# 小記憶容量計算機用BASICプログラムの実行速度向上法

永　田　昭　三

(受理　昭和57年5月31日)

# IMPROVEMENT OF THE EXECUTION SPEED OF BASIC PROGRAM

Shozo NAGATA

In BASIC programs running on small memory-size personal computers, variables and statements are organized into files of which records are accessed sequentially.

The program execution speed is improved by means of rearrangement of records:

1) variables accessed frequently are addressed at the top of file,
2) subroutines are placed before the main program.

An example runs about 40% faster.

## 1. 緒　　言

半導体素子の性能が向上し，価格が下がった結果，8ビットの小形計算機が急速に普及している．使用言語はBASICである．これの特長の1つは記憶容量が小さくても使えることである．

記憶装置の番地は16ビットで表現され，0番地から65535番地までを扱い得る．その半分程度の記憶容量しか持たない計算機でもBASICは使用可能である．しかし，そのような計算機では，プログラムを解読実行するインタープリタに多くの記憶容量を割り当てることができず，その性能に制約が生ずる．すなわち，プログラムの作り方によって実行速度が速くなったり，おそくなったりする．

本報告ではデータ領域の構造とプログラムの構造を調べ，実行速度を速くする方法を述べる．

## 2. 変数への番地割り当て

### 2.1 実行時間の測定

図1-1のプログラムの実行時間を測る．ディスプレイにSTARTが出てからENDが出るまでの時間を測る．その時間は行番号100の文を2000回繰り返す時間であり，その時間内に変数A0の番地を10000回探さねばならない．所要時間は18.2秒であった．

```
 10  REM VARIABLES
 20  PRINT "START"
 30  LET A0=0
 40  LET A1=0
 50  LET A2=0
 60  LET A3=0
 70  LET A4=0
 80  LET A5=0
 90  FOR I=1 TO 2000
100  LET A0=A0+A0+A0+A0
110  NEXT I
120  PRINT "END"
130  END
```

図1-1　変数を探す時間を測るプログラム

```
100  LET A5=A5+A5+A5+A5
```

図1-2　プログラムの変更

次に行番号100の文を図1-2に示す文に書き直して所要時間を測ったら，22.9秒を要した．変数A5を探すのに要する時間がA0を探す時間よりも長かった．

### 2.2 データ領域の構造

このBASICのプログラム中では，変数名はその文字に相当するコードに変換されて記憶され，それに割り当てられる番地は不明である．

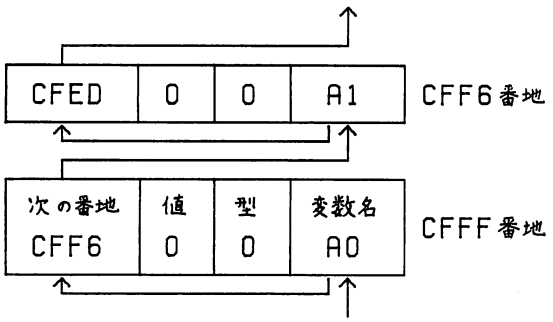


図2　変数の構造と配置

変数の値は記憶装置の末尾，例えば CFFF 番地，から前に向かって格納される．格納の順序は，プログラムを実行した場合にプログラム中に現れる順である．図1-1のプログラムでは，変数への番地割り当ての順序は A0, A1, A2, A3, A4, A5, I の順である．図2は下から上に見て行く．

変数 A5 を探したければ，まず1番目の変数名を調べる．これは A0 であり，A5 でない．「次の番地」の位置に CFF6 と書いてある．これを見て CFF6 番地の変数名を取り出して調べる．以下，矢印の順に調べて行くことにより，欲しい変数が見つかる．

このようにデータを始めから順に探すファイルは順編成ファイルと呼ばれる．この BASIC のデータ領域は順編成ファイルである．

### 2.3 最適化

順編成ファイルの先頭近くにある程，データを探す時間は短い．従って使用回数の多い順に変数を並べたら，プログラムの実行時間は最も短くなる．そのための方法として，プログラムを作成した後に各変数について使用回数をかぞえ，回数の多い順に図1-1の行番号30-80のような文をプログラム本体の前に書き加える．使用回数が多い変数とは FOR-NEXT ループ中で使われる変数および配列である．それ以外の変数については番地割り当ての順番を考慮しても効果は少ない．

## 3. 主プログラムと副プログラムの前後関係

### 3.1 実行時間の測定

行番号を探す時間を測るプログラムを図3-1に示す．このプログラムでは行番号40を2000回探す時間が

```
10  REM GOSUB
20  PRINT "START"
30  GOTO 50
40  RETURN
50  FOR I=1 TO 2000
60  GOSUB 40
70  NEXT I
80  PRINT "END"
90  END
100 RETURN
```

図3-1　行番号を探す時間を測るプログラム

```
60  GOSUB 100
```

図3-2　プログラムの変更

測定される．所要時間は 13.2秒であった．

次に行番号60の文を図3-2のように訂正したら，時間は 15.2秒であった．行番号がプログラムの先頭に近い程，短い時間で探し出される．

### 3.2 プログラムの構造

命令 GOSUB に続く行番号はプログラム中では定数の形式で記憶され，番地を指さない．

プログラムはユーザ領域の先頭から後へ後へと格納される．ある行番号を探すときには，まずプログラムの先頭の行番号を調べる．違っていればその前の「文の字数」を用いて，次の行番号の番地を算出する．以下，矢印の順に行番号を探して行く．プログラムも順編成ファイルである．

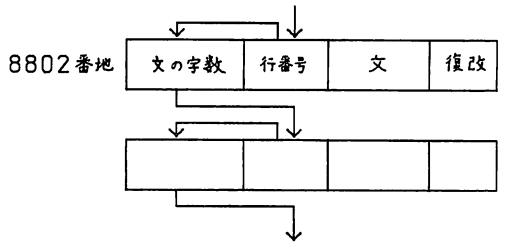


図4　文の構造

### 3.3 最適化

行番号も順編成ファイルを構成しているので，引用回数の多い行番号をプログラムの先頭近くに置いたら，それを探す時間が短くなる．つまり，副プログラムを主プログラムの前に置く．副プログラムが複数個あれ

ば，引用回数の多いものを前に置く．

## 4. 総　　括

以上の説明により，実行時間を短くするためのプログラムの構成順序は次のとおりになる．

1. 主プログラムへの GOTO 文
2. 副プログラム
   - 2.1 使用回数の多い副プログラム
   - 2.2 使用回数の少い副プログラム
3. 主プログラム
   - 3.1 変数への番地割り当ての順番を決めるための文
   - 3.2 主プログラム本体

プログラムの実行時間の例を示す．最小2乗法の計算を12回行うプログラムを作った．まずFORTRANにならって副プログラムを主プログラムの後に置いた．変数への番地割り当ての順番を考慮しなかった．次いで前述の最適化を行った．実行時間を表1に示す．時間は40%短くなった．

表1　実行時間の例

| 方　　法 | 実行時間(秒) | 短縮(秒) |
|---|---|---|
| FORTRAN 類似 | 40.5 | |
| 変数への番地割り当て | 30.1 | 10.4 |
| 副プログラムの位置 | 24.5 | 5.6 |
| | | 計16.0 |

## 5. 結　　論

市販の BASIC の中には，変数と行番号が順編成ファイルを構成しているものがある．その場合には，引用回数の多い変数や行番号をファイルの先頭近くに置くことにより，プログラムの実行時間を短くすることができる．