プログラム構造に基づいた編集機能をもつテキスト・エディタ†

宮本衛市†† 浅見可津志†††

従来のテキスト・エディタは，文字列の処理を中心としているが，PASCAL のような構造を持つ言語の編集に際しては，各種のプログラミング方法論を支援することからも，構造自体を扱える機能が必要となる．そこで筆者らは PASCAL による構造化プログラミングの支援を目指すテキスト・エディタ POESY (PASCAL Oriented Editing System) の開発を行った．

以下に POESY の主な特長を示す．

（1） 編集の単位を手続きおよび文としたことにより，プログラムの構造に基づく処理が可能である．

（2） 段階的詳細化法によるプログラミングを支援するため，詳細化機能を用意している．

（3） 構文チェック，Pretty Printing，構造に基づく表示，識別子名の入力時省略機能などの諸機能を有している．

POESY によるプログラムの作成は，まず手続きを個別に作成し，次にそれをプログラムの階層構造に組み込んでいくという手順をとる．手続きを単位とする編集処理としては，挿入，置換，削除および転送が可能である．また文を単位とする編集処理としては，挿入，削除および複製のほかに，構造文の変換および生成の処理が可能である．これらの編集処理は，プログラム・テキストを一旦リスト構造に基づく内部表現に変換した上で行う．

1. まえがき

TSS の普及により，プログラムの作成あるいは修正を端末からエディタの支援のもとで行うことが一般化してきている．このエディタは，その編集対象と編集内容によって，次の2つに大別することができる．

（1） 汎用的なエディタで，編集対象は任意の文字列であり，したがってプログラム，データ，英文などを文字列の集まりと見なして扱うものである．

（2） 特定のプログラミング言語の構文規則を取り込んだエディタであって，文字列処理のほかに BASIC, LISP, FORTRAN, COBOL などで書かれたプログラムの構文チェックもあわせて行うものである．

一方，構造化プログラミング[1] の手法がシステムソフトウェア開発にまで浸透しており[2]，構造化プログラミング言語によってプログラムを作成する際の強力な支援体制が望まれる．この場合，単なる文字列を編集対象とするのではなく，プログラムあるいはデータ構造をも取り込んだ編集支援機能が必須である．

そこで筆者らは PASCAL による構造化プログラミングの支援を目指すテキスト・エディタ POESY (PASCAL Oriented Editing System) の開発を行った．POESY では PASCAL で書かれたプログラムテキストを入力とするが，内部的には抽象的な形式に変換して編集の対象としている．したがって，構文解析部を置き換えることにより，他言語のためのエディタに変更することも可能である．

プログラムの構造を編集対象とするテキスト・エディタとしては，IRIA の MENTOR[3),4)] がある．MENTOR ではプログラムテキストを構文規則に基づいて，プログラムから識別子および記号までにいたる階層構造を持つ抽象形式 (abstract representation) に展開して把握する．したがって，プログラマも編集に際しては構文規則を念頭に置く必要がある．

これに対し，POESY ではプログラムの階層構造を構成するものとして手続き* および文** に着目する．構造把握のため手続きと文に着目したのは，これらがプログラミング上での概念をつくる単位であり，文をさらに構文解析し細分化してもプログラミング上での支援はあまりないものと考えたからである．

すなわち，プログラムを構成するものは手続きの階層構造的な集まりであり，さらに手続きは文の階層構造的な集まりであると考える．したがって，POESY では手続き単位および文単位での構造的編集支援に重

---

† A Text Editor Having Editing Facilities Based on Program Structure by EIICHI MIYAMOTO (Division of Information Engineering, Faculty of Engineering, Hokkaido University) and KAZUSHI ASAMI (Computer Works, Mitsubishi Electric Corp.).

†† 北海道大学工学部情報工学専攻

††† 三菱電機(株)計算機製作所

* メインプログラムおよび関数も含めて，編集上手続きと呼ぶことにする．

** 実行文 (statement) のほか，宣言部における各識別子の定義，宣言も含めて，編集上文と呼ぶことにする．

点を置いている.

POESY の主な特長を次に示す.

(1) 編集の単位を手続きおよび文としたことによって, プログラムの構造に基づく処理が可能である.

(2) 段階的詳細化法 (stepwise refinement)[5] によるプログラミングを支援するため, 詳細化機能を用意している.

(3) 構文チェックはもちろん, プログラムの文書化能力を高めるため pretty printing を行う.

以下, 2章では POESY の編集機能とコマンドについて, 3章では POESY の内部処理の概要を述べ, 最後に使用例を掲げる.

2. 編集機能とコマンド

2.1 編集環境

編集は会話型処理で行うが, POESY 内でのテキストファイルの流れを図1に示す. プログラムを修正する場合には, まず入力ファイルからプログラムテキストを読み出し, 手続き格納ファイルへ手続きを個別に格納する一方, 手続き間の関係をプログラム構造表に登録する.

新たにプログラムを作成する場合は, 端末からの入力により手続き編集バッファで各手続きを作成する. 修正の場合は, 対象となる手続きを手続き格納ファイルから手続き編集バッファに移して行う. 作成あるいは修正が終了すると, 処理ずみの手続きを手続き編集バッファから手続き格納ファイルに転送する.

プログラムの編集が終了すると, 手続き格納ファイルの内容を完全な PASCAL プログラムに再構成して出力ファイルに書き出す.

2.2 コマンドの形式

テキスト・エディタのコマンドは, 憶えやすくしかも使いやすくなければならない[6]. そのため POESY ではコマンドの設計にあたって, 簡潔化と体系化を図った. コマンドの形式を以下に示す.

〈複コマンド〉::=〈コマンド〉|〈複コマンド〉
;〈コマンド〉

〈コマンド〉::=〈繰り返し〉〈操作〉〈引数〉

〈繰り返し〉はコマンドの繰り返し回数を指定する2桁までの正の整数, 〈操作〉はコマンドの種類を示し, 英文2文字からなる. 〈引数〉は〈操作〉にとも

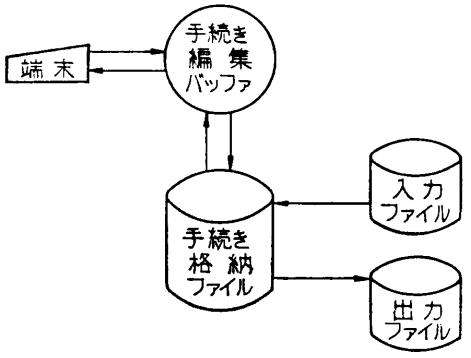

図1 POESY におけるテキストデータの流れ
Fig. 1 Flow diagram of text data in POESY.

表1 コマンド一覧表
Table 1 The table of commands.

分類		コマンド	機能
手続きを編集単位とする		IP	作成した手続きをプログラムに登録する.
		EP	手続きを削除する.
		RP	手続きを編集バッファに読み込む.
		SP	修正した手続きをもとの手続きと取換える.
		TP	手続きを転送する.
文を編集単位とする	ポインタ移動	MT	ポインタをテキストのはじめに移動する.
		ME	ポインタをテキストのおわりに移動する.
		MP	ポインタを特定の場所に移動する.
		MB	ポインタを直前の位置に戻す.
		MF	ポインタを前向きに移動する.
		MR	ポインタを後向きに移動する.
		MU	ポインタを入れ子に対して上向きに移動する.
		MD	ポインタを入れ子に対して下向きに移動する.
		MS	ポインタの指示範囲を拡張する.
		FW	指定の文字列を持つ文にポインタを移動する.
		CP	指示ポインタに値を与える.
	編集処理	ES	文を削除する.
		CS	文を複製する.
		TF	文章を指示ポインタの指す文の前へ転送する.
		TT	文章を指示ポインタの指す文の後へ転送する.
		SW	文字列を置き換える.
		GS	構造文を生成する.
		AS	構造文の型を変換する.
	挿入処理	HE	手続きの見出しの宣言.
		LA	ラベルの宣言.
		CO	定数識別子の定義.
		TY	タイプ識別子の定義.
		VA	変数識別子の宣言.
		IF	文章をポインタの指す文の前に挿入する.
		IT	文章をポインタの指す文の後に挿入する.
表示機能		DA	作成中の手続きをすべて表示する.
		DS	ポインタの指示する文章を表示する.
		DE	ポインタの指示する文の型と数式を表示する.
		DC	プログラムの構成を表示する.
		DB	一時バッファの内容を表示する.
		DI	省略された識別子名を表示する.
		ID	変数の情報を表示する.
その他		AI	識別子名の省略を指示する.
		AR	詳細化機能で, inline 展開する文章を与える.
		PA	構文解析を指示する.
		" "," "	一時バッファの内容を変更する.
		RE	プログラムテキストを読み込む.
		WR	プログラムテキストを書き出す.
		BY	POESY の終了を指示する.

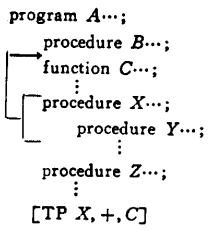

図 2 内包する手続き Y を含めた手続き X の，関数 C の直前への転送例
Fig. 2 An example of the transfer of procedure X together with included procedure Y just before function C.

なう引数であり，複数の場合は“,”で区切る．

コマンドの一覧表を**表 1** に示す．実際の編集処理を行うコマンドは，手続きを単位とするものと文を単位とするものに分類することができる．

2.3 手続き単位での編集

POESY では，手続きを個別に作成あるいは修正した後，階層的構造のもとに手続きを組み込んでいくことによってプログラムを構成していく．そのため，手続きの挿入 (IP)，置換 (SP)，削除 (EP) および転送 (TP) を行う各コマンドを有している．**図 2** は転送コマンド [TP X, +, C] により手続き X を関数 C の直前（+の指示により）に転送する例を示したものである．この場合，手続き X が包含する手続き Y も含めて転送の対象とする．

2.4 文単位での編集

各手続きは，宣言部（手続き・関数の見出し，ラベルの宣言と，定数，タイプおよび変数の各識別子の宣言あるいは定義）と実行部 (statement part) に分けて考える．このうち，宣言部の作成は各宣言あるいは定義に対して個別に用意した作成コマンド (HE, LA, CO, TY, VA) により随時行うことができる．一方，実行部は文あるいは文章（並記された文）を作成単位として，文挿入コマンド (IF, IT) により作成していく．

作成あるいは編集される文の位置を明示するため，文を指示するポインタを用いる．ポインタには主ポインタと，これから派生する拡張ポインタと指示ポインタがある．拡張ポインタは編集対象を文から文章に拡張する場合に用いる．指示ポインタは指示された文または文章を転送する際に，その移動先を指示するものである．**図 3** は，あらかじめ指示ポインタの設定 (CP) と拡張ポインタの設定 (MS) を行っておき，文章の転送コマンド (TT) による文章転送の様子を示したものである．

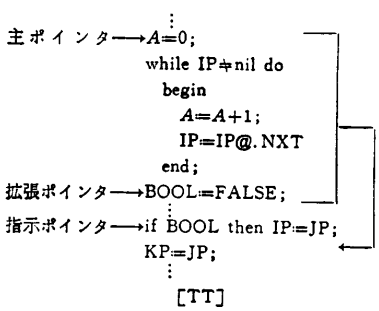

図 3 主ポインタおよび拡張ポインタで指示する文章の，指示ポインタの直後への転送例
Fig. 3 An example of the transfer of the text segment indicated by main pointer and extension pointer just behind indication pointer.

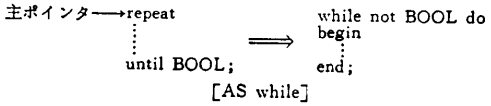

図 4 **repeat** 文から **while** 文への変換例
Fig. 4 An example of the conversion of a **repeat** statement into a **while** statement.

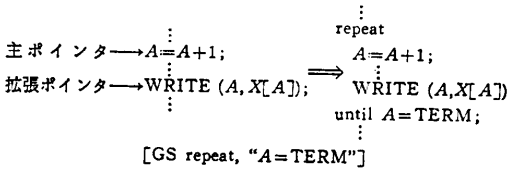

図 5 ポインタおよび拡張ポインタで指示する文章から **repeat** 文の生成例
Fig. 5 An example of the generation of a **repeat** statement from the text segment indicated by main pointer and extension pointer.

POESY には，上述の転送 (TF, TT) のほかに削除 (ES) および複製 (CS) の処理を行うコマンドがある．一方，構造文を処理するものとして，

（1） 構造文の変換 (AS)：構造文を他の型の構造文に変換する．

（2） 構造文の生成 (GS)：構造文の型と付随する数式を与えて構造文を生成する．

の 2 種のコマンドを備えている．**図 4**，**図 5** にその例を示す．

2.5 その他の機能

（1） 詳細化機能　段階的詳細化法 (stepwise refinement) によるプログラミングを支援するため，テキストに一時的に挿入しておいた〈.と.〉で囲まれた注釈文に対し，後で与える文章で inline 展開するものである．この場合，inline 展開する文章に再び注釈文が含まれていてもかまわない．**図 6** に，注釈文

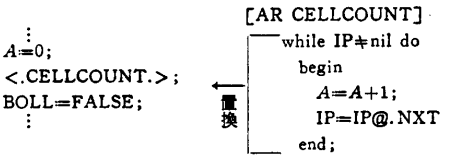

図 6 詳細化プログラムテキストによる注釈文の inline 展開例

Fig. 6 An example of the inline expansion of a comment statement by a refined text segment.

PROCEDUREDECLARATION ⇒ P!
[AI PROCEDUREDECLARATION, P]

図 7 識別子の省略名の定義例

Fig. 7 An example of the definition of an abbreviation of an identifier.

〈.CELLCOUNT.〉に対して与えた **while** 文が inline 展開される様子を示す.

(2) 構文チェック機能　PASCAL の parser を内蔵しており，挿入されるテキストの構文チェックを行っている.

(3) pretty printing 機能　プログラムテキストを，そのプログラム構造にあわせて適切な段落付け (indentation) をすることは，プログラムの読みやすさ (readability) を高めるために不可欠である．そこで POESY では，手続きおよび文の表示 (DC, DS) あるいは手続き格納ファイルおよび出力ファイルへの格納に際して，自動的に pretty printing を施こす. **(図 12 参照)**

(4) 識別子名の入力時省略機能　プログラムの文書化能力を高めるために，識別子名は自身の役割を適切に表現するように名付けられなければならないが，そのために識別子名は一般に長くなる傾向にある．テキスト入力時に長い識別子名をたびたび打ち込むのは繁雑であるので，POESY では識別子名の省略形を指示することによって，以後その省略形が打ち込まれた場合，自動的にもとの識別子名に変換する機能を持っている．**(図 7 参照)**

(5) プログラムテキストの表示機能　作成あるいは修正中のプログラムテキストの一部または全部を適宜，構造的に表示する機能を有している．(図 12 参照)

3. 内部処理の概要

3.1 機能的構成

POESY の機能的構成の概要を**図 8** に示す．端末からの入力がコマンドの場合は，コマンド解析部がコマンドの解析を行い，コマンドが手続き単位に関するものであればプログラム構成部へ，文単位に関するものであれば手続き編集部へ必要な制御情報を転送する．端末からの入力がテキストの場合には，テキストを一時バッファに格納する．入力終了後，読み込んだテキストの構文チェックを行い，テキストに誤りがない時は内部表現に変換し，手続き編集バッファに挿入する．構文エラーがある時は端末にエラーの種別を表示して修正を促す．一時バッファの内容は文字列単位の修正が可能であるが，通常の編集処理は手続き編集バッファの内部表現を対象として行う.

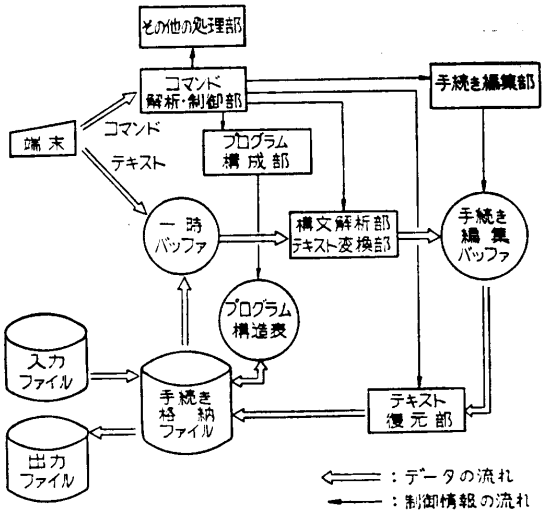

図 8 POESY の構成の概要と，データおよび制御情報の流れ

Fig. 8 The outline of the composion of POESY, and the flow of text data and control information.

手続きの編集が終了すると，手続き編集バッファの内部表現をテキストに再構成して手続き格納ファイルに格納する．また同時に，手続き格納ファイルに付随するプログラム構造表に，手続きの構造に基づく登録を行う．プログラム構造表は，入力ファイルを手続き格納ファイルに読み込む時に作成するが，手続き格納ファイルの手続き群を完全な PASCAL プログラムに再構成して出力ファイルへ書き出すときは，このプログラム構造表に基づいて行う.

3.2 手続き編集バッファ

手続き編集バッファ内では，**図 9** に示すように，テキストをリスト構造に基づく内部表現のもとで扱っている．リスト構造の各要素は文に対応しており，要素には，文の型，文の持つ数式，文の並びあるいは文の

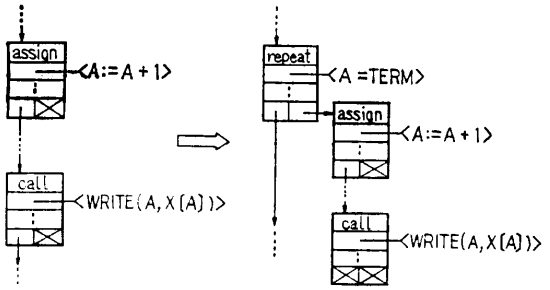

図 9 構造文生成によるシステム内でのテキスト表現の変化例
Fig. 9 An example of the change of internal text representation by the generation of a structured statement.

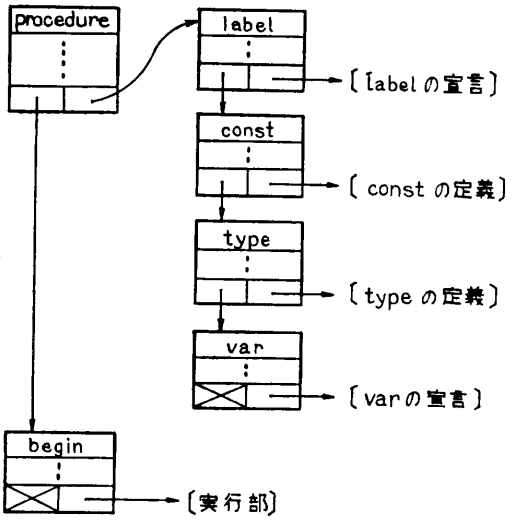

図 10 手続き編集バッファ内にあらかじめ設定される枠組み
Fig. 10 The frame set up in the procedure editing buffer in advance.

入れ子を表現するポインタなどの情報を格納する.

手続き編集バッファには，あらかじめ**図10**に示すような枠組みを用意してあり，挿入されたテキストをこの枠組みに付け加えることにより手続きを作成する．編集処理は，この内部表現を変更することによって行う．例えば，図5に示したように，新たに **repeat** 構造を生成した場合には，内部表現を図9のように変更する．また，文の転送，削除，挿入などはポインタを変更するだけで実現することができる.

3.3 プログラム構造表

プログラム構造表は，手続き格納バッファに個別に格納している手続き間の関係を，リスト構造で保存している．リスト構造の各要素には，手続きの型 (program, procedure, function の区別)，手続きの識別子名，手続きの並びおよび入れ子を表現するポインタなどの情報を格納している．手続きの転送，削除および挿入は，このプログラム構造表を変更するだけで実現することができる．**図11**に図2に示した手続きの転送時におけるプログラム構造表の変化の様子を示す.

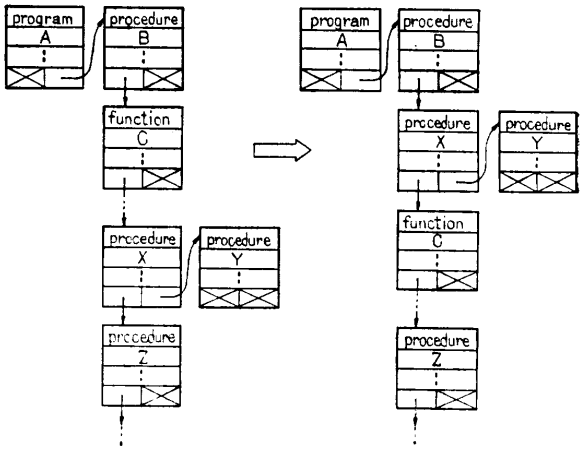

図 11 手続きの転送によるプログラム構造表の変化例
Fig. 11 An example of the change of the program structure table by the transfer of procedures.

4. 使 用 例

POESY の使用例を**図12**に示す．例では，最初にメインプログラム BUILDTREE を作成，登録した後，手続き PRINTTREE を作成し，メインプログラムに対して内側の入れ子になるように登録している．なお，？はコマンドの，＞はテキストの入力を促す，POESY からの促進記号である.

5. あ と が き

POESY は PASCAL プログラムの作成および修正を支援するのが目的であるが，プログラムの構造に基づく強力な編集能力を発揮することができる．また，構造化プログラミング，モジュール化プログラミング，段階的詳細化法などのプログラミング方法論を編集時に支援することが可能である.

最近，言語プロセッサおよびプログラムのデバッグ，テスト，評価などを支援するソフトウェアツールを一体化し，ソフトウェア作成支援システムを構成しようという試みがあるが[7]，テキスト・エディタは，このようなシステムにおいて，今までのように単なるプログラムの作成および修正の道具としてではなく，システムの中核として作置づけ，プログラマとソフト

```
PASCAL EDITING SYSTEM
NEW OR OLD
?NEW
COMMAND MAY BE ENTERED NOW
?HE
>PROGRAM BUILDTREE(INPUT,OUTPUT);"
INPUT TEXT IS CORRECT
?TY
>REF=@NODE;
>NODE=RECORD KEY:INTEGER; LEFT,RIGHT:REF END;"
INPUT TEXT IS CORRECT
?IT
>READ(N); ROOT := TREE(N); PRINTTREE(ROOT,0);"
INPUT TEXT IS CORRECT
?VA
>N:INTEGER; ROOT:REF;"
INPUT TEXT IS CORRECT
?DA
 PROGRAM BUILDTREE ( INPUT , OUTPUT ) ;
   TYPE
     REF = @ NODE ;
     NODE = RECORD
              KEY : INTEGER ;
              LEFT , RIGHT : REF
            END;
   VAR
     N : INTEGER ;
     ROOT : REF ;
 BEGIN
   READ ( N ) ;
   ROOT := TREE ( N ) ;
   PRINTTREE ( ROOT , 0 )
 END .
?IP
?HE
>PROCEDURE PRINTTREE(T:REF; H:INTEGER);"
INPUT TEXT IS CORRECT
?VA
>I:INTEGER;"
INPUT TEXT IS CORRECT
?IF
>IF T<.NIL THEN WITH T@ DO BEGIN
>PRINTTREE(LEFT,H+1); FOR I := 1 TO H DO WRITE('   ');
>PRINTTREE(RIGHT,H+1) END;"
ERROR DISPLAY
IF T<.NIL THEN WITH T@ DO BEGIN
ERRCODE:340
ERRCODE:357
?"T<.NIL","T<>NIL"
?DB
 1:IF T<>NIL THEN WITH T@ DO BEGIN
 2:PRINTTREE(LEFT,H+1); FOR I := 1 TO H DO WRITE('   ');
 3:PRINTTREE(RIGHT,H+1) END;
?PA
INPUT TEXT IS CORRECT
?DE
 IF T<>NIL
?3MD;3MF
?DS
 PRINTTREE ( RIGHT , H + 1 )
?IF
>WRITELN(KEY);"
INPUT TEXT IS CORRECT
?DA
 PROCEDURE PRINTTREE ( T : REF ; H : INTEGER ) ;
   VAR
     I : INTEGER ;
 BEGIN
   IF T <> NIL THEN
     WITH T @ DO BEGIN
       PRINTTREE ( LEFT , H + 1 ) ;
       FOR I := 1 TO H DO
         WRITE ( '   ' ) ;
       WRITELN ( KEY ) ;
       PRINTTREE ( RIGHT , H + 1 )
     END
 END;
?IP <,BUILDTREE
?DC
 PROGRAM BUILDTREE
  PROCEDURE PRINTTREE
?WR
?BY
PASCAL EDITING SYSTEM IS TERMINATED
```

図 12 ROESY のもとでのテキスト編集例

Fig. 12 An example of text editing using POESY.

ウェアツールおよび言語プロセッサ間のインタフェースとしての役割を果すべきである．これは，筆者らの開発した POESY の目的でもある．現在，テキスト・エディタとしての POESY からプログラム作成支援システムとしての POESY とするために次のような拡張を行っている．

（1） データフロー解析ツールとの接続[8] 現在の POESY では，手続きの転送に際して問題となる変数の有効範囲（局所的変数，大域的変数）の扱いをプログラマに任せているが，データフロー解析を行うことによってこの問題を解決できる．また，編集に際して必要となる変数に関する各種の情報を与えることも可能となる．

（2） ソフトウェア作成ツールとの連携[9] ソフトウェアの自動作成は将来に向っての研究課題であるが，その目的を限定することによってプログラムの枠組み的なものは作成支援することができる．その際，テキスト・エディタは不完全な部分を補うのに用いることができる．

（3） エラー訂正機能[10] 構文チェックによりエラーを指摘するばかりでなく，テキストの自動訂正によって冗長なエラー情報の排除および無用なテキストの読み捨てを防ぐ．

なお，POESY は PASCAL で記述されており，その大きさはソースイメージで約 3,000 行である．現在，北海道大学大型計算機センターの FACOM 230-75 上で稼動している．

謝辞 北海道大学田川遼三郎教授および永田邦一教授には，本研究を進めるに当って貴重な御教示と御配慮をいただいた．ここに記して深謝申し上げます．

参 考 文 献

1) Dijkstra, E. W.: Note on Structured Programming, Structured Programming, Academic Press, pp, 1-82 (1972).
2) 松下，山崎，高橋，吉田，伊藤：ストラクチャードプログラミング導入と効果，ソフトウェア工学研究会資料 8-2 (1978).
3) Donzeau-Gouge, V., Huet, G., Kahn, G., Lang, B. and Lévy, JJ.: A Structure Oriented Program Editor: A First Step towards Computer Assisted Programming, Rapport de Recherche IRIA, No. 144 (1975).
4) 玉井，吉村：海外における最近のソフトウェア工学の動向，ソフトウェア工学研究会資

料 6-1 (1978).
5) Wirth, N.: Program Development by Stepwise Refinement, Commun. ACM, Vol. 14, No. 4, pp. 221-227 (1971).
6) Sneeringer, J.: User-Interface Design for Text Editing: A Case Study, Software-Practice and Experience, Vol. 8, pp. 543-557 (1978).
7) 佐々木, 竹沢, 山本: ソフトウェア一貫生産のツール, ソフトウェア工学シンポジウム, pp. 217-225 (1979).
8) 宮本, 開発: PASCALプログラムにおけるデータフローの解析, 第 18 回情報処理学会全国大会講演論文集, pp. 267-268 (1977).
9) 宮本, 浅見: コンパイラ作成支援システム, 第 19 回情報処理学会全国大会講演論文集, pp. 225-226 (1978).
10) Tai, K. C.: Syntactic Error Correction in Programming Languages, IEEE Trans. Software Eng., Vol. SE-4, No. 5, pp. 414-425 (1978).

(昭和 54 年 4 月 4 日受付)
(昭和 54 年 6 月 21 日採録)