フレキシブルなモジュールの構築について

7K－8

恐神正博　西田富士夫
福井工業大学

1．はじめに

プログラムやモジュールの再利用が提唱されているが、一見、千差万別に見えるユーザのニーズに適合したモジュールをうるのは必ずしも容易でないと言われている[1]。従来、モジュールは機械コードやコンパイラ言語で書かれ、処理のデータ構造や制御はマシンのロジックに即して所与のものとし、主要なデータの値や呼び名のみを引数を通してユーザが指定することが出来た。ところが、最近ではプログラムの変更や拡張要求の機会が増大する一方で、ソースコードへの変換生成技法が進み変換生成時間があまりかからなくなってきている。従って、データ構造や制御の枠組みなどもカスタマイズできるようにフレキシブルな汎用モジュールを設計文書や中間言語の表現形式で作成しておき、適用時に引数を指定してソースコードに変換する手法が考えられる。設計時に利用出来るモジュールを検討することはより自然であり、また、重要である。ここではＣの関数モジュールを対象として考える。

2．変数のデータタイプについて

Ｃ言語でモジュールを作成するには、変数のデータタイプが異なれば同じ処理であっても、一般にタイプ毎に別の関数として準備しなければならない。そこで汎用モジュールの本体はプログラムの設計と同様にカスタマイズしやすい設計言語で書いておき、データタイプはモジュールの適用時に所望のものを指定しＣに変換し、実行する方法が考えられる。例えば「大きさnタイプ,TYPE,の１次元配列,A,の配列要素の和をオート変数tmpに求める」という汎用モジュールの呼出文には、この呼出文に対応する述語suma(A,TYPE)を

```
suma(A,TYPE):-
  fd([TYPE,FUNCT_SYMB,[[TYPE,A]]),
  vd([[TYPE,tmp],[int,n]]),proc([..]).  (1)
```

のように定義する。fdは関数の頭部の記述項で関数の戻り値のタイプ、関数記号、関数引数部の変数名とタイプなどを記述する。vdは局所変数記述項であり、局所変数名やデータタイプ名を記述する。PROCは手続き部であり、入出力文などタイプ指定が必要な場合には、TYPEによる直接指定、またはfd部やvd部の参照変換などによりＣのソースコードを生成する。なお、必要に応じて普通のデータタイプの他、桁数指定付きタイプを用いる。

3．構造体の処理モジュール

通常の一つのタイプよりなる配列データの基本処理の汎用モジュールは作成されているが、構造体やレコードの基本処理に対する汎用モジュールは従来からあまり作成されていない。このような処理もリスト処理が容易な言語で汎用モジュールを作成しておき、処理対象のデータ構造の入力により、具象化してソースコード化すればシステム拡張や変更に対して便利である。試作システムMAPPでは構造体のいろいろな基本処理を、一般的なデータ構造に対してプロログで記述し、日本語で呼び出して汎用モジュールを作成し、これに構造体タグのデータ構造を入力してＣのモジュールを作る[2]。各汎用モジュール述語には日本語文による呼出文、その述語形、Ｃの関数形、プロトタイプなどを付加してメインプログラム作成の自動化を図っている。

先ずはじめに構造体のデータ構造をMAPPからの問い合わせにより入力する。これは(2)に示すようなリストまたはツリー状のデータ構造をキーボードから順次入力する。次にユーザは処理の概略設計に基づき、フレーム処理、構造体処理、基本関数処理などの呼出文辞書を参照して、必要な呼出文をカスタマイズして構造図の所望の位置に、置き換えや挿入を行い設計文書を作成する。構造体処理の場合には、どのような汎用モジュールがあるかを見出し辞書を表示一覧し、所要のも

A mehod of Constructing Flexible Modules
Masahiro Osogami and Fujio Nishida
Fukui University of Technology

のを番号で指定し、Cの関数引数としては導入が難しい条件文などを引数の値として指定しプログラムに埋め込む。また、この関数の実引数や仮引数名やこの関数を収納するファイル名などを指定する。このようにしてカスタマイズされた呼出文をMAPPはユーザの指定した付加行番号位置に挿入したり置き換える。また、実引数のタイプ名や、処理関数のプロトタイプなどは、未登録なら自動的に各タイプ述語に登録する。メイン文書の呼出文のソースコードへの変換は各変換述語を参照して自動的に行う。構造体処理の場合、汎用モジュール述語の関数表現を参照して呼出文をCの関数に置き換える。また、呼出文の述語表現により、構造体のデータ構造と仮引数を用いて、汎用モジュール述語の処理本体部をCのソースコードに変換詳細化し、指定したファイルに書き出す。(3)は呼出文の1例で4個の引数に例のような値を指定する。カスタマイズした呼出文は(4)の対応するステートメント述語のjp部により、その述語形に変換し、Cのソースコードを作成し、関数名やプロトコルを参照してメイン部を自動的に作成する。

```
struct_member(stock_item,[[[name,[20]],string],
  [code,int],[price,int],[number,int],[in_date,
  struct(date)]]).

struct_member(date,[year,int],[[month,int],[day,
  int]]].                                    (2)
p(6,[FILE,N,COND,RETR_AR],[ﾚｺｰﾄﾞ数,N,のﾌｧｲﾙ,
  FILE,を読み込み条件,COND, を満たす構造体を
  表示し構造体配列,RETR_AR,に出力する /* 引数列
  の例 [fn,n,[tmp.number,>,750],retr_ar]  */
  ']).                                       (3)
statement([jp([ﾚｺｰﾄﾞ数,N,のﾌｧｲﾙ,FILE,
  を読み込み条件,COND,を満たす構造体を表示し
  構造体配列,RETR_AR,に出力する],
  gen_funct(retrfile_to_str_parr(FILE,N,
                    COND,RETR_AR)) ),
  proto([pv(0),[void,retrfile_to_str_parr(
       char('*'),int,struct(STRUCT) )]],
       [pv(1),[void,retrfile_to_str_parr(
       char('*'),int,struct(STRUCT,'*') )]]),
  funct(retrfile_to_str_parr(FILE,N,RETR_AR))
  ]).                                        (4)
```

4．フレームの導入

モジュールの可変性を増すには一般のプログラム設計と同じくトップダウン設計が有力であり、また、ある処理のまとまり毎にフレーム化することが有用な一つの方法と考えられる。ここでフレームとは、ある枠組みの中の各スロットに、項の他に、いくつかの処理や性質などの引数を指定して、あるまとまった処理や関係を表すものとする。その基本的で汎用的なものに制御フレームがある。

制御フレームもタイプを分類しておけば要求に適合する可変な構造に設定することが出来る。図の2行目以下は条件群t,t1,...,tnの順に条件tiを始めて満たす場合に指定した処理を実行するif_else_if文のフレーム図であり、フレーム見出し番号の指定と、if_else_if([COND,COND_L]).の引数部の、[t,[t1,...,tn]]を指定することにより、フレーム図が得られる。このようにして場合分けの変更などの場合、比較的容易にモジュールの変更を行うことが出来る。例えば、頻度計算の区間変更の場合、場合分けの条件変更だけで、各場合の処理は変わらない。

データ数が既知の場合と不明な場合の繰り返し制御フレームの選択も同様に行うことが出来る。

```
○--start
|--:[1,proc]
|---<>---(IF:[2,t])
|     |-then---:[4,proc]
|     |     |--end_then(5)
|     |---<>---(ELSE_IF:[6,t1])
|     |     |--:[7,proc]
|     |     |--end_else_if(8)
|     |---<>---(ELSE_IF:[9,t2])
|     |     |--:[10,proc]
|     |     |--end_else_if(11)
|     |-else---:[13,proc]
|     |     |--:[14,end_else]
|--:sp_end
```

[参考文献]

[1] B．メイヤ著，二木厚吉監訳：オブジェクト指向入門，第3章，サイエンス社

[2] 恐神正博，西田富士夫：フレームワークの一構成法，情報処理学会，第49回全国大会，5M-3