第3章　演算と転記

指導目標

データの型によりそのデータを扱う命令（文）があり、データを数字データと文字データに分類して扱う上での基礎知識を学習する。

本章では、プログラミング技法のデータ処理方法として、計算（四則演算）とデータ操作について学習する。

計算では四則演算について演算子の優先順位や四捨五入などが使いこなせるようにする。データ操作については転記や表操作が習熟できるように指導する。また、文字列操作については文字列データの検査や分解、結合操作の概要を理解させる。

本章で解説する命令文、は次のとおり。

・ＣＯＭＰＵＴＥ文
・ＡＤＤ文
・ＳＵＢＴＲＡＣＴ文
・ＭＵＬＴＩＰＬＹ文
・ＤＩＶＩＤＥ文
・ＭＯＶＥ文
・ＳＥＡＲＣＨ
・ＩＮＳＰＥＣＴ
・ＳＴＲＩＮＧ
・ＵＮＳＴＲＩＮＧ

内容のあらまし

内　　容	説　明	議　論	机上実習	計算機実習
算術文	算術文の概要と演算子の機能、四捨五入、桁あふれなど四則演算上の注意事項を説明する。	一般的な表現とﾌﾟﾛｸﾞﾗﾐﾝｸﾞでの記述について議論する。		
ＣＯＭＰＵＴＥ文	COBOLでの算術式について実行順序を中心に説明する。			四捨五入と結果の編集を実習する。
ＡＤＤ文	加算については主に記述形式と機能について説明する。			
ＳＵＢＴＲＡＣＴ文	減算については主に記述形式と機能について説明する。			
ＭＵＬＴＩＰＬＹ文	乗算については主に記述形式と機能について説明する。			
ＤＩＶＩＤＥ文	除算については記述形式と余りや商について説明する。		DIVEDEを使用しない場合の余りの求め方を実習する。	余りの求め方を実習する。
ＭＯＶＥ文	転記の機能と編集操作について説明する。			ﾃﾞｰﾀ型の異なる項目間の転記を実習する
ＳＥＡＲＣＨ文	表操作について逐次探索、非逐次探索を説明する。	どのような処理に有効なのか議論する。		SEARCHが使いこなせるように実習する。
文字列操作	文字列の表現形式と文字列操作の命令文について説明する。		INSPECTが使いこなせるようにする。	STRING、UNSTRINGを使用して簡単なﾊﾟﾗﾒｰﾀ解析を実習する。

第３章　演算と転記

３．１　算術文

ＣＯＢＯＬの算術文には、ＡＤＤ（加算）、ＣＯＭＰＵＴＥ、ＤＩＶＩＤＥ（除算）、ＭＵＬＴＩＰＬＹ（乗算）、ＳＵＢＴＲＡＣＴ（減算）がある。

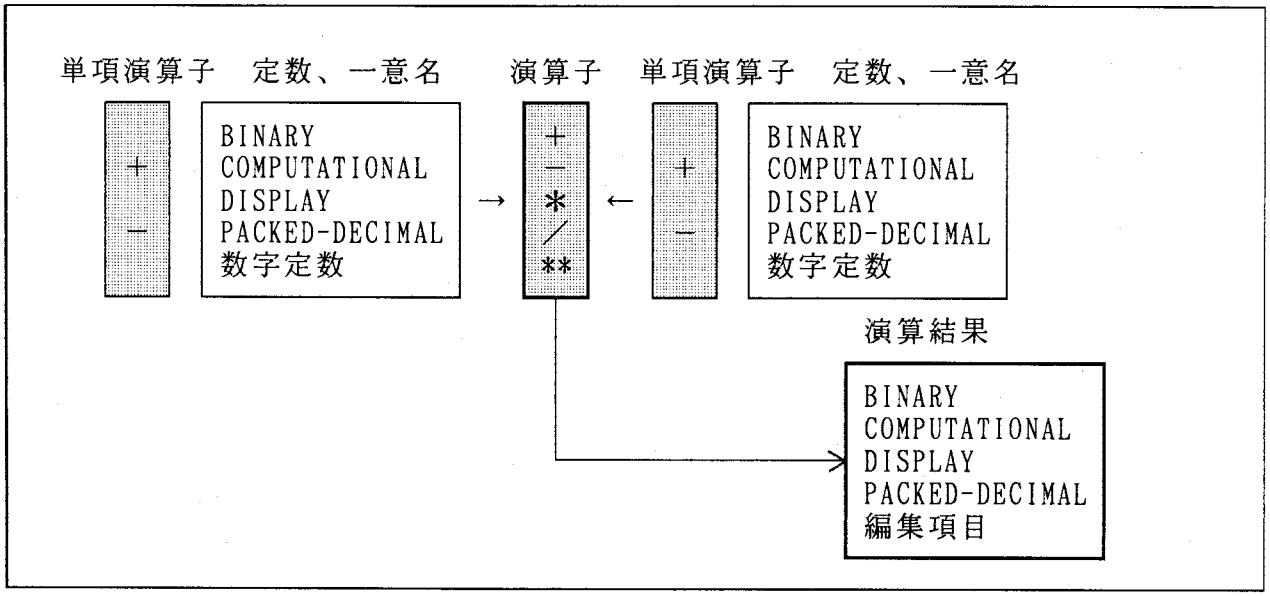
Ｃ言語には簡潔な記述が用意されているが、ＣＯＢＯＬでも効率的な記述ができる。

	ＣＯＢＯＬ記述	Ｃ言語記述
① 加 算	COMPUTE CNT = CNT + 3 ADD 3 TO CNT ADD 4 5 TO CNT —	cnt = cnt + 3 cnt += 3 cnt += 4+5 ++cnt
② 減 算	COMPUTE CNT = CNT - 4 SUBTRACT 4 FROM CNT SUBTRACT 5 6 FROM CNT —	cnt = cnt - 4 cnt -= 4 cnt -= 5+6 --cnt
③ 乗 算	COMPUTE CNT = CNT * 5 MULTIPLY 5 BY CNT —	cnt = cnt * 5 cnt *= 5 cnt *= 6+7
④ 除 算	COMPUTE CNT = CNT / 6 DIVIDE 6 INTO CNT —	cnt = cnt / 6 cnt /= 6 cnt /= 7+8
⑤	COMPUTE CNT = CNT ** 7	—
⑥	DIVIDE 100 INTO CNT GIVING YYY REMAINDER XXX	xxx = cnt % 100
⑦	— — — — —	cnt <<= 8 cnt >>= 9 cnt &= 10 cnt ^= 11 cnt |= 12

算術演算のデータ項目の最大桁数は、１８桁である。

```
  123456789012345678                          ←整数部のみ18桁
+                    .123456789012345678      ←小数部のみ18桁
-------------------------------------------
  123456789012345678.123456789012345678      ←中間結果
                     ↓
          結果は最大18桁   整数部、小数部が切り捨てられて格納される。
```

データ形式が異なっても自動的に変換され、小数点位置を考慮して算術演算を行い、結果のデータ形式に変換され格納する。



算術演算子は次の通り。
算術演算子の前後には、空白等の分離符をおく。

単項演算子	意　味	
＋	数字定数の＋１を掛けることと同じ。	+100
－	数字定数の－１を掛けることと同じ。	-（データ名）

２項演算子	意　味				一般的な表現
＋	加算	ﾃﾞｰﾀ名	＋	ﾃﾞｰﾀ名	＋
－	減算	ﾃﾞｰﾀ名	－	ﾃﾞｰﾀ名	－
＊	乗算	ﾃﾞｰﾀ名	＊	0.03	×
／	除算	ﾃﾞｰﾀ名	／	100	÷
＊＊	べき乗	ﾃﾞｰﾀ名	＊＊	2	

演算の優先度

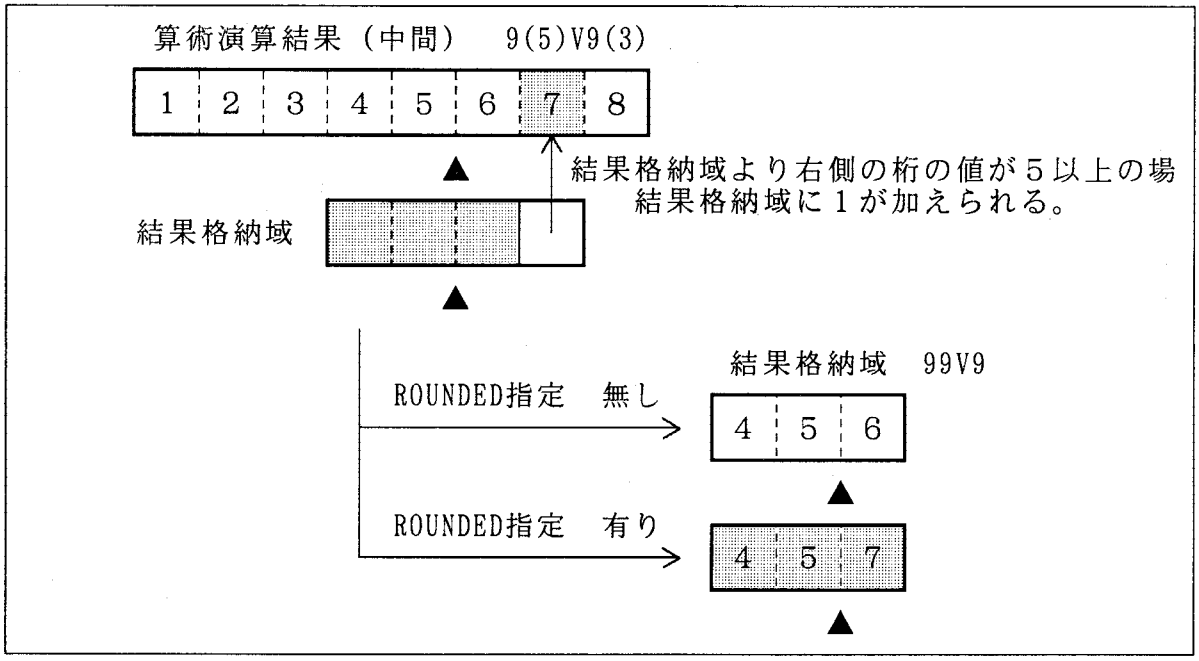
順　位	意　味
1	かっこ内
2	単項演算子（＋、－）
3	べき乗（＊＊）
4	乗算または除算
5	加算または減算

同一順位の場合の計算は左から右へ行われる。

３．１．１　四捨五入(ROUNDED)

算術演算（ADD、COMPUTE、DIVIDE、MULTIPLY、SUBTRACT）でＲＯＵＮＤＥＤ句を指定すると結果格納域より演算結果（中間）の小数部の桁数が大きい場合に四捨五入が行われる。

```
ADD       一意名1  ･･･  TO    一意名2  ROUNDED
COMPUTE   一意名1  ROUNDED  =  算術式
DIVIDE    一意名1  INTO  一意名3  ROUNDED
MULTIPLY  一意名1  BY    一意名2  ROUNDED
SUBTRACT  一意名1  ･･･  FROM  一意名2  ROUNDED
```



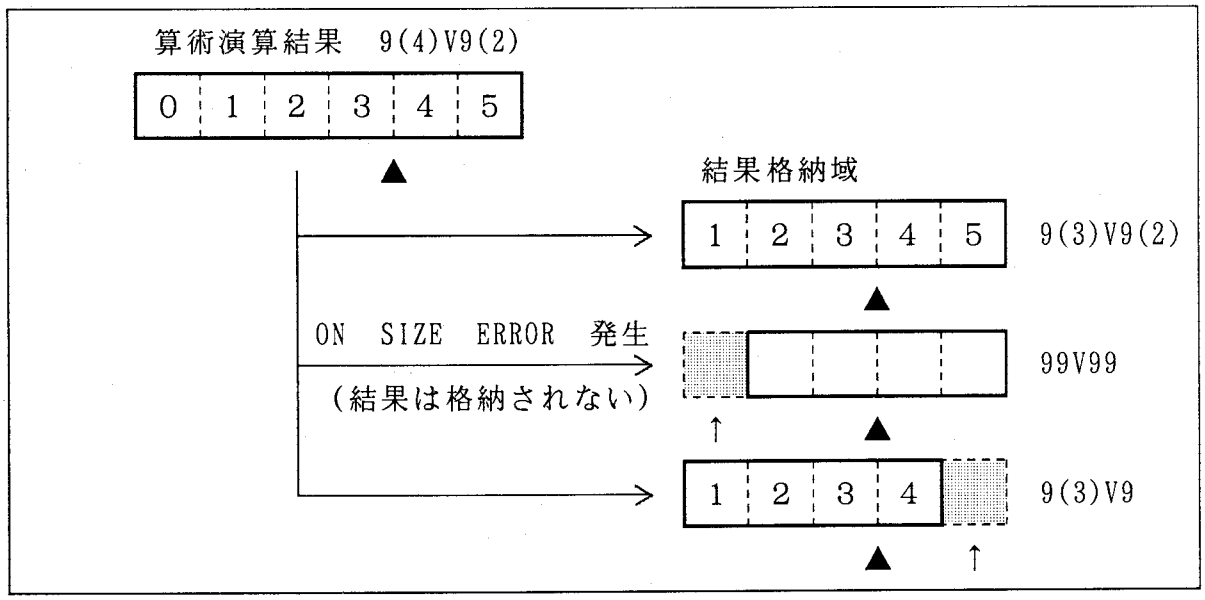
３．１．２　桁あふれ（ON　SIZE　ERROR）

算術演算（ADD、COMPUTE、DIVIDE、MULTIPLY、SUBTRACT）でON　SIZE　ERROR句を指定すると、算術演算結果の値が結果格納域の最大値を越える場合、また、ゼロの除算の場合に桁あふれ条件が発生する。計算途中の中間結果では無視される。

ON SIZE ERROR句の指定がない場合で、桁あふれが発生すると結果についての規定はないが、通常、切り捨てられて格納される。

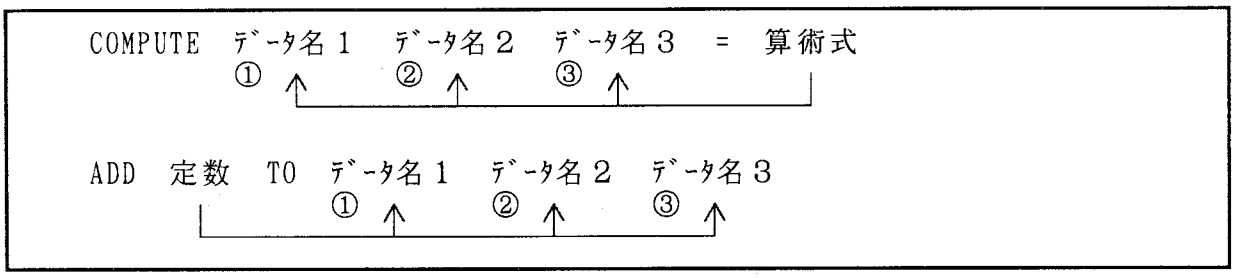
```
DATA  DIVISION.
    :
WORKING-STORAGE  SECTION.
01  MAXIMUM    PICTURE  S9  VALUE  8.      ←

PROCEDURE  DIVISION.
    :
    COMPUTE  MAXIMUM  =  MAXIMUM  +  1      ←
    ON SIZE ERROR  DISPLAY  "桁あふれ発生".
    COMPUTE  MAXIMUM  =  MAXIMUM  +  1      ←
    ON SIZE ERROR  DISPLAY  "桁あふれ発生".
```

MAXIMUMの状態		
前	後	SIZE ERROR
8	－	－
8	9	発生しない
9	9	発生 表示の実行



３．１．３　複数の結果格納

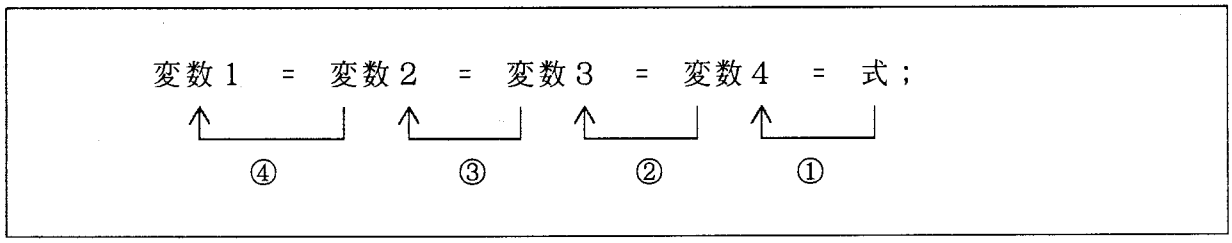
COMPUTE文などの算術文には、結果を複数個のデータ項目に格納することができる。



文の実行は、初期評価としてデータ項目に必要な演算を行って、その結果を一時記憶領域に入れておく。その後、一時記憶領域から結果を入れるデータ項目へ移す。動作の順序は、左から右に書かれたものと同じとする。

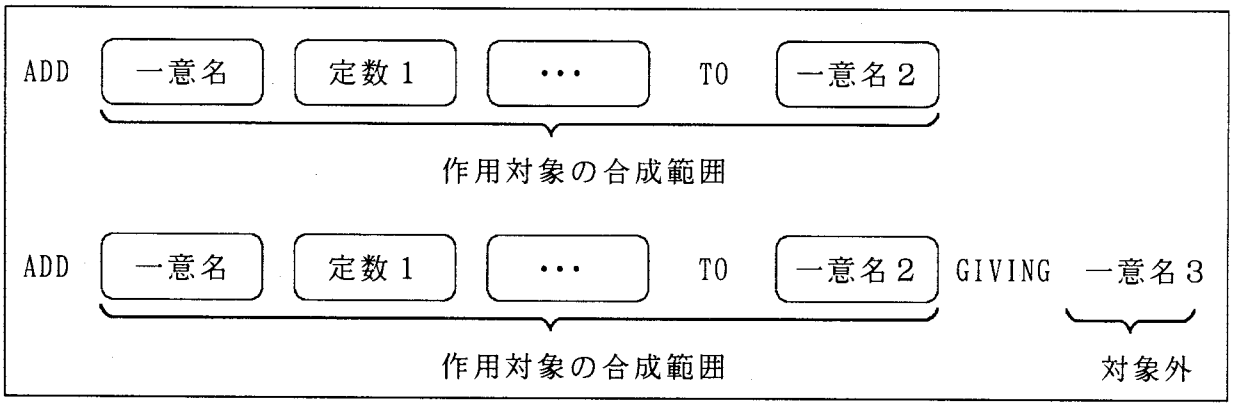
```
ADD  A  B  TO  X  C (X)

    ①  ADD  A  B  GIVING  ﾃﾝﾎﾟﾗﾘ領域
    ②  ADD  ﾃﾝﾎﾟﾗﾘ領域  TO  X
    ③  ADD  ﾃﾝﾎﾟﾗﾘ領域  TO  C (X)

COMPUTE  X  C (X)  =  A  +  B

    ①  A＋B      ﾃﾝﾎﾟﾗﾘ領域
    ②  ﾃﾝﾎﾟﾗﾘ領域  →  X
    ③  ﾃﾝﾎﾟﾗﾘ領域  →  C (X)
```

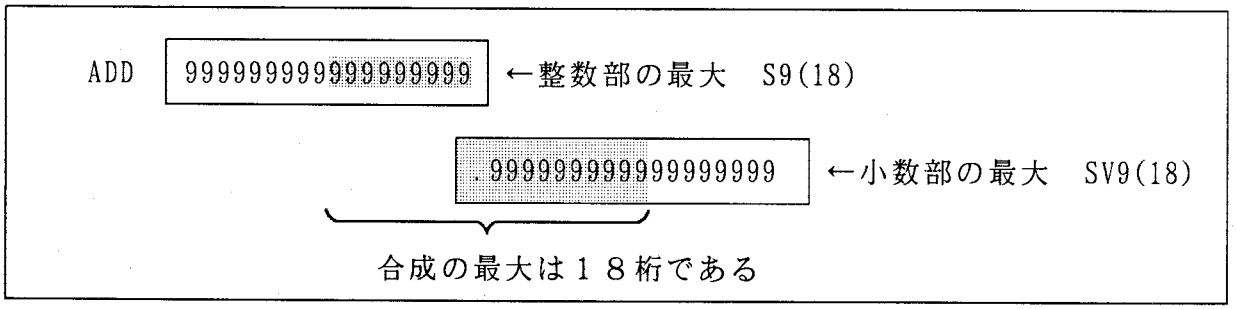
Ｃ言語でも同様の記述はできるが、格納動作はＣＯＢＯＬと異なる。



３．１．４　作用対象の合成

ＡＤＤ、ＤＩＶＩＤＥ、ＭＵＬＴＩＰＬＹ、ＳＵＢＴＲＡＣＴ文では、作用対象の最大桁数を１８桁とする。
また、作用対象の合成桁数も１８桁以内とする。





COMPUTE文などで複雑な算術式が書かれると、一時的に演算結果を保持するために中間結果が求められることになる。ＪＩＳ－ＣＯＢＯＬでは、その場合の方式と精度は各コンパイラで規定することになっている。

```
        COMPUTE  結果  =  A＋B＊C／D

計算順序     ①B   ＊   C              →  中間結果１
             ②中間結果１   ／   D     →  中間結果２
             ③A   ＋   中間結果２     →  結果
```

例（概要）として、中間結果の桁数は、次のように決定される。

整数部１．小数部１　（演算）　整数部２．小数部２

演算	整数部の桁数	小数部の桁数
＋、　－	どちらか大きい方＋１	どちらか大きい方
＊	整数部１＋整数部２	小数部１＋小数部２
／	整数部１＋小数部２	小数部２－小数部１

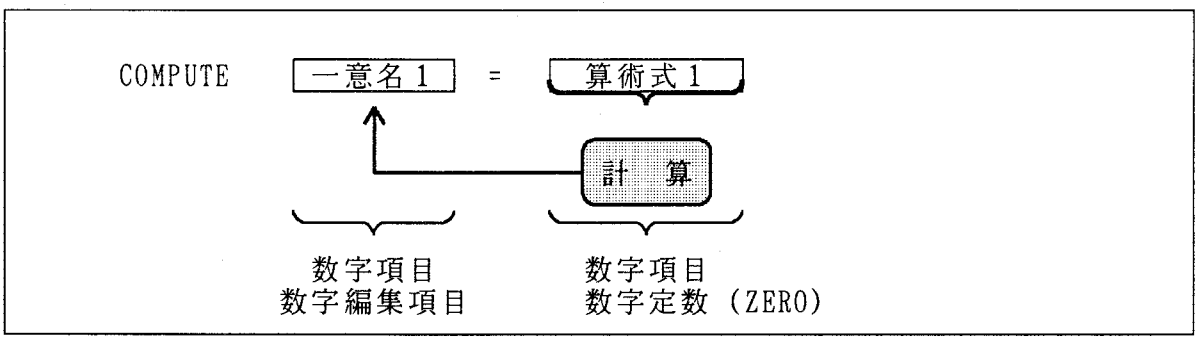
複雑な算術式、桁数の大きなデータ項目などでは、途中で中間結果の一部が失われることもあるが、コンパイラは警告メッセージを出して注意を促す。
これらに対処するには、むやみに大きな桁数にしないようにすることが重要である。

３．２　ＣＯＭＰＵＴＥ

ＣＯＭＰＵＴＥ（計算）文は、算術式の結果を一つ以上のデータ項目に格納する。

```
COMPUTE   {一意名1   [ROUNDED] } ･･･   =   算術式1

          [ON  SIZE  ERROR  無条件文1]
          [NOT  ON  SIZE  ERROR  無条件文2]
 [END-COMPUTE]
```

結果を入れる一意名は、数字項目または数字編集項目でなければならない。



算術式の算術演算子には、２項演算子と単項演算子がある。
算術演算子の前後には、空白をおかなければならない。

演算子	使　用　例	COMPUTE以外の算術文
加　算	COMPUTE　A　=　B　+　C	ADD　B　TO　C　GIVING　A
減　算	COMPUTE　A　=　B　-　C	SUBTRACT　C　FROM　B　GIVING　A
乗　算	COMPUTE　A　=　B　*　C	MULTIPLY　B　BY　C　GIVING　A
除　算	COMPUTE　A　=　B　/　C	DIVIDE　B　BY　C　GIVING　A
べき乗	COMPUTE　A　=　B　**　C	—

ＣＯＢＯＬの算術式とは、次のとおり。

①数字項目の一意名、数字定数、表意定数（ZERO）
```
COMPUTE  ﾃﾞｰﾀ名  =  6.2.
```

②一意名、定数、表意定数を算術演算子（+、-、*、/、**）でつないだもの
```
COMPUTE  ﾃﾞｰﾀ名  =  ﾃﾞｰﾀ名  *  0.03.
```

③２個の算術式を算術演算子でつないだもの
```
COMPUTE  ﾃﾞｰﾀ名  =  ZERO  -  ﾃﾞｰﾀ名
```

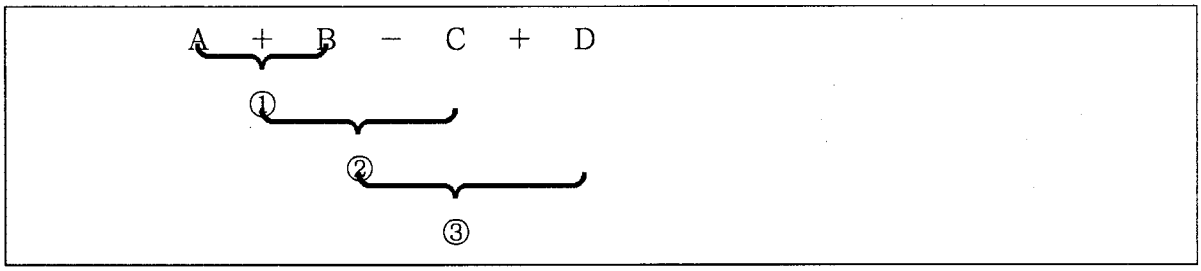
④算術式を括弧で囲んだもの
```
COMPUTE  ﾃﾞｰﾀ名  =  ﾃﾞｰﾀ名  +  (ﾃﾞｰﾀ名  +  0.5)
```

⑤算術式の前に単項演算子（-、+）を付けたもの
```
COMPUTE  ﾃﾞｰﾀ名  =  -(ﾃﾞｰﾀ名  *  +0.03)
```

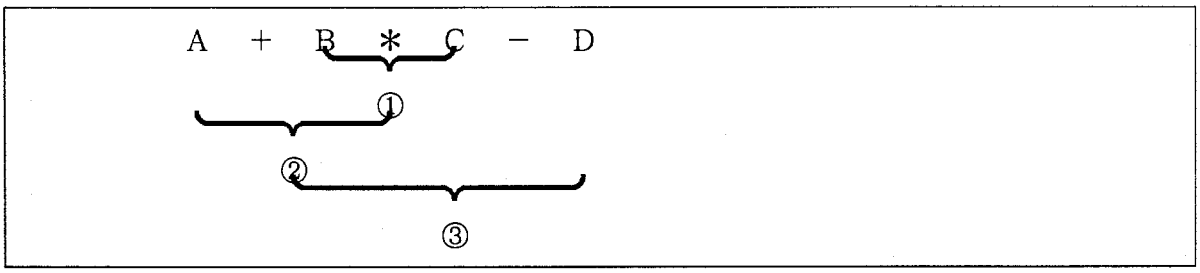
算術式中の一意名や定数は、数字項目や数字定数でなければならない。

算術式の評価規則（実行順序）は、次のとおり。

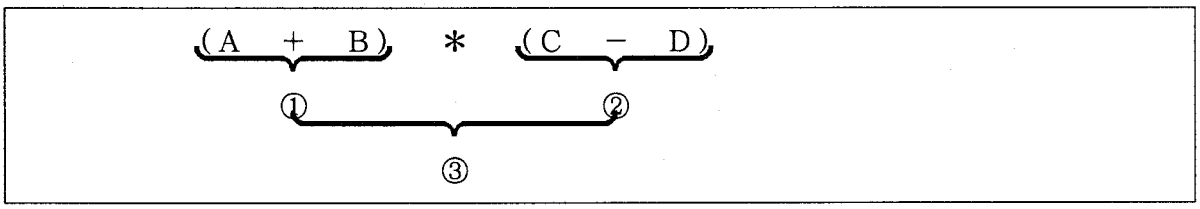
同じ順位の演算子が続いている場合、評価は左から右へ実行する。



異なる順位の演算子が使われている場合、評価は演算子の順位に従う。



括弧は、同じ順位や異なる順位の間で評価順序を変更する場合に使用する。



算術式で許される一意名、定数、演算子、括弧の組合せは、次のとおり。

左　側 ＼ 右　側	一意名　定数	(	単項演算子	２項演算子	)
単項演算子	○	○	×	×	×
(	○	○	○	×	×
２項演算子	○	○	○	×	×
)	×	×	×	○	○
一意名　定数	×	×	×	○	○

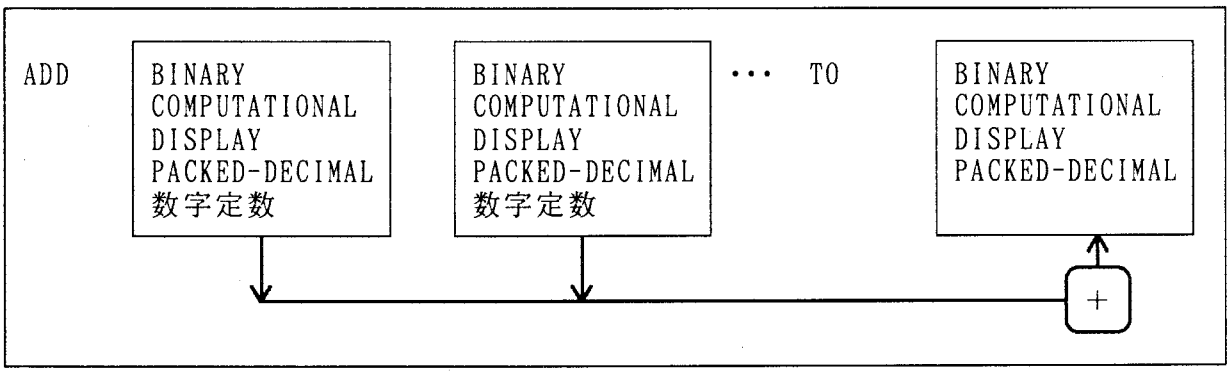
○は組合せが可能、×は組合せが不可。

３．３　ＡＤＤ

ＡＤＤ（加算）文は、いくつかの数字作用対象（数字項目、定数）の和をとって、その結果を一つ以上のデータ項目に格納する。

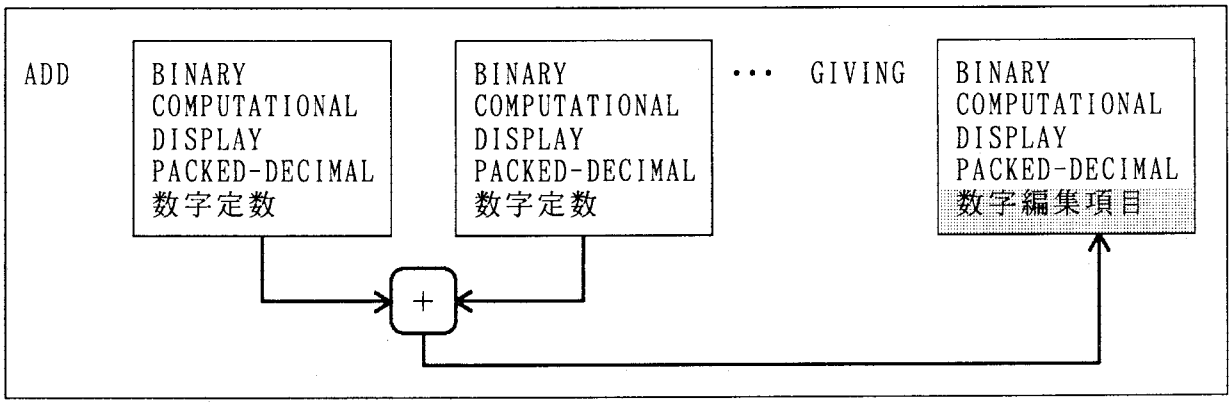
```
ADD { 一意名1 }
    { 定数1   }  ･･･  TO  {一意名2  [ROUNDED] }  ･･･
      [ON SIZE ERROR 無条件文1]
      [NOT ON SIZE ERROR 無条件文2]
[END-ADD]

ADD { 一意名1 }          { 一意名2 }
    { 定数1   }  ･･･  TO  { 定数2   }
      GIVING  {一意名3  [ROUNDED] }  ･･･
      [ON SIZE ERROR 無条件文1]
      [NOT ON SIZE ERROR 無条件文2]
[END-ADD]

ADD { CORRESPONDING }  一意名1 TO 一意名2 [ROUNDED]
    { CORR          }
      [ON SIZE ERROR 無条件文1]
      [NOT ON SIZE ERROR 無条件文2]
[END-ADD]
```

「ＴＯ」指定では、ＴＯの前にある数字項目または定数の値を加えてから、さらに、その和をＴＯの後の一意名に加える。



「ＧＩＶＩＮＧ」指定では、ＧＩＶＩＮＧの前にある数字項目または定数の値を加えてから、その和をＧＩＶＩＮＧの後の一意名に入れる。

「ＧＩＶＩＮＧ」指定では、結果を入れる一意名に数字編集項目が使える。

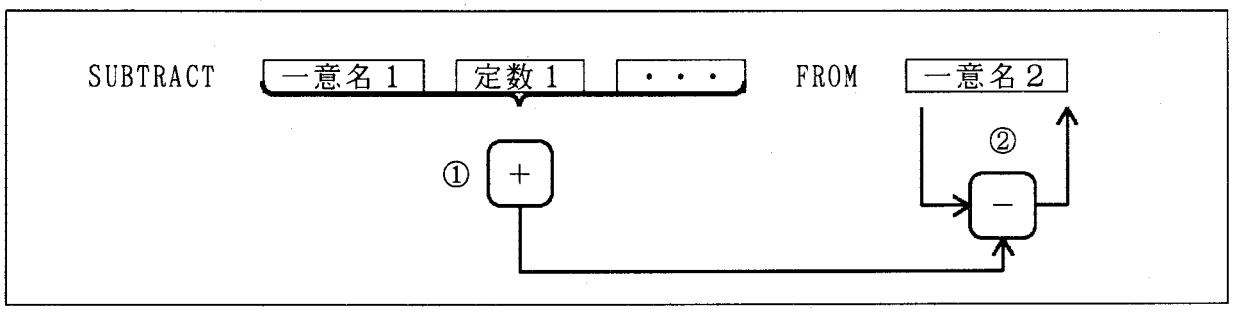


３．４　ＳＵＢＴＲＡＣＴ

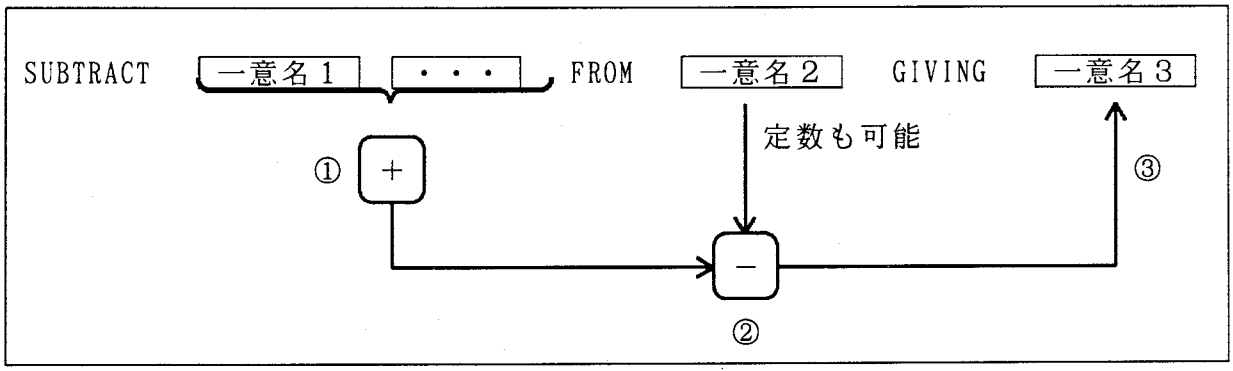
ＳＵＢＴＲＡＣＴ（減算）文は、いくつかの数字作用対象（数字項目、定数）の和を、データ項目から減じ、その結果をデータ項目に収める。

```
SUBTRACT { 一意名1 } ･･･ FROM {一意名2 [ROUNDED] } ･･･
         { 定数1   }
         [ON SIZE ERROR 無条件文1]
         [NOT ON SIZE ERROR 無条件文2]
 [END-SUBTRACT]

SUBTRACT { 一意名1 } ･･･ FROM { 一意名2 }
         { 定数1   }          { 定数2   }
                    GIVING {一意名3 [ROUNDED] } ･･･
         [ON SIZE ERROR 無条件文1]
         [NOT ON SIZE ERROR 無条件文2]
 [END-SUBTRACT]

SUBTRACT {CORRESPONDING} 一意名1 FROM 一意名2 [ROUNDED]
         {CORR         }
         [ON SIZE ERROR 無条件文1]
         [NOT ON SIZE ERROR 無条件文2]
 [END-SUBTRACT]
```

ＦＲＯＭの前までの作用対象の和をもとめてから、その和をＦＲＯＭの後の数字項目から減算する。そして減算結果をＦＲＯＭの後の数字項目へ入れる。



「ＧＩＶＩＮＧ」指定では、ＦＲＯＭの前までの作用対象の和をもとめてから、その和をＦＲＯＭの後の数字項目から減算する。そして減算結果をＧＩＶＩＮＧの後の数字項目（数字編集項目）へ入れる。

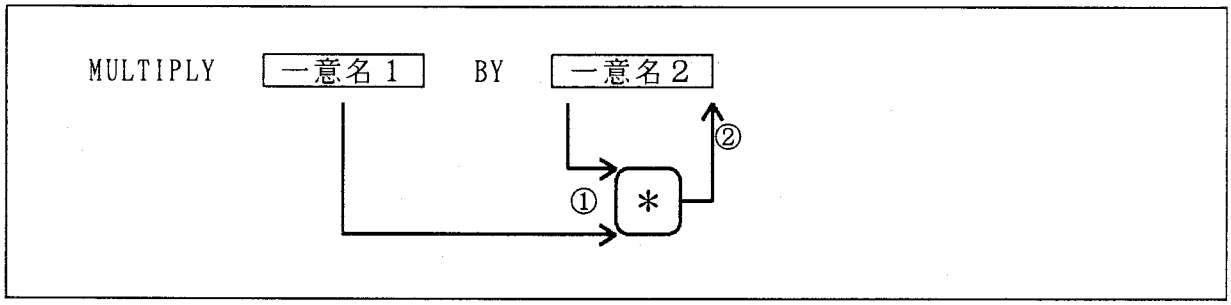


３．５　ＭＵＬＴＩＰＬＹ

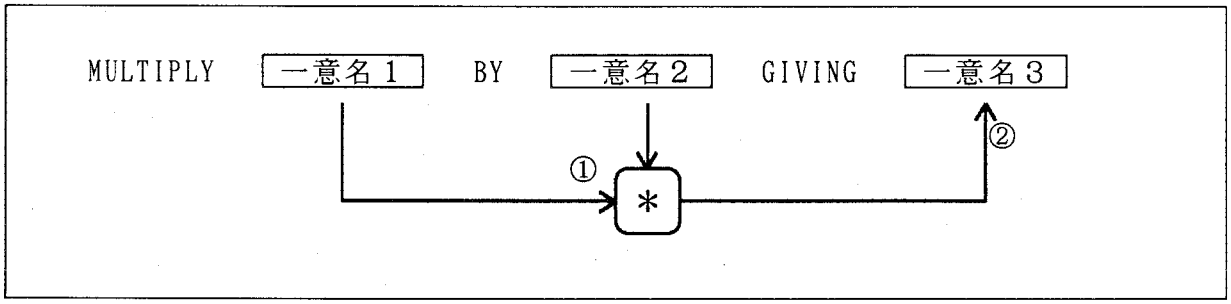
ＭＵＬＴＩＰＬＹ（乗算）文は、数字作用対象同士の積を計算し、その結果をデータ項目に収める。

```
MULTIPLY  { 一意名1 }   BY   {一意名2   [ROUNDED] } ･･･
          { 定数1   }
          [ON  SIZE  ERROR  無条件文1]
          [NOT  ON  SIZE  ERROR  無条件文2]
 [END-MULTIPLY]

MULTIPLY  { 一意名1 }   BY  { 一意名2 }
          { 定数1   }       { 定数2   }
                GIVING   {一意名3   [ROUNDED] } ･･･
          [ON  SIZE  ERROR  無条件文1]
          [NOT  ON  SIZE  ERROR  無条件文2]
 [END-MULTIPLY]
```

ＢＹの前後の数字作用対象の積を計算して、その結果をＢＹの後の数字項目へ収める。



ＢＹの前後の数字作用対象の積を計算して、その結果をＧＩＶＩＮＧの後の数字項目（数字編集項目）へ収める。

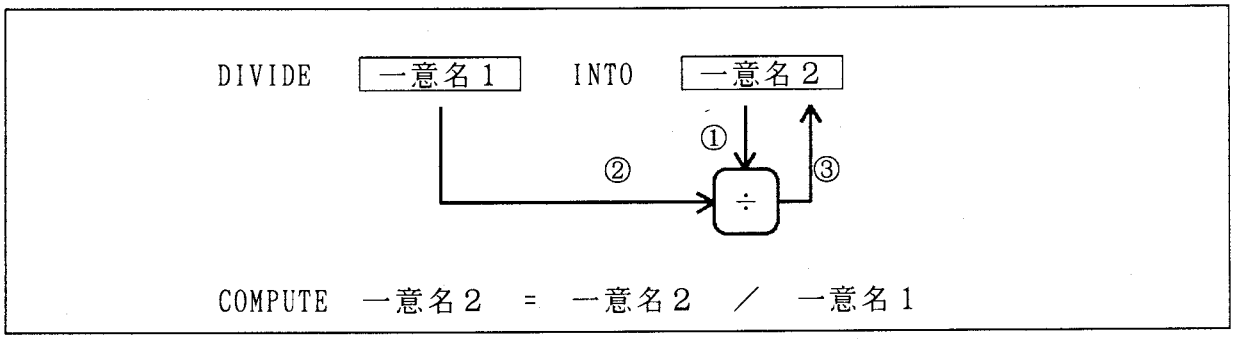


３．６　ＤＩＶＩＤＥ

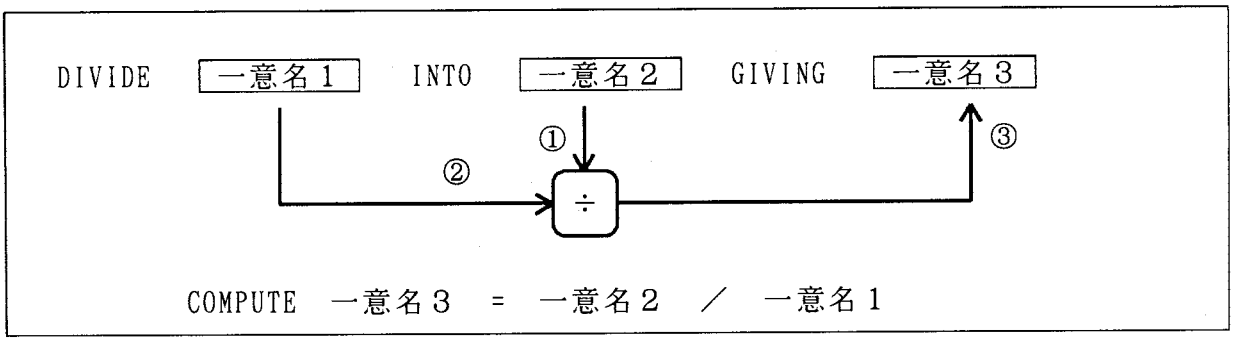
ＤＩＶＩＤＥ（除算）文は、一つの数字作用対象を他の数字作用対象で割り、その商を剰余をデータ項目に収める。

```
DIVIDE  { 一意名1 }  INTO  {一意名2   [ROUNDED] } ･･･
        { 定数1   }

DIVIDE  { 一意名1 }  INTO  { 一意名2 }
        { 定数1   }        { 定数2   }
                  GIVING   {一意名3   [ROUNDED] } ･･･
          [ON SIZE  ERROR  無条件文1]
          [NOT  ON  SIZE   ERROR  無条件文2]
 [END-DIVIDE]

DIVIDE  { 一意名1 }  BY    { 一意名2 }
        { 定数1   }        { 定数2   }
                  GIVING   {一意名3   [ROUNDED] } ･･･
          [ON SIZE  ERROR  無条件文1]
          [NOT  ON  SIZE   ERROR  無条件文2]
 [END-DIVIDE]

DIVIDE  { 一意名1 }  INTO  { 一意名2 }  GIVING  一意名3   [ROUNDED]
        { 定数1   }        { 定数2   }
                  REMAINDER  一意名4
          [ON SIZE  ERROR  無条件文1]
          [NOT  ON  SIZE   ERROR  無条件文2]
 [END-DIVIDE]

DIVIDE  { 一意名1 }  BY    { 一意名2 }  GIVING  一意名3   [ROUNDED]
        { 定数1   }        { 定数2   }
                  REMAINDER  一意名4
          [ON SIZE  ERROR  無条件文1]
          [NOT  ON  SIZE   ERROR  無条件文2]
 [END-DIVIDE]
```

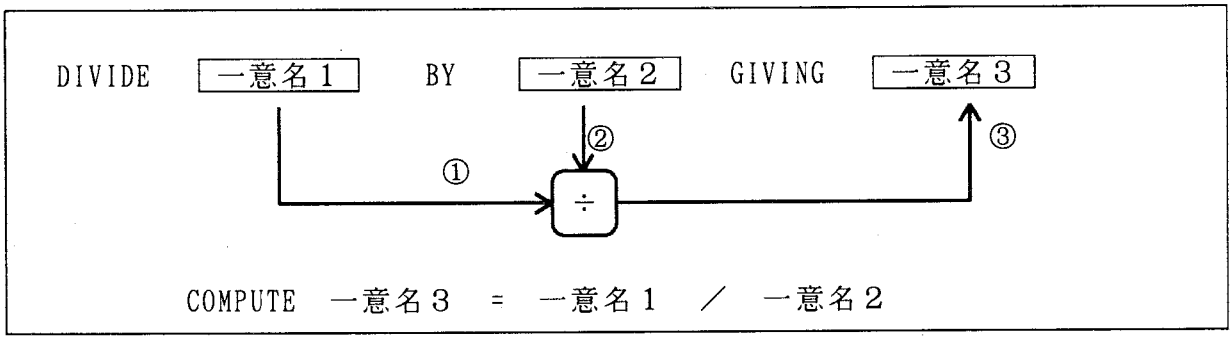
ＩＮＴＯは、一意名２を一意名１で割り、その結果をＩＮＴＯの後の一意名２に収める。



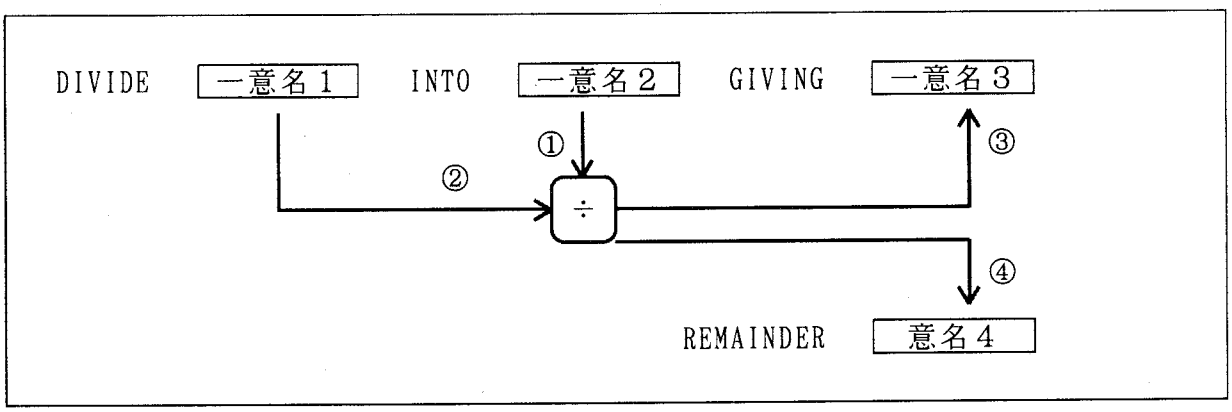
「ＧＩＶＩＮＧ」指定は、一意名２を一意名１で割り、その結果をＧＩＶＩＮＧの後の一意名３に収める。



「ＢＹ」指定は、一意名１を一意名２で割り、その結果をＧＩＶＩＮＧの後の一意名３に収める。



「ＲＥＭＡＩＮＤＥＲ」指定は、一意名２を一意名１で割り、その結果の商をＧＩＶＩＮＧの後の一意名３に収める。そして剰余をＲＥＭＡＩＮＤＥＲの後の一意名４に入れる。



剰余（ＲＥＭＡＩＮＤＥＲ）は、商（一意名３）と除数との積をその被除数から減じた値とする。
ＲＯＵＮＤＥＤ付きの剰余を計算する場合の商は、四捨五入する前の切り捨てられた値である。

GIVINGとREMAINDER指定の書き方で、ON SIZE ERRORを指定して桁あふれが起こると次のようになる。

・商に桁あふれが起こると、剰余の計算は無意味になり、商と剰余のデータ項目の内容は変わらない。
・剰余に桁あふれが起こると、剰余のデータ項目の内容は変わらない。
・商と剰余のどちらで桁あふれになったかは、利用者が調べなければならない。

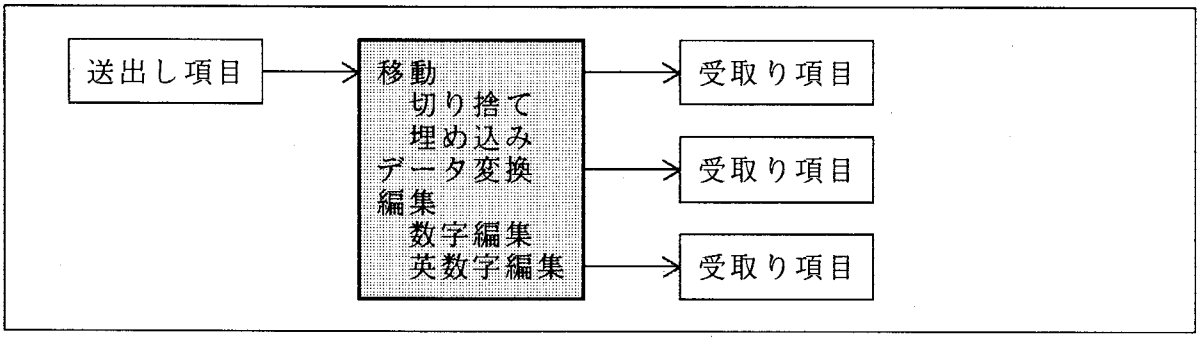
３．７　転記文

３．７．１　ＭＯＶＥ文

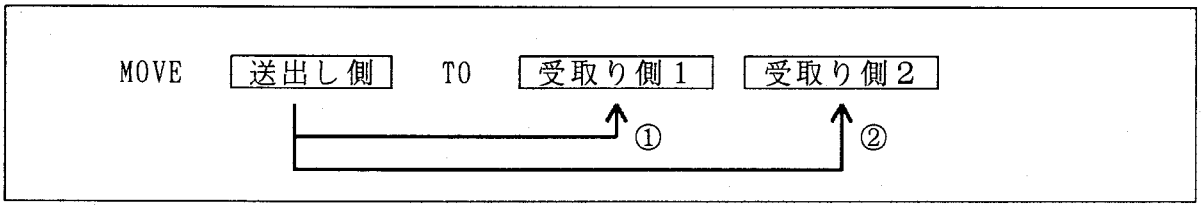
ＭＯＶＥ（転記）文は、データを編集規則に従って一つ以上のデータ領域に移す。

```
MOVE  { 一意名1 }  TO  {一意名2} ･･･
      { 定数1   }

MOVE  { CORRENPONDING }  一意名1  TO  一意名2
      { CORR          }
```

ＭＯＶＥの機能は、次のとおり。



一意名１や定数のデータ項目を送出し側とし、一意名２を受取り側とする。



送出し側が定数や基本項目で受取り側が基本項目の場合を「基本項目転記」という。
基本項目転記は、必要ならば内部表現形式の変換や受取り側に指定された編集を行う。
基本項目転記以外の転記は、英数字項目間の基本項目転記と同じであるが内部表現形式の変換は行われない。

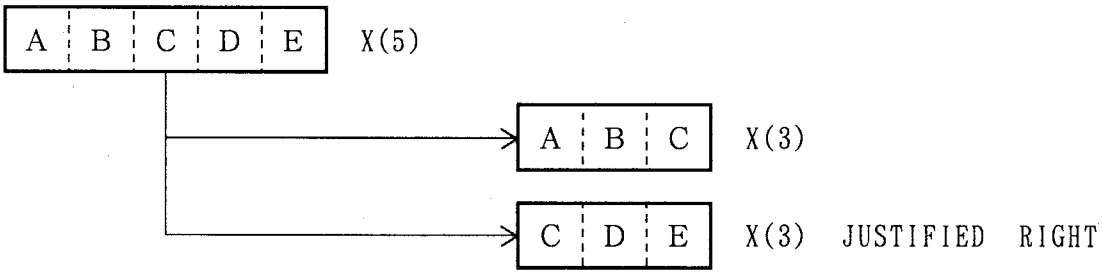
送出し側にある添字は、移す直前に１回だけ評価される。
受取り側にある添字は、それぞれのデータ項目に転記する直前に評価される。

```
ＭＯＶＥ　Ａ（Ｘ）　ＴＯ　　Ｘ　　Ｂ（Ｘ）

実行順序　①　MOVE　A(X)　TO　temp
　　②　MOVE　temp　TO　X
　　③　MOVE　temp　TO　B(X)
```

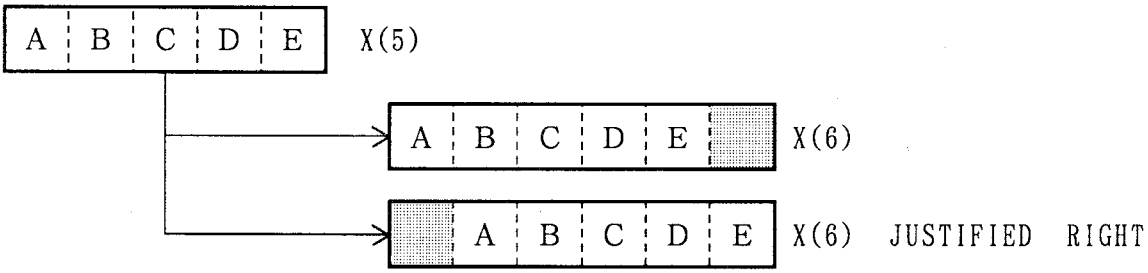
３．７．２　英数字転記

受取り側が英数字項目や英数字編集項目の場合、けたよせ及び空白づめが標準けたよせ規則に従って行われる。

（１）　送り出し項目ｻｲｽﾞ　＞　受取項目ｻｲｽﾞ　→　切り捨て処理



（２）　送り出し項目ｻｲｽﾞ　＜　受取項目ｻｲｽﾞ　→　SPACEによる埋め込み処理

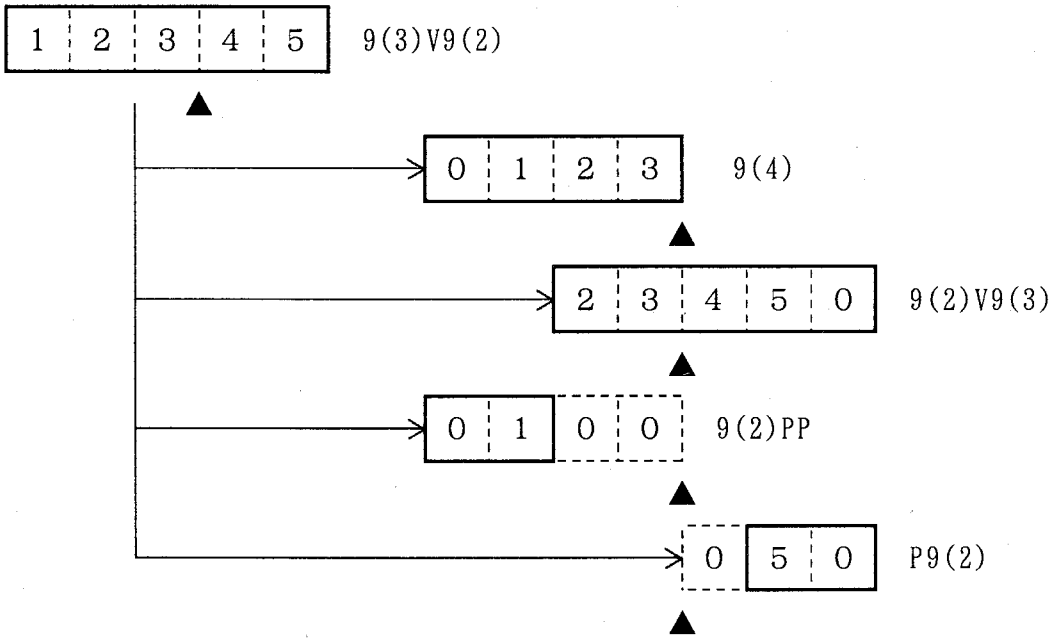


受取り側が英数字項目や英数字編集項目で送出し側が符号付き数字項目の場合、符号は転記されない。送出し項目にPICTURE文字列の「Ｐ」を含むならばＰで指定されるすべての桁は、値ゼロをもつとみなされる。

３．７．３　数字転記

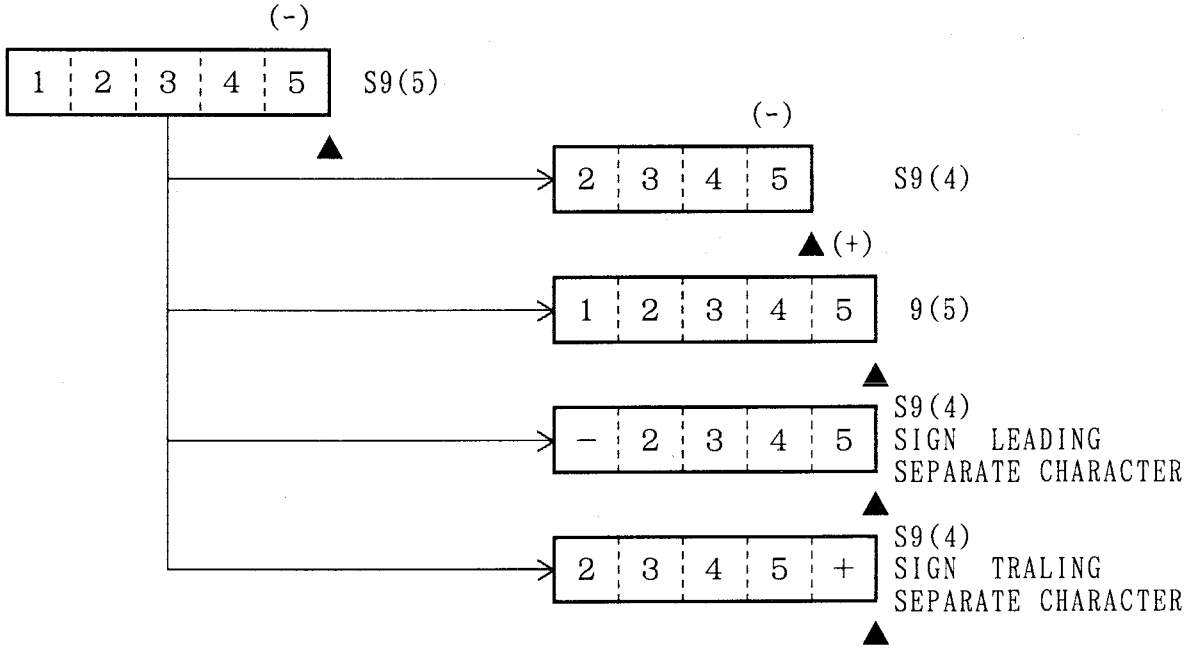
受取り側が数字項目の場合、小数点の位置合せ及び必要なゼロづめが標準けたよせ規則に従って行われる。

（１）　小数点位置合わせ



受取り側が符号付き数字項目の場合、送り出し側の符号を受取り側の符号とする。
送り出し側に符号がない場合、受取り側に正符号が付けられる。
必要ならば符号の表現形式の変換が行われる。

（２）　符号処理

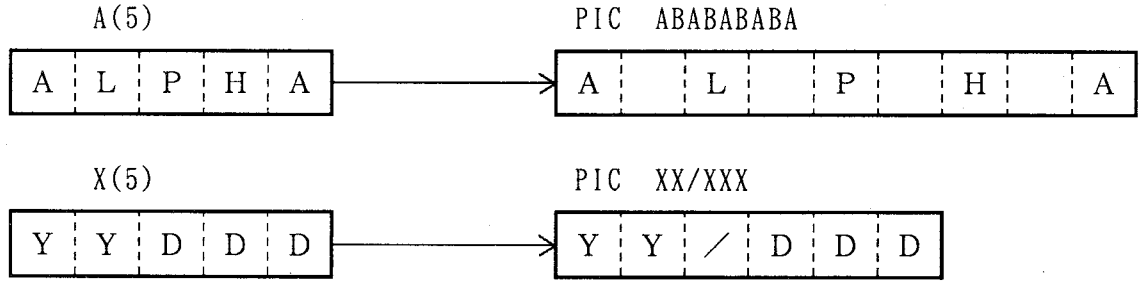


送り出し側が英数字項目の場合、そのデータ項目は符号なし整数項目とみなされて転記される。

３．７．４　編集操作

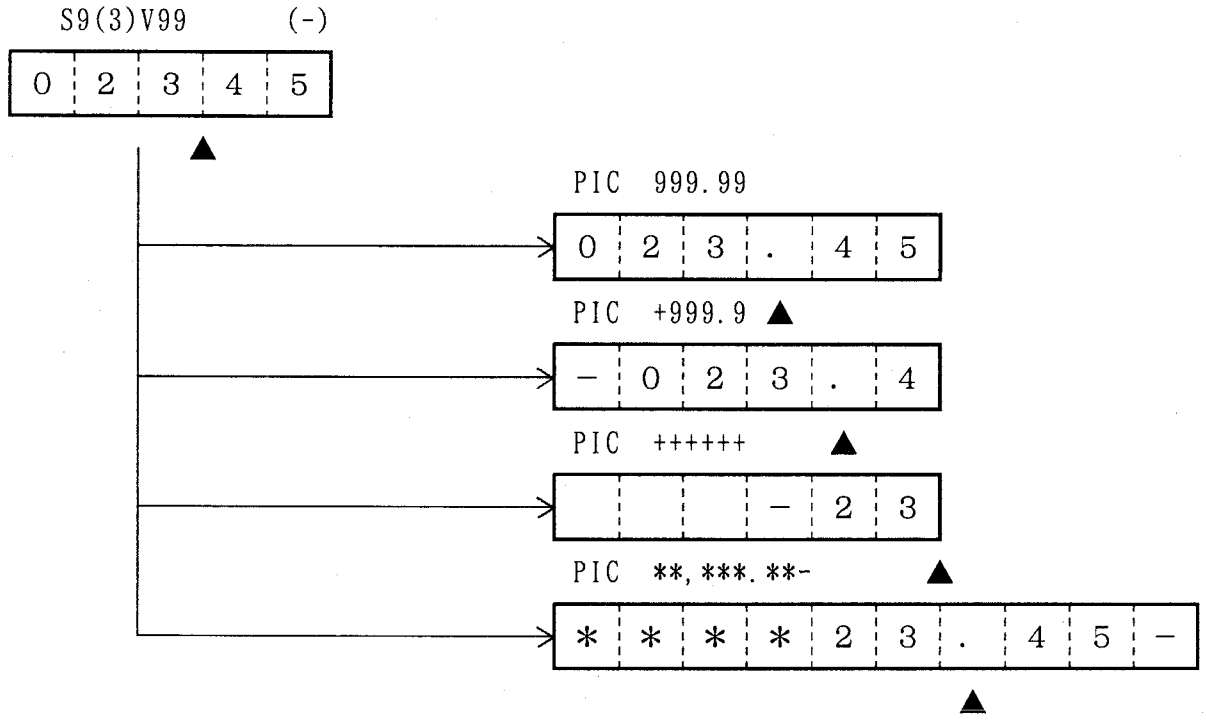
（１）　英数字編集

英数字編集項目が受取り側の場合、けたよせ及び空白づめが標準けたよせ規則に従って行われる。



（２）　数字編集

数字編集項目が受取り側の場合、小数点の位置合せ及び必要なゼロづめが標準けたよせ規則に従って行われる。ゼロは編集によって他の文字に変換されることもある。



３．７．５　転記の規則

基本項目は、英字、英数字、英数字編集、数字、数字編集の項類に分類される。
数字定数の項類は数字とし、文字定数の項類は英数字とする。
表意定数のZERO（ZEROS、ZEROES）の項類は、数字項目や数字編集項目に転記する場合は数字、英数字や英数字編集項目に転記する場合は英数字とする。
表意定数のSPACE（SPACES）の項類は、英字とする。その他の表意定数は英数字とする。

これら項類間の基本項目転記は、次のとおり。
・表意定数SPACE、英数字編集項目、英字項目は、数字項目または数字編集項目に転記できない。
・数字定数、表意定数ZERO、数字項目、数字編集項目は、英字項目に転記できない。
・非整数の数字定数または数字項目は、英数字項目や英数字編集項目に転記できない。

送出し側＼受取り側	英字項目	英数字項目	英数字編集項目	数字項目	数字編集項目
英字項目	○	○	○	×	×
英数字項目	○	○	○	○	○
英数字編集項目	○	○	○	×	×
数字　整数	×	○	○	○	○
非整数	×	×	×	○	○
数字編集	×	○	○	○ 注	○ 注
文字定数	○	○	○	×	×
数字定数　整数	×	○	○	○	○
非整数	×	×	×	○	○
表意定数					
ZERO	×	○	○	○	○
SPACE	○	○	○	×	×
その他	×	○	○	×	×

注：送出し側が数字編集の場合、逆編集によって符号も含めて編集されない数値が求められて、その数値が受取り側に転記される。

３．８　表操作

３．８．１　ＳＥＡＲＣＨ

ＳＥＡＲＣＨ（表引き）文は、指定した条件を満足する表要素を探し、対応する指標名の値がその表要素を指すようにする。

```
SEARCH  一意名1  [ VARYING { 一意名2 }  ]
                           { 指標名1 }
                 [AT  END  無条件文1]

    [ WHEN  条件1 { 無条件文2      } ]  …
                  { NEXT  SENTENCE }
[END-SEARCH]

SEARCH  ALL  一意名1  [AT  END  無条件文1]

        WHEN  { ﾃﾞｰﾀ名1 { IS  EQUAL  TO } { 一意名3 }
              {         { IS  =        } { 定数1   }
              {         {              } { 算術式1 }
              { 条件名1                            }

      [ AND { ﾃﾞｰﾀ名2 { IS  EQUAL  TO } { 一意名4 } ] }  …
            {         { IS  =        } { 定数2   } }
            {         {              } { 算術式2 } }
            { 条件名2                            } ]

              { 無条件文2      }
              { NEXT  SENTENCE }
[END-SEARCH]
```

一意名１には、添字（指標）や部分参照を付けてはいけない。
一意名１のデータ記述には、OCCURS句とINDEXED　BY句がなければいけない。さらに、「SEARCH　ALL」の場合は、OCCURS句に「KEY　IS」句があること。

```
01  ﾃｰﾌﾞﾙ.
    03  表    OCCURS  100  TIMES  INDEXED  BY  指標名.
        05  要素１  PICTURE  X(5).
        05  要素２  PICTURE  S9(4).

    SET  指標名  TO  1
    SEARCH  表  AT  END  CONTINUE
      WHEN  要素２(指標名)  >  500
            MOVE  表(指標名)  TO  ～
    END-SEARCH
```

表操作の比較

逐次探索	非逐次探索
①表の定義 ﾚﾍﾞﾙ番号　ﾃｰﾌﾞﾙ名　OCCURS　回数 INDEXED　BY　指標名 ﾚﾍﾞﾙ番号　表要素　PIC　～ ： ﾚﾍﾞﾙ番号　表要素　PIC　～	ﾚﾍﾞﾙ番号　ﾃｰﾌﾞﾙ名　OCCURS　回数 ASCENDING／DESCENDING KEY　IS　ｷｰ項目 INDEXED　BY　指標名 ﾚﾍﾞﾙ番号　ｷｰ項目　PIC　～ ： ﾚﾍﾞﾙ番号　表要素　PIC　～
②初期設定 SET　指標名　TO　開始位置番号	不要
③命令文 SEARCH　ﾃｰﾌﾞﾙ名 [VARYING　指標名2] [AT　END　無条件文] WHEN　条件1　　文1 [WHEN　条件2　　文2] [WHEN　条件3　　文3] [END-SEARCH]	SEARCH　ALL　ﾃｰﾌﾞﾙ名 [AT　END無条件文] WHEN　条件　[AND　条件2]　文 [END-SEARCH]
④条件文 任意の条件式が指定できる。 比較条件 字類条件 正負条件 条件名条件 スイッチ状態条件	ｷｰ項目（指標名）　＝　{ 一意名 / 定数 / 算術式 }

３．８．２　逐次探索

①テーブル定義

```
01  TABLES.
    02  ﾃｰﾌﾞﾙ名  OCCURS  100  TIMES  INDEXED  BY  指標名1.
        03  要素  PICTURE  ～
```

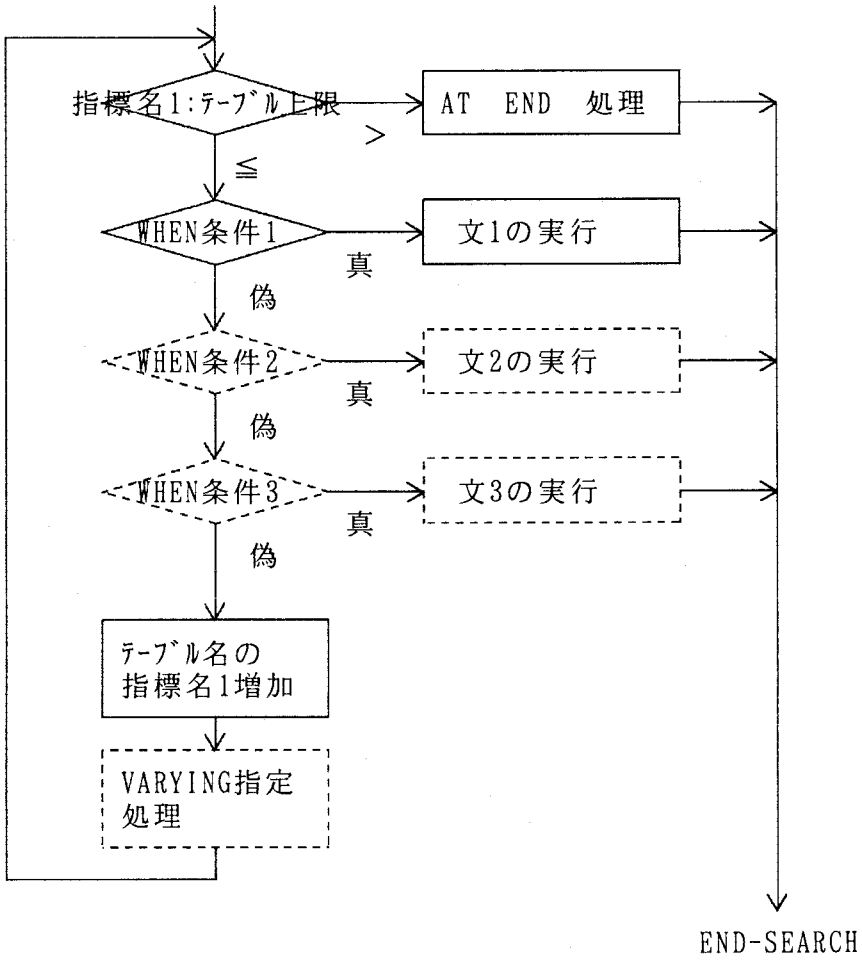
②初期値設定

```
SET  指標名1  TO  1.
```

③逐次探索実行

```
SEARCH  ﾃｰﾌﾞﾙ名   [VARYING  指標名2]
                  [AT  END  無条件命令]
 WHEN  条件1     文1
[WHEN  条件2     文2]
[WHEN  条件3     文3]
[END-SEARCH]
```

逐次探索の実行順序



３．８．３　非逐次探索

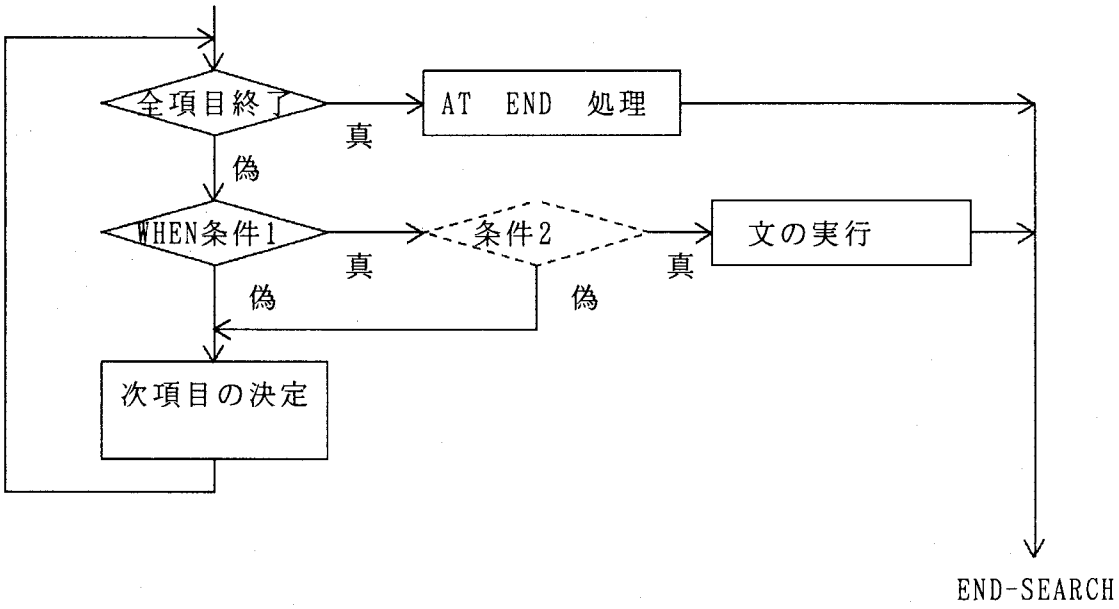
①テーブル定義

```
01  TABLES.
    02  ﾃｰﾌﾞﾙ名   OCCURS  500  TIMES
                    ASCENDING  KEY  IS  年齢
                    INDEXED  BY  指標名1.
    03  年齢      PIC  9(3).
    03  氏名      PIC  X(20).
```

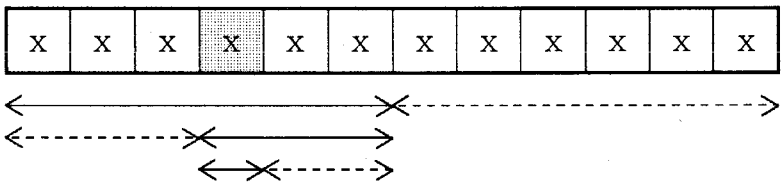
②非逐次探索実行

```
    SEARCH  ALL  ﾃｰﾌﾞﾙ名   [AT  END  無条件命令]
    WHEN  比較条件   [AND  条件2]   文
  [END-SEARCH]
```

非逐次探索の実行順序



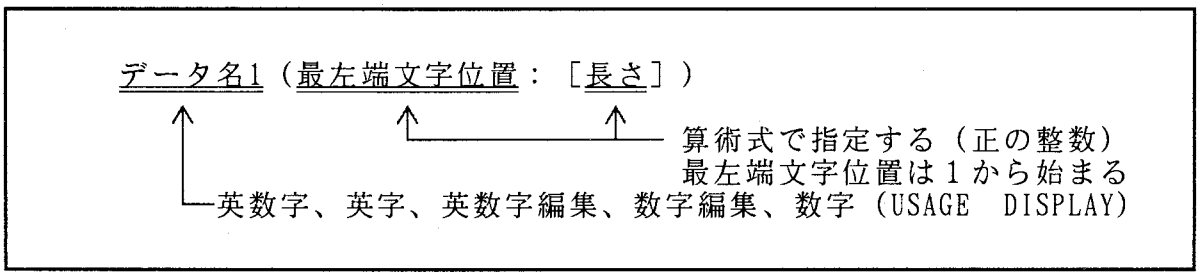
非逐次探索の概要



３．９　文字列操作

３．９．１　部分参照

通常、ＣＯＢＯＬではデータ部（DATA　DIVISION）でデータ名とＰＩＣＴＵＲＥ句を記述してデータの長さ（桁数）を定義する。
ＣＯＢＯＬの部分参照は、手続き部（PROCEDURE　DIVISION）でデータ項目を参照する場合、データ項目の最左端の文字とそこからの長さを指定することによってデータ項目を新たに定義する。



文字列の部分参照は、データ名が英数字項目として再定義されたかのようにデータを参照する。

```
DATA  DIVISION.
WORKING-STORAGE  SECTION.
    :
01  AAAA       PICTURE  X(500).
01  BBBB       PICTURE  9(15).
01  XTABLES.
    05  XTAB  PICTURE  X(8)  OCCURS  100  TIMES.

PROCEDURE  DIVISION.
    :
    MOVE  SPACE  TO  AAAA(10:5)     ←英数字項目の部分参照例
    MOVE  SPACE  TO  AAAA(200:)     ←英数字項目の部分参照例、長さ省略
    MOVE  "747"  TO  BBBB(1:3)      ←数字項目の部分参照例
    MOVE  "abc"  To  XTAB(5)(5:3)   ←添字と部分参照例
```

手続き部では、特に指定がないかぎり、英数字項目の一意名が許されるところならどこでも部分参照が使える。
部分参照が使えない文の例としては、ＰＯＩＮＴＥＲ句でデータ項目の文字位置を指定できるような場合である。

```
    MOVE 開始位置  TO  一意名4
    STRING    一意名1  DELIMITED  BY  一意名2
        INTO 一意名3 WITH  POINTER  一意名4

    MOVE 開始位置  TO  一意名7
    UNSTRING 一意名1 DELIMITED  BY  ALL  一意名2
        INTO  一意名4
        WITH  POINTER  一意名7
```

３．９．２　ＩＮＳＰＥＣＴ

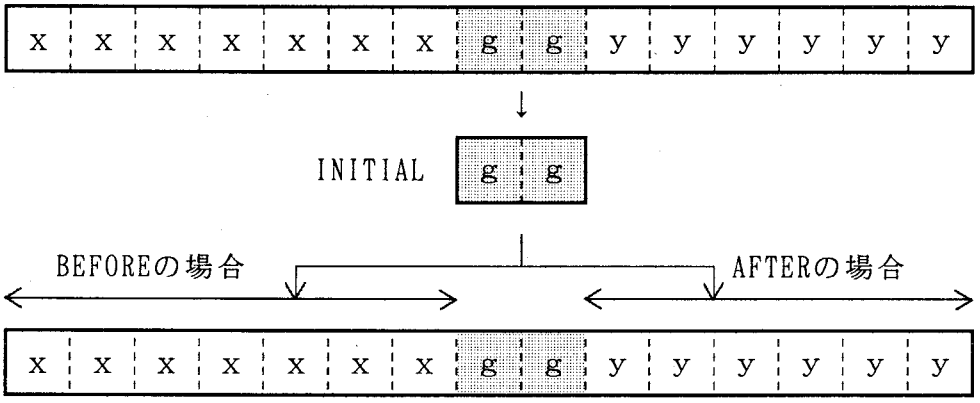
ＩＮＳＰＥＣＴ文は、データ項目中の文字や文字列の出現回数を数えたり、それらを他の文字や文字列で置き換える。

```
INSPECT  一意名1  TALLYING
{一意名2  FOR{ { CHARACTERS
               { {ALL    } { 一意名3 }
               { {LEADING} { 定数1   } }
             { { BEFORE } INITIAL { 一意名4 } }
             { { AFTER  }         { 定数2   } } } ...} ...

INSPECT  一意名1  REPLACING
{ CHARACTERS BY { 一意名5 } { { BEFORE } INITIAL { 一意名4 } } ...
                { 定数3   } { { AFTER  }         { 定数2   } }
{ {ALL    } { { 一意名3 } BY { 一意名5 }
{ {LEADING} { { 定数1   }    { 定数3 }
{ {FIRST  }                  { BEFORE } INITIAL { 一意名4 } } ...}
                             { AFTER  }         { 定数2   } }
```

一意名1は集団項目または基本項目（USAGE DISPLAY）である。
定数は、全て文字定数でなければならない。また、ALLで始まる表意定数であってはならない。

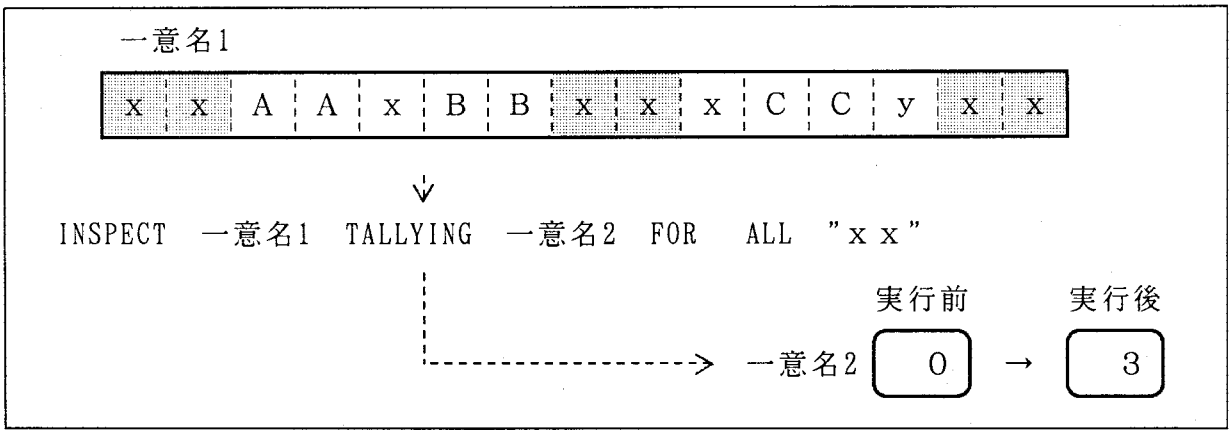
ＩＮＳＰＥＣＴ操作対象範囲の決定

BEFORE　INITIALを指定すると一意名1の最左端文字位置からBEFOREのデータ項目の内容とが最初に一致する直前までが操作対象範囲である。
AFTER　INITIALを指定すると一意名1とAFTERのデータ項目の内容と一致した直後から最右端文字までが操作対象範囲となる。
BEFOREまたはAFTERを省略すると一意名1の最左端文字位置から最右端文字位置までが操作対象範囲となる。

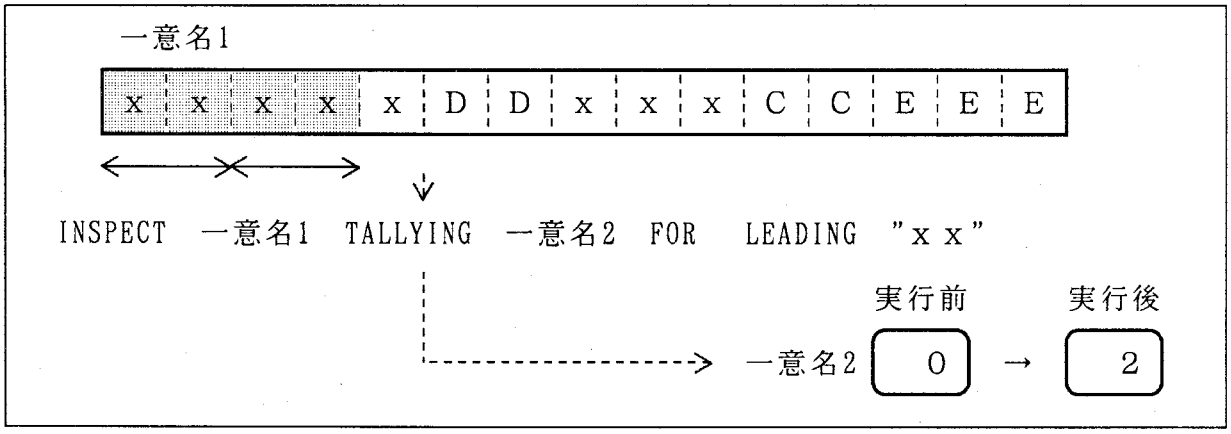


特定の文字列の出現回数を数える

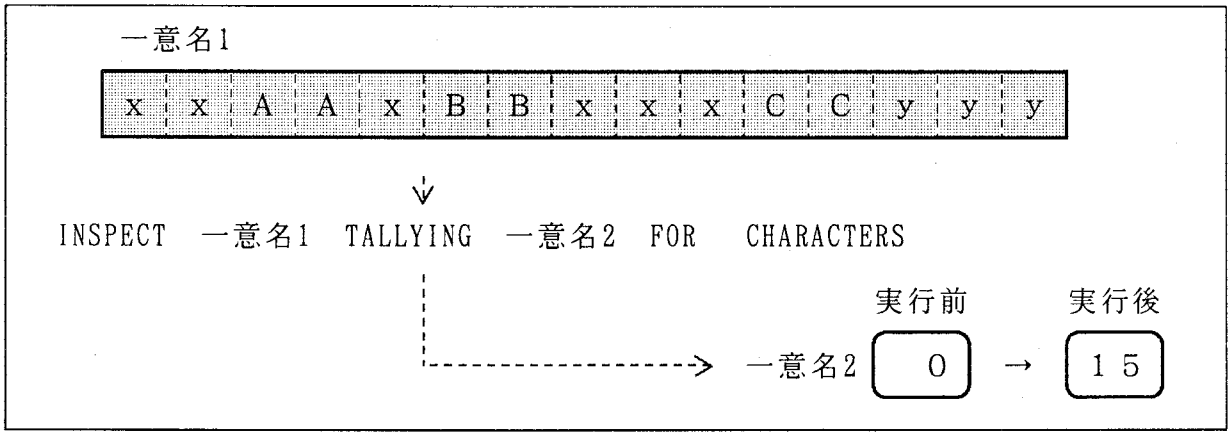
ＩＮＳＰＥＣＴ文でＴＡＬＬＹＩＮＧを指定すると出現回数を検査できる。
ＡＬＬ指定は、操作範囲内で一致した出現回数を一意名2の数字データ項目に加える。



ＬＥＡＤＩＮＧ指定は、操作範囲内の最左端位置で一致が起こり、それ以後一意が続く間の出現回数を一意名2の数字データ項目に加える。



ＣＨＡＲＡＣＴＥＲＳ指定は、操作範囲内の文字数を一意名2の数字データ項目に加える。



特定の文字に置換

一意名1（実行前）

x	x	A	A	x	B	B	x	x	x	C	C	y	y	y

↓

INSPECT　一意名1　REPLACING　CHARACTERS　BY　"@"　AFTER　"Bx"

↓

一意名1（実行後）

x	x	A	A	x	B	B	x	@	@	@	@	@	@	@

特定文字列ならば特定文字列に置換

一意名1（実行前）

x	x	A	A	x	B	B	x	x	x	C	C	y	y	y

↓

INSPECT　一意名1　REPLACING　ALL　"xx"　BY　"@@"

↓

一意名1（実行後）

@	@	A	A	x	B	B	@	@	x	C	C	y	y	y

一意名1（実行前）

x	x	x	x	x	B	B	x	x	x	C	C	y	y	y

↓

INSPECT　一意名1　REPLACING　LEADING　"xx"　BY　"@@"

↓

一意名1（実行後）

@	@	@	@	x	B	B	x	x	x	C	C	y	y	y

一意名1（実行前）

x	x	A	A	x	B	B	x	x	x	C	C	y	x	C

↓

INSPECT　一意名1　REPLACING　FIRST　"xC"　BY　"@@"

↓

一意名1（実行後）

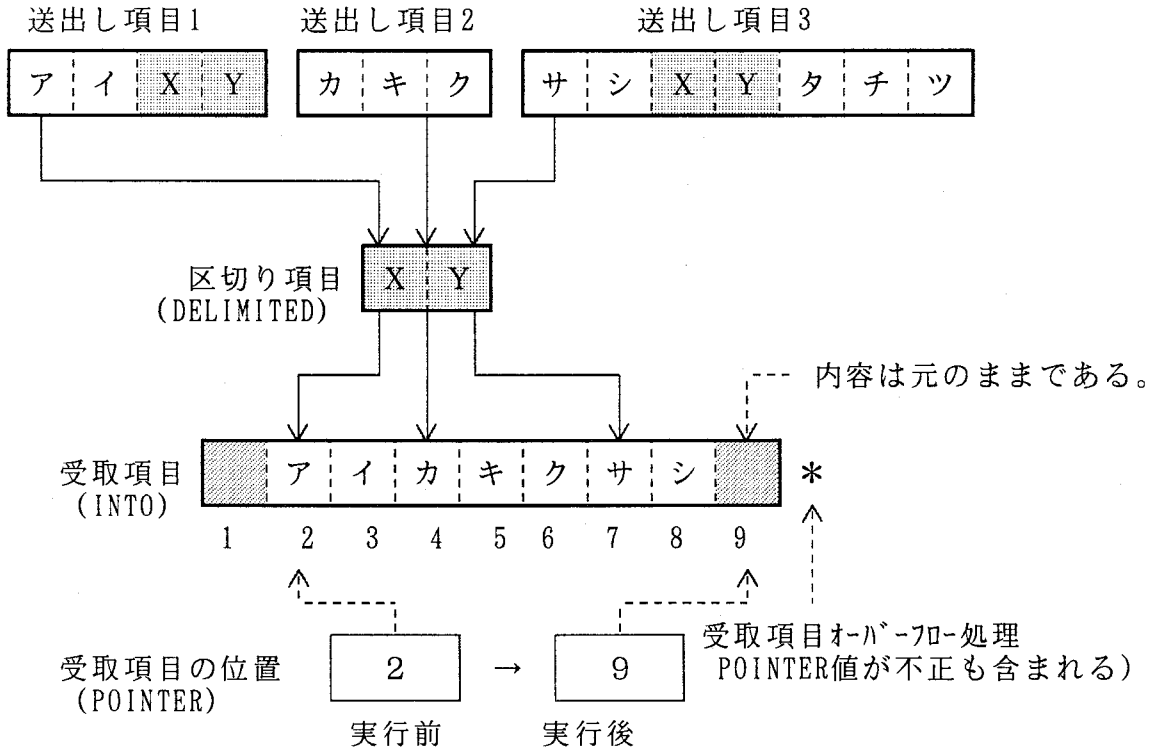
x	x	A	A	x	B	B	x	x	@	@	C	y	x	C

３．９．３　ＳＴＲＩＮＧ

ＳＴＲＩＮＧ文は、幾つかのデータ項目の内容の一部または全部をつなぎ合わせて一つのデータ項目に入れる。

```
STRING { { 一意名1 } ･･･ DELIMITED BY { 一意名2 } } ･･･
       { { 定数1   }                  { 定数2   }
                                      { SIZE    }
        INTO 一意名3 [WITH POINTER 一意名4]
        [ON OVERFLOW 無条件文1]
        [NOT ON OVERFLOW 無条件文2]
[END-STRING]
```

```
STRING 送出し項目1 送出し項目2 送出し項目3 DELIMITED BY 区切り項目
  INTO 受取項目 WITH POINTER 受取項目の位置
END-STRING
```

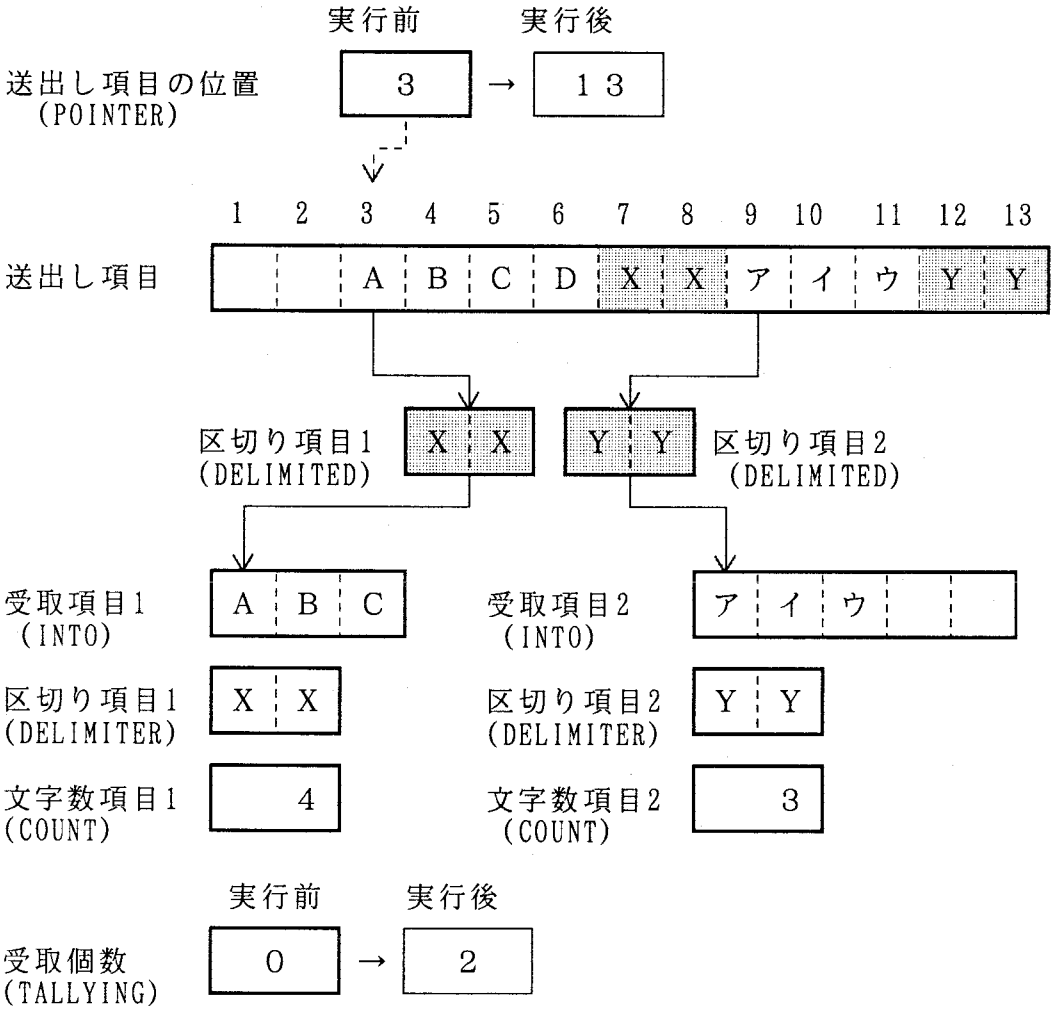


３．９．４　ＵＮＳＴＲＩＮＧ

ＵＮＳＴＲＩＮＧ文は、送出し項目の連続したデータを分解して、幾つかの受取り項目に入れる。

```
UNSTRING  一意名1

     [DELIMITED BY  [ALL] { 一意名2 | 定数1 } [ OR  [ALL] { 一意名3 | 定数2 } ] ･･･]

        INTO   {一意名4   [DELIMITER  IN  一意名5]
                          [COUNT  IN  一意名6] } ･･･
         [WITH  POINTER  一意名7]
         [TALLYING  IN  一意名8]
         [ON  OVERFLOW  無条件文1]
         [NOT  ON  OVERFLOW  無条件文2]
[END-UNSTRING]
```

```
UNSTRING  送出し項目  DELIMITED  BY  区切り項目1  OR  区切り項目2
    INTO  受取項目1  DELIMITER  IN  区切り項目1  COUNT  IN  文字数項目1
          受取項目2  DELIMITER  IN  区切り項目2  COUNT  IN  文字数項目2
    WITH  POINTER  送出し項目の位置
    TALLYING  受取個数
END-UNSTRING
```



指導上の留意点

◎全体的な留意点として、データ形式の種類（BINARY、PACKED-DECIMAL等）を考慮することで性能（実行ｽﾋﾟｰﾄﾞ、ﾒﾓﾘ容量）が向上できる。

◎小数点の桁合わせ
データ項目に小数点以下の桁がある場合の演算は、できるだけ小数点以下の桁数を揃えると性能が向上する。
小数点以下の桁数が異なる演算は、小数点以下を合わせる機械命令が生成されその後、演算が行われる。

◎演算結果を代入するデータ項目には、PICTURE文字列に「Ｓ」を指定して符号付きにすると性能が向上する。
汎用コンピュータの機械命令の展開例は、次のとおり。

```
          ＡＤＤ　Ａ　ＴＯ　Ｂ

01  A  PIC  |  9(5)  |  9(5)  |  USAGE  PACKED-DECIMAL
01  B  PIC  |  9(7)  |  S9(7) |  USAGE  PACKED-DECIMAL

符号なし                          符号付き

AP   B(4),A(3)                    AP   B(4),A(3)
OI   B+3,X'0F'
```

←演算結果の符号を無条件に正にする命令が作られる

代入されない演算対象については、符号の有無は関係ない。

◎算術演算で数字項目に「USAGE　DISPLAY」を使用すると、最初に変換が行われて、演算を行い、そして元に戻すために変換が行われる。
汎用コンピュータの機械命令の展開例は、次のとおり。

```
01  ＣＮＴ  PIC  S9(5)  [USAGE  DISPLAY]

ＡＤＤ  1  ＴＯ  ＣＮＴ

PACK  temp(3),ＣＮＴ(5)
AP    temp(3),=P'1'
UNPK  ＣＮＴ(5),temp(3)
```

PAKED-DECIMALの場合、変換はない

```
AP    ＣＮＴ(5),=P'1'
```

データ項目「ＣＮＴ」が「USAGE　BINARY」の場合は、次のとおり。

```
L     R4,ＣＮＴ
A     R4,=F'1'
ST    R4,ＣＮＴ
```

BINARYはPAKED-DECIMALと同じ３個の機械命令で実現されるが、１０進演算命令よりも２進演算命令の実行ｽﾋﾟｰﾄﾞは速い。

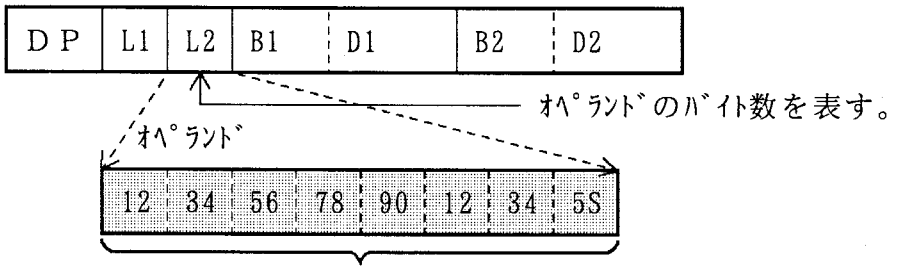
このように、内部で使用するデータ項目は、事情の許す限り次の優先度で使用すると性能が向上する。

効率が良い ←---------- 効率が悪い

BINARY　　PACKED-DECIMAL　　DISPLAY

◎規格ＣＯＢＯＬの数字項目の桁数は１８桁までとしているが、現実の数値としては事務処理ではほとんど１８桁も使用しない。
汎用コンピュータにもよるが１０進数演算命令の乗数（MULTIPLY）、除数（DIVIDE）は数字１５桁が機械命令の限度となっている。
このため、１６桁以上のデータ項目はＣＯＢＯＬライブラリがシミュレーションしている。

命令形式の例

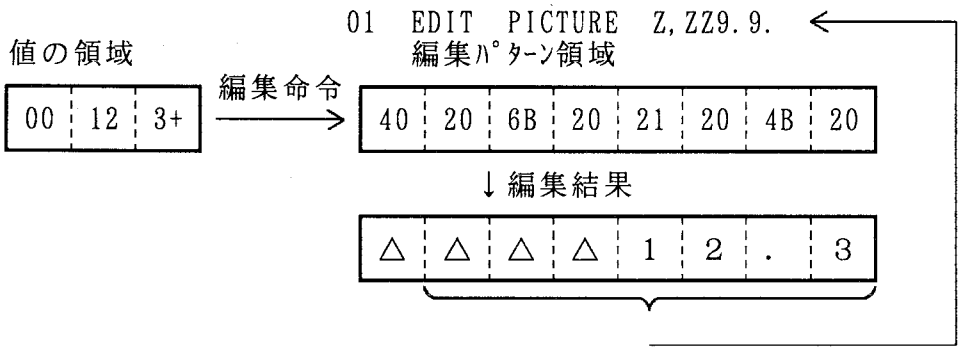


８バイト１５桁

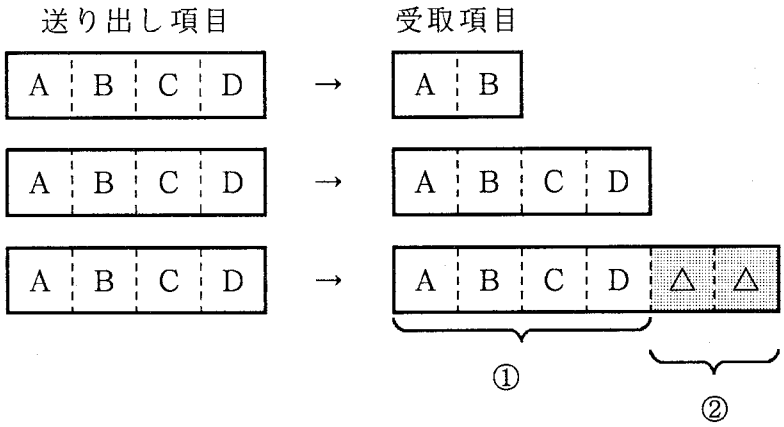
◎PAKED-DECIMALを使用した場合の符号ｺｰﾄﾞは次のように分類される。
汎用コンピュータの例は、次のとおり。

４ﾋﾞｯﾄｺｰﾄﾞ	意味
００００	数字０
０００１	数字１
００１０	数字２
００１１	数字３
０１００	数字４
０１０１	数字５
０１１０	数字６
０１１１	数字７
１０００	数字８
１００１	数字９
１０１０	符号　正
１０１１	符号　負
１１００	符号　正
１１０１	符号　負　（標準）
１１１０	符号　正　（標準）
１１１１	符号　正

◎演算子と機械命令
ＣＯＢＯＬの演算子には加算、減算、乗算、除算、べき乗がある。
べき乗はＣＯＢＯＬライブラリがシミュレーションで実行するが、その他のは機械命令で実行する。
汎用コンピュータでは、数字編集も機械命令で実現できる。



◎英数字項目の転記（ＭＯＶＥ）
英数字項目の転記では、送り出し項目より受取項目の長さが小さいか同じ場合が効率がよい。
受取項目のサイズが送り出し項目より大きいと２段階の転記が行われることになる。



転記を命令回数についての性能を比較すると、次のようになる。

```
01  ＳＷ項目     PICTURE  X(3).   ←定数「ON」、「OFF」が設定されるものとする。

MOVE  "OFF"  TO  ＳＷ項目   ←サイズが同じなので１回で転記できる。
MOVE  "ON "  TO  ＳＷ項目   ←サイズが同じなので１回で転記できる。
MOVE  "ON"   TO  ＳＷ項目   ←「ON」を転記後、１桁の空白を転記する。
```