フレームワークの一構成法

5M－3

恐神正博　西田富士夫
福井工業大学

１．はじめに

近年、モジュールやクラスなどの再利用の研究が盛んである。ここでは、モジュールの組み合わせを再調整して、求める機能を持つプログラムやモジュールを作成するためのフレームワークの構成とその適用法、ならびに、構造体やレコードを対象として一般モジュールの構築と利用について述べる。

２．フレームワークの構成

仕様からプログラム設計を行うときや、既成のシステムを修正するときには、仕様や既成の設計書のほかに、利用できるフレームワークやモジュールをフレームの一覧表から探索する。ここにフレームワークは、ある機能や処理の枠組みを実現するためにいくつかのまとまった処理からなり、下位のフレームワーク名や、モジュール名を並べたものからなっている。モジュールや命令はそのグループ毎の見出し辞書に、それぞれ関数名など一つの呼出文をもっており、この呼出文をモジュール名とする。これらの呼出文は、入力データとしていくつかの変数をパラメータとして含む[1]。これに対しフレームワークの変項名は処理文単位のもので、例えば、空を含む処理文名の PROC である。フレームワークの名前は、１つのタイプ名を示すもので、定項名であっても、その構成要素であるフレームワークやモジュールはいろいろ異なるものをとることができるものとする。フレームワークはファイル処理や図形処理などの応用分野や制御フレームなどに分け、各分野毎にフレーム名やモジュール呼出文をツリー上に配置する。図１に例として、ファイル処理の中、ファイルからの繰り返し入力演算処理に関するフレームワークを示す。

```
dict([headline( ファイル繰り返し読み込み演算 ,
 file_rep_rd_comp),[
  body([
   p([1,N, ファイルオープン ,
      [1, ｵｰﾌﾟﾝ失敗時の表示付き ,
       2,file_open 見出し辞書 ])
   p([2,P0, 初期値設定 ,
      [1,compute 見出し辞書 ]),
   p([3,N, くり返し制御文 ,
      [1, 有限回 ,2,eof まで ,
       3,repeat 見出し辞書 ]),
   begin_iter,
    p([4,N, レコード読み込み ,
       [1, ﾍﾞｸﾄﾙ 2, 構造体
        3,read 見出し辞書 ]),
    p([5,P1, 反復計算処理 ,
       [1,compute 見出し辞書 ]]),
   end_iter,
   p([6,P2, 後計算処理 ]),
   p([7,N, 出力表示 ,([1,print 見出し辞書 ])
 ])],     ...     ]).
```

図１

ここに body 部の各項目 p は見出しの機能を達成するための各手続きの概要であり、ユーザはこれらを次に詳細化すべきモジュールなどの呼出を各 p の第４項目で与えた選択項目の中から番号で N に選定する。また、Pi には計算式を指定する。これにより対応する処理の見出し文を選び変数をカスタマイズなどして詳細化する。初期設定や繰り返し計算式などは、ユーザが所望のものを入力する。詳細化におけるおける記号列の置き換え、挿入、削除などの演算はプロログの上の記号演算で行う。

３．構造体処理モジュールの構築

近年、モジュールの再利用の重要性が叫ばれいろいろ研究がなされているが、所要のものに適合する再利用可能なモジュールが得にくいのが現状

A method of constructing frameworks
Masahiro Osogami and Fujio Nishida
Fukui University of Technology

のようである[2]・ここに再利用可能とは関数の引数の変数名などを指定することにより、そのまま利用できるソース言語の既成のモジュールの他、短時間にこれを合成し、利用できるモジュールを指すものとする。変数や配列を引数とするＣ言語の多くの関数は前者に属するが、任意の構造体を引数にもつ対象を処理するモジュールは現在のＣや、Ｃ＋＋の機能では作成が困難なように思われる。ここでは、後者の場合、すなわち、 構造体のある処理に関する一般の手続きを記号処理が比較的容易な、 PROLOG で作成しておき、適用する構造体のデータ構造を入力すれば、Ｃなどの対応するモジュールに自動変換する手法について述べる。これらの一般的な処理モジュールの呼出文は以下のようなものである。

(1) 構造体配列 ,A, に (kb /,FILE,) から ,
(INDEX/POINTER), を用いて ,(N, 回 /END, まで)
繰り返し読み込む。

(2) 構造体配列 ,A, から (display/,FILE,) に ,
(INDEX/POINTER), を用いて ,
(N 回 /END, まで) くり返し書き出す。

(3) 大きさ ,N, の構造体配列 ,A, を ,MEMBER, をキーとして (ascend/descend) order にソートする。

(4)FILE から ,(INDEX/POINTER), を用いて条件 ,
COND, を満たす構造体を表示し構造体配列 ,TR,
にコピー したり ,NTR, で、FILE, を更新する。

次にソートを例に取り、一般的な構造体から、指定した構造体処理のモジュールを自動作成する一つの方法の概要を述べる。ソートの場合、あるメンバをキーとして大小関係を比較するのであるが、文字列を比較する場合と数を比較する場合とで処理が一般に異なり、Ｃでは、関数 strcmp や関係演算子 '>' などを場合に応じて使い分けている。 [KEY_NAME,[SIZE]] なる（配列型）変数が構造体Ｘのメンバリスト Ｌの文字列型メンバで、これをソートのキーに選ぶとき、にソートの比較部分とスワップ部分のＣ生成部を①に示す。

```
c(sort_struct(X,I,L,KEY_N)):-
    (member([[KEY_N,[S]],string],L),
     c(if(conv(greater(t(strcmp([X,I,KEY_N],
          [X,nxt(I),KEY_N])),t(0))),
               swaplist(X,I,nxt(I),L)));
     c(if(conv(greater(t([X,I,KEY_N]),
                    t([X,nxt(I),KEY_N]))),
               swaplist(X,I,nxt(I),L))) ).          ①
```

②は任意の長さのメンバリストをもつ 構造体Ｘの配列要素ＩとＪの各メンバを交換する関数の作成部である。

```
c(swaplist(X,I,J,[[LNH,LTH]|LT])):-
                swap(X,I,J,LNH,LTH),
                c(swaplist(X,I,J,LT)).
c(swaplist(X,I,J,[])).                              ②
```

③ａは交換するメンバが文字列の場合、作業用の文字列型配列 tmp を用いて、 strcpy 関数によりスワップを行う手続きを出力する。 write_sp はインデント用の指定個数のスペースを出力する。 交換するメンバーが数のときには③ａにおいて作業用の整数型などの変数 tmp_num や代入演算子 '=' に置き換え③ｂのようにすればよい。

```
swap(X,I,nxt(I),[LNH,[LSH]],string):-
  write_sp,writelist([
  'strcpy(tmp,',X,'[',I,'].',LNH,');']),nl,
   write_sp,writelist
  (['strcpy(',X,'[',I,'].',
       LNH,',',X,'[',I,'+1].',LNH,');']),nl,
   write_sp,writelist
  (['strcpy(',X,'[',I,'+1].',
       LNH,',',tmp,');']),nl.                       ③ａ

swap(X,I,nxt(I),LNH,int):-
  write_sp,writelist
  (['tmp_num = ',X,'[',I,'].',LNH,';']),nl,
  write_sp,writelist([X,'[',I,'].',LNH,' =          ',
  X,'[',I,'+1].',LNH,';']),nl,
  write_sp,writelist([X,'[',I,'+1].',LNH,=          ',
  tmp_num,';']),nl.                                 ③ｂ
```

同様にして、入出力や更新などの処理に関しても、一般の構造体に対する処理記述から指定した構造体に対する処理を行うモジュールを自動生成することができる。

[参考文献]

[1]恐神正博，西田富士夫： 仕様からの構造化図の簡易作成とプログラムの合成，情報処理学会第４６回全国大会8J-5

[2]B. メイヤ著，二木厚吉監訳：オブジェクト指向入門 第３章