CLU 言語入門　＃ 1

久野　靖

1993.5.11

「データ型序説」読むだけじゃなんとなくつまらないでしょうし、いざデータ抽象の話に来た時にその他の細かいところまで全部説明するととっても大変そうなので、小出しに CLU の説明もやりましょう。できるだけ「序説」の話題と対応する箇所を説明できたらいい具合だと思うのですが、うまく行きますかどうか。

1　最初のプログラム

まずはできるだけ簡単なプログラムから。

```
% sam01.clu -- simple line-by-line copy       % 1

start_up = proc()                             % 2
  pi:stream := stream$primary_input()         % 3
  po:stream := stream$primary_output()        % 4
  while ~stream$empty(pi) do                  % 5
    line:string := stream$getl(pi)            % 6
    stream$putl(po, line)                     % 7
  end                                         % 8
end start_up                                  % 9
```

これの解説は次の通り。

1 %から行末まではコメント。

2 C プログラムは関数の集まりだが、CLU では proc の集まり。

2 主プログラムは start_up という名前になっている。

3 変数定義は「変数名:型指定」による。

3 変数定義は代入文が書けるところならどこにでも書ける。

3 型に付随する操作は「型指定$操作名」の形で書く。

3 stream 型はごく普通の文字単位入出力。代表的な操作は下記参照。

3 pi には標準入力につながった stream を入れる。

4 po には標準出力につながった stream を入れる。

5 入力がおしまいでない間以下を繰り返す。

6 1 行読み込み、その内容を文字列として変数 line に入れる。

7 変数 line の内容を書き出す。

8 while 文の最後は必ず end 必要。そのかわり begin-end はいらない。

9 手続きの終わりは「end 手続き名」。

2 動かし方

ここでは clu2c という処理系によって、CLU を C に変換して動かす。ただし Sun4 だけ。例えば上のプログラムが sam01.clu に入っていたとして、次のようにする。

```
% clu2c #spec sam01 #comp sam01  ← CLU コンパイラ
Creating DU specs from sam01.clu
time = 0.010
Compiling sam01.clu
time = 0.090
% clulink sam01.c  ← C コンパイラ+各種ライブラリ結合
% a.out            ←実行
This is a pen.
This is a pen.
That is a dog.
That is a dog.
^D
%
```

clu2c を使う時は基本的にまず spec 機能でインタフェース情報を取得し、続いて comp 機能で翻訳する。clulink はごく簡単なシェルスクリプトで GCC を呼ぶだけ。

3 主要な制御構造

CLU の制御構造でまず覚えて欲しいのは手続きと宣言/代入と while と if くらい。手続きは

```
手続き名 = proc(引数名,...:型,...) returns(型,...)
    文...
end 手続き名
```

引数が 0 個の時でも () は必要。値は何個でも返すことができる。だから C の関数の感じに近い。値を返さないときは returns は不要。1 つのファイルに何個でも手続きを書ける。

文はとりあえず次のくらい覚えてほしい。

```
変数名,... : 型指定    ←変数定義
変数名 := 式           ←代入
変数名:型指定 := 式    ←上 2 つを併せたもの
操作名 (式,...)        ←手続き呼び出し
while 式 do 文... end ← while 文
if 式 then 文... [elseif 式 then 文...]... [else 文...] end ← if 文
return                 ←値を返さない
```

```
return(式,...)          ←値を返す
break                   ← C の break と同じ
continue                ← C の continue と同じ
```

では次の例題。

```
% sam02.clu -- procedure, int, ...

start_up = proc()
  pi:stream := stream$primary_input()
  po:stream := stream$primary_output()
  num:int := count_lines(pi)
  stream$putl(po, int$unparse(num))
end start_up

count_lines = proc(s:stream) returns(int)
  if stream$empty(s) then
    return(0)
  else
    stream$getl(s)
    return(1 + count_lines(s))
  end
end count_lines
```

これは入力ファイルの行数を数えるプログラムである。上記の説明でだいたいカバーされているが、int$unparse というのが目慣れないでしょう? これは、整数型の値を対応する文字列表現に変換する。その他、+なども使われているけれど、これも実は int$add というのが正式。

```
% clu2c #spec sam02 #comp sam02  ←※
Creating DU specs from sam02.clu
time = 0.060
Compiling sam02.clu
time = 0.140
% clulink sam02.c
% a.out <sam02.clu
17
% wc sam02.clu
  17      38     371 sam02.clu
```

練習 1: 上の例題群を打ち込んで実行してみよ。

練習 2: count_lines をもっと普通の実現に直してみよ。

練習 3: ファイルの 10 行目だけを打ち出すプログラムを作れ。

練習 4: ファイルを上下さかさまに打ち出すプログラムを作れ。

練習 5: ファイルを上下左右さかさまに打ち出すプログラムを作れ。

4 stream の主要な操作

```
primary_input = proc() returns(stream)  -- 標準入力
primary_output = proc() returns(stream) -- 標準出力
error_output = proc() returns(stream) -- 標準エラー出力
empty = proc(stream) returns(bool) signals(not_possible(string)) -- EOF か?
getc = proc(stream) returns(char) signals(end_of_file,not_possible(string)) -- 1 字読む
getl = proc(stream) returns(string) signals(end_of_file,not_possible(string)) -- 1 行読む

gets = proc(stream,string) returns(string)
    signals(end_of_file,not_possible(string)) -- 第 2 引数のどれかの字まで読む
putc = proc(stream,char) signals(not_possible(string)) -- 1 字書く
putl = proc(stream,string) signals(not_possible(string)) -- 1 行書く
puts = proc(stream,string) signals(not_possible(string)) -- 文字列を書く
```

5 int の主要な操作

```
add = proc(int,int) returns(int) signals(overflow) -- 加算
sub = proc(int,int) returns(int) signals(overflow) -- 減算
mul = proc(int,int) returns(int) signals(overflow) -- 乗算
minus = proc(int) returns(int) signals(overflow) -- 符号かえ
div = proc(int,int) returns(int) signals(zero_divide,overflow) -- 除算
mod = proc(int,int) returns(int) signals(zero_divide,overflow) -- 乗余
power = proc(int,int) returns(int) signals(negative_exponent,overflow) -- べき
abs = proc(int) returns(int) signals(overflow) -- 絶対値
max = proc(int,int) returns(int)
min = proc(int,int) returns(int)
parse = proc(string) returns(int) signals(bad_format,overflow) -- 文字列から
unparese = proc(int) returns(string) -- 文字列に変換
lt = proc(int,int) returns(bool)
gt = proc(int,int) returns(bool)
le = proc(int,int) returns(bool)
ge = proc(int,int) returns(bool)
equal = proc(int,int) returns(bool)
similar = proc(int,int) returns(bool) -- int では equal と同じ
copy = proc(int) returns(int) -- int では引数をそのまま返す
```