(仮称)十進 BASIC による
JIS Full BASIC 入門

はじめに

JIS Full BASIC は，国際標準化機構(ISO)が BASIC 言語に関する国際規格を改定したのに合わせて改定された BASIC 言語の新しい日本工業規格(JIS)です。

Full BASIC では，旧規格の基本 BASIC(minimal BASIC)と比較して機能が大幅に強化されています。

Full BASIC では，数値演算の正確さに関する厳格な規定が導入されています。これによって，BASIC の利用者は，BASIC の計算結果を信頼してプログラムを書くことができるようになります。

また,プログラムの論理的な構造を明確に表現するための文法が追加されています。また,プログラムを機能に分割して書くことができるようになっているので,規模の大きなプログラムもわかりやすく書けます。

Full BASIC では，グラフィックスに関する規定も整備されています。利用者が設定した仮想的な座標空間に対する操作を基礎とする,一貫した体系を持つ命令が用意されています。

(仮称)十進 BASIC は，JIS Full BASIC の環境を実現することを目標に作成された言語処理システムです。行番号に頼らずにプログラムの編集が可能なスクリーンエディタを装備するなど，操作性にも気を配っています。(仮称)十進 BASIC を利用して，Full BASIC プログラミングに一歩を踏み出してみませんか。

2000.4.6 改定／2000.11.13 修正／2008.3.13 改定／2010.3.3 修正

白石和夫

目次

1 BASIC の操作

1.1　プログラムの入力と実行

1.1.1　(仮称)十進 BASIC の起動

十進 BASIC がシステムのメニューにないときは，次の方法で起動することができます。

エクスプローラで(仮称)十進 BASIC のシステムのあるフォルダを開き，BASIC のアイコンをダブルクリックすると，(仮称)十進 BASIC が起動します（Windows の場合です）。

1.1.2　プログラムの読込みと実行

 をクリックすると，ファイルからプログラムを読み込むことができます。(仮称)十進 BASIC には多数のサンプルプログラムが付属しているので読み込んで実行してみてください。  をクリックすると実行します。

推奨：　FUNCTION フォルダの ABS.BAS，INT.BAS，MOD.BAS，SQR.BAS，TAN.BAS，
　　STATEMEN フォルダの PRINT.BAS，PLOTPOIN.BAS，PLOTLINE.BAS など。

注意：　Textfile フォルダに含まれるプログラムはファイルを生成します。プログラムの意味を理解するまでは実行しないでください。

プログラムによっては実行に時間がかかるものがあります。プログラムの実行を途中で打ち切りたいときは，  をクリックしてください。

1.1.3　プログラムの入力と実行

BASIC が起動すると，プログラムの入力が可能な状態になっています。

オプションメニューの入力時オートフォーマットで，「機能語を大文字に変換する」にチェックしておくと入力が楽になります。

キーボードから実行の開始を指示したいときは F9 キーを押してください（MAC は F8）。

[補足] キーボードの使い方（MAC では異なります）

Enter キー　　改行の挿入

Back Space キー　カーソルの直前の文字を消す　Delete キー　　カーソルの直後の文字を消す

☆ 改行の取り消し方　行頭で Back Space キーを押すか，行末で Delete キーを押す。

Home キー　行の先頭へ　　End キー　　行末へ

Ctrl-Home　　文書頭へ　　Ctrl-End　　文書末へ

Insert キー　挿入モードと上書きモードの切り替え

1.1.4　プログラムの編集

切り取り，コピー，貼り付けなどの機能を利用してプログラムの編集が可能です。切り取り，コピーは，実行前にテキストを選択しておく必要があります。

1.1.5　BASIC の終了

「ファイル」メニューから「BASIC の終了」を選択してください。

2 BASIC の計算機能

2.1 変数と数値式

2.1.1 数値式

加減乗除の演算を，それぞれ，＋，－，＊，／で表します。また，べき乗の演算は，＾という記号を用いて，たとえば，2^3 を 2^3 のように表します。

異なる演算記号を含む式を書くと，最初にべき乗を実行し，次に乗除算を実行して，最後に加減算を実行します。同順位の演算は左から実行します。

計算を実行してその結果を表示させるのに PRINT 文を用います。

```
10 PRINT 2+3*4^2
20 PRINT 3/4*5
30 PRINT 2^4^5
40 END
```

10 行は $2+3\times4^2$ を計算します。

20 行は $3\div4\times5$ を計算します（$\frac{3}{4\times5}$ ではありません）。

30 行は $(2^4)^5$ を計算します（2^{4^5} ではありません）。

2.1.2 括弧（かっこ）

演算の優先順位を変更したいときは，括弧を用いることができます。括弧を 2 重，3 重，･･･に用いることもできますが，｛ ｝や［ ］を用いず，すべて（ ）を用います。

また，負号で始まる式を計算式のなかに書くときは，その前後を括弧で括ります。

```
10 PRINT ((2+3)*4)^2
20 PRINT 3*(-4)
30 END
```

10 行は，$\{(2+3)\times4\}^2$ を計算します。

20 行は，$3\times(-4)$を計算します。

2.1.3 行番号

プログラムの各行の左端の数字を**行番号**といいます。(仮称)十進 BASIC では行番号を省くことができます。本手引きでは JIS との整合性を保持する目的で行番号を記述しますが，(仮称)十進 BASIC でプログラムを作成する場合には，行番号を省くことを推奨します。

2.1.4 変数

BASIC では，コンピュータ内部の数値の記憶場所のことを**変数**といいます。変数は，A,B,C などの名前を付けて利用します。また，変数名には，LEFT，RIGHT など，長い綴りを用いることもできます。変数名の付け方の詳細は，JIS 規格を参照してください。

英字は，大文字と小文字のいずれも使用できますが，対応する英字の大小の違いは無視されます。たとえば，A と a は同じ変数を表します。

変数に数値を記憶させることを**代入**といいます。変数に新たな数値を代入すると，それ以前に保持していた数値は消滅して参照できなくなります。

変数に数値を代入するのに LET 文を用います。

```
10 LET x=10
20 PRINT 2*x^2+3*x+4
30 END
```

上の例のように，変数名を数値式で用いると，変数名はその変数に記憶されている数値を表します。

LET 文は，

LET　数値変数名 ＝ 数値

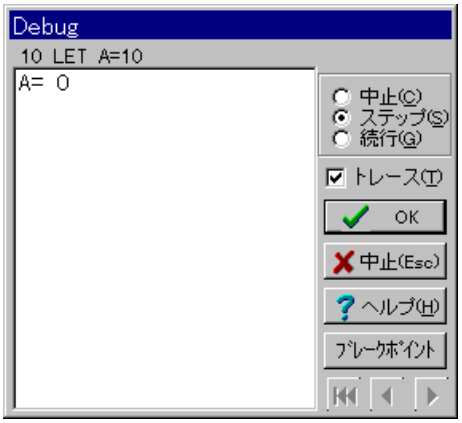
の形式で書きます。LET 文が実行されると，右辺の値が計算されて左辺の変数名が示す変数に代入されます。

[Note] BASIC では，乗算の記号を省くことはできません。x*y のつもりで xy と書くと，xy は，x や y とは別の単一の変数と解釈されます。

問1 次のプログラムを実行すると結果はどうなりますか？

```
10 LET a=10
20 LET a=a+1
30 PRINT a
40 END
```

(仮称)十進 BASIC には，ステップ実行の機能があります。をクリックすると，一行ずつ実行するモードで実行を開始します。右図のような画面が現れたら，Enter キーを押すか，Ok ボタンをクリックするごとに一行ずつ実行します。Debug ウィンドウには，これから実行する文と各変数の値が表示されます。一行実行するごとに変数の値がどのように変化するか調べてみてください。なお，Debug ウィンドウ下部の をクリックすると，履歴をさかのぼって見返すことができます。

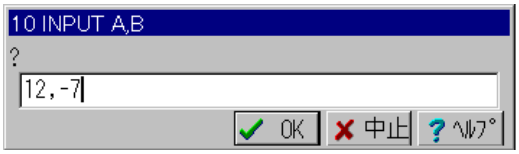


2.1.5 INPUT 文

プログラムの実行時に変数に値を代入したいときには INPUT 文を用います。INPUT 文では，INPUT に続けて数値を代入したい変数をコンマで区切って並べます。

次のプログラムは，入力された 2 数の積を表示します。

```
10 INPUT a,b
20 PRINT a*b
30 END
```



このプログラムを実行すると，図のようなダイアログが表示されるので，入力したい 2 数をコンマで区切って打ち込み，最後に Enter キーを押してください。

2.1.6　　大きな数と小さな数

```
10 PRINT 1/7
20 END
```

を実行すると，

```
 .142857142857143
```

のように表示されます。このように，実行結果の絶対値が 1 より小さい場合には，小数点の左側の 0 を省いて表示されます。

また，

```
10 PRINT 2^100,2^(-100)
20 END
```

を実行すると，実行結果は次のように表示されます。

```
      1.26765060022823E30       7.88860905221012E-31
```

これらは，それぞれ，$1.26765060022823\times10^{30}$，$7.88860905221012\times10^{-31}$ を表します。

2.2　PRINT 文

2.2.1　　文字列

PRINT 文を利用して文字列を表示することができます。文字列は，その前後を引用符(")で囲んで表します。

[補足] 「"」は，Shift キーを押しながら文字キーの上に並んでいる数字キーのうちから「２」のキーを押して入力します。また，" と ″ は印刷上のデザインの相違で，同じ文字です。

```
10 PRINT ″ABC″
20 END
```

10 行を PRINT ABC とすると，数値変数 ABC の値を表示するという意味になってしまいます。

2.2.2　　項目の区切り記号（コンマとセミコロン）

PRINT 文では計算結果を出力するときにその様式を制御することができます。そのうち，特に重要なものを説明します。

PRINT 文には，コンマまたはセミコロンで区切って複数の項目を書くことができます。項目の区切り記号にセミコロンを用いると，各項目は詰めて表示されます。項目の区切りにコンマを用いると，各項目は一定の桁位置まで空白を出力した後に出力されます。たとえば，

```
10 PRINT 3; 3^2, 1/3, 2*3
20 END
```

を実行すると，

```
 3  9                .333333333333333          6
```

のように表示されます。

通常，PRINT 文を実行すると，最後に改行します。改行したくない場合には，PRINT 文の末尾にコンマかセミコロンを書いておきます。たとえば，

```
10 PRINT 1;2;3;
20 PRINT 4;5
30 END
```

を実行すると,

```
1  2  3  4  5
```

のように出力されます。

PRINT 文には，出力項目を書かない特別な形式があります。この PRINT 文は，改行させる働きを持ちます。たとえば,

```
10 PRINT 1/3
20 PRINT
30 PRINT 3^2
40 END
```

を実行すると

```
.333333333333333

 9
```

のようになります。

2.3 FOR～NEXT

2.3.1 FOR～NEXT 構文

FOR～NEXT は BASIC の応用プログラムで頻繁に用いられる繰り返しの構文です。FOR と NEXT は常に対にして用いられる命令で，FOR 文に指定された変数の値を順に変化させながら FOR 行と NEXT 行にはさまれた各行を繰り返し実行させます。

次に示すプログラムは n=1,2,3,…10 に対して n^2 を計算して表示します。

例1

```
10 FOR n=1 TO 10
20    PRINT n,n^2
30 NEXT n
40 END
```

例 1では, まず, n=1 に対して 20 行が実行され, 次に n=2 に対して 20 行が実行されます。次は n=3 に対して 20 行を実行します。同様のことを順に繰り返して，最後に n =10 に対し 20 行を実行して終了します。ステップ実行を選択して，変数の値の変化と実行順序を調べてみてください。

FOR～NEXT の性質をくわしく調べるために次のプログラムを実行してみてください。

例2

```
10 INPUT n
20 FOR k=1 TO n
30    PRINT k
40 NEXT k
50 PRINT "Last",k
60 END
```

このプログラムを実行して n に 10 を入力してみると，50 行の PRINT 文を実行したときには k の値が 11 になっていることがわかると思います。上のプログラムでは，最初，k に 1 が代入され，NEXT 文を実行するごとに k の値に 1 が加算され，それが n より大きくなると NEXT 文の次に進むという動作をしています。

それでは，n に 0 を入力するとどうなるでしょうか。この場合，30 行は 1 回も実行されずに 50 行のみが実行されます。このときの k の値は 1 です。この場合は，k の値が最初から n より大きいので 30 行や NEXT 文を実行せずに 50 行に進んでしまうのです。この性質は重要ですの覚えておいてください。

2.3.2 FOR～NEXT の応用（数列の和，積）

次のプログラムは自然数 n をキーボードから入力すると $1^2+2^2+3^2+\cdots+n^2$ を計算します。

例3

```
10 INPUT n
20 LET S=0
30 FOR k=1 TO n
40    LET S=S+k^2
50 NEXT k
60 PRINT S
70 END
```

和は変数 S を用いて求めます。まず，20 行で S に 0 を代入しておき，30～50 行で k=1,2,3,…,n について S に k^2 を加算します。

同様の手法が数列の積の計算に使えます。順列の数 ${}_nP_r$（$=n(n-1)(n-2)\cdots(n-r+1)$）は次のように計算できます。

例4

```
10 INPUT n,r
20 LET p=1
30 FOR k=(n-r+1) TO n
40    LET p=p*k
50 NEXT k
60 PRINT p
70 END
```

[Note] (仮称)十進 BASIC では変数の初期値は 0 ですが, JIS の規定では変数の初期値は 0 でなくてもよいことになっています。したがって，互換性のあるプログラムを作るためには例 3で 20 行を省略することはできません。

2.3.3 FOR～NEXT の応用（漸化式で定義された数列）

変数の値を順に更新していく手法を用いると, 漸化式で定義された数列の計算ができます。次のプログラムは，a_1=5，a_{n+1}=3a_n+2 で定義される数列{a_n}の第 n 項を計算します。

例 5

```
10 INPUT n
20 LET a=5
30 FOR k=2 TO n
40    LET a=3*a+2
50 NEXT k
60 PRINT a
70 END
```

このプログラムは n=1 の場合にも正しい答えが得られます。それは，k に 2 を代入した時点で k の値が n を超えているために，40 行が 1 回も実行されないためです。

2.3.4 STEP

FOR～NEXT 構文で繰り返しのたびごとに制御変数に加算される数値を 1 以外の数値にすることができます。次のプログラムでは，x の値を 0 から 1 まで 0.1 ずつ加算しながら x^2 を計算して出力します。

例 6

```
10 FOR x=0 TO 1 STEP 0.1
20   PRINT x,x^2
30 NEXT x
40 END
```

また，数値を大きい値から小さい値へ変化させる目的でこの構文を用いることができます。

例 7

```
10 FOR k=10 TO 1 STEP -1
20   PRINT k
30 NEXT k
40 END
```

問 2　次のプログラムは n に偶数を入力することを想定しているように思えますが，奇数を入力しても動作します。どのように動作するのでしょうか。

```
10 INPUT n
20 FOR k=0 TO n STEP 2
30   PRINT k
40 NEXT k
50 END
```

2.4　DEF 文

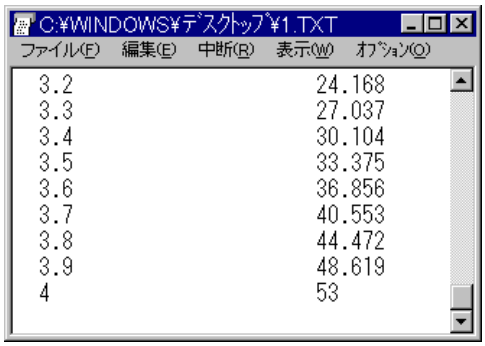
FOR～NEXT の機能を利用して関数値の表を作ってみましょう。次のプログラムは，関数 $f(x)=x^3-3x+1$ について x の値を－4 から 4 まで 0.1 刻みで変化させて関数値を計算します。

例 8

```
10 DEF f(x)=x^3-3*x+1
20 FOR x=-4 TO 4 STEP 0.1
30    PRINT x,f(x)
40 NEXT x
50 END
```

このプログラムを実行すると結果が図のように表示されますが，右端のスクロールバー

をマウスで操作することで出力結果の全体を見ることができます。



10 行の DEF 文は関数を定義する命令です。関数の名前の付け方は変数の場合と同様です。ただし，同じ名前を関数名と変数名とに用いることはできません。10 行では，関数名として f を用いています。関数名に続けて括弧を書き，括弧内に変数名を書きます。＝に続けて括弧内に書いた変数を利用して計算式を書きます。左辺で括弧内に書いた変数は，プログラムの他の部分で用いる変数とは別の変数になります（つまり，数値の記憶場所が別）。

2.5　組込み関数

2.5.1　　平方根，絶対値

BASIC では，$\sqrt{x}$ を SQR(x)のように書きます。また，x の絶対値 $|x|$ は ABS(x)で表します。

[Note]　SQR は，SQuare Root の，ABS は，ABSolute の略です。

直角三角形の直角をはさむ 2 辺の長さがそれぞれ a，b であるとすると斜辺の長さは $\sqrt{a^2+b^2}$ です。a，b を入力して $\sqrt{a^2+b^2}$ を求めるプログラムは次のようになります。

例9

```
10 INPUT a,b
20 PRINT SQR(a^2+b^2)
30 END
```

2.5.2　　三角関数

x の正弦(sine)，余弦(cosine)，正接(tangent)は，それぞれ，SIN(x)，COS(x)，TAN(x)を用いて求めることができます。角の大きさの単位は標準ではラジアンですが，

OPTION ANGLE DEGREES

をプログラムのはじめに書くことで角の大きさの単位を度(degrees)に変えることができます。次のプログラムは，余弦定理を用いて，三角形の 2 辺の長さ a，b とそれらがはさむ角の大きさ C を入力すると残りの辺の長さを答えます。

例 10

```
10 OPTION ANGLE DEGREES
20 INPUT a,b,C
30 PRINT SQR(a^2+b^2-2*a*b*COS(C))
40 END
```

2.5.3　　逆三角関数

余弦定理の式を変形すると，

$$\cos A = \frac{b^2+c^2-a^2}{2bc}$$

となりますから，三角形 ABC の 3 辺の長さ a,b,c から頂角 A の大きさ A を求めることができます。$\cos A$ の値から，対応する A の値を求めるのに，BASIC の組込み関数 ACOS(x)を用

いることができます。ACOS(*x*)は，0≦*t*≦180の範囲で cos *t*° = *x* となる *t* を求める組込み関数です。

例11　*a,b,c* を入力して，*A* を求めるプログラム。

```
10 OPTION ANGLE DEGREES
20 INPUT a,b,c
30 PRINT ACOS((b^2+c^2-a^2)/(2*b*c))
40 END
```

a=7，*b*=3，*c*=5 を入力したときの実行結果を次に示します。

```
? 7,3,5
 120
```

同様の組込み関数に ASIN(*x*)，ATN(*x*)，ANGLE(*x,y*)があります。

ASIN(*x*)は，−90≦*t*≦90の範囲で sin *t*° = *x* となる *t* を求めます。

ATN(*x*)は，−90＜*t*＜90の範囲で tan *t*° = *x* となる *t* を求めます。

ANGLE(*x,y*)は，−180＜*t*≦180 の範囲で原点と点(*x, y*)を結ぶ線分が *x* 軸の正の向きとなす角を求めます。

2.5.4　INT 関数と MOD 関数

INT(*x*)は *x* を越えない最大の整数です。*n* が整数であれば INT(*n*)は *n* と一致しますが，小数部を持つ場合には負の方向に切り捨てます。たとえば，INT(2.3)＝2, INT(-2.3)＝-3 です。

MOD(*a*,*b*)は *a* を *b* で割った余りです。MOD(*a*,*b*)＝*a*－*b**INT(*a*/*b*)で定義されています。たとえば，MOD(5, 3)＝2，MOD(-4, 3)＝2，MOD(1.3, 0.4)＝0.1 となります。

b＞0 であれば，0≦MOD(*a*,*b*)＜*b* となります。*b*＜0 のときは，*b*＜MOD(*a*,*b*)≦0 です。

2.5.5　PI 関数など

BASIC には特別な定数を表す組込み関数が用意されています。PI は円周率πの近似値です。MAXNUM は BASIC で扱うことのできる最大の数です。

例12

```
10 PRINT PI, MAXNUM
20 END
```

2.5.6　乱数（RND 関数）

RND は特殊な関数です。この関数は引数を持ちませんが，計算するごとに異なる値を返します。たとえば，次のプログラムを実行してみてください。

例 13

```
10 FOR k=1 TO 10
20 PRINT RND
30 NEXT k
40 END
```

RND 関数が返す数値のことを乱数といいます。RND 関数は，0 以上 1 未満の範囲で偏りなくでたらめな数値を発生させます。上のプログラムは実行するたびごとに同じ結果を返します。実行するごとに異なる系列の乱数がほしい場合には，次に示すように，RND 関数を実行するまえに RANDOMIZE 文を実行しておきます。

RND 関数をさいころの代わりに使うためには次のようにします。このプログラムでは，

RND の値を 6 倍して 1 を加え，整数部分を取り出すことで 1 から 6 までの整数が得られるようにしています。

```
10 RANDOMIZE
20 FOR n=1 TO 20
30     PRINT INT(RND*6+1)
40 NEXT n
50 END
```

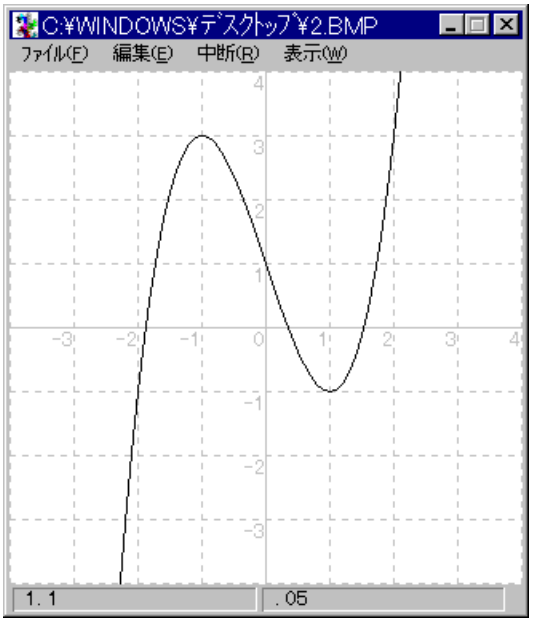
2.6　グラフィックス

2.6.1　　関数のグラフ

グラフィック機能を利用すると関数のグラフを描くことができるようになります。

例14　$-4 \leqq x \leqq 4$，$-4 \leqq y \leqq 4$ の範囲で関数 $y=x^3-3x+1$ のグラフを描く。

```
10 DEF f(x)=x^3-3*x+1
20 SET WINDOW -4,4,-4,4
30 DRAW GRID
40 FOR x=-4 TO 4 STEP 0.1
50    PLOT LINES: x,f(x);
60 NEXT X
70 END
```



このプログラムを実行すると，右の図のような結果が得られます。結果が表示されるウィンドウでマウスカーソルを動かすとその点の座標が下欄に表示されます。

描画に用いられる領域の形状は，縦横の長さが等しい正方形です。縦横のドット数は画面の大きさから(仮称)十進 BASIC が最適な大きさを選択しますが，メニューから選択して決めることもできます。

20 行の SET WINDOW 文は，画面上の描画領域に，x 座標の範囲が -4 から 4，y 座標の範囲が -4 から 4 となるような座標系を設定します。

30 行の DRAW 文は grid を描きます。grid は JIS には規定されていませんが，JIS で規定される命令を組み立てて同様の機能を実現することができます。

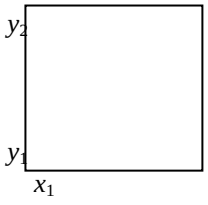
50 行の PLOT LINES 命令は，点$(x\ ,\ x^3-3x+1)$を順に結ぶ線分を描くことで関数のグラフを描くために用いられています。一般に，

PLOT LINES: x,y;

と書くと，次に実行される PLOT LINES 命令で指定される点との間が線分で結ばれます。

2.6.2 SET WINDOW

SET WINDOW x_1, x_2, y_1, y_2

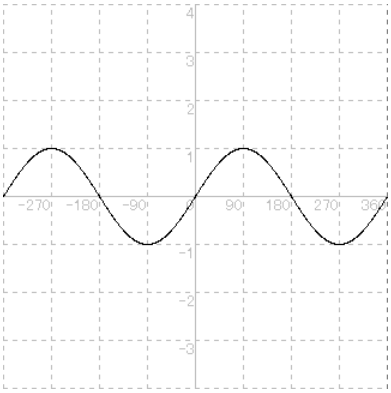


と書くと，描画領域の横方向に左端が x_1，右端が x_2，縦方向に下端が y_1，上端が y_2 であるような座標系が設定されます。

幾何学的な図形を描くような場合には $x_2 - x_1 = y_2 - y_1$ でないと具合が悪いのが普通ですが，関数のグラフのように縦軸と横軸が性質の異なる量を表す場合にはそうである必要はありません。たとえば，角の大きさの単位を度(°)にして正弦関数のグラフを描く場合には次のようにします。

例15

```
10 OPTION ANGLE DEGREES
20 DEF f(x)=sin(x)
30 SET WINDOW -360,360,-4,4
40 DRAW GRID(90,1)
50 FOR x=-360 TO 360
60    PLOT LINES: x,f(x);
70 NEXT x
80 END
```



10 行の OPTION ANGLE DEGREES は，SIN 関数が書かれる行よりも手前の行に書かなければなりません。

40 行で用いている grid(*a*,*b*)は，横軸方向 *a* 間隔，縦軸方向 *b* 間隔の格子です。

2.6.3 PLOT LINES

PLOT LINES は，指定された点を線分で結んで折れ線を描く命令です。点の指定は，*x* 座標と *y* 座標をコンマ(,)で区切って書きます。一つの PLOT LINES 文に複数の点を指定することもできます。その場合，点と点はセミコロンで区切ります。また，点の並びの最後にセミコロンを書くこともできます。直前に実行された PLOT LINES 文がセミコロンで終わるものであった場合には，直前に実行された PLOT LINES 文で最後に指定された点と，最初に指定された点との間も線分で結ばれます。

たとえば，未だ PLOT LINES 文が実行されていないか，または，直前に実行した PLOT LINES 文の末尾にセミコロンがなかったとき，

PLOT LINES: $x_1, y_1; x_2, y_2$

を実行すると，2 点$(x_1, y_1), (x_2, y_2)$ を結ぶ線分が描かれます。これは，

PLOT LINES: $x_1, y_1;$

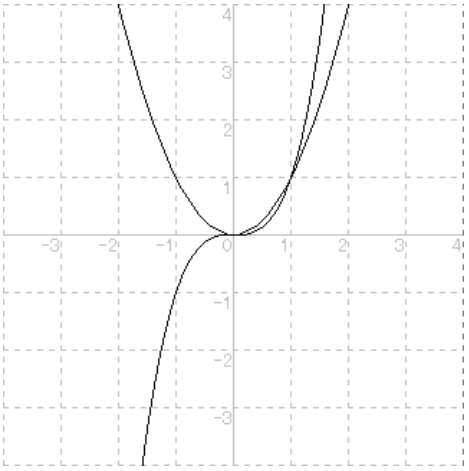
PLOT LINES: x_2, y_2

のように 2 文に分けて書くことができます。

PLOT LINES 文には，点の指定を持たない特別な形式があります。これは，直前に実行した PLOT LINES 文がセミコロンであったときにその効果を取り消すために用いられます。たとえば，次のプログラムのように 2 つの関数のグラフを続けて描く場合に用います。

例16

```
100 DEF f(x)=x^2
110 DEF g(x)=x^3
120 SET WINDOW -4,4,-4,4
130 DRAW GRID
140 FOR x=-4 TO 4 STEP 0.1
150     PLOT LINES: x,f(x);
160 NEXT x
170 PLOT LINES
180 FOR x=-4 TO 4 STEP 0.1
190     PLOT LINES: x,g(x);
200 NEXT x
210 END
```



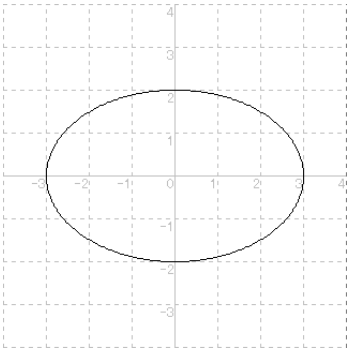
170 行の PLOT LINES がないと，2 点 $(4,f(4)),(-4,g(-4))$が線分で結ばれてしまいます。

2.6.4 媒介変数表示の曲線

媒介変数方程式を用いて表された曲線を描くのは簡単です。次のプログラムは，曲線 $x=3\cos t$, $y=2\sin t$ を描きます。

例17

```
10 OPTION ANGLE DEGREES
20 DEF f(t)=3*COS(t)
30 DEF g(t)=2*SIN(t)
40 SET WINDOW -4,4,-4,4
50 DRAW grid
60 FOR t=0 TO 360
70     PLOT LINES: f(t),g(t);
80 NEXT t
90 END
```

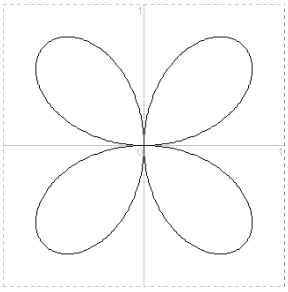


2.6.5 極方程式表示の曲線

極方程式 $r=f(\theta)$は，$x=f(\theta)\cos\theta, y=f(\theta)\sin\theta$ とすれば媒介変数形に直せます。次のプログラムは，正葉線 $r=\sin 2\theta$ を描きます。

例18

```
10 DEF f(t)=SIN(2*t)
20 SET WINDOW -1,1,-1,1
30 DRAW grid
40 FOR t=0 TO 2*PI STEP PI/360
50 PLOT LINES: f(t)*COS(t), f(t)*SIN(t);
60 NEXT t
70 END
```



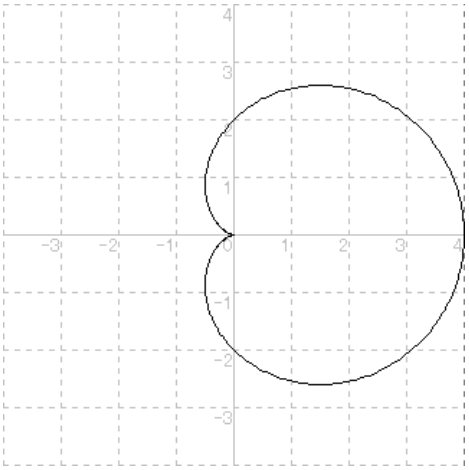
2.6.6 極座標

BASIC では，直交座標から極座標への変換は簡単です。直交座標(x,y)に対する極座標を(r,θ)とすれば，r=SQR(x^2+y^2) , θ=ANGLE(x,y)となります。ただし，$-\pi<\theta\leqq\pi$ です。

次のプログラムは，円 $x=1+\cos t$，$y=\sin t$ 上を動く点 P の動径の長さを 2 乗し，偏角を 2 倍にした点の軌跡を描きます。

例19

```
100 SET WINDOW -4,4,-4,4
110 DRAW grid
120 FOR t=0 TO 2*pi STEP pi/180
130     LET x=cos(t)+1
140     LET y=sin(t)
150     LET r=x^2+y^2
160     LET a=ANGLE(x,y)*2
170     PLOT LINES: r*cos(a),r*sin(a);
180 NEXT t
190 END
```



2.6.7 SET LINE COLOR

複数の曲線を同一の座標平面上に描く場合には曲線ごとに色を変えたくなるかも知れません。線の色を変える命令は SET LINE COLOR です。色は番号で指定します。(仮称)十進 BASIC では，通常，次のように各番号に対する色が割り当てられています。

0 白， 1 黒， 2 青， 3 緑， 4 赤， 5 水色， 6 黄色， 7 赤紫， 8 灰色，9 濃い青，
10 濃い緑，11 青緑， 12 えび茶，13 オリーブ色，14 濃い紫，15 銀色，・・・

たとえば，

SET LINE COLOR 4

を実行すると，それ以後，線は赤で描かれます。

2.6.8 点を描く命令

点を描きたいときには，まず，SET POINT STYLE を実行して点の形を指定します。点の形は次の番号で指定します。指定しないと 3 番の形が使われます。

1・ 2＋ 3 ＊ 4 ○ 5 ×

点の色は SET POINT COLOR で変更できます。

点を打つ命令は PLOT POINTS です。次の形に書きます。

PLOT POINTS: x,y

2.7 数値計算

2.7.1 平方根

加減乗除の演算のみを用いて，正の数の平方根の近似値を求める方法を考えます。

縦の長さが a，横の長さが 1 の長方形と面積の等しい正方形の 1 辺の長さが $\sqrt{a}$ です。そこで，はじめの長方形を，面積を変えないようにして縦と横の長さが近づくように変形していけば，この長方形の辺の長さは $\sqrt{a}$ に近づくはずです。

変形して得られる長方形の縦の長さをもとの長方形の縦，横の長さの平均にとることにします。すなわち，第 k 回目に得られる長方形の縦，横の長さをそれぞれ x_k，y_k とするとき，

$$x_{k+1}=\frac{x_k+y_k}{2}$$

とします。すると，この変形で面積を変えないという条件から，横の長さ y_{k+1} は，

$$y_{k+1}=\frac{a}{x_{k+1}}$$

として定められます。これらの関係式から x_k のみの式を導くと，

$$x_0=a,\quad x_{k+1}=\frac{1}{2}(x_k+\frac{a}{x_k})$$

となります。次のプログラムは，この数列$\{x_k\}$の第 8 項までを求めます。

例20

```
10 INPUT a
20 LET x=a
30 FOR n=1 TO 8
40     LET x=(x+a/x)/2
50     PRINT n,x
60 NEXT n
70 END
```

2.7.2 BASIC の数値

(仮称)十進 BASIC では，数値は十進小数として表現されます。そして，正しい値が 16 桁の十進数で表現できるような演算の結果は正しい結果が得られます (保証はできませんが)。たとえば，SQR(36)の計算結果は正確に 6 になります。また，角の大きさの単位が度である場合には，SIN(30)の計算結果は 0.5 になります。しかし，計算結果の真の値が 16 桁の十進数で表現できないとき，その結果は近似値になります。たとえば，1/3 の計算結果には誤差が含まれます。

(仮称)十進 BASIC では，数値変数は 15 桁の精度を持ちます。有効数字の桁数が 15 桁を越える数値を代入すると，有効数字が 15 桁になるように数値が丸められます。

一方，数値式の計算結果は，16 桁を越える精度を持ちます。たとえば，1/3 の計算結果は，0.333333333333333333333333333333 となって，有効数字が 27 桁あります。

通常，PRINT 文は有効数字が 15 桁になるように計算結果を丸めて表示しますが，オプションメニューー数値で「表示桁数を多く」にチェックを入れると，十進モードではすべての桁を表示するようになります。

数値式の桁数が数値変数の桁数より大きいのは日本工業規格(JIS)の規定によるものですが，ときとしてわかりにくい現象を引き起こします。たとえば，次のプログラムでは，20 行の実行結果は 0 ではありません。これは，丸める桁数の相違によるものです。

```
10 LET A=1/3
20 PRINT 1/3-A
30 END
```

2.7.3 桁落ち

```
10 LET A=1/3
20 PRINT A-0.33333333333333
30 END
```

を実行すると，20 行では A の値が 0.333333333333333 になっているのですから，実行結果

は .000000000000003 になります。けれども，10 行で変数 A に代入した数値は$\frac{1}{3}$であると考える立場からすると，上の結果は有効数字が 1 桁になってしまったものといえます。BASIC では単一の計算の結果の正確さは JIS によって保証されますが，少なくも一方の計算結果が 15 桁では正確に表現できない数値の引き算を行うと，その 2 数の値が接近している場合に著しく有効数字の桁数が減ってしまう現象が起こります。これを**桁落ち**といいます。

関数 $y=f(x)$ の $x=a$ における微分係数 $f'(a)$ は，増分 h を限りなく 0 に近づけたときの平均変化率 $\frac{f(x+h)-f(x)}{h}$ の極限で定義されます。次のプログラムは，$f(x)=\sqrt{x}$ であるときに x=2 における平均変化率が h の値を $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4},\cdots, 10^{-15}$ と変化させたときにどう変化するか調べようとしたものです。h を 0 に近づけたときの極限は，理論的な計算では $\frac{1}{2\sqrt{2}}$ （≒0.353553390593274）ですが，実行結果をみると h が 0 に近づくとかえって正しい値から離れていってしまっています。これは，上に述べた減算に伴う桁落ちによるものです。

例 21

```
10 DEF f(x)=SQR(x)
20 FOR i=1 TO 15
30     LET h=10^(-i)
40     PRINT h, (f(2+h)-f(2))/h
50 NEXT i
60 END
```

実行結果

```
 .1                    .349241122458488
 .01                   .35311255026876
 .001                  .3535092074646
 .0001                 .353548971286
 .00001                .35355294866
 .000001               .3535533464
 .0000001              .353553386
 .00000001             .3535534
 .000000001            .3535534
 .0000000001           .353554
 .00000000001          .35356
 .000000000001         .3536
 .0000000000001        .354
 .00000000000001       .36
 .000000000000001      .4
```

2.7.4 円周率 π

直径 1 の円に内接する正 n 角形の辺の長さの和は $n\sin\frac{180°}{n}$ ですから，正 2^k 角形の辺の長さの和を a_k とすると，$a_k=2^k\sin\frac{180°}{2^k}$ となります。

半角の公式から $\cos\frac{180°}{2^k}=\sqrt{\frac{1}{2}\left(1+\cos\frac{180°}{2^{k-1}}\right)}$ となるので，

$$\cos 90°\ ,\ \cos 45°\ ,\ \cos\frac{45°}{2}\ ,\ \cdots,\ \cos\frac{180°}{2^k}\ ,\ \cdots$$

が順に計算できます。$\sin\frac{180°}{2^k}=\sqrt{1-\cos^2\frac{180°}{2^k}}$ の関係を利用すれば a_k の計算ができるのですが，実際にはうまくいきません。k が大きくなると，$\frac{180°}{2^k}$ が 0 に近くなるので，$\cos^2\frac{180°}{2^k}$ は 1 に近くなります。そのため，$1-\cos^2\frac{180°}{2^k}$ の計算で桁落ちが発生します。

そこで，2 倍角の公式を変形して得られる関係式

$$\sin\frac{\theta}{2}=\frac{\sin\theta}{2\cos\frac{\theta}{2}}$$

を用いて，

$$\sin 90°\ ,\ \sin 45°\ ,\ \sin\frac{45°}{2}\ ,\ \cdots,\ \sin\frac{180°}{2^k}\ ,\ \cdots$$

を計算することにします。

$\cos 90°=0$，$\sin 90°=1$ を出発点としてこれらの値を順に計算し，2^k と $2^k\sin\frac{180°}{2^k}$ を出力するプログラムを作ると次のようになります。

例 22

```
10 LET c=0              ! cos90°
20 LET s=1              ! sin90°
30 FOR k=2 TO 30
40    LET c=SQR((1+c)/2)
50    LET s=s/c/2
60    PRINT 2^k,2^k*s
70 NEXT k
80 END
```

2.7.5 PRINT USING

PRINT 文に書式指定を書くことができます。次の形式で書きます。

PRINT USING 書式指定文字列 : 数値式 , 数値式 , … , 数値式

書式指定文字列には，出力しようとする数値式の個数分の書式を含めます。書式と書式の間は空白文字で区切ります。さらに，書式指定文字列の前後は引用符（"）で括ります。書式を指定すると，小数部は指定された桁数に合わせて四捨五入されます。

書式は，次の形式を推奨します。書式文字の詳細は JIS を参照してください。

正の整数	#######	桁数分の#を書く。右詰めで表示される。
整数（負数にも使える）	-------%	桁数分の負号，最後に%。
正の小数	#######.####	整数部，小数部をそれぞれ#で示す。
小数（負数にも使える）	-------%.####	

書式を作成するとき，整数部の桁数が不足しないように注意してください。特に，負数の場合，負号を出力する分の桁の確保を忘れないように。

例23

```
10 FOR i=0 TO 17
20   LET n=10^i
30    PRINT USING "####################    #.###############":n,(1+1/n)^n
40 NEXT i
50 END
```

実行結果

```
                 1    2.00000000000000
                10    2.59374246010000
               100    2.70481382942153
              1000    2.71692393223589
             10000    2.71814592682522
            100000    2.71826823717449
           1000000    2.71828046931938
          10000000    2.71828169254497
         100000000    2.71828181486764
        1000000000    2.71828182709990
       10000000000    2.71828182832313
      100000000000    2.71828182844545
     1000000000000    2.71828182845769
    10000000000000    2.71828182845891
   100000000000000    2.71828182845903
  1000000000000000    2.71828182845904
 10000000000000000    2.71828182845905
100000000000000000    2.71828182845905
```

例24

```
10 FOR x=3.14159265 TO 3.14159266 STEP 0.000000001
20     PRINT USING "%.#########   ---%.################":x,SIN(x)
30 NEXT x
40 END
```

出力結果

```
3.141592650      0.000000003589793
3.141592651      0.000000002589793
3.141592652      0.000000001589793
3.141592653      0.000000000589793
3.141592654     -0.000000000410207
3.141592655     -0.000000001410207
3.141592656     -0.000000002410207
3.141592657     -0.000000003410207
3.141592658     -0.000000004410207
3.141592659     -0.000000005410207
3.141592660     -0.000000006410207
```

Note. 書式指定の詳細は JIS 規格を参照してください。本稿の記述は，そのほんの一部に過ぎません。

3 アルゴリズムの記述

3.1 IF ～ END IF 構文と IF 文

3.1.1 IF ～ ELSE ～ END IF

例25　係数 a,b,c を入力すると 2 次方程式 $ax^2+bx+c=0$ の解を表示するプログラム

```
10 INPUT a,b,c
20 LET D=b^2-4*a*c
30 IF D>=0 THEN
40     PRINT (-b-SQR(D))/(2*a), (-b+SQR(D))/(2*a)
50 ELSE
60     PRINT "解なし"
70 END IF
80 END
```

このプログラムは，判別式 $D=b^2-4ac$ の値によって処理を分けています。$D \geqq 0$ のときには解の公式を用いて 2 つの解を計算して表示し，そうでないときには，「解なし」と表示します。不等号「≧」は，BASIC では「>=」で代用します。

このように，ある条件が成立するときは～を実行し，そうでないとき～を実行する処理を記述するのに IF ～ ELSE ～ END IF を用います。この構文は，右図の形式で記述します。～の部分には複数の文を書くことができます。FOR～NEXT 構文や IF～END IF 構文のように複数行で構成される命令を書くこともできます。

```
IF 条件 THEN
    ～
ELSE
    ～
END IF
```

なお，条件が成立しないときに実行すべき文がないときは，ELSE を省略して右のように書くことができます。

```
IF 条件 THEN
    ～
END IF
```

3.1.2 IF ～ ELSEIF ～ ELSE ～ END IF

例 25では $D=0$ のとき解が 2 個表示されてしまいます。$D=0$ のときの処理を別扱いにするためには，たとえば，

```
100 INPUT a,b,c
110 LET D=b^2-4*a*c
120 IF D>0 THEN
130     PRINT (-b-SQR(D))/(2*a), (-b+SQR(D))/(2*a)
140 ELSE
150     IF D=0 THEN
160         PRINT -b/(2*a)
170     ELSE
180         PRINT "解なし"
190     END IF
200 END IF
210 END
```

のようにすればよいのですが，これと同じことを次のように書くことができます。

例26

```
100 INPUT a,b,c
110 LET D=b^2-4*a*c
120 IF D>0 THEN
130     PRINT (-b-SQR(D))/(2*a), (-b+SQR(D))/(2*a)
140 ELSEIF D=0 THEN
150     PRINT -b/(2*a)
160 ELSE
170     PRINT "解なし"
180 END IF
190 END
```

3.1.3 IF 文

例 25のプログラムを

```
10 INPUT a,b,c
20 LET D=b^2-4*a*c
30 IF D>=0 THEN PRINT (-b-SQR(D))/(2*a), (-b+SQR(D))/(2*a) ELSE PRINT "解なし"
40 END
```

と書くことができます。このように，条件が成立するとき，成立しないときに実行する文がどちらも一つであり，それらがともに**単純実行文**と呼ばれる種類の文である場合には，

IF 条件 THEN 単純実行文 ELSE 単純実行文

の形の IF 文が利用できます。LET 文, INPUT 文, PRINT 文, SET 文, PLOT 文, RANDOMIZE 文などは単純実行文です。しかし，IF 文は単純実行文ではありません。

なお，条件が成立しないときに実行する文がない場合には，

IF 条件 THEN 単純実行文

の形の IF 文が使えます。たとえば，$D<0$ のとき「解なし」と表示する必要がなければ，

```
10 INPUT a,b,c
20 LET D=b^2-4*a*c
30 IF D>=0 THEN PRINT (-b-SQR(D))/(2*a), (-b+SQR(D))/(2*a)
40 END
```

とします。

3.1.4 条件の書き方

不等式 $a \leqq b$，$a \geqq b$ は，それぞれ，a<=b，a>=b のように書きます。また，a と b とが等しくないこと($a \neq b$)は，a<>b と書きます。

AND，OR，NOT と括弧を用いて複雑な条件を記述することができます。

たとえば，「p かつ q」という条件を BASIC では p AND q と書きます。また，「p または q」を BASIC では p OR q と書きます。

否定を表す NOT は，否定対象の直前に書きます。AND，OR，NOT が混じった条件を書いたときには，NOT が最も優先します。AND と OR とでは AND が優先します。優先順位を変更したいときは括弧を用いることができます。

なお，数学では「$a<x$ かつ $x<b$」のことを $a<x<b$ と書きますが，BASIC ではこういう書き方はできません。

例 27　3 または 8 で割り切れる 3 桁の自然数の総和を求めるプログラム

```
10 LET t=0
20 FOR n=100 TO 999
30    IF MOD(n,3)=0 OR MOD(n,8)=0 THEN
40       LET t=t+n
50    END IF
60 NEXT n
70 PRINT t
80 END
```

3.1.5　　ピタゴラス数

$x^2+y^2=z^2$ となる 3 整数 x,y,z をピタゴラス数といいます。ピタゴラス数を系統的に探すのには，x,y の値を順に変化させて $z=\sqrt{x^2+y^2}$ が整数となるときの x,y,z を出力するようなプログラムを作ればいいでしょう。z が整数であるかどうかは，INT(z)が z と一致するかどうか調べればわかります。次のプログラムは，$1 \leqq x \leqq y \leqq 100$ の範囲でピタゴラス数を探します。

例28

```
10 FOR x=1 TO 100
20    FOR y=x TO 100
30       LET z=SQR(x^2+y^2)
40       IF INT(z)=z THEN PRINT x,y,z
50    NEXT y
60 NEXT x
70 END
```

[Note]　上のプログラムは, N88-BASIC のような古いタイプの BASIC でも動作するでしょう。しかし，正しい結果は得られないかも知れません。JIS Full BASIC では，べき乗や平方根などの計算で，真の値が有効数字の桁数以下の十進数で表現できる場合には真の値が得られることになっているので，上のプログラムで正しい結果が得られます。

3.2　DO～LOOP

3.2.1　　DO～LOOP

DO 文と LOOP 文は対にして用いられ，繰り返しの処理を記述するのに用いられます。

例29 無限ループ

```
10 LET A=0
20 DO
30    LET A=A+1
40    PRINT A
50 LOOP
60 END
```

DO～LOOP の構文を書くと, DO 文と LOOP の間に書かれた文が繰り返し実行されます。上の例では 30 行と 40 行が繰り返し実行されます。

[補足]　上のプログラムを実行したときは，■をクリックするか，実行メニューから中断を選んで実

行を打ち切ってください。

3.2.2 DO WHILE～LOOP

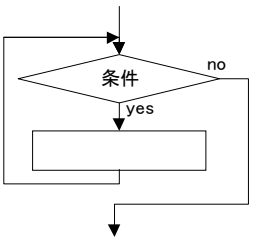
条件が成立する間，繰り返しを実行する構文です。

例30

```
10 LET A=0
20 DO WHILE A<100
30    LET A=A+1
40    PRINT A
50 LOOP
60 END
```

例 29の 20 行を修正して上のようにすると，A の値が 100 になると繰り返しをやめるようになります。

DO WHILE ～ LOOP は次の形式で書きます。この構文の動作を流れ図で表すと右の図のようになります。



DO WHILE 条件

LOOP

この構文では繰り返すべき処理を実行する前に条件を調べますから，最初から条件が成立しない場合には，1 回も繰り返しを実行しません。たとえば，例 30 で 10 行を

10 LET A=100

に変えると 30 行，40 行は 1 回も実行されません。

3.2.3 DO UNTIL ～ LOOP

DO WHILE ～ LOOP は条件が成立する間，繰り返しを実行する構文ですが，DO UNTIL ～LOOP は，繰り返しを終了する条件を指定して繰り返しを実行させる構文です。

例31

```
10 LET A=0
20 DO UNTIL A>=100
30    LET A=A+1
40    PRINT A
50 LOOP
60 END
```

例 31は例 30とまったく同じ意味です。一般に，条件の否定を利用すれば，DO WHILE ～ LOOP と DO UNTIL ～ LOOP は相互に書き換えが可能です。UNTIL *p* は WHILE NOT *p* と同じです。どちらを用いるかは，プログラムを書く人がどちらを向いて思考しているかの相違によります。

3.2.4 素因数分解

DO～LOOP を利用して入力された自然数を素因数分解するプログラムを作ることができます。変数 n に分解すべき自然数を入力するものとします。まず，最初に n が 2 で割れるか

どうか調べ，割れるときは 2 を出力して n を 2 で割ったもので置き換えます。これを n が 2 で割り切れる間，繰り返し実行します。n が 2 で割れなくなったら割る数を 3 にして同様のことを繰り返します。次は割る数を 5 にして同様のことを繰り返せばよいのですが，「次の素数」をコンピュータに答えさせるのは難しいので，割る数は単に 1 を加算していくだけにします。すると，割る数 3 の次は割る数 4 になるのですが，それでも不都合はありません。なぜかというと，すでに 2 で割れるだけ割ってあるからです。

例32 素因数分解

```
110 INPUT n
120 LET f=2
130 DO UNTIL n=1
140     DO WHILE MOD(n,f)=0
150         PRINT f;
160         LET n=n/f
170     LOOP
180     LET f=f+1
190 LOOP
200 END
```

変数 f は割る数(因数，factor)を表します。120 行で最初の割る数 2 をセットしています。140 行から 170 行までの繰り返しで f で割れる間，割ります。割れなくなると(180 行)，f は 1 加算されます。これを 130 行の DO 文と 190 行の LOOP 文とで繰り返すのですが，いつか繰り返しを止めなければなりません。その条件は，n からすべての因数を取り尽くしたときですから，n=1 になります。それが，130 行に書かれた条件です。

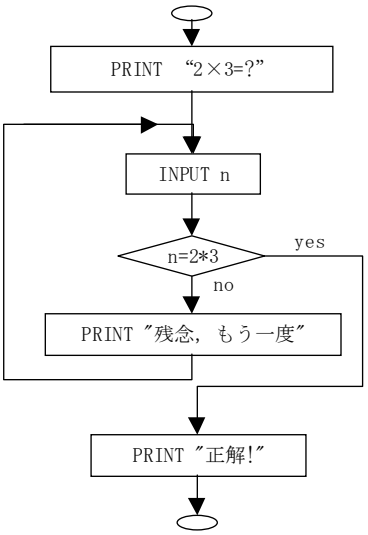
3.2.5 EXIT DO

EXIT DO 文は，DO ～ LOOP の繰り返しを終了して LOOP 文の次の行に制御を移す命令です。DO WHILE ～ LOOP や DO UNTIL ～ LOOP は繰り返しの中身を実行する前にそれを実行すべきかどうか判断します。しかし，その位置では繰り返しを継続すべきかどうかの判断ができない場合があります。そのような場合，EXIT DO 文を用います。

次のプログラムは，正しい答えを入力するまで繰り返し入力を要求します。この場合，入力してからでないと正しいかどうかの判断のしようがありません。

例33

```
10 PRINT "2×3=?"
20 DO
30   INPUT n
40   IF n=2*3 THEN EXIT DO
50   PRINT "残念，もう一度"
60 LOOP
70 PRINT "正解!"
80 END
```



例 33の処理の流れを図に表すと右のようになります。

3.2.6　　基底の変換

正の整数 *n* を入力すると *n* を 2 進法で表示するプログラムを作ります。

たとえば，*n* =11 とすると $11=1\times2^3+0\times2^2+1\times2+1$ ですから，11 を 2 進法で表すと 1011 です。これを下位の桁から順に求めたいときは，右図のようにします。整数 *a* を整数 *b* で割ったときの商を INT(*a*／*b*)，余りを MOD(*a*,*b*)で求められるので，次のプログラムが得られます。

```
2) 11 ･･･1
2)  5 ･･･1
2)  2 ･･･0
2)  1 ･･･1
    0
```

例34

```
10 INPUT n
20 DO
30    LET q=INT(n/2)
40    LET r=MOD(n,2)
50    PRINT r
60    IF q=0 THEN EXIT DO
70    LET n=q
80 LOOP
90 END
```

3.2.7　　ユークリッドの互除法

2 つの正の整数 *a*,*b* に対して，*a*,*b* の最大公約数を gcd(*a*,*b*)で表すことにすると，次の性質が成り立ちます。

a を *b* で割ったときの余りを *r* とするとき，
r = 0 のとき，gcd(*a*,*b*) = *b*
r≠0 のとき，gcd(*a*,*b*) = gcd(*b*,*r*)

この性質を繰り返し利用し，次のようにすると，2 つの整数の最大公約数を求めることができます。これを**ユークリッドの互除法**といいます。

a を *b* で割り，余りを *r* とする。
r=0 であれば，*b* が最大公約数である。
r＞0 であれば，*b*,*r* を新たな *a*,*b* としてこの操作を再度実行する。

この操作は，繰り返しのたびに *b* の値が小さくなっていくので，何回かの繰り返しの後，必ず終了します。

例35　ユークリッドの互除法により最大公約数を求める

```
10 INPUT a,b
20 DO
30    LET r=MOD(a,b)
40    IF r=0 THEN EXIT DO
50    LET a=b
60    LET b=r
70 LOOP
80 PRINT b
90 END
```

繰り返し処理の最後のところで，b,r の値を次の a,b の値とするために 50 行，60 行に示すような順に代入文を用いることに注意してください。

3.3 配列

3.3.1 DIM 文

プログラムのはじめに

DIM A(10)

と書くと，A(1)，A(2)，A(3)，A(4)，A(5)，A(6)，A(7)，A(8)，A(9)，A(10)の計 10 個の変数が用意されます。A(1)，A(2)，A(3)，…，A(10)は，それぞれ，普通の変数として使えます。このとき，この 10 個の変数をまとめて**配列**といって，個々の変数は**添字付き変数**といいます。そして，括弧内を**添字**(そえじ)といいます。

```
10 DIM A(10)
20 LET A(4)=15
30 PRINT A(4)
40 END
```

DIM 文では，配列名と添字の上限を上のような形に書きます。配列名の付け方は普通の変数と同じです。添字の上限には 10，20 のような定数のみを書くことができます。

[Note] 配列の扱いは旧規格の JIS 基本 BASIC から変更されています。特に，配列の添字が 1 から始まることに注意してください。

添字付き変数では，添字に変数を含む式を書くことができます。たとえば，

```
10 DIM B(4)
20 LET i=3
30 LET B(i)=17
40 PRINT B(i)
50 END
```

この機能を利用すると，処理の対象となる変数をプログラムのなかで選ぶことが可能になります。BASIC では，B1，B2，B3 のような変数名を使うこともできます。しかし，この場合には 1, 2, 3 の数字の部分は変数名の一部になっていますから，プログラムで番号を指定して B1, B2, B3 のうちのいずれかを選ぶことはできません。

例 36　フィボナッチ数列

フィボナッチ数列$\{f_n\}$は，

$$f_1=1,\ f_2=1,\ f_n=f_{n-1}+f_{n-2}\quad (n=3,4,5,\cdots)$$

で定義される数列です。配列を利用してフィボナッチ数列のはじめの 20 項を求めるプログラムを書くと，次のようになります。

```
100 DIM f(20)
110 LET f(1)=1
120 LET f(2)=1
130 FOR i=3 TO 20
140     LET f(i)=f(i-1)+f(i-2)
150 NEXT i
160 FOR i=1 TO 20
170     PRINT i,f(i)
180 NEXT i
190 END
```

3.3.2　　分布表を作る

配列は分布表を作るのに利用できます。ここでは，例として 100 点満点の試験の結果を 10 点刻みで分布を調べるプログラムを作成します。

m(0)から m(10)までの 11 個の添字付き変数を用意し，0 点から 9 点までの人数を m(0), 10 点から 19 点までの人数を m(1)，20 点から 29 点までの人数を m(2)，…，で表します。

m(0)から m(10)での 11 個の添字付き変数が使えるようにするために

DIM m(0 TO 10)

のような DIM 文をプログラムのはじめに書きます。　そして，

MAT m=ZER

を実行し，配列ｍの全要素に 0(zero)を代入しておきます。数値を入力すると，どのランクに属するかを表す指標 i を計算し，その指標が示す添字付き変数に 1 を加算します。

すべての数値が入力されたら，負の数を入力することにします。

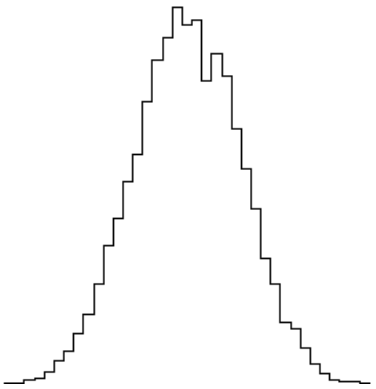
[Note] MAT は，matrix(行列)のはじめの 3 文字をとったもので，行列演算命令の意味です。ZER は zero のはじめの 3 文字です。

例37

```
100 DIM m(0 TO 10)
110 MAT m=ZER
120 DO
130     INPUT n
140     IF n<0 THEN EXIT DO
150     LET i=INT(n/10)
160     LET m(i)=m(i)+1
170 LOOP
180 FOR i=0 TO 10
190     PRINT i*10;"～";i*10+9,m(i)
200 NEXT i
210 END
```

例38　10 個のさいころを振るときの目の数の和を求めるシミュレーションを 10000 回繰り返し，その結果を集計してヒストグラムに表示するプログラム。

```
100 DIM A(10 TO 60)
110 MAT A=ZER
120 FOR k=1 TO 10000
130     LET t=0
140     FOR n=1 TO 10
150         LET t=t+INT(RND*6+1)
160     NEXT n
170     LET A(t)=A(t)+1
180 NEXT k
190 SET WINDOW 9,61,0,1000
200 FOR t=10 TO 60
210     PLOT LINES: t-0.5,A(t); t+0.5,A(t);
220 NEXT t
230 END
```



目の数の和は 10 以上 60 以下になるので, 添字の範囲を 10 から 60 までにした配列Aを用意し，度数分布を求めています。

3.3.3 エラトステネスのふるい

整数の集合から合成数を順序よく取り除き, 素数のみを残す方法で素数の集合を得ることができます。この手法は「エラトステネスのふるい」として古くから知られたものです。

2 以上の整数を順に書き並べた表を用意し，n=2,3,4,…について，n の倍数であるような合成数 n^2, (n+1)n, (n+2)n, … に印をつけると，最後に残るのは素数です。

2 から 1000 までの添字のついた配列 s を用意し, 添字付き変数 s(n) で 整数 n を表します。そして，整数 n に印がついていることを s(n)=1，印がついていないことを s(n)=0 で表現します。

例 39

```
100 DIM s(2 TO 1000)
110 MAT s=ZER
120 FOR n=2 TO 1000
130    IF s(n)=0 THEN
140       PRINT n
150       FOR  k=n^2 TO 1000 STEP n
160          LET s(k)=1
170       NEXT k
180    END IF
190 NEXT n
200 END
```

n より小さいすべての k(ただし 2 以上)について，k の倍数すべてに印をつける処理が終わっていれば，n に印がないことから n は素数であるという判断が下せます。この性質を利用して手順を合理化し，120～190 行の FOR～NEXT ループですべてを処理しています。

150～170 行の FOR～NEXT で n の倍数に印をつけています。k の値を n^2 から順に n ずつ増加させていくのでは k の値が 1000 にならないことがありますが，FOR～NEXT 構文の性質から k の値が 1000 を超えたとき繰り返しを終了するのでこれで問題なく動作します。

3.3.4 配列宣言

配列には 2 次元，3 次元のものも使えます。たとえば，DIM A(2,3) と書くと，A(1,1), A(1,2)，A(1,3)，A(2,1)，A(2,2)，A(2,3)の 6 個の添字付き変数が用意されます。

DIM 文では，複数の配列をまとめて宣言することができます。その場合，各配列宣言はコンマで区切って書きます。たとえば,

```
DIM A(4),B(10),C(20)
```

のように書きます。

Full BASIC では，配列の添字は 1 から始まるものと考えています。配列の添字が 0 から始まるようにしたいときは，最初の配列宣言を書く行よりも上の行に OPTION BASE 0 と書きます。

```
10 OPTION BASE 0
20 DIM A(4)
30 LET A(0)=7
40 MAT PRINT A
50 END
```

添字の下限を 1 または 0 以外の値にしたい場合や,配列ごとに異なる値にしたい場合には次の例に示すような構文を用いることもできます。

```
DIM A(-4 TO 5),B(0 TO 2, 1 TO 3)
```

この配列宣言で A(-4)から A(5)までの 10 個の添字付き変数と，B(0,1)から B(2,3) までの 9 の添字付き変数が用意されます。

3.3.5 2 項係数（パスカルの三角形）

2 次元配列を利用してパスカルの三角形，すなわち，組合せの数 ${}_nC_r$ の表を作ってみましょう。${}_nC_r$ は，次の漸化式を用いて計算することができます。

$r=0$ または $r=n$ のとき ${}_nC_r=1$

$0<r<n$ のとき ${}_nC_r = {}_{n-1}C_{r-1}+{}_{n-1}C_r$

例40

```
100 DIM C(1 TO 10, 0 TO 10)
110 MAT C=ZER
120 FOR n=1 TO 10
130     FOR r=0 TO n
140         IF r=0 OR r=n THEN
150             LET C(n,r)=1
160         ELSE
170             LET C(n,r)=C(n-1,r-1)+C(n-1,r)
180         END IF
190     NEXT r
200 NEXT n
210 MAT PRINT C;
220 END
```

210 行で用いた MAT PRINT 文は，配列に属するすべての添字付き変数の値をまとめて出力します。210 行のようにのように配列名に続けてセミコロンを書いておくと，数値が詰めて出力されます。

3.3.6 MAT 文（入出力）

BASIC は配列を処理するための便利な機能を用意しています。たとえば，MAT INPUT 文と MAT PRINT 文を用いると配列入出力が簡単になります。

例41

```
10 DIM A(5)
20 MAT INPUT A
30 MAT PRINT A
40 END
```

MAT INPUT 文は，配列に数値をまとめて入力します。20 行の MAT INPUT 文の実行時には 5 個の数値をコンマで区切って入力してください。MAT PRINT 文は，配列に属するすべ

ての添字付き変数の値をまとめて出力します。30 行のように配列名の後にセミコロンを書かない MAT PRINT 文を実行すると，数値が一定の位置に出力されます。

MAT READ 文は DATA 文から配列に数値を代入します。 DATA 文には必要な個数だけの数値をコンマで区切って書いておきます。テスト段階のプログラムなどで毎回同じデータを入力したいとき便利です。

例42　最小数を求める

```
10 DIM A(10)
20 DATA 45,61,73,91,43,12,65,34,96,12
30 MAT READ A
40 LET k=1
50 FOR i=2 TO 10
60    IF a(i)<a(k) THEN LET k=i
70 NEXT i
80 PRINT a(k)
90 END
```

3.3.7　　整列（選択ソート）

10 個の数値を入力すると大きさの順に並べ替えて出力するプログラムを作ります。

10 個の数値は a(1)から a(10)までの 10 個の添字付き変数に入力するものとします。まず，a(1)から a(10)までの数値のうちで最小のものを a(1)と交換します。すると，a(1)が一番小さい数になります。次に，a(2)から a(10)までのうちで最小のものを a(2)と交換します。すると，a(2)に 2 番目に小さい数が入ります。以下，同様に，

```
FOR i=1 TO 9
  a(i)から a(10)までのうちで最小のものを a(i)と交換する
NEXT i
```

を実行すれば，配列 a には数値が小さい順に並べ替えられて入ることになります。

例43

```
100 DIM a(10)
110 MAT INPUT a
120 FOR i=1 TO 9
130    LET k=i                          ! k は最小値の番号
140    FOR j=i+1 TO 10
150       IF a(j)<a(k) THEN LET k=j
160    NEXT j
170    LET t=a(i)                       ! t を作業用の変数として用いて
180    LET a(i)=a(k)                    ! a(k)と a(i)の値を交換する
190    LET a(k)=t
200 NEXT i
210 MAT PRINT a;
220 END
```

130～160 行は a(i)～a(10)の最小値の番号を変数 k に取得する処理です。170～190 行では t を作業用の変数として利用して a(i)の値と a(k)の値とを交換しています。

！以降は**行末注釈**です。プログラムの実行には影響しません。プログラムの意味を説明したいときなどに利用します。

なお，このプログラムがどのように動いているのか見たいときは，210 行の MAT PRINT 文を適当な位置に移してみるとよいでしょう。

3.3.8 挿入ソート

再帰的な発想法を用いると，少し異なった種類の整列アルゴリズムを作ることができます。a(1)～a(i-1)がすでに大きさの順に並んでいるとしたら，a(i)をしかるべき位置に挿入して以後の数値を順に後ろにずらしてやれば a(1)～a(i)が大きさの順に並びます。そこで，i=2 から i=10 まで順に上に述べた処理を繰り返せば大きさの順に並ぶことになります。

例44

```
100 DIM a(10)
110 MAT INPUT a
120 FOR i=2 TO 10
130    LET b=a(i)                    ! a(i)を b に退避する
140    LET j=i                       ! b を挿入すべき位置を j に求める
150    DO WHILE j>1 AND a(j-1)>b
160       LET a(j)=a(j-1)            ! データを後方にずらす処理も同時に行う
170       LET j=j-1
180    LOOP
190    LET a(j)=b
200 NEXT i
210 MAT PRINT a;
220 END
```

130～180 行で a(i)の値を挿入すべき位置を探しています。そのとき，同時に各変数の値を後ろにずらしています。

JIS Full BASIC の AND と OR には特徴的な性質があります。p AND q では，p が偽であるとわかったときには q の真偽を調べません。また，p OR q では，p が真であるとわかったときには q の真偽を調べません。150 行の AND 演算では，j>1 と a(j-1)>b を調べる順序が重要です。150 行では j=1 になることがありますから，j>1 よりも先に a(j-1)>b を調べると添字が範囲外となってエラーになってしまいます。

3.4　例外状態処理

3.4.1　　実行時エラー（例外）

PRINT を PLINT と綴ったり，FOR に対応する NEXT がなかったりするとコンパイラはエラーを報告します。これは文法上の誤り（**構文誤り**）です。

x の値が 0 であるときに 1/x を計算すれば当然エラーになります。また，BASIC には扱いうる最大の数が定められていて，計算結果の絶対値がその値(MAXNUM)を超える場合にもエラーになります（桁あふれ）。このように実行時に起こるエラーを特に**例外**と呼んで，文法上の誤りと区別します。

ゼロ除算のエラーならばあらかじめ除数が 0 であるかどうか調べておけば防げますが，桁あふれのエラーを予測するのは実質的に不可能です。このような場合にはエラーが起きてから対処するしか方法がありません。

3.4.2　　WHEN EXCEPTION 構文

WHEN EXCEPTION 構文は，実行時エラー(例外)が発生したときの処理を記述する構文です。次の形に書きます。

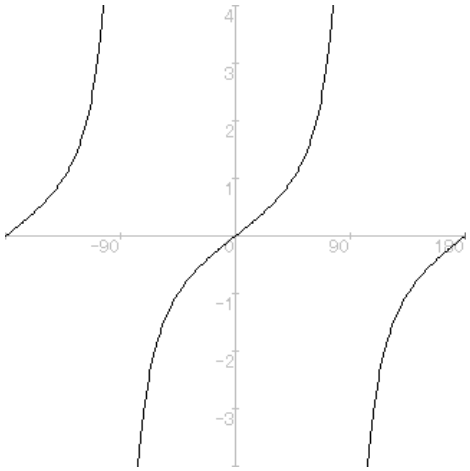
```
WHEN EXCEPTION IN
   例外を起こす可能性のある処理
USE
   例外が起きたとき実行すべき処理
END WHEN
```

WHEN EXCEPTION IN 行と USE 行の間に書かれた文で例外が起こると，その文の実行を中断して USE 行の次の行に制御を移します。一方，WHEN EXCEPTION IN 行と USE 行の間に書かれた文が例外を発生することなく実行されたときは，USE 行と END WHEN 行の間に書かれた文は実行されません。

この構文は関数のグラフを描く場合には不可欠なものといえます。

例46 関数 $y = \tan x$ のグラフ

```
100 OPTION ANGLE DEGREES
110 DEF f(x)=TAN(x)
120 SET WINDOW -180,180,-4,4
130 DRAW AXES(90,1)
140 FOR x=-180 TO 180
150     WHEN EXCEPTION IN
160         PLOT LINES: x,f(x);
170     USE
180         PLOT LINES
190     END WHEN
200 NEXT x
210 END
```



なお，180 行の PLOT LINES がないと，たとえば，2 点(89,TAN(89))，(91,TAN(91)) を結ぶ線分が描かれてしまいます。

4 プログラム分割

4.1 外部関数定義

4.1.1 プログラム分割の必要性

例 28を実行すると，6,8,10 のような無意味なピタゴラス数も出力します。これを防ぐためには x,y の最大公約数が 1 でないものを除外すればいいのですから，例 28のプログラム中に最大公約数を求めるプログラム例 35を埋め込んで

```
100 FOR x=1 TO 100
110     FOR y=x TO 100
120         LET a=x
130         LET b=y
140         DO
150             LET r=MOD(a,b)
160             IF r=0 THEN EXIT DO
170             LET a=b
180             LET b=r
190         LOOP
200         IF b=1 THEN
210             LET z=SQR(x^2+y^2)
220             IF INT(z)=z THEN PRINT x,y,z
230         END IF
240     NEXT y
250 NEXT x
260 END
```

とすればいいでしょう。しかし，このようなプログラムを作るのは，実際には容易ではありません。たとえば，例 28のプログラムが変数名として x,y,z でなく a,b,c を用いていたとしたら，合体させたプログラムは正常に動作しません。

4.1.2 外部関数定義

BASIC には多くの関数が組込み関数として用意されていますが，それで十分とはいえません。DEF 文は利用者が関数を定義する機能を提供していますが，DEF 文では数値式で書けるような関数しか定義できません。BASIC には，より複雑な処理をしなければ求められないような関数でも定義できる機能が用意されています。

前項に示したプログラムは，2 数 a,b の最大公約数を GCD(a,b) と書くことにすればもっとわかりやすく書くことができます。そのために**外部関数定義**というものを使います。

BASIC のプログラムでは END 行までの部分を**主プログラム**といいます。そして，END 行以降には**外部手続き**と呼ばれるものが続きます。外部関数定義も外部手続きの一種です。

例 47(次ページ)は，200 行から 280 行までの部分が GCD 関数を定義する外部関数定義です。外部関数定義部は EXTERNAL FUNCTION 行で始まり，END FUNCTION 行で終わります。EXTERNAL FUNCTION 行には，関数名と**引数**(ひきすう)を書きます。引数は括弧でくくって書きます。引数が 2 個以上ある場合にはそれらをコンマで区切って書きます。

例47

```
100 DECLARE EXTERNAL FUNCTION GCD
110 FOR x=1 TO 100
120    FOR y=x TO 100
130       IF GCD(x,y)=1 THEN
140          LET z=SQR(x^2+y^2)
150          IF INT(z)=z THEN PRINT x,y,z
160       END IF
170    NEXT y
180 NEXT x
190 END
200 EXTERNAL FUNCTION GCD(a,b)
210 DO
220    LET r=MOD(a,b)
230    IF r=0 THEN EXIT DO
240    LET a=b
250    LET b=r
260 LOOP
270 LET GCD=b
280 END FUNCTION
```

主プログラムと外部手続きは**プログラム単位**と呼ばれます。外部関数を利用するプログラム単位には，その外部関数の名前を関数名として用いることの宣言文を書きます。上のプログラムでは，主プログラムで外部関数として GCD を用いるために 100 行のような宣言文を書いています。このような宣言を書くことで，外部関数の名前と同じ組込み関数があったとしても正しく動作させることが可能となります。

宣言文は，実際に関数を使う行よりも上の行に書きます。文法的には，DECLARE EXTERNAL FUNCTION に続けて目的の関数名を書くだけです。

[Note] (仮称)十進BASIC の現バージョンでは組込み関数名と一致しない外部関数を用いる場合には宣言文を省略できますが，将来のバージョンでは組込み関数を増やすかも知れないので，宣言文は必ず書いてください。また，将来のバージョンでは外部関数宣言の省略を許さないように文法を変更することも検討しています。

上のプログラムでは，GCD 関数を 130 行で利用しています。これを関数の呼び出しといいます。また，関数を呼び出すとき書く引数（上のプログラムでは x と y）を実引数と呼びます。GCD 関数が呼び出されると，変数 a,b に実引数の値が代入されて，関数定義部の各行が順に実行されます。関数の値は，270 行にあるような関数名に代入する形の LET 文で決定されます。

プログラム単位は，変数名に関して独立した存在です。異なるプログラム単位に同じ名前の変数があってもコンピュータの内部では別の変数です。たとえば，例 47の主プログラムの変数 x,y,z を a,b,c に書き換えたとしても，プログラムは正常に動作します。

4.1.3　　3 数の最大公約数

外部関数定義を利用すれば 3 数の最大公約数を求めるのは簡単です。次のようにすればいいのです。

```
100 DECLARE EXTERNAL FUNCTION GCD
110 INPUT a,b,c
120 PRINT GCD(GCD(a,b),c)
190 END
200 EXTERNAL FUNCTION GCD(a,b)
    例 47と同じ
280 END FUNCTION
```

4.1.4　　関数の再帰呼び出し

n!に関して漸化式 0!=1，*n*!=*n*･(*n*-1)! が成立します。ですから，7!は 6!に 7 をかければ求められます。そして，6!は 5!に 6 をかければ求められます。このように数値を逆にたどっていくと，最後は 0!=1 という式に行き着いて問題が解決します。

このように，同種の，しかし，より規模の小さい問題に帰着させることを繰り返し実行して問題を解決する手法を**再帰**と呼びます。

BASIC では，関数を定義するときに自分自身を利用することができます。次のプログラムでは，階乗（factorial）を計算する関数 FACT の定義のなかで自分自身を利用しています。コンピュータのプログラミングでは，関数の定義のなかで自分自身を呼び出すことを**再帰呼び出し**といいます。

例48 階乗の計算

```
10 DECLARE EXTERNAL FUNCTION FACT
20 INPUT n
30 PRINT FACT(n)
40 END
100 EXTERNAL FUNCTION FACT(n)
110 IF n=0 THEN
120     LET FACT=1
130 ELSE
140     LET FACT=n*FACT(n-1)
150 END IF
160 END FUNCTION
```

BASIC で関数の再帰呼び出しがうまく機能するのは，関数が呼び出されるごとに変数が新たに割り当てられるからです。たとえば，140 行は

```
LET FACT=FACT(n-1)*n
```

としても正しく動作しますが，FACT(n-1)の計算の途中で n の値が変化してしまうと正しく動作しないはずです。つまり，関数 FACT が呼び出されるごとに，別の場所にその回の呼び出しにだけ通用する変数 n が新たに割り当てられているのです。

4.1.5　　ユークリッドの互除法

最大公約数を求めるユークリッドの互除法も再帰的な算法の典型例です。BASIC ではこの算法を次のように書くことができます。

例 49

```
10 DECLARE EXTERNAL FUNCTION GCD
20 INPUT a,b
30 PRINT GCD(a,b)
40 END
100 EXTERNAL FUNCTION GCD(a,b)
110 LET r=MOD(a,b)
120 IF r=0 THEN LET GCD=b ELSE LET GCD=GCD(b,r)
130 END FUNCTION
```

4.2　外部絵定義と外部副プログラム

4.2.1　　外部絵定義（PICTURE）

Full BASIC には円を描く命令は用意されていませんが，円を描く命令を定義して利用する手段が用意されています。それを**絵定義**と呼びます。

例50

```
10 DECLARE EXTERNAL PICTURE circle
20 OPTION ANGLE DEGREES
30 SET WINDOW -8,8,-8,8
40 DRAW circle
50 DRAW circle WITH SCALE(2)
60 DRAW circle WITH SCALE(3,2)
70 DRAW circle WITH SCALE(3,2)*SHIFT(3,4)
80 DRAW circle WITH SCALE(5,3)*ROTATE(60)
90 END
100 EXTERNAL PICTURE circle
110 OPTION ANGLE DEGREES
120 FOR t=0 TO 360
130     PLOT LINES:COS(t),SIN(t)
140 NEXT t
150 END PICTURE
```

100 行から 150 行までの部分を**外部絵定義**といいます。外部絵定義の最初の行には EXTERNAL PICTURE に続けて絵名を書きます。上のプログラムでは，circle が絵名です。もし必要があれば外部関数定義の場合と同様に引数部を書くこともできます。

絵 circle は原点を中心とする半径 1 の円を描きます。正確にいえば正 360 角形を描いているだけですが，コンピュータのディスプレーでは円に見えると思います。

プログラム単位は OPTION 文に関しても独立です。OPTION ANGLE DEGREES を主プログラムに書いても外部絵定義では角の大きさの単位は変更されません。

絵を実行するのに DRAW 文を用います。DRAW 文では絵を描くときに原点を中心とする拡大・縮小や回転，あるいは平行移動などの変形を指定することができます。

SCALE(*a*,*b*)は原点を中心とする *x* 軸方向に *a* 倍，*y* 軸方向に *b* 倍の拡大です。*a*,*b* は負の数でも差し支えありません。*a*＝*b* の場合には，SCALE(*a*)のように書くこともできます。

SHIFT(*a*,*b*)は，*x* 軸方向に *a*，*y* 軸方向に *b* の平行移動です。

ROTATE(α)は，原点を中心とする角αの回転です。

変形は＊で結合することで合成することもできます。合成した変形は左から順に実行されます。たとえば，SCALE(3,2)*SHIFT(3,4)は拡大してから平行移動します。

4.2.2 外部副プログラム

外部副プログラムの機能はほとんど外部絵定義のなかに含まれています。ただし，**副プログラム**では図形を変形して描くことはできません。

例51

```
100  DECLARE EXTERNAL SUB circle
120  SET WINDOW -4,4,-4,4
130  CALL circle(1,-1,2)
140  END
200  EXTERNAL SUB circle(a,b,r)
210  OPTION ANGLE DEGREES
220  FOR t=0 TO 360
230      PLOT LINES: a+r*COS(t),b+r*SIN(t);
240  NEXT t
250  END SUB
```

例 51では，点(*a*,*b*)を中心とする半径 *r* の円を描く副プログラム circle(*a*,*b*,*r*)を定義しています。副プログラムを呼び出すのに CALL 文を用います。副プログラムに引数があるときは関数と同様の形式で引数を記述します。

4.2.3 変数引数

絵および副プログラムには関数定義と異なる特性があります。それがこれから説明する**変数引数**です。

例52

```
10 DECLARE EXTERNAL SUB double
20 LET a=3
30 CALL double(a)
40 PRINT a
50 END
100 EXTERNAL SUB double(x)
110 LET x=x*2
120 END SUB
```

例 52で副プログラム double は実引数として書いた変数の値を 2 倍にして返します。副プログラムでは，実引数に変数を書くと，副プログラムの実行中，実引数の変数に割り当てられた領域を引数の領域として使います。したがって，副プログラムの実行中に引数の値を変更すると，実引数の値が変化します。

なお，実引数が変数そのものではないとき，たとえば，30 行を

```
30 CALL double(2*a)
```

とした場合には，副プログラムに 2*a の計算結果の数値が引き渡されるだけです。

4.2.4 不定方程式の整数解

a,*b*,*c* が整数の定数であるとき，方程式 $a\,x + b\,y = c$ の整数解を探すという問題を考えてみ

ます。なお，この型の方程式は 1 つ解を持てば無数の解を持つことになるので，ここでは解をひとつ求めればよいものとします。

$b=0$ のときの解は明らかです。c が a で割り切れれば $x=\frac{c}{a}$ が解です。y の値は何でもよいのですが，たとえば，$y=0$ とでもしておけばよいでしょう。反対に c が a で割り切れなければ解はありません。

$b\neq 0$ のときは，a を b で割ったときの商を q，余りを r とします。$a=bq+r$ ですから，はじめの方程式を $b(qx+y)+rx=c$ と書き換えることができます。したがって，方程式 $bu+rv=c$ の解が求まれば，$x=v$，$y=u-qv$ が求める解ですし，$bu+rv=c$ に解がなければ解はありません。

ここで，$bu+rv=c$ の解は必ず確定します。なぜかというと，$ax+by=c$ と $bx+ry=c$ を比較したとき，y の係数は確実に 0 に近づいているからです。y の係数は整数ですから，何回かの繰り返しの後，必ず 0 になります。

例53

```
10 DECLARE EXTERNAL SUB solve
20 INPUT a,b,c
30 WHEN EXCEPTION IN
40    CALL solve(a,x,b,y,c)
50    PRINT x,y
60 USE
70    PRINT "解なし"
80 END WHEN
90 END
100 EXTERNAL SUB solve(a,x,b,y,c)
110 IF b=0 THEN
120    IF MOD(c,a)=0 THEN
130       LET x=c/a
140       LET y=0
150    ELSE
160       CAUSE EXCEPTION 999
170    END IF
180 ELSE
190    LET q=INT(a/b)
200    LET r=MOD(a,b)
210    CALL solve(b,u,r,v,c)
220    LET x=v
230    LET y=u-q*v
240 END IF
250 END SUB
```

160 行の CAUSE EXCEPTION 文は指定した例外番号(EXTYPE)の実行時エラー(例外)を起こします。 1 ～999 の例外番号は利用者用として予約されています。例外を起こした文が WHEN EXECPTION 構文で囲まれていなければ，例外は呼び出し元に伝達されます。

5 データの入力と出力

5.1 文字列

5.1.1 文字列定数

プログラム中に文字列を記述する場合，その前後を" "でくくります。" "でくくった文字列を文字列定数といいます。文字列定数に"を含めるときは，"を重ねて""とします。

例 54

```
10 PRINT "Say ""Hello!"""
20 END
```

実行結果

```
Say "Hello!"
```

5.1.2 文字列変数

文字列を値としてもつ変数を文字列変数といいます。BASIC では，文字列変数には，末尾に$(ドル記号)を持つ名前を用いる約束になっています。

5.1.3 文字列連結

文字列には，連結の演算が定義されています。文字列の連結を文字列と文字列の間に&を書いて表します。

例 55

```
10 LET s$="Brown"
20 LET s$="Mr. " & s$
30 PRINT s$
40 END
```

実行結果

```
Mr. Brown
```

5.1.4 STR$関数

組込関数 STR$(*x*)は数値 *x* を文字列化します。数値の前後には空白を含めません。

次のプログラムは，入力された数値を 2 進法に変換し上位桁が左にくるように表示します。

例 56

```
10 INPUT n
20 LET s$ = ""
30 DO UNTIL n=0
40    LET r = MOD(n,2)
50    LET s$ = STR$(r) & s$
60    LET n = (n-r)/2
70 LOOP
80 PRINT s$
90 END
```

実行結果

```
? 11
1011
```

文字を 1 個も含まない文字列を空文字列といいます。空文字列は""で表されます。

上のプログラムは，最初，s$を空文字列としておき，入力された数値を 2 で割った余りを求めるたびに，それを文字列化し，s$の左に連結しています。

5.2 DATA 文

5.2.1 READ 文と DATA 文

BASIC にはプログラム中にデータを記述する文法が用意されています。

DATA 文には，コンマで区切ってデータを記述します。

変数にデータを読み込む命令は READ 文です。READ 文に複数の変数を書くこともできます。その場合，変数名はコンマで区切って書きます。変数には，DATA 文に書かれた順に代入されていきます。

例 57

```
10 DATA 1,2,3,4,5,6,7,8,9,10
20 FOR i=1 TO 5
30     READ a,b
40     PRINT a,b
50 NEXT i
60 END
```

実行結果

```
1             2
3             4
5             6
7             8
9             10
```

すべてのデータを 1 行に記述しきれないときは，複数行に分けて記述することができます。そのとき，行の末尾にコンマを書く必要はありません（書くと，文法誤りになります）。

たとえば，上の例の DATA 文は，

```
10 DATA 1,2,3,4
12 DATA 5,6,7,8
14 DATA 9,10
```

のように 3 行に分けて書くこともできます。

5.2.2 READ IF MISSING THEN

上の例では，データは 2 個セットで 5 組あるので，FOR～NEXT ループを 5 回くり返してデータを読んでいますが，データ個数が変化したり，データ個数自体がいくつあるか分からないときは面倒です。

Full BASIC には，不定個数のデータを読んで処理するための命令が用意されています。

READ 文を DO～LOOP の中で使うとき，READ と変数名の間に IF MISSING THEN EXIT DO：を書くと，データがなくなったときに DO～LOOP の繰り返しを抜けます。

次の例は，DATA 文に書かれたデータの平均を計算するプログラムです。

例 58

```
100 DATA 34,71,85,80,58,49,52,13
110 LET n=0
120 LET t=0
130 DO
140     READ IF MISSING THEN EXIT DO: x
150     LET t=t+x
160     LET n=n+1
170 LOOP
180 PRINT t/n
190 END
```

5.2.3 RESTORE 文

RESTORE 文は，READ 文が再び先頭のデータから読むようにします。

次のプログラムは，各データについて，平均との差（偏差）を計算します。偏差の計算には平均があらかじめ計算されていることが必要なので,データを 2 回,読むことが必要です。

例 59

```
100 DATA 34,71,85,80,58,49,52,13
110 LET n=0
120 LET t=0
130 DO
140    READ IF MISSING THEN EXIT DO: x
150    LET t=t+x
160    LET n=n+1
170 LOOP
180 LET m=t/n
190 RESTORE
200 FOR i=1 TO n
210    READ x
220    PRINT x-m
230 NEXT i
240 END
```

5.3 ファイル

5.3.1 OPEN 文と CLOSE 文

BASIC では，テキストファイルに書かれたデータを読み込んで処理したり，結果をテキストファイルに出力したりすることができます。テキストファイルは，Windows のメモ帳やワープロソフトなどで読み書きすることのできる最も単純な形式のファイルです。

ファイルを使うとき，ファイルを経路という仮想的な存在に割り当てます。経路には，正の整数で識別番号を付加します。通常,1や2などの若い番号から順に用いる習慣があります。

ファイルを経路に割り当てる命令はOPEN文です。OPEN文は,

OPEN #経路番号 : NAME ファイル名

の形式に書きます。経路番号は数値式で指定します。ファイル名は文字列式です。

ファイルの使用が終わったら,

CLOSE #経路番号

の形式のCLOSE文を実行します。

5.3.2 ファイルからの入力

INPUT 文の入力応答と同じ内容をテキストファイルに書いておくと，ファイルから連続的にデータを入力することができます。通常，INPUT 文の 1 回の実行に対応する入力応答がテキストファイルでの 1 行に対応しますが，行末にコンマを書くことで，1 回分の入力応答を複数行に分けて書くこともできます。

ファイルからの入力を行うとき，INPUT 文は，経路番号を指定して,

INPUT #経路番号： 変数，変数，･･･，変数

の形に書きます。

たとえば, A ドライブに作成した DATA1.TXT という名称のファイルからデータを入力する場合は，次のようなプログラムを作成します。（ファイル名の指示の仕方は OS により異なります。"A:¥DATA1.TXT"は Windows のファイル名です。ドライブ名の部分は，実際に利用可能なドライブに変更してください。）

例 60

```
10 OPEN #1: NAME ″A:¥DATA1.TXT″
20 FOR i=1 TO 4
30    INPUT #1:s$,n
40    PRINT s$,n
50 NEXT i
60 CLOSE #1
70 END
```

DATA1.TXT には, 30 行の入力に対応する文字列定数と数値定数をコンマで区切って書いたものを 4 行分，用意します。たとえば,

```
″Yamada″, 78
″Nakano″, 90
″Sakai″, 100
″Okano″, 76
```

なお，コンマの直後で改行して 8 行で書くこともできます。

IF MISSING THEN EXIT DO を書いた INPUT 文を DO～LOOP で用いると, データ件数が不定な入力に対応したプログラムが作れます。この場合，IF MISSING … は経路番号の後にコンマを打って続け，EXIT DO と変数名の区切りにはコロンを書くのが約束です。

例 61

```
10 OPEN #1: NAME ″A:DATA1.TXT″
20 DO
30    INPUT #1, IF MISSING THEN EXIT DO: s$,n
40    PRINT s$,n
50 LOOP
60 CLOSE #1
70 END
```

5.3.3 ファイルへの出力

ファイルに書き出す方法には，上書きと追記の 2 通りがあります。上書きは，もし，指定されたファイルに既存の内容があれば, それを消去して新たな内容で書き換える書き出し方です。追記は，既存のテキストの末尾に付け加える形で書き出します。

上書きで書き出すときは，OPEN 文の実行直後に ERASE 文を実行します。なお，指定された名称のファイルが存在しなければ，新たに作成されますが，空のファイルに対して ERASE を実行しても問題はありません。

例 62

```
10 OPEN #2:NAME "A:DATA2.TXT"
20 ERASE #2
30 FOR x=1 TO 10
40    PRINT #2: x,SQR(x)
50 NEXT x
60 CLOSE #2
70 END
```

テキストファイルへの書き出しは，通常の画面への出力と同じ形式で出力されます。つまり，テキスト出力のウィンドウの内容をファイルに保存したのと同じになります。項目間には空白が出力されるので，再び BASIC で読み込むのには対応しませんが，区切り文字にスペースを指定すれば，Excel などの表計算ソフトで読むことができます。

追記書き出しの場合は，20 行を

```
20 SET #2: POINTER END
```

に変更します。

5.3.4 内部形式ファイル

項目が 2 個以上あるデータをテキストファイルとして書き出すと再び BASIC で読み込むことができませんが，内部形式を利用すれば，BASIC で書き出したデータを再び BASIC で読み込むことができます。内部形式は，OPEN 文の末尾に ,RECTYPE INTERNAL を付加し，出力に PRINT 文の代わり WRITE 文，入力に INPUT 文の代わりに READ 文を用います。（仮称）十進BASICの内部形式ファイルの実体は，コンマ区切りのテキストファイル(CSV)なので，表計算ソフトなどとの間のデータのやり取りに用いることができます。

例 63　内部形式での書き出し

```
10 OPEN #1:NAME "A:DATA2.CSV" ,RECTYPE INTERNAL
20 ERASE #1
30 FOR x=1 TO 10
40    WRITE #1: x,SQR(x)
50 NEXT x
60 CLOSE #1
70 END
```

例 64　内部形式ファイルの読み込み

```
10 OPEN #1: NAME "A:DATA2.CSV" ,RECTYPE INTERNAL
20 DO
30    READ #1, IF MISSING THEN EXIT DO: x,y
40    PRINT x,y
50 LOOP
60 CLOSE #1
70 END
```

なお，READ文，WRITE文を内部形式でないファイルで用いることもできますが，その場合の効果は，INPUT文，PRINT文と同じです。

付録

付録１　Full BASIC の予約語

以下の綴りは予約語です。変数名や関数名として使えません。

NOT，ELSE，PRINT，REM，PI，RND，MAXNUM，TIME，DATE，EXTYPE，EXLINE，ZER，CON，IDN，TRANSFORM

付録２　その他の重要な命令

1. 制御

EXIT FOR　　FOR～NEXT ループから抜ける。（EXIT DO と似ている）

GOTO 行番号　　指定された行番号をもつ行に制御を移す。

☆ EXIT DO，EXIT FOR は，それぞれ，最も内側の DO～LOOP，FOR～NEXT から脱出します。内側のループを飛び越して外側のループを抜けたいときは GOTO 文を用います。

2. グラフィックス

GET POINT: x,y　　マウスでクリックするまで待ち, その点の座標を変数 x,y に代入する。

PLOT AREA: x_1,y_1 ; x_2,y_2 ; … ; x_n,y_n

(x_1,y_1)，(x_2,y_2)，…，(x_n,y_n) を結ぶ折れ線で囲まれる領域の内部を塗りつぶす。なお，(x_n,y_n)と(x_1,y_1)の間は自動的に結ばれる。

SET AREA COLOR *n*　　領域色を *n* にする。（SET LINE COLOR と似ている）

付録３　BASIC のおもな組込み関数（数値関数のみ）

一般

ABS(*x*)	*x* の絶対値
SQR(*x*)	*x* の非負の平方根
INT(*x*)	*x* を超えない最大の整数
MOD(*x*,*y*)	*x* を *y* で割った余り
CEIL(*x*)	*x* 以上の最小の整数
IP(*x*)	*x* の整数部分
FP(*x*)	*x* の小数部分
ROUND(*x*,*n*)	*x* を小数点以下 *n* 桁に丸めた値
SGN(*x*)	*x* の符号(sign)。*x*>0 のとき SGN(*x*)=1, SGN(0)=0, *x*<0 のとき SGN(*x*)=-1

指数・対数

EXP(*x*)	指数関数
LOG(*x*)	自然対数
LOG2(*x*)	２を底とする対数
LOG10(*x*)	常用対数

三角関数

SIN(*x*)	正弦関数　sine
COS(*x*)	余弦関数　cosine
TAN(*x*)	正接関数　tangent
CSC(*x*)	余割関数　cosecant
SEC(*x*)	正割関数　secant

COT(*x*)	余接関数　cotangent
ASIN(*x*)	$\sin\theta = x$　となる θ。　ただし，$-\pi/2 \leqq \theta \leqq \pi/2$
ACOS(*x*)	$\cos\theta = x$　となる θ。　ただし，　$0 \leqq \theta \leqq \pi$
ATN(*x*)	$\tan\theta = x$　となる θ。　ただし，$-\pi/2 < \theta < \pi/2$
ANGLE(*x*,*y*)	原点と点(*x*,*y*)を結ぶ半直線が *x*軸の正の向きとなす角。
	$-\pi <$ ANGLE(*x*,*y*) $\leqq \pi$
双曲線関数	
TANH(*x*)	双曲線正接関数
SINH(*x*)	双曲線正弦関数
COSH(*x*)	双曲線余弦関数
乱数	
RND	疑似乱数　0≦RND＜1
時刻	
TIME	その日の午前 0 時からの経過秒数
その他	
PI	円周率πの近似値
MAXNUM	表現可能な最大の正の数
EPS(*x*)	*x* の直前, 直後の数との差, および機械最小値のうちの最も大きい数値。
MAX(*a*,*b*)	*a*,*b* のうち大きい方
MIN(*a*,*b*)	*a*,*b* のうち小さい方

付録 4　基本 BASIC との相違

（１）配列添字の下限

旧規格の基本 BASIC では配列の添字の下限は 0 ですが，Full BASIC では添字は 1 からはじまるのを仮定します。添字 0 が必要な場合には，dim 文より手前の行に OPTION BASE 0 を追加してください。

（２）暗黙の配列宣言

Full BASIC では，すべての配列に宣言を要求します。暗黙の配列宣言の機能はありません。

参考書

1. BASIC の成立過程について

「Back to BASIC」J.G.Kemeny and T.E.Kurtz 著,松田健生 訳，啓学出版

2. BASIC の数学への応用

「数学とコンピュータ１・２」佐藤公作・白石和夫・高橋雅信 著，森北出版

「コンピュータによるグラフィックス」片桐重延・白石和夫 著，東京電機大学出版局

3. BASIC によるプログラミング

「Full BASIC による算法通論」森口繁一・伊理正夫・武市正人 著，東京大学出版会

４. JIS Full BASIC 規格の全文

「JIS 電子計算機プログラム言語 Full BASIC」，JIS X 3003，日本規格協会