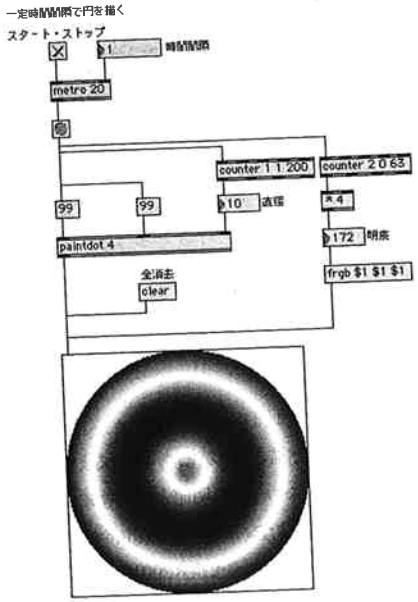
中心座標は(99,99)になる。そこで、paintdotオブジェクトの第1インレットも第2インレットも、メッセージ・ボックスを使って99を入力する。これに対して、円の大きさは200から1へと減少させるので、counter 1 1 200というオブジェクトを作り、paintdotの第3インレットに入力する。これらをmetroからの出力につなげば、lcdオブジェクトの境界線に接する大きな円から直径1の点になるまで、次第に小さくなる同心円を描く。

一方、グレースケールの色はfrgbメッセージを使って指定する。frgbは3つの整数のアーギュメントを持ち、それぞれ0から255までの範囲でRGBの各要素に指定する。例えば、color 0 0 0なら黒、color 255 0 0なら赤、color 255 255 255なら白だ。もちろん中間的な色も指定できる。ここではfrgb $1 $1 $1メッセージとして色を指定すれば、RGBの各要素が同じ数値になるので、グレースケールになる。この値は0から255までの範囲で変化すればよいが、それでは変化が緩慢であるので、counter 2 0 63とし、そのカウンターの値を＊4オブジェクトで4倍にしている。これで色の変化が明確になる。

■3-10-16 グレースケールで同心円を描画



3-11 条件分岐の処理

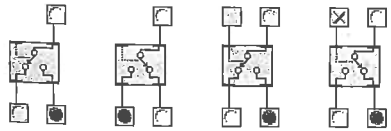
Maxではパッチ・コードをメッセージが流れることによって処理が行われるため、どのパッチ・コードにメッセージを送るかを変えれば、パッチ全体としての動作を変えることになる。例えば、同じ数値であっても、10を足すオブジェクトに数値を送るか、10を掛けるオブジェクトに数値を送るかによって、当然処理結果が違うわけだ。このような処理は一般的なプログラミング言語では条件判断処理や条件分岐処理と呼ぶが、Maxでは同じような機能を担うオブジェクトで行う。以下、その使い方を見ていこう。

Ggate、gateオブジェクトによる出力選択

メッセージの流れを制御するオブジェクトとして、視覚的に分かりやすいのはGgate《Graphic Gate》とGswitch《Graphic Switch》だろう。これらは、オブジェクト・パレットの23番目と22番目にある。

Ggateオブジェクトは、メッセージの流れを図形的に表しており、第2インレットで受け取ったメッセージを、矢印がつながっているアウトレットから出力する。メッセージはどのような種類でも構わない。このオブジェクトをクリックすれば、ちょうど電気回路図の接点切り替えのように、矢印の向きが変わる。また、矢印の向きは、第1インレットに0または1を送ることでも切り替えることができる。0であれば左側に、1であれば右側に矢印がつながる。次のようなパッチで、上側のbuttonオブジェクトをクリックして、下側のbuttonオブジェクトのどちらが点灯するかを確かめることができる。Ggateオブジェクトまたは、第1インレットにつながっているtoggleオブジェクトをクリックすることによって、矢印の向きが変わることも分かるだろう。

■3-11-1 Ggateオブジェクトの動作



多くのオブジェクトはright-to-left orderのルールに従って、第2インレットにメッセージを受け取った際は、オブジェクトの内部状態を変えるだけであり、、第1インレットにメッセージが来たときに何らかの動作を行う。

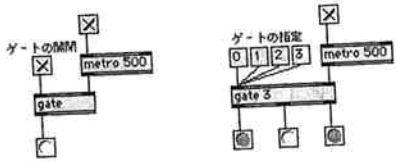
それに対して、Ggateオブジェクトは、第1インレットに0か1を受け取ったときは、矢印の向き、すなわち内部状態を変えるだけだ。そして、第2インレットにメッセージを受け取れば、矢印がつながっているアウトレットから、そのメッセージをすぐに出力する。その点で一般的なオブジェクトとは動作が異なるので、混乱しないように注意しよう。このことは、以後に説明するgateオブジェクトや、Gswitch、switchなどのオブジェクトでも同様だ。

Ggateオブジェクトは視覚的に分かりやすいものの、より実用的なのがgateオブジェクトである。gateオブジェクトは、アーギュメントでアウトレットの数を指定する。アーギュメントを省略した場合は、アウトレットの数は1となる。そして、第1インレットに入力される整数によって、どのアウトレットから出力するかを指定する。第1インレットへの整数が0であれば、どのアウトレットからも出力されない。ちょうどゲート(門)が閉じた状態になるわけだ。そして、1なら第1アウトレットから、2なら第2アウトレットから、というように出力先が切り替わる。

3-11-2の例ではmetroオブジェクトを使って0.5秒間隔でbangメッセージを出力しているが、gateオブジェクトに付けたtoggleオブジェクトによって、bangメッセージを通過させるか否かを決めることができる。そのため、toggleオブジェクトがチェックされているときは、buttonオブジェクトが点滅することになる。

またgate 3とした場合は、gateは3つのアウトレットを持ち、第1インレットへの整数によって、どのアウトレットからbangメッセージを出力するかを決めることになり、それぞれのアウトレットからつながっているbuttonオブジェクトが点滅する。

■3-11-2 gateオブジェクトの動作



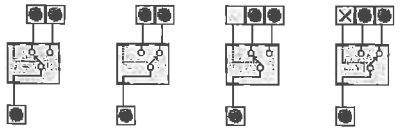
Gswitch、switchオブジェクトによる入力選択

先に解説したGgateとgateは、出力するアウトレットを選択できるオブジェクトだが、これらとは逆に、メッセージを受け取るインレットを選択できるのが、Gswitchオブジェクトとswitchオブジェクトだ。

まず、Gswitchオブジェクトはオブジェクト・パレットの22番目にある。Ggateオブジェクトと見た目が似ているので、気を付けたい。Gswitchオブジェクトの場合は、矢印がつながっているインレットからメッセージを受け取り、それをそのままアウトレットへ出力する。矢印がつながっていないインレットにメッセージが入力されても、それは無視される。Gswitchオブジェクト自体をクリックすれば、矢印の方向が切り替わり、入力を受け付けるインレットが変わる。また、第1インレットに0が送られたときは、矢印が左側になり、第2インレットからの入力を受け付けるようになる。1の場合は、矢印が右側になり、第3インレットからの入力が有効になる。

次のパッチで、上部のbuttonオブジェクトをクリックして、Gswitchオブジェクトの状態によって、どちらの入力が通過するかを確かめよう。また、Gswitchやtoggleオブジェクトをクリックして、Gswitchオブジェクトの状態が変わることも分かるはずだ。

■3-11-3 Gswitchオブジェクトの動作

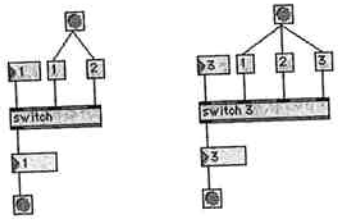


switchもメッセージの入力を受け取るインレットを選択するオブジェクトだが、第1インレットに0が入力された場合は、第2インレット以降のどのインレットからも入力を受け付けない。そして、1の場合は第2インレット、2の場合は第3インレットという具合に、入力を受け付けるインレットが変化する。有効であるインレットにメッセージが入力されると、そのメッセージを通過させてアウトレットから出力する。有効でないインレットで受け取ったメッセージは無視する。

ちなみにswitchはアーギュメントで入力を選択するインレットの数を指定する。アーギュメントが3であれば、第1インレットの他に3つのインレットが作られるので、合計4つのインレットを持つことになる。アーギュメントを省略した場合は2を指定したことになる。3-11-4のパッチで、上部のナンバー・ボックスの値によって、入力を受け付けるインレットを設定し、上部のbuttonオブジェクトをクリックしてみよう。buttonオブジェクトからbangメッセージが出力され、それぞれメッセージ・ボックスから数値メッセージがswitchオブジェクトに送られるが、有効になっているインレットに届いたメッセージだけがswitchを通過して、下部のナンバー・ボックスに表示されるはずだ。

また、その下のbuttonオブジェクトは、ナンバー・ボックスにメッセージが届いたことを点滅で知らせるために付けている。

■3-11-4 switchオブジェクトの動作



なお、すでに説明したように、第2インレット以降にメッセージを送る前に、第1インレットに整数メッセージを送って、入力を受け付けるインレットを選択しておかなければならない。

条件判断を行うオブジェクト

これまでに説明したgateやswitchオブジェクトは、toggleオブジェクトなどによってユーザーがオブジェクトの動作を設定していた。これを日常生活で考えれば、前者はエアコンのスイッチを手動で入れたり、切ったりすることに例えられるだろう。

一方、温度計によってエアコンを制御すれば、自動的に適切な室温に保つことができる。冷房の場合は、設定した温度より室温が高くなればエアコンのスイッチを入れ、設定温度より室温が低くなれば、エアコンのスイッチを切るように制御すればよい。このような一定の条件に基づく判断を"条件判断"と呼び、条件判断によって何らかの動作を行うことを"条件処理"と呼ぶ。

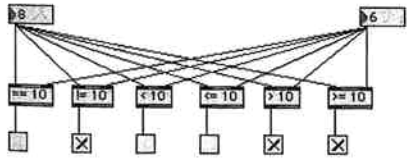
条件判断を行うためには、次のような条件演算オブジェクトが利用できる。

==	等しい
!=	等しくない
<	より小さい
<=	等しいか小さい
>	より大きい
>=	等しいか大きい

== オブジェクトの場合は、アーギュメントで指定した数値または第2インレットに入力された数値と、第1インレットに入力された数値を比べて、それらが等しければ1を、そうでなければ0を出力する。同じように、< オブジェクトなら、アーギュメントまたは第2インレットの数値よりも、第1インレットに入力された数値が小さければ1を、そうでなければ0を出力する。論理用語に置き換えると、1は"真"を、0は"偽"を表す。より正確には"0でない数値"が"真"であり、0が"偽"を表す。

次のパッチで、ナンバー・ボックスの値を変えて、それぞれの条件演算オブジェクトの出力を確かめておこう。

■3-11-5 条件判断のためのオブジェクト



2つの条件演算オブジェクトの結果から論理演算を行うためには、次のオブジェクトが用意されている。

&&	かつ(論理積、論理AND演算)
||	または(論理和、論理OR演算)

&&オブジェクトは、第2インレットから受け取った数値も、第1インレットから受け取った数値も、いずれもが"0でない数値(真)"であれば、1(真)を出力し、そうでなければ0(偽)を出力する。したがって&&の演算は次のようになる。

偽 && 偽 → 偽
偽 && 真 → 偽
真 && 偽 → 偽
真 && 真 → 真

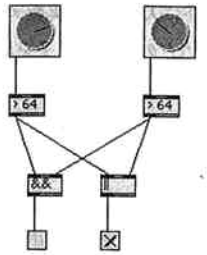
一方、||オブジェクトは、第2インレットから受け取った数値か、第1インレットから受け取った数値のいずれかが"0でない数値(真)"であれば、1"真"を出力し、そうでなければ0"偽"を出力する。したがって、||の演算は次の通りだ。

偽 || 偽 → 偽
偽 || 真 → 真
真 || 偽 → 真
真 || 真 → 真

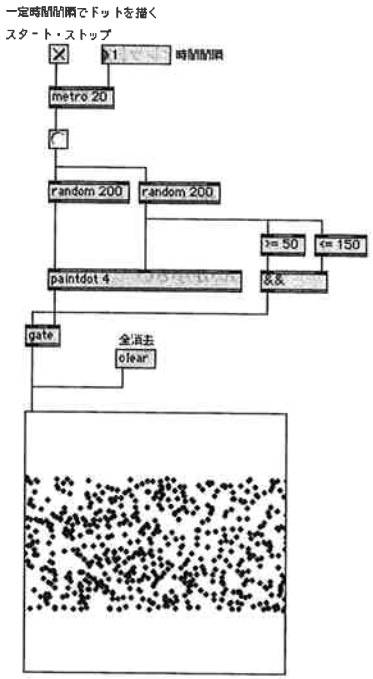
3-11-6のパッチでは、左右それぞれのdialオブジェクトの値が64より大きいか否かを調べ、それらの論理積と論理和を出力している。ダイアルを動かして、どのような結果が得られるか、toggleオブジェクトのチェック・マークで確認しよう。もちろん、right-to-left orderのルールがあるので、右のdialオブジェクトのダイアルを動かしてから、左のダイアルを動かす必要がある。

また条件判断のオブジェクトとgateオブジェクトを使って、簡単な条件処理を行うのが、次の3-117のパッチだ。ここでは、randomオブジェクトから出力されるY座標が50以上かつ150以下の場合にgateを開くようにしている。この処理により、lcdオブジェクトの上部と下部には円が描かれず、中央部だけにドットが描かれることになる。

■3-11-6 論理積と論理和を求めるオブジェクト



■3-11-7 ドットを描く座標による条件処理



● ifオブジェクトによる条件処理

==や>といった条件判断のオブジェクトでは表現しにくい条件判断は、ifオブジェクトを使って記述することができる。ifオブジェクトは、C言語などのif文に似た次のようなアーギュメントを与える。

if 条件式 then 出力1
if 条件式 then 出力1 else 出力2

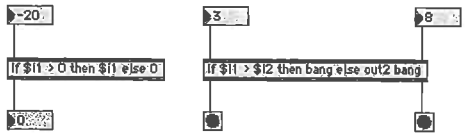
条件式は、$i1、$f1、$s1といった記号を用いて、インレットから入力されるメッセージを扱う。2文字目のiは整数(int)、fは実数(float)、sはシンボル(symbol)というメッセージのタイプを表し、3文字目の数値はインレット番号を表す。したがって、$i2なら第2インレットで受け取る整数メッセージを指し、$f3なら第3インレットに入力される実数メッセージになる。このようなインレットからのメッセージを==や>などの条件演算子を用いて条件判断を行う。次のような条件演算子を使用することができる。

==	等しい
!=	等しくない
<	より小さい
<=	等しいか小さい
>	より大きい
>=	等しいか大きい
&&	かつ(論理積、論理AND演算)
||	または(論理和、論理OR演算)

条件式を評価した結果が"真"であれば、出力1の部分で指定したメッセージがアウトレットから出力される。"偽"の場合は、elseに続いて出力2として記述したメッセージが出力される。else以降を記述していなければ、条件式の結果が"偽"の場合は、何も出力されない。また第2アウトレットから出力する場合は、out2というキーワードに続けて出力するメッセージを記述する。

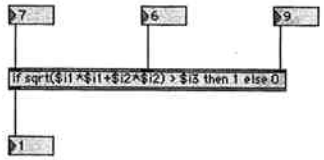
具体的な例で考えてみよう。if $i1 > 0 then $i1 else 0なら、第1インレットで受け取った整数が0よりも大きければ、その整数を出力し、そうでない場合0を出力する。これは、正の整数はそのまま、負の整数は0として出力されることになる。 if $i1 > $i2 then bang else out2 bangなら、第2インレットで受け取った整数よりも第1インレットで受け取った整数が大きければ、第1アウトレットからbangメッセージを出力する。それ以外の場合は、第2アウトレットからbangメッセージを出力する。

■3-11-8 ifオブジェクトによる条件処理



また、条件式には、exprと同じ演算子や関数を含めることができるため、かなり複雑な条件式を表すことも可能だ。3-11-9の例では、第1インレットの整数と第2インレットの整数をそれぞれ掛け合わせた合計の平方根が、第3インレットの整数よりも大きければ1を出力し、そうでなければ0を出力する。

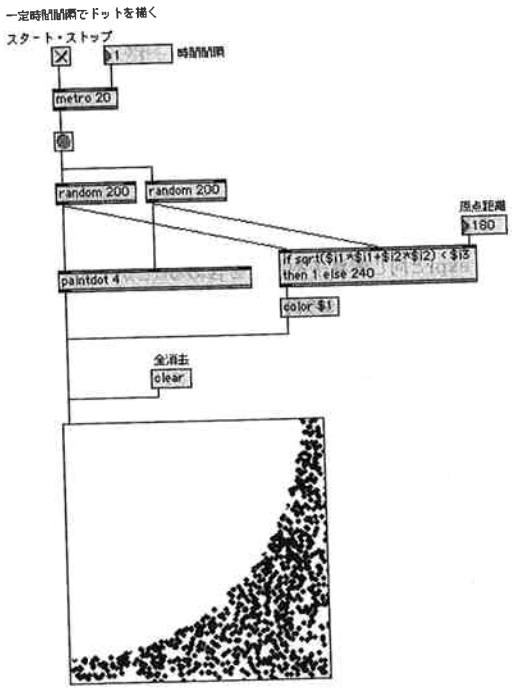
■3-11-9 ifオブジェクトの計算を伴う条件式



この条件式の$i1と$i2を座標として考えると、この条件式は原点からの距離が一定範囲内であるかどうかを判断していることになる。これを利用して、原点からの距離によって、色を変えてドットを描いてみよう。ドットはランダムな位置に描かれるが、原点からの距離が一定の値よりも小さければ、パレット番号1の色(薄い黄色)で描き、そうでなければパレット番号240の色(濃い紺色)で描くわけだ。時間が経過するにつれて、次第に大きな円の一部が現れてくるだろう。ifオブジェクトの第3インレットで指定している数値は、この大きな円の半径になる。

なお、この条件式は、$i1*$i1 + $i2*$i2 < $i3*$i3と記述することもできる。

■3-11-10　原点からの距離による描画

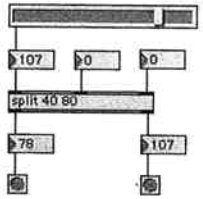


splitオブジェクトによる数値の範囲指定

splitオブジェクトを用いれば、アーギュメントで指定した範囲によって、数値を振り分けて出力することができる。アーギュメントは2つの整数によって、最小値と最大値を指定する。その範囲の整数を第1インレットに受け取った場合は、その整数を第1アウトレットから出力する。範囲外の整数であれば、第2アウトレットから出力する。

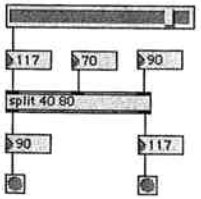
次のパッチで、hsliderオブジェクトを動かし、splitオブジェクトのどのアウトレットから整数が出力されるか、見ておこう。アーギュメントで指定した範囲は40から80までなので、この範囲の整数であれば第1アウトレットから出力される。39以下の整数や81以上の整数であれば、第2アウトレットから出力されるはずである。

■3-11-11　splitオブジェクトの動作



なお、splitオブジェクトの第2インレットに整数を入力すると、範囲指定の最小値を設定することになる。また、第3インレットに整数を送ることで、最大値を変更することができる。最大値よりも最小値を大きく指定した場合は、指定範囲に該当する整数は存在しないので、どのような整数を入力しても第2アウトレットから出力される。

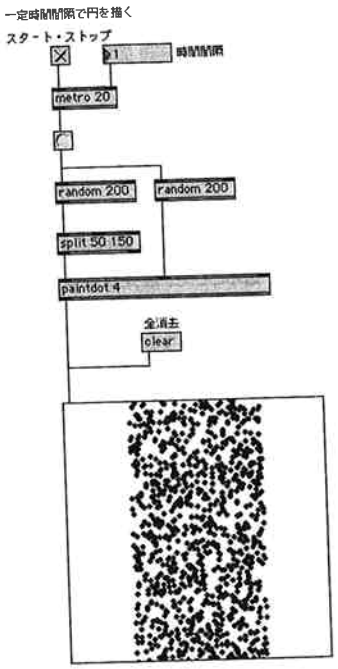
■3-11-12　splitの最小値と最大値の設定



3-11-7の例で、ランダムな位置にドットを描画する場合に、Y座標の値に応じた条件処理を行ったが、このパッチをX座標の値に応じて描画するか否かを決めるように変更してみよう。例えば、X座標が50以上150以下の場合に描画するという条件なら、splitオブジェクトを使って簡単に処理することができる。具体的には円の左端座標を出力するrandomオブジェクトの次にsplit 50 150オブジェクトを追加するだけでよい。

その結果50以上150以下の整数だけがsplitオブジェクトを通過し、ドットが描かれるのである。もちろん範囲外の数値の場合はsplitオブジェクトを通過しないので、ドットは描かれない。これは、packオブジェクトが第1インレットにメッセージを受け取ったときに出力を行うright-to-left orderの性質を利用している。

■3-11-13 ドットを描く座標による条件処理



なお、指定範囲外の数値を範囲内に収まるように処理したい場合は、splitオブジェクトよりもナンバー・ボックスを用いる方が簡単だ。例えば、受け取った数値が0から127までならそのまま出力、0未満の数値は0とし、127より大きな数値は127として出力する場合は、ナンバー・ボックスのインスペクターを開き、最小値を0に、最大値を127と設定すればよい。

❸ selectオブジェクトによるメッセージ選択

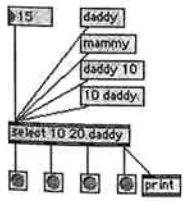
selectは、入力されたメッセージが、アーギュメントに一致するかどうかを調べるオブジェクトだ。一致した場合は、アーギュメントの順番に対応したアウトレットからbangメッセージを出力する。どのアーギュメントにも一致しない場合は、一番右のアウトレットから受け取ったメッセージをそのまま出力する。したがって、アーギュメントの個数よりも1つ大きい数のアウトレットを持つことになる。なお、selectオブジェクトはselと短縮しても構わない。

3-11-14の例では、select 10 20 daddyと3つのアーギュメントがあるので4つのアウトレットが作られている。ナンバー・ボックスをドラッグして数値をselectオブジェクトに送ると、数値が10のときは1番目のアーギュメントに一致するので第1アウトレットからbangメッセージが出力され、その下のbuttonオブジェクトが点滅する。20のときは、第2アウトレットにつながるbuttonオブジェクトが点滅する。それ以外の数値は一致するアーギュメントがないので、最後のアウトレットから数値が出力され、buttonオブジェクトが点滅するとともに、printオブジェクトによってMaxウィンドウに表示されることになる。

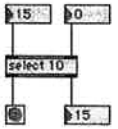
また、アーギュメントにはシンボルも指定できるので、daddyというメッセージ・ボックスをクリックすれば、3番目のbuttonオブジェクトが点滅する。mammyの場合は一致しないので、最後のアウトレットから出力される。複数の要素からなるメッセージの場合は、先頭の要素について一致を調べることになる。例えばdaddy 10なら3番目、10 daddyなら1番目のアウトレットからbangメッセージが出力されるわけだ。

なお、アーギュメントに整数を1つだけ指定した場合は、自動的に第2インレットが作られる。第2インレットに整数を送ることで、アーギュメントを変更することができ、一致を調べる整数を設定することになる。

■3-11-14 selectオブジェクトの動作

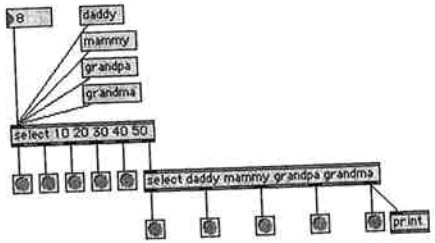


■3-11-15 第2インレットによる選択する整数の設定



アーギュメントに一致しないメッセージは最後のアウトレットから出力されることを利用して、複数のselectオブジェクトを直列的につなぐ処理も考えられる。一致を調べるメッセージの数が多い場合や、メッセージの種類ごとに処理を行いたい場合は、この方法を用いると分かりやすいパッチを作ることができる。次の例では、2つのselectオブジェクトを使って、数値とシンボルとを異なるselectで判断している。1つのselectオブジェクトにアーギュメントをまとめてもよいが、この方が便利なことが多いだろう。

■3-11-16　2つのselectオブジェクトによる処理



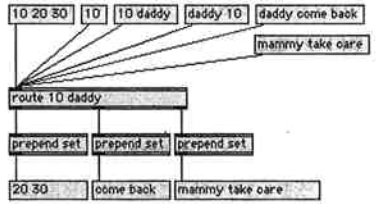
routeオブジェクトによるメッセージ分配

routeオブジェクトは、selectオブジェクトと似ており、アーギュメントと受け取ったメッセージの一致を調べる。ただし、複数の要素からなるメッセージの受け取った場合は、アーギュメントとメッセージの最初の要素との一致を調べ、一致した場合は2番目以降の要素を出力するところが異なる。メッセージの要素が1つの場合は、bangメッセージが出力されるので、この場合はselectオブジェクトと同じ動作になる。

3-11-17の例では、routeオブジェクトに10とdaddyというアーギュメントを与えている。ここで10 20 30というメッセージを送ると、1番目のアーギュメントに一致するので、第1アウトレットから20 30というメッセージが出力される。次のprependオブジェクトは、受け取ったメッセージの先頭にアーギュメントを付加して出力する。ここではアーギュメントはsetなので、set 20 30というメッセージとなって出力されるわけだ。メッセージ・ボックスはsetメッセージを受け取ると、それ以降の内容を表示するので、メッセージ・ボックスには20 30と表示される。つまり、routeオブジェクトが出力したメッセージがメッセージ・ボックスに表示されると考えればよい。

一方、daddy 10ならrouteの2番目のアーギュメントに一致するので、2番目のメッセージ・ボックスに10が表示される。その他のメッセージについても、どのように処理されるかを確認しておこう。

■3-11-17　routeオブジェクトの動作



selectオブジェクトと同じように、routeオブジェクトはアーギュメントの数よりも1つ大きい数のアウトレットを持ち、アーギュメントに一致しないメッセージは、最後のアウトレットから出力される。これを利用して、複数のrouteオブジェクトをつないで、処理を分けることができる。

■3-11-18　2つのrouteオブジェクトによる処理