1 qbgraph の概要

1.1 使用例

\qbgraph.sty は，高等学校で数学を教える立場で，LaTeX の便利さを実感しながらも，その picture 環境の使い勝手の悪さに長年我慢してきた数学教師の勝手マクロ集です。

世の中には metapost[*1]や PSTrick[*2]，当に初等中等教育での使用を目的に作られた emath シリーズ[*3]など色々と便利なものが存在しています。ただ何れも、Perl などの別のスクリプト言語やソフトウェアを必要としたり，typeset[*4]の途中にただでさえログファイルや DVI ファイルなどの関連ファイル＋[*5]が量産される TeX で，さらに雑多なファイルをフォルダ内に置いてしまう等，整理が不得手な僕としては，個人的に面倒だなあと思い，実用には至りませんでした。

ちょっと使う，日常的に使う，週に何枚も作る，そういう使い方だと面倒なわけです。第一，現場で使うプリントは B4 一枚が基本だし，手間がかからず散らからないものが良い。できれば，基本的な LaTeX セットとちょっとしたマクロだけで済ませたい。というわけで，マクロ群\qbgraph.sty を作成しました。「車輪の再発明」でなければ良いのですが。

また、指導要領の改訂で、高校数学に中学校で習っていた幾何の一部が廻って来ています。そこで特に点や線、円、角度といった図形を楽に扱いたいというのも強い動機です。

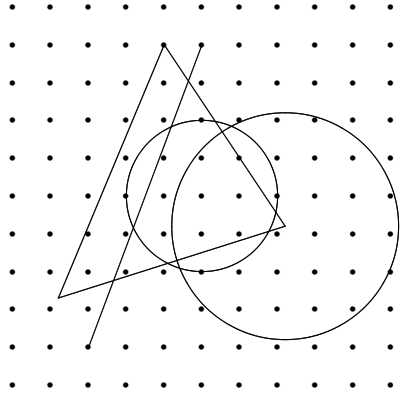
ネット上のあちらこちらに存在する掲示板では，「任意の傾きの直線はどうすれば引けるか？」などの図形に関する質問が散見できます。回答はいずれも，上述の外部依存の解決策です。それはそれで良いのですが，picture 環境でも十分じゃないか，ということです。つまり上記の問には\qbezier(**始点の座標**)(**中点の座標**)(**終点の座標**) が答え[*6]の候補にあるはずです。ただこれを実現するには，中点の座標を計算する必要があり，その計算は一々人間がするのか，外部のソフトに任せるのか，TeX のマクロに組み込むのかということを考えると，既に在るもので楽に？済ませちゃおうってことになるのでしょう。

実際私も今回の指導要領改訂までは，良いじゃないかと思い，直線は手で中点の座標を計算していました。でも，今は面倒臭い。楽がしたいわけです。それも気軽に。

さて，念のためですが，本を作ったり論文を書いたりするのには，この\qbgraph.sty は向かないでしょう。あくまで，一枚から十枚程度のプリントを作るためのものです。

では早速どういうことが出来るか見て頂きましょう。

```
\begin{picture}(10,10)(0,0)
   \qbPline(2,1)(5,9)  (2,1) から (5,9) へ線分
   \qbPointDef A(4,9)  点 A(4,3) を覚えて
   \qbPointDef B(1.2,2.3)  点 B も
   \qbPointDef C(7.2,4.2)  点 C も
   \qbGline AB   覚えたんだから線分 AB を引いて
   \qbGline BC                          BC も
   \qbGline CA                          CA も
   \qbPcirc(5,5)2 中心 (5,5) 半径 2 の円を描いて
   \qbGcirc C(3)  中心が C で半径 3 の円を描いて
\end{picture}
```



[*1] TeXWiki の http://oku.edu.mie-u.ac.jp/~okumura/texwiki/?MetaPost などを参照のこと

[*2] 同じく TeXWiki の http://oku.edu.mie-u.ac.jp/~okumura/texwiki/?PSTricks などを参照のこと

[*3] http://emath.s40.xrea.com/を参照のこと

[*4] TeX は組版ソフトです。この文書も TeX で記述されています。ワードや一太郎といったワープロとは違いますし，Quark や InDesign のような DTP ソフトとも違います。近いのはネット上の html ファイルでしょうか。ファイル自身はテキストファイル。文章とコマンドの羅列です。それをコンパイルすることで，実際の文書 (dvi ファイル) を実現します。TeX はこの操作（コンパイル？）を typeset(文字を打つ？) と称します。

[*5] .aux,.dvi,.log,.bak, などなど，環境や OS によって違うようですが。

[*6] もちろん，標準的なオプションでもある，epic.sty，\eepic.sty や exline.tex のような既成のものも候補でしょうけど

上の図[*7]は簡単な例ですが，基本的な使い方はこれで十分理解できると思います。

それぞれのマクロは名前の由来である\qbezier による bezier 曲線を用いて線を描きます。そのために，座標を覚えたり，座標で計算したり，その計算結果を利用して描いたり，繰り返したり（円は確か 30 度ずつ繰り返しで描いています。）といったことを実現しています。再利用可能（というよりは部品）なマクロ群ですので，自分で更に特殊なマクロを組上げることはメモリの問題などがなければ大丈夫でしょう。

もっと大事なことは picture 環境をそのまま使っていますから，いわゆる普通のコマンド (\put(**座標**){**置くもの**},$\cdots$) が従来通り使えます。あくまでも，epic.sty や eepic.sty のような picture 環境の機能拡張マクロなわけです。必要になる実数計算には eclarith.sty を利用しています。従って，そこそこの正負の実数の加減乗除と三角関数 (eclarith.sty 準拠) が使えます。

1.2 必要なファイルと構成

自分で使うために作り出したので，当初は一つのスタイルファイルにまとめていました。が，エディタ[*8]が一々TEX,LATEX のコマンドを解釈しようと簡易文法解析？するので，煩くなって分けました。

本体である\qbgraph.sty は, 基本的にはスタイルファイル群の呼出し役のファイルです。\qbgrapha〜z.sty[*9]が中身です。また，eclarith.sty を必要としますから，

```
\usepackage{ascmac,mtcastle,eclarith,qbgraph}
```

のように eclarith.sty と一緒に\usepackage{...}で呼出し指定して下さい。ちなみに，上の例の mtcastle.sty はまた別のマクロ集です。気になる方も，気にせずに済ませてください。

2 qbgraph の実際

2.1 早速使うために，最低限のコマンドを

さて，先に書いた通り，無手勝流の気楽なマクロ群です。必要なマクロから順に作って構成したので，誰でも改造，改良して好きに作り直すことが出来るでしょう。そこで，基本的な命令 (コマンド，マクロ) の仕様と使用を紹介していきます。

2.2 線分を描くマクロ

qbPline, qbPlined

\qbPline(x_1, y_1)(x_2, y_2)　　使用例　\qbPline(1,2)(3,4)

という形です。このマクロは実は

qbline, qblined

\qbline{x_1}{y_1}{x_2}{y_2}　　使用例　\qbline{1}{2}{3}{4}

\qbline{x_1}{y_1}{x_2}{y_2} という\qbgraph の最初期に作成したマクロを呼んでいます。ずっと LATEX の\newcommand でマクロ定義していたので，この形[*10]なのです。

$$\frac{x_1+x_2}{2} = \texttt{\tmpx}$$

[*7] この図では，\unitlength=5truemm としています。また，図の範囲が判りやすいように，\qblatticed{0}{0}{11}{11}というマクロで格子点を描き加えています。もちろん，実際に使う時は，縦横のサイズ（上の例では (10,10)）や原点の位置（上の例では (0,0)），\unitlength の長さも好きにして下さい。個人的には\unitlength=5truemm で縦横 10 なら要するに縦横 5cm だなあって，図の大きさが把握しやすいので，こう設定してみました。

[*8] WinShell というエディタです。

[*9] いくつあるかは，依然開発途上状態ですから，この原稿執筆時でもわかりません。それほどに整理下手なのでしょう。

[*10] LATEX の\newcommand でマクロ定義する場合，複数の引数の一つ一つを{}で囲んで並べて記述しなくてはならないようです。違うのかもしれません。

のような[*11]計算で，中点の座標を一時記憶させて，\qbezier の中点に使っているわけです。

同じようなマクロですが

qbGline, qbGlined

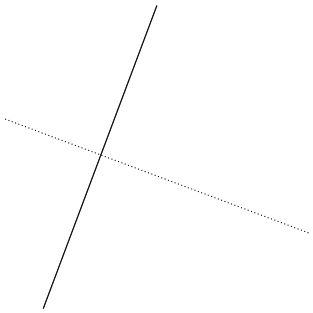
\qbGline $P_{点の名前}Q_{点の名前}$ 使用例 \qbGline AB

は，点の名前と座標を後述のマクロ\qbPointDef#1(#2,#3) で前もって定めておけば，線分の名前 (例えば，AB とか PM とか) で線分を描画することができるマクロです。

実際に使ってみた様子を観てみましょう。

```
\qbline{2}{1}{5}{9}
\qblined{1}{6}{9}{3}
```

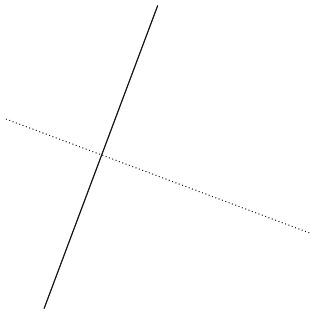
これが最初に作成したマクロ
点の座標を一々{}で括るので
ソースが見辛いし面倒です。
そこで，新しく定義しなおして
座標や点の名前で描けるように



\qb のあとに P が入るマクロ群

```
\qbPline(2,1)(5,9)
\qbPlined(1,6)(9,3)
```

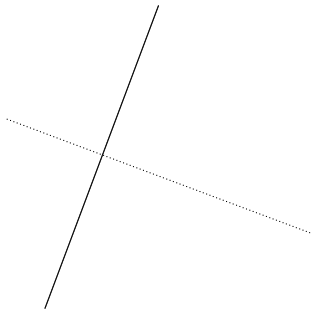
点の指定を座標表記にしました
右のは\qbG〜の名称で，
点の座標と名前を定義すれば，
点の名前で線を引きます。



```
\qbPointDef P(2,1)
\qbPointDef Q(5,9)
\qbPointDef A(1,6)
\qbPointDef B(9,3)
```

と点の名前と座標を定義しておけば

```
\qbGline PQ
\qbGlined AB
```



2.3 点を描くマクロ，点を決めるマクロ

さて，線分が簡単に引けるようになれば，ほぼどんな図形でも描けるわけですが，\qbGline のように，点の名前で線分を引こうと思えば，点の名前と座標の定義が必要です。また，初等幾何の問題を描画しようと思えば，線分の交点であるとか中点であるとか，垂線の足であるとか，様々な点が必要になります。円を考える場合は，中心の点も見たい。更には，点に名前があるのなら，描いたときに点の名前も描き込みたい。

そんなこんなのマクロ群です。

qbPpoint

\qbPpoint(x_1, y_1) 使用例 \qbPpoint(1,2)

\qbP〜ですから，座標 (#1,#2) を引数にするマクロです。点を描きます。やはり，

qbpoint

\qbpoint{x_1}{y_1} 使用例 \qbpoint{1}{2}

というマクロを呼び出しています。また，

qbGline, qbGlined

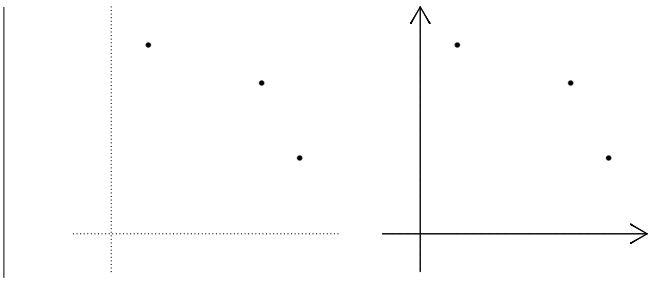
\qbGpoint $P_{点の名前}$ 使用例 \qbGline A

と点の名前で描くこともできます。ただし，このマクロの前に\qbPointDef #1(#2,#3) などで，点の名前と座標が

[*11] 実際は\eclarith.sty のマクロで\Add{#1}{#2}{\tmpx}足して \Div{\tmpsx}{2}{\tmpx}2で割る という具合に計算しています。

定められている必要があります。では用例[*12]です。

```
\begin{picture}(7,7)(-1,-1)
   \qbPpoint(5,2)  (5,2) に点
   \qbpoint{1}{5}  (1,5) に点
   \qbPointDef A(4,4)  点 A(4,4) を覚えて
   \qbGpoint A    覚えたんだから点 A を描いて
\end{picture}
```



次に，点の名前と座標の設定について少し見ておきましょう。既に書いたとおり，当初は判らない事が多くて，マクロの引数を全て，{#1}{#2}〜{#N}[*13]のように一々{}で括る標記の仕様で作っていました。それが，後になって，TEX の\def だと，引数の形が自由に設定できることが判ったので，様々な座標を引数にするコマンドについて，座標を (#1,#2) の形で指定できるように定義しなおし[*14]ました。

その過程[*15]で，マクロの名前を引数で指定できるコマンド\@namedef があることがわかりました。

少し難しいですが，例えば 点 P の座標を (x,y) としたいときに，点の名前 P をマクロ名に含んだマクロを P(x,y) を引数にすることで定義 (\def) することができるわけです。これは凄いことです。これなら，配列は要らないのも当然なのかもしれません。もう少し具体的には，例えば点 A(3,4) を引数にすれば、名前\def\qbPtA{A}, x 座標\def\qbxofPtA{3}, y 座標\def\qbyofPtA{4} の 3 つのマクロを自動的に作成する仕様にしました。実際のコードです。

```
\def\qbPointDef#1(#2,#3){
   \def\tmpDummyname{qbPt#1}\@namedef\tmpDummyname{#1}
   \def\tmpDummyname{qbxofPt#1}\@namedef\tmpDummyname{#2}
   \def\tmpDummyname{qbyofPt#1}\@namedef\tmpDummyname{#3}}
```

\def\tmpDummyname{qbPt#1}の部分で点名#1 を含んだコマンド名をマクロ\tmpDummyname に定義し，その名前を持ったマクロを\@namedef\tmpDummyname{#1}で定義[*16]しています。

実際のコードと実行例[*17]です。

```
\qbPointDef G(3,-2) 点の名前は\qbPtG で，$x$座標は$\qbxofPtG$で,$y$座標は$\qbyofPtG$です。
```

typeset

点の名前は G で，x 座標は 3 で，y 座標は -2 です。

さて，一々点 G の x 座標を得るのに\qbxofPtG と書くぐらいなら，こんなの要らないじゃないかと思ったあなたは正解です。\qbPointDef と対を成すマクロ，\qb##ofPointDef という一連のマクロを作りました。これは，点の名前を引数にして，別の引数に x 座標や y 座標を返すものです。

```
\def\qbxofPointDef(#1,#2){\def\tmpDummyname{qbxofPt#1}\qbdef{\@nameuse\tmpDummyname}{#2}}
```

上手く動いてくれるようにした結果がこうです。@nameuse というマクロが引数を名前に持つマクロを実行してくれます。上の例では点の名前#1 を引数にして，その点の x 座標を#2 に戻します。

この項の最初に紹介した，\qbGpoint の場合，

[*12] 位置関係が判りにくいので図には座標軸を入れています。

[*13] といっても確か\newcommand で設定できるマクロの引数は 8 個だか 9 個だかまでなので，$N \leqq 9$ 程ですけど。

[*14] 所謂，\qbP シリーズです。定義しなおしは今のところ\qbgraphp.sty 内で行っています。

[*15] やりたいことを実現する為にネット上を調べたり，尋ねたりしていて，ある日，良いページを発見しました。

[*16] ここに現れるマクロはいずれも，何かを「する」ものではなく，単なる値の「入れ物」的な扱いです。従って，マクロ定義の内容部分である\def の最後の{}内はいずれも拍子抜けするほど単純です。判りたい人はよく読んで下さい。

[*17] 最初の\qbPointDef G(3,-2) は, 定義するだけで表面上は何もしません。この定義により産まれたマクロ\qbPtG, \qbxofPtG, \qbyofPtG はそれぞれ点の名前 G，x 座標 3, y 座標 -2 を返しています。

```
\def\qbGpoint#1{\qbxyofPointDef(#1,\tmpSx,\tmpSy) \qbPpoint(\tmpSx,\tmpSy)}
```

という具合に，\qbxyofPointDef で名前#1 の点の x, y 座標を\tmpSx,\tmpSy に返してもらって，それを引数に，\qbPpoint で点を描くという仕様になっていますね。とっても単純で見易いソースコードではないでしょうか。

ただし，案の定というか何というか，点の名前が定義されていない場合の処理[*18]は想定外です。正しく使って下さい。

では、先程の例をもう一度みて，ソースと結果を見比べて見ましょう。

```
\qbPointDef G(3,-2) 点の名前は\qbPtG で，$x$座標は$\qbxofPtG$で,$y$座標は$\qbyofPtG$です。
```

typeset

点の名前は G で，x 座標は 3 で，y 座標は -2 です。

```
\qbPointDef G(3,-2) 点の名前は G で，\qbxyofPointDef(G,\tmpx,\tmpy)%
$x$座標は$\tmpx$で,$y$座標は$\tmpy$です。
```

typeset

点の名前は G で，　　x 座標は 3.0 で,y 座標は -2.0 です。

違いが判るでしょうか。\qbxyofPointDef を実行しているところに，不用なスペース[*19]が入っていますね。また，x 座標は,y 座標の\tmpx，\tmpy の値が実数仕様[*20]になっていますね。

では，\qbG シリーズを使うのに必要な点の定義マクロは，

qbPointDef

\qbPointDef $\mathrm{P}_{\text{点の名前}}(x_{\text{座標}}, y_{\text{座標}})$　　使用例　\qbPointDef A(2,-3)

です。

また，ある程度点が指定できてから使うものとして，

中点 qbGmidPoint

\qbGmidPoint $\mathrm{AB}_{\text{線分}}$ $\mathrm{M}_{\text{中点名}}$　　使用例　\qbGmidPoint PQ N

交点 qbGcrossPoint

\qbGcrossPoint $\mathrm{AB}_{\text{線分}}$ $\mathrm{CD}_{\text{線分}}$ $\mathrm{P}_{\text{交点名}}$　　使用例　\qbGcrossPoint AM BN G

回転点 qbGrotPoint

\qbGrotPoint $\mathrm{AB}_{\text{線分}}$ $\theta_{\text{回転角度}}$ $\mathrm{P}_{\text{点名}}$　　使用例　\qbGrotPoint PQ 45 R

A を回転の中心にして，線分 AB を θ(度数法で指定) 回転したときの，B の位置を P とする。

分点 qbGdivPoint

\qbGdivPoint $\mathrm{AB}_{\text{線分}}$ $m : n_{\text{分比}}$ $\mathrm{P}_{\text{分点名}}$　　使用例　\qbGdivPoint BC 5:3 L

垂線の足 qbGverticalPoint

\qbGverticalPoint $\mathrm{A}_{\text{垂線を下す始点}}$ $\mathrm{BC}_{\text{垂線を下す線分}}$ $\mathrm{H}_{\text{垂線の足}}$　　使用例　\qbGverticalPoint A BC H

などがあります。

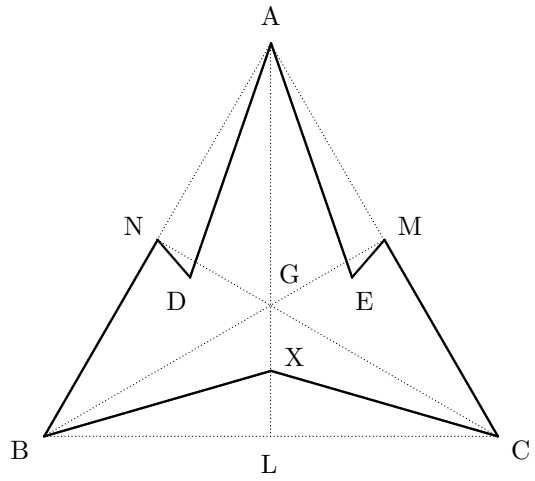
[*18] いわゆるエラー処理などと呼ばれているプログラミングでは大切で手間の掛る部分です。

[*19] 何故なのか原因がわかりません。おそらく，@nameuse 等に問題があるのでしょう。誰か修正して下さい。

[*20] こちらは計算用に用意した数値定義の\qbdef で実数値扱いにしています。ですから当然といえば当然の結果です。所謂単位抜きの寸法ということです。

点の定義，回転点，中点，交点などを使った作図（某学校の校章の作図を具体例に）

```
\qbPointDef B(0,0)     \qbPointDef C(5,0)
\qbGrotPoint BC 60 A
\qbGlined AB   \qbGlined BC   \qbGlined CA
\qbGverticalPoint A BC L
\qbGverticalPoint B CA M
\qbGverticalPoint C AB N
\qbGlined AL   \qbGlined CN   \qbGlined BM
\qbGcrossPoint AL BM G
\qbGmidPoint GL X
\qbGmidPoint GB Y
\qbGmidPoint GC Z
\qbGcrossPoint AY NX D
\qbGcrossPoint AZ MX E
\thicklines
\qbGline AE   \qbGline EM   \qbGline MC
\qbGline CX   \qbGline XB
\qbGline BN   \qbGline ND   \qbGline AD
```



点 B を原点にとり，点 C を (5, 0) にとり，線分 BC を B を中心に 60° 回転させた (\qbGrotPoint) 先の点を A として，正三角形 ABC を設定します。\qbGlined で点線で描きます。

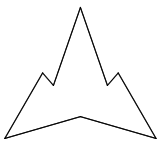
\qbGverticalPoint で各頂点から対辺への垂線の足を決め，垂線を点線で描きます。\qbGCrossPoint で垂線の交点を G と決めます。\qbGmidPoint で GL，GB，GC の中点を X,Y,Z として，再び\qbGCrossPoint で AY と NX の交点を D，AZ と MX の交点を E とします。

最後に，\thicklines で太線に設定して，AEMCXBNDA と結んで出来上がりです。

図には判りやすいように，\qbGplabel で点の名前 (Label) を入れています。

マクロの名前や引数の形式が組上げた本人も怪しいのですが，うまく使えれば，この通り，作図作法 (描き方) 通りに図をが描けます。

勿論，作図の過程や計算の結果を全て見せる必要は無いので，先の校章の場合でも，



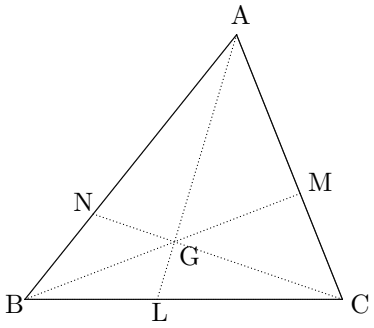
のように結果だけ，或いは必要な部分だけ表示すれば良いわけです。

*

次はもう少し高等学校数学用に実用的な例を見ましょう。チェバの定理やメネラウスの定理の問題です。

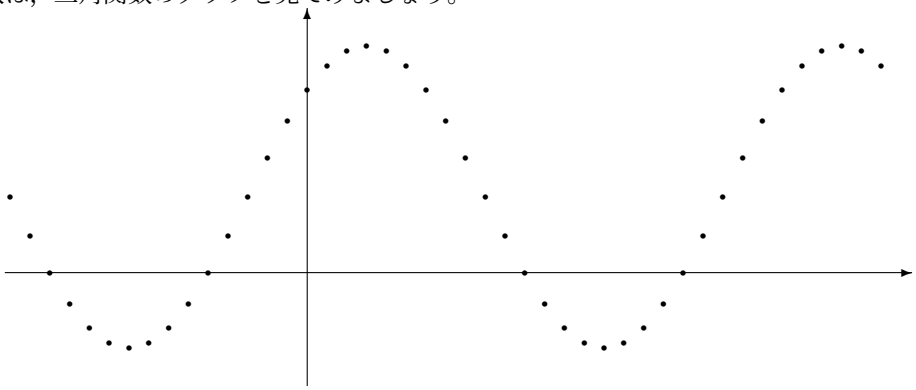
問題　三角形 ABC で，辺 BC を 5:3 に，辺 AC を 3:2 に内分する点を L，M とする。AL，BM の交点を G とし，CG と AB の交点を N とするとき，AG:GL, AN:NB を求めなさい。

```
\qbPointDef A(4,5)   \qbPointDef B(0,0)
\qbPointDef C(5,0)
\qbGline AB   \qbGline BC   \qbGline CA
\qbGdivPoint BC 5:7 L
\qbGdivPoint AC 3:2 M
\qbGlined AL   \qbGlined BM
\qbGcrossPoint AL BM G
\qbGcrossPoint CG AB N
\qbGlined CN
```



*

次は，三角関数のグラフを見てみましょう。



```
\qbdef{-15}{\tmpTt}            pi=3.1416 として 180 度を 12 分割しています。
\Div{180}{12}{\tmpDx}          -15/12*pi から 29/12*pi まで 座標はラジアン，計算は度数法で
\loop                          \loop で繰り返します。
\ifdim \tmpTt pt<30 pt         （-12 から始めて）まだ 30 より小さければ，実行しなさい。
   \Mul{\tmpTt}{\tmpDx}{\tmpFx}      度数法で x の値を決めて
   \qbTdegtorad(\tmpFx,\tmpFGx)      グラフを描く為に x 座標を度数法からラジアンに変換
   \Add{\tmpFx}{45}{\tmpFx}          45 度ずらして
   \Mul{\tmpFFx}{1}{\tmpFFx}
   \qbsin{\tmpFFx}{\tmpFy}           sin の値を計算して
   \Mul{\tmpFy}{2}{\tmpFy}           3 倍する。
   \Add{\tmpFy}{1}{\tmpFy}           で 1 を加える。つまり，y=2sin(x+45 度)+1
   \put(\tmpFx,\tmpFy){\circle*{0.08}}点を描く
   \Add{\tmpTt}{1}{\tmpTt}           進める
\repeat                           を繰り返す。
```

\loop...\ifXXX...\repeat という構文で繰り返しを使用しています。\qbsin を使って，三角関数の値を計算しています。少し面倒くさいので，これをマクロに組めば良いのでしょう。ある程度準備ができているので，計算式が使えないことを除けば，Basic 等のプログラムを打つ感覚ですね。