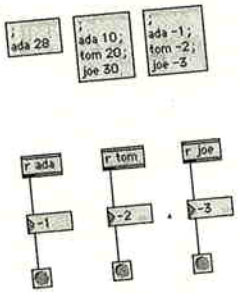
きる。この場合は、最初に ; (セミコロン)を入れて改行し、次の行にreceiveオブジェクトのアーギュメント名を指定し、スペースで区切ってメッセージを記述する。こうすれば1つのメッセージ・ボックスに複数のアーギュメントとメッセージを持たせることができるので、同時に異なるアーギュメントを持つ複数のreceiveオブジェクトにメッセージを送ることができる。この方法は、同時に多くメッセージを送りたいときに役立つだろう。

■3-4-16 メッセージ・ボックスによるreceiveオブジェクトへのメッセージ送信



3-5 数値計算の処理

Maxでは数値計算を行うオブジェクトが数多くある。ここでは、足す、引くといった簡単なオブジェクトや、三角関数やランダム関数などのオブジェクトについて説明する。C言語の数値演算ライブラリと同等の機能を持つオブジェクトもある。また、数値計算では整数と実数の違いが重要になるので、適切に用いるようにしなければならない。なお、これらを使いこなすには、ある程度数学的な知識が必要になるので、他の参考書なども合わせて参照してほしい。

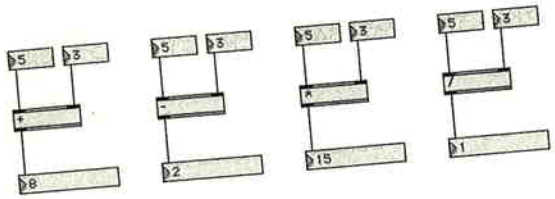
四則演算

Maxでは四則演算のために、以下のオブジェクトを用意している。

+	加算(足し算)
−	減算(引き算)
*	乗算(掛け算)
/	除算(割り算)

これらは、乗算の記号×を * で、除算の記号÷を / として表す違いはあるが、一般的な数学での用法と同じと考えてよい。これらのオブジェクトは、2つのインレットと1つのアウトレットを持ち、アーギュメントを1つ持つ。アーギュメントは省略可能であり、省略した場合は初期値として0が用いられる。なお、/ オブジェクトの初期値は1となる。オブジェクトの動作としては、第1インレットで受け取った数値を、アーギュメントもしくは第2インレットで受け取った数値で演算を行い、その結果をアウトレットから出力する。ただし、後述するright-to-left orderのルールにより、2つの数値を与えるときは第2インレットから先に数値を入力しなければならない。したがって、5−3という計算を行うには、−オブジェクトの第2インレットに3を送り、次いで第1インレットに5を送る。これで計算結果である2が−オブジェクトのアウトレットから出力される。

■3-5-1 四則演算のオブジェクト



さらに、次のような算術オブジェクトも利用できる。

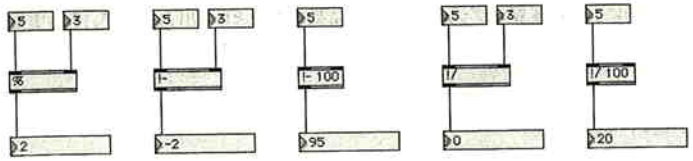
%	剰余算(割り算の余り)
!−	減算(逆順の引き算)
!/	除算(逆順の割り算)

%オブジェクトは剰余算を行い、整数での除算で生じる余りを求める。5%3なら商は1で、余りは2となる。

!−オブジェクトは減算だが、インレットの役目が−オブジェクトとは逆になっている。つまり、!−オブジェクトではアーギュメントまたは第2インレットに入力された数値から、第1インレットに入力された数値を引くことになる。例えば、5!−3は3−5と同じで、結果は−2となる。!−オブジェクトは、100−Xのように特定の数値から任意の数値を引きたい場合に役立つだろう。

同じように、!/オブジェクトは除算だが、/ オブジェクトとはインレットの役目が逆になる。!/オブジェクトはアーギュメントまたは第2インレットに入力された数値で、第1インレットに入力された数値を割るわけだ。例えば、5!/3は3/5のように計算され、結果は0となる。!/オブジェクトは、100/Xのように特定の数値を任意の数値で割りたい場合に役立つだろう。

■3-5-2 その他の演算オブジェクト

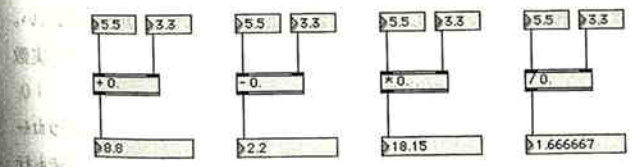


整数と実数

先の3-5-1では、5/3の演算結果は1.666……ではなく1になっている。これは整数として演算処理されたからだ。整数演算では、小数点以下の小数部が生じても、それは単に切り捨てられて整数として扱われる。

実数として演算を行うには、まず、オブジェクトが実数演算を行うように指定しなければならない。これは、* 0. のように、オブジェクトのアーギュメントに実数を与えることで行う。アーギュメントを省略した場合は、初期値は0という整数になり、整数演算になってしまう。そのため、実数演算を行うオブジェクトの入力や出力につながるオブジェクトは、フロート・ナンバー・ボックスなどの実数を扱うオブジェクトを用いるのが一般的だ。

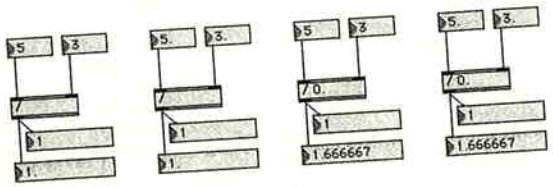
■3-5-3 実数演算のオブジェクト



ところで、実数演算オブジェクトへの入力は、整数でも実数でも構わない。整数が実数演算オブジェクトに入力された場合は、その整数は実数と見なされて演算が行われる。例えば、5という整数を入力した場合は、5.0という実数として扱われる。

同じように、実数演算オブジェクトは演算結果を実数として出力するが、その実数を受け取るオブジェクトが整数を扱うオブジェクトなら、実数の小数部が切り捨てられて整数として扱われる。3-5-4の例は除算オブジェクトでの整数と実数の処理結果の違いを示している。

■3-5-4 整数と実数の演算の違い



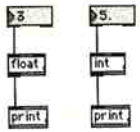
このように、整数と実数では処理結果が異なる場合がある。計算結果が正しくない場合は、整数と実数とを取り違えていることが考えられる。実数計算を行うパッチなら、実数を扱うオブジェクトを用いければならないので、このことを確認しておこう。

もちろん、整数計算で済む場合は、整数を扱うオブジェクトでパッチを構成すればよい。ただし、整数は有効範囲、すなわち、−2,147,483,648から2,147,483,647までの整数しか扱えないことに注意しよう。一方で、実数の場合は、誤差が生じることも覚えておくべきだ。

なお、整数を明確に実数に変換したいときは、floatオブジェクトを用いればよい。floatオブジェクトは実数を格納するオブジェクトだが、受け取ったintメッセージを実数に変換して、floatメッセージとして出力するからだ。

同様に、実数を整数に変換するにはintオブジェクトを利用する。同じ目的で、フロート・ナンバー・ボックスやナンバー・ボックスを用いてもよい。しかし、これらは表示も行うので、単に変換をしたい場合には冗長だろう。

■3-5-5 整数と実数の明示的変換



ビット演算

われわれが日常的に使う数値は十進数で表されているが、コンピューター内部では二進数で数値が表されている。そこで、数値をビット列として演算を行うオブジェクトが用意されている。

& ビット積(ビット単位のAND演算)
| ビット和(ビット単位のOR演算)

1ビットのビット積の演算は次のようになる。

0 & 0→0
0 & 1→0
1 & 0→0
1 & 1→1

また、ビット和は次の通りだ。

0 | 0→0
0 | 1→1
1 | 0→1
1 | 1→1

例えば、十進数での12と7を二進数で表すと、1100と0111というビット列になる。これらのビット列の桁ごとに、先のビット積演算を行えば、

1 & 0→0
1 & 1→1
0 & 1→0
0 & 1→0

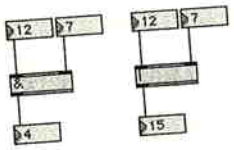
となり、0100というビット列になる。これを十進数に直せば4となる。

同じように、ビット和演算では、

1 | 0→1
1 | 1→1
0 | 1→1
0 | 1→1

となり、1111というビット列なので、十進数に直せば15となる。

■3-5-6 ビット積とビット和のオブジェクト



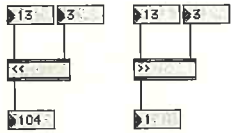
さらに、ビット列をシフトさせるオブジェクトもある。

<< ビット列を左にシフト
>> ビット列を右にシフト

<<オブジェクトも>>オブジェクトもアーギュメントまたは第2インレットに入力される数値をシフトするビット数とし、第1インレットに入力される数値をシフトする。

例えば、十進数の13は二進数では1101というビット列になるが、これを3ビット分左にシフトすれば、新たに生じる右側の3ビット分を0で埋めて1101000というビット列になる。これを十進数に直せば104になる。同じように十進数の13を3ビット右にシフトすると、1101というビット列の右から3ビット分が消滅し、1というビット列になる。これは十進数に直せば1になる。

■3-5-7 ビット・シフトのオブジェクト



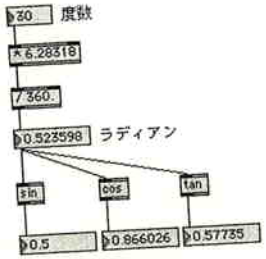
三角関数

Maxには三角関数としてサイン関数、コサイン関数、タンジェント関数と、それらの逆関数および双曲線のオブジェクトが用意されている。オブジェクト名の最初にaが付くのが逆関数で、オブジェクト名の最後にhが付くのが双曲線だ。

sin	サイン関数
asin	アーク・サイン関数
sinh	ハイパーボリック・サイン関数
cos	コサイン関数
acos	アーク・コサイン関数
cosh	ハイパーボリック・コサイン関数
tan	タンジェント関数
atan	アーク・タンジェント関数
atan2	アーク・タンジェント関数(二値入力)
tanh	ハイパーボリック・タンジェント関数

これらのオブジェクトは、角度をラディアンとして扱う。ラディアンは全周である360°を2πとする単位だ。したがって、度数をラディアンに変換するには、度数に6.28318(2×3.14159)を掛けて、360.0で割ればよい。

■3-5-8 三角関数のオブジェクト

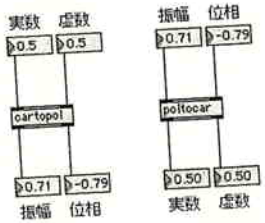


座標変換関数

2次元空間の位置を表す座標系である直交座標と極座標とを変換するために、以下のオブジェクトがある。これらのオブジェクトは、例えばFFT変換によって得られる実数と虚数を直交座標である複素数平面とし、対応するサイン波の振幅と位相を極座標として、これらを相互に変換するために用いられる。

cartopol	直交座標から極座標への変換
poltocar	極座標から直交座標への変換

■3-5-9 座標変換オブジェクト



ランダム関数

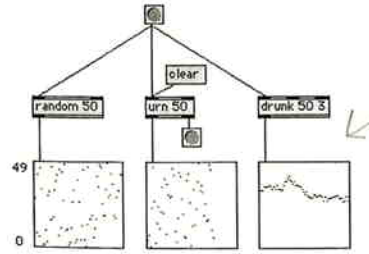
ランダムな数値を得るには、3つのオブジェクトを利用することができる。いずれもbangメッセージを受け取ったときに、0からアーギュメントで指定した数値より1小さい値までの範囲でランダムな数値を出力する。

randomオブジェクトは、bangメッセージを受け取る度にランダムな数値を出力する。これに対して、urnオブジェクトは重複しないようにランダムな数値を出力するので、アーギュメントで指定した数の数値を出力すれば、それ以上数値を出力せず、代わりに第2アウトレットからbangメッセージを出力する。clearメッセージを受け取れば、初期状態に戻って、ランダムな数値を出力するようになる。

drunkオブジェクトは、直前の値を基準として、第2アーギュメントで指定した数値より1小さい範囲で増減するような数値を出力する。例えば、直前の値が20で、第2アーギュメントが3であれば、次に出力される値は−2から2までの範囲で変化するので、18、19、20、21、22のいずれかとなる。

random	ランダムな数値を出力
urn	重複しないようにランダムな数値を出力
drunk	直前の値を基準として一定範囲で増減する数値を出力

■3-5-10 ランダムな数値を出力するオブジェクト

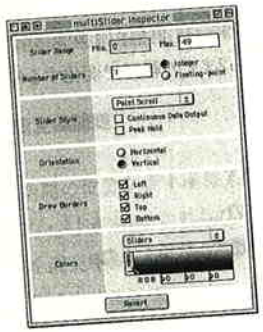


3-5-10の例では、アーギュメントを50としてランダムな数値を出力するオブジェクトを作成し、その出力をグラフ化している。buttonオブジェクトを数十回クリックすれば、そ

れらの違いが分かるだろう。randomオブジェクトは、常にランダムな数値を出力するので、同じ数値が出力されることがある。urnオブジェクトは0から49までの数値を重複しないようにランダムに出力するが、50個の数値を出力すれば、それ以上数値を出力しない。drunkオブジェクトの第2アーギュメントを3としたので、−2から2の範囲でランダムに増減しながら数値が変化している。これからも分かるように、泥酔者の千鳥足の様子に似ているので、drunkすなわち"酔っぱらい"という名前が付いている。

なお、3-5-10のパッチではmultiSlider《Multislider》オブジェクトを使ってグラフを表示させている。multiSliderのインスペクターでSlider StyleをPoint Scrollとすれば、グラフ表示に利用できる。

■3-5-11　multiSliderオブジェクトの設定



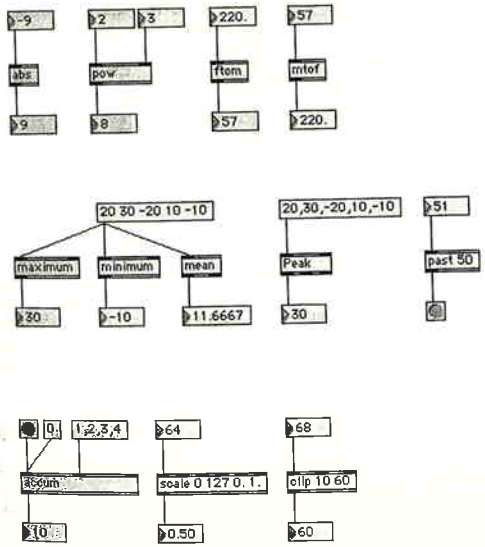
その他の数値処理

これまでに説明した他にも、Maxには数値を処理するオブジェクトがいくつかある。代表的なオブジェクトを以下にまとめておこう。

abs	入力される数値の絶対値を出力
pow	アーギュメントまたは第2インレットの数値を指数として、べき乗を出力
ftom	周波数からMIDIノート・ナンバーへ変換した結果を出力
mtof	MIDIノート・ナンバーから周波数へ変換した結果を出力

maximum	2つの数値のうち大きい数値、またはリスト中の最大値を出力
mean	入力される数値の平均値を出力
minimum	2つの数値のうち小さい数値、またはリスト中の最小値を出力
Peak	入力される数値から最大値を出力する
past	特定の値よりも大きな数値またはリストが入力されるとbangメッセージを出力
accum	第2インレットに受け取った数値を加算、第3インレットに受け取った数値で乗算、第1インレットにbangメッセージを受け取ると出力
scale	アーギュメントや第2インレット以降で指定した入力範囲と出力範囲に従って、入力される数値をスケーリングして出力
clip	アーギュメントや第2インレット以降で指定した範囲に収まるように、入力される数値をクリッピングして出力

■3-5-12　その他の数値処理オブジェクト

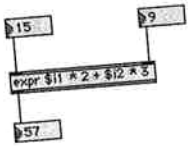


exprオブジェクトの演算式

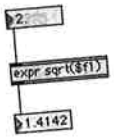
これまで説明した数値処理のオブジェクトは機能ごとに独立しているが、exprオブジェクトを用いれば、演算子や関数を組み合わせた演算式を記述することができる。演算式には$i1や$f2といった記号を含めて、演算式へのアーギュメントとして用いる。最初の$はアーギュメントであることを示し、次のiは整数であることを、fなら実数であることを示す。そして、3文字目の数値は対応するインレットの番号を示している。

したがって、$i1は第1インレットから入力される整数に置き換えられ、$f2は第2インレットから入力される実数に置き換えられて演算が行われる。オブジェクトのアーギュメントは、オブジェクトの処理の初期値といった意味合いだったが、exprオブジェクトのアーギュメントは、演算式とインレットから受け取る数値を表している。なお、$s2[1]のようにして、後述するテーブル・オブジェクトの要素を用いて演算を行うこともできる。簡単な例を挙げれば、expr $i1 * 2 + $i2 * 3なら、第2インレットに入力された整数を3倍、第1インレットに入力された整数を2倍し、それらを足して出力することになる。

■3-5-13 exprオブジェクトでの四則演算



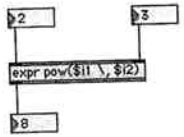
■3-5-14 関数を使った演算



exprオブジェクトにはC言語に準拠した演算子を含めることができるが、それ以外にもC言語に準拠した関数も利用できる。関数を用いる場合は3-5-14のように、関数名に続くカッコ内にアーギュメントを指定する。例えば、平方根を求める関数はsqrtなので、expr sqrt($f1)のように表記する。

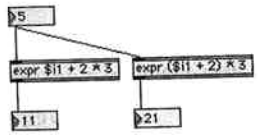
また関数の中にはpowやatan2のように複数のアーギュメントを必要とする関数がある。このような関数ではカッコ内に\,(バック・スラッシュとカンマ)で区切ってアーギュメントを記述する。バック・スラッシュ文字は日本語フォントや日本語キーボードでは¥(円マーク)に相当する。

■3-5-15 2つのアーギュメントを用いる関数の演算



exprオブジェクトに表記する演算式には、C言語に準拠した演算の優先順位があり、カッコを付けることで演算順位を指定することができる。例えば、* は+よりも演算順位が高いので、expr $i1 + 2 * 3は、2 * 3が先に計算されて6となり、入力された整数に6を加えて出力される。これに対して、expr ($i1 + 2) * 3とすれば、入力された整数に2を加えたあとに3を掛けた結果が出力されることになる。次の例で、演算順位によって結果が異なることが確認できる。

■3-5-16 演算順位の違い



exprオブジェクトでは次のような演算子が使用できる。

+	加算(足し算)
−	減算(引き算)
*	乗算(掛け算)
/	除算(割り算)
%	剰余算(割り算の余り)
~	ビット補数(ビット単位の補数)

^	ビット排他和(ビット単位のXOR演算)
&	ビット積(ビット単位のAND演算)
|	ビット和(ビット単位のOR演算)
&&	論理積(論理的AND演算)
||	論理和(論理的OR演算)
!	論理否定(論理的NOT演算)

演算子の優先順位は、次のようになる。同じ優先順位の演算子は、演算式の左側に表記されたものから演算が行われる。

優先順位	演算子		
1	()	関数	
2	!	-	
3	*	/	%
4	+	-	
5	&		
6	^		
7	|		
8	&&		
9	||		

また、exprオブジェクトで利用できる関数は次の通り。

abs	絶対値
sqrt	平方根
fact	階乗
pow	べき乗
exp	eを底数とするべき乗
log10	常用対数
ln	自然対数

log	自然対数
random	ランダム関数
min	最小値
max	最大値
int	整数への変換
float	実数への変換
sin	サイン関数
asin	アーク・サイン関数
sinh	ハイパーボリック・サイン関数
cos	コサイン関数
acos	アーク・コサイン関数
cosh	ハイパーボリック・コサイン関数
tan	タンジェント関数
atan	アーク・タンジェント関数
atan2	アーク・タンジェント関数(2値入力)
tanh	ハイパーボリック・タンジェント関数

以上のようにしてexprオブジェクトを用いれば、複雑な演算式を記述することができる。ただし、途中の演算状態を確認できないので、込み入った複雑な式は避けた方がよいかもしれない。

● vexprオブジェクトのリスト演算式

exprオブジェクトのインレットへの入力は単一の数値だが、vexprオブジェクトではインレットへリストを入力するようになっている。vexprオブジェクトは、入力されたリストの要素ごとに演算を行い、その結果をリストとして出力する。演算式はexprオブジェクトと同じように記述すればよい。同じ数値計算を並列的に行う場合は、それらをリストとしてまとめてvexprを用いることで、簡潔なパッチになるだろう。

次のvexpr sqrt($f1)という例では、平方根を求める関数を用いている。ここで1.2.3.の3つの実数から成るリストを入力すれば、リストの要素ごとに平方根が計算される。