CLU 言語入門　＃10

久野　靖

1993.10.12

1　前回の練習問題の回答例

最初は sort にちょっと機能を増やして…と思ったのですが、作っているとそれではきたない! と、uniq を別にしてしまいました。どのみち、

```
where t has equal: proctype(t,t) returns(bool)
```

を追加するところがみそですが。

```
% sam19.clu -- sort + uniq

start_up = proc()
  ai = array[string]
  pi:stream := stream$primary_input()
  po:stream := stream$primary_output()
  a:ai := ai$new()
  while true do
    s:string := stream$getl(pi)
    ai$addh(a, s)
  end
    except when end_of_file, bad_format: end
  sort[string](a)
  uniq[string](a)
  for i:string in ai$elements(a) do
    stream$putl(po, i)
  end
end start_up

sort = proc[t:type](a:array[t])
  where t has gt: proctype(t,t) returns(bool)
  at = array[t]
  done:bool := false
  while ~done do
    done := true
    for i:int in int$from_to(at$low(a), at$high(a)-1) do
      if a[i] > a[i+1] then
        x:t := a[i]; a[i] := a[i+1]; a[i+1] := x; done := false
      end
    end
  end
end sort

uniq = proc[t:type](a:array[t])
  where t has equal: proctype(t,t) returns(bool)
  at = array[t]
  i:int := at$low(a)
```

```
  k:int := i + 1
  while k <= at$high(a) do
    if a[i] = a[k] then
      k := k + 1
    else
      i := i + 1; a[i] := a[k]; k := k + 1
    end
  end
  while at$high(a) > i do at$remh(a) end
end uniq
```

2　ルービックキューブのプログラム

いくつかやってみたのですが、あんまり大した例が解けないので少し意地になりまして、ハッシュ関数を使うように直したりしてきたなくなりました。ここでもう 1 回作り直すとよいのですけれど…

```
% solve4 -- breadth first search + hash table

start_up = proc()
  hmax = 59997
  ent = record[c:cube, h:int, l:int, p:int, s:string, done:bool]
  aent = array[ent]
  ai = array[int]
  a:aent := aent$predict(1, 10000)
  hb:ai := ai$fill(0, hmax, 0)
  pi:stream := stream$primary_input()
  po:stream := stream$primary_output()
  c0:cube := cube$read(po, pi)
  cube$write(c0, po)
  stream$puts(po, "correct? ")
  s:string := stream$getl(pi)
  if s = "n" then return end
  h:int := cube$hash(c0)
  aent$addh(a, ent${c:c0, h:h, l:0, p:cube$point(c0), s:"start", done:false})
  hb[h // hmax] := aent$high(a)
  n:int := 1
  m:int := 0
  while true do
    c:cube := a[n].c
    stream$putl(po, "")
    stream$putl(po, int$unparse(n))
    cube$write(c, po)
    c1:cube
    c1 := cube$copy(c); cube$rotfcw(c1);  putent(a, c1, n, hb, "f+")
    c1 := cube$copy(c); cube$rotfccw(c1); putent(a, c1, n, hb, "f-")
    c1 := cube$copy(c); cube$rotbcw(c1);  putent(a, c1, n, hb, "b+")
    c1 := cube$copy(c); cube$rotbccw(c1); putent(a, c1, n, hb, "b-")
    c1 := cube$copy(c); cube$rotlcw(c1);  putent(a, c1, n, hb, "l+")
    c1 := cube$copy(c); cube$rotlccw(c1); putent(a, c1, n, hb, "l-")
    c1 := cube$copy(c); cube$rotrcw(c1);  putent(a, c1, n, hb, "r+")
    c1 := cube$copy(c); cube$rotrccw(c1); putent(a, c1, n, hb, "r-")
    c1 := cube$copy(c); cube$rotucw(c1);  putent(a, c1, n, hb, "u+")
    c1 := cube$copy(c); cube$rotuccw(c1); putent(a, c1, n, hb, "u-")
    c1 := cube$copy(c); cube$rotdcw(c1);  putent(a, c1, n, hb, "d+")
    c1 := cube$copy(c); cube$rotdccw(c1); putent(a, c1, n, hb, "d-")
    n := n + 1
  end
    except when not_found: stream$putl(po, "give up!"); return
           when success(i:int): n := i
    end
  stream$putl(po, "success!")
  x:ai := ai$new()
  while n > 0 do
    stream$putl(po, int$unparse(n)); ai$addl(x, n); n := a[n].l
  end
  for i:int in ai$elements(x) do
    stream$putl(po, a[i].s); cube$write(a[i].c, po); stream$getl(pi)
  end
end start_up
```

```
putent = proc(a:aent, c:cube, l:int, hb:ai, s:string) signals(success(int))
  hmax = 59997
  ent = record[c:cube, h:int, l:int, p:int, s:string, done:bool]
  aent = array[ent]
  ai = array[int]
  h:int := cube$hash(c)
  g:int := h // hmax
  while hb[g] > 0 do
    i:int := hb[g]
    if a[i].h = h cand cube$similar(a[i].c, c) then return end
    g := g + 1
    if g >= hmax then g := 0 end
  end
  aent$addh(a, ent${c:c, h:h, l:l, p:cube$point(c), s:s, done:false})
  hb[g] := aent$high(a)
  if cube$is_goal(c) then signal success(hb[g]) end
end putent
```

クラスタ cube もそれで少し操作を増強しました。

```
cube = cluster is read, write, is_goal, copy, similar, point, hash,
    rotfcw, rotfccw, rotbcw, rotbccw,
    rotlcw, rotlccw, rotrcw, rotrccw, rotucw, rotuccw, rotdcw, rotdccw
  face = array[char]
  rep = record[f,b,l,r,u,d:face]

  %       3 4 5
  %       2 9 6 (U)
  %       1 8 7
  %
  % 3 4 5  3 4 5  3 4 5  3 4 5
  % 2 9 6  2 9 6  2 9 6  2 9 6
  % 1 8 7  1 8 7  1 8 7  1 8 7
  %  (L)    (F)    (R)    (B)
  %
  %       3 4 5
  %       2 9 6 (D)
  %       1 8 7

  read = proc(f1,f2:stream) returns(cvt)
    stream$putl(f1, "reading cube data.")
    stream$putl(f1, "up face.")     fu:face := readface(f1, f2)
    stream$putl(f1, "left face.")   fl:face := readface(f1, f2)
    stream$putl(f1, "front face.")  ff:face := readface(f1, f2)
    stream$putl(f1, "right face.")  fr:face := readface(f1, f2)
    stream$putl(f1, "back face.")   fb:face := readface(f1, f2)
    stream$putl(f1, "down face.")   fd:face := readface(f1, f2)
    return(rep${f:ff, b:fb, l:fl, r:fr, u:fu, d:fd})
  end read

  readface = proc(f1,f2:stream) returns(face)
    stream$puts(f1, "top row: ")  l1:string := readrow(f1, f2)
    stream$puts(f1, "mid row: ")  l2:string := readrow(f1, f2)
    stream$puts(f1, "btm row: ")  l3:string := readrow(f1, f2)
    return(face$[l3[1],l2[1],l1[1],l1[2],l1[3],l2[3],l3[3],l3[2],l2[2]])
  end readface

  readrow = proc(f1,f2:stream) returns(string)
    while true do
      line:string := stream$getl(f2)
      if string$size(line) ~= 3 then
        stream$putl(f1, "3 letters, please.")
      elseif string$indexc(line[1], "rbgywo") = 0 cor
             string$indexc(line[2], "rbgywo") = 0 cor
             string$indexc(line[3], "rbgywo") = 0 then
        stream$putl(f1, "color must be one of r, b, g, y, w, o.")
      else
        return(line)
      end
      stream$puts(f1, "reenter: ")
    end
  end readrow

  write = proc(r:cvt, f:stream)
    stream$putl(f, "    "||toprow(r.u))
    stream$putl(f, "    "||midrow(r.u))
    stream$putl(f, "    "||btmrow(r.u))
    stream$putl(f, "")
```

```
  stream$putl(f, toprow(r.l)||" "||toprow(r.f)||" "||
                                  toprow(r.r)||" "||toprow(r.b))
  stream$putl(f, midrow(r.l)||" "||midrow(r.f)||" "||
                                  midrow(r.r)||" "||midrow(r.b))
  stream$putl(f, btmrow(r.l)||" "||btmrow(r.f)||" "||
                                  btmrow(r.r)||" "||btmrow(r.b))
  stream$putl(f, "")
  stream$putl(f, "    "||toprow(r.d))
  stream$putl(f, "    "||midrow(r.d))
  stream$putl(f, "    "||btmrow(r.d))
end write

toprow = proc(f:face) returns(string)
  return(string$append(string$append(string$c2s(f[3]), f[4]), f[5]))
end toprow

midrow = proc(f:face) returns(string)
  return(string$append(string$append(string$c2s(f[2]), f[9]), f[6]))
end midrow

btmrow = proc(f:face) returns(string)
  return(string$append(string$append(string$c2s(f[1]), f[8]), f[7]))
end btmrow

is_goal = proc(r:cvt) returns(bool)
  return(onecolor(r.f) cand onecolor(r.b) cand onecolor(r.l) cand
         onecolor(r.r) cand onecolor(r.u) cand onecolor(r.d))
end is_goal

point = proc(r:cvt) returns(int)
  return(fp(r.f) * fp(r.b) * fp(r.l) * fp(r.r) * fp(r.u) * fp(r.d))
end point

fp = proc(f:face) returns(int)
  p:int := 1
  if f[1] = f[9] cand f[2] = f[9] cand f[8] = f[9] then p := p + 1 end
  if f[2] = f[9] cand f[3] = f[9] cand f[4] = f[9] then p := p + 1 end
  if f[4] = f[9] cand f[5] = f[9] cand f[6] = f[9] then p := p + 1 end
  if f[6] = f[9] cand f[7] = f[9] cand f[8] = f[9] then p := p + 1 end
  return(p)
end fp

hash = proc(r:cvt) returns(int)
  return(fh(r.f) + fh(r.b)*3 + fh(r.u)*5 +
         fh(r.d)*7 + fh(r.l)*11 + fh(r.r)*13)
end hash

fh = proc(f:face) returns(int)
  h:int := 0
  for k:int in int$from_to(1, 8) do
    h := h * 3 + (char$c2i(f[k]) - char$c2i('@'))
  end
  return(h)
end fh

onecolor = proc(f:face) returns(bool)
  return(f[1] = f[2] cand f[1] = f[3] cand f[1] = f[4] cand
         f[1] = f[5] cand f[1] = f[6] cand f[1] = f[7] cand
         f[1] = f[8] cand f[1] = f[9])
```

```
end onecolor

copy = proc(r:cvt) returns(cvt)
  return(rep$copy(r))
end copy

similar = proc(r1,r2:cvt) returns(bool)
  return(rep$similar(r1, r2))
end similar

simf = proc(f1,f2:face) returns(bool)
  return(f1[1] = f2[1] cand f1[2] = f2[2] cand f1[3] = f2[3] cand
         f1[4] = f2[4] cand f1[5] = f2[5] cand f1[6] = f2[6] cand
         f1[7] = f2[7] cand f1[8] = f2[8] cand f1[9] = f2[9])
end simf

rotfcw = proc(r:cvt)
  rotcw(r.f); mov(r.l,5,6,7,r.d,3,4,5,r.r,1,2,3,r.u,7,8,1)
end rotfcw

rotfccw = proc(r:cube)
  rotfcw(r); rotfcw(r); rotfcw(r)
end rotfccw

rotbcw = proc(r:cvt)
  rotcw(r.b); mov(r.r,5,6,7,r.d,7,8,1,r.l,1,2,3,r.u,3,4,5)
end rotbcw

rotbccw = proc(r:cube)
  rotbcw(r); rotbcw(r); rotbcw(r)
end rotbccw

rotlcw = proc(r:cvt)
  rotcw(r.l); mov(r.b,5,6,7,r.d,1,2,3,r.f,1,2,3,r.u,1,2,3)
end rotlcw

rotlccw = proc(r:cube)
  rotlcw(r); rotlcw(r); rotlcw(r)
end rotlccw

rotrcw = proc(r:cvt)
  rotcw(r.r); mov(r.f,5,6,7,r.d,5,6,7,r.b,1,2,3,r.u,5,6,7)
end rotrcw

rotrccw = proc(r:cube)
  rotrcw(r); rotrcw(r); rotrcw(r)
end rotrccw

rotucw = proc(r:cvt)
  rotcw(r.u); mov(r.l,3,4,5,r.f,3,4,5,r.r,3,4,5,r.b,3,4,5)
end rotucw

rotuccw = proc(r:cube)
  rotucw(r); rotucw(r); rotucw(r)
end rotuccw

rotdcw = proc(r:cvt)
  rotcw(r.d); mov(r.l,7,8,1,r.b,7,8,1,r.r,7,8,1,r.f,7,8,1)
end rotdcw
```

```
  rotdccw = proc(r:cube)
    rotdcw(r); rotdcw(r); rotdcw(r)
  end rotdccw

  rotcw = proc(f:face)
    c:char
    c := f[8]; f[8] := f[6]; f[6] := f[4]; f[4] := f[2]; f[2] := c
    c := f[7]; f[7] := f[5]; f[5] := f[3]; f[3] := f[1]; f[1] := c
  end rotcw

  mov = proc(f1:face,a,b,c:int,f2:face,d,e,f:int,
             f3:face,g,h,i:int,f4:face,j,k,l:int)
    c1:char := f1[a]; c2:char := f1[b]; c3:char := f1[c]
    f1[a] := f2[d]; f1[b] := f2[e]; f1[c] := f2[f]
    f2[d] := f3[g]; f2[e] := f3[h]; f2[f] := f3[i]
    f3[g] := f4[j]; f3[h] := f4[k]; f3[i] := f4[l]
    f4[j] := c1; f4[k] := c2; f4[l] := c3
  end mov
end cube
```