ファイル処理を含むプログラム作成の自動化

5U−3

恐神　正博　　西田　富士夫
福井工業大学

1．まえがき

従来、プログラムの作成やその自動化には、基本的な手続きを機能単位にまとめ、変数パラメータを含んだ一種の関数形に機能モジュールとして一般化し再利用する方法がよく知られている。これはいろいろな処理や機能を、アルゴリズムの内容や手続きの詳細は省略して、慣用的な手続き名と二，三の主なパラメータを指定するだけで比較的広い範囲にわたって再利用を可能にしようとするもので、プログラム作成を効率的にし、またプログラム全体を見やすくするものである[1]。

他方、仕様やプログラムに共通によく現れるものに、制御や処理の枠組みのパターンがあり、これらを積極的に定式化して再利用し、考察の中心を枠組み内の特定処理の詳細化に絞ることなどが考えられる。枠組みには繰り返しや場合分けなどの一般の制御に関するものや、大きなものではいろいろな編成のファイル処理に関する慣用的なフレームなどがある。これらのフレームは定形的な構造や機能をもっているが、このままでは、上記の機能モジュールと有機的にリンクして再利用することは難しい。フレームによる対象の処理は、フレーム固有の処理や資源によりいくつかの段階に分かれ、従って機能モジュールの手続きは、一般にいくつかのサブフレームに分割される。機能モジュール手続きの分割、配置や結合は、従来のような基本モジュールの入出力条件に基づく論理的結合のみによって能率的に行うのは困難である。一つの方法としてフレームにはサブフレーム名を付けてパターン化し、他方、機能モジュールの手続きは適用するフレームのサブフレーム毎に、その名前を付けて分割して利用することが考えられる。以下では順ファイルのデータ処理を対象としてフレームの再利用によりコボルやCなどのプログラムを生成する一手法について述べる。

2．対象の型とデータ構造

通常のプログラム仕様と同様に、はじめに主な対象のデータ構造や型を選定しまた必要に応じて初期値を指定する。データ構造が、レコードや構造体の場合には項目やメンバの名前や型、構造を次のように選定する。

```
obj(name(NAME),type(TYPE),value(VALUE)).
obj(name(NAME),type(record),items([
        [name(NAME),type(TYPE),value(VALUE)],
        ..............................] ]).
obj(name(NAME),type(file),record(RECORD_NAME)).
```

(1)

なお、型指定では出力時の表示の指定と兼用させるためにCOBOLのPIC指定に対応して　int(z3d5) float(f8.3)のように、オプションで引き数部に出力時の印字フォーマットを指定するものとする。また、モジュールの引き数や中間変数などに現れる関連対象の型やデータ構造は仕様に出現の度毎に自動的にチェック、登録、管理して必要に応じて一括表示するものとする。

3．順ファイル処理のフレーム

1個の順ファイルSEQ_FILEの順次データ処理は一般に

ファイルオープン，　初期読み込みと前処理、
繰り返し制御文、　レコード読み込みと順次処理、
繰り返しの終わり、ファイルクローズ、後処理

(2)

などからなり、これらは次のようなクローズのボディ部を構成する各サブフレームで表される。

```
seq_file_proc_frame(in(SEQ_FILE),spl(SPL)):-
        file_open(SEQ_FILE),
        pre_processing(SEQ_FILE),
        repeat_preset_list(SPL),
        repeat_until_end(SEQ_FILE),
           repeat_body(SPL,SEQ_FILE),
        repeat_end,
        file_close(SEQ_FILE),
        post_processing.
```

(2a)

からなる。このようなフレームを用いれば、仕様の概略は、(1)の対象のデータ構造に基づいて、(2a)の頭部の具象形の否定形で与えることができる。ただし、SPLは Sequential_file_data　の概略の処理（Processing）のListを表す。

各サブフレームにおいては、標準的な処理を指定する他、必要に応じて既成のモジュールを検索利用したり、新しいモジュールを作成して、仕様を詳細する[2]。

① file_open(SEQ_FILE):-
　　open_file(SEQ_FILE,read).
② read_file_processing(SEQ_FILE):-

Frame_based Program Generation
in File Data Processing
Masahiro Osogami and Fujio Nishida

```
        read_file(SEQ_FILE),
        file_end_processing.
③   repeat_body(SPL,SEQ_FILE):-
        SPL,read_file_processin(SEQ_FILE).
                                        (2b)
```

例えばファイルの終わりで、FILE_END_PROCESSに代入した処理を行わせたいときには、

```
    file_end_processing:-
  file_end_process(FILE_END_PROCES).    (2c)
```

などで指定する。同様に、SPLの要素であるsp(SP)などの処理の詳細を(2c)のような形で指定することができる。詳細化の後、必要に応じてコボルやCのプログラムに変換する。

4．フレーム内におけるモジュールの適用

節1．で述べたように、通常の機能モジュールをある枠組みで適用するには、モジュールの手続きのどの部分を枠組みのどの部分に配置すべきかを、手続きにサブフレーム名を付加して示せばよい。以下では、このためのモジュールの構成とこれらをフレームに自動的に組み込む1つの方法を例により示す。

(3)は順次和を求めるモジュールの一部であるが、そのプログラム部分を埋め込むべきサブフレーム名を関数記号として前置している点が通常のメモリ上の演算モジュールと異なる点である。(4)は(2)の頭部のspl(SPL)で指定した順次処理列のモジュールの中、サブフレームrepeat_presetに関する手続き部分を取り込む（プロログ）プログラムを示す。

```
file([head(順次和を求める),
   arg_form([[変数,'SEQ_SUM',に],['OBJ',の]]),
   com([変数,SEQ_SUM,に,OBJ,の順次和を求める]),
   arg([SEQ_SUM,OBJ]),num(106)],
   [pred([repeat_preset(set(SEQ_SUM,0)),
         repeat_body(set(SEQ_SUM,plus(SEQ_SUM,OBJ)))
        ]),
    prog([repeat_preset([SEQ_SUM,' = 0']),
          repeat_body([SEQ_SUM,' = ',
                   SEQ_SUM,' + ',OBJ])],cobol)
   ]).
                                        (3)
repeat_preset_list([H|T]):-
     repeat_preset(H),repeat_preset_list(T).
repeat_preset_list([]).
repeat_preset([HEAD,ARG]):-
     file(H,T),member(head(HEAD),H),
     member(arg(ARG),H),
     member(prog(PROG,cobol),T),
     member(repeat_preset(PRESET),PROG),
     writelist(PRESET),nl.              (4)
```

このような順次和を求める基本モジュールに基づいて、グループ別集計やクラス別の頻度計算、相関分析のような時系列解析など、ファイルの上のデータを対象とする各種のデータ処理モジュールを作成することができる。(5)は 同じファイルの同じレコード上の2個のアイテムX,Y間の相関係数を求めるモジュールのprog記述部のmodule部である。

```
prog([mod( av(AVX,X),av(AVY,Y),
       seq_sum(X2,sq(X)),seq(Y2,sq(Y)),
       seq(XY,times(X,Y)) )],
       post_processing([compute ...]) ]).
                                        (5)
```

ここにmodの引き数部は（3）に示したような既に定義された機能的なモジュールを表し、 post_processingの引き数部は(2a)のフレーム内のpost_processingなるサブフレームにおける計算手順を表す。アイテムX,Yが異なる順ファイルSQF1,SQF2に存在するときには(2a)のフレームにおいてSQFの代わりにこれらの2個のファイルで置き換えたフレームの中で処理を指定する。

5．順ファイル更新フレームの例

昇順にソートしたマスターファイルMSTR_FILEを、昇順にソートしたトランサクションファイルTRNS_FILEで更新したり挿入削除して,ニュウマスターファイルNEW_MSTR_FILEを作成するフレームseq_file_update_frame(MSTR_FILE,TRNS_FILE,NEW_MSTR_FILE)は、これらの3個の引き数をもった(2a)と類似のフレームで指定利用することができる。なお、各サブフレームは次にように設定する。

```
file_open(MSTR_FILE,TRNS_FILE,NEW_MSTR_FILE):-
    open_file([MSTER_FILE,TRNS_FILE],read),
    open_file(NEW_MSTR_FILE,write).
pre_processing(MSTR_FILE,TRNS_FILE):-
    read_file(MSTR_FILE),read_file(TRNS_FILE).
repeat_body(MSTR_FILE,TRNS_FILE):-
  (cond(less(code(MSTR_FILE),code(TRNS_FILE)) ),
    write(NEW_MSTR_FILE,from(MSTR_FILE)),
    read_file(MSTR_FILE));
  (cond(equal(code(MSTR_FILE),code(TRNS_FILE)) ),
     update(UPDATE),read_file(TRNS_FILE));
  (cond(grtr(code(MSTR_FILE),code(TRNS_FILE)) ),
       error_processing(ERROR_PROCESSING),
       read_file(TRNS_FILE)).
                                        (6)
```

参考文献

[1] 西田,高松：プロログによるプログラム生成システム 情報処理学会第42回全国大会 5R-2(1991)
[2] 恐神,西田：仕様の記述とプログラムの作成 情報処理学会第44回全国大会 2J-8(1992)