クラスモジュールの一作成法

6M－5

恐神正博　西田富士夫
福井工業大学

１．まえがき

近年、いわゆるOODやOOPの手法が導入され、大きな応用プログラムの開発や保守への利用が進んでいる。これは処理関連対象のデータ構造と、これらに対するいくつかの処理とをひとまとめにしてクラス化した方が、手続きだけのクラス分けと汎用化よりもモデル化し易く一まとまりのモジュールとして実用しやすいことによっている。クラスはデータメンバを通じての処理の緩やかな結合と見ることができ、必要に応じてメソッドを追加したり置き換えたりすることができる。また継承性の活用技法の開発により、プログラムの小変更が容易になり、特に自家用の応用システムの保守や拡張に対しては有用である。しかし、クラスの新構築や、再構築において、必要な汎用クラスや汎用メソッドの作成法や検索方法などはまだ十分開発されていないように思われる。汎用性のあるモジュールを作成するには、モジュールを抽象度のやや高い設計文のレベルで書いておき、適用時にリスト形の処理対象のデータ構造により具体化することが考えられる。この講演では、日本語文風の引数名置換可能なCの関数呼出文を用いて、プロログで書いたCのライブラリ関数を呼び出し、処理対象のデータ構造に対応して引数を指定し、クラスのメソッドやクラスを作成する１つの方法を述べる。変換システムはプロログによるMAPP, 目的言語はC++である。

２．メソッドの追加，置き換え

メソッドを新しくクラスに追加するとか、変更しようとするとき、所望の機能を持ち、引数名置換可能な関数があり、これを再利用できれば好都合である。引数名置換可能な関数とは、関数頭部の仮引数変数名をカスタマイズする（実変数名に改名する）ことにより関数本体部の仮引数名も実引数名に置き換わる関数をいう。適用時には局所変数名との衝突の注意が若干必要となる。MAPPではプロログの単一化の技法を用いてこのような関数本体部を簡単に作成している。引数名置換可能な関数は通常のCのユーザライブラリ関数と同様に関数プロトタイプを定義することができる。また、関数プロトタイプの関数名と仮引数名に若干の情報を補ってこの関数がもつ機能を、１～２行程度の日本語文などで入出力用に用いることができる。(1)(2)はこれらの語文の例をその述語形とともに示す。

print_size_num(X, M, N)　(1)

X, を大きさ, M, 桁小数部桁で, N, 個のデータ毎に改行印字する。

bin_search(A, SIZE, KEY)　(2)

大きさ, SIZE, の２次元配列, A, の列, KEY, をキーとして２分割探索する。

これらの語文は関数呼出文として機能別に表に集めてユーザの要求に応じて表示する。

ライブラリ関数の実引数は、メソッドして取り込むクラスのデータメンバとなることが多い。メソッドの関数プロトタイプはこのようなデータメンバをライブラリ関数のプロトタイプの引数変数から取り去って作る。

３．メソッド用ライブラリ関数の作成

メソッドをクラスに補充する際、適用できる関数が関数ライブラリの中に見つかない場合には、基本関数や分岐や繰り返しの制御（フレーム）関数の呼出文を呼び出し、構造化図の上で挿入置換などの編集を行って求めるライブラリ関数を作る[2]。このときタイプを除いて関数の処理が同じ場合、タイプ毎に関数を設けるのは非能率である。それで必要に応じていくつかのタイプ変数 TYPEi

---

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

を設け、局所変数のタイプ宣言などの関数本体部の記述にはタイプ変数で記述し、ライブラリ関数適用時に、データメンバと同様に、タイプを指定して関数の本体部をカスタマイズする。また、入出力関数として、 scanf 関数や printf 関数を用いたときには指定した入出力変数のタイプに対応するフォーマットを自動的に出力する。このようにライブラリ関数は主要な処理関連対象を引数変数とし日本語文などの呼出文をつけて関数ライブラリに登録する。

4．クラスモジュールの作成

例えば事務処理においてはレコードのような構造体データの登録、入出力、検索、更新などの処理をまとめて行う場合が多い。このような場合、どの構造体にも適用できる汎用クラスモジュールを作成しておくことが望まれる [1]。ここでは２節の関数呼出文を集めてクラスモジュールを作成する１つの手法について述べる。例を (3) に示す。クラス名には分類名の他、詳細化項目を入力する。データメンバには処理関連対象のデータ名とタイプ名をリスト形などの形で入力する。さて、次にメソッドであるが、２節のライブラリ関数の見出表から関連するものを見出番号を指定して選定する。カスタム用変数はデータメンバの変数名を指定してカスタマイズしクラスのメソッドに取り込む。

```
クラス名（レコード更新([cat(ファイル処理),
    sub_cat(kb 入力による )]).                    ①
データメンバ([[struct, A1],  *ptr],
    [struct, A2], *ptr2], [fl, FL],
    [int, n1], [int, n2]]).                       ②
[[ 大きさ, N1, の構造体配列, A1 にファイル, FL1,
から繰り返し読み込む ],                           ③
[ 大きさ, N2, の構造体配列, A2, に kb から,
N, 回繰り返し読み込む ],                          ④
[ 大きさ, N1, の構造体配列, A1, から条件,
COND1, を満たす構造体を検索表示する ],            ⑤
[ 大きさ, N1, の構造体配列, A1, において条件,
COND2, を満たす構造体を更新する ],                ⑥
[ 大きさ, N1, の構造体配列, A1, からファイル, FL1,
に繰り返し書き込む ]].                            ⑦
                                                  (3)
```

次にこのようにして作成した汎用クラスモジュールを当面する処理対象のデータ構造に適用して所要のファイル更新クラスを作成する方法を述べる。まず、ユーザは (4a)(4b) のような処理対象のデータ構造を構造体あるいはクラスとして定義し入力する。

```
struct_member(lend_item, [[[name, [30]], string], [code, int],
[lend_date, date], [usr_id, int],    [[usr_name, [20]], string],
[return_date, date]]).                            (4a)
struct_member(date, [[year, int], [month, int],    [day, int]]).
                                                  (4b)
```

(4a) はある図書館の図書貸し出し帳更新におけるデータメンバで、 MAPP からのメンバ入力プロンプトによりユーザは構造体のタグ名、データメンバとタイプのリストを入力する。 MAPP は各メソッドのプロトタイプ作成後、各メソッドの本体部を関数本体部の処理対象の変数リストを (4) のデータメンバで具象化して作成する。例としてコンソールからの貸出データ入力のメソッド (3)-④の本体部作成について概略を説明する。 (3)-④のメソッドの呼出文の述語形から規則 (5) の頭部が呼ばれ、先に指定した (4) の述語 struct_member の引数が述語 read_member に渡され、 (6) により (4) の構造体の各メンバに対し入力プロンプトメッセージ付き
scanf 関数や cin ストリームが作成される。

```
c(read_struct(STRUCT, X)):-
        obj([name(X), type(struct(STRUCT))]),
        struct_member(STRUCT, L),
        c(read_memberlist(X, L)).                 (5)
c(read_memberlist(X, [[HN, HT]|T]):-
        c(prompt_read_member([X, HN], HT)),
        c(read_memberlist(X, T)).                 (6)
c(read_memberlist(X, []).                         (7)
```


[参考文献]

[1] 恐神正博, 西田富士夫：フレキシブルなモジュールの構築について, 情報処理学会, 第50回全国大会, 7K-8

[2] 恐神正博, 西田富士夫：プログラム設計と構造化図作成の自動化, 福井工業大学研究紀要, 24号, pp. 261-268, (1994)