do
      if r[k].s > r[k+1].s then
        e:ent := r[k]; r[k] := r[k+1]; r[k+1] := e
      end
    end
  end store
  entries = iter(r:cvt) yields(string,int)
    for e:ent in rep$elements(r) do yield(e.s, e.i) end
  end entries
end nametable
```

なお、ここで cvt という見なれないものが出てきましたが、これは「クラスタの外から見ると nametable 型、中から見ると rep 型」を意味します。抽象データ型は外部から見ると完備した型に見えるけれど、中からみると何らかのデータ構造で実現されているのでこうするわけです。なお、Simula 流のオブジェクト指向言語ではデータ抽象といっても内部表現はスロット (実体変数) の集まりだからこういうことは考えていない。悪くいえばクラス内部では外からの視点と中からの視点が混ざっているわけですね。(難しい?)

それで、もちろんクラスタの内部実現は外部からアクセスできないように保護されている (だから store で内容を整列しておけばあとは揃ったままになっている)。また、実現を取り替えても構わない。2 分木にしたものを示しておく。

```
nametable = cluster is new, store, fetch, entries
  ent = record[left,right:nametable, s:string, i:int]
  rep = variant[e:ent, n:null]
  new = proc() returns(cvt)
    return(rep$make_n(nil))
  end new
  store = proc(r:cvt, s:string, i:int)
    if rep$is_n(r) then
```

```
      rep$change_e(r, ent${left:new(), right:new(), s:s, i:i})
    else
      e:ent := rep$value_e(r)
      if e.s = s then     e.i := i
      elseif e.s < s then e.right[s] := i
      else                e.left[s] := i
      end
    end
  end store
  fetch = proc(r:cvt, s:string) returns(int) signals(not_found)
    if rep$is_n(r) then
      signal not_found
    else
      e:ent := rep$value_e(r)
      if e.s = s then    return(e.i)
      elseif e.s < s then return(e.right[s])
      else                return(e.left[s])
      end
    end
      except when not_found: signal not_found end
  end fetch
  entries = iter(r:cvt) yields(string,int)
    e:ent := rep$value_e(r)
      except when wrong_tag: return end
    for s:string, i:int in entries(e.left) do yield(s, i) end
    yield(e.s, e.i)
    for s:string, i:int in entries(e.right) do yield(s, i) end
  end entries
end nametable
```

練習 1 二つの nametable の内容を併合する操作 add を nametable 型に追加せよ。(どっちの実現を用いてもよい。) もちろん、演算子+が使えることを start_up で確認すること。