ABC 実習マニュアル

第一版
2005 年 3 月 8 日（火）

目次

第 1 章　ABC の基礎

1.1　ABC とは何か

1.1.1　楽譜浄書ソフト

楽譜を浄書するという仕事は、かつては特殊な職人芸を必要とするものでしたが、コンピュータのソフトを使って楽譜を浄書することができるようになった現在では、それほど高度な技能とは言えないものになっています（ただし、出版物に載せることのできる品質の楽譜を作るためには、コンピュータのソフトを使ったとしても、かなりの熟練が必要です）。

楽譜を浄書するためのコンピュータのソフトは、「楽譜浄書ソフト」(notation software) と呼ばれます。

楽譜浄書ソフトは、楽譜を作成する方式の違いによって 2 種類に分類することができます。ひとつは、コンピュータの画面を紙に見立てて、その上に記号を並べていくことによって楽譜を作成する、という方式のもので、もうひとつは、まず文字を使って楽曲を記述して、その結果としてできた文書を楽譜に変換する、という方式のものです。

1.1.2　音楽記述言語

楽曲を文字で記述するために使われる言語は、「音楽記述言語」(music description language) と呼ばれます。音楽記述言語は、楽譜に変換される文書を書くことを目的とするものと、MIDI データに変換される文書を書くことを目的とするものに分類することができます（ただし、両者の境界線は明確ではありません）。

楽譜に変換される文書を書くための音楽記述言語としては、MusicTEX、Lilypond、Mup、Philip's Music Writer、ABC など、さまざまなものがあります。

1.1.3　ABC の概要

ABC は、Chris Walshaw さんによって設計された音楽記述言語です。ABC を使って楽曲を記述した文書は、「ABC 譜」(ABC tune) と呼ばれます。

ABC について詳細な情報を知りたい方は、たとえば、

The ABC Music project　`http://abc.sourceforge.net/`

などを参照していただくといいでしょう。

1.1.4　abcm2ps

ABC 譜を取り扱うソフトは、さまざまな人によってさまざまなものが書かれています。Jean-François Moine さんによって書かれた abcm2ps も、それらのソフトのうちのひとつです。このソフトは、ABC 譜を読み込んで、それを楽譜に変換して、その結果を PostScript のデータとして出力します。ただし、abcm2ps が扱う ABC は、複数の声部を持つ楽譜を記述する機能などをオリジナルな ABC に追加したものです。

abcm2ps については、作者自身によって運営されている、

Jef's page　`http://moinejf.free.fr/`

というサイトに、さらに詳細な情報があります。

なお、この「ABC 実習マニュアル」という文章は、abcm2ps で処理することのできる ABC を使って楽譜を作成する方法について解説したものです。

1.2　abcm2ps の使い方

1.2.1　ABC 譜の入力

ABC 譜というのは単なる文字列ですので、それを入力するためのソフトとしては、何らかのテキストエディターがあれば充分です。

それでは、次の ABC 譜を入力して、`cmajor.abc` というファイルに保存してください。

ABC 譜の例　`cmajor.abc`

```
X: 1
M: none
L: 1/1
```

```
K: C
CDEFGABccBAGFEDC
```

なお、ABC 譜を格納するファイルには、このように .abc という拡張子を付けます。

1.2.2 楽譜への変換

abcm2ps を起動するコマンドは、基本的には、

abcm2ps パス名

と書きます。このコマンドをシェルに入力すると、パス名で指定されたファイルに格納されている ABC 譜が処理されて、それを PostScript に変換した結果が、カレントディレクトリの Out.ps というファイルに出力されます。

それでは実際に試してみましょう。 cmajor.abc があるディレクトリをカレントディレクトリにして、

```
abcm2ps cmajor.abc
```

というコマンドをシェルに入力してください。すると、先ほど入力した ABC 譜が PostScript に変換されて、カレントディレクトリの Out.ps というファイルに出力されます。ちなみに、PostScript のデータを表示することのできるソフトを使って Out.ps を開くと、ハ長調の音階の楽譜が表示されるはずです。

なお、引数を何も渡さないで abcm2ps を起動すると、コマンドのオプションの一覧が表示されますので、試してみてください。

1.2.3 データ形式の変換

abcm2ps が出力することのできる楽譜のデータ形式は、PostScript のみです。ですから、それ以外のデータ形式で表現された楽譜がほしい場合は、別のソフトを使ってデータ形式を変換する必要があります。たとえば、PDF の形式の楽譜がほしいならば、ps2pdf などのソフトを使って、PostScript から PDF へデータ形式を変換しないといけないわけです。

ps2pdf を使って PostScript のデータを PDF に変換したいときは、

ps2pdf パス名

というコマンドをシェルに入力します。そうすると、パス名で指定されたファイルに格納されている PostScript のデータを PDF に変換した結果が、カレントディレクトリのファイルに出力されます。結果が出力されるファイルの名前は、PostScript のファイルの拡張子を .pdf に変更したものです。たとえば、

```
ps2pdf Out.ps
```

というコマンドをシェルに入力したとすると、Out.ps というファイルに格納されている PostScript のデータが PDF に変換されて、その結果が Out.pdf というファイルに出力されます。

1.3 ABC 譜

1.3.1 ABC 譜の構成

ABC 譜は、「ヘッダー」(header) と「本体」(body) と呼ばれる二つの部分から構成されます。

ABC 譜の本体というのは、楽曲そのもの、つまりそれを構成する音 (tone) や休止 (rest) などを記述する部分です。それに対して、ヘッダーというのは、拍子 (meter) や調 (key) のような、楽曲についての情報 (information) を記述する部分です。

ひとつのファイルに格納されるひとつの文書の中には、ひとつの ABC 譜だけではなくて、複数の ABC 譜を書くことができます。ただし、ひとつの文書の中に複数の ABC 譜を書く場合、それぞれの ABC 譜は、連続する 2 個以上の改行 (newline) で区切られていないといけません。

改行という文字が二つ連続すると、2 個目の改行は「空行」(blank line) と呼ばれる行になります。空行というのは、改行以外の文字を含んでいない行のことです。ですから、「連続する 2 個以上の改行で区切る」というのは、「1 個以上の空行で区切る」というのと同じ意味になります。

ABC では、空行は、ABC 譜と ABC 譜とのあいだの区切りだと解釈されますので、ひとつの ABC 譜の途中には空行を入れることができません。

1.3.2　ヘッダー

ABC 譜のヘッダーというのは、楽曲についての情報を記述する部分です。ABC 譜を書くときは、その先頭に、かならずヘッダーを書く必要があります。

楽曲についての情報は、拍子、調、テンポ、題名など、さまざまな項目に分類できます。この実習マニュアルでは、それらの分類項目のことを「属性」(attribute) と呼ぶことにします。

属性に対して情報を設定したいときは、「フィールド」(field) と呼ばれる記述を書きます。フィールドというのは、

属性名 : フィールド値

という構文を持つ記述のことです（コロン (colon, `:`) とフィールド値とのあいだには、空白を挿入してもかまいません）。

属性名 (attribute identifier) というのは、それぞれの属性に名前として与えられた 1 文字の英字のことです。たとえば、調 (key) という属性には、`K` という属性名が名前として与えられています。

ひとつのフィールドは、属性名によって指定された属性に対して、フィールド値であらわされるものを情報として設定する、という意味を持っています。たとえば、

```
K: C
```

というフィールドは、楽曲の調という属性に対してハ長調という情報を設定する、という意味です。

なお、ヘッダーの中にフィールドを書く場合、フィールドとフィールドとのあいだや、フィールドと ABC 譜の本体とのあいだは、改行で区切る必要があります。

前の節で紹介した ABC 譜の例では、`K` のほかに、`X` と `M` と `L` という属性にも情報を設定していますので、これらの属性についても簡単に説明しておくことにしましょう（属性については、次の節でもう少し詳しく説明します）。

`X` は、参照番号 (reference number) という属性に与えられた属性名です。参照番号というのは、ABC 譜を扱うソフトがそれぞれの ABC 譜を識別するための番号のことです。通常は、ひとつの文書を構成するそれぞれの ABC 譜に、1 から始まる整数を参照番号として与えます。

`M` は、拍子 (meter) という属性に与えられた属性名です。`M` のフィールドに、フィールド値として `none` と書くと、「拍子記号は何も書かない」という意味になります。

`L` は、音の長さの単位 (unit note length) という属性に与えられた属性名です。たとえば、`L` のフィールドに、フィールド値として `1/1` という記述を書いたとすると、音の長さの単位として全音符の長さが設定されます。

ABC 譜のヘッダーの中には、最低限、参照番号 (`X`) と調 (`K`) という二つの属性に対して情報を設定するフィールドを書く必要があります。また、参照番号 (`X`) を設定するフィールドの位置は、ヘッダーの先頭でないといけない、と定められています。

1.3.3　音名

ABC 譜の本体は、音 (tone) をあらわす記述や休止 (rest) をあらわす記述などから構成されます。それぞれの音は、基本的には、

```
C D E F G A B c d e f g a b
```

という音名 (pitch name) によって記述されます。大文字は 1 点ハから 1 点ロまで、小文字は 2 点ハから 2 点ロまでの音をあらわします[1]。

ABC 譜の本体の中に音名を並べて書いて、その ABC 譜を楽譜に変換すると、音名が並んでいる順番と同じ順番で、音名に対応する音符 (note) が楽譜の上に並ぶことになります。

それぞれの音の長さは、明示的に記述しなければ、`L` のフィールドで設定されている単位の長さになります。音の長さを記述する方法については、第 2.2 節で説明する予定です。

1.3.4　縦線

縦線（じゅうせん）(bar line) は、縦棒 (vertical bar, `|`) という文字によってあらわされます。

同じように、複縦線 (double bar line) は二つの連続する縦棒 (`||`) によってあらわされ、終止線 (final line) は縦棒と右角括弧を並べたもの (`|]`) であらわされます。

[1] 音部記号として低音部記号を設定した場合は、2 オクターブだけ低くなります。

ABC 譜の例 barline.abc

```
X: 1
M: none
L: 1/1
K: C
C|D|E|F|G|A|B|c||c|B|A|G|F|E|D|C|]
```

1.3.5 空白

ABC 譜の本体の中では、空白 (space) は、結果として出力される楽譜に対して、あまり影響を及ぼしません。ですから、空白は、ABC 譜を人間にとって読みやすいものにするために利用することができます。

ABC 譜の例 space.abc

```
X: 1
M: none
L: 1/1
K: C
CDEFGFED CDEF GFED CD EF GF ED C D E F G F E D
```

1.3.6 改行

ABC 譜の本体に含まれるひとつの行 (line) は、基本的には、楽譜のひとつの行に変換されます。つまり、ABC 譜の本体の中では、改行 (newline) という文字は楽譜の改行を意味しているわけです。

ABC 譜の例 newline.abc

```
X: 1
M: none
L: 1/1
K: C
CDEFGABccBAGFEDC
cBAGFEDCCDEFGABc
```

場合によっては、楽譜のひとつの行をあらわす ABC の記述を複数の行に分割したい、ということもあります。そんなときは、改行の直前にバックスラッシュ(backslash, \) という文字を書きます。そうすると、バックスラッシュの直後の改行は、楽譜の改行という意味を失うことになります。

一般に、バックスラッシュを書くことによって、その直後にある文字の意味を変質させることを、文字を「エスケープする」(escape) と言います。

なお、バックスラッシュは、日本語の環境では円マーク (¥) で表示されることがあります。

ABC 譜の例 escape.abc

```
X: 1
M: none
L: 1/1
K: C
CDEFGABccBAGFEDC \
cBAGFEDCCDEFGABc
```

逆に、ABC 譜の行の途中で楽譜の改行を記述したいときは、その位置に感嘆符 (exclamation mark, !) を書きます。

1.3.7 インラインフィールド

属性のうちには、それに情報を設定するフィールドを、ヘッダーだけではなくて本体の中に書くことができるものもあります。たとえば、拍子 (M) や調 (K) や音の長さの単位 (L) などの属性は、本体の中にフィールドを書くことも可能です。本体の中に書かれたフィールドは、「インラインフィールド」(inline field) と呼ばれます。

インラインフィールドを書くためには、それを角括弧 (bracket, []) で囲む必要があります。たとえば、

```
[L:1/4]
```

という形の記述は、インラインフィールドになりますので、本体の中に書くことが可能です。ちなみに、このインラインフィールドは、「ここから先は音の長さの単位を 4 分音符の長さにする」という意味になります。

ABC 譜の例　inline.abc

```
X: 1
M: none
L: 1/1
K: C
CDEFGABccBAGFEDC[L:1/4]CDEFGABccBAGFEDC
```

1.4　注釈と注解

1.4.1　注釈

ABC 譜は、ソフトによって処理される単なるデータではなくて、人間が読んだり書いたりする文書の一種です。ABC 譜を書く人間は、その ABC 譜が、それを読む人にとって理解しやすいものになるように注意を払う必要があります。

人間にとって理解しやすい ABC 譜を書く上で効果のある技法のひとつは、人間による理解を助けるための記述を ABC 譜の中に書いておくことです。そのような記述は、「注釈」(comment) と呼ばれます。

注釈は、ABC 譜を処理するソフトが、「この部分は注釈だ」ということを認識できるように書かないといけません。なぜなら、ABC 譜を処理するソフトが、自分にとって意味のある記述として注釈を解釈しようとした場合、エラーが発生したり、正しくない結果が得られたりするからです。

ABC の文法では、パーセント (percent, `%`) という文字から最初の改行 (newline) までの文字列は注釈とみなされる、と定められていますので、注釈は、この規則にしたがって書く必要があります。

ABC 譜の例　comment.abc

```
%     example of comments
X: 1     % reference number
M: none  % meter
L: 1/1   % unit of length
K: C     % key
CDEFGABccBAGFEDC % scale of C major
```

1.4.2　注解とは何か

ABC 譜の場合と同じように、楽譜についても、その上に注釈を表示することによって、それを読む人間の理解を助けることができます。ABC は、ABC 譜の中に注釈を書くことができるだけではなくて、楽譜の上に注釈を表示するという機能も持っています。

このことは、ABC 譜の中には 2 種類の注釈があるということを意味しています。ひとつは ABC 譜の中だけに留まる注釈で、もうひとつは楽譜の上に表示される注釈です。それらを両方とも「注釈」(comment) と呼ぶのは紛らわしいので、楽譜の上に表示される注釈のほうは「注解」(annotation) と呼ぶことにしたいと思います。

1.4.3　位置指定文字

注解を書くためには、それを付ける対象となる音符や縦線などに対して、相対的にどのような位置にその注解を表示するのか、ということを指定する必要があります。

ABC では、1 個の文字を書くことによって、注解を表示する位置を指定します。そのような文字のことを、ここでは「位置指定文字」と呼ぶことにしたいと思います。

位置指定文字として書くことができるのは次の五つの文字のうちのいずれかです。

サーカムフレックス (circumflex, `^`)	上
アンダースコア (underscore, `_`)	下
小なり (less than, `<`)	左
大なり (greater than, `>`)	右

アットマーク (at mark, @) プログラムに依存

それぞれの文字は、対象となる音符や縦線などの上、下、左、右などを指定します。アットマークが指定する位置は、ABC 譜を処理するプログラムに依存します。

1.4.4 注解の書き方

ABC では、注解を記述するために、それを二重引用符 (double quotation mark, ") で囲んだものを書きます。ただし、左側の二重引用符の直後には、位置指定文字のうちのいずれかを書く必要があります。つまり、

" 位置指定文字 注解 "

という形のものを書くことになります。この形のものを、注解を付ける対象となる音符や縦線などの記述の左側に書くと、その中の注解が、その左側に書かれた位置指定文字によって指定された位置に表示されます。

ABC 譜の例 annota.abc

```
X: 1
M: none
L: 1/1
K: C
CDE"^above"FGAB"_below"ccBA"<left"GFED">right"C
```

1.5 基本的な属性

1.5.1 音の長さの単位

L という名前を持つ属性には、(unit note length) という情報を設定します。
音の長さの単位をあらわす記述は、

1/ 分母

という分数の形で書きます。たとえば、1/8 と書けば 8 分音符の長さ、1/16 と書けば 16 分音符の長さが単位になります。

ABC 譜の例 unit.abc

```
X: 1
M: none
L: 1/4
K: C
CDEFGABccBAGFEDC
```

1.5.2 拍子

M という名前を持つ属性には、拍子 (meter) という情報を設定します。
拍子をあらわす記述は、

分子 / 分母

という分数の形で書きます。つまり、2/4、2/2、3/4、3/2、4/4 というように書くわけです。
M のフィールドで設定された拍子は、拍子記号 (time signature) として楽譜の上に表示されます。

ABC 譜の例 meter.abc

```
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|]
```

4 分の 4 拍子は、C という記述で書きあらわすこともできます。その場合、楽譜の上には C という拍子記号が表示されます。
同じように、2 分の 2 拍子は C| と書くこともできます。その場合、𝔠 という拍子記号が楽譜の上に表示されることになります。

ABC 譜の例　meter2.abc

```
X: 1
M: C|
L: 1/2
K: C
CD|EF|GA|Bc|cB|AG|FE|DC|]
```

1.5.3　調

K という名前を持つ属性には、調 (key) という情報を設定します。

調は、

トニック　長短

という形の記述であらわされます。トニックの音名は英字の大文字を使います。トニックが派生音の場合は、音名の右側に、シャープ (sharp) ならば井桁 (number sign, #)、フラット (flat) ならば英字の小文字の b を書きます。

調の長短は、major または minor という単語であらわされます。長調の場合は、major を省略してもかまいません。

ABC 譜を楽譜に変換すると、K のフィールドで設定された調に応じた調号 (key signature) が楽譜の上に表示されます。

ABC 譜の例　key.abc

```
X: 1
M: none
L: 1/1
K: G# minor
GABcdefggfedcBAG
```

転調 (modulation) は、第 1.3 節で説明したインラインフィールドを使うことによって記述することができます。

ABC 譜の例　modula.abc

```
X: 1
M: none
L: 1/1
K: C
CDEFGABccBAGFEDC||[K:G]GABcdefggfedcBAG
```

1.5.4　音部記号

K のフィールドには、調の記述だけではなくて、音部記号 (clef) を設定する記述を書くこともできます。

音部記号は、

高音部記号（ト音記号、treble clef）	treble
低音部記号（ヘ音記号、bass clef）	bass
アルト記号 (alto clef)	alto
テノール記号 (tenor clef)	tenor

という単語であらわされます。これらの単語は、調の記述の右側に、1 個以上の空白を入れてから書きます。

なお、音部記号として低音部記号を設定した場合、ABC 譜の本体に書かれた音名は、高音部記号を設定した場合よりも 2 オクターブだけ低い音、つまり大文字が平仮名ハから平仮名ロまで、小文字が片仮名ハから片仮名ロまでをあらわすことになります。アルト記号またはテナー記号を設定した場合の音名の意味は、高音部記号を設定した場合と同じです。

ABC 譜の例　clef.abc

```
X: 1
M: none
L: 1/1
K: C bass
```

```
CDEFGABccBAGFEDC
```

音部記号のデフォルトは高音部記号です。しかし、音部記号の設定を省略した場合、中央のC(middle C) よりも低い音を ABC 譜の中に書くと、高音部記号と低音部記号の両方が楽譜の上に表示されてしまいます。そうならないようにするためには、高音部記号の記述を明示的に書く必要があります。

ちなみに、音部記号をあらわす単語を書く場所に none という単語を書くことによって、どの音部記号も表示されていない楽譜を作ることもできます。

1.5.5 テンポ

Q という名前を持つ属性には、楽曲を演奏するテンポ (tempo) という情報を設定します。

テンポを、1 分間に何個の拍 (beat) を演奏するかという数値で設定したい場合は、

1/ 分母 = 拍数

という形の記述を書きます。イコール (equal, =) の左側は、1 拍の長さをあらわす分数で、右側は、1 分間に演奏される拍の個数です。たとえば、

```
1/8=140
```

という記述は、8 分音符を 1 分間に 140 個だけ演奏するテンポをあらわしています。

テンポが拍の個数で設定されている場合、楽譜の上には、そのテンポをあらわすメトロノーム記号 (metronome mark) が表示されます。

ABC 譜の例 tempo.abc

```
X: 1
M: none
L: 1/2
Q: 1/2=60
K: C
CDEFGABccBAGFEDC
```

楽譜の上に速度標語 (tempo mark) や発想記号 (expression mark) を表示したいときは、Q のフィールドに、その速度標語や発想記号を二重引用符 (double quotation mark, ") で囲んだものを書きます。速度標語または発想記号と拍数の両方を Q のフィールドに書くこともできます。

ABC 譜の例 allegro.abc

```
X: 1
M: none
L: 1/4
Q: "Allegro" 1/4=120
K: C
CDEFGABccBAGFEDC
```

1.5.6 題名

T という名前を持つ属性には、楽曲の題名 (title) という情報を設定します。

T のフィールドにフィールド値として書かれた文字列は、そのまま題名として設定されます。その題名は、楽譜の冒頭にセンタリングされて表示されます。

ABC 譜の例 title.abc

```
X: 1
T: Scale of C major
M: none
L: 1/1
K: C
CDEFGABccBAGFEDC
```

T のフィールドを 2 個以上書くと、2 個目以降のタイトルはサブタイトルとみなされて、1 個目のタイトルよりも小さな文字で楽譜に表示されます。

ABC 譜の例 subtitle.abc

```
X: 1
```

```
T: Main Title
T: subtitle
M: none
L: 1/1
K: C
CDEFGABccBAGFEDC
```

1.5.7　作曲者名

C という名前を持つ属性には、楽曲の作曲者 (composer) の名前という情報を設定します。

C のフィールドも、T のフィールドと同じように、フィールド値として書かれた文字列が、そのまま作曲者名として設定されます。

作曲者名は、楽譜の右上に右寄せされて表示されます。C のフィールドを 2 個以上書いた場合は、それらのフィールドの順番と同じ順番で、縦の方向に作曲者名が並びます。

ABC 譜の例　composer.abc

```
X: 1
C: Tanioka Hitomi
C: Konishi Sadahiro
M: none
L: 1/1
K: C
CDEFGABccBAGFEDC
```

1.6　歌詞

1.6.1　歌詞の基礎

それぞれの音に対応する歌詞が譜表の下に表示された楽譜を、ABC を使って作りたいときは、小文字の w という名前を持つ属性に、その歌詞を設定します。この属性に歌詞を設定するフィールドは、ヘッダーに書くのではなくて、歌詞に対応している音を含む譜表を記述した行のすぐ下に書きます。

歌詞は、基本的には、空白で区切られたそれぞれの部分が個々の音に対応するとみなされます。たとえば、

```
w: tsu ki a ka ri
```

というフィールドで、w に歌詞を設定したとすると、1 個目の音に tsu、2 個目の音に ki、3 個目の音に a、というように音と歌詞とが対応するとみなされます。

ABC 譜の例　lyrics.abc

```
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|]
w: wa ta shi ha se ka i de i chi ba n no n ki da
```

1.6.2　歌詞を記述するための特殊文字

歌詞の記述の中では、次の文字は特殊な意味で使われます。

文字	説明
マイナス	ひとつの単語の内部をマイナス (minus, -) で区切ると、それぞれの部分がひとつの音に割り当てられます。
チルダ	複数の単語をチルダ (tilde, ~) で連結すると、その全体がひとつの音に割り当てられます。
アンダースコア	(underscore, _) は、直前の単語の末尾にある音節を延長して、ひとつの音符に割り当てます。アンダースコアを連続して書くことによって、音節をいくらでも延長することができます。
バックスラッシュ	改行の直前にバックスラッシュ(backslash, \) を書くことによって、1 行の

	譜表に対応する歌詞を 2 個以上のフィールドに分割することができます。また、バックスラッシュハイフン (\-) はハイフンそのものを意味しますので、それを使うことによって、ハイフンを含む単語をひとつの音に割り当てることができます。
縦棒	縦棒 (vertical bar, `|`) は、次の歌詞の先頭を次の小節の先頭に割り当てます。
アスタリスク	アスタリスク (asterisk, `*`) は、対応する位置にあるひとつの音に歌詞を何も割り当てません。

ABC 譜の例　`lyrics2.abc`

```
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|
w: wa-ta-shi ha se-ka-i de i-chi-ba-n no-n-ki da
[L:1/1]C|E|G|c|c|G|E|C|
w: watashi~ha sekai~de ichiban nonki~da \
w: nonkina ningen~ga sekai~niha hitsuyou~da
[L:1/4]CDEF|GABc|cBAG|FEDC|
w: wa-ta-shi ha _ _ _ _ se-ka-i de _ _ _ _
CDEF|GABc|cBAG|FEDC|
w: i-chi-ba-n | | no-n-ki da | |
CDEF|GABc|cBAG|FEDC|
w: wa * ta * shi * ha * no * n * ki * da *
[L:1/1]C|E|G|c|c|G|E|C|]
w: watashi\-ha sekai\-de ichiban nonki\-da \
w: nonkina ningen\-ga sekai\-niha hitsuyou\-da
```

1.6.3　楽譜の末尾に表示される歌詞

歌詞を、それに対応する音を含む譜表の下に表示するのではなくて、楽譜の末尾にまとめて表示したい、というときは、小文字の `w` ではなくて、大文字の `W` という名前の属性に歌詞を設定します。この属性に歌詞を設定するフィールドは、ABC 譜の中のどこに書いてもかまいません。

ABC 譜の例　`endlyri.abc`

```
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|]
W: I am the most easygoing person in the world.
W: An easygoing person is needed for the world.
```

第 2 章　音符

2.1　音の高さ

2.1.1　オクターブ

第 1 章ですでに説明したように、ABC 譜では、

```
C D E F G A B c d e f g a b
```

という音名 (pitch name) を書くことによって、個々の音を記述します。音部記号として低音部記号以外のものが設定されている場合、大文字は 1 点ハから 1 点ロまで、小文字は 2 点ハから 2 点ロまでの音をあらわします。音部記号として低音部記号が設定されている場合は、大文字が平仮名ハから平仮名ロまで、小文字が片仮名ハから片仮名ロまでになります。

小文字よりも高いオクターブ (octave) の音を記述したいときは、小文字の右側にアポストロフィー (apostrophe, `’`) の列を書きます。アポストロフィーの個数が多いほど、オクターブが高くなります。たとえば、音部記号として高音部記号が設定されているとすると、`c’’’` という記述は、5 点ハの音をあらわすことになります。

大文字よりも低いオクターブの音を記述したいときは、大文字の右側にコンマ (comma, `,`) の列を書きます。コンマの個数が多いほど、オクターブが低くなります。たとえば、音部記号として高音部記号が設定されているとすると、`C,,,` という記述は、下 1 点ハの音をあらわすことになります。

ABC 譜の例　`octave.abc`

```
X: 1
M: none
L: 1/1
K: C treble
c'd'e'f'g'a'b'c''d''e''f''g''a''b'' \
c'''d'''e'''f'''g'''a'''b''' \
c''''d''''e''''f''''g''''a''''b''''c'''''
CB,A,G,F,E,D,C,B,,A,,G,,F,,E,,D,,C,, \
B,,,A,,,G,,,F,,,E,,,D,,,C,,, \
B,,,,A,,,,G,,,,F,,,,E,,,,D,,,,C,,,,
```

2.1.2　派生音

音名によってあらわされる音の高さは、基本的には、`K` のフィールドで設定されている調にもとづいて、幹音 (natural tone) になるか派生音 (derived tone) になるかということが決まります。たとえば、調としてト長調が設定されているとするならば、`C` や `c` はハという幹音になって、`F` や `f` は嬰へという派生音になります。

設定されている調では幹音になっている音を嬰音 (sharp) にしたいときは、音名の左側にサーカムフレックス (circumflex, `^`) を書きます。たとえば、`^C` や `^c` は、調にかかわらず嬰ハという派生音をあらわすことになります。

同じように、幹音を変音 (flat) にしたいときは、音名の左側にアンダースコア (underscore, `_`) を書きます。たとえば、`_E` や `_e` は、調にかかわらず変ホという派生音をあらわすことになります。

調の設定にもとづいて派生音になっている音を幹音に戻したいときは、音名の左側にイコール (equal, `=`) を書きます。たとえば、ト長調の `F` は、`=F` と書くことによって幹音に戻りますし、ヘ長調の `B` は、`=B` と書くことによって幹音に戻ります。

ABC 譜を楽譜に変換したとき、サーカムフレックスを付けて記述された音をあらわす音符の左側には、嬰記号 (sharp) が臨時記号 (accidental) として表示されます。同じように、アンダースコアを付けて記述された音をあらわす音符の左側には変記号 (flat) が表示され、イコールを付けて記述された音をあらわす音符の左側には本位記号 (natural) が表示されます。

ABC 譜の例　`derived.abc`

```
X: 1
M: none
L: 1/1
K: C
GABcde^fgg^fedcBAG
FGA_Bcdeffedc_BAGF

X: 2
M: none
L: 1/1
K: G
CDE=FGABccBAG=FEDC
```

なお、重嬰音 (double sharp) は、2 個のサーカムフレックスを並べたもの (`^^`) を音名の左側に書くことによってあらわされます。同じように、重変音 (double flat) は、2 個のアンダースコアを並べたもの (`__`) を音名の左側に書くことによってあらわされます。

2.2　音の長さ

2.2.1　音の長さの基礎

音の長さを指定したいときは、音の高さをあらわす記述（音名と、その右側のアポストロフィーまたはコンマの列）の右側に、音の長さをあらわす記述を書きます。

音の長さは、Lのフィールドで設定されている音の長さの何倍なのか、ということを示す整数または分数によってあらわされます。

たとえば、音の長さの単位として、1/8、つまり 8 分音符の長さが設定されているとすると、C2という記述は 4 分音符の長さの 1 点ハをあらわし、A,1/2という記述は 16 分音符の長さの平仮名イをあらわすことになります。

付点音符 (dotted note) の長さは、単位の 3^n 倍、または単位の $3/2^n$ 倍を示す整数または分数を書くことによってあらわされます。

ABC 譜の例 length.abc

```
X: 1
M: none
L: 1/4
K: C none
G4 G2 G1 G1/2 G1/4 G1/8 G1/16 \
c4 c2 c1 c1/2 c1/4 c1/8 c1/16 \
G3 G3/2 G3/4 G3/8 G3/16 G3/32 \
c3 c3/2 c3/4 c3/8 c3/16 c3/32
```

音の長さを省略した場合は、単位の 1 倍だと解釈されます。ですから、C1とCとは同じ意味になります。

また、音の長さをあらわす分数の分母と分子のそれぞれを省略することもできます。省略された分子は 1 と解釈され、省略された分母は 2 と解釈されます。ですから、C1/4とC/4とは同じ意味で、C3/2とC3/も同じ意味、C1/2とC/も同じ意味です。

なお、C/4は、C//と書いても同じ意味になります。

2.2.2 連桁

楽譜を書くとき、8 分音符またはそれよりも短い音符が何個か連続する場合は、連桁（れんこう）(balken) と呼ばれる横棒を使ってそれらを連結することによって、何拍かの長さを持つグループを作ると、楽譜が読みやすくなります。

abcm2ps は、8 分音符またはそれよりも短い音符が何個か連続する場合、それらの音符を自動的に連桁で連結します。ただし、音の記述と音の記述とのあいだに空白が書かれている場合、それらの音符は連結されません。ですから、空白を挿入することによって、連桁を使って音符をグループにする拍数を自由に指定することができます。

ABC 譜の例 balken.abc

```
X: 1
M: none
K: C
[L:1/8]  CDEFGABccdefgabc'| \
[L:1/16] CDEFGABccdefgabc'c'bagfedccBAGFEDC|
[L:1/8]  CDEFGABc cdefgabc'| \
[L:1/16] CDEFGABccdefgabc' c'bagfedccBAGFEDC|
[L:1/8]  CDEF GABc cdef gabc'| \
[L:1/16] CDEFGABc cdefgabc' c'bagfedc cBAGFEDC|
```

2.2.3 休止

音の休止 (rest) は、小文字のzという文字によってあらわされます。休止の長さは、zの右側に書かれた長さに記述によって指定されます。休止の長さの書き方は、音の長さの場合とまったく同じです。

2 小節以上の長い休止を記述する方法は、二つあります。ひとつは、全休符の長さのzを小節の個数だけ書くという方法で、もうひとつは、大文字のZを書いて、その右側に小節の個数を書く、という方法です。長い休止を後者の方法で記述したものを楽譜に変換すると、小節の個数をあらわす数字を伴った休符が表示されることになります。

ABC 譜の例 rest.abc

```
X: 1
M: none
L: 1/4
K: C none
z4 z2 z1 z1/2 z1/4 z1/8 z1/16 \
```

```
z3 z3/2 z3/4 z3/8 z3/16 z3/32
z4|z4|z4|z4|z4|z4|z4|z4|Z8|
```

2.2.4　タイ

ひとつの音を、2 個以上の音を連結したものとして記述したいときは、それぞれの音の記述をマイナス (minus, -) という文字で連結します。たとえば、

```
C2-C1-C1/2
```

という記述は、C2 の音と C1 の音と C1/2 の音を連結することによってできるひとつの音をあらわします。

マイナスによって連結された 2 個以上の音の記述を楽譜に変換すると、タイ (tie) によって連結された音符が表示されることになります。

ABC 譜の例　tie.abc

```
X: 1
M: none
L: 1/4
K: C none
E2-E2 E2-E1 E1-E1 E1-E1/2 E2-E1-E1/2 \
f2-f2 f2-f1 f1-f1 f1-f1/2 f2-f1-f1/2
```

2.2.5　連音符

指定された個数の拍を指定された個数に等しく分割して、それぞれの長さで音または休止を演奏すること、つまり、楽譜の上に連符 (tuplet) として表示されるようなものを ABC で記述したいときは、

(*p*:*q*:*r* 音の記述 ･･･

という形のものを書きます。この中の *p*、*q*、*r* のそれぞれの場所には、ひとつの整数を書きます。*p* は分割の個数、*q* は連符全体の拍数、そして *r* は音または休止の個数です。楽譜の上に表示される音符の形は、「音の記述」の中に記述された音の長さによって決定されます。

たとえば、拍子として 4 分の 4 拍子が設定されているとすると、

```
(3:2:3E1F1G1
```

という記述は、2 拍を 3 分割して、それぞれの長さで E と F と G を演奏するという意味になります。音の長さの単位として 1/4 が設定されているとすると、それぞれの音は楽譜の上に 4 分音符で表示されます。

ABC 譜の例　tuplet.abc

```
X: 1
M: 4/4
L: 1/4
K: C
(3:4:3E2F2G2 | (5:4:5g1f1e1d1c1 | \
E E (3:2:3E1F1G1 | g g (5:2:5g1/2f1/2e1/2d1/2c1/2 | \
E E E (3:1:3E1/2F1/2G1/2 | g g g (5:1:5g1/4f1/4e1/4d1/4c1/4 ||
[M:3/4] (2:3:2E2F2 | (4:3:4f1e1d1c1 | (5:3:5g1f1e1d1c1 || \
[M:3/8] (2:3:2E1F1 | (4:1:4f1/2e1/2d1/2c1/2 | \
(5:1:5g1/2f1/2e1/2d1/2c1/2 ||
[M:4/4] [L:1/8] \
(3:1:3d1z1d1 (3:1:3z1d1d1 (3:1:3d1z1z1 (3:1:3z1z1d1 | \
(3:1:2d2d1 (3:1:2d1d2 (3:1:4d1d1/2d1/2d1 (3:1:4d1d1/2z1/2d1 ||
```

分割の個数と音または休止の個数とが等しい場合、つまり *p* と *r* とが等しい場合は、:*r* を省略することができます。たとえば、(3:2:3 は、(3:2 と書いても同じ意味になります。

2.2.6　装飾音

装飾音 (grace) を記述したいときは、中括弧 (brace, {}) を使います。音の記述を中括弧で囲んだものを音の記述の左側に書くと、中括弧の中の音は、その右側の音を装飾する装飾音だと解釈されます。装飾音は、装飾音符 (grace note) として楽譜の上に表示されます。

長前打音 (long appoggiatura) ではなくて短前打音 (short appoggiatura) だということを示す斜線を装飾音符に入れたいときは、左中括弧の直後にスラッシュ(slash, `/`) を書きます。

ABC 譜の例 `grace.abc`

```
X: 1
M: none
L: 1/8
K: C
{E2}F4 {f2}e4 {E1}F2 {f1}e2 {E/}F1 {f/}e1 \
{F1E1}F2 {f1g1}e2 {A/G/F/E/}F2 {e/f/g/a/}e2 \
{/E2}F4 {/f1}e2 {/F1E1}F2 {/e/f/g/a/}e2
```

2.3 和音

2.3.1 和音の記述

ABC では、和音 (chord) は、角括弧 (bracket, `[]`) を使うことによって記述することができます。一組の角括弧の中にいくつかの音の記述を書くと、それらの音はひとつの和音を構成するという意味になります。たとえば、

```
[CEG]
```

という記述は、ハの音を根音 (root) とする長三和音 (major triad) をあらわしています。

和音を構成するそれぞれの音の記述は、長さをあらわす整数を含んでいてもかまいません。たとえば、音の長さの単位として 8 分音符の長さが設定されているとすると、

```
[D2F2A2]
```

という記述は、4 分音符の長さを持つ、ニの音を根音とする短三和音 (minor triad)、という意味になります。

同じ高さの音から構成される二つの和音を連結してひとつの音にしたいときは、それらの和音をあらわす記述のあいだにマイナス (minus, `-`) を書きます。楽譜に変換したとき、それらの和音の音符は、同じ高さの音同士がタイで連結されることになります。

ABC 譜の例 `chord.abc`

```
X: 1
M: none
K: C
[L:1/1]  [CEG][C_EG][C_E_G][CE^G] \
[L:1/4]  [A^ce][Ace][Ac_e][A^c^e] \
[L:1/8]  [CEG][DFA][EGB][FAc] [D6F6A6]-[DFA]
```

2.3.2 コードネーム

演奏するべき和音をコードネーム (chord name) で記述したいときは、コードネームを二重引用符 (double quotation mark, `"`) で囲んだものを書きます。

コードネームの先頭には、かならず、その根音をあらわす記述を書く必要があります。根音の音名は英字の大文字で書きます。そして、根音が嬰音ならば音名の右側に井桁 (number sign, `#`) を、変音ならば英字の小文字の `b` を書きます。

根音の記述だけから構成されるコードネームは、その音を根音とする長三和音をあらわすことになります。たとえば、`C` というコードネームはハの音を根音とする長三和音をあらわしていて、`Eb` というコードネームは変ホの音を根音とする長三和音をあらわしています。

長三和音以外の和音を記述するためには、根音の記述の右側に、表 2.1 に示されているような、和音の種類をあらわす記述を書く必要があります。たとえば、`Em` というコードネームはホの音を根音とする短三和音をあらわしていて、`G7` というコードネームはトの音を根音とする属七の和音 (dominant seventh chord) をあらわしています。

和音の種類をあらわす記述は、組み合わせて使うことも可能です。たとえば、`Gm7` はトを根音とする短七の和音 (minor seventh chord)、`Gmaj7` はトを根音とする長七の和音 (major seventh chord)、`Gdim7` はトを根音とする減七の和音 (diminished seventh chord) をあらわしています。

ベースノートが指定された和音を記述したいときは、和音の種類をあらわす記述の右側にスラッシュ(slash, `/`) を書いて、そのさらに右側にベースノートをあらわす記述を書きます。ベー

記述	説明
maj	長音程 (major interval)
m または min	短音程 (minor interval)
aug または +	増音程 (augmented interval)
dim	減音程 (diminished interval)
7, 9, etc.	第 7 音 (seventh)、第 9 音 (ninth)、その他
sus	掛留音 (suspension)

表 2.1: 和音の種類をあらわす記述

スノートの書き方は、根音の書き方と同じです。たとえば、Em/G というコードネームは、ベースノートとしてトの音が指定された、ホの音を根音とする短三和音をあらわしています。

コードネームによって和音が記述されている ABC 譜を楽譜に変換すると、譜表の上の然るべき位置に、そのコードネームが表示されます。

ABC 譜の例　choname.abc

```
X: 1
M: 4/4
L: 1/4
K: C
"C"CDEF|"C#"GABc|"Cb"cBAG|"Cm"FEDC| \
"Caug"CDEF|"Cdim"GABc|"C7"cBAG|"C9"FEDC|
"Csus"CDEF|"Cm7"GABc|"Cmaj7"cBAG|"Cm7"FEDC| \
"Cdim7"CDEF|"C/E"GABc|"Cm/Eb"cBAG|"C#maj7/G#"FEDC|]
```

第 3 章　声部

3.1　声部の基礎

3.1.1　声部の情報を設定する属性

この章では、複数の声部 (voice) を持つ楽曲を ABC で記述する方法について説明したいと思います。

複数の声部を持つ楽曲を ABC で記述するためには、それぞれの声部を識別するために、それぞれの声部に異なる名前を与える必要があります。声部の名前は、英字または数字を並べることによって作ります。たとえば、

```
namako    UMIUSHI    uni37kani    40381
```

などは、声部の名前として使うことのできる文字列です。

個々の声部を記述するためには、そのための準備として、V という名前を持つ属性に対して、その声部に固有の情報を設定する必要があります。この属性に設定することのできる情報は何種類もあるのですが、最低限必要なのは、その声部の名前だけです。ですから、

```
V: namako
```

というフィールドを書くことによって、namako という声部名を持つ声部を記述するための準備ができたことになります。

V のフィールドを書いて、その下に音や休止の記述を書くと、それらの音や休止は、V に設定されている声部名を持つ声部に所属するとみなされます。ですから、異なる声部名を設定する複数の V フィールドを書いて、それぞれの V フィールドの下に音や休止の記述を書くことによって、複数の声部を持つ楽曲を記述することができます。

次の ABC 譜は、三つの声部を持つ楽曲を、それぞれの声部に vocal、piano、organ という名前を与えて記述したものです。

ABC 譜の例　voices.abc

```
X: 1
M: 4/4
```

```
L: 1/4
K: C
V: vocal
CDEF|GABc|cBAG|FEDC|
CDEF|GFED|CDEF|GFED|]
V: piano
EFGA|Bcde|edcB|AGFE|
EFGA|BAGF|EFGA|BAGF|]
V: organ
GABc|defg|gfed|cBAG|
GABc|cBAG|GABc|cBAG|]
```

3.1.2 インラインフィールドによる声部の設定

V という属性に対して声部に固有の情報を設定するフィールドは、インラインフィールドとして書くことも可能です。

複数の声部を持つ楽曲を ABC で記述するとき、それぞれの声部をひとつの場所にまとめて記述する場合は普通の形式のフィールドを書けばいいわけですが、楽譜のひとつの行に相当する部分をひとつの場所にまとめて記述する場合は、インラインフィールドを使うほうがいいでしょう。

次の ABC 譜は、三つの声部を持つ楽曲を、楽譜の行に相当する部分をひとつにまとめた形で記述したものです。

ABC 譜の例 voices2.abc

```
X: 1
M: 4/4
L: 1/4
K: C
[V:vocal] CDEF|GFED|CDEF|GFED|
[V:piano] EFGA|BAGF|EFGA|BAGF|
[V:organ] GABc|dcBA|GABc|dcBA|
[V:vocal] CDEF|GFED|CDEF|GFED|C4|]
[V:piano] EFGA|BAGF|EFGA|BAGF|E4|]
[V:organ] GABc|dcBA|GABc|dcBA|G4|]
```

3.2 声部の情報

3.2.1 声部の情報を設定するフィールドの書き方

V という属性には、声部に固有のさまざまな情報を設定することができます。この節では、この属性に設定することのできる声部の情報について説明したいと思います。

V という属性に情報を設定するフィールドは、

V: 声部名 情報の記述 ・・・

と書きます。このように、声部の名前はかならずフィールド値の先頭に書かないといけません。

3.2.2 声部の音部記号

V 属性には、譜表に表示する音部記号の情報を設定することができます。V のフィールドの中で音部記号を記述する方法は、K のフィールドの中でそれを記述する場合と同じです。たとえば、高音部記号は treble、低音部記号は bass と書きます。また、none と書くことによって、譜表に音部記号を表示しないようにすることもできます。

なお、V 属性に対して低音部記号を設定すると、その声部では、音名があらわす音の高さが 2 オクターブだけ低くなります。

ABC 譜の例 clefvoi.abc

```
X: 1
M: 4/4
L: 1/4
K: C
V: flute  treble
V: faggot bass
[V:flute]  CDEF|GABc|cBAG|FEDC|]
```

```
[V:faggot] CDEF|GABc|cBAG|FEDC|]
```

3.2.3　声部の名前の表示

譜表の左側に声部の名前を表示したいときは、V のフィールドの中に、

name="声部名"

という形の記述を書きます。この記述の中に書く声部の名前は、V 属性に対して声部の名前として設定したものとは異なっていてもかまいません。

この形の記述によって設定された名前は、楽譜の最初の行だけにしか表示されません。楽譜の 2 行目以降にも声部の名前を表示したいときは、

sname="声部名"

という形の記述を書きます。この記述によって設定された名前は、最初の行を除いたすべての行に表示されます。

なお、改行をあらわす \n という文字列を声部の名前の中に入れることによって、複数の行から構成される名前を設定することも可能です。

ABC 譜の例　namevoi.abc

```
X: 1
M: 4/4
L: 1/4
K: C
V: vocal    name="vocal"             sname="vo"
V: clarinet name="clarinet\ntrumpet" sname="cl\ntp"
[V:vocal]    CDEF  |GABc|cBAG|FEDC|
[V:clarinet] A,B,CD|EFGA|AGFE|DCB,A,|
[V:vocal]    CDEF  |GFED |CDEF  |GFED |C4 |]
[V:clarinet] A,B,CD|EDCB,|A,B,CD|EDCB,|A,4|]
```

3.2.4　音符の符尾の方向

音符の符尾 (stem) を上向きにするか下向きにするかというのは、普通、その音符が五線の第 3 線よりも上にあるか下にあるかということによって自動的に決定されます。しかし、場合によっては、ひとつの声部を構成するすべての音符について、符尾の方向を上または下に統一する必要が生じることもあります。

符尾の方向を上向きに統一したいときは、V のフィールドの中に up という単語を書きます。同じように、下向きに統一したいときは down という単語を書きます。

ABC 譜の例　stemvoi.abc

```
X: 1
M: 4/4
L: 1/4
K: C
V: vocal name="vocal"
V: piano name="piano" up
V: organ name="organ" down
[V:vocal] CDEF|GABc|cdef|gabc'|]
[V:piano] CDEF|GABc|cdef|gabc'|]
[V:organ] CDEF|GABc|cdef|gabc'|]
```

3.3　譜表のグループ

3.3.1　擬似注釈

楽曲というのは、調や拍子やテンポなどのさまざまな属性を持っていて、フィールドというものを書くことによって、それぞれの属性に対して情報を設定することができます。

ABC 譜というのは楽譜を作ることが主要な目的ですので、ABC 譜では、楽曲そのものの属性だけではなくて、それを楽譜で記述する上でのフォーマット (format) に関する属性に対しても、情報を設定することが必要になります。そのような、フォーマットに関する属性のことを、

「フォーマット属性」(format attribute) と呼ぶことにしたいと思います。

楽曲そのものの属性に情報を設定したいときはフィールドを書けばいいわけですが、フォーマット属性に情報を設定したいときは、「擬似注釈」(pseudocomment) と呼ばれるものを ABC 譜の中に書く必要があります。擬似注釈というのは、

%% 属性名 文字列 改行

という形の文字列のことです。このように、擬似注釈はかならずパーセントパーセント (%%) で始まって、改行で終わります。

擬似注釈によって情報を設定することのできる属性には、それぞれを識別するための名前が与えられています。擬似注釈のパーセントパーセントの右側に属性の名前を書くことによって、その名前を持つ属性に対して、その右側に書かれた文字列によってあらわされる情報を設定することができます。

3.3.2 声部に関するフォーマット属性

フォーマット属性のひとつで、 staves という名前を持っているものがあります。この属性には、楽曲を構成するそれぞれの声部を楽譜の上にどのように表示するかという情報を設定することができます。

staves 属性に対して声部を設定する擬似注釈は、基本的には、

%%staves 声部名 ...

と書きます。たとえば、

```
%%staves vocal piano organ
```

という擬似注釈は、 staves 属性に対して、 vocal 、 piano 、 organ という三つの声部を表示する、ということをあらわしています。

staves 属性に声部名を設定する擬似注釈を書いた場合、楽譜には、その属性に設定されている声部だけが表示されます。つまり、たとえ ABC 譜の中に声部が記述されていたとしても、 staves 属性に声部名が設定されていなければ楽譜には表示されないということです。

また、 staves 属性に声部を設定した場合、楽譜の中で譜表を上下に並べる順番は、擬似注釈の中で声部名を並べた順番のとおりになります。たとえば、上の擬似注釈の例では、上から順番に、 vocal 、 piano 、 organ と譜表が並ぶことになります。

3.3.3 総譜

総譜 (score) と呼ばれる、多数の声部から構成される楽譜を作るときは、それを構成する声部を、木管楽器、金管楽器、弦楽器などのグループに分けて、それぞれのグループの譜表を角括弧 (bracket, []) で囲む、というのが普通です。

ABC 譜では、声部のグループを作りたいときは、 staves 属性の擬似注釈の中で、グループを構成する声部の名前を角括弧で囲みます。たとえば、

```
%%staves [trumpet trombone] [violin cello]
```

という擬似注釈を書いたとすると、 trumpet と trombone がひとつのグループになり、 violin と cell がもうひとつのグループになります。

ABC 譜の例 score.abc

```
X: 1
M: 4/4
L: 1/4
K: C
%%staves [flute oboe] violin
V: flute  name="flute"  sname="fl"
V: oboe   name="oboe"   sname="ob"
V: violin name="violin" sname="Vn"
[V:flute]  GABc|defg|gfed|cBAG|
[V:oboe]   EFGA|Bcde|edcB|AGFE|
[V:violin] CDEF|GABc|cBAG|FEDC|
[V:flute]  GABc|dcBA|GABc|dcBA|G4|]
[V:oboe]   EFGA|BAGF|EFGA|BAGF|E4|]
[V:violin] CDEF|GFED|CDEF|GFED|C4|]
```

3.3.4　大譜表

ピアノやオルガンやハープなどのような音域の広い楽器の声部を楽譜で表示するときは、「大譜表」(grand staff) と呼ばれる、複数の譜表を中括弧 (brace, {}) で囲んだものが使われます。

ABC 譜では、大譜表を作りたいときは、staves 属性の擬似注釈の中で、大譜表を構成する声部の名前を中括弧で囲みます。たとえば、

```
%%staves {right left}
```

という擬似注釈を書いたとすると、right と left という二つの譜表は、ひとつの大譜表を構成することになります。

なお、ひとつの大譜表にすることのできる譜表の個数は、最大 3 個までです。

ABC 譜の例　grand.abc

```
X: 1
M: 4/4
L: 1/4
K: C
%%staves {right left}
V: right treble
V: left  bass
[V:right] CDEF|GABc|cBAG|FEDC|
[V:left]  GABc|defg|gfed|cBAG|
[V:right] CDEF|GFED|CDEF|GFED|C4|]
[V:left]  GABc|dcBA|GABc|dcBA|G4|]
```

3.3.5　複数の声部を詰め込んだ譜表

複数の声部は、基本的にはそれぞれ異なる譜表によって表示されるわけですが、場合によっては、ひとつの譜表の中に複数の声部を詰め込みたい、ということがあります。

ABC 譜では、ひとつの譜表を使って複数の声部を表示したいときは、staves 属性の擬似注釈の中で、ひとつの譜表の中に詰め込みたい声部の名前を丸括弧 (parenthesis, ( )) で囲みます。たとえば、

```
%%staves [(soprano alto) (tenor bass)]
```

という擬似注釈を書いたとすると、soprano と alto はひとつの譜表に詰め込まれ、tenor と bass もひとつの譜表の中に詰め込まれることになります。

ABC 譜の例　compact.abc

```
X: 1
M: 4/4
L: 1/4
K: C
%%staves [(soprano alto) (tenor bass)]
V: soprano treble name="soprano" up
V: alto    treble name="alto"    down
V: tenor   bass   name="tenor"   up
V: bass    bass   name="bass"    down
[V:soprano] GABc|defg |gfed|cBAG|
[V:alto]    CDEF|GABc |cBAG|FEDC|
[V:tenor]   cdef|gabc'|c'bag|fedc|
[V:bass]    GABc|defg|gfed|cBAG|
[V:soprano] GABc|dcBA|GABc|dcBA|G4|]
[V:alto]    CDEF|GFED|CDEF|GFED|C4|]
[V:tenor]   cdef|gfed|cdef|gfed|c4|]
[V:bass]    GABc|dcBA|GABc|dcBA|G4|]
```

第 4 章　記号

4.1 強弱記号

4.1.1 この章について

楽譜というものは、さまざまな記号から構成されています。この実習マニュアルのこれまでの部分でも、音符、音部記号、拍子記号、メトロノーム記号など、すでにさまざまな記号が登場しましたが、楽譜で使われる記号は、ほかにもまだたくさんあります。

この章では、楽譜を作るために使われる記号のうちで、すでに登場した記号を除いた残りのものを紹介していきたいと思います。

4.1.2 記号指定

ABC では、楽譜の上に表示される記号のうちのいくつかについては、

! 記号名 !

という形の記述を書くことによって、その表示を指示することができます。この中に書く「記号名」(symbol name) というのは、それぞれの記号に与えられている、ABC での名前のことです。

たとえば、ABC では、特定の音を強く演奏することを指示する記号、すなわちアクセント記号 (accent mark) は、`accent` という記号名によって識別されますので、`!accent!` という記述を書くことによって、その記号の表示を指示することができます。

なお、これから先、このような形の記述のことは、「記号指定」(symbol description) と呼ぶことにしたいと思います。

記号指定を書く位置は、その記号が対象とするものの記述の直前です。たとえば、`!accent!` のような、音を対象とする記号の表示を指示する記号指定は、その音の記述の直前に書きます。

ABC 譜の例 `accent.abc`

```
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GAB!accent!c|cBAG|FED!accent!C|]
```

4.1.3 ピアノやフォルテなど

音の強さを指示する記号、すなわち強弱記号 (dynamic mark) のうちで、ピアノ (piano) やフォルテ (forte) などの仲間は、

`!ppp!`	ピアニッシッシモ (pianississimo)
`!pp!`	ピアニッシモ (pianissimo)
`!p!`	ピアノ (piano)
`!mf!`	メゾフォルテ (mezzo forte)
`!f!`	フォルテ (forte)
`!ff!`	フォルティッシモ (fortissimo)
`!fff!`	フォルティッシッシモ (fortississimo)
`!sfz!`	スフォルツァート (sforzato)

という記号指定を書くことによって、表示を指示することができます。

なお、これらの記号は、デフォルトでは譜表の下に表示されます。それらを譜表の上に表示したいときは、

```
%%exprabove true
```

という疑似注釈を書きます。

ABC 譜の例 `dynamic.abc`

```
X: 1
M: 4/4
L: 1/4
K: C
%%exprabove true
!ppp!CDEF|!pp!GABc|!p!cBAG|!mf!FEDC|
!f!CDEF|!ff!GABc|!fff!cBAG|!sfz!FEDC|]
```

4.1.4　クレッシェンドとディミヌエンド

音の強さを少しずつ変化させていくことを指示する、クレッシェンド (crescendo) とディミヌエンド (diminuendo) という記号は、

`!crescendo(!`　クレッシェンド（始まり）
`!crescendo)!`　クレッシェンド（終わり）
`!diminuendo(!`　ディミヌエンド（始まり）
`!diminuendo)!`　ディミヌエンド（終わり）

という記号指定を書くことによって、表示を指示することができます。

なお、これらの記号も、デフォルトでは譜表の下に表示されますので、譜表の上に表示したい場合は、そのための疑似注釈を書く必要があります。

ABC 譜の例　`credimi.abc`

```
X: 1
M: 4/4
L: 1/4
K: C
%%exprabove true
CDEF|!crescendo(!GABc!crescendo)!|cccc| \
!diminuendo(!cBAG!diminuendo)!|FEDC|]
```

4.2　反復記号

4.2.1　反復記号の基礎

楽譜では、「反復記号」(repeat mark) と呼ばれる記号を書くことによって、楽譜の中の指定された範囲を繰り返して演奏するということを記述することができます。

ABC 譜では、本体の中に、

`|:`　繰り返す範囲の始まり
`:|`　繰り返す範囲の終わり
`::`　繰り返す範囲の終わりと始まり

という記述を書くことによって、楽譜に反復記号を表示することができます。

ABC 譜の例　`repeat.abc`

```
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GFED|CDEF|GABc|cBAG|FEDC:|
CDEF|GFED|:CDEF|GABc|cBAG|FEDC:|
|:CDEF|GFED::CDEF|GABc|cBAG|FEDC:|
```

4.2.2　1 番括弧と 2 番括弧

何小節かの範囲を繰り返す場合に、その末尾の部分だけを違った演奏にしたい、ということがしばしばあります。そのような場合、楽譜では、「1 番括弧」(first ending) と「2 番括弧」(second ending) という記号が使われます。

ABC 譜では、縦線の直後に `[1` という記述を書くと、その位置から始まる 1 番括弧が楽譜に表示されます。同じように、縦線の直後に `[2` という記述を描くと、その位置から始まる 2 番括弧が楽譜に表示されます。

ABC 譜の例　`ending.abc`

```
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GFED|CDEF|GABc|cBAG|FEDC|
cBAG|FEDC|[1CDEF|GFED:|[2CEGc|cGEC|]
```

4.3 奏法記号

4.3.1 奏法記号の基礎

楽譜では、「奏法記号」(articulation symbol) と呼ばれる記号を書くことによって、奏法 (articulation) に関する指示を記述することができます。

ABC でも、楽譜の上にさまざまな奏法記号を表示するための指示を記述する方法が定められています。

4.3.2 スタッカート

音を短く切って演奏する、つまりスタッカートで演奏するという奏法をあらわす記号は、音の記述の左側にドット (dot, `.`) を書くことによって表示することができます。たとえば、`.C` という記述は、1 点ハの音符にスタッカートの記号を付け加えるという意味になります。

ABC 譜の例 `staccato.abc`

```
X: 1
M: 4/4
L: 1/4
K: C
.C.D.E.F|.G.F.Gz|.g.f.e.d|.c.d.cz||
```

4.3.3 スラー

音と音とのあいだに切れ目を入れないで、なめらかに演奏する、つまりレガート (regato) に演奏するという奏法は、楽譜では、スラー (slur) という記号によってあらわされます。

スラーを表示するという指示を ABC 譜の中で記述したいときは、その範囲の記述を丸括弧 (parenthesis, `()`) で囲みます。たとえば、

```
CDE(FGA)Bc
```

という記述は、音階のうちの `FGA` の部分にスラーを付けるという意味になります。

ABC 譜の例 `slur.abc`

```
X: 1
M: 4/4
L: 1/4
K: C
(CDEF|GFG)z|(gfed|cdc)z||
```

4.3.4 テヌートとフェルマータ

音の長さを、次の音の直前まで充分に保つ、つまりテヌート (tenuto) で演奏するという奏法をあらわす記号は、音名の左側に `!tenuto!` という記号指定を書くことによって表示することができます。

音や休止の時間を延長させるという奏法をあらわす記号、つまり、フェルマータ (fermata) と呼ばれる奏法記号は、`!fermata!` という記号指定を書くことによって表示することができます。

ABC 譜の例 `tenuto.abc`

```
X: 1
M: 4/4
L: 1/4
K: F
.F.G.A.B|.c.d.e.f| \
!tenuto!f!tenuto!e!tenuto!d!tenuto!c| \
!tenuto!B!tenuto!A!tenuto!G!fermata!F|]
```

4.3.5 その他の奏法

ABC には、奏法記号を表示するための記号指定として、`!tenuto!` や `!fermata!` のほかに、次のようなものがあります。

```
!trill!          トリル
```

!turn!	ターン
!uppermordent!	モルデント（上昇）
!lowermordent!	モルデント（下降）
!arpeggio!	アルペジオ
!0! – !5!	指番号

ABC 譜の例　articul.abc

```
X: 1
M: 4/4
L: 1/4
K: C
!trill!C!turn!D!uppermordent!E!lowermordent!F| \
!arpeggio![C4E4G4]|!0!C!1!D!2!E!3!F|!4!G!5!A2z|]
```

4.3.6　省略形の定義

ひとつの ABC 譜の中で同一の記号を何回も記述する必要がある場合、そのたびに記号指定を書いてもいいのですが、ABC は、記号指定と同じ意味を持つ省略形を定義することができる、という機能を持っていますので、その機能を使うことによって、ABC 譜を簡潔に記述することができるようになります。

記号指定の省略形は、1 個の文字で構成されます。省略形として使うことのできる文字は、英字の大文字の H から Z まで、小文字の h から w まで、そしてチルダ (tilde, ~) です。

記号指定の省略形を定義したいときは、

U: 省略形 = 記号指定

という形のフィールドを書きます。たとえば、

```
U: P = !arpeggio!
```

というフィールドを書くことによって、アルペジオを表示する記号指定と同じ意味を持つ P という省略形を定義することができます。

ABC 譜の例　abbrevi.abc

```
X: 1
M: 4/4
L: 1/4
K: C
U: T = !tenuto!
TCTDTETF|TGTATBTc|TcTBTATG|TFTETDTC|]
```

第 5 章　フォーマット属性

5.1　ページの大きさ

5.1.1　この章について

第 3.3 節でも説明しましたが、楽譜のフォーマットに関する属性（つまりフォーマット属性）に情報を設定したいときは、フィールドではなくて、「擬似注釈」(pseudocomment) と呼ばれる、

%% 属性名 文字列 改行

という形の文字列を書く必要があります。

フォーマット属性にはさまざまなものがあるのですが、この章では、それらのうちの主要なものをいくつか紹介したいと思います。

5.1.2　長さをあらわす文字列

フォーマット属性のうちには、情報として何かの長さを設定するものが数多くあります。そのようなフォーマット属性に情報を設定するためには、長さをあらわす文字列を擬似注釈の中に書かないといけません。長さをあらわす文字列は、

10 進数 単位

という形で記述します。「単位」のところには、長さをあらわすために使われている単位を書きます。

長さの単位としては、cm とインチとポイントを使うことができます。1 インチは 2.54 cm で、1 ポイントは 1/72 インチ（約 0.352778 mm）です。

cm は `cm` という文字列であらわされ、インチは `in` という文字列であらわされ、ポイントは `pt` という文字列であらわされます。たとえば、`2.37cm` という文字列を書くことによって、2.37 cm という長さを記述することができます。

5.1.3 ページの横の長さと縦の長さ

abcm2ps は、デフォルトではページの大きさを A4 判 (21.0 cm × 29.7 cm) と考えて楽譜をレイアウトします。ページの大きさを変更したいときは、

`pagewidth` 横の長さ
`pageheight` 縦の長さ

というフォーマット属性のそれぞれに、長さをあらわす文字列を設定します。

ABC 譜の例 `pagesize.abc`

```
%%pagewidth  14.8cm % (A5)
%%pageheight 21.0cm % (A5)
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|]
```

この ABC 譜を abcm2ps を使って楽譜に変換すると、その楽譜は、A5 判 (14.8 cm × 21.0 cm) のページの中にレイアウトされます。

なお、ps2pdf は、デフォルトではページの大きさを A4 判とみなして PostScript の文書を PDF に変換します。それ以外の大きさのページを変換させたいときは、ps2pdf を起動するコマンドに、

`-sPAPERSIZE=` 判型

というオプションを付けます。たとえば、

```
ps2pdf -sPAPERSIZE=a5 Out.ps
```

というコマンドを入力することによって、`Out.ps` の文書を A5 判とみなして PDF に変換することができます。

5.1.4 ページの余白

ページの余白 (margin) の大きさを設定したいときは、

属性名	方向	デフォルト
`topmargin`	上	1 cm
`botmargin`	下	1 cm
`leftmargin`	左	1.8 cm
`rightmargin`	右	1.8 cm

というフォーマット属性に、ページの端から楽譜までの距離をあらわす文字列を設定します。

ABC 譜の例 `margin.abc`

```
%%leftmargin  7cm
%%rightmargin 3cm
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|]
```

5.1.5　拡大率

ページの上に書き込まれる楽譜の拡大率を設定したいときは、scale というフォーマット属性に、拡大率をあらわす文字列を設定します。たとえば、この属性に 2 を設定すると、楽譜は 2 倍の大きさで表示されることになります。同じように、0.5 を設定すると、半分の大きさで表示されます。なお、このフォーマット属性のデフォルトは、0.75 です。

ABC 譜の例　scale.abc

```
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|
%%scale 1
CDEF|GABc|cBAG|FEDC|
%%scale 0.5
CDEF|GABc|cBAG|FEDC|CDEF|GABc|cBAG|FEDC|
%%scale 2
CDEF|GABc|]
```

5.2　空白

5.2.1　行と譜表の間隔

行と行とのあいだや譜表と譜表とのあいだに挿入される垂直方向の間隔を設定したいときは、次のようなフォーマット属性に、間隔をあらわす文字列を設定します。

属性名	対象	デフォルト
staffsep	行と行とのあいだ	46 ポイント
sysstaffsep	譜表と譜表とのあいだ	36 ポイント

ABC 譜の例　staffsep.abc

```
%%staffsep    6cm
%%sysstaffsep 3cm
X: 1
M: 4/4
L: 1/4
K: C
V: vocal
CDEF|GABc|cBAG|FEDC|
CDEF|GFED|CDEF|GFED|]
V: piano
EFGA|Bcde|edcB|AGFE|
EFGA|BAGF|EFGA|BAGF|]
```

5.2.2　冒頭部分の空白

楽譜の冒頭部分に表示される、タイトルや作曲者名などの上に挿入される垂直方向の空白を設定したいときは、次のようなフォーマット属性に、垂直方向の長さをあらわす文字列を設定します。

属性名	場所	デフォルト
titlespace	タイトルの上	0.2 cm
subtitlespace	サブタイトルの上	0.1 cm
composerspace	作曲者の上	0.2 cm
musicspace	最初の譜表の上	0.2 cm

ABC 譜の例　tispace.abc

```
%%titlespace    1cm
%%subtitlespace 2cm
```

```
%%composerspace 4cm
%%musicspace    8cm
X: 1
T: The Titlespace
T: The Vertical Space at the Beginning of the Song
C: Nakaoka Hideo
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|]
```

5.2.3 歌詞と譜表とのあいだの空白

歌詞と譜表とのあいだに挿入される垂直方向の空白を設定したいときは、次のようなフォーマット属性に、垂直方向の長さをあらわす文字列を設定します。

属性名	場所	デフォルト
vocalspace	譜表の下の歌詞の下	23 ポイント
wordsspace	楽譜の末尾の歌詞の上	0.5 cm

ABC 譜の例 lyspace.abc

```
%%vocalspace 3cm
%%wordsspace 5cm
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|
w: wa ta shi ha se ka i de i chi ba n no n ki da
CDEF|GABc|cBAG|FEDC|]
W: I am the most easygoing person in the world.
```

5.2.4 インデント

楽譜の 1 行目だけをインデント（字下げ）したいときは、indent というフォーマット属性に、字下げする長さを設定します。このフォーマット属性は、デフォルトではゼロに設定されています。

ABC 譜の例 indent.abc

```
%%indent 5cm
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|
CDEF|GFED|CDEF|GFED|]
```

5.2.5 改ページ

楽譜の途中で強制的にページを改めたいときは、ABC 譜のその場所に、

```
%%newpage
```

という擬似注釈を書きます。

ABC 譜の例 newpage.abc

```
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|
%%newpage
[L:1/8] CC DD EE FF|GG AA BB cc|cc BB AA GG|FF EE DD CC|
%%newpage
[L:1/2] CD|EF|GA|Bc|cB|AG|FE|DC|]
```

5.3　ヘッダーとフッター

5.3.1　ヘッダーとフッターを表示する方法

楽譜を出力するページにヘッダーやフッターを表示したいときは、ヘッダーやフッターにしたい文字列をフォーマット属性に設定します。

header というフォーマット属性に文字列を設定すると、その文字列は、楽譜が出力されるページの上端にヘッダーとして表示されます。同じように、footer というフォーマット属性に設定された文字列は、ページの下端にフッターとして表示されます。

ABC 譜の例　header.abc

```
%%header I am a header.
%%footer I am a footer.
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|]
```

5.3.2　水平方向の位置

原則的には、ヘッダーやフッターとして設定された文字列は、ページの中央にセンタリングされて表示されます。文字列を左寄せしたり右寄せしたりしたいときは、文字列の中に 1 個または 2 個のタブ (tab) という文字を挿入します。

文字列の中に 1 個のタブを挿入して、

文字列 1	タブ	文字列 2

という形の文字列をヘッダーまたはフッターとして設定した場合、文字列 1 は左寄せ、文字列 2 はセンタリングされて表示されます。

文字列の中に 2 個のタブを挿入して、

文字列 1	タブ	文字列 2	タブ	文字列 3

という形の文字列をヘッダーまたはフッターとして設定した場合、文字列 1 は左寄せ、文字列 2 はセンタリング、文字列 3 は右寄せされて表示されます。

左端だけ、中央と右端だけ、または右端だけにヘッダーやフッターを表示したいときは、タブで終わるかまたはタブで始まる文字列を書けばいいわけですが、その場合は、文字列の全体を二重引用符 (double quotation mark, ") で囲む必要があります。

ABC 譜の例　align.abc

```
%%header left→center→right
%%footer "→mannaka→migi"
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|]
```

タブは、人間の目には何個かの空白と同じように見えてしまいます。そのため、上の ABC 譜の例ではタブを矢印 (→) で置き換えていますので、注意してください。

5.3.3　変数

ヘッダーやフッターの中にページ番号や日付などの情報を表示したいということがしばしばあります。abcm2ps は、ページ番号や日付などの情報に対して「変数」(variable) と呼ばれる名前を与えていて、変数を使うことによって、それに対応している情報をヘッダーやフッターの中に表示することができるようにしています。表 5.1 は、それぞれの変数に対応している情報を示しています。

変数に対応している情報は、その変数の「値」(value) と呼ばれます。ヘッダーやフッターの

変数	情報
D	処理された日付と時刻
F	ファイル名
P	ページ番号
P0	ページ番号（偶数の場合のみ表示）
P1	ページ番号（奇数の場合のみ表示）
T	現在処理中の楽曲のタイトル
V	abcm2ps のバージョン

表 5.1: 変数

中に変数の値を表示したいときは、「変数展開」(variable expansion) と呼ばれる文字列を、ヘッダーやフッターのフォーマット属性に設定する文字列の中に書きます。

変数展開というのは、ドルマーク (dollar sign, $) の右側に変数を書いたもののことです。ヘッダーやフッターの中に変数展開が含まれている場合、その変数展開は、その中の変数の値に置き換えて表示されます。たとえば、$P という変数展開は、処理されているページの番号に置き換わります。

ABC 譜の例　pagenum.abc

```
%%header $P0→$D→$P1
%%footer $T($F)→$P→$V
X: 1
T: The Page Number
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|
%%newpage
[L:1/8] CC DD EE FF|GG AA BB cc|cc BB AA GG|FF EE DD CC|
%%newpage
[L:1/2] CD|EF|GA|Bc|cB|AG|FE|DC|
%%newpage
[L:1/4] CDEF|GFED|CDEF|GFED|C4|]
```

5.4 その他のフォーマット属性

5.4.1 小節番号の表示

abcm2ps は、小節番号、すなわちそれぞれの小節に与えられた番号を、小節の左上に表示するという機能を持っています。

小節番号を表示したいときは、measurenb というフォーマット属性に対して、小節番号を何番目ごとに表示するのかということをあらわす整数を設定します。たとえば、3 という整数を設定すると、3 番目、6 番目、9 番目、12 番目、……というように、3 の倍数ごとに小節番号が表示されます。

measurenb に対して 0 という整数を設定すると、楽譜のそれぞれの行の左端にある小節の小節番号だけが表示されます。

ABC 譜の例　measnum.abc

```
%%measurenb 4
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|
CDEF|GFED|CDEF|GFED|
CDEF|GABc|cBAG|FEDC|
CDEF|GFED|CDEF|GFED|C4|]
```

小節番号は、デフォルトでは先頭の小節が 1 番目になっていますが、measurefirst というフォーマット属性に整数を設定すると、その整数が先頭の小節の番号になります。先頭の小節にゼロやマイナスの整数を小節番号として与えることも可能ですが、1 以下の小節番号は、楽譜に表示することができません。

measurebox というフォーマット属性に対して true という真偽値を設定すると、abcm2ps は、小節番号を四角形で囲んで表示します。

ABC 譜の例　measbox.abc

```
%%measurenb 1
%%measurefirst 10
%%measurebox true
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|
CDEF|GFED|CDEF|GFED|C4|]
```

5.4.2　テキストの表示

abcm2ps は、楽譜が出力されるページの上に任意のテキスト（文字列）を書き込むこともできます。

1 行のテキストを書き込みたいときは、text または center というフォーマット属性に対して、書き込みたいテキストを設定します。text は、設定されたテキストを左寄せで書き込むフォーマット属性で、center は、設定されたテキストをセンタリングして書き込むフォーマット属性です。

複数の行から構成されるテキストを書き込みたいときは、begintext と endtext というペアになった属性名を指定した擬似注釈を書きます。それらの二つの疑似注釈のあいだには、

%% テキスト

という形の疑似注釈を何個でも好きなだけ書くことができて、それらの疑似注釈の中のテキストは、楽譜が出力されるページの上に書き込まれます。たとえば、

```
%%begintext
%%first line
%%second line
%%third line
%%endtext
```

という一連の疑似注釈を書くことによって、3 行のテキストをページの上に書き込むことができます。

ABC 譜の例　text.abc

```
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|
%%text one line of text
CDEF|GABc|cBAG|FEDC|
%%center one line of centered text
CDEF|GABc|cBAG|FEDC|
%%begintext
%%meny lines of text
%%Yoshikawa Kumiko
%%Togawa Arika
%%Kusanagi Motoko
%%endtext
CDEF|GABc|cBAG|FEDC|]
```

属性名	設定の対象	デフォルトのフォントと大きさ
titlefont	タイトル	Times-Roman 20
subtitlefont	サブタイトル	Times-Roman 16
composerfont	作曲者名	Times-Italic 14
headerfont	ヘッダー	Times-Roman 12
footerfont	フッター	Times-Roman 12
measurefont	小節番号	Times-Italic 14
textfont	テキスト	Times-Roman 16
vocalfont	譜表の下の歌詞	Times-Bold 13
wordsfont	楽譜の末尾の歌詞	Times-Roman 16

表 5.2: フォントを設定するフォーマット属性

5.4.3 フォント

abcm2ps は、文字を表示するために使われるフォントを設定するフォーマット属性をいくつも持っています。表 5.2 は、そのようなフォーマット属性のうちの主要なものを示しています。

フォントを設定するフォーマット属性には、フォントの名前とフォントの大きさ（単位はポイント）を設定します。たとえば、

```
%%titlefont Helvetica 30
```

という擬似注釈を書くことによって、タイトルを表示するために使うフォントとして、大きさが 30 ポイントの Helvetica というフォントを設定することができます。

なお、フォントの名前として使うことができるのは、

```
Times-Roman   Times-Italic   Times-Bold   Helvetica   Courier
```

などの、PostScript のフォントの名前です。

ABC 譜の例 font.abc

```
%%titlefont    Helvetica    40
%%subtitlefont Times-Roman  20
%%composerfont Courier      18
%%headerfont   Times-Italic 18
%%footerfont   Courier      24
%%measurefont  Helvetica    20
%%textfont     Times-Bold   36
%%vocalfont    Courier      16
%%wordsfont    Helvetica    22
%%header Times-Italic 18
%%footer Courier 24
%%measurenb 1
X: 1
T: Helvetica 40
T: Times-Roman 20
C: Courier 16
M: 4/4
L: 1/4
K: C
CDEF|GABc|cBAG|FEDC|
w: wa ta shi ha se ka i de i chi ba n no n ki da
%%text Times-Bold 36
CDEF|GABc|cBAG|FEDC|]
W: I am the most easygoing person in the world.
```

5.5　フォーマットファイル

5.5.1　フォーマットファイルとは何か

すでに説明したように、ABC 譜の中には、楽曲そのものの記述や楽曲についての情報の記述を書くことができるだけではなくて、楽曲を楽譜に変換するときに適用されるフォーマットに関する記述を書くこともできます。

実は、フォーマットに関する記述を書くことのできる場所は、ABC 譜の中だけではありません。ABC 譜が格納されているファイルとは別のファイルの中にフォーマットに関する記述を書いておいて、ABC 譜を楽譜に変換するときに、そのフォーマットを楽譜に適用する、ということも可能なのです。このように、楽曲の記述とフォーマットの記述とを分離することによって、ひとつの楽曲をさまざまなフォーマットの楽譜に変換したり、さまざまな楽曲のフォーマットをひとつに統一したりする、ということが簡単にできるようになります。

ABC 譜を含まないで、フォーマットに関する記述だけを含んでいるファイル、またはそのファイルの内容は、「フォーマットファイル」(format file) と呼ばれます。フォーマットファイルの名前に付ける拡張子としては、通常、 `.fmt` が使われます。

フォーマットファイルの中では、どのフォーマット属性に対しても情報を設定することができます。しかし、ページにテキストを書き込んだり、強制的にページを改めたりする記述などは、フォーマットファイルの中に書いたとしても楽譜には適用されません。

5.5.2　フォーマットファイルの書き方

フォーマットファイルの中では、フォーマット属性に情報を設定する記述は、疑似注釈という形ではなくて、

属性名	文字列	改行

という形で書きます（つまり、擬似注釈の先頭からパーセントパーセント (`%%`) を取り除いた形です)。

フォーマットファイルの中に注釈を書く方法は、ABC 譜の場合と同じです。つまり、パーセント (percent, `%`) という文字を書けば、その直後から改行までが注釈とみなされます。

フォーマットファイルの中では、連続する 2 個の改行、つまり空行は、意味を持ちません。ですから、フォーマットファイルを読みやすくするためなどの目的で自由に空行を使うことができます。

フォーマットファイルの例　`forfile.fmt`

```
pagewidth    21.0cm % (A4)
pageheight   29.7cm % (A4)
topmargin    2.5cm
botmargin    2.5cm
leftmargin   3cm
rightmargin  3cm
scale        0.8

header       $F
footer       $P

measurefont  Courier 12
measurenb    1
measurebox   true
```

5.5.3　フォーマットファイルの適用

abcm2ps を使って ABC 譜を楽譜に変換するとき、フォーマットファイルに記述されたフォーマットをそれに適用するためには、そのためのオプションをコマンドの中に書く必要があります。

ABC 譜を楽譜に変換するときにフォーマットファイルを適用したいときは、

`abcm2ps -F` | フォーマット | | ABC 譜 |

という形のコマンドで、abcm2ps を起動します。「フォーマット」のところにはフォーマットファイルを指定するパス名を書いて、「ABC 譜」のところには ABC 譜が格納されているファイルのパス名を書きます。たとえば、

```
abcm2ps -F forfile.fmt score.abc
```

というコマンドをシェルに入力することによって、score.abc というファイルに格納されている ABC 譜を、forfile.fmt というフォーマットファイルを適用して楽譜に変換することができます。

第 6 章　記号の定義

6.1　記号の定義の基礎

6.1.1　この章について

abcm2ps は楽譜の上にさまざまな記号を表示することができるわけですが、abcm2ps といえども、音楽で使われるすべての記号に対応しているわけではありません。ですから、場合によっては、表示させたい記号があるのに、それを表示する機能が abcm2ps の側に備わっていない、ということもあります。

しかし、abcm2ps は、新しい記号を定義するという機能を持っています。ですから、abcm2ps が対応していない記号も、それを定義することによって abcm2ps に表示させることができます。

そこで、この章では、楽譜の上に表示される記号を新しく定義する方法について説明していきたいと思います。

6.1.2　記号を定義するための 2 段階の記述

ABC では、記号を定義するために 2 段階の記述を書く必要があります。それは、

(1) PostScript による手続きの定義
(2) ABC による記号の定義

という 2 段階です。

PostScript というのは、ページの上に表示される図形や文字などを記述するための言語のひとつです。abcm2ps が出力する楽譜も、PostScript で記述されています。そして、「手続き」(procedure) というのは、何らかの動作をあらわしている記述のことです。記号を定義するための第一段階は、PostScript という言語を使って、記号を描画するという動作をあらわしている手続きを定義する、ということです。

6.1.3　PostScript による手続きの定義

ABC 譜の中やフォーマットファイルの中に PostScript の記述を書くためには、postscript というフォーマット属性を使う必要があります。このフォーマット属性に設定された文字列は、PostSciptによる記述とみなされて処理されます。

PostScript で手続きを定義する方法については、次の節以降で詳しく説明する予定ですので、ここでは、例をひとつ挙げるだけにしておきたいと思います。たとえば、

```
%%postscript /arcstroke { 2 0 360 arc stroke } def
```

という疑似注釈を書くことによって、arcstroke という名前を持つ手続きを定義することができます。この手続きは、半径が 2 ポイントの円を描画するという動作をあらわしています。

6.1.4　ABC による記号の定義

PostScript による手続きの定義を使って ABC による記号の定義を書きたいときは、deco というフォーマット属性を使います。このフォーマット属性に情報を設定する疑似注釈は、

%%deco | 記号名 | 位置 | 手続き名 | 高さ | 左横幅 | 右横幅 |

と書きます。「記号名」のところには任意の名前、「位置」のところには記号を表示する位置、「手続き名」のところには PostScript の手続きの名前、「高さ」のところには記号の高さ、「左横幅」と「右横幅」のところには左方向と右方向への横幅を書きます（高さと横幅の単位はポイント）。

記号を表示する位置は、位置をあらわす次のような整数で指定します。

0　音符の符頭の上または下。
1　音符の符頭の左。

2　音符の符頭の左下。
3　音符の上。
4　音符の下。
5　譜表の上（開始位置と終了位置で指定する場合）。
6　譜表の下。
7　譜表の下（開始位置と終了位置で指定する場合）。

このような擬似注釈を書くことによって、指定された PostScript の手続きを実行することによって描画される記号に対して、「記号名」のところに書いた名前を与えることができます。たとえば、

```
%%deco circle 0 arcstroke 6 2 2
```

という擬似注釈は、 arcstroke という PostScript の手続きを実行することによって描画される記号に対して circle という名前を与えます。

deco を使って記号を定義すると、その記号に与えられた名前を ABC の記号指定の中（つまり二つの感嘆符のあいだ）に書くことによって、楽譜の上にその記号を表示することができます。たとえば、音の記述の左側に !circle! という記号指定を書くことによって、 circle という名前が与えられた記号を、その音をあらわす音符のそばに表示することができます。

ABC 譜の例　circle.abc

```
%%postscript /arcstroke { 2 0 360 arc stroke } def
%%deco circle 0 arcstroke 4 2 2
X: 1
M: 4/4
L: 1/4
K: F
!circle!F!circle!G!circle!A!circle!B| \
!circle!c!circle!d!circle!e!circle!f| \
fedc|BAGF|]
```

6.2　PostScript の基礎

6.2.1　スタック

PostScript による動作の記述は、「スタック」(stack) と呼ばれるものを使って実行されます。スタックというのは、データの列を格納することのできる容器の一種で、データを入れるときの順番と、それらを取り出すときの順番とが逆になるもののことです。スタックにデータを入れることを「プッシュ」(push) と言い、スタックからデータを取り出すことを「ポップ」(pop) と言います。

6.2.2　トークン

PostScript では、空白または改行で区切られた、「トークン」(token) と呼ばれる文字列を並べていくことによって動作を記述します。

トークンは、処理の対象となるデータをあらわしているものと、何らかの動作に与えられている名前とに分類されます。動作に与えられている名前は、さらに「オペレーター」(operator) と「手続き名」(procedure name) とに分類されます。オペレーターというのは、PostScript で書かれた記述を処理するプログラムに最初から組み込まれている動作に与えられている名前のことで、手続き名というのは、PostScript で書かれた記述によって定義された動作の名前のことです。

PostScript の記述は、基本的には先頭のトークンから順番に処理されていきます。トークンの処理というのは、それがあらわしているものがデータならばそれをスタックにプッシュして、動作ならばそれを実行する、ということです。

6.2.3　座標系

PostScript では、紙の上での点の位置を指定するために、「座標系」(coordinate system) と呼ばれるものを使います。PostScript の座標系は、「x 軸」(x-axis) と「y 軸」(y-axis) と呼ばれる、方向のある 2 本の直線を持っています。デフォルトでは、x 軸は右を向いていて、y 軸は上を向いています。

x 軸と y 軸が交わっている点は「原点」(origin) と呼ばれます。原点は、デフォルトでは紙の左下の隅にあります。

紙の上での点の位置は、「座標」(coordinates) と呼ばれる 2 個の数値の列によって指定することができます。座標を構成する数値の 1 個目は、原点から指定される点までの x 軸方向の距離で、「x 座標」(x-coordinate) と呼ばれます。同じように、2 個目は y 軸方向の距離で、「y 座標」(y-coordinate) と呼ばれます。デフォルトでは、距離の単位はポイントです。

6.2.4 カレントパス

PostScript で図形を描画するためには、まず図形を作って、そののちそれを描画する、という 2 段階の操作をする必要があります。図形を作るというのは、「カレントパス」(current path) と呼ばれるものに図形を追加するということです。カレントパスというのは、図形の列を格納することのできる容器、またはそれに格納されている図形の列のことです。

`arc` というオペレーターは、スタックから 5 個のデータをポップして、それらのデータにしたがってひとつの円弧をカレントパスに追加します。データの意味は、スタックにプッシュする順番で言うと、1 個目が中心の x 座標、2 個目が中心の y 座標、3 個目が半径、4 個目が開始角度、5 個目が終了角度です。角度は右方向が 0 度で、円弧を描画するための回転の方向は反時計回りです。たとえば、

```
200 150 3 20 340 arc
```

という記述を処理することによって、中心の座標が $(200, 150)$、半径が 3、開始角度が 20 度、終了角度が 340 度、という円弧をカレントパスに追加することができます。

6.2.5 図形の描画

カレントパスは、`stroke` や `fill` というオペレーターを使うことによって描画することができます。`stroke` はカレントパスを線だけで描画するオペレーターで、`fill` はカレントパスで囲まれた領域を塗りつぶすオペレーターです。たとえば、

```
200 150 3 20 340 arc stroke
```

という記述を処理することによって、ひとつの円弧を描画することができます。

6.2.6 手続きの定義

PostScript では、

/ 手続き名 { 動作の記述 } def

という形のものを書くことによって、「動作の記述」というところに書かれた動作に対して、「手続き名」のところに書かれた名前を与えることができます。ですから、

```
/rightgap { 200 150 3 20 340 arc stroke } def
```

という記述を書けば、先ほどの円弧を描画する動作に対して `rightgap` という名前が与えられることになります。

abcm2ps によって扱われる PostScript の手続きは、記号を表示する位置の x 座標と y 座標がスタックにプッシュされたのちに実行されます。ですから、楽譜の上に記号を表示するための手続きを定義するときは、図形を描画するべき位置の x 座標と y 座標はすでにスタックにプッシュされていると考えて動作の記述を書く必要があります。たとえば、

```
/rightgap { 3 20 340 arc stroke } def
```

という記述を書くことによって定義された手続きは、楽譜の上の適切な位置に円弧を描画することになります。

ABC 譜の例 `rightgap.abc`

```
%%postscript /rightgap { 3 20 340 arc stroke } def
%%deco rg 6 rightgap 6 3 3
X: 1
M: 4/4
L: 1/4
K: F
!rg!F!rg!G!rg!A!rg!B|!rg!c!rg!d!rg!e!rg!f|fedc|BAGF|]
```

6.3　直線の描画

6.3.1　カレントポイント

次に、PostScript で直線を描画する方法について説明しましょう。

PostScript で直線を扱うときには、「カレントポイント」(current point) と呼ばれるものが重要な役割を果たします。カレントポイントは、日本語で言えば「現在の点」です。この言葉のとおり、カレントポイントは、状況の推移にともなって変化していく点のことです。

直線をカレントパスに追加するためには、どこかにカレントポイントが存在していないといけません。それがまだ存在していないならば、新しく作る必要があります。カレントポイントを作りたいときは、`moveto` というオペレーターを使います。このオペレーターは、スタックから x 座標と y 座標をポップして、それらによって指定された位置にカレントポイントを作ります。たとえば、

```
480 630 moveto
```

という記述を実行することによって、$(480, 630)$ という位置にカレントポイントを作ることができます。

6.3.2　直線を作るオペレーター

直線をカレントパスに追加したいときは、`rlineto` というオペレーターを使います。このオペレーターは、カレントポイントを出発点として、指定された位置で終わる直線をカレントパスに追加します。そして、カレントポイントは、直線の終わりの位置へ移動します。終わりの位置は、カレントポイントからの x 軸方向への移動量と y 軸方向への移動量で指定します。たとえば、

```
40 30 rlineto
```

という記述を実行すると、カレントポイントを出発点として、そこから x 軸方法へ 40、y 軸方向へ 30 だけ移動した位置で終わる直線がカレントパスに追加されて、それとともにカレントポイントも移動します。

`rlineto` を実行するたびにカレントポイントは直線の終わりの位置へ移動しますので、`rlineto` を連続的に実行することによって一連の折れ線をカレントパスに追加することができます。たとえば、

```
/kanako {
    moveto
     10   0 rlineto
      0 -10 rlineto
    -10   0 rlineto
    stroke
} def
```

という記述によって定義される `kanako` という手続きは、「コ」という片仮名の形の図形を描画します。

なお、このような複数の行にまたがる PostScript の記述を ABC 譜の中に書きたいときは、それぞれの行を設定する `postscript` の擬似注釈を連続して書きます。

ABC 譜の例　`kanako.abc`

```
%%postscript /kanako {
%%postscript     moveto
%%postscript      10   0 rlineto
%%postscript       0 -10 rlineto
%%postscript     -10   0 rlineto
%%postscript     stroke
%%postscript } def
%%deco ko 6 kanako 10 0 10
X: 1
M: 4/4
L: 1/4
K: F
!ko!F!ko!G!ko!A!ko!B|!ko!c!ko!d!ko!e!ko!f|fedc|BAGF|]
```

6.3.3 カレントポイントの移動

カレントパスを作るために rlineto しか使わないとすると、一筆書きで描ける図形しか作ることができないということになります。それでは、一筆書きでは描くことができない図形を作るためには、いったいどうすればいいのでしょうか。

一筆書きでは描くことができない図形を作るためには、直線をカレントパスに追加しないでカレントポイントを移動させる必要があるわけですが、そのような動作は、rmoveto というオペレーターを使うことによって実行することができます。このオペレーターは、rlineto と同じように、x 軸方向の移動量と y 軸方向の移動量をスタックからポップして、それらによって指定された位置へカレントポイントを移動させます。

ABC 譜の例 batsu.abc

```
%%postscript /batsu {
%%postscript     moveto
%%postscript      -5   5 rmoveto
%%postscript      10 -10 rlineto
%%postscript     -10   0 rmoveto
%%postscript      10  10 rlineto
%%postscript     stroke
%%postscript } def
%%deco bt 6 batsu 10 5 5
X: 1
M: 4/4
L: 1/4
K: F
!bt!F!bt!G!bt!A!bt!B|!bt!c!bt!d!bt!e!bt!f|fedc|BAGF|]
```

6.4 文字列の描画

6.4.1 フォント

次に、PostScript で文字列を描画する方法について説明しましょう。

文字列を描画するためには、それに先立って、そのために使うフォントを設定する必要があります。フォントは、selectfont というオペレーターを使うことによって設定することができます。

フォントを設定したいときは、フォント名とフォントの大きさを、この順番でスタックにプッシュして、そののち selectfont を実行します。そうすると、selectfont は、フォント名と大きさをポップして、指定されたフォントを指定された大きさで設定します。なお、フォント名のような名前をスタックにプッシュするためには、その名前の左側にスラッシュ(/)を書いておく必要があります。たとえば、

```
/Helvetica 16 selectfont
```

という記述を書くことによって、Helvetica というフォントを 16 ポイントの大きさで設定することができます。

6.4.2 文字列を描画するオペレーター

文字列を描画するためには、その文字列をスタックにプッシュする必要があります。文字列をスタックにプッシュしたいときは、それを丸括弧で囲んだものを書きます。たとえば、(rit.) という記述を書くことによって、rit. という文字列をスタックにプッシュすることができます。

文字列は、show というオペレーターを実行することによって描画することができます。show は、スタックから文字列をポップして、設定されているフォントを使ってカレントポイントにその文字列を描画して、文字列の末尾の位置へカレントポイントを移動させます。たとえば、

```
(poco a poco) show
```

という記述を書くことによって、poco a poco という文字列をカレントポイントに描画することができます。

ABC 譜の例 accel.abc

```
%%postscript /accelerando {
%%postscript     /Times-Italic 14 selectfont
%%postscript     moveto
```

```
%%postscript     -10 5 rmoveto
%%postscript     (accel.) show
%%postscript } def
%%deco accel 6 accelerando 14 30 30
%%exprabove true
X: 1
M: 4/4
L: 1/4
K: C
CDEF|GABc|!accel!cBAG|FEDC|]
```

6.4.3　任意の文字列を描画することのできる手続き

deco というフォーマット属性に情報を設定する疑似注釈は、

%%deco 記号名 位置 手続き名 高さ 左横幅 右横幅 文字列

というように、末尾に任意の文字列（空白を含んでいてもかまいません）を書くことができます。このように、deco の疑似注釈の末尾に文字列を書いた場合、PostScript の手続は、その文字列、x 座標、y 座標、という順番でデータがスタックにプッシュされたのちに実行されます。ですから、このことを利用することによって、任意の文字列を描画することのできる手続きを定義することができます。

文字列、x 座標、y 座標、という順番でデータがスタックにプッシュされているとするとき、

```
/string {
    /Times-Italic 14 selectfont
    moveto
    -10 5 rmoveto
    show
} def
```

という記述によって定義された string という手続きを実行すると、スタックからそれらのデータがポップされて、指定された位置に指定された文字列が描画されます。

ABC 譜の例　string.abc

```
%%postscript /string {
%%postscript     /Times-Italic 14 selectfont
%%postscript     moveto
%%postscript     -10 5 rmoveto
%%postscript     show
%%postscript } def
%%deco accel  6 string 14 30 30 accel.
%%deco rit    6 string 14 20 20 rit.
%%deco atempo 6 string 14 30 30 a tempo
%%exprabove true
X: 1
M: 4/4
L: 1/4
K: C
CDEF|!accel!GABc|!atempo!cBAG|!rit!FEDC|]
```

6.5　伸縮性のある記号

6.5.1　伸縮性のある記号についての復習

abcm2ps が表示することのできる記号の中には、クレッシェンドのような、横の長さに伸縮性のある記号もあります。伸縮性のある記号を表示したいときは、その開始位置と終了位置のそれぞれに対応する記述の左側に、開始の記号指定と終了の記号指定を書きます。たとえば、クレッシェンドの場合は、

開始　!crescendo(!

終了　!crescendo)!

という記号指定のそれぞれを、位置に対応する記述の左側に書くことによって、適切な長さを持つ記号を表示することができます。

6.5.2 伸縮性のある記号の定義

abcm2ps では、伸縮性のある記号を新しく定義する、ということも可能です。その場合は、ひとつの記号を定義するために deco の疑似注釈を二つ書く必要があります。ひとつは開始の記号名を、もうひとつは終了の記号名を定義する疑似注釈です。

記号を描画する PostScript の手続きの名前は、終了の記号名を定義する疑似注釈のほうだけに書きます。開始の記号名を定義する疑似注釈のほうは、PostScript の手続き名を書く位置に、マイナス (-) という文字を書きます。たとえば、

```
%%deco long( 5 -           8 0 0
%%deco long) 5 stretchable 8 0 0
```

という 2 行の疑似注釈を書くことによって、long(を開始の記号名、long) を終了の記号名とする記号を定義することができます。

普通、開始の記号名の末尾には左丸括弧を、終了の記号名の末尾には右丸括弧を付けます。ただし、これは単なる慣習にすぎませんので、丸括弧のない名前を付けても間違いではありません。

伸縮性のある記号を定義する場合、記号を表示する位置をあらわす整数として指定することができるのは、5 と 7 だけです。5 は譜表の上、7 は譜表の下を意味する整数です。

左右の横幅は、伸縮性のある記号を定義する場合には意味を持ちませんので、普通、どちらも 0 と書きます。

6.5.3 伸縮性のある記号を描画する手続き

伸縮性のある記号を描画する PostScript の手続きは、記号の横の長さ、開始位置の x 座標、開始位置の y 座標、という三つのデータが、この順番でスタックにプッシュされたのちに実行されます。ですから、

```
/line {
    moveto
    0 rlineto
    stroke
} def
```

という記述を書くことによって、開始位置から終了位置までの直線を描画する、line という手続きを定義することができます。

ABC 譜の例 line.abc

```
%%postscript /line {
%%postscript     moveto
%%postscript     0 rlineto
%%postscript     stroke
%%postscript } def
%%deco ol( 5 -    1 0 0
%%deco ol) 5 line 1 0 0
%%deco ul( 7 -    1 0 0
%%deco ul) 7 line 1 0 0
X: 1
M: 4/4
L: 1/4
K: C
!ol(!CDEF|GABc!ol)!|!ul(!cBAG|FEDC!ul)!|]
```

索引