FORTRAN プログラムの実行回数および制御構造解析システム†

野崎剛一†† 阪上直美††

本論文は，FORTRAN プログラムの各実行文の実行回数とプログラムの制御の流れをソースプログラム・リスト上に図示するシステムの開発について述べている．そして，また FORTRAN の各ステートメントがどのような頻度で使用されているかという統計情報をも提示しかつ，その累計をとり，FORTRAN 言語についてのコンパイラ設計やプログラミング言語教育の検討材料を得る機能をもたせた．このシステム開発中に，長崎大学情報処理センターのシステム内にカタログされている利用者の FORTRAN プログラムについて，各ステートメントの使用頻度の統計情報を得たので報告する．このシステムを使用すると，FORTRAN ソースプログラムリスト上に各ステートメントの実行回数と，DO 文の入れ子構造，GO TO 文等による飛び先および飛び込みが図示されるので，処理の流れが一見でわかり，プログラム作成時の不注意によるミスを容易に発見することが可能である．したがって，このシステムは，FORTRAN プログラムのディバッキングツールならびにプログラミング教育用のツールとして使用されると，その効果は大きいものと思う．

1. まえがき

FORTRAN は，1956年ごろに世界最大の計算機メーカである IBM 社により開発されたプログラム言語であり，その後，数多くの利用実績を通じて，しだいに改善されて，ISO において国際的な標準言語としてまとめられている．そして，現在では，この言語の利用者層は拡大し，とくに大学の学術研究においてその利用者は非常に多い．たとえば，全国共同利用の九州大学大型計算機センターにおける利用者のコンプリート形式プログラムの使用頻度の調査[1]のうちコンパイラ，インタプリタ，プリプロセッサおよびアナライザ等の使用頻度**表1**によると，使用言語の比率は，FORTRAN GE, HE を合わせると実に97%にも達している．九州大学大型計算機センターにおいて，このような数値が出ているので，多少の差があったとしても，全国の大学の計算機センターでも同じような傾向があると思われる．

そこで，本稿では，圧倒的に利用者の多いコンパイラ言語 FORTRAN で作成されたプログラムについて，そのプログラムの各文の実行回数と制御文による処理の流れの変化を表す情報を合わせて提示するシステムについてその概要を示す．

† An Analysis System of Executions and Control Structure of FORTRAN Programs by KOICHI NOZAKI and NAOMI SAKAUE (Information Processing Center, Nagasaki University).

†† 長崎大学情報処理センター

2. FORTRAN プログラムのディバッグ機能

FORTRAN プログラムのディバッグ作業は，まず文法チェックを行い，テストデータを入力し実行してみることから始まる．そして，実行エラーが発生した場合には，エラーの発生個所の発見を行う．最近の多くの処理系には，ディバッグの作業が容易になるように，プログラムの実行に関して次のような情報を出力するディバッグ機能が用意されている[2]．

（1） 副プログラムの実行表示
（2） 文番号付実行文の実行表示
（3） 代入文や READ 文などによるデータ格納領域の更新値表示
（4） 配列の添字の値のチェック
（5） その他

これらの機能を利用する場合，通常の処理系では，プログラム内で，ディバッグの対象とする部分を指定できるようになっている．

しかし，プログラムの制御の流れを追跡する機能(2)を利用すると，追跡情報を大量に印刷する結果になってしまうことがあり実用に耐えないことが多い．

多くの大学の計算機センターでは，FORTRAN 利用者がプログラムのディバッグの過程や未知の FORTRAN プログラムを解読するときなどに使うツール(道具)として FORTRAN プログラム動的解析システム[3]，FORTRAN のフローアナライザ[4]などが開発され使用されている．

これらのツールは，ディバッグ時に非常に有力なも

表 1 コンプリート形式プログラム使用頻度調査
Table 1 Frequency of use of complete programs.

Program	1980年4月～1981年3月	1979年4月～1980年3月
ALGOL	279(0.1%)	357(0.2%)
ASSEMBLER	216(0.1%)	279(0.1%)
BASIC	7(0.0%)	0(0.0%)
COBOL	646(0.3%)	311(0.1%)
COBOLDAP	45(0.0%)	0(0.0%)
FAST	0(0.0%)	2(0.0%)
FORDAP	1,093(0.5%)	574(0.3%)
FORPREX	67(0.0%)	38(0.0%)
FORT-77	0(0.0%)	105(0.0%)
FORTFLOW	209(0.1%)	276(0.1%)
FORTRAN GE	155,327(65.2%)	138,640(60.4%)
FORTRAN HE	74,780(31.4%)	82,039(35.7%)
FORTUNE	101(0.0%)	138(0.1%)
HLISP & REDUCE 2	0(0.0%)	11(0.0%)
LISP	29(0.0%)	143(0.1%)
PASCAL & PASDAP	1,105(0.5%)	998(0.4%)
PL/I	3,685(1.5%)	5,212(2.3%)
RATFOR	1(0.0%)	0(0.0%)
RATFORC	367(0.2%)	327(0.1%)
SL/100	0(0.0%)	0(0.0%)
SNOBOL 4	122(0.1%)	23(0.0%)
SSOPTRAN	0 (0.0%)	99(0.0%)
Total	238,079	229,572

のとなる場合が多いが，動的解析とフローアナライザの二つのツールはあわせて使用したほうがその効果が大きいように思われる．そこで筆者らは，プログラムの動的解析と静的（制御の流れ）解析機能を合わせもつシステムを開発した．

3. システムの概要

このシステムの処理の流れを図1[5]に示す．このシステムでは，被解析プログラムは，すでにFORTRAN コンパイラにより文法的なエラーがないことが確認されたものであると仮定している．したがって，文法的なチェックを行わなくて済むので，その分だけ処理が簡単になる．

このシステムは，プログラムの実行時の流れをソースプログラムリスト上に各実行文の実行回数とともに表示するので，処理の流れを視覚的にとらえるのに有効である．そして，読みやすいプログラムを記述するための参考になり，プログラム言語教育にも活用できる．また，被解析プログラムのステートメント種別累計情報を出力するとともに，システムのデータセットにその累計情報を累積させておく機能を有している．

解析の対象となる FORTRAN 原始プログラムが実行を伴う場合は，前処理を行い，そうでない場合は，直接制御構造を図示するルーチンで処理を行う．

実行を伴う被解析原始プログラムには，前処理部により実行回数計数用のステートメントなどが挿入される．この変換されたプログラムは，FORTRAN コンパイラによってコンパイルされ，オブジェクトモジュールに変換される．そして，リンケージ・エディタあるいは，ローダによって，通常の FORTRAN ライブラリのほかに実行解析用のルーチンを結合してロードモジュールを作成する．このロードモジュールが実行されると，実行終了の直前に前処理部により作成されたプログラムが読み込まれ，被解析プログラムの各実行文の実行回数を原始プログラムに対応付けて出力する．

これらの処理が終了した後で，FORTRAN プログラムの制御構造を図示するルーチンが起動される．

3.1 前処理部

この部分では，被解析プログラムの各ステートメントの実行回数計数用のカウンタをソースプログラムに文法的な誤りが生じないように次のような規則[5]で挿

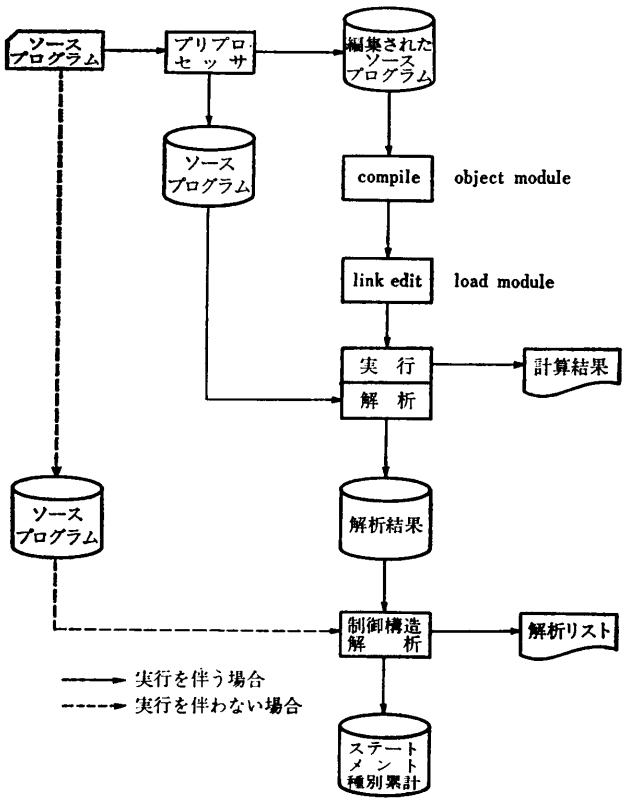

図 1 システムの流れ図
Fig. 1 General flow of this system.

入していく．

(1) カウンタ用配列の宣言

(2) 最初の実行文の直前にカウンタ用配列の初期値設定

(3) 継続行か注釈行かの解析

(4) 宣言文か FORMAT 文かの解析

(3),(4)の場合はカウンタを挿入しない．

(5) SUBROUTINE 文の場合

カウンタ用配列の宣言文を挿入

(6) 論理 IF 文の場合

IF(論理式) 実行文

⇒IF(論理式) ¥IC999(k)=¥IC999(k)+1
IF(論理式) 実行文

IF(論理式) STOP

⇒IF(論理式) ¥IC999(k)=¥IC999(k)+1
IF(論理式) CALL ¥¥¥¥
IF(論理式) STOP

(7) STOP 文の場合

CALL ¥¥¥¥
STOP

(8) DO 文の場合

新たな DO ループを作成し，その端末文番号は元の端末文番号に 20000 を加えたものとする．

$$\left[\begin{array}{l}\text{DO } n_1\ I=m_1, m_2, m_3\\ n_1\ \text{CONTINUE}\end{array}\right.$$

$$\Rightarrow\left[\begin{array}{l}\text{DO } (n_1+20000)\ I=m_1, m_2, m_3\\ n_1\ \text{¥IC 999}(k)=\text{¥IC 999}(k)+1\\ \quad\text{CONTINUE}\\ (n_1+20000)\quad\text{CONTINUE}\end{array}\right.$$

(9) その他の実行文の場合

¥IC 999(k)=¥IC 999(k)+1
実行文

この場合の実行文に文番号が存在するときは，カウンタ文にその文番号を付け，その実行文の文番号は削除する．

以上の処理により，被解析プログラムには，次のような制限事項が要求される．

(イ) 変数名 ¥IC 999 を使用していないこと

(ロ) 20000 以上の文番号を使用していないこと
もし使用した場合には，前処理部が生成した文番号と重複することがある

(ハ) 文関数定義文を使用していないこと

(ニ) 入出力機番 98, 99 を使用していないこと

(ホ) サブルーチン名 ¥¥¥¥を使用していないこと

これらの制限事項に対する対策としては，(イ),(ホ)に関するような名前はほとんど使用されないであろうから問題はないと思う．(ロ)に関する文番号の制限については，プログラム単位ごとに使用されている文番号をすべて調べて重複しない文番号を生成するようにすればよい訳であるが，処理を簡単にするためにこれまでの機能はもたせていない．(ハ)の文関数定義文に関しては，この文を使用している場合は，プログラム単位の先頭に最初の実行文の位置（mn：先頭から mn 行目）を示す文（*mn 以下空白）を置けばよいようにしている．これは，このシステムで，すべての実行文の実行回数を計数するために各実行文の直前にカウンタを挿入しているので，文関数定義文の前にカウンタ文（実行文）を挿入しないようにしなければならないからである．なぜなら文関数定義文の位置は，最初の実行文よりも前でかつ宣言文よりも後にでなければならないという文法上の約束があるためである．この制限事項を除去するためには，関数名と思われる名前が宣言文中に出てきた配列名と一致しないことを確かめなければならない．(ニ)に関しては，大部分の FORTRAN プログラムでは，この入出力機番は使用されないであろうから問題はないと思う．もし使用されている場合は，被解析プログラム側で入出力機番を変更しなければならない．

ところで，前処理部による被解析プログラムのプログラムサイズの増加の度合はおおよそ次のとおりである．

カウンタ用配列…実行文の個数×整数型配列領域

カウンタ用配列の初期値設定サブルーチン…7 ステートメント

カウンタ文…実行文の個数

実行解析結果の出力部サブルーチン…69 ステートメント

3.2 制御構造解析部

この部分では，被解析プログラムの各ステートメントを読み込み，制御文による処理の流れの変化に関連する情報を取り出し，それらの情報テーブルを作成する．解析の対象としているステートメントを次に示す．

(1) 単純 GO TO 文

(2) 計算型 GO TO 文

(3) 割当て型 GO TO 文

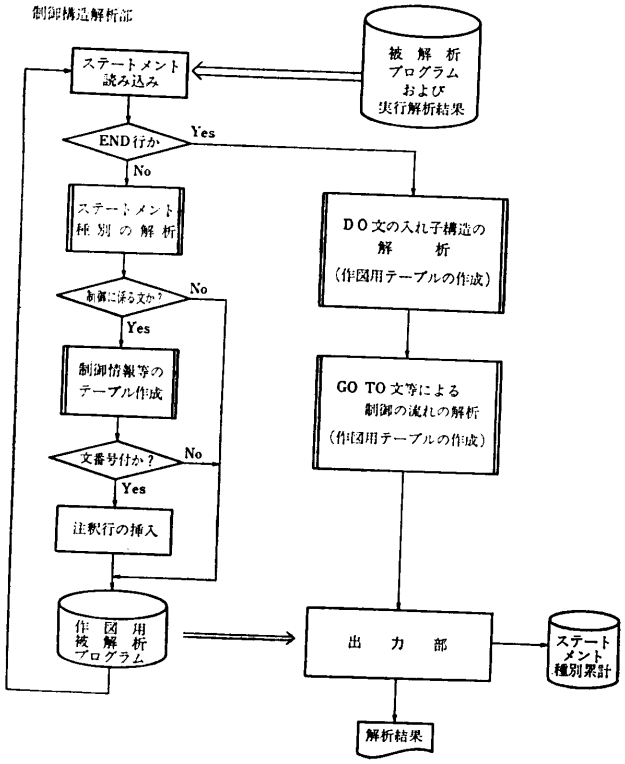

図 2 制御構造解析部の流れ図
Fig. 2 General flow of control structure analysis.

(4) 論理 IF 文中の GO TO 文
(5) 算術 IF 文
(6) DO 文
(7) 文番号付実行文 (GO TO 文等による飛込口)
(8) END 指示子付 READ 文
(9) ERR 指示子付 READ 文

ただし，これらの各ステートメントは，1行以内に納まっているものとする．

この制御構造解析部の処理の流れは**図2**に示すとおりであり，その概要を次に示す．

(イ) 制御文ならば制御情報等のテーブル作成
(ロ) 文番号を有するならば，次に注釈行を挿入
(ハ) (イ), (ロ)の処理を一つのプログラム単位が終了するまで繰り返す．
(ニ) DO 文の入れ子構造の解析
ソースプログラム・リストの左側の余白に縦の流れ線を引き，入れ子構造を成すときは順次内側に縦の流れ線を引く．
(ホ) GO TO 文等による制御構造の解析
(イ)で作成した被解析プログラムの制御情報等のテーブルをもとに，処理の流れを変える実行文に関して，その流れの変化をソースプログラム・リストの右側の余白に次の要領で図示する．
(a) 制御が移るステートメントには「−>」の記号を入れ，そこから飛び先のステートメントまで縦の流れ線を入れる．
(b) 縦の流れ線は，下方向を「|」，上方向を「:」で示す．
(c) 制御が入るステートメントには「<−」の記号を入れる．
(d) 縦の流れ線の位置から「−>」または「<−」の記号の位置までを「−」で結ぶ．
(e) 縦横の流れの重なる部分は「+」記号でその重なりを示す．
(f) 同一文への上下方向からの飛び込みについては上方向を示す「:」を優先させる．
(g) 制御の切れ目の検出
制御の切れ目では，「−>」または「<−」の記号の直後から流れ線を描く．
(ヘ) 全体の出力情報の横方向の寸法調整
1行当りの出力制限桁数を越える行がある場合は全体の出力情報のうちソースプログラムの末尾の部分をカットするように調整する．

3.3 解析例

このシステムによる FORTRAN プログラムの解析例を**図3**に示す．この出力リスト中の注釈行はすべてこのシステムにより挿入されたものである．この注釈行の挿入でプログラムの制御の流れが視覚的により明確にとらえられるようになる．この出力例では 31 行目から処理の流れは，その行以下の部分に移っていることが明確である．DO 文の範囲は，左側の余白に示されていて DO 文の中の GO TO 文等による制御の変化と対応させて把握できる．また，ソーステキストの右端に示されている数字は，その実行文の実行回数であり，論理 IF 文の場合にはさらにその論理式が真となった回数も表示するようにしている．

一般的に，制御構造が複雑になってくると，このシステムの出力情報が1行当りの出力制限桁数を越えてしまう場合が多い．この場合は，ソースプログラムの末尾の部分からその不足桁数分のテキストを表示しないようにする．筆者が調査した約 17 万枚の FORTRAN ソースプログラムの統計結果によると，注釈行以外のステートメントの末尾の空白桁数の平均値は44.8桁であったことから考えて，ソースプログラムの末尾の部分が多少表示されなくても，プログラムの制御構

```
1           INTEGER*2 IS,ISP,I1,I0,ISS
2           DIMENSION IS(68)
3           DATA ISP,I1,I0/1H ,1H1,1H0/
4           M=1
5           NUM=1
6           LL=0
7           IPG=1
8      1000 READ(66,60,END=200)ISS
9      C
10       60 FORMAT(A1)
11          BACKSPACE 66
12          READ(66,1,ERR=200)IS
13        300 FORMAT(68A2)
14          NUM=NUM+1
15        1   FORMAT(1X,68A2)
16          IF(ISS.EQ.ISP)LL=LL+1
17          IF(ISS.EQ.I0)LL=LL+2
18          IF(ISS.EQ.I1)GO TO 10
19          IF(ISS.NE.ISP.AND.ISS.NE.I0.AND.ISS.NE.I1)LL=LL+1
20          IF(LL.EQ.67.OR.LL.EQ.68)GO TO 10
21          GO TO 20
22       10 IPG=IPG+1
23     C
24          LL=1
25          IF(ISS.EQ.I0)LL=2
26       20 WRITE(60,2)ISS,IS,IPG,LL
27     C
28        2   FORMAT(A1,68A2,2I2)
29          MP=IPG
30          GO TO 1000
31       200 REWIND 60
32     C
33          GO TO 500
34       201 READ(5,51)NPG
35     C
36          IF(NPG.EQ.0)STOP
37          IF(NPG.GT.MP)GO TO 500
38     |--      DO 101 J1=1,NUM
39     |        READ(60,2)ISS,IS,IPG,LL
40     |        IF(IPG.EQ.NPG)GO TO 103
41     |--  101 CONTINUE
42        C
43       103 BACKSPACE 60
44     C
45          104 WRITE(6,502)
46     C
47          502 FORMAT(1H ,14Hｼｭﾘｮｸｻﾚﾏｾﾝﾃﾞｼﾀ?)
48          READ(5,51)NLL
49          600 M3=0
50        C
51     |--      DO 105 J2=1,66
52     |        READ(60,2,END=107)ISS,IS,IPG,LL
53     |        M3=M3+1
54     |        IF(NPG.NE.IPG)GO TO 107
55     |        IF(NLL.EQ.LL)GO TO 102
56     |--  105 CONTINUE
57        C
58          107 NLL=NLL+1
59        C
60               IF(NLL.EQ.67)GO TO 700
61     |--      DO 106 J3=1,M3
62     |        BACKSPACE 60
63     |--  106 CONTINUE
64        C
65               GO TO 600
66          102 BACKSPACE 60
67        C
68     |--      DO 111 I=1,66
69     |        READ(60,2,END=400)ISS,IS,MJ
70     |        IF(MJ.NE.NPG)GO TO 400
71     |        WRITE(6,3)ISS,IS
72     |    3   FORMAT(A1,68A2)
73     |--  111 CONTINUE
74        C
75          400 REWIND 60
76        C
77               GO TO 510
78               STOP
79          500 WRITE(6,501)MP
80        C
81          510 WRITE(6,504)
82        C
83          504 FORMAT(1H ,18Hｼｭﾘｮｸｸﾝｻｸﾍﾟｰｼﾞﾊ ?)
84          501 FORMAT(1H ,11Hｼｭﾘｮｸｸﾝｻｸﾍﾟｰｼﾞﾏﾃﾞﾃﾞｽ,)
85               GO TO 201
86           51 FORMAT(I2)
87          700 WRITE(6,509)
88        C
89          509 FORMAT(1H ,16Hｺﾉﾓﾉﾖｳｶﾗﾊ ｳﾂｶﾅｲｼ.)
90               M2=M3
91     |--      DO 800 JJ=1,M2
92     |        BACKSPACE 60
93     |--  800 CONTINUE
94        C
95               GO TO 104
96               END
```

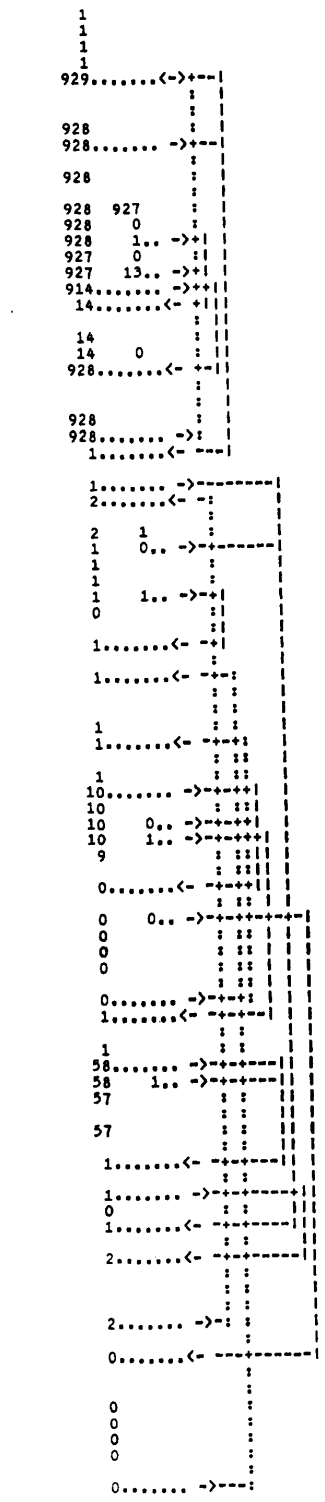

図 3 システムの出力例 (1)

Fig. 3 An example output of this system (1).

```
ASSIGNMENT STATEMENT  12      15.58%
TYPE*BYTES STATEMENT   1       1.30%
GO TO                  7       9.09%
IF( )                  5       6.49%
IF( ) GO TO            8      10.39%
DO                     5       6.49%
CONTINUE               5       6.49%
WITH STATEMENT LABEL  19      24.68%
WITH STATEMENT LABEL  14      18.18%
(EXCEPT CONTROL STATEMENT)
STOP                   1       1.30%
READ                   3       3.90%
READ( ,END= )          3       3.90%
READ( ,ERR= )          1       1.30%
WRITE                  6       7.79%
BACKSPACE              5       6.49%
REWIND                 2       2.60%
DIMENSION              1       1.30%
INTEGER                1       1.30%
DATA                   1       1.30%
FORMAT                10      12.99%
END                    1       1.30%
TOTAL OF SOURCE TEXT  77
MEAN OF SPACE COLUMNS         50.88
```

図 4 システムの出力例 (2)

Fig. 4 An example output of this system (2).

造および実行解析には問題がないと思われる．そして，また，ソースプログラムの末尾の空白を少なく表示したほうが右側余白に示される流れ線と対応づけやすい．

図4 は，被解析 FORTRAN プログラムのステートメント種別累計表である．

4. FORTRAN ステートメントの出現頻度について

このシステムの各ステートメントを解析するルーチンを使って，長崎大学情報処理センターの利用者の FORTRAN プログラム 173,071 枚について調査した結果を**表 2** に示す．

この調査したソースプログラムは，おもに理工学系の科学技術計算処理を行っている利用者のもので，プログラムサイズはおおよそ 200～300K バイト程度のものであった．

これによると，FORTRAN は数値計算向きの言語であることから，代入文の出現率が 46% で圧倒的に多いのは当然であるが，制御文が 29.4% で，制御文に関連する文番号付実行文まで合わせると，その出現率は，約 35.7% であることがわかる．このように，制御構造に関連する文の出現頻度が高いことは，FORTRAN では，PL/I 風の IF-THEN-ELSE が使えないどころか，文をグループにまとめることさえもできないから IF-THEN-ELSE 型の構造を IF 文と GO TO 文を使って組み立てる必要があるためでもある[6]．このために，プログラムの作成の仕方によっては，プログラムがたいへん読みにくくなる．また DO 文の構造についてみると，DO 文の端末文として CONTINUE 文が使用されている割合は，62.4% であり，残りは文番号付実行文が端末文となっているか共通の端末文が DO 文の端末として使用されていることを示している．とくに DO 文の範囲が入れ子を成している場合，DO 文とその端末文との対応が 1 対 1 でないと，プログラムが読みにくかったり，論理ミスの原因になったりするので，DO 文はその端末文と

表 2 FORTRAN ステートメント使用頻度調査

Table 2 Frequency of use of FORTRAN statements.

ステートメント	テキスト数(比率%)
単純 GO TO 文	4,621 (2.67)
計算型 GO TO 文	295 (0.17)
割当型 GO TO 文	7 (0.00)
算術 IF 文	2,425 (1.40)
IF 文	3,268 (1.89)
IF ( ) GO TO 文	6,540 (3.78)
DO 文	15,181 (8.77)
CONTINUE 文	9,473 (5.47)
ASSIGN 文	72 (0.04)
STOP 文	969 (0.56)
CALL 文	5,705 (3.30)
RETURN 文	2,378 (1.37)
文番号付実行文	24,865 (14.37)
この内制御文以外の文番号付実行文	10,927 (6.31)
READ 文	2,336 (1.35)
END 指示子付 READ 文	51 (0.03)
ERR 指示子付 READ 文	20 (0.01)
WRITE 文	5,844 (3.38)
BACKSPACE 文	17 (0.01)
REWIND 文	98 (0.06)
ENDFILE 文	15 (0.01)
DIMENSION 文	1,641 (0.95)
COMMON 文	5,060 (2.92)
EQUIVALENCE 文	29 (0.02)
INTEGER 文	425 (0.25)
REAL 文	572 (0.33)
LOGICAL 文	14 (0.01)
DOUBLE PRECISION 文	288 (0.17)
COMPLEX 文	5 (0.00)
IMPLICIT 文	261 (0.15)
型宣言*バイト数	393 (0.23)
DATA 文	1,126 (0.65)
FORMAT 文	6,231 (3.60)
SUBROUTINE 文	1,786 (1.03)
FUNCTION 文	40 (0.02)
EXTERNAL 文	18 (0.01)
DEFINE FILE 文	55 (0.03)
END 文	2,596 (1.50)
代入文	79,438 (45.90)
注釈行	17,075 (9.87)
継続行	14,137 (8.17)
その他	431 (0.25)
総テキスト枚数	173,071
末尾の空白欄平均	44.77

1対1の対応をもたせるように心掛けるべきであるといえる.

5. あとがき

本稿では，FORTRAN プログラムの解読，ディバッグ，実行解析のための一つのツールの開発について述べた．このシステムでは，FORTRAN プログラムの制御構造がソースプログラムリスト上に流れ線により図示されるとともに，各実行文の実行回数までがわかりやすい形で示される．したがって，プログラムのディバッグおよび実行効率の改善に有効であり，プログラム解説用のドキュメント作成のツールとしても活用できる．

今後，FORTRAN 77 の利用が多くなると予想されるので，このシステムを FORTRAN 77 用に改造し，実際の利用に供する予定である．

参考文献

1) 九州大学大型計算機センター広報，Vol. 12, No. 4, p. 403 (1979)；Vol. 13, No. 2, p. 247 (1980)；Vol. 13, No. 4, p. 548 (1980)；Vol. 14, No. 2, p. 329 (1981).
2) *FACOM OSIV/F4 FORTRAN GE Manual*, p. 76, pp. 115-125 (May, 1978).
3) 牛島和夫：FORDAP—FORTRAN プログラム動的解析システムーについて，九州大学大型計算機センター広報, Vol. 9, No. 2, pp. 100-110 (1976).
4) University of Tokyo Computer Centre: *Annual Report*, No. 8, pp. 47-48 (1978).
5) 牛島和夫：FORTRAN プログラム診断のすすめ，*bit*, *8*, p. 46 (1977)；*bit*, *9* (1977).
6) Kernighan, Brian W. and Plauger, P. J.: *The Elements of Programming Style* (木村 泉訳)：プログラム書法，p. 49. 共立出版，東京 (1976).

(昭和56年8月13日受付)
(昭和57年4月19日採録)