HIPAC 103 による HARP 5020 ソースプログラムのエラーチェック*

田中　一**　塩崎洋一**　遠藤　修**
江丸敏夫**　村田茂昭***　栃内香次***

1. 序　論

東京大学大形計算センターは，全国国公私立大学関係研究者等の共同利用を目的として設立されたものである．この目的のためには，プログラムや計算結果の輸送の不便さを何らかの方法で解決しなければならない．もし，当大学所属の中形機を用いて FORTRAN (HARP 5020) のプログラムエラーを検出することができれば，大形機利用の便宜さは格段に増大するはずである．以上の理由から，われわれは HIPAC 103 を用いて FORTRAN (HARP 5020) のプログラムをチェックするプログラムを作成したので，ここにその概要を報告する．

2. チェック・プログラムの構成

2.1 基本方針

HARP 5020 の文法により作られたソースプログラムがカードイメージどおりに HIPAC 103 のコードで紙テープにパンチされているとする．ただし，各行のうしろのスペースは不要である．この紙テープの HITAC 5020 側の処理については東大の分担である[1]．また現段階では各デックごとにチェックを行なう．

2.2 メモリーの配分

当大学の HIPAC 103 はコア 1024 語，ドラム 8192 語のメモリーをもつが（1語 48 ビット），これを 256 語ずつのページにわける．ドラムの 0～3 ページは外部メモリー，4～31 ページは内部メモリーであるが，速度向上のため演算（ソースプログラムのチェック）はコアで行なうのを原則とする．チェックプログラム本体はドラム 4 ページ以降にあり，ブロック転送されてコア 3 ページで動作する（このブロック転送を制御するサブルーチンを BTR と称する）．またコア 0 ページは各種共通ルーチン（BTR, SIR など）各種 WS 用に使い，コア 1～2 ページは各種表をもってくるための WS である．HISIP アセンブラーはドラムの 0～3 ページに格納されている．

2.3 メインルーチン

2.3.1 ソースプログラムの読み込み

まず，1行分紙テープをよみ，WSA にストアする (primary input)．HIPAC 103 のコードは上中下 3 段にわたっているので，これを 1 コード 1 字に対応させるためにパターン変換（以下 P 変と略す）を行なう．P 変されたコードをこのチェックプログラムでは内部コードと称する．WSB は内部コードに変換された 1 ステートメント（以下 St. と略す）分のソースプログラムを入れる WS である（はじめに WSB は空である）．WSB がゼロであると，WSA の内容は新しくはじまった行と断定される．新しくはじまった行は正しく書かれている場合は次の二通り (a) および (b) である．

（a） つづきがないと断定できる行
（第1字が下記のどれかである場合）
¥（$）…コントロールカードにあたる行．
#（☆）…HITAC 5020 への紙テープ入力プログラム用[1] special function のマーク（現段階では，これらの行のチェック（**第1図**点線部分）のプログラムは組み込まれていない．将来，job ごとのチェックを考えるとき組みこむ予定）．

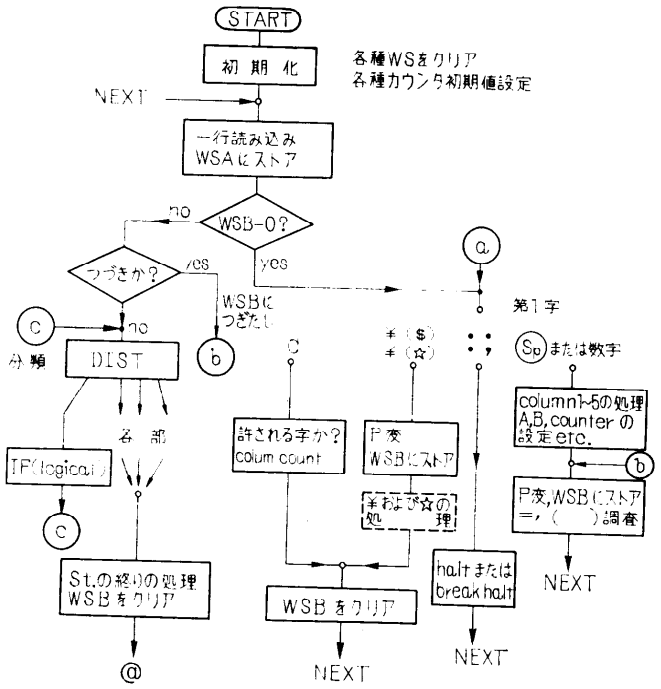
C … comment
: … halt mark
; … break halt mark
} チェックを一時停止するマーク．HITAC 5020 側では no effect とされる

これらは，WSA の内容を P 変していったん WSB にうつす．そして必要な処理が終ると WSB をクリアする．

---

* Error Check of HARP 5020 Source Program on HIPAC 103, by Hajime Tanaka Yooichi Shiozaki, Osamu Endo and Toshio Emaru (Faculty of Science, Hokkaido University), Shigeaki Murata, and Kooji Tochinai, (Faculty of Technology, Hokkaido University)

** 北海道大学理学部　*** 北海道大学工学部



第1図　メインプログラム フローチャート

input routine) と称する．SIR は入口が四つあり，リンクの仕方により働きがかわる．WSB の内容を分類する部分が第1図の DIST にあたる．

エラー検出の便宜のために A counter と B counter とをもうける．

A counter：最近に定義された St. No. を示す．

B counter：その St. No. の定義点より何 St. 目かを示す．

あるプログラムの最初の St. においては A=0, B=1 とする．

2.3.2　分類プログラム

まず =，( ) の現われ方で DO と式を分離する（**第2図**参照…このための情報は前述のごとくP変段階で記録している）．次に IF (の分離をし残りを一般の St. とする．

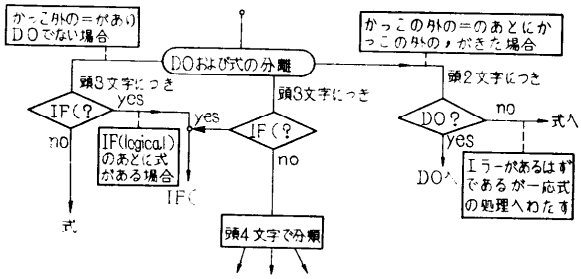
一般の St. は頭4文字をしらべて下記に分類する．

（b）　つづきのある可能性のある行．（(a) 以外）

これらは HARP 5020 の St. である．ステートメントナンバー（以下 St. No. と略す）のある場合は St. No. を記録（定義された St. No. の処理は St. の種類が判明したのち行なう）したのち，St. ごとに必要な初期化を行ない WSB にP変してストアする．この際に =，( ) の現われ方を記録する．これはあとで DO と式を分離するための情報となる．その後1行読み込みにもどる．

1行読み込みのあと，WSB がゼロでないときは，WSA の第1字から6字までを調べる．第1字～5字まで space で第6字がゼロでない数字のとき，WSA の内容は一つ前の行につづくとみなし，WSA へ P 変つぎたしを行なう．

それ以外の場合は WSA をそのままとし WSB（上記のごとく continuation の処理のすんだ 1St. 分のソースプログラムが，内部コードに変換されてストアされている）の内容のチェックにかかる．WSB から1文字ずつ読みだすサブルーチンを SIR (secondary



第2図　分類プログラム (DIST) フローチャート

（a）　transfer St.…GO TO など（STOP も含める）

（b）　(a) 以外の実行可能 St.

（c）　非実行 St.（FORMAT を除く）

（d）　FORM (FORMAT)

（e）　END Ⓔ（Ⓔ は End Mark … St. の終りを示す内部コード）

それぞれについて

- St. No. の定義の有無に関する処理．
- 制御表（後述）に登録する必要があれば登録．
- その他．

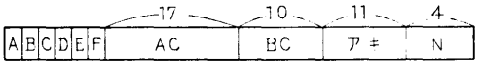
の処理を行なう．ただし5文字以降の St. 本文に関してはここではたちいらない．

式は論理式，数式と St. 関数の区別がついてから，IF(は，IF (logical) と IF (arithmetic) の区別がついてから上記の処理をする．なお，論理式，数式は (b)，St. 関数は (c)，IF (arithmetic) は (a) とする．IF (logical) は一応 (b) に分類しておき，チェックが，それにつづく St. まで進行したときに必要に応じて変更する．

2.4 制御表，DO 表および St. No. 表

2.4.1 制御表 (1023 語)

オブジェクトプログラムを作らない代わりに，St. No. の定義された St. および St. No. を指定する St. を表に書きこんでいく（**第3図**参照）．



第3図　制御表の構成

第3図の ABC の各ビットは St. の種類を示す（**第1表**参照）．なお，これは DIST. における分類とは異なる．

第　1　表

St. の種類	A	B	C
雑	0	0	0
入出力	0	0	1
DO	0	1	0
transfer	0	1	1
FORMAT	1	0	0
DEBUG	1	0	1

その他のビットについては

D: IF (logical) がついているとき 1，他は 0．

E: St. No. の定義点で 1，他は 0．

F: E が1のときのみ意味をもつ．その St. No. が一度以上指定されると1となる．

AC: Acounter の内容（最近の St. No.）
本来15ビットでよいが，5桁以内で32767(2^{15}-1) より大きな数を St. No. として定義しても Acounter には保存するため17ビットとする．なお，このような大きな数字は後述の St. No. 表には入らない．

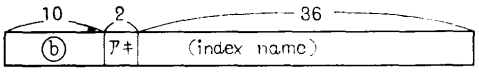
BC: Bcounter の内容（AC より数えて何 St. 目かを示す）．

N: DO nest の深さ
初期値は 0，DO St. が現われるごとに制御表の次の行から深さを一つふやす．DO の終りの St. の次から終了した分だけ減らす．

〔注〕 E と F と AC, BC については冗長度大であるが，これは，F ビットに対応する情報を St. No. 表に入れることが語長の関係でむずかしかったことと，END 処理を容易にするために，この形となった．

2.4.2 DO 表 (15 語)

チェック点でまだ閉じていない DO ループにつき表を作る．ある DO が閉じると nesting が正しいか



ⓑ: その index name に対応する DO の範囲のはじまりの制御表における相対番地（制御表において DO の次の行にあたる）

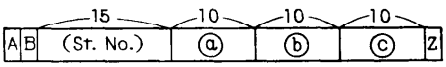
第4図　DO 表の構成

どうかしらべたのち，表の一番深い nest に対応する行をクリアする（このため DO ループの nesting のエラーが表われると2度以上ちがう所で，エラーとなる）．

2.4.3 St. No. 表

スピードアップのため St. No. の下位9ビットをその St. No. 表中における相対番地とする．すでにそこに他の St. No. が登録されているときは512を法として次をさがす．メモリーの節約のため表は可変長にしてあり，このため，St. No. のソースプログラムにおける出現の仕方により，同じ St. No. を使用しても違った表となることがある．

（1） はじめの語（**第5図**参照）



A=0　B: 既定義 1　未定義 0　Z: つづきあり 1　〃 なし 0

第5図　St. No. 表（はじめの語）

（a） St. No. が定義された場合
ⓐ: 定義点の制御表における相対番地
B=1 ⓑ, ⓒ, z=0

（b） 未定義で3回以内指定された場合
ⓐ, ⓑ, ⓒ: 指定点の制御表における相対番地
B=0, Z=0

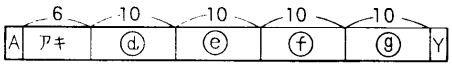
（c） 未定義で4度以上指定された場合
つづきの語をもうける．この時はじめの語にお

いて

ⓒ: つづきの語の St. No. 表における相対番地

Z=1

（2） つづきの語（**第6図**参照）



A=1　Y: つづきあり 1, なし 0

第6図　St. No. 表（つづきの語）

ⓓ, ⓔ, ⓕ はそれぞれ 3, 4, 5 回目の指定点を表わす. ⓖ と Y については, はじめの語の ⓒ と Z と同様である.

このようになっているので表全体の大きさが 512 語をこえぬ限り St. No. 未定義のままの指定回数に制限はない.

何度も指定されたあとで, St. No. が定義されると, 指定の処理をしたあとで, 定義済の形にする. このとき, つづきの語は, そのまま残される（これをクリアすると表をひくときの原則がみだれる）.

2.4.4　定義された St. No. の処理

（1） 二重定義のチェック, St. No. 表へのかきこみ

（2） 既指定なら, 下記3項目につきSt. No. 表の指定点のリストおよび制御表より, St. No. の定義された St. と指定 St. の対応をしらべる.

（a） I/O と FORMAT

（b） transfer St. 関係

（c） DO ループ

（b）の重要部分は DO nest に関連する飛び先のチェックである. ここで指定点を Ⓡ, 定義点を Ⓓ（チェック時点）. チェック時点における最も深い nest のはじまりを Ⓢ とする（いずれも, 制御表における相対番地である）. nest の深さ N に R, D, S の suffix をつけて表わすものとすると

$N_D=0$　　O.K.

$N_R<N_D$　エラー

$N_R \geq N_D$…のときは

Ⓢ>Ⓡ　　エラー

Ⓢ≤Ⓡ　　O.K.

となる.

2.4.5　指定された St. No. の処理

（1） 未定義のときは St. No. 表に記録する.

（2） 既定義のときはただちに上記 (a), (b) の処理を行なう. (b) の重要部分は上記と同様であるがただし Ⓡ がチェック時点となる. また Ⓢ′ を N_D と同じ深さの nest のチェック時点におけるはじまりとすると

$N_R \geq N_D$ のときは

Ⓢ′>Ⓓ　エラー

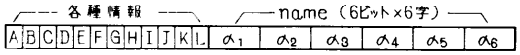
Ⓢ′≤Ⓓ　O.K.

となる.

2.5 Name 表およびサブプログラム表

2.5.1　Name 表とサブプログラム表

変数, サブプログラム, ブロック名の3種の name は, すべて name 表に登録される. 表は 512 語あり, 1語あたり1個の name を登録する. 構造は**第7図**



第7図　name 表

に示すとおりで, 上位 12 ビットに各種の情報が入り, その下 36 ビットに name 各文字を格納する.

A～L 12 ビットの内容は次のとおりである.

（1） AB: name の種類

00…コアにとった変数. 01…ドラムにとった変数.

10…サブプログラム名. 11…ブロック名.

（2） C～L: AB の内容により, 次の3種にわかれる.

1） AB=00 あるいは 01（変数）の場合

C: Common ステートメントに出現したとき 1

DE: 変数の型　00…実数,　01…整数,　10…複素数,　11…論理変数

FG: 語長　00…1倍精度,　01…2倍精度,　10…4倍精度

HI: 添字の数　00…単純変数,　01…一次元,　10…二次元,　11…三次元

J: 変数のとる値が確定したとき 1

K: 変数が数式の右辺などで使用されたとき 1

L: サブプログラムの仮の引数であるとき 1

2） AB=10（サブプログラム）の場合

C: 種類　0…関数,　1…サブルーチン,

D～K: 附随するサブプログラム表の相対番地

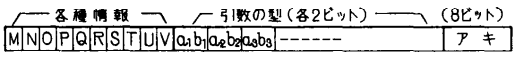
L: External ステートメントに出現したとき 1

3） AB=11（ブロック名）の場合

C: 種類　0…コア,　1…ドラム.

以下36ビットを6ビットずつに区切り，内部コード表現で，name 各文字を格納する．文字は右につめられ，6文字未満のときは上位に0が入る．

サブプログラムの場合はその引数の型を記憶するため，もう1個表を作り，サブプログラム表とする．表の構造は**第8図**のとおりで，name 表の D～K ビットでサブプログラム表の相対番地を示す．そのため変数の場合の D～K ビットにあたる情報はサブプログラム表に移される．



第8図　サブプログラム表

M～V 10 ビットの内容は次のとおりである．

MN：関数の（結果の）型　00…実数，　01…整数，
　　10…複素数，　11…論理数

OP：関数の（結果の）語長　00…1倍精度，
　　01…2倍精度，　10…4倍精度

Q：サブプログラムが定義されたとき1

R：サブプログラムが数式等で使用されたとき1

STUV：引数の個数（0～15）

なお，サブルーチンの場合は MNOP＝0000 である．

以下は2ビットずつに区切り，各引数の型を name 表の DE ビットと同じやり方で登録する．ステートメント関数，組み込み関数も全く同じ取り扱いをしており，後者は，初期化の際に108個全部を登録する．

2.5.2　Name およびサブプログラムの登録

name 登録プログラムは，チェックと登録を合わせて行なうようになっており，登録しようとする name が既に登録済みであるかどうかをしらべ，その情報を特定番地にセットする，登録，問合わせ両用の機能をもつ．

ある name を表の何番地に対応させるかが問題になるが，HIPAC 103 は索表命令をもっていないので，索表時間の短縮と一様化のため，name のコードから計算する方法をとった．すなわち，name を構成するコードを36ビットの2進数とみて，定数を乗じ，結果の下から7～15の9ビットをとり，それを番地としている．この方法ではいくつかの name に対し，同じ番地が計算されることがあるので，その場合は求めた番地の先を探し，空いている最初の番地を割当てる．

サブプログラムの登録は，上のようにして name 表に登録したあとで，情報の移動，附随するサブプログラム表の番地算出などを行なう．サブプログラムは変数よりずっと数が少ないので，サブプログラム表は先頭から出現順に割当てている．表の容量は40個である．

2.6　エラーメッセージルーチン

チェック中にエラーが発見されると，エラーメッセージルーチンにより8文字以内のエラーメッセージと a＋b の形のエラー検出点とを印字する．ただし a は A counter の内容（最近の St. No.）であり，b は B counter の内容であり a から数えて何番目の St. であるかを示す．リンクの仕方によってはさらに8文字以内の内部コードを外部コードにかえて印字できる．その他の情報についてはエラーメッセージルーチン終了後各所で必要に応じて印字する．

3.　各ステートメントの処理

3.1　仕様ステートメント

本チェックプログラムでは，処理の内容がほぼ同一な次の各ステートメントを，仕様ステートメントとして処理している．

1）型の宣言．　2）配列宣言．　3）Common，Equivalence　4）関数，サブルーチン
5）External　6）Word length

これらはいずれも，述語のうしろに name，数字などのリストが並ぶ形式である．したがってチェックは次のように行なわれる．

1）述語のつづりは正しいか．　2）位置はプログラム上，正当か．　3）リストの各要素はこのステートメントに出現してよい種類か，形式は正しいか，区切り記号は正しいか．

これらのチェックを行ない，正しければ name 表への登録など，必要な処理を行なう．正しくない場合はエラーの処理へ移るが，代表的なエラーとしては，述語のつづりの間違い，記号の間違い，name の字数の多すぎ，組み込み関数名を変数として使用，などがある．

word length ステートメントの場合は，宣言されたビット数により，語長を登録しておき，型の宣言をうけない name の語長を決めるのに用いる．

3.2　式

3.2.1　式チェックの概略

式のルーチンは等式の右辺，算術 IF ステートメン

ト，論理 IF ステートメント，関数およびサブプログラムの引数として共用できるようにサブルーチン形式に作られている．各場合における式の終了記号は次のようである．等式右辺の場合は End mark, IF ステートメントの場合は)，引数の場合は，または) である．

式のチェックに先き立ち，式が論理式になり得るかどうか，すなわち式中に論理名，論理定数，論理演算子，比較演算子が存在しているかどうかで，論理式として扱うか，算術式として扱うかを決定している．

算術式においては，算術演算子（+，−，*，/，**）の使用法，これら算術式で結合される算術項の型（real, integer, complex）が調べられる．算術式終了と共に左右のカッコが対をなしていたかどうかが調べられる．また算術式は比較演算子（.EQ.,. GE., etc.）で結合されて式中に現われることがあるので，算術式の終了記号としては上述の式の終了記号に加えて小数点以外の・も含まれる．

論理式においては論理項が正しく論理演算子で結ばれているかどうかを調べる．算術式を比較演算子で結合させて論理項として用いられるので，この場合は，この演算子で結ばれる算術式の型が正しいかどうかの判定が先立って行なわれる．

3.2.2 各要素のチェック

（1） 添字のチェック

変数の添字を読みとり，チェックを行なう．おもなチェック点は次のとおり．

1） 添字の次元数が配列宣言に一致しているか．

2） 添字の構成要素が整数型で $\alpha*I+\beta$ の型をしているか．

（2） サブプログラム表の索表

サブプログラム表を調べて必要な情報を読み出すサブルーチンがある．使い方はサブプログラム表の登録ルーチンとほぼ同様である．

（3） パラメータのチェック

ステートメント関数のパラメータの型をサブプログラム表の a_ib_i の所に入れてゆき，前述したように 15 個までしかかけないので，それ以上のときはエラーとなる．

（4） 定数処理

定数（real, complex, logical および Hollerith の 4 種のみ）の書式についてチェックする．

（5） 論理，比較演算子の検出

“.” 記号が出現すると，論理または比較演算子であると考えられるから，それを検出する．ただし，定数のうしろに 12. のように現われた場合は real const. の場合であるが，その場合もまず，その後が論理比較演算子であるかどうかを調べ，そうでなければあらためて，real const の処理に入るようにしている．

（6） 組み合わせのチェック

演算子の両側の数の型が許される組み合わせであるかどうかは数式チェックの重要な部分であるが，ここでは，巾乗以外の数演算子，巾乗演算子，比較演算子の 3 種に分けてチェックを行なっている．等号の両側の数の間の組み合わせも同様にしてチェックされる．

3.2.3 数式および論理式

これらの全体を制御する論理式処理ルーチンと数式処理ルーチンがサブルーチン型式に作られており，式のチェックルーチンを形成する．これは CALL St. または関数などで引数がさらに式になっている場合や，IF St. のカッコの中の式などの場合にも，そのまま用いられるようになっている．式のチェック全体を制御するために，さらに等式の左辺および右辺の処理ルーチンがあって，最終的にはこれら二つのルーチンに入ることによって式のチェックが行なわれる．IF のカッコの中の式や，CALL，関数の引数の場合は，各々の処理ルーチンから入ってくる．以下式のルーチンを使用する各場合について，概略を記す．

（1） 等　　式

左辺が等式のものとして（ステートメント関数の定義を含む）許される名であることを調べて，左辺の型をあとのチェックのために保存しておく．等号の次から右辺のルーチンへ入り前述の順に従って調べられ，最後に左辺の型と右辺の型との組合わせが許されるものであるかどうかを調べて，等式のチェックを終了する．

（2） IF St. のカッコ内の式

左カッコの次から，式のチェックルーチンで処理され，右カッコの数が左カッコより多くなったところを終了点とする．

（3） 引　　数

サブルーチンサブプログラムの引数の場合は単に式のルーチンに接続してチェックを行なうだけであるが，関数の場合は式のルーチン内でまた同じ式のルーチンを用いるので，リンクの情報，式のチェック段階における現在までの型，関数の型などを一時待避さ

せて行なっている．引数の型は，式チェックを終了したときの型とされ，サブプログラム表の情報に従って，引数の数と型の一致が調べられる．

3.3 transfer St. などと DO St.

このチェックプログラムでは STOP, RETURN などもプログラムの流れの上で transfer St. として扱っている．その他 ASSIGN, IF (logical) などもこの項で扱う．

3.3.1 transfer St. など

各 St. の形式上のチェックを行なったのち St. No. のリストのチェックを行なう（**2.4** 参照）．

RETURN については存在の有無のみを記録しておき END 処理のときにサブプログラム中で正しく使われたかどうかを調べる．

IF（が来た場合あとにつづく表現を式処理ルーチンの一部により判定する．これにより IF (logical) と IF (arithmetic) と区別する．その後，今まで保留されていたメインルーチンにおける処理を行なう．（2. 3.2 参照）IF (logical) の場合は，ただちにそれにつづく St. の分類にとりかかる．computed GO TO に現われる name は name 登録ルーチンにより型を調べる．assigned GO TO についても全く同様に行なう．このためこれについてはチェックは不完全なものになる．assigned GO TO および ASSIGN についてはエラーが検出できなくとも警告のメッセージを出す．

3.3.2 DO St.

DO St. にゆるされた形のチェックをするが，なお St. No., Index name については各種表（**2.4** 参照）を照合しながら必要な処理を行なう（この節は **2.4** に密接に関係があるのでそちらも参照のこと）．

3.4 入出力と DEBUG St.

3.4.1 I/O

チェックの方法は左から1字ずつ区切りまで読みこみ，文法的に正しいかどうかを見て，右に進む．一度エラーが出た場合は，つづいてチェックをすることができない場合が多いので，変数の最初が数字のように見なされたときに，チェックを中止する．St. No. に関しては，St. No. 表に登録する．特にリストの場合は，サブルーチン形式で，添字付とか，繰り返えし形式については，そのサブルーチンの終りをみるのに左かっこと等号の個数に合わせて行なっている．繰り返えし指定の添字付変数で等号の前の整数型変数（Index）は name 表に登録した上で，添字付変数が DO St. の Index かのチェックを行なう．出て来る変数については，すべて入力と出力に対して区別して name 表に登録する．

3.4.2 FORMAT (t)

t は仕様で，チェックは左から1字ずつ行なう．仕様で許される文字は H-field を除いて，かなり制限されるので，すべての文字に対して一様に調べていく方法をとっている．特に，行に対する文字の制限（1行の field）に対してチェックを行なうが，この場合，FORMAT が READ St. で指定されるか，WRITE であるかの区別がついていないので，仕様の中に H-field があれば，WRITE とみなして 120 まで，H-field がない場合は READ とみなして，80 までチェックする．チェックの最後に左向きかっこと右向きかっこのチェックを行なう．したがって，途中にエラーがあっても中止せずに最後までエラーチェックを行なうので，エラー表示の印字で個数の制限はしない．

3.4.3 DEBUG

文法的には七通りの様式に分けられる．チェックはすべて左より1字ずつ行なっている．St. No. は St. No. 表に登録する．特に IF(e) に対しては数式ルーチンへとばして，チェックが行なわれ，エラーがない場合はもどって来てチェックを続行する．チェックの中止は，I/O 関係と同じで，特にエラー表示の印字の個数に制限はない．

3.5 END ステートメント

チェックプログラムという性格上，なるべく多くの情報を出力するようにしている．

（1） 総体的情報

メイン，サブプログラムの別．後者の場合は関数，サブルーチンの別とその name を出す．いずれの場合も RETURN に関するエラーの有無を出力する．

（2） name

変数，サブプログラム，ブロック名のおのおのに分けて登録された name を各種情報とともに出力する．

（3） 未定義ステートメントナンバ

指定のみされていて未定義のものを出力する．

（4） 定義済ステートメントナンバ

定義のみなされ，一度も指定されていないものには ? をつける．

4. データの処理

4.1 データチェックの方法

このデータ処理はチェックプログラム本体と独立のプログラムとなっている.

このプログラムはデータと FORMAT の対応をチェックしリストとの対応は行なわない. またデータをカードイメージどおりに紙テープにパンチすることは困難である. このプログラムはこれらのチェックと困難解消とを目的としたものである. 実際のチェックには読みこまれるデータに対応する FORMAT を書きデータをスペースもしくは復改1個以上を区切りとしてパンチすればよい. FORMAT のコンパイルは左右のかっこ, 各要素ごとの規定のチェックを行なう. このようにコンパイルされた FORMAT によりスペース間を1個のデータとしてチェックを行なう. データは各要素ごとに分類され, I, F, E, D 要素であればそれぞれデータがその型にあっているか, O要素であれば7以下の数字か, L 要素であれば最初が F, T であるかをチェックする. A, H 要素は特にチェックしない. ついで要素共通にデータが FORMAT で指定したフィールド以下であるかを, チェックしたのち, フィールドに合うようにスペースをつけて紙テープに打ち出す. このようにしてデータのチェックおよびカードイメージへの変更を行なっている. データが二つ以上の FORMAT で読み込まれるときはその区切は：をつけ, 同じ FORMAT でくり返えし読み込まれるデータの区切りに；を使用する. データの最後は：；とする.

5. 結 び

HIPAC 103 のメモリーは, コア 1024 語, ドラム 8192 語であって, 中形計算機としても小規模である. さてチェックの対象となるソースプログラムの規模の制限はほとんどの場合, 制御表と name 表の制約からくると考えてよい. チェックは各デックごとに行なわれるので, 各プログラムの St. No. の数が 500 程度を下まわり, かつ name の数が 400 個以下でなければならない. この制限をみたすソースプログラムの St. の数は通常の場合, 多くて 2000 個程度である. 実用に供されている大部分のソースプログラムの各デックはこの制限をみたすと思われるので, 実用的な意味においても, 中形計算機を用いて大形計算機のプログラムエラーをチェックすることは可能であることが示されたものと思われる. なお, 現在のチェックプログラムでは, 各種の表がドラムに常駐しているので, チェックには多少の時間を要している. またプログラム単位でチェックが行なわれるので, サブプログラムのリンクに関係するエラーはチェックされていない. チェックに要する時間を短縮し, チェック処理をプログラム単位から job 単位に改善することは今後の課題である.

最後にチェックプログラムの構想の検討と name 表と name 登録ルーチンの作成に協力頂いた内山美代子さん, および紙テープパンチとオペレーションに協力頂いた北大計算センターの皆さんに感謝の意を表明する.

参考文献

1) 綜合研究「電子計算機の全国的綜合使用法」資料集: 1967, 1,28. (東京大学の部「HITAC 5020 への紙テープ入力プログラムについて」高橋秀俊, 後藤英一, 亀田寿夫)

2) HITAC 5020 FORTRAN (HARP), プログラムマニュアル (第2版), 日立製作所, 昭和 40 年11月

〔附録〕 チェック例 (附図参照)

1行目の CONTCARD はコントロールカードにあたる行の存在を示すメッセージである.

次に, 附図に現われたエラーメッセージのみを順を追って説明しよう.

(1)	SYMBOL	区切り記号のまちがい (この St. のチェックは次の行にうつる).
(2)	FORM NO. 3	Fa, b で a−b≥3 でない.
(3)	I≡R	compare operator による許されない組み合わせ.
(4)	St. IN. DO	DO ループ中に St. 関数がある (この例では (1) のために生じた).
(5)	NOT EXC	プログラムの流れがここに到達しない.
(6)	TRANSID	transfer St. の飛先が DO nest の中でゆるされない所にある.
(7)	R : I	real と integer の許されない組み合わせ.
(8)	DO 4	DO の Index name がダブッて

〔付 図〕

```
CONTCARD
SYMBOL   error  00000+00001
FORMNO.3 error  00003+00000
I=R      error  00003+00002  COMB-TYP
ST.IN DO error  00100+00005  00000C
NOT  EXC error  00050+00003
TRANS 1D error  00200+00000  rfd by  0005C+00002
R:I      error  00200+00000  COMB-TYP
DO     4 error  00200+00002  0000000I
INDEX    error  00301+00004  -
DO     6 error  00401+00000
FM120.0V error  00002+00000

MAIN PROGRAM

VARIABLE
    B5    R  S  1
    KK    I  S  0
    B4    R  S  1
     M    I  S  0
    B3    R  S  1
   B21    R  S  1
     T    R  S  0   COMMON              UNDEFINED
     L    I  S  0
    B2    R  S  1
    C1    R  S  1
     K    I  S  0
    A2    R  S  0   COMMON
    B1    R  S  1
   PAI    R  S  0
     J    I  S  0
    A1    R  S  0   COMMON
     B    R  S  1
     I    I  S  0                  P
    B6    R  S  0

SUBPROGRAM
    SS    SR              P  2
     C    RF  S           P  1

BLOCK NAME
MTSEEN

Undef.St. No.
   20
  201
 Defd.St. No.
    3      30 ?    100      40      50      15      101      200      35
  301      501      401       2
END OF CHECK
```

```
*         HARP
      DIMENSION B21(1000),C1(1000),B1(1000),B1(1000),B2(1000),
     1B3(1000),B4(1000),B5(1000),,C(1000)
      COMMON /MTSEEN/T,A1,A2,
      READ(5,3) A1,A2
    3 FORMAT(2F10.8)
      DO 100 I=1,1000
      IF(I.EQ.1000.) GO TO 30
      B21(I)=FLOAT(I)/101.*(-1.)**I*(A1/A2)**I
      GO TO 100
   30 B21(I)=-1./101.*(-1.)**I*(A1/A2)**I
  100 CONTINUE
      C1(1)=B21(1)
      C1(2)=B21(2)/2.
      DO 50 I=3,1000
      M=(I+1)/2
      C(I)=B21(I)
      K=2
      KK=I
      DO 40 J=2,M
      B6=C(I)*(FLOAT(K),/FLOAT(I-1))
      C1(I)=B6*FLOAT(I)/FLOAT(I)**C1(1)
      K=K+2
   40 KK=KK-1
   50 C1(I)=C1(I)/I.**I
      CALL SS(B21,C1)
      IF(T) 15,20,200
      PAI=3.1415926
   15 B(1)=T
      B(2)=T*T
      DO 101 I=3,1000
  101 B(I)=B(I-2)*4.*FLOAT(I-1)/FLOAT(I)
      DO 401 K=1,1000
      DO 201 I=1,999
  200 B1(I)=(101.*B(I)/2.**I-1.+(-1.)**I*(101.*B(I)/2.**I-1))/2.-
     1(COS(PAI/101.))**I
      B2(1)=B1(1)
      DO 301 I=2,999
      B2(I)=B1(I)/FLOAT(I)*(-1.)**I
      DO 35 J=1,I
      L=I-J
   35 B2(I)=B2(I)+(-1.)**J*B2(J)*B1(L)*(-1.)**I/FLOAT(I)
  301 CONTINUE
      B3(K)=C1(1000)
      B4(K)=(COS(PAI/101.))**99
      DO 501 J=1,999
      B3(K)=B3(K)+C1(1000-J)*B2(I)*(-1.)**I
      B4(K)=B4(K)+(-1.)**I*(COS(PAI/101.))**99*B2(I)
  501 CONTINUE
      B5(K)=B3(K)/B4(K)/SIN(PAI/101.)
  401 CONTINUE
      WRITE(6,2)(B1(J),B2(J),B3(J),B4(J),B5(J),J=1,1000)
    2 FORMAT(1H ,5E25.7)
      GO TO 20
:     END
```

いる.

(9) Index 添字にまちがいがある.

(10) DO 6 DO ループの nesting のエラー

(11) FM 120. OV 印字される字が，120 をこえた.

（昭和 42 年 4 月 8 日受付）