kpic のいい加減な使用法

Ver.1.98.5

石野　信義

nob.asaoka@jcom.home.ne.jp

2001/04/25

TEX で数学のプリントを作成するとき，簡単に図を書くためにkpic.sty を作成しました。サイエンス社発行の「LATEX 自由自在」及び eclarith.sty を参考（というよりほとんどそのまま）に取り敢えず動くものを 1994 年 10 月に作成し、校内[1]の数学の教員で使用していたのですが、同僚からの要望と前からの直したいと思っていたので、今回わかりやすくするためにほとんど書き直し、おまけにマニュアル[2]まで作ってしまいました。

このマクロは以下のスタイルファイルを読み込みますので，必ずパスの通ったところに用意して置いて下さい。用意しておけば、kpic.sty で読み込みます。

epic.sty
eepic.sty
eclarith.sty

●できること

このパッケージでできることは、

- picture 環境と同時に「枠」「方眼紙」「座標」を設定することができる。
- eclarith.sty の¥node の定義を拡張し、「座標の表示文字の位置と距離を指定」「極座標設定」「空間座標」ができる。
- ２点間の線の長さ、名称を表示できる。２直線間に直角の記号、角度の記号が表示できる。

などです。

[1] 2000 年現在、神奈川県立上矢部高校のこと。タイトルを含むほとんどのマクロが K ではじまるのは、KAMIYABE の K だから。こうすると他のスタイルのマクロを名前がぶつからないから

[2] 今読んでいるこの文書です。これが欲しいと同僚に泣きつかれました。でもこんなのを作っているよりマクロを書いている方が楽しい。

目 次

第1章　picture環境の拡張

原点を中央にする picture 環境を定義、さらに枠も付ける。そうすると以下のようになり、同じ座標を 2 度ずつ書かなくてはなりません。そして数値を変更する場合も、2 度手間となります。

```
¥begin{picture}(8,4)(-4,-2)
  ¥put(-4,-2){¥framebox(8,4){}}
¥end{picture}
```

同じことが方眼紙のようなますめを書く時や、座標軸を書く時にも考えられます。そこで、KPic.sty では、これらに対応した環境を設定しました。とはいっても内部で置き換えているだけです。

ここでは ¥Knode 命令が出てきますが、原点の場所を表示しているだけです。詳しいことは 28 ページを見て下さい。

この章の定義で「(大きさ)」の所は、「(横の大きさ，縦の大きさ)」を入れ、「(左下座標)」は「(左下の x 座標, 左下の y 座標)」を入れて下さい。

以下にこの章に出てくる環境名を表にしておきました。

基本形	実線で格子を描く	点線で格子を描く
Pic	PicGrid	PicDot
PicC	PicGridC	PicDotC
PicFrame	PicFrameGrid	PicFrameDot
PicFrameC	PicFrameGridC	PicFrameDotC

基本形	実線で格子を描く	点線で格子を描く	目盛りをふる
Axes	AxesGrid	AxesDot	AxesScale
AxesC	AxesGridC	AxesDotC	AxesScaleC
AxesFrame	AxesFrameGrid	AxesFrameDot	AxesFrameScale
AxesFrameC	AxesFrameGridC	AxesFrameDotC	AxesFrameScaleC

1.1 Pic 環境

1.1.1 基本環境 Pic

定義

```
¥begin{Pic}(大きさ)[(左下座標)]
  - - - - - - -
¥end{Pic}
```

Pic 環境は、picture 環境と同じ働きをします。しかし内部で上下左右の数字を記憶していますので。特にこの KPic.sty の命令を使う時や作成中のKGraph.sty を使用するときは、picture 環境ではなく、この Pic 環境を使用して下さい。

1.1.2 Pic 環境で原点を中心にする環境 PicC

定義

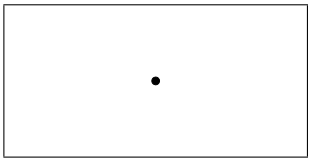
```
¥begin{PicC}(大きさ)
  - - - - - - -
¥end{PicC}
```

PicC 環境は、picture 環境の横と縦の大きさだけで原点を中央にします。次の例を参照して下さい。

入力

```
¥unitlength=5mm%
¥begin{PicC}(8,4)
  ¥put(-4,-2){¥framebox(8,4){}}
  ¥Knode*(0,0){O}
¥end{PicC}
```

出力



どこだかわかるように枠表示に、¥framebox も入れてあります。このように枠も同時に表示するための命令が、次の PicFrame[C] 環境です。

この章の全環境の最後に「C」をつけることによって同様の働きをします。

1.1.3 枠付きの環境 PicFrame[C]

定義

```
¥begin{PicFrame[C]}(大きさ)[(左下座標)]
  - - - - - - -
¥end{PicFrame[C]}
```

picture 環境と同時に枠も付けられる環境です。最後に「C」を付けることによって、中心を原点にします。枠の太さは¥FrameLen で変更できます。初期値は¥FrameLen=0.8pt、¥thicklines と同じ太さです。

以下の例を参考にして下さい。

●基本

入力

```
¥unitlength=5mm%
¥begin{PicFrame}(8,4)
  ¥Knode*(0,0){O}
¥end{PicFrame}
```

出力

●原点をずらす

入力

```
¥unitlength=5mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥Knode*(0,0){O}
¥end{PicFrame}
```

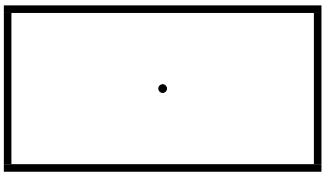
出力

●中心が原点で枠の太さを 1mm にした例

入力

```
¥unitlength=5mm%
¥FrameLen=1mm
¥begin{PicFrameC}(8,4)
  ¥Knode*(0,0){O}
¥end{PicFrameC}
```

出力



この章の全ての環境の「¥Pic」の次に書くことによって、同様の働きをします。

1.1.4 実線格子を表示する環境 Pic[Frame]Grid[C]

定義

```
¥begin{Pic[Frame]Grid[C]}[間隔](大きさ)[(左下座標)]
- - - - - - -
¥end{Pic[Frame]Grid[C]}
```

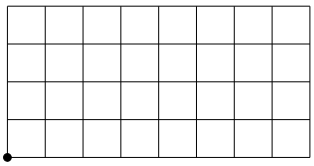
picture 環境で指定された環境内を、指定された間隔で実線の格子を引く命令です。間隔に置く数字の数によって動作が異なります。詳細は、19 ページの表を参考にして下さい。

●基本

入力

```
¥unitlength=5mm%
¥begin{PicGrid}(8,4)
  ¥Knode*(0,0){O}
¥end{PicGrid}
```

出力

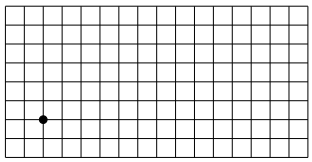


●原点をずらして間隔を半分にした

入力

```
¥unitlength=5mm%
¥begin{PicGrid}[0.5](8,4)(-1,-1)
  ¥Knode*(0,0){O}
¥end{PicGrid}
```

出力

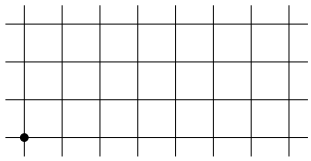


●原点を単位系以外にする

入力

```
¥unitlength=5mm%
¥begin{PicGrid}(8,4)(-0.5,-0.5)
  ¥Knode*(0,0){O}
¥end{PicGrid}
```

出力

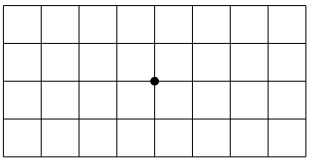


●中心を原点にした

入力

```
¥unitlength=5mm%
¥begin{PicGridC}(8,4)
  ¥Knode*(0,0){O}
¥end{PicGridC}
```

出力

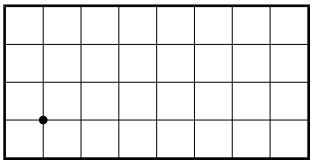


●枠付きで原点を単位系以外にする

入力

```
¥unitlength=5mm%
¥begin{PicFrameGrid}(8,4)(-1,-1)
  ¥Knode*(0,0){O}
¥end{PicFrameGrid}
```

出力

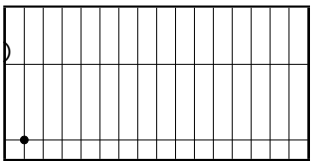


●枠付きで原点をずらし、横は 0.5 づつ、縦は 2 づつにした

入力

```
¥unitlength=5mm%
¥begin{PicFrameGrid}[0.5,2](8,4)(-0.5,-0.5)
  ¥Knode*(0,0){O}
¥end{PicFrameGrid}
```

出力

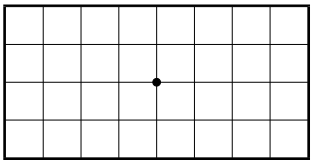


●枠付きで原点中心

入力

```
¥unitlength=5mm%
¥begin{PicFrameGridC}(8,4)
  ¥Knode*(0,0){O}
¥end{PicFrameGridC}
```

出力



1.1.5 点線格子を表示する環境 Pic[Frame]Dot[C]

定義

```
¥begin{Pic[Frame]Dot[C]}[間隔](大きさ)[(左下座標)]
- - - - - - -
¥end{Pic[Frame]Dot[C]}
```

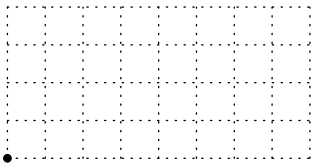
picture 環境で指定された環境内を、指定された間隔で実線の格子を引く命令です。間隔に置く数字の数によって動作が異なります。また点線の間隔も変更できます。詳細は、19 ページの表を参考にして下さい。

●基本

入力

```
¥unitlength=5mm%
¥begin{PicDot}(8,4)
  ¥Knode*(0,0){O}
¥end{PicDot}
```

出力

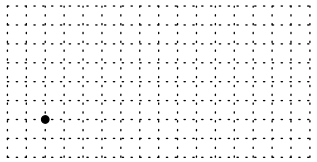


●原点をずらして間隔を半分にした

入力

```
¥unitlength=5mm%
¥begin{PicDot}[0.5](8,4)(-1,-1)
  ¥Knode*(0,0){O}
¥end{PicDot}
```

出力

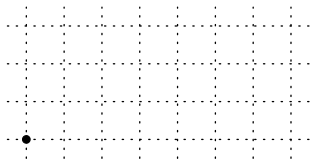


●原点を単位系以外にする

入力

```
¥unitlength=5mm%
¥begin{PicDot}(8,4)(-0.5,-0.5)
  ¥Knode*(0,0){O}
¥end{PicDot}
```

出力

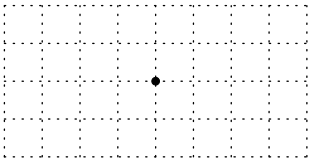


●中心を原点にした

入力

```
¥unitlength=5mm%
¥begin{PicDotC}(8,4)
  ¥Knode*(0,0){O}
¥end{PicDotC}
```

出力

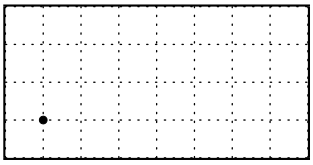


●枠付きで原点をずらし、横は 0.5 づつ、縦は 2 づつにした

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)(-1,-1)
  ¥Knode*(0,0){O}
¥end{PicFrameDot}
```

出力

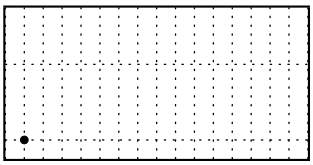


●枠付きで原点を単位系以外にする

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}[0.5,2](8,4)(-0.5,-0.5)
  ¥Knode*(0,0){O}
¥end{PicFrameDot}
```

出力

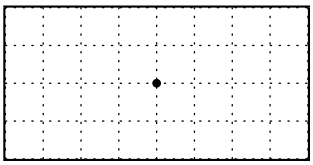


●枠付きで原点中心

入力

```
¥unitlength=5mm%
¥begin{PicFrameDotC}(8,4)
  ¥Knode*(0,0){O}
¥end{PicFrameDotC}
```

出力



1.2 xy 座標を表示する Axes 環境

1.2.1 基本環境 Axes[Frame][C]

定義

```
¥begin{Axes[Frame][C]}(大きさ)[(左下座標)]
- - - - - - - -
¥end{Axes[Frame][C]}
```

Axes 環境は、Pic 環境を宣言すると同時に、xy 座標軸を表示します。座標軸を表示する以外は、Pic 環境、PicC 環境、PicFrame[C] 環境と同じ動作をします。

●基本

入力

```
¥unitlength=5mm%
¥begin{Axes}(8,4)
¥end{Axes}
```

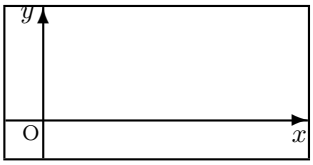
出力

●原点をずらして枠付き

入力

```
¥unitlength=5mm%
¥begin{AxesFrame}(8,4)(-1,-1)
¥end{AxesFrame}
```

出力

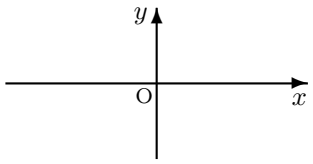


●中心を原点にした

入力

```
¥unitlength=5mm%
¥begin{AxesC}(8,4)
¥end{AxesC}
```

出力



1.2.2 実線格子を表示する環境 Axes[Frame]Grid[C]

定義

```
¥begin{Axes[Frame]Grid[C]}[間隔](大きさ)[(左下座標)]
- - - - - - -
¥end{Axes[Frame]Grid[C]}
```

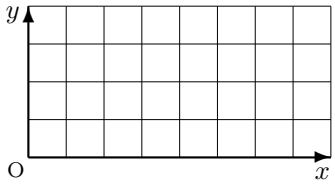
Axes 環境をベースに、PicGrid 環境が重なった環境です。間隔は、19 ページの表を参考にして下さい。

●基本

入力

```
¥unitlength=5mm%
¥begin{AxesGrid}(8,4)
¥end{AxesGrid}
```

出力

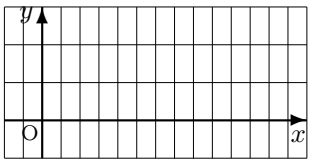


●横だけ間隔半分で原点をずらす

入力

```
¥unitlength=5mm%
¥begin{AxesGrid}[0.5,1](8,4)(-1,-1)
¥end{AxesGrid}
```

出力

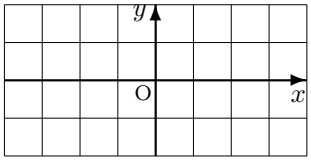


●中心を原点にした

入力

```
¥unitlength=5mm%
¥begin{AxesGridC}(8,4)
¥end{AxesGridC}
```

出力

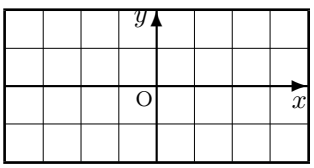


●枠付きで中心を原点にした

入力

```
¥unitlength=5mm%
¥begin{AxesFrameGridC}(8,4)
¥end{AxesFrameGridC}
```

出力



1.2.3 点線格子を表示する環境 Axes[Frame]Dot[C]

定義

```
¥begin{Axes[Frame]Dot[C]}[間隔](大きさ)[(左下座標)]
- - - - - - -
¥end{Axes[Frame]Dot[C]}
```

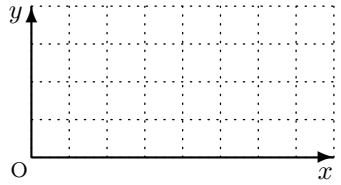
Axes 環境をベースに、PicDot 環境が重なった環境です。間隔は、19 ページの表を参考にして下さい。

●基本

入力

```
¥unitlength=5mm%
¥unitlength=5mm%
¥begin{AxesDot}(8,4)
¥end{AxesDot}
```

出力

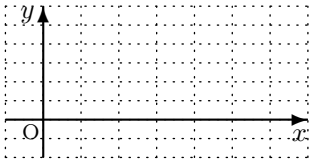


●縦間隔半分で原点をずらす

入力

```
¥unitlength=5mm%
¥begin{AxesDot}[1,0.5](8,4)(-1,-1)
¥end{AxesDot}
```

出力

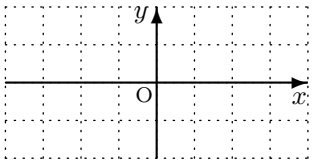


●中心を原点にした

入力

```
¥unitlength=5mm%
¥begin{AxesDotC}(8,4)
¥end{AxesDotC}
```

出力

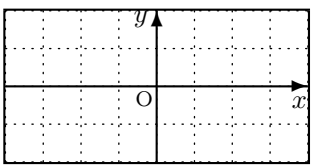


●枠付きで中心を原点にした

入力

```
¥unitlength=5mm%
¥begin{AxesFrameDotC}(8,4)
¥end{AxesFrameDotC}
```

出力



1.2.4 目盛りと数値を入れる環境 Axes[Frame]Scale[C]

定義

```
¥begin{Axes[Frame]Scale[C]}[目盛間隔, 数値間隔](大きさ)[(左下座標)]
- - - - - - -
¥end{Axes[Frame]Scale[C]}
```

xy 座標に目盛りと数値をつけます。数値を表示しない場合は、「数値間隔」を省略して下さい。この環境で使用する場合は、以下の制限があります。

- 目盛りも、数値も縦横同じ割合でしか表示されない。
- 目盛りの種類は 1 種類だけ。
- 実線格子や点線格子を同時に表示することが出来ない

以上の制約外で使用したいときは、第 2.2.2 章 を見て下さい。また、以下の命令は使用することができます。

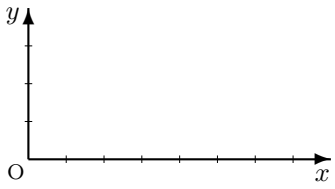
補助命令	説明	初期値
¥KSmallScaleLen	目盛りの長さ (片側)	0.5mm

●基本

入力

```
¥unitlength=5mm%
¥begin{AxesScale}(8,4)
¥end{AxesScale}
```

出力

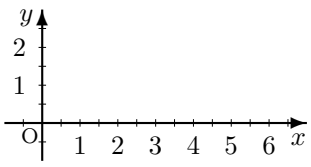


●原点をずらして間隔半分

入力

```
¥unitlength=5mm%
¥begin{AxesScale}[0.5,1](8,4)(-1,-1)
¥end{AxesScale}
```

出力

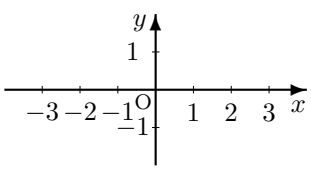


●中心を原点にした

入力

```
¥unitlength=5mm%
¥begin{AxesScaleC}[1,1](8,4)
¥end{AxesScaleC}
```

出力

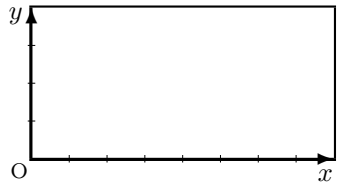


●基本

入力

```
¥unitlength=5mm%
¥begin{AxesFrameScale}(8,4)
¥end{AxesFrameScale}
```

出力

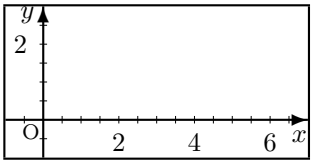


●原点をずらす

入力

```
¥unitlength=5mm%
¥begin{AxesFrameScale}[0.5,2](8,4)(-1,-1)
¥end{AxesFrameScale}
```

出力

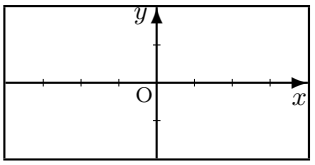


●中心を原点にした

入力

```
¥unitlength=5mm%
¥begin{AxesFrameScaleC}(8,4)
¥end{AxesFrameScaleC}
```

出力



第2章　環境内で使用する

第 1 章で説明した環境の命令の内、いくつかは別に定義してあります。

2.1　¥KGrid ¥KDot マクロ

この節で説明するマクロの、指定された間隔で実線・点線の格子を引く命令です。間隔に置く数字の数によって動作が異なります。以下の表を参考にして下さい。

間隔に置く数	動作
省略	縦横とも単位ごとに実線・点線を引きます
1 つ	縦横ともその間隔ごとに実線・点線を引きます
2 つ	初めが横，次が縦の間隔ごとの実線・点線を引きます

2.1.1　¥KGrid マクロ

定義

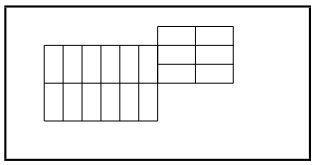
¥KGrid[線の間隔](大きさ)[(左下座標)]

●基本

入力

```
¥unitlength=5mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥put(0,0){¥KGrid[0.5,1](3,2)}
  ¥put(3,1){¥KGrid[1,0.5](2,1.5)}
¥end{PicFrame}
```

出力

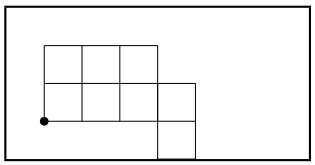


●基本

入力

```
¥unitlength=5mm%
¥begin{PicFrame}(8,4)(-1,-1)
```

出力



```
  ¥KGrid(3,2)
  ¥put(3,-1){¥KGrid(1,2)}
  ¥Knode*(0,0){O}
¥end{PicFrame}
```

2.1.2 ¥KDot マクロ

定義

¥KDot[線の間隔](大きさ)[(左下座標)]

補助命令	説明	初期値
¥KDotSpace	点線の間隔	1mm

2.2 ¥KAxes 関係 マクロ

xy 座標に関するマクロです。

2.2.1 ¥KAxes マクロ

定義

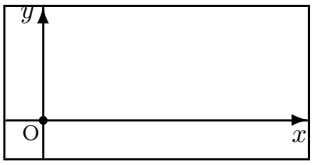
¥KAxes(大きさ)[(左下座標)]

●基本

入力

```
¥unitlength=5mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥KAxes(8,4)(-1,-1)
  ¥Knode*(0,0){O}
¥end{PicFrame}
```

出力

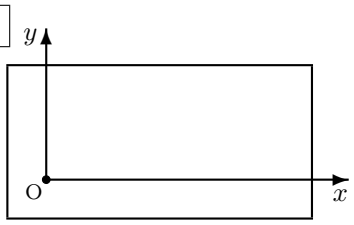


●基本

入力

```
¥unitlength=5mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥KAxes(8,4)
  ¥Knode*(0,0){O}
¥end{PicFrame}
```

出力



2.2.2 ¥KScale マクロ

定義

¥KScale(小目盛り間隔)[(大目盛り間隔)]

17 ページの Axes[Frame]Scale[C] では目盛りの細かい設定が出来ません。このマクロを使用すると、目盛りのを 2 種類使用することが出来ます。

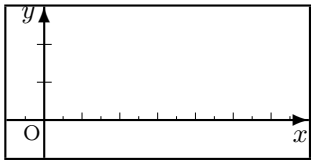
補助命令	説明	初期値
¥KSmallScaleLen	小目盛りの長さ (片側)	0.5mm
¥KLargeScaleLen	大目盛りの長さ (片側)	1mm

●基本

入力

```
¥unitlength=5mm%
¥begin{AxesFrame}(8,4)(-1,-1)
  ¥def¥KScaleDirectionX{1}
  ¥KScale(0.5,1)(1,1)
¥end{AxesFrame}
```

出力



2.2.3 ¥KScaleNumber マクロ

定義

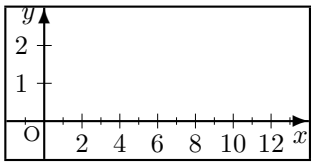
¥KScaleNumber[(軸に振る初期値)][(目盛りの間隔)]

●基本

入力

```
¥unitlength=5mm%
¥begin{AxesFrame}(8,4)(-1,-1)
  ¥KScale(0.5,1)(1,1)
  ¥KScaleNumber(2,1)(1,1)
¥end{AxesFrame}
```

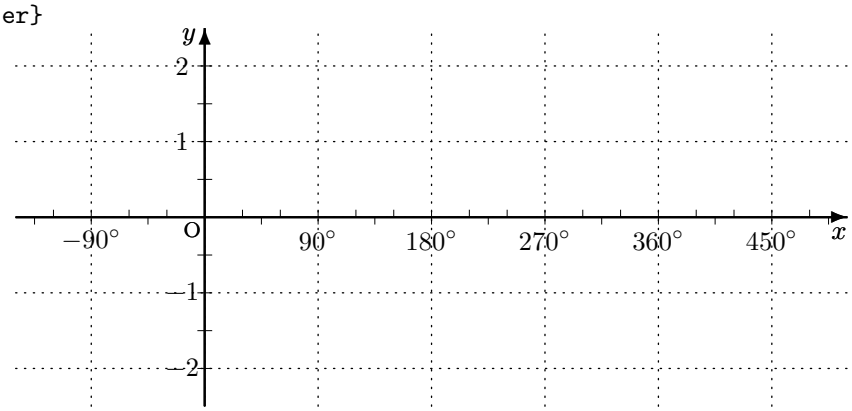
出力



2.3　応用

この章の応用として、以下の例を参考にして活用して下さい。

```
¥begin{center}
  ¥unitlength=5mm%
  ¥begin{AxesDot}[3,2](22,10)(-5,-5)
    ¥KAxes(22,10)(-5,-5)
    % 軸の目盛り
    ¥def¥KScaleDirectionX{1}
    ¥KScale(1,1)(1,1)
    ¥def¥KScaleDirectionX{-1}
    ¥KScale(1.5,0)(1.5,0)
    % x 軸の数値
    ¥kSNDegreetrue
    ¥KScaleNumber(90,0)(3,0)
    % y 軸の数値
    ¥kSNDegreefalse
    ¥KScaleNumber(0,1)(0,2)
  ¥end{AxesDot}
¥end{center}
```



第3章　表示文字と表示位置

この章では、この章以降に頻繁に出てくる命令をまとめました。

3.1　表示文字と表示位置

この章以降に、「表示文字名」「表示位置」に関する命令が多く出てきます。使用法は同じなので、ここでまとめておきます。

表示する文字，文字を置く位置，置く位置の微調整は省略可能ですが、使用する場合じゃ「[](大カッコ)」は必要です。

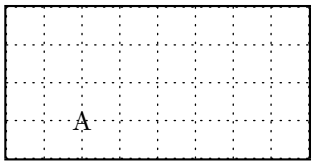
3.2　その座標を中心として文字を表示

次の例は、横 8，縦 4（1 単位は 5mm），左下を原点とする方眼の座標（2, 1）の位置に A を表示します。でもこれくらいなら¥put(2,1){¥makebox(0,0){A}}で充分ではないでしょうか。

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)
  ¥WordSep=5mm%
  ¥Knode(2,1){A}[¥KSame]
¥end{PicFrameDot}
```

出力



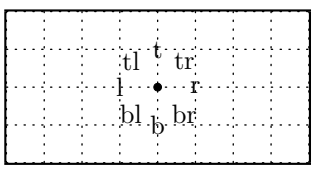
3.3　文字を置く位置

文字を置く位置ですが，座標に対してどの位置に置くかを指定します。¥picture 環境の¥framebox や ¥makebox の位置と同じようにしています。8 方向の例と表示位置を参考にして下さい。

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)
  ¥WordSep=5mm%
  ¥Knode*(4,2){A}[r][r]
```

出力



```
  ¥Knode(4,2){A}[tr][tr]
  ¥Knode(4,2){A}[t][t]
  ¥Knode(4,2){A}[tl][tl]
  ¥Knode(4,2){A}[l][l]
  ¥Knode(4,2){A}[bl][bl]
  ¥Knode(4,2){A}[b][b]
  ¥Knode(4,2){A}[br][br]
¥end{PicFrameDot}
```

3.4　座標からの距離

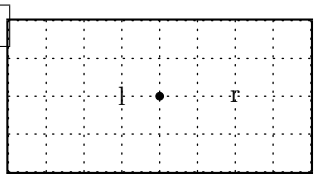
座標からの距離は¥WordSep で定義されています。デフォルトは 3mm です。この数値を変えるだけで座標からの距離が変わります。何故絶対指定にしたのかというと、図形の大きさを変えても文字との距離が変わらない方が図を修正する方が楽だからです。

●文字の位置を左は 5mm，右は 10mm にした例

入力

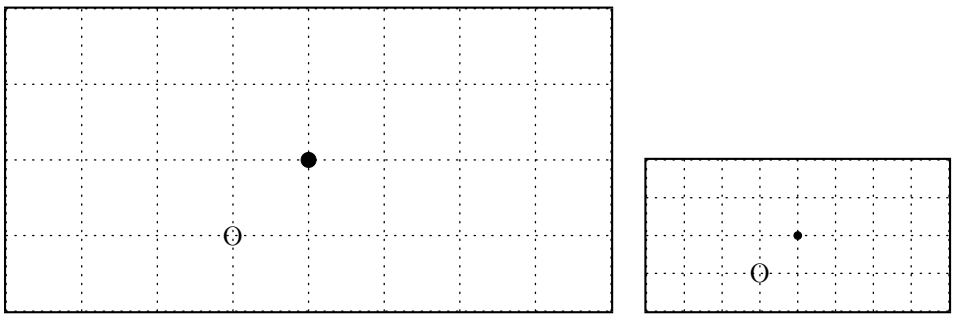
```
¥unitlength=5mm
¥begin{PicFrameDot}(8,4)
  ¥WordSep=5mm%
  ¥Knode*(4,2){A}[l][l]
  ¥WordSep=10mm%
  ¥Knode(4,2){A}[r][r]
¥end{PicFrameDot}
```

出力



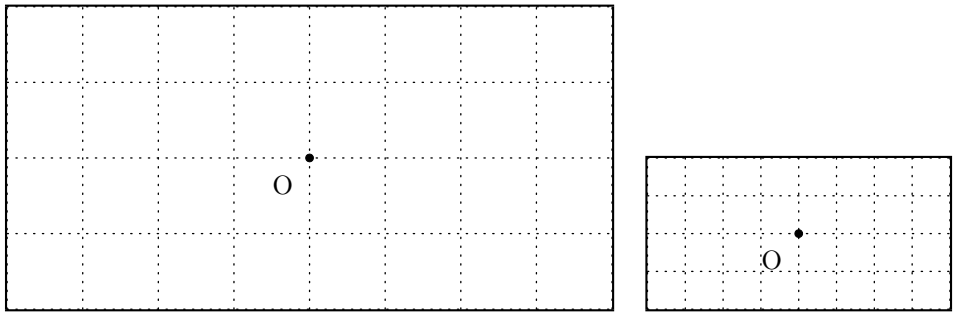
● ¥Knode を使用しないで図を縮小した例

```
¥def¥例図{%
  ¥begin{PicFrameDot}(8,4)
    ¥put(4,2){¥circle*{0.2}}
    ¥put(3,1){¥makebox(0,0){O}}
  ¥end{PicFrameDot}
}% End of ¥例図

¥unitlength=1cm%
¥例図
¥hspace{2mm}
¥unitlength=5mm%
¥例図
```



● ¥Knode を使用して図を縮小した例

```
¥def¥例図{%
  ¥begin{PicFrameDot}(8,4)
    ¥WordSep=5mm%
    ¥Knode*(4,2){O}[¥KSame][bl]
  ¥end{PicFrameDot}
}% End of ¥例図

¥unitlength=1cm
¥例図
¥hspace{2mm}
¥unitlength=5mm
¥例図
```



これを指定したのは，文字と線が重ならないための工夫です。

3.5 置く位置の微調整

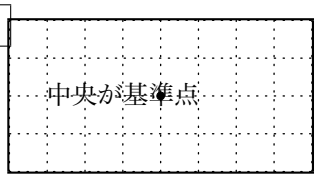
最後の定義は前にふれた置く位置を微調整します。次の例を見て下さい。実際に置く文字が多いと、こういう結果になってしまいます。

●置く文字多い例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)
  ¥WordSep=5mm%
  ¥Knode*(4,2){A}[中央が基準点][l]
¥end{PicFrameDot}
```

出力

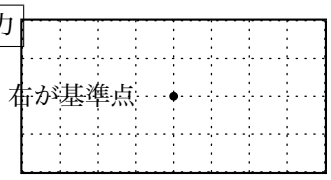


●文字を置く場所の基準点を右にして表示

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)
```

出力



```
  ¥WordSep=5mm%
  ¥Knode*(4,2){A}[右が基準点][l,r]
¥end{PicFrameDot}
```

第4章　点を定義

picture環境では、線を多く表示するとき，座標を何回も書かなければなりません。それが、picture環境を使いにくくしている原因のひとつであると思います。しかし、「LaTeX 自由自在」を読んだとき、点を定義しておいて何回も使用できることがわかり、その本を見よう見まねでスタイルファイルを書いたのがこの「KPic.sty」の前身にあたる「K-PIC1.STY」です。

今回の Ver.2 へ向けての改訂の第一歩は、この¥node の改良から始まりました。点を定義する方法として、平面は「平面座標」「極座標」の２つを作成しました。空間座標も作ってありますが、現在は平面に絞り作業は凍結しています。

- 平面は「xy 座標」「極座標」の 2 つの定義方法しました。
- 座標名はアルファベット 1 文字で定義（A〜Z,a〜z 等で定義）
- 「表示文字と位置等」については第 3 章を参照して下さい。
- 半角「$*$」を付けるとその座標に塗りつぶされた円 (¥circle*{直径}) を置く。直径は¥KBullet で定義され、初期値は¥KBullet=1mm です。

4.1　点の定義

4.1.1　平面座標 (xy 座標) で定義 ¥Knode

定義

1．¥Knode[*](x 座標,y 座標){座標名}[表示文字][文字位置]
2．¥Knode[*][{node で定義した座標}](x 座標,y 座標){座標名}[表示文字][文字位置]

(x 座標,y 座標) を指定した「座標名」で定義する命令です。「表示文字」と「文字位置」は省略可能です。「文字位置」だけ記載することは無意味です。

2 は、定義済みの座標を基準 (仮原点) として座標を定義します。「¥Knode」単独では手計算でできるのであまり意味がありませんが、次の「¥Pnode」と組み合わせると有効でしょう。

また、表示文字の所に、文字の代わりに次の命令を置くことが出来ます。

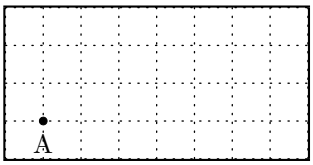
¥KSame	定義した座標名を表示する
¥Cnode	定義した座標を表示する
¥CnodeName	定義した座標名と座標を表示する

●点 A(1,1) を定義

入力

```
¥unitlength=5mm
¥begin{PicFrameDot}(8,4)
  ¥Knode*(1,1){A}[¥KSame][b]
¥end{PicFrameDot}
```

出力

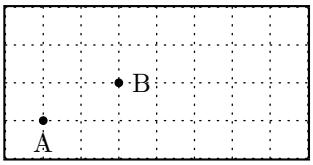


●点 A を基準として点 B(3,2) を定義

入力

```
¥unitlength=5mm
¥begin{PicFrameDot}(8,4)
  ¥Knode*(1,1){A}[¥KSame][b]
  ¥Knode*{A}(2,1){B}[¥KSame][r]
¥end{PicFrameDot}
```

出力



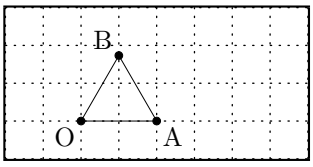
¥Knode の欠点というより、平面座標の欠点ですが、角度が出てくる図形、例えば正三角形の三点を定義する場合のように小数の座標が出てきてしまします。次のマクロのように 2 点を整数にしても、残りの 1 点は小数となってしまいます。

● ¥Knode で正三角形を定義

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)(-2,-1)
  ¥Knode*(0,0){O}[¥KSame][lb]
  ¥Knode*(2,0){A}[¥KSame][rb]
  ¥Knode*(1,1.73205){B}[¥KSame][lt]
  ¥KPath{OABO}
¥end{PicFrameDot}
```

出力



このように複雑にしないで、簡単に点を定義する命令として、次の¥Pnode を作成しました。

4.1.2　極座標で定義 ¥Pnode

定義

1．¥Knode[*](*x* 座標,*y* 座標){座標名}[表示文字][文字位置]

2．¥Knode[*][{node で定義した座標}](*x* 座標,*y* 座標){座標名}[表示文字][文字位置]

定義

¥Pnode[*][(元になる点の x 座標, 元になる点の y 座標)](大きさ , 角度){座標名}[表示文字][文字位置]

¥Pnode[*][{node で定義した座標}](大きさ , 角度){座標名}[表示文字][文字位置]

前述したように¥Knode だと、正三角形を表示するのに事前に計算をしなければなりません。また普段数学で使用する角度以外の特殊な角度は三角関数表を見るか、関数電卓をたたかなければなりません。そこで極座標の考えを利用して、基準になる点からの距離と角度で座標を定義します。角度は「**度**（degree)」使用して下さい。ラジアンでは小数表記なのでわかりにくいです。極座標を内部で直交座標に変換して定義しています。このために必要な「$\sin\theta$と $\cos\theta$」は eclairth.sty で用意されているのでできました。

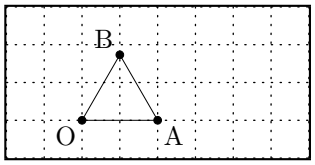
ここでも表示文字の代わりに¥Knode の 3 つの命令が使用できます。ただし、座標は計算された数値が表示されるので、小数以下の桁数 (無しも含めて) を考えなくてはいけません。小数点以下の桁数は¥KFigure で設定できます。詳しいことは「初期設定」を参照して下さい。

Pnode で正三角形を定義

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)(-2,-1)
  ¥Pnode*(0, 0){O}[¥KSame][lb]
  ¥Pnode*(2, 0){A}[¥KSame][rb]
  ¥Pnode*(2,60){B}[¥KSame][lt]
  ¥KPath{OABO}
¥end{PicFrameDot}
```

出力

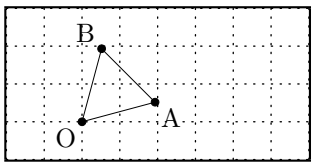


Pnode で 15° 回転した正三角形を定義

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)(-2,-1)
  ¥Pnode*(0, 0){O}[¥KSame][lb]
  ¥Pnode*(2,15){A}[¥KSame][rb]
  ¥Pnode*(2,75){B}[¥KSame][lt]
  ¥KPath{OABO}
¥end{PicFrameDot}
```

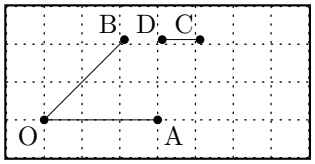
出力



入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)(-1,-1)
  ¥Pnode*(0, 0){O}[¥KSame][lb]
  ¥Pnode*(3, 0){A}[¥KSame][rb]
```

出力



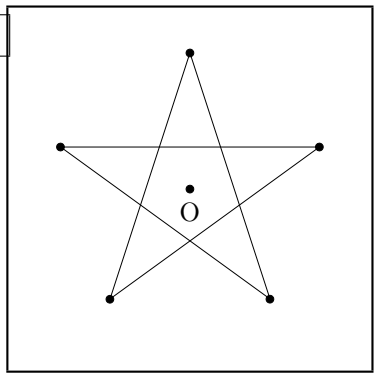
```
    ¥Pnode*(3,45){B}[¥KSame][lt]
    ¥Pnode*(2,0)(3,45){C}[¥KSame][lt]
    ¥Pnode*{C}(1,180){D}[¥KSame][lt]
    ¥KPath{AOB,CD}
  ¥end{PicFrameDot}
```

●星形に結ぶ、72° おきに座標を定義

入力

```
¥unitlength=6mm%
¥begin{PicFrameC}(8,8)
  ¥Pnode*(0,  0){O}[¥KSame][b]
  ¥Pnode*(3, 18){A}
  ¥Pnode*(3, 90){B}
  ¥Pnode*(3,162){C}
  ¥Pnode*(3,234){D}
  ¥Pnode*(3,306){E}
  ¥KPen{¥drawline}
  ¥KPath{ACEBDA}
¥end{PicFrameC}
```

出力

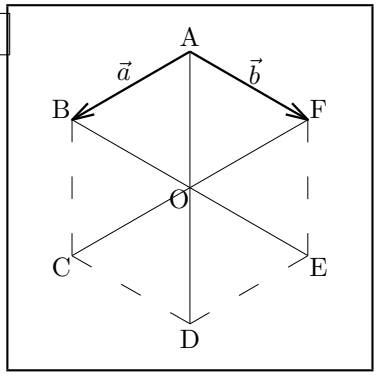


●六角形問題

入力

```
¥unitlength=6mm%
¥begin{PicFrameC}(8,8)
  ¥WordSep=2mm%
  ¥Pnode(0,0){O}[¥KSame][bl]
  ¥Pnode(3, 90){A}[¥KSame][t]
  ¥Pnode(3,150){B}[¥KSame][lt]
  ¥Pnode(3,210){C}[¥KSame][lb]
  ¥Pnode(3,270){D}[¥KSame][b]
  ¥Pnode(3,330){E}[¥KSame][rb]
  ¥Pnode(3, 30){F}[¥KSame][rt]
%   ¥KPen{¥drawline}
  ¥KPath[¥dashline{0.5}]{BCDEF}
  ¥KPath[¥drawline]{AD,BE,CF}
  ¥thicklines
  ¥KVec{AB}[$¥vec{a}$][t,r]
  ¥KVec{AF}[$¥vec{b}$][t,l]
  ¥thinlines
¥end{PicFrameC}
```

出力



4.2 定義した点から新たな点を定義

¥Knode，¥Pnode で定義した点を元に、新たな点が作成できます。ここで、作成した点も定義されますので、その座標を¥Knode，¥Pnode と同様に使用することができます。作成可能な点は、「内分点」「外分点」「2 点から指定された距離の点」「垂線の交点」「2 直線の交点」「三角形の重心」「三角形の外心」「三角形の内心」です。

この節のマクロ名は、長い命令でしかも他のマクロとぶつからないと思い、初めの「K」を省略しています。

4.2.1 共通項目

この章で使用可能な命令は次の通りです。

¥KSame	定義した座標名を表示する
¥Cnode	定義した座標を表示する
¥CnodeName	定義した座標名と座標を表示する

4.2.2 内分点

定義

¥Inode[*]{両端の座標}(内分する比){座標名}[文字][方向，文字位置]

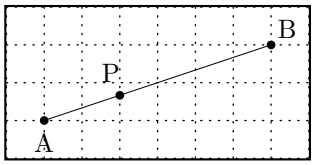
定義した 2 点を元に内分点を計算します。

●線分 AB を 1:2 の比に内分

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)
  ¥Knode*(1,1){A}[¥KSame][b]
  ¥Knode*(7,3){B}[¥KSame][rt]
  ¥Inode*{AB}(1:2){P}[¥KSame][t,r]
  ¥KPen{¥drawline}
  ¥KPath{AB}
¥end{PicFrameDot}
```

出力



4.2.3 外分点

定義

¥Enode[*]{両端の座標}(外分する比){座標名}[文字][方向，文字位置]

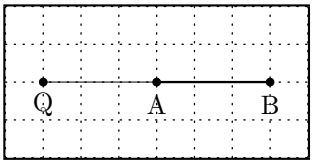
定義した 2 点を元に外分点を計算します。

●線分 AB を 1:2 比に外分

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)
  ¥Knode*(4,2){A}[¥KSame][b]
  ¥Knode*(7,2){B}[¥KSame][b]
  ¥thicklines
  ¥KLine{AB}
  ¥thinlines
  ¥Enode*{AB}(1:2){Q}[¥KSame][b]
  ¥KLine{QA}
¥end{PicFrameDot}
```

出力

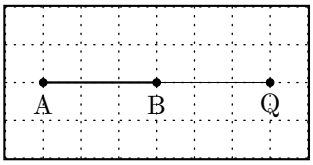


●線分 AB を 2:1 比に外分

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)
  ¥Knode*(1,2){A}[¥KSame][b]
  ¥Knode*(4,2){B}[¥KSame][b]
  ¥thicklines
  ¥KLine{AB}
  ¥thinlines
  ¥Enode*{AB}(2:1){Q}[¥KSame][b]
  ¥KLine{QB}
¥end{PicFrameDot}
```

出力



4.2.4 2 点から指定した距離の交点

定義

¥TwoCirclesRight[*]{両端の座標}(両端からの距離){座標名}[文字][方向, 文字位置]
¥TwoCirclesLeft[*]{両端の座標}(両端からの距離){座標名}[文字][方向, 文字位置]

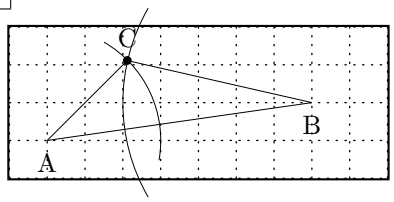
定義してある 2 点から指定した距離の点を計算します。以下の例を参考にして下さい。

● A から 3, B から 5 の距離の点の A から左側

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(10,4)(-1,-1)
  ¥Knode(0,0){A}[¥KSame][b]
  ¥Knode(7,1){B}[¥KSame][b]
  ¥TwoCirclesLeft*{AB}(3,5){C}[¥KSame][t]
  ¥KArc[6]{A}(350,60)
  ¥KArc[10]{B}(150,210)
  ¥KPath{CABC}
¥end{PicFrameDot}
```

出力

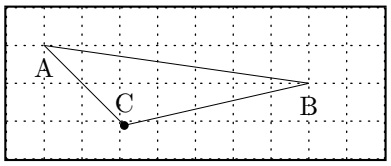


● A から 3, B から 5 の距離の点の A から右側

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(10,4)(-1,-1)
  ¥Knode(0,2){A}[¥KSame][b]
  ¥Knode(7,1){B}[¥KSame][b]
  ¥TwoCirclesRight*{AB}(3,5){C}[¥KSame][t]
  ¥KPath{CABC}
¥end{PicFrameDot}
```

出力



4.2.5　垂線の交点の座標

定義

¥Perpendicularfoot[*]{直線上の 2 点}{垂線上の他の点}{垂点座標名}[文字][方向, 文字位置]

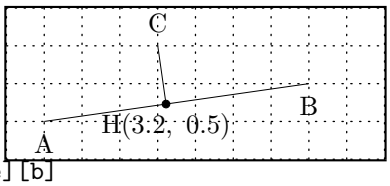
2 点を通る直線上に他の点からの垂線の座標を求めます。以下の例を参考にして下さい。

● A から 3, B から 5 の距離の点の A から左側

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(10,4)(-1,-1)
  ¥Knode(0,0){A}[¥KSame][b]
  ¥Knode(7,1){B}[¥KSame][b]
  ¥Knode(3,2){C}[¥KSame][t]
  ¥Perpendicularfoot*{AB}{C}{H}[¥CnodeName][b]
  ¥KPath{AB,CH}
¥end{PicFrameDot}
```

出力



4.2.6　２直線の交点の座標

定義

¥Intersection[*]{直線上の 2 点}{他の直線上の 2 点}{交点座標名}[文字][方向, 文字位置]

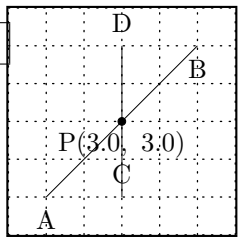
定義された 2 点を通る直線と、別に定義された 2 点を通る直線の交点の座標を求めます。以下の例を参考にして下さい。

●

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(6,6)
  ¥Knode(1,1){A}[¥KSame][b]
  ¥Knode(5,5){B}[¥KSame][b]
  ¥Knode(3,1){C}[¥KSame][t]
  ¥Knode(3,5){D}[¥KSame][t]
  ¥Intersection*{AB}{CD}{P}[¥CnodeName][b]
  ¥KPath{AB,CD}
¥end{PicFrameDot}
```

出力



4.2.7 三角形の重心

定義

¥Barycenter[*]{3 点の座標}{座標名}[文字][方向, 文字位置]

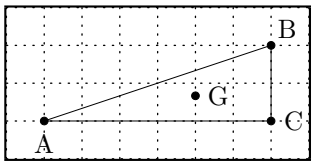
定義された 3 点を結ぶ三角形の重心の座標を求めます。

●三角形 ABC の重心

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)
  ¥Knode*(1,1){A}[¥KSame][b]
  ¥Knode*(7,3){B}[¥KSame][rt]
  ¥Knode*(7,1){C}[¥KSame][r]
  ¥Barycenter*{ABC}{G}[¥KSame][r]
  ¥KPen{¥drawline}
  ¥KPath{ABCA}
¥end{PicFrameDot}
```

出力



4.2.8　三角形の外心

定義

¥Circumcenter[*]{3 点の座標}{座標名}[文字][方向, 文字位置]

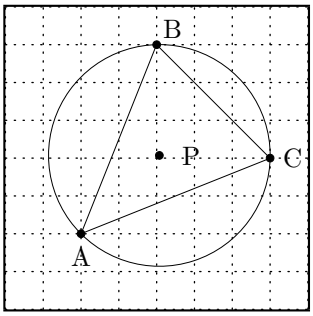
定義された 3 点を結ぶ三角形の外心の座標を求めます。その直後なら、半径は「¥KRadius」、直径は「¥KDiameter」に記憶されますので使用可能です。以下の例を参考にして下さい。

●三角形 ABC の外心

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,8)
  ¥Knode*(2,2){A}[¥KSame][b]
  ¥Knode*(4,7){B}[¥KSame][rt]
  ¥Knode*(7,4){C}[¥KSame][r]
  ¥Circumcenter*{ABC}{P}[¥KSame][r,l]
  ¥Kput{P}{¥circle{¥KDiameter}}
  ¥KPen{¥drawline}
  ¥KPath{ABCA}
¥end{PicFrameDot}
```

出力



4.2.9 三角形の内心

定義

¥Incenter[*]{3 点の座標}{座標名}[文字][方向, 文字位置]

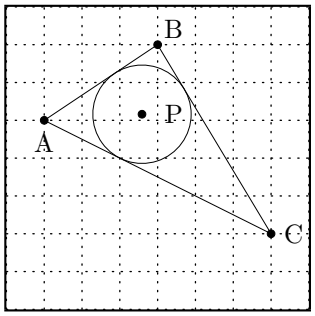
定義された 3 点を結ぶ三角形の内心の座標を求めます。その直後なら、半径は「¥KRadius」、直径は「¥KDiameter」に記憶されますので使用可能です。以下の例を参考にして下さい。

●三角形 ABC の内心

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,8)
  ¥Knode*(1,5){A}[¥KSame][b]
  ¥Knode*(4,7){B}[¥KSame][rt]
  ¥Knode*(7,2){C}[¥KSame][r]
  ¥Incenter*{ABC}{P}[¥KSame][r,l]
  ¥Kput{P}{¥circle{¥KDiameter}}
  ¥KPen{¥drawline}
  ¥KPath{ABCA}
¥end{PicFrameDot}
```

出力



4.3 定義された点を元の点とする

定義

¥Kput

定義された点を「¥put」のようにして使うことが出来ます。

第5章　点を結ぶ

5.1　eclairth.sty の拡張

5.1.1　線の種類 ¥KPen

¥KPen 命令は、¥KPath 命令で結ぶ線の種類を指定します。eclairth.sty の¥throughbrush と同じ働きというより、¥throughbrush のマクロをコピーして名前を変えただけです。理由は、スペルが長いからです。変更して元に戻したいときは、¥path と定義して下さい。

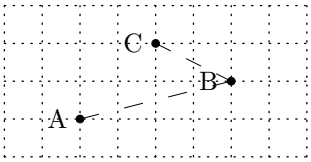
定義

¥KPen{線種}

入力

```
¥unitlength=5mm%
¥begin{PicDot}(8,4)
  ¥Knode*(2,1){A}[¥KSame][l]
  ¥Knode*(6,2){B}[¥KSame][l]
  ¥Knode*(4,3){C}[¥KSame][l]
  ¥KPen{¥dashline{0.5}}
  ¥KPath{ABC}
¥end{PicDot}
```

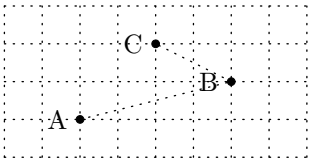
出力



入力

```
¥unitlength=5mm%
¥begin{PicDot}(8,4)
  ¥Knode*(2,1){A}[¥KSame][l]
  ¥Knode*(6,2){B}[¥KSame][l]
  ¥Knode*(4,3){C}[¥KSame][l]
  ¥KPen{¥dottedline{0.2}}
  ¥KPath{ABC}
¥end{PicDot}
```

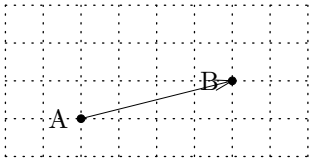
出力



入力

```
¥def¥ArrowHeadSize{0.5}%
¥unitlength=5mm%
¥begin{PicDot}(8,4)
  ¥Knode*(2,1){A}[¥KSame][l]
  ¥Knode*(6,2){B}[¥KSame][l]
  ¥KPen{¥arrow¥drawline}
  ¥KPath{AB}
¥end{PicDot}
```

出力



5.1.2　置いた点を結ぶ

¥Knode,¥Pnode 命令等は点の座標を記憶しておきます。これを結んで表示するのが¥KPath 命令です。基本的に eclairth.sty の¥through と同じ働きをしますが、カンマで区切れば複数指定可能にしました。また、¥KPen の線種を最初に指定しておくと、¥KPen の役割もします。これらの命令は、eclairth.sty の¥through のマクロを改良しています。

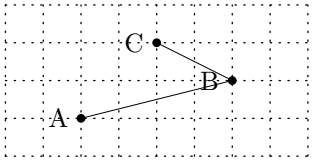
定義

¥KPath[線種]{node 座標 (,node 座標, …)}

入力

```
¥unitlength=5mm%
¥begin{PicDot}(8,4)
  ¥Knode*(2,1){A}[¥KSame][l]
  ¥Knode*(6,2){B}[¥KSame][l]
  ¥Knode*(4,3){C}[¥KSame][l]
  ¥KPen{¥drawline}
  ¥KPath{ABC}
¥end{PicDot}
```

出力



5.2 2 点を結ぶ

この節では、前節の続きで定義した点の内、2 点を結ぶ直線、ベクトルについて解説します。この章で扱う命令は以下の表のものです。

基本命令	範囲を実線で示す	範囲を破線で示す
¥KLine	¥KLineArc	¥KLineDashArc
¥KVec	¥KVecArc	¥KVecDashArc
¥KLineName	¥KLineNameArc	¥KLineNameDashArc

●この章に限り「方向, 文字位置」の所に以下の命令が追加使用可能です。

left	中点で始点から左 90° の位置に表示
right	中点で始点から右 90° の位置に表示

距離は¥WordSep を使用します。

5.2.1 2 点を結び、名前等を入れる

2 つの座標を結び、名前を入れます。数字でもかまいません。2 点の中点を基準として、「方向・文字位置」に基づいて文字等を表示します。

定義

¥KLine{始点終点の座標名}[文字][方向, 文字位置]

●「文字」の所に以下の命令が使用可能です。

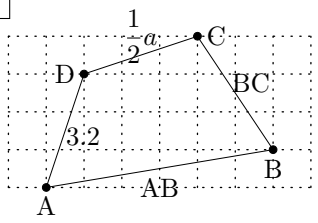
¥KSame	2 点の座標名を表示する
¥KTrue	2 点間の実距離を表示する

¥KTrue の小数の有効桁は¥KFigure で指定します。

入力

```
¥unitlength=5mm%
¥begin{PicDot}(8,4)
  ¥WordSep=2.5mm%
  ¥Knode*(1,0){A}[¥KSame][b]
  ¥Knode*(7,1){B}[¥KSame][b]
  ¥Knode*(5,4){C}[¥KSame][r]
  ¥Knode*(2,3){D}[¥KSame][l]
  ¥KLine{AB}[¥KSame][b]
  ¥KLine{BC}[¥KSame][right]
  ¥KLine{CD}[$¥displaystyle¥frac{1}{2}a$][t,tl]
  ¥KLine{DA}[¥KTrue][left]
  ¥end{PicDot}
```

出力



5.2.2　ベクトルと名前を入れる ¥KVec

2 つの座標を結んだ線に名前を入れます。矢の長さは、¥KVecHeadSize で絶対的に定義されています。初期値は¥KVecHeadSize=3mm です。

定義

¥KVec{始点終点の座標名}[文字][方向, 文字位置]

●「文字」の所に以下の命令が使用可能です。

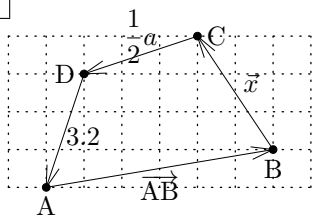
¥KSame	始点から終点へのベクトル名
¥KTrue	2 点間の実距離を表示する

¥KTrue の小数の有効桁は¥KFigure で指定します。

入力

```
¥unitlength=5mm%
¥begin{PicDot}(8,4)
  ¥WordSep=2.5mm%
  ¥Knode*(1,0){A}[¥KSame][b]
  ¥Knode*(7,1){B}[¥KSame][b]
  ¥Knode*(5,4){C}[¥KSame][r]
  ¥Knode*(2,3){D}[¥KSame][l]
  ¥KVec{AB}[¥KSame][b]
  ¥KVec{BC}[$¥vec{x}$][right]
  ¥KVec{CD}[$¥displaystyle¥frac{1}{2}a$][t,tl]
  ¥KVec{DA}[¥KTrue][left]
¥end{PicDot}
```

出力



5.2.3 線に名前を入れる

2 つの座標の間に名前を入れます。数字でもかまいません。この命令は、線を描かない以外は、¥KLine と同じです。

定義

¥KLineName{始点終点の座標名}[文字][方向, 文字位置]

●「文字」の所に以下の命令が使用可能です。

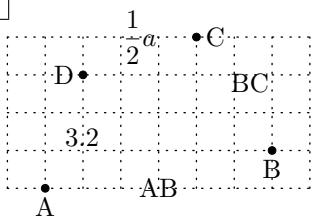
¥KSame	2 点の座標名を表示する
¥KTrue	2 点間の実距離を表示する

¥KTrue の小数の有効桁は¥KFigure で指定します。

入力

```
¥unitlength=5mm%
¥begin{PicDot}(8,4)
  ¥WordSep=2.5mm%
  ¥Knode*(1,0){A}[¥KSame][b]
  ¥Knode*(7,1){B}[¥KSame][b]
  ¥Knode*(5,4){C}[¥KSame][r]
  ¥Knode*(2,3){D}[¥KSame][l]
  ¥KLineName{AB}[¥KSame][b]
  ¥KLineName{BC}[¥KSame][right]
  ¥KLineName{CD}[$¥displaystyle¥frac{1}{2}a$][t,tl]
  ¥KLineName{DA}[¥KTrue][left]
¥end{PicDot}
```

出力



5.2.4 範囲を実線で示す

ここまで出てきた 3 つの命令の始点から終点にかけて、(表示) 文字を中心に範囲を実線で示します。範囲は、2 点と表示する文字の基準点の 3 点を通る円弧を描きます。文字にかからないように、文字の基準点から指定した長さは描きません。ただし、¥KAngle の中心マクロに掃き出しますので、時間がかかります。

定義

¥KLineArc{始点終点の座標名}[文字][方向, 文字位置]
¥KVecArc{始点終点の座標名}[文字][方向, 文字位置]
¥KLineNameArc{始点終点の座標名}[文字][方向, 文字位置]

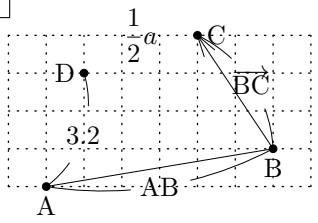
●文字にかからない長さは以下の命令で変更可能

説明	命令	初期値
開始方向から文字の基準点まで	¥KLineArcLenS	4mm
文字の基準点から終了方向まで	¥KLineArcLenE	4mm

入力

```
¥unitlength=5mm%
¥begin{PicDot}(8,4)
  ¥WordSep=2.5mm%
  ¥Knode*(1,0){A}[¥KSame][b]
  ¥Knode*(7,1){B}[¥KSame][b]
  ¥Knode*(5,4){C}[¥KSame][r]
  ¥Knode*(2,3){D}[¥KSame][l]
  ¥KLineArc{AB}[¥KSame][b]
  ¥KVecArc{BC}[¥KSame][right]
  ¥KLineName{CD}[$¥displaystyle¥frac{1}{2}a$][t,tl]
  ¥KLineNameArc{DA}[¥KTrue][left]
¥end{PicDot}
```

出力



5.2.5　範囲を破線で示す

こちらは、(表示) 文字を中心に範囲を破線で示します。ただし、きちんと表示しないバグがあります。こちらも、文字間の空白は実線と同様、またこちらも同様に¥KAngle の中心マクロに掃き出しますので、時間がかかります。

定義

¥KLineDashArc{始点終点の座標名}[文字][方向, 文字位置]
¥KVecDashArc{始点終点の座標名}[文字][方向, 文字位置]
¥KLineNameDashArc{始点終点の座標名}[文字][方向, 文字位置]

●破線部分と空白部分の設定は以下の命令で変更可能

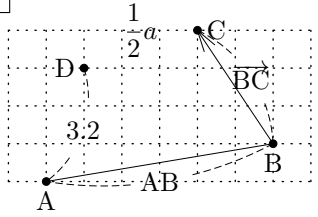
説明	命令	初期値
破線部分	¥KDashArcLine	1mm
空白部分	¥KDashArcSpace	0.5mm

●文字にかからない長さは以下の命令で変更可能

説明	命令	初期値
開始方向から文字の基準点まで	¥KLineArcLenS	4mm
文字の基準点から終了方向まで	¥KLineArcLenE	4mm

入力

出力



```
¥unitlength=5mm%
¥begin{PicDot}(8,4)
  ¥WordSep=2.5mm%
  ¥Knode*(1,0){A}[¥KSame][b]
  ¥Knode*(7,1){B}[¥KSame][b]
  ¥Knode*(5,4){C}[¥KSame][r]
  ¥Knode*(2,3){D}[¥KSame][l]
  ¥KLineDashArc{AB}[¥KSame][b]
  ¥KVecDashArc{BC}[¥KSame][right]
  ¥KLineName{CD}[$¥displaystyle¥frac{1}{2}a$][t,tl]
  ¥KLineNameDashArc{DA}[¥KTrue][left]
¥end{PicDot}
```

第6章　円弧・一般角・直角

eepic.sty の¥arc を使用すれば円弧・角度を表示することができます。しかし、孤度法でしかも tpic に吐き出すので、時計方向の角を指定しなければなりません。そこで、度数法・一般角に対応した、時計の逆回りの円弧・角度を表示するための命令を定義しました。いずれのマクロも処理の最後は、eepic.sty の¥arc に掃き出しています。

以下の表は、この章に出てくる基本命令です。

●実線で円弧を描く

基本形	正の方向に矢印を付ける	負の方向に矢印を付ける
¥KArc	¥ArrowKArc	¥RevArrowKArc
¥KAngle	¥ArrowKAngle	¥RevArrowKAngle
¥PAngle	¥ArrowPAngle	¥RevArrowPAngle

●破線で円弧を描く

基本形	正の方向に矢印を付ける	負の方向に矢印を付ける
¥KDashArc	¥ArrowKDashArc	¥RevArrowKDashArc
¥KDashAngle	¥ArrowKDashAngle	¥RevArrowKDashAngle
¥PDashAngle	¥ArrowPDashAngle	¥RevArrowPDashAngle

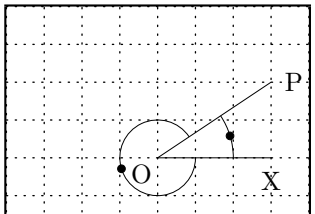
6.1　共通項目

まずは、この章で共通な項目について説明します。

6.1.1　文字表示の基準点

円弧・一般角と同時に表示する文字の基準点について説明します。基準点は、表示する円弧・一般角上の中央部分になるようにしてあります。次の図の黒い点がその基準点です。この基準点を元に、「方向」「文字位置」を指定して下さい。

この条件外に表示したいときは、改めて¥Knode 等で設定して下さい。



●基準点を表示する

入力

```
¥unitlength=5mm%
¥begin{PicFrameDotC}(8,8)
  ¥Knode(0,0){O}[¥KSame][bl]
  ¥Knode(3,0){X}[¥KSame][b]
  ¥Knode(3,2){P}[¥KSame][r]
  ¥KPen{¥drawline}
  ¥KPath{XOP}
  ¥KAngle[2]{POX}[¥circle*{0.2}]
  ¥KAngle[4]{XOP}[¥circle*{0.2}]
¥end{PicFrameDotC}
```

出力

6.1.2　文字

「文字」の所に以下の命令が使用可能です。

¥KSame	座標名を表示する
¥KTrue	角度を度数で表示する

¥KTrue は座標を指定したときも、自動で角度を計算してくれます。また小数の有効桁は¥KFigure で指定します。

6.1.3　円弧の直径

角の直径は,「¥AngleLen」で指定します。初期値は ¥AngleLen=10mm です。¥circle や¥arc と互換性を持たせる為に、直径にしたので注意して下さい。

このほか、この章の全ての定義の「直径」の所に数値を入れておくと、その角に関しての直径として処理します。ただし、単位は座標系なので相対的です。

6.2 円弧・角度の表示

6.2.1 ¥arc の拡張 ¥KArc

定義

¥KArc[直径](\$x\$座標,\$y\$座標)(開始角, 終了角)[文字][方向, 文字位置]
¥KArc[直径]{node 座標}(開始角, 終了角)[文字][方向, 文字位置]

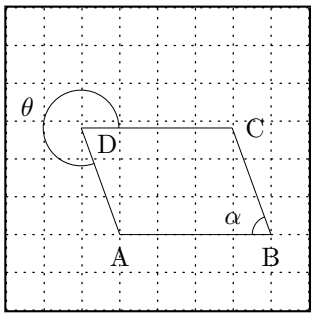
¥KArc は、指定した座標を中心をして一般角の開始角、終了角を指定することで、円弧を描きます。ただし、角度は degree(度) で指定します。開始角と終了角の指定範囲は、-360° から 360° です。また、時計の反対周りに開始角と終了角がくるように指定し下さい。

● ¥KArc の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][rb,l]
  ¥KPath[¥drawline]{ABCDA}
  ¥KArc[1](4,0)(110,180)[$¥alpha$][l,b]
  ¥KArc{D}(0,290)[$¥theta$][l]
¥end{PicFrameDot}
```

出力



6.2.2 3 つの座標から角を表示 ¥KAngle

定義

¥KAngle[直径]{3 つの座標名}[文字][方向, 文字位置]

定義した 3 つの座標のうち、真ん中の座標を中心として角の記号を表示する命令です。初めの座標と最後の座標の間に角を表示します。時計の逆周りの順に指定しておくように。順番を変えると反対側に描かれます。

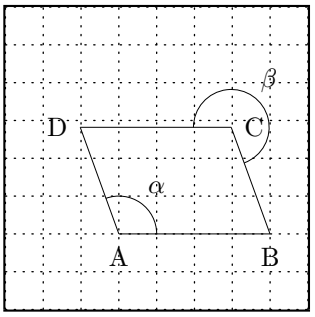
真ん中の座標に対し、他の 2 点の座標の一般角を計算して表示します。しかし、計算は単に 2 分法を使用しているだけで、「とにかく動けばいいや」で作成しましたので、少々計算時間がかかります。将来的に早くするつもりです。

● ¥KAngle の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][l]
  ¥KPath[¥drawline]{ABCDA}
  ¥KAngle{BAD}[$¥alpha$][rt]
  ¥KAngle{BCD}[$¥beta$][rt]
¥end{PicFrameDot}
```

出力



6.2.3　¥Pnode を利用して角を表示 ¥PAngle

定義

¥PAngle[直径]{３つの座標名}[文字][方向, 文字位置]

¥Pnode で定義した時、原点に対しての角度を入力している場合、ここではそれを利用して、角度を表示します。当然、¥KAngle より高速です。ただし、原点に対して座標を入力したときのみ使用できるので、あまり使い道がないかもしれません。

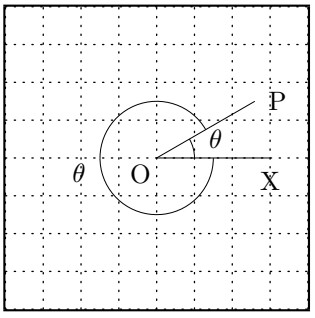
定義した 3 つの座標のうち、真ん中の座標を中心として角の記号を表示する命令です。初めの座標と最後の座標のの間に角を表示します。時計の逆周りの順に指定しておくように。順番を変えると反対側に描かれます。

● ¥PAngle の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDotC}(8,8)
  ¥Pnode(0,0){O}[¥KSame][bl]
  ¥Pnode(3,0){X}[¥KSame][b]
  ¥Pnode(3,30){P}[¥KSame][r]
  ¥KPath[¥drawline]{XOP}
  ¥PAngle{XOP}[$¥theta$][r,b]
  ¥PAngle[3]{POX}[$¥theta$][l]
¥end{PicFrameDotC}
```

出力



6.3 矢印を付ける

6.3.1 矢印の大きさ

この節に出てくる矢印の大きさは、¥KArrowLen で設定しています。初期値は¥KArrowLen=2mm です。小さい角度の時は、この大きさを変えて下さい。

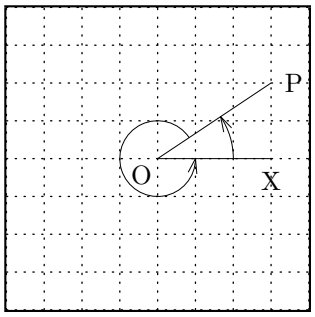
また、初期設定で矢印は塗りつぶされます。塗られない矢印との切り替えは、「¥KArrowHeadNoFilltrue」とします。戻すためは、「¥KArrowHeadNoFillfalse」とします。次の例でもわかるように、余りきれいではありません。

●塗られない矢を表示する

入力

```
¥unitlength=5mm%
¥begin{PicFrameDotC}(8,8)
  ¥Knode(0,0){O}[¥KSame][bl]
  ¥Knode(3,0){X}[¥KSame][b]
  ¥Knode(3,2){P}[¥KSame][r]
  ¥KPen{¥drawline}
  ¥KPath{XOP}
  ¥KArrowHeadNoFilltrue% ここ
  ¥ArrowKAngle[2]{POX}
  ¥ArrowKAngle[4]{XOP}
  ¥KArrowHeadNoFillfalse% ここ
¥end{PicFrameDotC}
```

出力



6.3.2 正の方向に付ける

これまで出てきた 3 種類の角度の表示命令の前に「Arrow」を付けると一般角の正の方向に矢印が付きます。矢印をつける以外の使い方は同じなので省略します。

定義

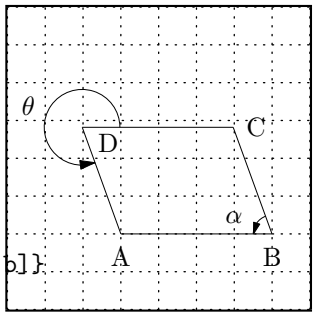
¥ArrowKArc[直径](中心)(開始角, 終了角)[文字][方向, 文字位置]
¥ArrowKArc[直径]{座標名}(開始角, 終了角)[文字][方向, 文字位置]
¥ArrowKAngle[直径]{３つの座標名}[文字][方向, 文字位置]
¥ArrowPAngle[直径]{３つの座標名}[文字][方向, 文字位置]

● ¥ArrowKArc の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][rb,1]
  ¥KPath[¥drawline]{ABCDA}
  {¥KArrowLen=1.5mm
  ¥ArrowKArc[1](4,0)(110,180)[$¥alpha$][l,b]}
  ¥ArrowKArc{D}(0,290)[$¥theta$][l]
¥end{PicFrameDot}
```

出力

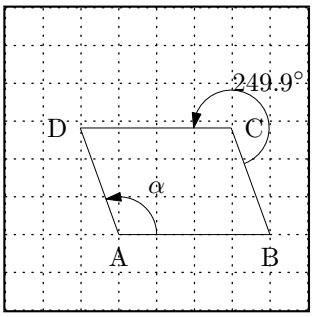


● ¥ArrowKAngle の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][l]
  ¥KPath[¥drawline]{ABCDA}
  ¥ArrowKAngle{BAD}[$¥alpha$][rt]
  ¥ArrowKAngle{BCD}[¥KTrue][rt]
¥end{PicFrameDot}
```

出力

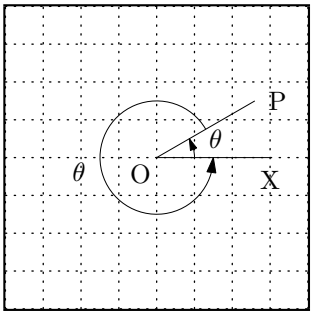


● ¥ArrowPAngle の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDotC}(8,8)
  ¥Pnode(0,0){O}[¥KSame][bl]
  ¥Pnode(3,0){X}[¥KSame][b]
  ¥Pnode(3,30){P}[¥KSame][r]
  ¥KPath[¥drawline]{XOP}
  {¥KArrowLen=1.5mm
  ¥ArrowPAngle{XOP}[$¥theta$][r,b]}
  ¥ArrowPAngle[3]{POX}[$¥theta$][l]
¥end{PicFrameDotC}
```

出力



6.3.3 負の方向に付ける

これまで出てきた 3 種類の角度の表示命令の前に「RevArrow」を付けると一般角の負の方向に矢印が付きます。矢印をつける以外の使い方は同じなので省略します。

定義

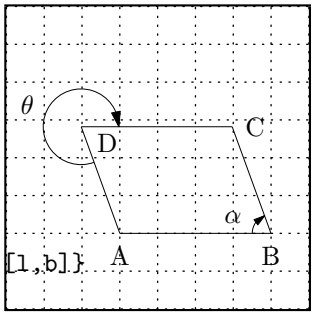
¥RevArrowKArc[直径](中心)(開始角, 終了角)[文字][方向, 文字位置]
¥RevArrowKArc[直径]{座標名}(開始角, 終了角)[文字][方向, 文字位置]
¥RevArrowKAngle[直径]{３つの座標名}[文字][方向, 文字位置]
¥RevArrowPAngle[直径]{３つの座標名}[文字][方向, 文字位置]

● ¥RevArrowKArc の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][rb,l]
  ¥KPath[¥drawline]{ABCDA}
  {¥KArrowLen=1.5mm
  ¥RevArrowKArc[1](4,0)(110,180)[$¥alpha$][l,b]}
  ¥RevArrowKArc{D}(0,290)[$¥theta$][l]
¥end{PicFrameDot}
```

出力

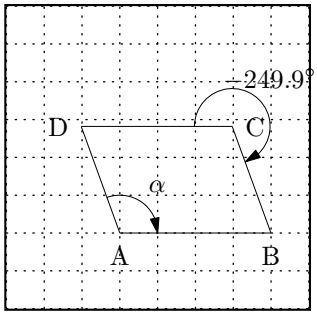


● ¥RevArrowKAngle の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][l]
  ¥KPath[¥drawline]{ABCDA}
  ¥RevArrowKAngle{BAD}[$¥alpha$][rt]
  ¥RevArrowKAngle{BCD}[¥KTrue][rt]
¥end{PicFrameDot}
```

出力

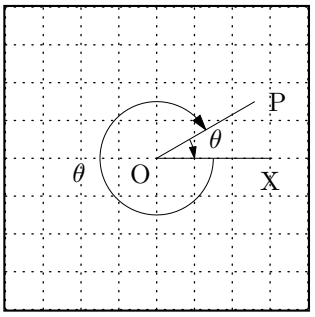


● ¥RevArrowPAngle の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDotC}(8,8)
  ¥Pnode(0,0){O}[¥KSame][bl]
  ¥Pnode(3,0){X}[¥KSame][b]
  ¥Pnode(3,30){P}[¥KSame][r]
  ¥KPath[¥drawline]{XOP}
  {¥KArrowLen=1.5mm
  ¥RevArrowPAngle{XOP}[$¥theta$][r,b]}
  ¥RevArrowPAngle[3]{POX}[$¥theta$][l]
¥end{PicFrameDotC}
```

出力



6.4 破線で円弧・一般角を描く

6.4.1 破線のパターン

ここで説明する破線は以下の補助命令の数値で設定されています。

補助命令	説明	初期値
¥KDashArcLine	破線の表示部分の長さ	1mm
¥KDashArcSpace	破線の空白部分の長さ	0.5mm

6.4.2 破線で描く

円弧・一般角を破線で描きます。それ以外は前と同じことなので、定義と簡単な例だけ載せます。

定義

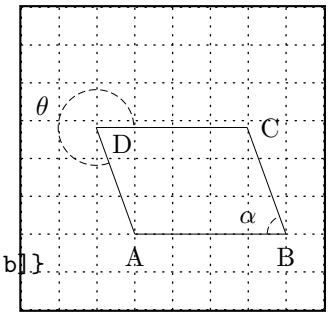
```
¥KDashArc[直径](中心)(開始角, 終了角)[文字][方向, 文字位置]
¥KDashArc[直径]{座標名}(開始角, 終了角)[文字][方向, 文字位置]
¥KDashAngle[直径]{３つの座標名}[文字][方向, 文字位置]
¥PDashAngle[直径]{３つの座標名}[文字][方向, 文字位置]
```

● ¥KDashArc の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][rb,l]
  ¥KPath[¥drawline]{ABCDA}
  {¥KArrowLen=1.5mm
  ¥KDashArc[1](4,0)(110,180)[$¥alpha$][l,b]}
  ¥KDashArc{D}(0,290)[$¥theta$][l]
¥end{PicFrameDot}
```

出力



6.4.3　矢印を正の方向へ付ける

これも破線になる以外これまで出てきたものと同じなので、定義と簡単な例だけ載せます。

定義

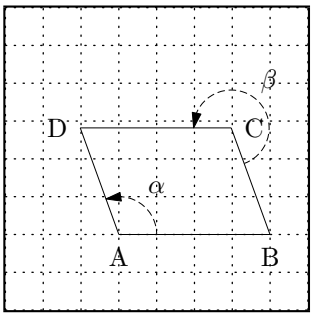
¥ArrowKDashArc[直径](中心)(開始角, 終了角)[文字][方向, 文字位置]
¥ArrowKDashArc[直径]{座標名}(開始角, 終了角)[文字][方向, 文字位置]
¥ArrowKDashAngle[直径]{３つの座標名}[文字][方向, 文字位置]
¥ArrowPDashAngle[直径]{３つの座標名}[文字][方向, 文字位置]

● ¥ArrowKDashAngle の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][l]
  ¥KPath[¥drawline]{ABCDA}
  ¥ArrowKDashAngle{BAD}[$¥alpha$][rt]
  ¥ArrowKDashAngle{BCD}[$¥beta$][rt]
¥end{PicFrameDot}
```

出力



6.4.4　矢印を負の方向に付ける

これも破線になる以外これまで出てきたものと同じなので、定義と簡単な例だけ載せます。

定義

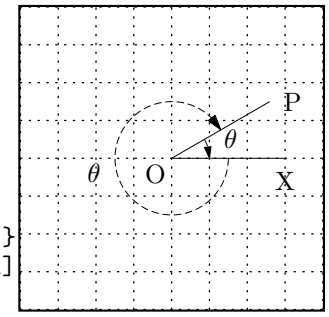
¥RevArrowKDashArc[直径](中心)(開始角, 終了角)[文字][方向, 文字位置]
¥RevArrowKDashArc[直径]{座標名}(開始角, 終了角)[文字][方向, 文字位置]
¥RevArrowKDashAngle[直径]{３つの座標名}[文字][方向, 文字位置]
¥RevArrowPDashAngle[直径]{３つの座標名}[文字][方向, 文字位置]

● ¥RevArrowPDashAngle の使用例

入力

```
¥unitlength=5mm%
¥begin{PicFrameDotC}(8,8)
  ¥Pnode(0,0){O}[¥KSame][bl]
  ¥Pnode(3,0){X}[¥KSame][b]
  ¥Pnode(3,30){P}[¥KSame][r]
  ¥KPath[¥drawline]{XOP}
  {¥KArrowLen=1.5mm
  ¥RevArrowPDashAngle{XOP}[$¥theta$][r,b]}
  ¥RevArrowPDashAngle[3]{POX}[$¥theta$][l]
¥end{PicFrameDotC}
```

出力



6.5　応用例

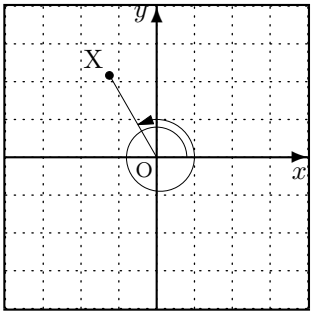
6.5.1　一般角で 360° 以上を描く

● 360° を越える角度・上側

入力

```
¥unitlength=5mm%
¥begin{AxesFrameDotC}(8,8)
  ¥Pnode(0,0){O}
  ¥Pnode*(2.5,120){A}[X][lt]
  ¥KPath[¥drawline]{OA}
  ¥KArc[1.6](0,0)(0,180)
  % 1 単位は 5mm だから 1mm は 0.2
  ¥KArc[1.8](0.1,0)(180,360)
  ¥ArrowKArc[2.0](0,0)(0,120)
¥end{AxesFrameDotC}
```

出力

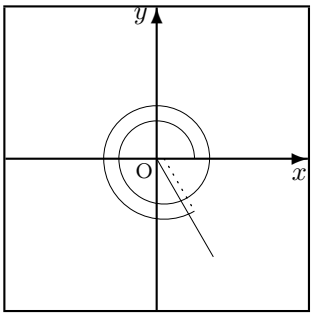


● 360° を越える角度・下側

入力

```
¥unitlength=5mm%
¥begin{AxesFrameC}(8,8)
  ¥Pnode(0,0){O}
  ¥Pnode(3,300){A}
%  ¥KPath[¥dottedline{0.2}]{OA}
  ¥KPath{OA}
  ¥Pnode(0.2,0){B}
  ¥Pnode(0.2,0)(1.6,300){C}
%  ¥KPath{BC}
  ¥KPath[¥dottedline{0.2}]{BC}
%  ¥Enode{OC}(2:1){D}
%  ¥KPath[¥dottedline{0.2}]{OD}
  ¥KArc(0,0)(0,180)
  % 1 単位は 5mm だから 2mm は 0.4
  ¥KArc[2.4](0.2,0)(180,360)
  ¥KArc[2.8](0,0)(0,180)
  ¥KArc[3.2](0.2,0)(180,300)
¥end{AxesFrameC}
```

出力



6.6 直角の記号表示 ¥KNinty

2 本の線の交点に直角の記号を表示します。空間座標や斜交座標に直角の記号を表示する事もあるので、直角でなくても記号としてでます。ようは、一辺が¥NintyLen の平行四辺形として考えているだけです。

定義

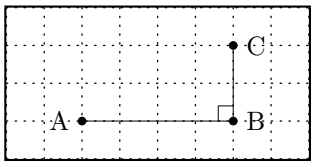
KNinty{３つの座標名}

直角の大きさは,「¥NintyLen」で指定します。初期値は¥NintyLen=2mm です。
KNinty{ABC}もKNinty{CBA}同じ場所に表示します。

入力

```
¥unitlength=5mm%
¥begin{PicFrameDot}(8,4)
  ¥Knode*(2,1){A}[¥KSame][l]
  ¥Knode*(6,1){B}[¥KSame][r]
  ¥Knode*(6,3){C}[¥KSame][r]
  ¥KPen{¥drawline}
  ¥KPath{ABC}
  ¥KNinty{ABC}
¥end{PicFrameDot}
```

出力



第7章　空間座標を平面で表示 ¥Snode

現在作業凍結の ¥Snode です。平面座標に関わっていて、現在は旧マクロの名前のを変更しただけです。

空間座標を平面座標へ変換して表示します。変換の行列は

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -0.5 & 1 & 0 \\ -0.375 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

だと思います (自作のマクロの解析に自信がない)。数値は見た目で選びました。eclarith.sty の¥obbase ならびに¥obcalc を使用しています。

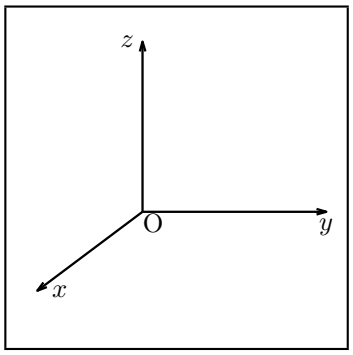
この命令は欠点があり，何度かコンパイルしてみないと picture 環境の大きさがわからないことです。

まず、空間の正の座標軸を表示してみます。

入力

```
{¥def¥ArrowHeadSize{0.3}
    ¥WordSep=2mm
    ¥setlength{¥unitlength}{4.5mm}%
    ¥begin{PicFrame}(10,10)(-4,-4)
      ¥Snode(0,0,0){O}[¥KSame][br]
      ¥Snode(6.2,0,0){X}[$x$][r,l]
      ¥Snode(0,5.4,0){Y}[$y$][b]
      ¥Snode(0,0,5){Z}[$z$][l]
      ¥thicklines
      ¥KPen{¥arrow¥drawline}
      ¥KPath{OX,OY,OZ}
     ¥end{PicFrame}
    }
```

出力

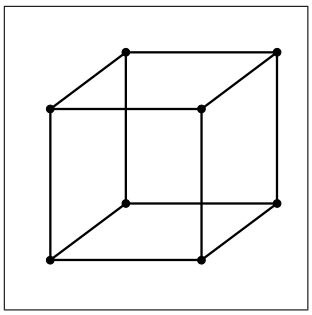


次に、一辺の長さが 1 の立方体を描いてみます。左奥を原点とします。

入力

```
{¥def¥ArrowHeadSize{0.3}
    ¥WordSep=2mm
    ¥setlength{¥unitlength}{20mm}%
    ¥begin{picture}(2,2)(-0.8,6.3)
      ¥put(-0.8,-0.7){¥framebox(2,2){}}
      ¥Snode*(0,0,0){A}
      ¥Snode*(1,0,0){B}
      ¥Snode*(1,1,0){C}
      ¥Snode*(0,1,0){D}
%
      ¥Snode*(0,0,1){E}
      ¥Snode*(1,0,1){F}
      ¥Snode*(1,1,1){G}
      ¥Snode*(0,1,1){H}
%
      ¥thicklines
      ¥KPen{¥drawline}
      ¥KPath{ABCDA}
      ¥KPath{EFGHE}
      ¥KPath{AE,BF,CG,DH}
    ¥end{picture}
}
```

出力



関連図書

[1] 磯崎秀樹「LaTeX 自由自在」(サイエンス社、1992)

[2] 奥村晴彦「LaTeX2e 美文書作成入門」(技術評論社、1997)

[3] 乙部厳己＋江口庄英「pLaTeX 2_ε for Windows Another Manual Vol.1 Basic Kit」(ソフトバンク、1996)

[4] 乙部厳己＋江口庄英「pLaTeX 2_ε for Windows Another Manual Vol.2 Extended Kit」(ソフトバンク、1997)

[5] 藤田眞作「LaTeX マクロの八衢」(アジソン・ウェスレイ・パブリッシャーズ・ジャパン、1995)