対話型修正編集機能を持つ PASCAL 構文解析システム†

菅井賢三†† 岡根幸宏†††
荒木俊郎††† 都倉信樹†††

プログラミングの支援環境として，既存のコンパイラ，エディタ等の改善が要求されている．コンパイラに関しては，構文誤りからの回復，およびエラーメッセージが必ずしも適切であるとは言えず，ユーザが誤りの修正に多くの時間を費やすことがある．また，エディタによる修正操作は，コマンドが複雑で扱いにくいことが多い．

そこで，筆者らは PASCAL に慣れていないユーザを対象として，ユーザとシステムが，CRT ディスプレイを用いて，対話方式で簡略化された修正操作によって誤りの修正ができ，自動的に修正位置より前に戻って構文解析をやり直す機能を持ち，短時間で構文的に正しいプログラムが得られる，対話型修正編集機能を持つ PASCAL 構文解析システムを作成し，小型計算機 NOVA 3 上で稼働させた．

本システムでは，構文誤りの修正を，対話方式によりユーザにまかせ，修正位置以前から構文解析をやり直すことにより，構文誤りからの回復は必要でない．また，誤り検出時には，CRT 画面にエラーメッセージとともに，誤り検出箇所にカーソルを置き，ソースプログラムを表示するため，真の誤り箇所の発見および修正に費やす時間が短縮される．さらに，本システムを構文解析機能を持つエディタとして見れば，ソースプログラムを部分的に正しいか否かをチェックしたり修正しながら入力することができるので，PASCAL の初心者の構文学習にも用いられ，編集されたプログラムは，PASCAL 構文上誤りのないものとなる．

1. まえがき

コンパイラの重要な機能の一つとして，構文誤りの検出と回復，および誤りの修正等が要求されるが，必ずしも十分満足のできる方法は得られていない[1]．また，コンパイラのエラーメッセージは，適切なものばかりとは言えず，それが不適切な場合には，ユーザ，特に初心者にとっては真の誤り箇所の発見に手間取ることが多い．さらに，エディタを用いて修正を行う際，コマンドが複雑で扱いにくいことが多く，プログラムの修正に多くの時間を費やすことがある．

そこで，上記の時間を短縮するために，当研究室において小型計算機 NOVA 3 に移植した PASCALΣ* を対象にして，ユーザとシステムが CRT 端末を用いて対話方式で誤りの修正ができ，短時間で構文的に正しいソースプログラムが得られる“対話型修正編集機能を持つ PASCAL 構文解析システム PIPES**”を，NOVA 3 上に作成した．

このシステムは，PASCAL に慣れていないユーザを対象としており，ソースプログラムを部分的に正しいか否かをチェックしたり修正しながら入力することができる．そのため，PIPES を構文解析機能を持つエディタとして見れば，PASCAL の初心者の構文学習にも用いられ，編集されたプログラムは，PASCAL 構文上誤りのないものとなる．

PIPES のユーザから見た特徴として，

（1） 対話方式により誤りの修正を行うことができ，短時間で構文的に正しいソースプログラムが得られる．

（2） ユーザコマンドから複雑なパラメータ指定が取り除かれているため，ユーザは，ファンクションキーを1回押すだけで各コマンドが実行できる．

（3） 誤りが検出された際，エラーメッセージとともに，段付けされたソースプログラムが CRT 画面に表示され，誤りが検出された箇所にカーソルが置かれるので，真の誤り箇所の発見および修正が行いやすい．

ことなどがあげられる．また，システムとしての特徴には，

（1） 修正位置より以前に戻って構文解析をやり直す機能（これをあと戻り機能と呼ぶ）を持つため，誤りからの回復機能は必要ない．

---

† PASCAL Interactive Parsing and Editing System (PIPES) by KENZO SUGAI (Systems Engineering Group, Systems Engineering Department No. 2, FUJIFACOM CORPORATION), YUKIHIRO OKANE, TOSHIRO ARAKI and NOBUKI TOKURA (Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University).

†† 富士ファコム制御(株)第2システム部第1システム課

††† 大阪大学基礎工学部情報工学科

* 標準 PASCAL[2] から，実数やファイルタイプなどの機能を除いたもの[3]．

** PASCAL INTERACTIVE PARSING AND EDITING SYSTEM.

（2） ユーザによるプログラムの理解と修正を助けるため，CRT 画面表示時に，プログラムの構造を正しく，明確に表現する段付け機能 (indentation) を有する．

（3） プログラム構造が階層モジュール化されており，主要なデータ構造に対しては，抽象データ型の考えを導入しているので，将来の変更・拡張が容易である．

ことなどがある．

従来のコンパイラでは，ソースプログラムに構文誤りが検出された場合は，エラーメッセージを出して誤りから回復し，構文解析を進めてゆく．しかし，PIPES では，構文誤りが検出されれば構文解析を一時中断し，CRT 画面にエラーメッセージ，誤り検出箇所を含むソースプログラムの手続き名とともに，誤り検出箇所付近のソースプログラムを適度に段付けを施して表示し，誤り検出箇所にカーソルを置き，ユーザに修正を促す．そして，ユーザが対話方式で修正を行い，ユーザによる修正が終了した後，自動的に必要なだけあと戻りして構文解析を再開する．このため，不必要な部分の構文解析を再度行う必要はない．

本論文では，第2章でシステムの構成と処理の概略，第3章でデータ構造，第4章で主要な機能について述べる．

2. システムの構成と処理の概略

PIPES は，システムとユーザが対話しながら構文解析と誤りの修正を行う機能を持っている．**図1** に示すように，処理は大きく3段階に分けることができ，次のような手順で行われる．

（1） 第1段階（語彙解析）

ソースプログラムファイルを入力として，リスト作成部が，ソースプログラム中の構文単位であるトークンを順に両方向につないだトークンリストを作成する．(**3.1** 参照)

（2） 第2段階（構文解析，表示，修正）

（i） トークンリストを入力として，構文解析部が構文解析を行う．この際，現在解析中のソースプログラムの手続き名を CRT 画面に表示する．(**3.2** 参照)

（ii） 構文誤りが検出されると，対話処理部，CRT 入出力部を経て，CRT 画面上にエラーメッセージと誤り検出箇所付近のソースプログラムが段付けして表示される．(図1の①～④，**4.3** 参照)

（iii） ユーザは，**図2** のように CRT 画面に表示されたメッセージとプログラムを参照して，修正箇所が画面上にない場合は，位置コマンドを用いて修正箇所を画面上に表示させ，修正箇所にカーソルを移動させ，挿入，削除，置換コマンドを用いてプログラムの修正を行う．この際，カーソルが修正用ポインタである．(**4.2** 参照)

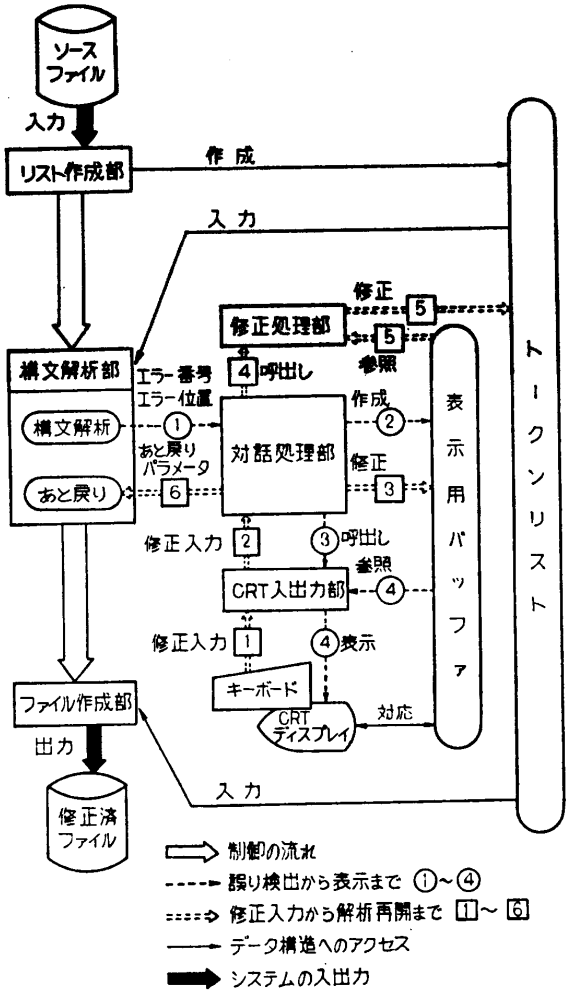

図 1 システム構成と処理の流れ
Fig. 1 Systems overview of PIPES.

（iv） 修正入力を受け取ると，対話処理部は修正処理部を呼出し，トークンリストを修正する．(図1の 1～5)

（v） (iii)(iv)が繰り返された後，ユーザの修正が終了すると，構文解析部は対話処理部より必要なパラメータを受け取り，そのパラメータにより決定される，修正位置以前の再開位置に戻り構文解析を再開する．(**4.1** 参照)

（3） 第3段階（修正済プログラム作成）

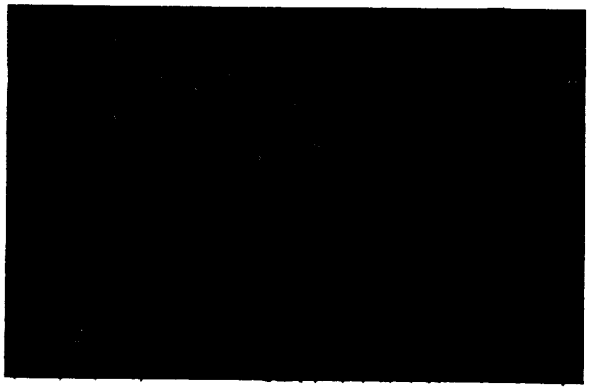

図 2 誤り検出時の CRT 画面
Fig. 2 An example of display.

構文解析の終了後，すなわち，構文誤りのないプログラムが得られると，修正されたトークンリストを入力として，ファイル作成部が修正済ソースファイルを作成する．

3. 主要なデータ構造

一般に，プログラムには信頼性や効率に加えて，理解のしやすさ，変更のしやすさ等が要求される．そこで，PIPES では主要なデータ構造に抽象データ型の考えを採用した．各々のデータへのアクセスは，対応する特徴づけオペレーションのみに限定し，データの安全とともに，変更・拡張に備えている．この章では，主要なデータ構造であるトークンリストおよび表示用バッファについて述べる．

3.1 トークンリスト

NOVA 3 は，入力テキストがシーケンシャルアクセスのみ可能なファイルとして与えられるため，ソースファイルをさかのぼって修正することは効率が悪い．したがって，PIPES ではプログラム内のトークンの系列を両方向リスト（トークンリスト）として作成し，このトークンリスト上で修正を行う．

トークンリストは，**図 3** に示すようなセルをつないだものであり，構文解析部だけでなく，CRT 画面への表示や修正済ファイルの作成などにも利用される．トークンリストの各セルには，トークンそのものが格納されるのではなく，トークンの種類が格納される．その際，トークンの種類が，名前，整定数*，ストリング，コメント**の場合，そのトークンの内容は次のように取り扱われる．

前のトークンへのポインタ
次のトークンへのポインタ
トークンの種類
呼出し番号（4.1 参照）
表示レベル（4.3 参照）
i）名前へのポインタ
ii）コメントへのポインタ
iii）手続きの先頭へのポインタ

・各セルは 6 ワードからなる．
・i）ii）iii）は可変部であり，この 3 フィールドのうち 1 つが選ばれる．

図 3 トークンリストのセルの構造
Fig. 3 Cell structure of token-list.

（1） 名前，整定数の場合

記号木 (symbol tree, 名前，整定数を格納する 2 分木）に登録し，その節点 (node) がトークンリストのポインタにより指される．この場合，記号木に登録するのは高々 8 文字であり，9 文字以上の場合は，9 文字目以降を 8 文字ずつ分けて，リストとして専用の領域に格納して記号木の節点につなぐ*．

（2） ストリング，コメントの場合

10 文字ずつ分けて，リストとして専用の領域に格納し，その最初の領域がトークンリストのポインタにより指される．

以上のようにして作成されるトークンリストの概略を**図 4** に示す．

3.2 表示用バッファ

PIPES は，CRT 画面と対応した二次元配列の表示用バッファを持っている．CRT 画面への表示およびキーボードからの入力はすべて，この表示用バッファを介して行われる．表示用バッファの構造を**図 5** に示す．

PIPES では，文字の置換や挿入などのコマンドは，カーソルの行移動により，その行の修正終了とみなしている．しかし，1 行 80 文字である CRT 画面の 80 文字目に入力を行うと，CRT の機能上，自動的に次の行にカーソルが移り，CRT 画面と表示用バッファとの対応がとりにくい．そこで，CRT 画面の 1 行を 79 文字に制限し，画面上でカーソルが 80 文字目に移動しないようにシステムが管理している．

図 5 における斜線部は，普通は利用されないが，文字の挿入コマンドによって 79 文字を越えた場合に，一時的に用いられる領域である．

* PASCAL Σ では，実数型はなく整数型のみである．
** 本来，コメントはトークンではないが，CRT 画面や修正済ファイル出力時にコメントも出力しなければならない．そのため，コメントもトークンとみなし，トークンリストにつないでいる．

* PASCAL Σ では，名前は 8 文字までが意味を持ち，9 文字目以降は読み飛ばされるが，CRT 画面への出力時には必要なので，9 文字目以降の情報も保存されている．

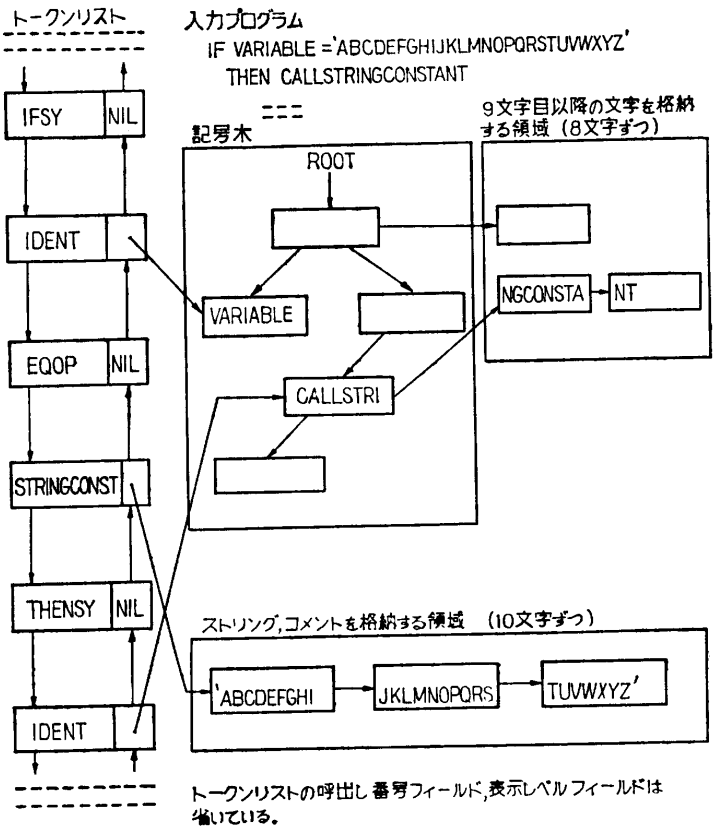

図 4　ソースプログラムの情報を格納しておくためのデータ構造

Fig. 4　Data structure for storing the information of source program.

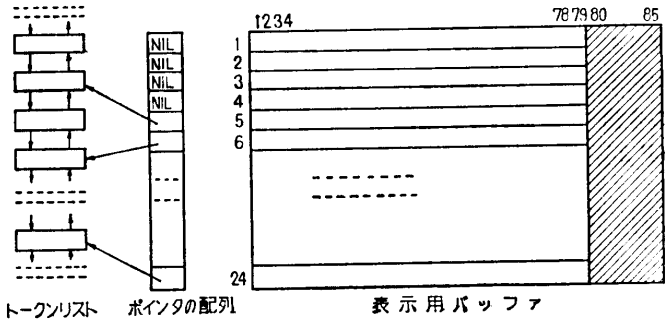

図 5　表示用バッファの構造

Fig. 5　Structure of display buffer.

CRT 画面の1行目には，現在構文解析されているプログラムの名前が表示され，2行目には，解析中のソースプログラムの手続き（関数）のネストの状態がわかるように，ネストした手続き（関数）が解析されはじめると，その親の手続き（関数）の右横にその名前が表示され，解析が終了すれば，一番右にあるその名前が消され，ユーザにはスタックのように見える．このため，解析の進行状態が一目でわかる．3行目には，エラーメッセージが数字ではなく，ことばで表示され，4行目は，1, 2, 3行目のシステムからのメッセージと，5行目以降のソースプログラムとの境界を示すのに用いられる．

なお，表示用バッファの各行に対して，トークンリストのセルを指すポインタの配列があり，表示用バッファの各行の先頭のトークンに対応するトークンリストのセルを指すようにセットされていて，表示用バッファの各行とトークンリストとの間の対応がとられている．

4. 主要な機能

この章では，PIPES の中枢をなす，あと戻り機能，修正編集機能，段付け機能について述べる．

4.1 あと戻り機能

構文解析部は，トークンリストを走査して，再帰的下向き法[4] (recursive descent) により構文解析を行う．この解析途中で誤りが検出されると，構文解析を一時中断し，ユーザによるプログラムの修正を待つ．その修正が終了した後，必要なだけあと戻りを行い構文解析を再開する．このため，従来のコンパイラに要求された誤りからの回復機能を持つ必要がない．

あと戻りを実現するために，トークンリスト上の再開位置に対応する状態に，構文解析部の状態を戻すことが必要である．その方法としては，プログラムが修正された後，

（1） プログラムの先頭を再開位置として，構文解析部の状態をプログラムの先頭を処理する状態まで戻す．

（2） 修正部分を含むソースプログラム中の手続き（これを S 手続きと呼ぶ）の先頭を再開位置として，構文解析部の状態を，そのS手続きの先頭を処理する状態に戻す．

ことが考えられるが，プログラムやS手続きの後部に誤りが検出された場合，あと戻りする距離が長くなり効率が悪い．そこで，PIPES では以下のような方法を用いて，あと戻りを行っている．

PIPES の構文解析部の手続き（これをP手続きと呼ぶ）に注目して，トークンリスト上の修正部分中最も前にあるトークンを処理するP手続きの先頭から構文解析を再開する．このとき，このP手続きの入口に対応する（このP手続きが処理する最初の）トークンから，その処理を行う．この方法は，ソースプログラムではなく，構文解析部の状態を基準にして，再開位置を決めるため，各P手続きとトークンリストとの対応をとる必要がある．そのため，**図6**に示すように，各P手続きが呼出されたとき，それぞれのP手続きの呼出しに固有の正整数（これを呼出し番号と呼ぶ）を，その呼出し時における各P手続きのローカル変数(lcallnum)に格納し，同時に，そのP手続きの入口に対応するトークンについても，トークンリストの呼出し番号フィールドにその呼出し番号を格納しておく．さらに，ポインタ型のローカル変数(lposition)にそのトークンを指させる．なお，各P手続きの入口に対応しないそのほかのトークンについては，呼出し番号フィールドを0にセットする．

PIPES は，ユーザの修正入力に従ってトークンリストを修正する．ユーザの修正が終われば，修正位置よりトークンリストをさかのぼって，初めて正の呼出し番号が格納されているトークン（図6のY）をソースプログラムの再開位置とする．一方，再開位置Yに対応する状態に構文解析部の状態を戻すには，Yの呼出し番号フィールドに格納されている番号と，P手続きの lcallnum を比較し，両者が一致しなければ，そのP手続きを抜け出して，そのP手続きを呼んだ親のP手続きへリターンする．番号が一致するまでこの操作を繰り返し，番号が一致したP手続き(図6のP2)の先頭に戻れば，構文解析部の状態は回復され，その時，その手続きの lposition がYを指しているので，あと戻りが実現できる．

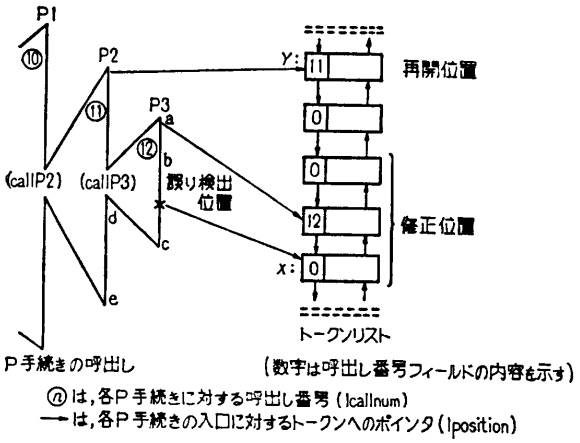

図 6 あと戻り実現の一例
Fig. 6 A way to resume parsing.

このあと戻り機能は，各P手続きの先頭部分，中間部分，終端部分，呼出し直後に，それぞれ以下に示す機能を持たせることにより，簡単に実現される．

（1） P手続きの先頭部分（図6のa）

呼出し番号を lcallnum に記憶させ，lposition にそのP手続きの入口に対応するトークンに対するセルを指させ，また，そのトークンに対するセルの呼出し番号フィールドに呼出し番号を格納する機能．

（2） P手続きの中間部分（図6のb）

解析中に誤りが検出されたとき，そのP手続きの最後（図6のc）に飛ばす機能．

（3） P手続きの終端部分（図6のc）

誤りがあった場合，そのP手続きの lcallnum と，再開位置に対応するトークンリスト上のセルの呼出し番号フィールドの番号とを比較し，一致すれば，そのP手続きの先頭に戻り，一致しない場合，あるいは誤りがなかった場合，そのP手続きの入口に対応するトークンのセルに格納されている，そのP手続き自身の呼出し番号をクリアして，親のP手続きにリターンする機能．

（4） P手続き呼出し直後（図6のd）

呼んだ子，およびその子孫のP手続き内で誤りが検出され，かつ，ユーザによる修正がそれら子孫のP手続きの解析範囲内におさまらない場合，最後（図6のe）に飛ばし，それ以外の場合，そのまま構文解析を続行する機能．

以上の4つの機能を持つP手続きの一例を**図7**に示す．ただし，図7中の番号(1)～(4)は，上の説明にそれぞれ対応する．

4.2 修正編集機能

PIPES は，既存の一般的なミニコン用のエディタ(以後，単にエディタと呼ぶ）とほぼ同等の機能を持っているが，エディタと比較した PIPES の修正編集機能の特徴としては，次のようなものがある．なお，実際のコマンド仕様を**表1**に示す．

（1） 編集中のプログラムと修正用ポインタがCRT 画面を通してユーザに見えること．

エディタでは，ユーザの打ったコマンド，および，表示コマンドにより指定されたプログラムの一部が，区別なく画面に残り，ユーザ側からは見えない修正用ポインタが存在する．したがって，ユーザはプログラ

```
PROCEDURE constdeclaration ;
  LABEL 1,2 ;
  VAR lcallnum:INTEGER ;
      lposition:tlistptr ;
      ldlevel: onemaxline ;
BEGIN
1:
  preprocess(lcallnum,lposition,ldlevel) ;  (1)
  IF sy <> ident THEN
    BEGIN
      error(2) ; GOTO 2  (2)
    END ;
  WHILE sy = ident DO
    BEGIN
      indent(0) ;
      tscanner ;
      IF sy = eqop THEN tscanner
      ELSE
        BEGIN
          error(16) ; GOTO 2  (2)
        END ;
      constant ;
      IF errflag THEN GOTO 2 ;  (4)
      IF sy = semicolon THEN tscanner
      ELSE
        BEGIN
          error(14) ; GOTO 2  (2)
        END ;
    END {while} ;
2:
  IF errflag AND (NOT breakflag)
             AND (lcallnum = startnum) THEN
    BEGIN
      return(lcallnum,lposition,ldlevel) ;
      GOTO 1
    END
  ELSE putcallnum(lposition,0)  (3)
END {constdeclaration} ;
```

図 7　P手続きの構造の一例

Fig. 7　A parser's procedure.

ムを修正する際，細かいコマンド指定をしなければならず，エディタを使い慣れていないユーザにとって，必ずしも扱いやすいとは言えない．PIPES では，CRT の特性を利用して，現在編集中のプログラムを表示し，CRT 画面上のカーソルを修正用ポインタに対応させることにより，ユーザにとって扱いやすいものにしている．

（2）コマンドから複雑なパラメータ指定を取り除き，コマンドはファンクションキーを1回押すだけで済むようにしてあること．

PIPES では，CRT 画面上のカーソル，および，キーボードのファンクションキーを利用することにより，コマンドからパラメータを削除している．また，文字の置換の場合は，カーソルを修正位置まで移動させて，文字を入力するだけでよい．

（3）プログラムの文脈を考慮したコマンドを設けたこと．

PASCAL などのブロック構造言語においては，手続きの先頭部分と本体が大きく離れてしまうことが考えられる．一方，PIPES では，CRT 画面上に 20 行のプログラムしか表示されない．したがって，本体で

表 1　コマンド

Table 1　Commands.

種類	コマンド名	機能
修正関係	1 行挿入	カーソルのある行と，その直前の行の間に1行挿入する．（カーソルは行の先頭になくてもよい）
	1 行削除	カーソルのある行を1行削除する．（カーソルは行の先頭になくてもよい）
	1文字挿入	カーソルのある位置とその直前の位置の間に1文字挿入する．
	1文字削除	カーソルの位置の1文字を削除する．
	1文字置換	カーソルのある位置の文字を入力された文字に置換する．
位置関係	1 頁戻し	表示されている先頭の行より，15行前の行から表示する．
	1 頁送り	表示されている最後の行から表示する．
	1 行戻し	表示されている先頭の行より，1行前の行から表示する．
	1 行送り	表示されている第2行から表示する．
	ヘッダへの戻り	カーソルが手続き（関数）の本体部分にある場合，その手続き（関数）の先頭部分から表示する．また，カーソルが手続き（関数）の宣言部分にある場合，親の手続き（関数）の先頭部分から表示する．
カーソル関係	カーソル右移動	カーソルを1列右へ移動する．ただし，カーソルが79列目にある場合は，同じ行の1列目に移動する．
	カーソル左移動	カーソルを1列左へ移動する．ただし，カーソルが1列目にある場合は，同じ行の79列目に移動する．
	カーソル上移動	カーソルを1行上へ移動する．ただし，カーソルが5行目にある場合は，同じ列の24行目に移動する．
	カーソル下移動	カーソルを1行下へ移動する．ただし，カーソルが24行目にある場合は，同じ列の5行目に移動する．
その他	修正終了	このコマンドにより，ユーザの修正が終ったものとみなし，構文解析が再開される．
	ブレイク	このコマンドにより，ユーザの修正が終ったものとみなされるが，構文解析は再開されずに，このコマンド以前に入力された修正情報を有効として，修正済ファイルが作成される．再び構文解析を行う場合は，この修正済ファイルを入力として，最初から構文解析を行う．

位置関係コマンドにおける行，頁は，それぞれ論理行，15論理行からなる論理頁を指す．

検出された誤りが宣言部に依存するものである場合，ユーザは宣言部を探すために手間取ることが多い．そこで，現在カーソルがある位置を含むS手続きの先頭部分を表示するコマンドを設けた．ただし，カーソルが手続きの先頭部分にある場合は，その親の手続きの先頭部分を表示する．

（4）構文解析を再開せずに修正済ファイルを作成するコマンドを設けたこと．

大きなプログラムになれば，構文解析に多くの時間がかかることが予想される．また，PIPES では，構文誤りが検出された場合，その誤りが修正されるまで構文解析を中断する．したがって，ユーザは構文解析が終了するまで，計算機から離れることができないことになるので，以前に入力された修正情報を有効として，構文解析を再開せず，直接，修正済ファイルを作

成するコマンドを設けた．これによって，プログラム作成を数回に分けて行うことができる．

4.3 段付け機能 (indentation)

プログラムを CRT 画面に表示する場合，ユーザに見やすいように段付けして表示している．その結果，プログラムの構造が明確に表現され，構文上の誤りが見つけやすくなり，ユーザがプログラムの修正を行う際の手助けになる．しかし，PIPES では，トークンリストを作成する際，入力プログラムがどのように段付けされていたかという情報は失われるため，トークンリストからプログラムを作成して表示する場合，新たに独自の段付けを行っている．

その段付け処理は次のように行われる．まず，構文解析部が，画面上で行の先頭より何文字目から表示するかを示す表示レベルを，構文解析部の定めた行（これを論理行と呼ぶ）の先頭のトークンに対するセルに書き込み，次に，バッファ作成部がトークンリストをもとに，以下の基準で，段付けされたプログラムに変換する．

（1） 原則として，1論理行は表示用バッファの1行（これを表示行と呼ぶ）に出力する．しかし，1表示行におさまらない場合は，2表示行目以降を2文字段付けして出力する．

（2） 1トークンが2表示行に亘って表示されないように出力する．ただし，1表示行におさまらない長いトークンに対しては，リスト作成部により，名前は，1表示行におさまらない (80 文字目以降の) 部分を読み飛ばし，ストリング，コメントは，1表示行におさまる長さに分割して，トークンリストを作成している．

（3） トークンの間には，1つの空白を出力する．

このように簡単な方法ではあるが，比較的見やすい出力画面が得られている．

5. あとがき

言語にあまり慣れていない初心者は，単純な構文誤りのために，何度もコンパイルをやり直すことが多く，プログラム作成上効率が悪い．PIPES では，構文解析とエディットを反復的，連続的に行うことができ，比較的応答速度もよいので，実用上かなり有用であるし，初心者に対する教育的効果もある．ただ，トークンリストがかなりのメモリを要求するため，主記憶 128 kB で (OS は，このうち 20 kB 以上を占める) 約 700 行程度の PASCAL プログラムまで処理できると見積もられる．実用上は，分離コンパイルコンパイラ[5),6)] なども利用できるので，この程度の大きさが処理できれば十分と思われる．PIPES の今後の方向としては，さらに大きいプログラムを扱うための方法として，トークンリストをディスク上に作成し，また，コンパイルを行ってオブジェクトコードを得ることも考えられる．

PIPES は構文解析システムであるが，段付け機能を持っているため，構文的に正しいプログラムのプリティプリンタとして利用でき，さらに，修正編集機能を用いて，PASCAL 専用のエディタとしても利用できる．

最後に，PIPES は PASCAL Σ で約 3,200 行から成るプログラムであるが，階層モジュール化されているため，プログラム本体だけを変更することで，1つのモジュール，あるいは，複数のモジュールを組み合わせた段階的なテストが行え，約2週間のデバッグを含めて，詳細設計から約4カ月で稼働させることができた．

謝辞 本研究に関し，ご助力，ご助言いただいた研究室の諸氏，および本文をまとめるに当って，細かくご検討いただいた萩原兼一博士に深謝いたします．

参考文献

1) Charles Wetherell: WHY AUTOMATIC ERROR CORRECTORS FAIL, Computer Languages, Vol. 2, pp. 179-186 (1977).
2) Jensen, K. and Wirth, N.: PASCAL User Manual and Report, 2nd Edition, Springer-Verlog (1975).
3) 鍵政，荒木，都倉：ミニコンピュータ NOVA 3 への PASCAL コンパイラの移植，電子通信学会総合全国大会，1474 (1979).
4) Gries, D.: COMPILER CONSTRUCTION FOR DIGITAL COMPUTERS, John Wiley & Sons (1971)， 牛島和夫訳：コンパイラ作成の技法，科学技術出版社 (1978).
5) 鍵政，荒木，都倉：手続き分離コンパイル可能な PASCAL コンパイラ，情報処理学会第 20 回全国大会論文集，4 K-2 (1979).
6) 鍵政，荒木，都倉：PASCAL 内部手続きの分離翻訳，電子通信学会論文誌 (D)， Vol. J 63-D, No. 2, pp. 177-182 (1980).
7) 菅井，荒木，都倉：CRT ディスプレイを用いた対話方式による構文エラーの回復と修正について，情報処理学会第19回全国大会論文集，5 C-3 (1978).

(昭和 54 年 9 月 14 日受付)
(昭和 55 年 10 月 23 日採録)