CLU 言語入門　＃４

久野　靖

1993.5.25

1　CLU の組み込みの構造型 (2)

先週は申し訳ありませんでした。おかげで CLU の方は２回も間があいてしまいました。(児玉さんはご苦労さまでした。) さて、前回が配列だけで終ってしまったので、残りをやります。その前にイテレータの作り方をやっておきましょうか。例えば

```
% sam09.clu -- sample of record type

start_up = proc()
  pi:stream := stream$primary_input()
  po:stream := stream$primary_output()
  for line:string in file_lines(pi) do
    stream$putl(po, line)
  end
end start_up

file_lines = iter(f:stream) yields(string)
  while ~stream$empty(f) do
    yield(stream$getl(f))
  end
end file_lines
```

1.1　列型

CLU の列 (sequence) 型は、要するに書き換えられない配列。その代わり連結とか部分列とか文字列っぽい操作がある。付録参照。

1.2　レコード型、構造型

レコード型は

record[名前 1:型 1, 名前 2:型 2, … 名前 n:型 n]

という形で指定する。普通は短い名前をつける。作る時には構成子

型名${名前 1:値 1, 名前 2:値 2, … 名前 n:値 n}

による。あとは欄の設定と読み出しがおもな操作。ではサンプル。

```
% sam10.clu -- sample of record type

start_up = proc()
  ent = record[s:string, i:int]
  aent = array[ent]
```

```
  pi:stream := stream$primary_input()
  po:stream := stream$primary_output()
  a:aent := aent$new()
  while ~stream$empty(pi) do
    s:string := stream$getl(pi)
    i:int := int$parse(stream$getl(pi))
    e:ent := ent${s:s, i:i}
    aent$addh(a, e)
  end
  for e:ent in aent$elements(a) do
    stream$putleft(po, e.s, 10)
    stream$putright(po, int$unparse(e.i), 5)
    stream$putc(po, '\n')
  end
end start_up
```

なお、構造 (struct) 型とは、書き換え不能 (immutable) なレコードです。

2 択一型、可変型

構造型の最後に択一型 (oneof) と可変型 (variant) をやります。例の「合併型」ですね。まず oneof は

oneof[名前 1:型 1, 名前 2:型 2, … 名前 n:型 n]

により指定します。操作は付録。さっそく例ですが。

```
% sam11.clu -- sample of oneof type

start_up = proc()
  one = oneof[i:int, n:null]
  ent = record[s:string, o:one]
  aent = array[ent]
  pi:stream := stream$primary_input()
  po:stream := stream$primary_output()
  a:aent := aent$new()
  while ~stream$empty(pi) do
    s:string := stream$getl(pi)
    if s[1] ~= '-' then
      i:int := int$parse(stream$getl(pi))
      e:ent := ent${s:s, o:one$make_i(i)}
      aent$addh(a, e)
    else
      for e:ent in aent$elements(a) do
        if e.s = string$rest(s, 2) then e.o := one$make_n(nil) end
      end
    end
  end
  for e:ent in aent$elements(a) do
    if one$is_i(e.o) then
      stream$putleft(po, e.s, 10)
      stream$putright(po, int$unparse(one$value_i(e.o)), 5)
      stream$putc(po, '\n')
    end
  end
end start_up
```

時間がないので略しますが、可変型は要するに mutable な oneof ですね。

A 構造型の操作 (その 2)

A.1 sequence[T] 型

以下 st = sequence[T] とする。

```
st$new = proc() returns(st)          -- 下限は 1 に固定
st$size = proc(st) returns(int)      -- 要素数かつ上限かつ大きさ
st$empty = proc(st) returns(bool)    -- size = 0?
st$subseq = proc(st, int, int) returns(st) -- 部分列
st$fill = proc(int, int, T) returns(st) signals(negative_size)
st$fill_copy = proc(int, int, T) returns(st) signals(negative_size)
st$fetch = proc(st, int) returns(T) signals(bounds) -- 要素アクセス
st$bottom = proc(st) returns(T) signals(bounds) -- 最初の要素
st$top = proc(st) returns(T) signals(bounds) -- 最後の要素
st$replace = proc(st, int, T) returns(st) signals(bounds) -- 置き換え
st$addh = proc(st, T) returns(st)
st$addl = proc(st, T) returns(st)
st$remh = proc(st) returns(st) signals(bounds)
st$reml = proc(st) returns(st) signals(bounds)
st$e2s = proc(T) returns(st) -- 1 要素の列
st$concat = proc(st, st) returns(st)
st$a2s = proc(array[T]) returns(st)
st$s2a = proc(st) returns(array[T])
st$elements = iter(st) yields(T)
st$indexes = iter(st) yields(int)
st$equal = proc(st, st) returns(bool)
st$similar = proc(st, st) returns(bool)
st$copy = proc(st) returns(st)
```

A.2 レコード型、構造型

以下 rt = record[名前 1:型 1, 名前 2:型 2, … 名前 n:型 n]、st = struct[名前 1:型 1, 名前 2:型 2, … 名前 n:型 n] とする。

```
rt$create = proc(型 1, … 型 n) returns(rt) --- 使ってはいけない。
rt$get_名前 i = proc(rt) returns(型 i) --- 欄の参照
rt$set_名前 i = proc(rt, 型 i) --- 欄の設定
rt$r_gets_r = proc(rt, rt) --- 内容コピー
rt$r_gets_s = proc(rt, st) --- struct からの内容コピー
rt$equal = proc(rt, rt) returns(bool)
rt$similar = proc(rt, rt) returns(bool)
rt$similar1 = proc(rt, rt) returns(bool)
rt$copy1 = proc(rt) returns(rt)
rt$copy = proc(rt) returns(rt)

st$create = proc(型 1, … 型 n) returns(st) --- 使ってはいけない。
st$get_名前 i = proc(st) returns(型 i) --- 欄の参照
st$replace_名前 i = proc(st, 型 i) returns(st) --- 欄の設定
st$s2r = proc(st) returns(rt) --- レコードに
st$r2s = proc(rt) returns(st) --- レコードから
st$equal = proc(st, st) returns(bool)
st$similar = proc(st, st) returns(bool)
st$copy = proc(st) returns(st)
```

A.3 択一型、可変型

例によって ot = oneof[名前 1:型 1, 名前 2:型 2, … 名前 n:型 n]、vt = variant[名前 1:型 1, 名前 2:型 2, … 名前 n:型 n] とします。

```
ot$make_名前 i = proc(型 1) returns(ot)
ot$is_名前 i = proc(ot) returns(bool)
ot$value_名前 i = proc(ot) returns(型 1) signals(wrong_tag)
ot$o2v = proc(ot) returns(vt)
ot$v2o = proc(vt) returns(ot)
ot$equal = proc(ot, ot) returns(bool)
ot$similar = proc(ot, ot) returns(bool)
ot$copy = proc(ot) returns(ot)

vt$make_名前 i = proc(型 1) returns(vt)
vt$change_名前 i = proc(rt, 型 1)
vt$is_名前 i = proc(vt) returns(bool)
vt$value_名前 i = proc(vt) returns(型 1) signals(wrong_tag)
vt$v_gets_v = proc(vt, vt)
vt$v_gets_o = proc(vt, ot)
vt$equal = proc(vt, vt) returns(bool)
vt$similar = proc(vt, vt) returns(bool)
vt$similar1 = proc(vt, vt) returns(bool)
vt$copy = proc(vt) returns(vt)
vt$copy1 = proc(vt) returns(vt)
```

練習問題 数人で分担してプロジェクト。入力ファイルの単語頻度表を作成するプログラム。まず次の型をインタフェースのため使う。

```
ent = record[s:string, i:int]
  単語と頻度の対。
aent = array[ent]
  上記レコードのならんだもの。
```

そしてプログラムは次のモジュールから成る。

```
start_up = proc()
  メインプログラム。空の配列と入出力ファイルを用意して下請けを呼ぶ。
file_words = iter(stream) yields(string)
  ファイルに含まれる単語を次々と返す。
is_alphabet = proc(char) returns(bool)
  与えられた文字が英字 (単語に含まれるべき文字) かどうか判定。
lookup_record = proc(aent, string) returns(ent)
  レコードの配列から渡された単語に該当するレコードを返す。
  (もしまだ配列に含まれていなければ新しく追加。)
sort_record = proc(aent)
  配列を単語の昇順に並べかえる。
print_records = proc(aent, stream)
  配列をファイルに打ち出す。
```