プログラムスキーマに基づく編集環境における ユーザインタフェイスと作業能率

吉沢 亮吉　渡辺 慎哉　宮本 衛市
（北海道大学 工学部）

1. はじめに

われわれはプログラムを作成する際に、以前に作ったことのあるプログラムやサブルーチンと類似したものを再度作成するということをよく経験する。そこで、あるまとまった処理を行うサブルーチンを、あらかじめ多数ファイルに蓄積しておき、新たにプログラムを作成する場合にはその中から適当なものを取り出して活用することが考えられる。

前もって用意蓄積しておいた多くのソフトウエアの部品を利用して、プログラミングに対するプログラマの負担を軽減させ、より効率的なプログラミング環境を実現しようとする研究は従来から行われており[1][2]、本研究室においても以前にこのような考えに基づいたエディタ（SCORE）を開発した。

SCOREは典型的なプログラム例をプログラムスキーマとしてあらかじめデータベースに蓄積しておき、通常のテキスト編集機能に加えてプログラムスキーマもその編集の対象に加えたエディタである。そこでは、プログラムスキーマをデータ構造に基づいた木構造に体系づけて分類し、蓄積していた。これはSCOREが編集の対象とするプログラムの言語としてPascalを前提としており、データ構造とアルゴリズムの関係を重視する抽象データ型の考えに着目したためである。

本稿ではSCOREの経験をもとにエディタ機能とユーザインタフェイスの向上を図った、プログラムスキーマに基づくエディタSCOREⅡ（Schema ORiented Editor Ⅱ）のプログラム編集環境について報告する。

2. SCOREⅡにおける編集環境

図1にSCOREⅡのシステム構成を示す。プログラムスキーマはVSAMファイル内に蓄積し、プログラマはプログラミングの過程においてこの中から随時一つのプログラムスキーマを取り出して、現在編集中であるプログラムの作成に応用する。

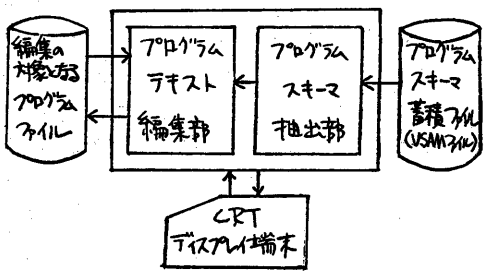


図1. SCOREⅡのシステム構成

SCOREⅡはPascalで記述されたプログラムを編集の対象とする。したがってプログラムスキーマもPascalで記述されている。

2.1 プログラムスキーマの形態

プログラムスキーマは、あるアルゴリズムを記述する一つの手続き（または関数）の宣言と、基本とするデータ構造の型定義から構成する。手続きの宣言中では、この型を参照することを許している。

図2にプログラムスキーマの一例を示す。Pascalの通常の手続き宣言とは異なり、プログラムスキーマは特定のプログラムに依存しないで、あるアルゴリズムを記述しようとするものなので、下線部のように具体化されていない未定部分が存在する。すなわち図2の例では、線形リストに挿入しようとする要素の型が決定していない。そこで、プログラマ自身がこのような未定部分を具体化し、編集の対象となるプ

ログラムにプログラムスキーマを取り込む。

プログラムスキーマ外部との変数値の受渡しは全て引数を介して行うものとし、これによりプログラムスキーマ内における未定義な大域的変数の参照を回避する。

```
TYPE
 TP=@T;
 T=RECORD
     KEY:<<1>>;
     NEXT:TP
     END;
 PROCEDURE PUSH(X:<<1>>; VAR P:TP);
 VAR
  Q:TP;
 BEGIN
  NEW(Q);
  WITH Q@ DO BEGIN
   KEY:=X;
   NEXT:=P
  END;
  P:=Q
 END;
```

図2. プログラムスキーマ例

2.2 プログラムスキーマを利用した編集作業

SCORE IIはCRTディスプレイを介して利用する。図3のように、CRT画面の左半面に編集の対象となるプログラムを表示し、右半面はプログラムスキーマの表示に利用する。行番号1～21、22～42はプログラム上での絶対的な行番号ではなく、CRT画面上の編集位置を指示するための相対的な行番号である。24行目はSCORE IIからのメッセージ表示欄であり、23行目がプログラマによるコマンド入力欄である。22行目にはプログラマが入力した直前のコマンドを表示する。図3に、蓄積ファイルから取り出したプログラムスキーマを編集中のプログラムに挿入するまでの作業過程を示す。

取り出したプログラムスキーマ（図3(a)）のアルゴリズムや使われている変数についての説明文を、プログラマはコマンド<WHAT>を用いて表示させ、プログラムスキーマについての理解の手助けを受けることができる。図3(b)はプログラムスキーマの処理内容についての説明文をSCORE IIが提示したところである。プログラムスキーマに対する理解を得たうえで、プログラマはまだ具体化されていない未定部分についての補完を、コマンド<1 CHAR>により行う（図3(c)）。

プログラムスキーマを編集対象のプログラム中に挿入するために、コマンド<ISC 1 RENAME PUSH, PUSHKEY REPLACE X,CH>を投入すると、左画面は図3(d)のように更新され

```
 1 PROGRAM SAMPLE(INPUT,OUTPUT);          22 TYPE
 2 VAR                                    23  TP=@T;
 3  CH:CHAR;                              24  T=RECORD
 4 BEGIN                                  25      KEY:<<1>>;
 5  READLN(CH);                           26      NEXT:TP
 6  WHILE CH<>'@' DO                      27      END;
 7   READLN(CH)                           28  PROCEDURE PUSH(X:<<1>>; VAR P:TP);
 8 END.                                   29  VAR
 9                                        30   Q:TP;
10                                        31  BEGIN
11                                        32   NEW(Q);
12                                        33   WITH Q@ DO BEGIN
13                                        34    KEY:=X;
14                                        35    NEXT:=P
15                                        36   END;
16                                        37   P:=Q
17                                        38  END;
18                                        39
19                                        40
20                                        41
21                                        42

PLEASE ENTER COMMAND
```

(a)

```
 1 PROGRAM SAMPLE(INPUT,OUTPUT);          22 A NEW ELEMENT IS ALLOCATED,AND INSER
 2 VAR                                    23 TED AT THE TOP OF THE LINEAR-LIST PO
 3  CH:CHAR;                              24 INTED BY THE POINTER 'P'. WHEN A NEW
 4 BEGIN                                  25  ELEMENT IS ALLOCATED,THE POINTER IS
 5  READLN(CH);                           26  BUILT IN THE ELEMENT,AND THE POINTE
 6  WHILE CH<>'@' DO                      27 R IS SWITCH TO THE NEW ELEMENT.
 7   READLN(CH)                           28
 8 END.                                   29
 9                                        30
10                                        31
11                                        32
12                                        33
13                                        34
14                                        35
15                                        36
16                                        37
17                                        38
18                                        39
19                                        40
20                                        41
21                                        42
WHAT
PLEASE ENTER COMMAND
```

(b)

図3. プログラムスキーマの挿入(I)

る。コマンド中の"1"は、プログラムスキーマを挿入する場所を指定するオペランドである。また"RENAME PUSH.PUSHKEY"および"REPLACE X.CH"はオプションコマンドで、前者はプログラムスキーマ中の識別子名PUSHをPUSHKEYに改名するものであり、後者は仮引数Xの宣言部を削除しプログラムスキーマ中のXの参照個所を大域的変数CHに置きかえるものである。

```
 1 PROGRAM SAMPLE(INPUT,OUTPUT);
 2 VAR
 3  CH:CHAR;
 4 BEGIN
 5  READLN(CH);
 6  WHILE CH<>'@' DO
 7   READLN(CH)
 8 END.
 9
10
11
12
13
14
15
16
17
18
19
20
21
 1  CHAR
```

```
22 TYPE
23  TP=@T;
24  T=RECORD
25      KEY:CHAR;
26      NEXT:TP
27      END;
28  PROCEDURE PUSH(X:CHAR; VAR P:TP);
29  VAR
30   Q:TP;
31  BEGIN
32   NEW(Q);
33   WITH Q@ DO BEGIN
34    KEY:=X;
35    NEXT:=P
36   END;
37   P:=Q
38  END;
39
40
41
42
```

```
PLEASE ENTER COMMAND
```

(C)

```
 1 PROGRAM SAMPLE(INPUT,OUTPUT);
 2 TYPE
 3  TP=@T;
 4  T=RECORD
 5      KEY:CHAR;
 6      NEXT:TP
 7      END;
 8 VAR
 9  CH:CHAR;
10  PROCEDURE PUSHKEY(VAR P:TP);
11  VAR
12   Q:TP;
13  BEGIN
14   NEW(Q);
15   WITH Q@ DO BEGIN
16    KEY:=CH;
17    NEXT:=P
18   END;
19   P:=Q
20  END;
21
ISC  1  RENAME PUSH,PUSHKEY  REPLACE X,CH
```

```
22 TYPE
23  TP=@T;
24  T=RECORD
25      KEY:CHAR;
26      NEXT:TP
27      END;
28  PROCEDURE PUSH(X:CHAR; VAR P:TP);
29  VAR
30   Q:TP;
31  BEGIN
32   NEW(Q);
33   WITH Q@ DO BEGIN
34    KEY:=X;
35    NEXT:=P
36   END;
37   P:=Q
38  END;
39
40
41
42
```

```
PLEASE ENTER COMMAND
```

(D)

図3. プログラムスキーマの挿入(II)

このように、プログラムスキーマの記述をある程度プログラマ自身のプログラムスタイルに合うように修正するための、簡素な編集機能を有している。

通常のプログラムテキスト編集コマンド以外の、プログラムスキーマの編集に関するコマンドを表1に示す。

SCORE IIは構造化エディタの機能を有し、Pascalの構文規則に従ってテキスト編集を行う。編集は文、手続きまたはプログラムスキーマ単位で行い、プログラマによるテキストの追加の際には常に構文上の部分解析を実行した後で挿入がなされる。

S	プログラムスキーマの抽出
メニュー番号	メニュー選択
キーワード	キーワードの再認識
WHAT	誤明文提示要求
RT	木構造の後戻り
未完部分番号	未完部分の補完
ISC	プログラムスキーマの挿入

表1. プログラムスキーマの編集に関するコマンド

3. プログラムスキーマの蓄積と検索

SCORE IIが提供するプログラム編集環境が有効なものであるためには、種々のプログラム例をどのくらい保持しているか、また蓄積している多量のプログラムスキーマの中から最も適切なものをいかに簡便に取り出すかということが、大きな課題となる。

3.1 プログラムスキーマの蓄積

あるデータ構造を決めると、それをもとにしたいくつかのアルゴリズムを考えることができる。そこでまずデータ構造について詳細化あるいは分類し、これを木構造に体系づける。そこでは上位に基本的なデータ構造の抽象的な分類項目を

記述し、下位に向って具体的なデータ構造を得るべく木構造に展開される。

プログラムスキーマは、具体的なデータ構造が決定した節の子孫に位置づけ、木構造の葉として保持する。

図4は、基本的なデータ構造として線形リストを仮定した場合に展開される木構造の例である。

3.2 キーワードによるプログラムスキーマの検索

プログラムスキーマを蓄積ファイル中から取り出すために、SCOREⅡは2通りのアプローチを用意した。

プログラマが、取り出したいプログラムスキーマについて、使用頻度が高いなどの理由でその名前が既知のときは、プログラムスキーマの名前を直接指定して取り出すことができる。

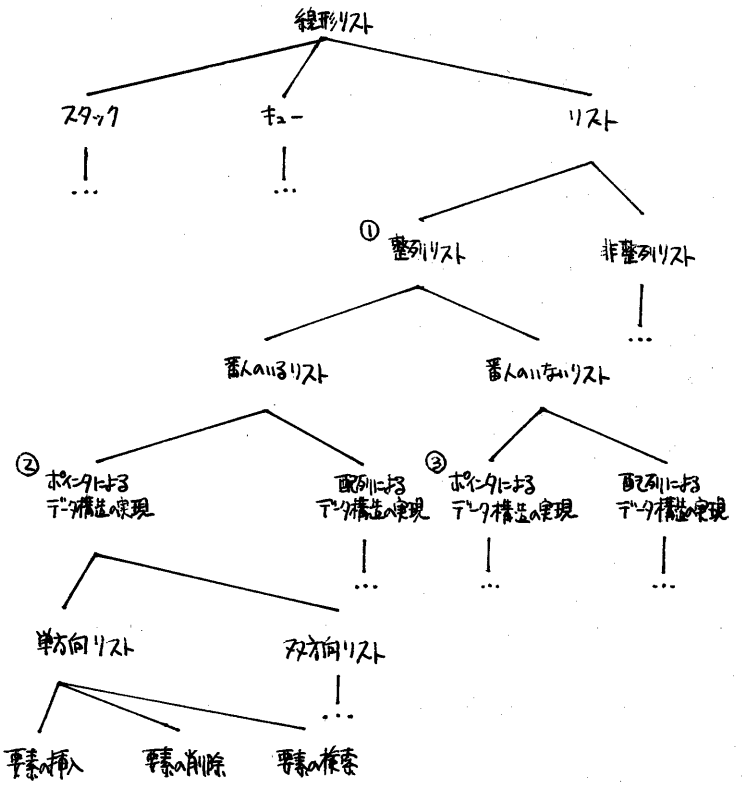


図4. 線形リストを基本データ構造とする木構造の例

別の方法としては、プログラマが必要としているプログラムスキーマについて、そのデータ構造あるいはアルゴリズムを言い表わすようなキーワード(英単語)をコマンドにより投入し、SCOREⅡが提示する候補者メニュの中から適当なものをプログラマが選択する、というものである。

プログラマが上述の後者の方法を用いて、必要なプログラムスキーマを取り出すまでの過程を例により説明する。

図5は線形リストを基本データ型として用い、新しい要素を整列された線形リスト中に挿入するプログラムスキーマを検索し、取り出した例である。

まずコマンド<S SORTED LINEAR LIST POINTER >を入力すると、木構造中の節、葉の名前、すなわち分類項目やプログラムスキーマの名前からなるメニューが提示される。このとき"SORTED"、"LINEAR"、"LIST"、"POINTER"はキーワードであり、これらのキーワード列が言い表わしていると思われる木構造中の最も適切な分類項目あるいはプログラムスキーマをSCOREⅡが選択し、図5(a)のように提示する。ここでは図4の木構造中の分類項目①,②,③が、提示されたのである。そこで、プログラマがメニュー中の2番を選択すると、SCOREⅡは'番人のいるリスト'それとも'番人のいないリスト'のいずれを欲しているのかという問いかけを、プログラマに対して発する。(図5(b))これは、最初にプログラマが入力したキーワード列の中に番人に関する情報

が欠けていたので、分類項目の②を選択した時点でその再確認を行い、不適当な木に入り込むことを防止しようとするものである。プログラマが'番人のいるリスト'の要求を入力すると、図5(c)に示される分類項目がプログラマの最も欲しているものであることが求まる。

このあと新たに提示されるその子孫の分類項目からなるメニューよりプログラマがさらに選択を続けると(図5(d),(e))、最後に具体的なプログラムスキーマ(図5(f))が得られる。

メニューからある項目を選択する際は、プログラマは必要に応じて各項目ごとにさらに詳しい説明文の提示を要求できる。

このようにプログラマはSCORE IIが提示するメニューを参考にして、木構造中を上位から下位あるいは逆方向にたどりながら必要なプログラムスキーマを探し出すわけである。

プログラマが投入したキーワード列とSCORE IIが提示したメニューから実際に選択した分類項目あるいはプログラムスキーマとが合致しなければ、SCORE IIが再確認を促すことで、プログラムスキーマの効果的な検索を図っている。これはプログラムスキーマの記述内容に対するプログラマからの問合せに対して、豊富な情報を用意しておくことと共にユーザインタフェイスの向上に寄与している。

4. SCORE IIの内部処理

SCORE IIはプログラムスキーマの蓄積にVSAMファイルを用いる。VSAMファイル内にはこれとは別に、プログラムスキーマの検索時に必要となるキーワード辞書と、キーワードとそれに関連の強い木構造中の節・葉の名前との対応関係を保持するインバーテッドリストを同時に用意する。

4.1 木構造の内部表現

個々のプログラムスキーマはVSAMファイル中の一つのレコードとして記述している。木構造の節に相当する分類項目も同様にVSAMファイルの一つのレコードに記述し、データ構造に基づく木構造をVSAMファイル内で構築した。

一つのレコードのキイの先頭文字は、プログラムスキーマと分類項目とを識別するための特殊文字であり、この文字を含めた10文字をプログラムスキーマあるいは分類項目のVSAMファイル中での名前として使用する。SCORE IIのコマンドにより、直接特定のプログラムスキーマや分類項目を取り

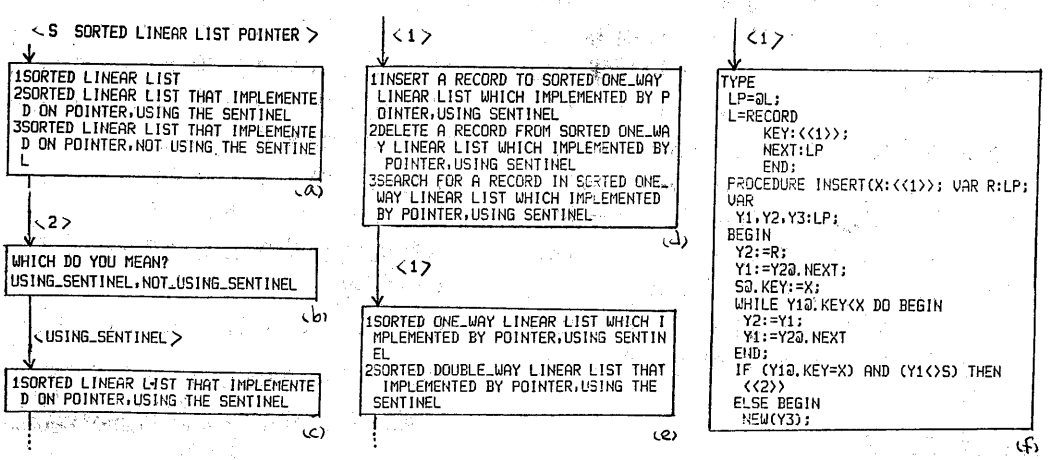


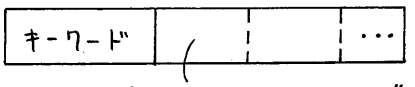
図5. プログラムスキーマの検索

出す場合は、この名前を用いて指名する。

プログラムスキーマを記述しているVSAMレコード内には、プログラムスキーマで用いられている変数やアルゴリズムなどについての説明文や、入力変数の初期化の要不要に関する情報などが合わせて格納されている。さらにプログラムスキーマと分類項目の内容を言い表わす複数個のキーワードを、個々のVSAMレコード内に格納している。

4.2 インバーテットリスト

プログラマが投入したキーワードをもとにして、それに関連のあるプログラムスキーマと分類項目を検索する際に、インバーテットリストを利用する。



プログラムスキーマまたは分類項目の名前

図6. インバーテットリスト中のレコード例

インバーテットリストの中の一つのレコードは図6のようにキーワードをVSAMのキイとして、そのフィールドにキーワードと関連の強いプログラムスキーマまたは分類項目の名前を登録する。

4.3 キーワード辞書

プログラムスキーマと分類項目を記述するVSAMレコード内にはそれらの内容を表現するのに適切なキーワードが格納されているが、キーワードと同義語である語をあらかじめキーワード辞書に登録しておく。たとえば「双方向リスト」という意味をもつ「doubly-linked-list」、「two-way-list」、「double-way-list」などのような同義語を同一キーワードとしてSCORE Iが把握できるようにした。これにより、プログラマが入力したキーワードに対する意図と、SCORE IIによるキーワードの解釈との整合を図っている。

5. おわりに

経験あるプログラマは実際のプログラミングや教育によって、さまざまな問題解決のためのパターンを抽象化・階層化して貯えており[4]、その有用な知識を常に適切に活用している。

SCORE IIではプログラミングにおける知識として、プログラムスキーマを木構造に体系づけた上で蓄積した。さらにユーザインタフェイスを改善して、プログミング知識の具体的な実体であるプログラムスキーマの抽出を簡便なものとし、編集作業の効率化を図った。

しかしプログラムスキーマの検索において適当な選択を行なう行為はあくまでプログラマが遂行するものであり、また取り出したプログラムスキーマを適切な利用についてもプログラマの能力に委ねられ、SCORE IIの管理外である。

SCORE IIでは、プログラマの判断に対する手助けとなるような情報を適宜提示するにとどまっているが、さらにプログラミングに関する意味情報を蓄積・管理して、強力なプログラミング環境をいかに構築するかについては今後の課題である。

また現在はプログラムスキーマを、データ構造についての分類を行った上で蓄積しているが、アルゴリズムの類似性をもとにした分類を行うことにより、データ構造の見地からはカテゴリに分けにくい数値計算などのためのプログラムスキーマの蓄積も容易になるのではないかと思われる。

さらに、プログラマ個人によるプログラムスキーマの蓄積ファイルへの登録機能を追加することを検討する。

参考文献

1) Waters R C: IEEE S.E. Vol. SE-8 NO.1 1982

2) 会田一夫・増豊・帳俊之：ソフトウエア工学研究会資料 33-3 11月 1983

3) 北山泰英・宮本衛市：第26回情報処理全国大会講演論文集 1983

4) 日本電子工業振興協会：ソフトウエアエンジニアリングに関する調査 3月 1983