愛知教育大学研究報告，42（教育科学編），pp. 187～193, February, 1993

構造化プログラミング教育のためのツールの開発

北岡　武・小谷哲三*・高折健司**
Takashi KITAOKA, Tetsuzo KOTANI, Kenji TAKAORI
（情報科学選修）

1．は　じ　め　に

いろいろな分野におけるコンピュータ利用が急速に進み，しかもその利用形態が多岐にわたるようになるにつれてソフトウェア開発・保守における人材不足は深刻な問題となりつつあり，ソフトウェア危機という言葉が聞かれるようになった。信頼性の高いプログラムあるいは保守容易なプログラムの開発という問題提起にたいして E. W. Dijkstra[1]の提唱した構造化プログラミングの理論は大きな反響をよび，以来ソフトウェア生産現場あるいは教育現場でこの技法が強く主張されるようになった。

しかし，その過程で　構造化プログラミング＝GOTO — less プログラミング　という誤った考え方が一時的にではあるが流行し，悪い言語の見本として FORTRAN や BASIC が取り上げられることが多かった。また，上記の理論をかたくなに守ろうとするために GOTO 命令を使用すれば1つですむ短い命令を IF 文構造のあちこちに置くスタイルが一部に見られた。[2]

しかしその後これらの誤った解釈の行き過ぎが指摘され，GOTO — less で行の上げ下げがしてあっても理解しにくいプログラムがあること，反対に算術 IF 文や GOTO 文を使っていても判りやすいプログラムが書けること，また，構造化を守るためにモジュールの数ケ所に同一のコードを繰返し置くことは将来プログラムの変更がある場合にかえってエラーが混入する原因になることなどが G. J. Myers 等によって示された。[3]

最近のように構造化プログラミングが可能な仕様に設計されている言語を使用する場合は，それほど多大な注意を払わなくても構造化定理にかなったプログラムが出来るが，しかし，プログラミングに慣れていない初心者はそれでも犯し易いプログラム・スタイルがいくつかある。そのなかでも特に多いのが次の2つのパターンである。

① IF 文の条件設定の不適切からくる GOTO 文の多用
②事前のアルゴリズム設計不十分に起因する無用の GOTO 文の多用

プログラミングにあたって初心者は手順の分析を十分に行うことなく，与えられた条件でそのままコーディングしようとする。このために，せっかく構造化された形式をとっているにもかかわらず IF 文の THEN あるいは ELSE ブロックから GOTO 命令を使ってそのブロックから抜け出さなければならなくなる例が非常に多い。図1(a)の場合(b)のように

*松下通信工業株式会社　　**日立中部ソフトウェア株式会社

条件を変更することによって簡単なパターンに直すことが出来る。初期プログラミング教育においてはこのような誤った習性を矯正することが非常に大切なことで，このような場合，無用の制御交差があることを示唆することによって欠点に気付かせることが必要である。

構造化プログラミングの定義に関しては一般論として共通の概念はあるが，はっきりとは定義されていない。本研究の目的はFORTRAN言語を題材として学生が作成した幾つかのソース・プログラムについて分析検討した結果をもとに，言語による特殊性を考慮して拡張された構造化標準型を定義し，それにもとづき制御の流れを調べてアドバイス情報を提供するプログラム構造分析ツールを開発することである。

```
IF (B. EQ. 0) THEN
    GO TO 10
END IF
      B≠0　に対する処理
10 … 次の処理 …
              (a)

IF (B. NE. 0) THEN
      B≠0　に対する処理
END IF
        … 次の処理 …
              (b)
```

図1　条件変更による変化の例

2．システムの構成

本システムは図2に示すようにTOOL1およびTOOL2で構成されている。

TOOL1は本システムの第1ステップでこのシステムの本体をなしている最も重要な部分であり，FORTRANのソースプログラムをファイルから読み込んで語彙分析と構文解析を行い，後述するように処理の単純化のためにこれを圧縮して制御の交差を調べて結果を画面に表示すると同時に次のTOOL2のためのデータをファイルに出力する。

TOOL2はTOOL1の出力データをもとに圧縮された命令の実行順に従って利用者にわかりやすいように流れ図を描くものである。パーソナルコ

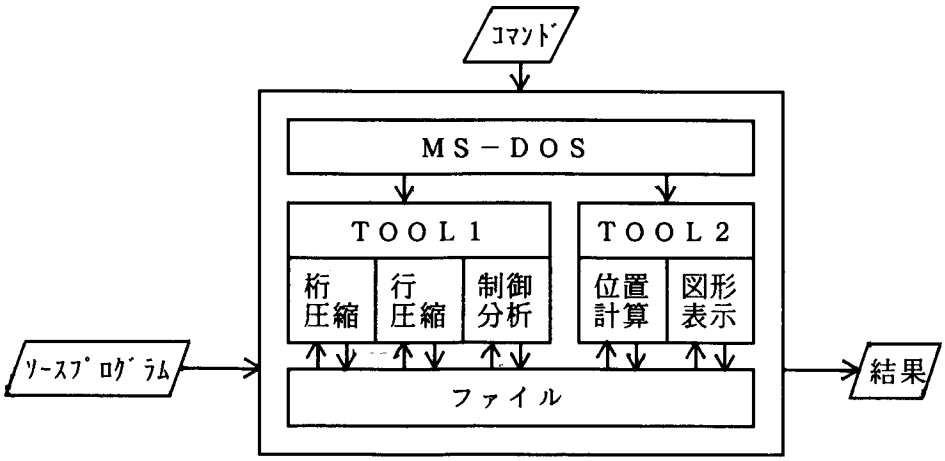


図2　システムの構成図

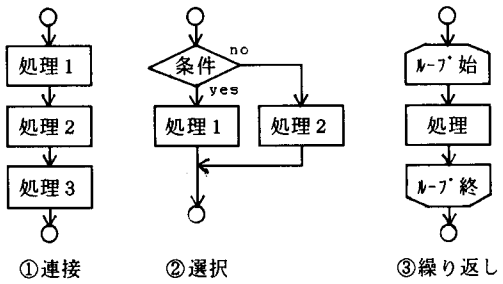


図3　3つの基本モジュール

```
   OPEN (1, FILE = 'TEST. DAT',
  1IOSTAT = IOS)
    .....
   DO 10 UNTIL (IOS. LT. 0)
    READ (1, 100, IOSTAT =IOS) X
    .....
10 CONTINUE
```

図4　ループからの飛び出し

ンピュータ PC — 9801の MS — DOS 上で稼動しているが大型計算機でも使用可能である。

なお，本システムで扱うプログラムは既に翻訳実行済みすなわちエラーの無いプログラムであることが前提となっている。

3．「構造化標準型」の定義

E. W. Dijkstra によればプログラムは①連接（concatination），②選択（selection）および③繰り返し（repetition）の3つの基本単位（モジュール）の組合せで構成出来る。しかもこれら3つのモジュールはそこへの入口，出口がそれぞれ1つであること，そしてこの原則を守ることによりプログラムが見易くなるとしている。[1]

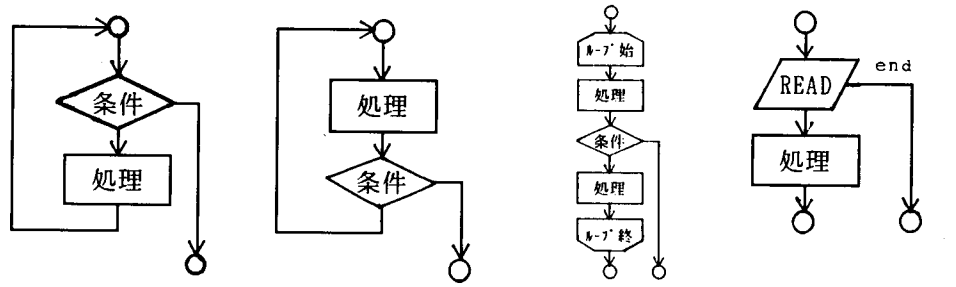
したがって，モジュールの中から IF 文あるいは GOTO 文を使ってモジュールの外へ飛び出すような場合構造化の原則を破ることになる。この原則を厳守しようとするならば，このような場合通常の FORTRAN77 では DO_UNTIL（またはWHILE）あるいはブロック IF 文を使うことになる。しかし，小規模の FORTRAN ではこのような命令が無い場合があり，またたとえあっても制御の構造上このような命令が無意味になることもある。

図4のようなプログラムの場合 READ 文でファイルの終りを検出するタイミングのほうが DO_UNTIL 命令による検出よりも先になる。このため READ 文を実行した段階でエラーになる。したがってこの場合の DO_UNTIL 文はただ繰り返しの範囲を明確にするということのためだけに使用されていることになる。

また構造化の旗手ともてはやされているブロック IF 文もその使い方によっては非常に読みにくいプログラムを作り出すこともある。すなわち ELSE ブロックにブロック IF 文と ELSE_IF 文を混在させたプログラムは構造上は完全な入れ子となっているが制御の流れが複雑で保守がやりにくい。

楠はプログラムのよいスタイルを次のように定義している。[4]

1）パターンを統一出来る部分は統一する。
2）一部分のメンテナンスが他の部分に影響しない。
3）わかりやすくするための冗長を認める。
4）なるべく単純化して考える。
5）字下げが出来ている。
6）空白行がある。



(a) DO_WHILE型　(b) DO_UNTIL型　(c) ﾙｰﾌﾟ終了子型　(d) ﾌｧｲﾙ終了子型

図5　拡張構造

7）類似の性質の部分は同じフォーマットになっている。

G. J. Myers は構造化を乱す2大要素と言われる算術 IF 文や GOTO 文を使用していても構成や記述を工夫すれば見やすいプログラムとすることが出来ると述べているが，本システムではこの2つの命令も必要に応じて認め，算術 IF 文については出来るだけ避けるように注意を促すことにとどめた。構造化の定義としては上にあげた3つの基本型のほかに図5に示す4つを加えて，これを守っている場合は構造化プログラミングの原則を乱さないものとした。すなわち

④DO_WHILE あるいは DO_UNTIL に対応する IF 文と GOTO 文の組合せ

⑤ループの中から条件によって下方へ飛び出す場合

⑥READ 文でファイル終了指定子あるいはエラー指定子により分岐する場合

ELSE_IF 文は連続して使用することが望ましいが，これに反しても必ずしも構造が不明確になるとは限定出来ないので本システムではこれに対する注意は除外した。

4．分 析 の 方 法

4．1 TOOL1

入力されたソースプログラムは以下の3段階の手順によって分析され，結果が出力される。

4．1．1桁圧縮と記号変換

このシステムではどのような命令が実行されるかは関係なく，制御構造がどのようになっているかが重要となる。

プログラムの1行全てがブランクのものあるいは注釈行は制御に関係がないので除去し，その他のものは7桁目以降の不要なブランク全てを取り除いた後，表1に示した対応に従ってそれぞれの記号に変換され作業ファイルに出力される。このとき，文番号を持っている場合はその番号をラベル表に登録すると同時にそれが CONTINUE 文あるいは FORMAT 文以外の時，DO 文の端末文の場合はその行の下それ以外の場合は上に文番号のついた CONTINUE 文が挿入される。論理 IF 文は本来は分岐にあたるがその行1行だけでの分岐なので単純命令（ASSIGNMENT）として処理する。FORMAT 文や各種の

FORTRAN の命令語	TOOL 1 の命令語
ブロック IF 文	B_IF
ELSEIF 文	ELSE_IF
ELSE 文	ELSE
ENDIF 文	END_IF
論理 IF 文+単純 GOTO 文	IF_GOTO
DO 文	DO
DO WHILE 文	DO_WHILE
DO UNTIL 文	DO_UNTIL
単純 GOTO 文	U_GOTO
ファイル終了指定子を含む READ 文	READ (E)
誤り指定子を含む命令文	ERR
算術 IF 文	3つの A_IF_GOTO
計算型 GOTO 文	複数の C_GOTO
文番号+ CONTINUE 文	文番号と CONTINUE
FORMAT 文以外の文番号 +CONTINUE 文以外の命令語	CONTINUE +該当するツールの命令語
END 文	THE_END
それ以外の命令語	ASSIGNMENT

表1　命令語の対応表

IF (N−5) 1, 2, 3⇨	A_IF 1 A_IF 2 A_IF 3
GOTO (1, 2, 3), K⇨	C_GOTO 1 C_GOTO 2 C_GOTO 3

図6　算術 IF 文等の変換例

宣言文も同様の処理をする。算術 IF 文や計算型 GOTO 文はその分岐の数だけ同じ命令が挿入される。(図6)

4．1．2行圧縮

桁圧縮されかつ1次変換された1次出力リストは次の段階で行圧縮を施される。これは分岐あるいはループに関係ない命令が複数行連続している場合，即ち連接した命令群は1行の命令と等価と考えられるので，リストの簡素化，処理の単純化のためにこれ等をまとめて1行に圧縮し，その先頭と最終行の行番号を命令表に記録する。また，分岐命令ではその分岐先の行番号とその移動方向（上向きあるいは下向き）を，ループ命令はループの端末行の行番号を命令表に記録する。ただループ命令はその勢力範囲を表わしているだけでそこから制御が変更されるという意味ではないので，制御の移動方向は NO_EXIST（向きが存在しない）と記録する。

4．1．3制御分析

第3段階では前の処理で出力された命令表をもとに

① GOTO 文と GOTO 文あるいは算術 IF 文によって出来る交差

```
   12行     構造化プログラミングにふさわしくない、命令語計算形ＧＯＴＯ文がある。

交点    12 -     15    C_GOTO        DOWN
  1      8 -     14    U_GOTO        UP

    8      10 WRITE(*,*) 'Input data(I5) & ¥/$='
   --           .....
   12          GO TO(1,2,3),K
   --           .....
   14           GO TO 10
   15        2 R=NINT(A/RATE*100)


交点    12 -     20    C_GOTO        DOWN
  2      8 -     14    U_GOTO        UP

                     ..) 'Input data(I5) & ¥/$='

交点    12 -     20    C_GOTO        DOWN
  3     19 -     22    U_GOTO        DOWN

   12          GO TO(1,2,3),K
   --           .....
   19          GO TO 20
   20        3  Y=NINT(A*RATE)
   --           .....
   22       20 STOP


            行              数          23
            実 際 の 行 数              23
            交　点　の　数               3
A:¥PROG¥P0302.FOR                の処理終了
```

```
    PROGRAM P0302
    INTEGER A,R,D,C,Y,K
    REAL RATE
    CHARACTER U*1,W*2
    DATA W/'¥$'/
    WRITE(*,*) 'Type in rate(yen/$) ='
    READ(*,*) RATE
 10 WRITE(*,*) 'Input data(I5) & ¥/$='
    READ(*,100,END=20) A,U
100 FORMAT(I5,A1)
    K=INDEX(W,U)+1
    GO TO(1,2,3),K
  1  WRITE(*,*) 'Data Error!!'
     GO TO 10
  2 R=NINT(A/RATE*100)
    D=R/100
    C=MOD(R,100)
    WRITE(*,*) A,'yen = $',D,'.',C
    GO TO 20
  3  Y=NINT(A*RATE)
     WRITE(*,*) '$',A,' = ',Y,'yen'
 20 STOP
    END
```

ソース・プログラム

図7　TOOL 1 の出力例

② DO ループから自分より上方向へ飛び出す GOTO 文や算術 IF 文による DO ループとの交差

③算術 IF と他の算術 IF 文や GOTO 文との交差

を調べてその交点の数と交点を構成する２つの命令の情報を表示する。（図７）

なお FORTRAN には予約語という規則がないので FORTRAN の命令語と全く等しい変数を使用することが可能であるが，これらについても完全に分析している。

４．２ TOOL2

TOOL2 は本システムでは補助システムで TOOL1 で出力されたデータをもとに圧縮された形でのフローチャートを表示して交差の様子をより明確にユーザーに提示するもので，ユーザーの要求により画面あるいは紙面上に出力するものである。

５．ま　と　め

構造化されたプログラムかどうかを判断する目安として７つの構造化基本型を定義し，これ等を基準としてプログラムの制御の流れに注目し，出来るだけ交差の少ないプログラムを作成するようにアドバイスする教育システムを開発した。従来大学等の教育機関でのプログラミング教育においてはプログラム・スタイルの教育にはあまり注意が払われていなかったが，PASCAL 等の新しい言語が普及するにつれてようやく動くプログラムを作ることだけでなく，わかりやすい保守容易なプログラムの作成教育に重点をおく教育・研究が進められるようになった。平成５年度から中学校においても技術・家庭科の教育内容に「情報基礎」が加えられることになり，簡単なプログラミング教育が実施されることに

```
 1     1START      NO_EXIST      0     0
 1     7ASSIGNMENTNO_EXIST       0     0
 8     8CONTINUE   NO_EXIST     10     0
 8     8ASSIGNMENTNO_EXIST       0     0
 9    22READ(E)    DOWN          0    20
10    11ASSIGNMENTNO_EXIST       0     0
12    13C_GOTO     DOWN          0     1
12    15C_GOTO     DOWN          0     2
12    20C_GOTO     DOWN          0     3
13    13CONTINUE   NO_EXIST      1     0
13    13ASSIGNMENTNO_EXIST       0     0
 8    14U_GOTO     UP            0    10
15    15CONTINUE   NO_EXIST      2     0
15    18ASSIGNMENTNO_EXIST       0     0
19    22U_GOTO     DOWN          0    20
20    20CONTINUE   NO_EXIST      3     0
20    21ASSIGNMENTNO_EXIST       0     0
22    22CONTINUE   NO_EXIST     20     0
22    22ASSIGNMENTNO_EXIST       0     0
23    23THE_END    NO_EXIST      0     0
```

⇧

ＴＯＯＬ１からのデータ　　ＴＯＯＬ２の出力 ☞

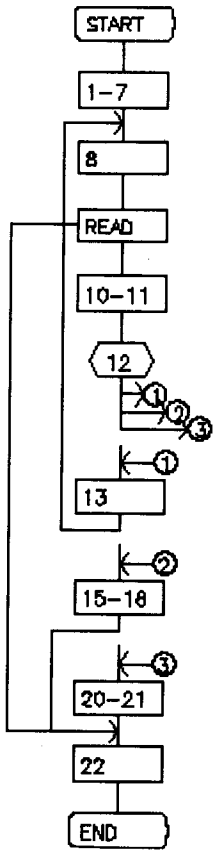


図８　流れ図出力（TOOL 2 出力）の１例

なったが，使用される言語はいろいろな事情で PASCAL や QUICK-BASIC のような構造化プログラミングにふさわしい言語ではなく，ほとんどの学校で従来からあるディスク BASIC が用いられるものと思われる。本システムはそのような場での活用が期待出来ると考える。

（平成４年９月11日受理）

引　用　文　献

1）E. W. ダイクストラ，C. A. R. ホーア，O. J. ダール共著，野下浩平，川合慧，武市正人共訳：" 構造化プログラミング"，サイエンス社（1989）

2）V.R.BASILI and H. D. MILLS : "Understanding and Documenting Programs" , IEEE Trans. Software Eng . , vol. SE-8, No.3, May, 1982, pp. 270-283

3）G. J. マイヤーズ著，有澤　誠訳：" ソフトウェアの信頼性"，近代科学社（1989）

4）楠　房子，宮内　新，小沢慎治：" アルゴリズムスタイルを重視した情報処理教育"，電子情報通信学会論文誌，vol. J75-A, No. 2, FEB. , 1992, pp. 441-448