CLU 言語入門　#

久野　靖

1993.9.21

1　ルービックキューブ

もはや言語の機能そのものとは全然関係ないけれど、テキストに出ていたルービックキューブを CLU のクラスタにしてみました。

```
cube = cluster is read, write, is_goal, copy, similar,
    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, "front face.")  ff:face := readface(f1, f2)
    stream$putl(f1, "left face.")   fl:face := readface(f1, f2)
    stream$putl(f1, "right face.")  fr:face := readface(f1, f2)
    stream$putl(f1, "up face.")     fu:face := readface(f1, f2)
    stream$putl(f1, "down face.")   fd:face := readface(f1, f2)
    stream$putl(f1, "back face.")   fb:face := readface(f1, f2)
    return(rep${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[3]])
  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

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
```

```
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 := f[8]; f[8] := f[7]; f[7] := f[6]; f[6] := f[5]
  f[5] := f[4]; f[4] := f[3]; f[3] := f[2]; f[2] := 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
```

さて、これが長いか短いか? わかりやすいかどうか? 本当はこれを使ってパズルを解くのですが、今回は準備する時間がなかったので単にこれでちゃんと本もの通りに動くのかどうか確認する簡単なドライバプログラムを用意しました。

```
start_up = proc()
  pi:stream := stream$primary_input()
  po:stream := stream$primary_output()
  c:cube := cube$read(po, pi)
  while true do
    cube$write(c, po)
    stream$puts(po, "next move> ")
    line:string := stream$getl(pi)
    if     line = "f+"  then cube$rotfcw(c)
    elseif line = "f-" then cube$rotfccw(c)
    elseif line = "b+" then cube$rotbcw(c)
    elseif line = "b-" then cube$rotbccw(c)
    elseif line = "l+" then cube$rotlcw(c)
    elseif line = "l-" then cube$rotlccw(c)
    elseif line = "r+" then cube$rotrcw(c)
    elseif line = "r-" then cube$rotrccw(c)
    elseif line = "u+" then cube$rotucw(c)
    elseif line = "u-" then cube$rotuccw(c)
    elseif line = "d+" then cube$rotdcw(c)
    elseif line = "d-" then cube$rotdccw(c)
    else stream$putl(po, "should be:{f,b,l,r,u,d}{+,-}")
    end
  end
    except when end_of_file: end
end start_up
```

課題 1 cube クラスタを利用して、ルービックキューブのパズルを解くプログラムを作れ。

課題 2 こういう解空間の探索という問題は実はルービックキューブに限らず各種のパズルに対して用いることができる。幅優先の全解探索 (しらみつぶし) を行なう汎用手続きを作ってそれを用いてルービックキューブのパズルを解くことを考えてみる。そのためには抽象データ型 cube に欠けている操作があるが、それは何か。また、これまでに習っただけでは足りない言語機能があるが、それは何か。