誌上討論

ブロック構造処理方法についての二，三の注意**

野崎昭弘**

コンパイラによるブロック構造の処理法について，筆者はさきに投稿した論文の一部で，その一例に言及したが，これに対して小島氏および石内氏が批評を寄せられ，別の方法を提示された***.

筆者がこれら二つの方法を検討したところ，いずれも一長一短で，どちらが特に優れているといえないもののように思われたが，小島・石内両氏の文面では問題点が明示されていないように思われるので，少し再論を述べてみたい．

小島・石内両氏の方法の要点は，原文をかりれば次のとおりである．

「テーブルの索引時間を最小にするためには，必要最小限の identifier をテーブルにのせておくこと，かつ索引すべきテーブルは分割しないことなどが必要である」

「I という identifier が宣言されたとき，IT（テーブル）を調べて同じ identifier があるかどうかをみる．あれば，そのコードワードを CWT から取り出して ACT に移し，新しいコードワードをその場所に入れる．同じ identifier が無ければ，I とそのコードワードを IT と CWT の最後に入れる」

ここでコードワードとは，その identifier に，特定ブロック内で結びつけられている情報のことである．IT と CWT とが筆者の SVT に相当する（SVT には，すべての identifier が登録される）．ACT は使用されないコードワードを収容するための，一時記憶場所である．

IT と CWT とを分割するのは技術的な理由で，本質的な問題ではない．比較の要点は，「必要最小限の identifier をテーブルにのせておく」ための処置を，とる（小島・石内両氏）か，とらない（筆者）かということである．以後，これらの方法をこの点でだけ区別して，それぞれ方法 I および方法 II とよぶことにする．

方法 I は，「identifier の登録あるいは消去に時間がかかっても，そのテーブルの索引に要する時間を最小にすることが全体として能率的である」という仮定に立てば，コンパイル時間の短縮をもたらす．実計算時間についてはもちろん，どちらでも同じであり，記憶場所については，記憶すべき情報の総量は同じなのであるから，同等と考えられる．（ワードの構成，identifier に許す有効字数，および 'LT' に入るべき情報量によって，計算機にもよって，一方が他方より有利または不利であることは起り得ると思うが，それは些細なことのように思う）

問題は上記仮定がどの程度正しいか，ということである．これは理論的な問題ではなくて，経験的な問題あるから，一義的な結論を出すのはむずかしいと思う．そこで，本誌**プログラムのページ**にのせられたいくつかのプログラムを素材にして，少し粗末な仮定の検定を行なってみたい．

方法 I が有利になるのは，「親ブロックとサブブロックとで，同じ identifier が，別の意味で使用されることが多いばあい」である*．そうでなければ，全体のテーブルのサイズは変らず，「identifier の既・宣言かどうかの検査，登録，消去」の手順が面倒である方法 I の方が，不利になる．そこで，プログラム中の変数の数と，サブブロックの中で re-declare された変数の数をかぞえてみたところ，第 1 表のようになった．

このような少数の事例からすべてを類推することは

第 1 表

プログラム番号	6216*	6217	6301*	6304	6305
変数宣言の総数	22	38	11	24	7
再宣言の回数	0	7	0	4	不明**

* ……ブロック構造をもっていない

** ……procedure だけが記されていて，その外側がわからないので，重複度不明（それ自身の中では，再宣言回数は 0 である）

* スコープの交わらないブロック同士では，二重に宣言されていても処理上の問題点はない．SVT の中でも，一時に一つしかあらわれない．

* Some Remarks on the Methods for Processing the Block Structure, by Akihiro Nozaki (Tokyo University)

** 東京大学教養学部基礎科学科

*** 小島博，石内祥介：ブロック構造の処理プログラムについて，情報処理，Vol 4, No 2, p 38,

もちろんできないが，少なくとも，これらの例に関する限り，方法Ⅰがプログラミングに手をかけただけのmeritを挙げうるかどうかは疑問である．定量的な比較はできないが，方法Ⅱの方が早い例も存在する！

次に，「テーブルが小さい」ということの意味を反省してみよう．方法Ⅱでは，テーブルは大きくなるが，その代り「最後のブロックで宣言された変数は，つねにテーブルの最後（終りから引くので，一番引きやすい位置）にある」ということが，re-declareされた変数（テーブルを大きくした原因）についてもいえる．したがって，もし「最後のブロックでdeclareされた変数は，そのブロック内では，最も頻繁に利用される」ことを仮定するならば，テーブルが大きくなったdemeritは，“引き易い位置にある”というmeritによって，カバーされる，ということも考えられる．

そこで，プログラム中のinnermost blocksについて，変数が引用される回数と，最後のブロックで宣言された変数が引用される回数とを数えてみた（第2表）

第2表

プログラム番号	6217	6304	6305	6306B
変数引用の総数	79	75	14	30
ブロック内変数の引用回数	63	33	12	14

引用の5割内外は，そのブロック内で宣言された変数の引用である．したがって，全体のテーブルを最小にすることの効果も，少し割引して考えなければなるまい．

この程度の差異であるならば，方法Ⅰ，Ⅱの選択は，コンパイラ全体の設計方針から決定されるべきではなかろうか．コンパイラの構造を単純化するために，できるだけサブルーチンを共用するという立場から，強い要求（指定）がなされるのはあり得ることである．これ以上の比較検討は，私としては，読者諸賢に一任したい．