記号処理 12－10
（1980. 6. 20）

Pascal Machine を仮想したデバッギングシステム INSIDER

宮本 衛市（北大工学部）
堀川 博史（三菱電機）

1. はじめに

プログラムの作成に当っては，さまざまな誤りの混入が避けられず，誤りを除去するためのデバッギングあるいはテストがプログラム作成コストの中で大きな比重を占めている．もちろん，誤りの混入を引起さないプログラミング言語あるいはプログラミングの方法論の開発は必須の要請であり，これまでに多数多くの言語および方法論が開発あるいは提案されてきた．[1] しかし，プログラミングをするのが人間である以上，程度の差はあれ，依然としてデバッギングあるいはテストの過程は残らざるをえない．ところで，プログラマの主体性を認めた上で，計算機側でプログラマの活動を支援するための各種のソフトウェア・ツールの開発，さらにはそのシステム化が強く望まれている．[2]

ここで述べる支援システム INSIDER（INteractive Structured Intelligent DEbugging Reinforced system）は，Pascal プログラムを会話的，かつ構造的に実行制御し，デバッギング支援，さらにはプログラムの教育，理解，あるいは診断等を支援するシステムである．INSIDER は TSS 環境下でユーザに，いわゆる Pascal Machine としての機能を提供する．すなわち，INSIDER のもとでは，ユーザはコンパイラの存在を一切意識せずに Pascal を通して INSIDER と会話することができる．したがって，ユーザから見ると TSS 端末をコンソールとした，一種の High-Level Language Computer[3] を専用に使うことのできる環境」と考えることができる．

デバッギング支援機能としての眼目を，

(1) プログラムの実行をいかに停止し，再開することができるか，
(2) デバッギングに有効な情報をいかに取出せるか，

の2点に集約することができる．(1) に関しては，INSIDER は Pascal プログラムの構造に合わせた実行・停止制御を有しており，その他論理式で指定された中断条件による停止，計算時間設定による停止，端末からの割込みによる停止，およびプログラム異常による停止と多彩である．一方，中断後のプログラムの実行の再開は，停止した個所から，実行してきた履歴をさかのぼった個所から，あるいは指定した個所から行うことができる．また，中断時に文を投入し，変数の値の監視・変更等も可能である．一方 (2) に関しては，動的な出力情報として，指定した変数の値が更新されるたびごとに出力し，静的には中断時に任意の変数の値を知ることができる．

INSIDER はその内部に Pascal コンパイラを抱えており，ソースプログラム，機械語モジュール，翻訳情報などはすべて多分岐の木構造に展開して把握している．そこでは，Pascal の文を Pascal Machine の機械語に，そして実際の機械語をマイクロ命令に見たてると，まさに Pascal Machine のファームウェアを形成している．プログラムの実行は直接機械語モジュールを制御することにより行なわれ，INSIDER によるオーバヘッドは僅かであり，実行速度は通常のコンパイラで翻訳し実行したものと大差ないことが特長である．

2. デバッギング機能

2.1 プログラムのノード展開

INSIDERでは実行制御の単位として文とノードを考える．まず，文というときには，プログラムの任意の個所において，Pascalの構文上の完全な文であって，入れ子構造になっている文を含むときにはそれらの全体を指し，文の実行というときには入れ子構造に包含する文を含めた実行を意味する．さらに文が手続き，あるいは関数呼出しを含んでいれば，それらの全体の実行をも含むものとする．したがって，プログラムの冒頭で文の実行といえば，プログラムは1つの複文であるので，プログラム全体の実行を意味する．

一方，ノードはINSIDERが制御するプログラム上での最小単位であって，プログラムを木構造に展開したときの各ノードに対応させるプログラムの断片である．したがって，前述の文は任意のノードで木構造を切断したときの部分木に対応する．図1はwhile文を例にとってノード展開する様子を示したものであって，whileノード以外は

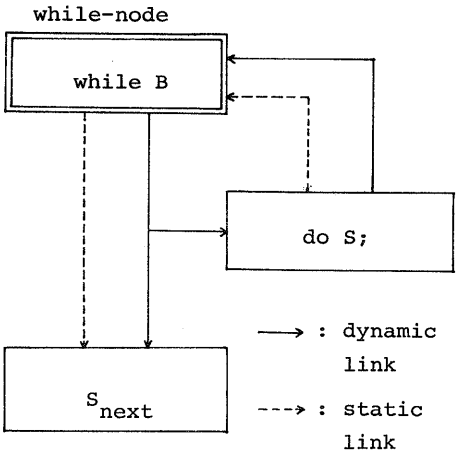

図1 while文のノード展開

文の種類に応じてさらにノード展開されるものとする．同図で実線はノードの実行順序を示す動的リンクであり，一方点線は各ノードを文の先頭ノードと考えたときに，次に移るべき文の先頭ノードを示す静的リンクである．各ノードからは1つの静的リンクと，1つ以上の動的リンクが出ていく．ifノードあるいはcaseノードのように，一般に複数個の動的ノードをもつノードが実行時にどのノードに移るべきかは，そのノードにおける数式の評価による．

2.2 デバッギング・コマンド

前述したように，INSIDERはユーザに専用のPascal Machine環境を提供する．したがって，INSIDERに対するコマンドはコンソールの押ボタンにも相当するので，できるだけの簡素化を図っている．図2に各コマンドの構文規則を示すが，コマンドを構成するものは，1ないし2文字で識別するコマンド名のほか，

```
<node> ::= <number> [.<digit>] | *
<source> ::= SO;
<bye> ::= BYE;
<jump> ::= J <node>;
<past> ::= P [<number>];
<break> ::= B <node>[,] <Boolean expression>;
<display> ::= D <node> {[,] <identifier>};
<break cancel> ::= BC <node>;
<display cancel> ::= DC <node> {[,] <identifier>};
<run> ::= R [<number>];
<continue> ::= C [<number>];
<step> ::= S [<number>];
<advance> ::= A;
<execute> ::= E <statement>;
<list> ::= L ( <Pascal I/O list> );
<where> ::= W <node>;
<frequency> ::= F [*];
```

図2 コマンドの構文規則

プログラムアドレスに相当する<node>，データアドレスに相当する<identifier>，さらにPascalの構分規則に従う<statement>，<expression>，<I/O list>，<number>のみである．

表1　コマンドの機能

分類	コマンド名	省略名	機能
指示コマンド	source	SO	テストプログラムの指示
	bye	BYE	デバッギング終了指示
	jump	J	指定したノードに現在ノードを移動
	past	P	過去の履歴上のノードへ現在ノードを移動
条件設定コマンド	break	B	指定したノードに中断条件を設定
	display	D	変数の動的な出力表示を指示
	break cancel	BC	指定したノードから中断条件を解除
	display cancel	DC	変数の動的な出力表示の解除
実行コマンド	run	R	プログラムの先頭から終りまで実行
	continue	C	現在ノードから終りまで実行
	step	S	1つの文を実行
	advance	A	1つのノードを実行
	execute	E	オペランドの文を実行
診断コマンド	list	L	指示した変数の値の出力
	where	W	指示したノードのプログラム断片等の表示
	frequency	F	ノードの実行回数の表示

表1には各コマンドの機能をまとめて示すが，大別すると指示コマンド，条件設定コマンド，実行コマンドおよび診断コマンドに分けて考えることができる．尚，コマンドにコメントを挿入することができ，Pascalコンパイラに対するオプションをこのコメントの中に含めて指定することができる．

(a) 指示コマンド　sourceコマンドはデバッギングをするプログラムをENSIDERに指示するものであり，ENSIDERはテストプログラムを翻訳してノード展開する．ENSIDERは次に実行すべきノードを現在ノードとして保持しており，sourceコマンドにより現在ノードはプログラムの先頭のprogramノードに設定される．byeコマンドはデバッギング終了を指示するものであり，ENSIDERは制御をTSSに戻す．他の2つは現在ノードの位置を変更するものであり，jumpコマンドでは直接ノード位置で指示するのに対し，pastコマンドではENSIDERが保持している最新の過去100個のノードの履歴のうち，オペランド指定した数だけさかのぼったノードに現在ノードの設定を指示する．ENSIDERは指示された現在ノードの値を出力するので，pastコマンドにより過去の履歴をたどることもできる．

(b) 条件設定コマンド　breakコマンドはプログラム中の任意のノードに対して，実行中断の条件を任意の論理式で設定するものである．これにより，後述するrunあるいはstepコマンドで実行中，中断条件が設定されているノードを通るときその条件を評価し，もしtrueであればそこで実行を中断し，設定されていた論理式と現在ノードとを知らせる．この条件設定は永久的に続き，任意の個数の条件を設定しておくことができる．この設定を解除するのがbreak cancelコマンドである．

一方，displayコマンドはプログラム実行中に，指定した変数の値が更新されるたびごとに，その値を出力するよう指示するものである．もしオペラ

ンドにノード位置のみが指定されていると，そのノードに対応する代入文が実行されるたびごとにその結果を出力するが，さらに変数識別子が指定されていると，指定されたノード位置から指定された変数識別子にスコープルールを適用したときに対応する変数の値が更新されるたびごとにその値を出力する．ただし，変数パラメータとして渡された手続き内で，他の変数名を通して値の更新がある場合は除く．値の更新が配列要素のときには，添字式をその時の値に置換えて出力する．displayコマンドも永久的に，かつ任意の個数の変数を指定することができる．これを解除するのがdisplay cancelコマンドであり，そのオペランドの意味はdisplayコマンドの設定を解除にしたそのに対応する．

(c) 実行コマンド　runおよびcontinueコマンドはプログラムの終りまでの実行を指示するものであるが，runコマンドは現在ノードには無関係にプログラムの先頭から実行を開始するのに対し，continueコマンドは現在ノードから実行を再開する．一方，stepコマンドは現在ノードから1つの文を，すなわち現在ノードの静的リンクが指すノードに到達するまで，advanceコマンドは現在ノードの実行のみを指示する．ただし，run，continueおよびstepコマンドに対しては打切り時間を設定することができるのでその時間を経過した場合，およびbreakコマンドで中断条件を設定したノードでその条件を満たした場合には実行を中断する．また，無限ループに入ったときなどのため，端末からbreak keyによる割込みにより中断させることもできる．いずれも，その時点における1ノードの実行を終了した後実行を中断し，中断の原因と現在ノードを出力する．なお，stepコマンドの実行中，対象としている文を飛出すgoto文があると，その飛び先を現在ノードとして実行を中断する．

これらの実行コマンドの実行中にプログラム異常によるOSからの割込みがあると，そのノードを実行しなかったものとして中断し，その原因コードと現在ノードを出力する．これにより，nilを値としてもつポインタで参照した場合や，0での割算，あるいはオーバフローなどの異常が，見かけ上発生直前の状態で検出されることになるので，その原因を調べるのに有効である．

executeコマンドは中断時に端末からPascalの構文規則に従う1つの文を与えて実行させるものであり，識別子は現在ノードからのスコープルールの適用を受ける．これにより，中断点における変数の値の診断・変更が可能となる．なお，executeコマンドで実行させる文は一時的なもので，プログラムに組込まれるわけではない．

(d) 診断コマンド　listコマンドはプログラムの実行が中断しているときに変数の値を出力させるためのものであり，同じことはexecuteコマンドでwriteln文を実行させても得られる．whereコマンドは指定したノードのノード位置と，そのノードに対応するプログラム断片を出力させるものであり，frequencyコマンドは各ノードの実行回数を出力させるものである．ただし，オペランドとして'*'を指定すると，通常は現在ノードを意味するが，frequencyコマンドのときだけは現在ノードを含む手続きまたは関数を意味する．

3. 内部処理方式

3.1 システム構成

図3にINSIDERのシステム構成を示す．INSIDERは1つの手続きとしたPascalコンパイラを内蔵しており，

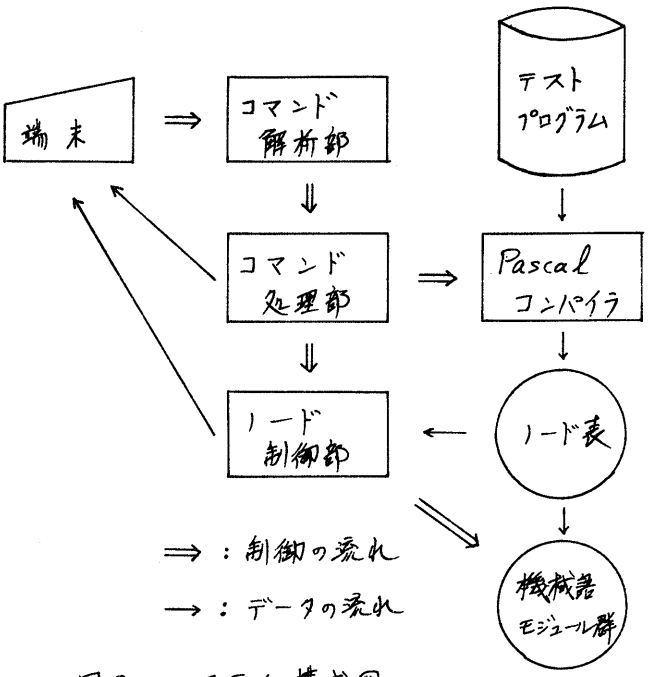

図3　システム構成図

Pascalの構文をオペランドとしてもつbreak, executeまたはlistコマンド，あるいはsourceコマンドのときに呼出され，プログラム全体，1つの文あるいは1つの数式の翻訳を行う．そこで直接機械語モジュールが作成され，実行の対象となる．

3.2 ノード構成と実行制御

各ノードは動的リンク，静的リンク，ノードに対応したプログラム断片のほか，ノードに対応した機械語モジュール，中断条件の有無と条件を示す論理式とその機械語モジュール，動的出力表示指定の有無とその機械語モジュール等を抱えている．

各ノードの実行はノード制御部のもとで行われる．ノード制御部は実行モードをINSIDER側からテストプログラム側に切換え，機械語モジュールを実行させる．機械語モジュールは最後に制御をノード制御部に戻すように作成されている．ifノード，whileノードなどのように複数個の動的リンクをもつノードでは，評価した数式の値をノード制御部に持ち帰り，その値に基づいてノード制御部は次の現在ノードを決定する．手続きおよび関数の終了ノードでは戻り先のスタックが構築され，再帰的呼出しの場合に備えている．

3.3 コマンド処理

端末からのコマンドはコマンド解析部を経て，コマンド処理部に渡される．ここで，指示，条件設定および診断の各処理が行われる．そこで必要に応じ，内蔵のコンパイラに機械語モジュールへの翻訳を行わせる．一方，実行コマンドに対しては，コマンド処理部から停止ノードを付してノード制御部に制御が渡されるので，ノード制御部は現在ノードから停止ノードまでの実行を制御する．ノード制御部は各ノードに対して，まずテストプログラムから翻訳された機械語モジュールを実行し，出力表示指定があれば表示用の機械語モジュールを実行し，さらに中断条件の設定があればその条件式に対応した機械語モジュールを実行させる．次に，break keyによる割込み，打切り時間の経過，中断条件の成立，あるいはプログラム異常の発生のいずれかが生じていると，プログラム異常のときのみ実行を中断して現在ノードを異常を起したノードに設定し，他はすべて実行中のノードの処理を終えた上で現在ノードを次に実行すべきノードに設定して実行を一旦停止する．

3.4 INSIDERの実現

INSIDER自身はPascalで記述されており，一部割込み処理や打切り時間設定，領域確保などのためのOSとのやりとりや，INSIDERから機械語モジュールへ実行を移し，再び受取る

ためなどにアセンブラ言語を用いている．INSIDERはテストプログラムの機械語モジュールを自身の変数領域に格納し，アセンブラ言語で記述したインタフェースを通して実行させる．そのために，INSIDERを記述したPascalでは次の3つの標準関数を追加している．

(i) addr(x)：変数xの番地を関数値とする．

(ii) loadmod(a)：a番地から始まる機械語モジュールを実行させる．

(iii) pasinit(m,n)：m番地からn語の変数領域の設定をし，さらにテストプログラム実行のために必要な初期設定を行う．

4. 適用例

図4にeight queensのプログラム[4]をINSIDERのもとで実験的に動かした例を示す．同図の冒頭はINSIDERを起動するためのデータセットの割付けを行うTSSコマンドのやりとりであり，CALLコマンドでINSIDERが起動される．'?'で始まる行がINSIDERへのコマンドの入力であり，コメントは説明の便宜上挿入したものである．ノード位置の表示は，ノードに対応したプログラム断片がある行番号とピリオド以下に同一行内でのノードの出現番号で行い，あわせてノード種別も表示している．以下に簡単な説明を加えるが，コマンド名は省略形を用いる．

<1>のSOコマンドでINSIDERはテストプログラムを翻訳し，ノード展開してこれを掌握する．<2>のRコマンドはテストプログラムを最後まで実行し，再び先頭に戻る．<3>，<4>で今度は逐次実行させ，<5>でAとXの値が更新したときの表示を指示する．<6>のCコマンドで再開すると，AとXを左辺にもつ代入文が実行されるたびごとに新しい代入値が表示される．この例では端末からbreak keyによる割込みをしてあり，中断後<7>でAの出力を解除し<8>で再開させるが，再び割込みで中断し，<9>でXの出力も解除し，<10>で200msを打切り時間に設定して再開している．打切り時間による中断後<11>で現在ノードを確認し，<12>で変数の値を出力させる．<14>，<15>はノードの履歴をさかのぼるものであり，手続き内まで戻って<16>でノードの実行回数を出力させる．<17>で中断条件を設定して実行させ，<19>で外から強制的にJの値を変えて別解を得ている．同様なことを<22>で無条件の中断を設定し，<24>で強制的に計算の流れをかえて行っている．<27>のBYEコマンドで制御はTSSに戻される．

これら一連の試行からもわかるように，INSIDERへはPascalのイメージのままで指示を与えてテストプログラムを思いのままに実行させ，その挙動を調べることができるので，特に誤りの個所を追いあげてゆくのには強力な手段を提供する．

5. あとがき

現在のINSIDERでは，たとえ誤りを発見してもそれを修正する手段は有していないが，高水準の編集機能[5]をあわせ持っていることが望まれる．そのためには変数の値の復旧等の対策を講じる必要があり，現在検討を進めている．

参考文献 1)鳥居他：情報処理20-1，2)和田：情報処理20-8，3) Chu, ed.: High-Level Language Computer Architecture, ACADEMIC PRESS, 4) Wirth: Algorithms+Data Structures=Programs, PRENTICE-HALL, 5)宮本他：情報処理20-6

```
ALLOC DD(INPUT) DS(*)
READY
ALLOC DD(OUTPUT) DS(*)
READY
ALLOC DD(PASIN) DS(EIGHT.DATA)
READY
CALL TOOL(INSIDER)
 INSIDER IS STARTED
?(*<1>*) SO;
 PASCAL PROGRAM SOURCE LIST
   1 PROGRAM EIGHTQUEEN(OUTPUT);
   2  VAR I:INTEGER; Q:BOOLEAN;
   3      A:ARRAY[ 1.. 8] OF BOOLEAN;
   4      B:ARRAY[ 2..16] OF BOOLEAN;
   5      C:ARRAY[-7.. 7] OF BOOLEAN;
   6      X:ARRAY[ 1.. 8] OF INTEGER;
   7
   8  PROCEDURE TRY(I:INTEGER;
   9                VAR Q:BOOLEAN);
  10   VAR J:INTEGER;
  11   BEGIN J:=0;
  12    REPEAT J:=J+1; Q:=FALSE;
  13     IF A[J] AND B[I+J] AND C[I-J]
  14      THEN
  15       BEGIN X[I]:=J;
  16        A[J]:=FALSE; B[I+J]:=FALSE;
  17        C[I-J]:=FALSE;
  18        IF I<8 THEN
  19         BEGIN TRY(I+1,Q);
  20          IF NOT Q THEN
  21           BEGIN A[J]:=TRUE;
  22            B[I+J]:=TRUE;
  23            C[I-J]:=TRUE
  24           END
  25         END
  26        ELSE Q:=TRUE
  27       END
  28    UNTIL Q OR (J=8)
  29   END(*TRY*);
  30
  31 BEGIN
  32  FOR I:= 1 TO  8 DO A[I]:=TRUE;
  33  FOR I:= 2 TO 16 DO B[I]:=TRUE;
  34  FOR I:=-7 TO  7 DO C[I]:=TRUE;
  35  TRY(1,Q);
  36  IF Q THEN
  37   FOR I:=1 TO 8 DO WRITE(X[I]:4);
  38  WRITELN
  39 END.
 NO ERRORS IN THIS PASCAL PROGRAM
     1.0   PROGRAM-NODE
?(*<2>*) R;
  1   5   8   6   3   7   2   4
     1.0   PROGRAM-NODE
?(*<3>*) A;
    32.0   FOR.1-NODE
?(*<4>*) S;
    33.0   FOR.1-NODE
?(*<5>*) D * A,X;
?(*<6>*) C;
    15.1:    X[    1] <--          1
    16.0:    A[    1] <--      FALSE
    15.1:    X[    2] <--          3
    16.0:    A[    3] <--      FALSE
    15.1:    X[    3] <--          5
    16.0:    A[    5] <--      FALSE
!
 INTERRUPTED
    22.0   ASSIGN-NODE
?(*<7>*) DC * A;
?(*<8>*) C;
    15.1:    X[    5] <--          8
    15.1:    X[    4] <--          7
    15.1:    X[    5] <--          2
!
 INTERRUPTED
    16.0   ASSIGN-NODE
?(*<9>*) DC * X;
?(*<10>*) C 20;
 TIME'S UP
    19.0   BEGIN-NODE
?(*<11>*) W *;
    19.0   BEGIN-NODE
 BEGIN
?(*<12>*) L(I,J,Q);
         5          3      FALSE
?(*<13>*) C;
  1   5   8   6   3   7   2   4
     1.0   PROGRAM-NODE
?(*<14>*) P;
    39.0   PROGEND-NODE
?(*<15>*) P 20;
    29.0   PROCEND-NODE
?(*<16>*) F *;
  NODE-ID    NODE-KIND     FREQUENCY
     8.0   PROCEDUR-NODE      113
    11.0   ASSIGN-NODE        113
    12.0   REPEAT-NODE        113
    12.1   ASSIGN-NODE        876
    12.2   ASSIGN-NODE        876
    13.0   IF-NODE            876
    15.0   BEGIN-NODE         113
    15.1   ASSIGN-NODE        113
    16.0   ASSIGN-NODE        113
    16.1   ASSIGN-NODE        113
    17.0   ASSIGN-NODE        113
    18.0   IF-NODE            113
    19.0   BEGIN-NODE         112
    19.1   PCALL-NODE         112
    20.0   IF-NODE            112
    21.0   BEGIN-NODE         105
    21.1   ASSIGN-NODE        105
    22.0   ASSIGN-NODE        105
    23.0   ASSIGN-NODE        105
    26.0   ASSIGN-NODE          1
    28.0   UNTIL-NODE         876
    29.0   PROCEND-NODE       113
?(*<17>*) B 12.1, J=1;
?(*<18>*) R;
 BREAK BY CONDITION:  J=1;
    12.2   ASSIGN-NODE
?(*<19>*) E BEGIN J:=2; WRITELN(J) END;
           2
?(*<20>*) BC 12.1;
?(*<21>*) C;
  2   4   6   8   3   1   7   5
     1.0   PROGRAM-NODE
?(*<22>*) B 12.2, TRUE;
?(*<23>*) R;
 BREAK BY CONDITION:  TRUE;
    13.0   IF-NODE
?(*<24>*) J 28;
    28.0   UNTIL-NODE
?(*<25>*) BC 12.2;
?(*<26>*) C;
  2   4   6   8   3   1   7   5
     1.0   PROGRAM-NODE
?(*<27>*) BYE;
 INSIDER IS TERMINATED
READY
```

図4　INSIDER の適用例