情報メディア基盤ユニット用資料（2015 年 6 月 9 日分）

Processing 言語による情報メディア入門

関数 (その 2)

神奈川工科大学情報メディア学科　佐藤尚

組み込み関数

今までにも、いくつか使ってきましたが、Processing では沢山の関数が用意されています。その中でよく使いそうなものを以下に挙げておきます。ここで紹介する関数は、呼び出すと何らかの値を求めて、その値を返すものです。この返される値のことを戻り値と呼んでいます。また、値を返す関数を呼び出すと、呼び出された関数がその戻り値に置き換わるような動作となります。

プログラミング言語において、事前に定義されている関数を**組み込み関数** (built-in function) と呼ぶことがあります。


```
int m = 60*hour() + minute();
```

現在の時刻が18時24分なら、hour()は18に、minute()は24に置き換わり、変数mには60*18+24=1104が代入される。

図 8-1 関数を呼び出すと

関数を実行する目的で、プログラム中に関数を置くことを**関数を呼び出す**と呼ぶことがあります。

これ以外にも沢山の組み込み関数が用意されています。気になるひとは、リファレンスマニュアルを見て下さい。

時間に関連した関数には以下のようなものがあります。これらは、パソコンの時計に連動して、情報を求めています。

表 8-1 時間関連の関数

関数名	関数が返す値の意味
year()	現在の年を返す。
month()	現在の月（1 ～ 12）を返す。
day()	現在の日（1 ～ 31）を返す。
hour()	現在の時刻の時間を返す。
minute()	現在の時刻の分を返す。
second()	現在の時刻の秒を返す。
millis()	プログラムを実行してから経過時間をミリ秒単位で返す。

関数と言うと数学で出てくるものを思い浮かべると思います。Processing では、数学で出てくるような関数が用意されています。まずは、数の大きさに係わる関数です。

表 8-2 最小、最大関連の関数

関数名	関数が返す値の意味
min(x1,x2)	x1 と x2 の中で小さい方の値（最小値）を求める。
min(x1,x2,x3)	x1,x2,x3 の中で最小値を求める。
max(x1,x2)	x1 と x2 の中で大きい方の値（最大値）を求める。
max(x1,x2,x3)	x1,x2,x3 の中で最大値を求める。

これらの関数には、別な使い方もあります。それは次回に紹介します。min は minimum、max は maximum を省略したものです。

もう少し数学っぽい関数もあります。

表 8-3 ちょっと数学っぽい関数

関数名	関数が返す値の意味
abs(x)	引数 x の絶対値を求めます。例えば、abs(-1.1) は 1.1、aba(3) は 3 になります。
sqrt(x)	引数 x の平方根の値を求めます。る。例えば、sqrt(4) なら 2.0 になります
sq(x)	引数 x の二乗を求めます。
pow(x,n)	x の n 乗を求めます。例えば、pow(2,4) は 16 になります。
exp(x)	指数関数の値を求めます。ネイピア数 e の x 乗を求めます。
log(x)	自然対数の値を求めます。
dist(x1, y1, x2, y2)	2 点 (x1,y1) と (x2,y2) の間の距離を求めます。
constrain(v, m0, m1)	引数 v の値が m0 以上 m1 以下なら v を返し、v の値が m0 よりも小さければ m0 を返し、v の値が m1 よりも大きければ m1 を返すような関数です。
lerp(v0,v1,t)	(1-t)*v0+t*v1 という値を求めます。線形補間と呼ばれる計算方法です。
map(v, low1, high1, low2, high2)	2 点 (low1,low2) と (high1,high2) を通る直線において、X 座標の値が v の時の Y 座標の値を求める関数です。別な言い方をすると low1 以上 high1 以下の値 v を low2 以上 high2 以下の値に変換するとどんな値になるかを求めるものです。要するに一次関数の値を計算しています。 $y = \frac{high2 - low2}{high1 - low1}(v - low1) + low2$

この 2 つの関数は数 III をやっていないと出てこないですね。

このように、言葉で説明するようも、式で説明する方が簡単になる場合もあります。この map 関数は意外に使い機会の多い関数です。

でもやっぱり、関数と言うと三角関数のような気がします。Processing でも三角関数が用意されています。

表 8-4 三角関数関連

関数名	関数が返す値の意味
sin(x)	正弦関数 sin の値を求めます。
cos(x)	余弦関数 cos の値を求めます。
tan(x)	正接関数 tan の値を求めます。
degrees(x)	ラジアンから度に値を変換します。
radians(x)	度からラジアンに値を変換します。
asin(x)	sin の逆関数の値を求めます。つまり、sin y = x となる y の値を求めます。ただし、y の値は -PI/2 から PI/2 となります。

sin 関数の逆関数のことを arcsin と呼ぶことがあります。そこで、asin、acos、atan という名称になっています。

関数名	関数が返す値の意味
acos(x)	cos の逆関数の値を求めます。つまり、cos y = x となる y の値を求めます。ただし、y の値は -PI/2 から PI/2 となります。
atan(x)	tan の逆関数の値を求めます。つまり、tan y = x となる y の値を求めます。ただし、y の値は -PI/2 から PI/2 となります。
atan2(y,x)	原点と点 (x,y) を通る直線と X 軸のなす角度をもとめます。ただし、角度の値は -PI から PI の範囲の値となります。

引数の順番が直感的なものと逆になっているので、注意して下さい。良く使う機会のある関数です。

全然サンプルがないのも何なので、少し載せておきます。まずは、map 関数を使ったものです。これは、マウスカーソルの動きに合わせて、真ん中にある円を動かすものです。円はある範囲 (x0 ～ x1) の間しか動きません。このような動作を map 関数を使って作りだしています。つまり、mouseX の値を x0 から x1 の値に変換し、その値を円の中心の X 座標値として使っています。

map 関数の使用例 サンプル 8-1

```
int x0,x1;

void setup(){
  size(400,200);
  smooth();
  x0 = 80;
  x1 = width-x0;
}

void draw(){
  background(255);
  strokeWeight(3);
  stroke(0);
  line(x0,height/2,x1,height/2);
  fill(100);
 // mouseX の値を x0 ～ x1 の間の値に変換
  float x = map(mouseX,0,width-1,x0,x1);
  strokeWeight(1);
  ellipse(x,height/2,20,20);
}
```

次は、atan2 を使ったサンプルです。原点とマウスカーソルの位置を結ぶ直線と X 軸のなす角を弧で示すようなサンプルです。ついでに、その角度の値を degree 関数を使って、度単位で表示しています。なの、弧の部分は arc 関数を使って描画しています。arc 関数では、弧がスタートする時の角度と終了する時の角度を指定する必要があります。また、原点との距離を計算して、3 分の 1 の位置に弧を表

示するようにしています。

atan2,dist などの関数の使用例 サンプル 8-2

```
PFont font;

void setup(){
  size(400,400);
  smooth();
  font = loadFont("Serif-48.vlw");
  textFont(font);
}

void draw(){
  background(255);
  stroke(0);
  line(0,0,mouseX,mouseY);
  noFill();
   float theta = atan2(mouseY,mouseX);// 線分と X 軸のなす角度を求める
  float l = dist(0,0,mouseX,mouseY);// 原点との距離を求める
  arc(0,0,2*l/3,2*l/3,0,theta);
  line(0,0,mouseX,mouseY);
  String deg = str(degrees(theta));
  fill(255,10,10);
  text(deg,width/2-textWidth(deg)/2,height/2);
}
```

これらのサンプルのように、色々な関数を組み合わせることでどんどん複雑なプログラムを作ることが出来るようになります。

関数の宣言（その 2）

Processing が用意している関数について説明してきました。今回説明した関数は、何らかの値（戻り値）を返すような関数でした。前回の講義では、処理をまとめるという観点から関数の説明をしました。そのため、値を返すという話はありませんでした。今回説明したような値を返すような関数を定義することも出来ます。

前回説明した値を返さない関数も void という特別なデータ型の値を返していると見なすことも出来ます。

そのためには、表 8-5 のような形でプログラムを書きます。値を返す必要があるために、戻り値のデータ型を指定する必要があります。関数定義の中で、戻り値を決定する（どんな値を返すのか）必要があります。そのために、関数名の前に戻り値のデータ型を置きます。戻り値を指定するために、return 命令を使います。「return　式;」とすると、この式の値が関数の戻り値となります。また、return 命令を実行すると、その場所で関数の実行が終わります。関数の定義中に、複数の return 命令があっても、問題はありません。逆にどこにも return 命令がないと、Processing はどんな値を戻り値とすればよいのか、わからないので、エラーとなります。関数の定義は、プログラム中のどこからでも始めることが出来ます。ただし、他の関数の定義中などでは出来ません。

1 つの関数内に複数の return 命令を置くことは、良くないと考える人たちもいます（いた？）。会社によっては、複数の return 命令を置くことを禁止してるところもあります。このような、プログラム作成上で決めた制限（規則）を、コーディング規約と呼ぶことがあります。

表 8-5 関数定義の仕方（その 3）

関数定義のパターン

戻り値のデータ型 関数名 (){
関数処理の内容を書きます。
どこかに、return 命令が必要です。
変数なども使うことができます。
}

戻り値のデータ型 関数名 (データ型名　引数名){
関数処理の内容を書きます。
どこかに、return 命令が必要です。
変数なども使うことができます。
}

戻り値のデータ型 関数名 (データ型名 1　引数名 1,
データ型名 2　引数名 2…){
関数処理の内容を書きます。
どこかに、return 命令が必要です。
変数なども使うことができます。
}

サンプル 8-3 では、” 年 / 月 / 日 ” の形式で、今日の日付を返す関数 today を定義しています。

戻り値を持った関数定義の例その 1 サンプル 8-3

```
PFont font;

// 今日の日付を返す関数 today を定義、戻り値は String 型
String today(){
  String msg = year()+"/"+month()+"/"+day();
  return msg; // 戻り値は msg
}

void setup(){
  size(300,200);
  smooth();
  font = loadFont("Serif-48.vlw");
  textFont(font);
}

void draw(){
  background(255);
  fill(0);
  String msg = today(); // 自分で定義した関数は自由に使うことが出来る。
  text(msg,width/2-textWidth(msg)/2,height/2);
}
```

サンプル 8-4 に関数定義の部分だけの部分の例を示します。

return 命令がないと、「This method must return a result of type データ型名」というエラーメッセージが表示されます。

関数の定義は、どこかのブロックに属しているところでは出来ません。

戻り値を持った関数定義の例その 2 サンプル 8-4

```
float myConstraint1(float v,float m0,float m1){
  flaot ans;
  ans = v;
  if(v > m1){
    ans = m1;
  }else if(v < m0){
    ans = m0;
  }
  return ans;
}

float myConstraint2(float v,float m0,float m1){
  if(v > m1){
    return m1;
  }else if(v < m0){
    return m0;
  }else{
    return v;
  }
}

float myDist(float x0,float y0,float x1,float y1){
  return sqrt(sq(x0-x1)+sq(y0-y1));
}
```

この場合は日本語の説明より、プログラムの方がわかり易いでよね。

dist 関数は三平方の定理を使うと、自分で作ることも出来ます。自分のプログラムの定義中に他の関数を利用することも出来ます。

動作しないサンプルでは、面白くないので、サンプル 8-3 を改良して、今日の日付を " 月 / 日 / 年 " の形で表示することにします。この際に、月は英語表記の略称とします。今回のサンプルでは if 命令の山になるので、ちょっとプログラムは長くなります。

戻り値を持った関数定義の例その 3 サンプル 8-5

```
PFont font;

void setup(){
  size(300,200);
  smooth();
  font = loadFont("Serif-48.vlw");
  textFont(font);
}

void draw(){
  background(255);
  fill(0);
   String msg = today();// 自分で定義した関数は自由に使うことが出来る。
  text(msg,width/2-textWidth(msg)/2,height/2);
}
```

```
// 今日の日付を返す関数 today を定義、戻り値は String 型
String today(){
  int m = month();
  String result = "/"+day()+"/"+year(); // 後ろの部分は簡単に作れる
  // 月の値で分岐する
  if(m == 1){
    result = "Jan"+result;
  }else if(m == 2){
    result = "Feb"+result;
  }else if(m == 3){
    result = "Mar"+result;
  }else if(m == 4){
    result = "Apr"+result;
  }else if(m == 5){
    result = "May"+result;
  }else if(m == 6){
    result = "Jun"+result;
  }else if(m == 7){
    result = "Jul"+result;
  }else if(m == 8){
    result = "Aug"+result;
  }else if(m == 9){
    result = "Sep"+result;
  }else if(m == 10){
    result = "Oct"+result;
  }else if(m == 11){
    result = "Nov"+result;
  }else if(m == 12){
    result = "Dec"+result;
  }else{
    result = "Unknow"+result;
  }
  return result; // 戻り値は result
}
```

一般的に、人為的に決めた規則に合うようにデータを変換することはちょっと面倒です。

10 月なのに October とか、12 月なのに December とかちょっと変に思いませんか？

関数を利用してプログラムを書くことにより、修正部分を一部にとどめることが出来ます。サンプル 8-5 でも、関数 today の定義部分を変更しただけですよね。

定義を書いている場所が少し移動していますが。

このように関数を利用してプログラムを作成すると、わかりやすく、変更しやすいプログラムを作成することが出来ます。単に関数を使えばわかりやすいプログラムが作れるわけではありません。上手い関数名や変数名をつけたり、複雑な処理をわかりやすい関数の組み合わせに分解するなど､色々なことが重要になります。ですから、ゲームのような複雑なプログラムを作るためには、様々な力をもった人が必要となります。

変更しやすいプログラムを作成することはとても重要です。ゲームの仕様が少し変わっただけで、プログラムを全て作り直していたら、ゲームは完成しませんよね。

サンプル 8-6 では、ウインドウ中心に表示されている円にマウスカーソルが来ると、円の色を変えるものです。ある点が円の中に入っているかどうかを、inDisk 関数を定義して、判定しています。入っ

ているかどうかをあらわすので、戻り値は boolean 型とするのが自然です。inDisk 関数は、ある点が円の中に入っているかどうを、円の中心とその点の距離を調べることで判定しています。つまり、点と円の中心の距離が半径以下なら円の中に入っています。dist 関数は 2 点の距離を求める組み込み関数です。これを使って、円の中心と点との距離を求めることができます。そして、この値と半径の値 r と比較することで、判定を行っています。

円の内外判定

戻り値を持った関数定義の例その 4 サンプル 8-6

```
int radius = 150;

void setup(){
  size(400,400);
  smooth();
}

void draw(){
  background(255);
  noStroke();
  if(inDisk(mouseX,mouseY,width/2,height/2,radius)){
    fill(255,10,10);
  }else{
    fill(10,10,255);
  }
  ellipse(width/2,height/2,2*radius,2*radius);
}

/*
inDisk 関数は点 (x,y) が中心座標が (cx,cy) で半径の r の円の中に入っているかどうかを判定します。
*/
boolean inDisk(float x,float y,float cx,float cy,float r){
  float d = dist(x,y,cx,cy);
  if(d <= r){
    return true;
  }else{
    return false;
  }
}
```

サンプル 8-6 の inDisk 関数は、次の様に書くことも出来ます。

戻り値を持った関数定義の例その 5 サンプル 8-6'

```
boolean inDisk(float x,float y,float cx,float cy,float r){
  float d = dist(x,y,cx,cy);
  return (d <= r);
}
```

コールバック関数

今までのように、mousePressed 変数などだけを使って、少し複雑なマウス操作を伴ったプログラムを作成することは、困難です。そこで、コールバック関数と言う仕組みが用意されています。

つまり、マウスなどが指定された動作（イベントと呼びます）が行われた時に、呼び出す関数を決めておき、その関数内でイベントに対応する処理を定義します。Processing 言語では、以下のようなコールバック関数が用意されています。当然、処理の中身はユーザが定義します。

イベントの処理を行うということで、コールバック関数のことをイベントハンドラと呼ぶこともあります。今まで使ってきた、setup 関数や draw 関数もコールバック関数です。setup は起動時というイベントにより呼び出される関数、draw は一定時間が経過したいうイベントで呼び出される関数です。

表 8-6 コールバック関数

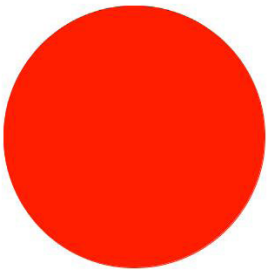
呼び出すイベント	コールバック関数名	補足
マウスボタンが押された	mousePressed()	この関数内で、mouseButton 変数の値が、LEFT なら左、CENTER なら真ん中、RIGHT なら右ボタンが押されています。
マウスボタンが離れた	mouseReleased()	
マウスボタンを押さない状態でマウスが動かされた	mouseMoved()	
マウスがドラッグされた	mouseDragged()	マウスボタンを押した状態で、マウスを移動させる動作です。この関数内で、mouseButton 変数の値が、LEFT なら左、CENTER なら真ん中、RIGHT なら右ボタンが押されています。
マウスがクリックされた	mouseClicked()	マウスをクリックするためには、マウスボタンを押して、離すという動作が必要なので、この関数が動作する前に、コールバック関数 mousePressed と mouseReleased が実行されます。
キーボードが押された	keyPressed()	システム変数 key にどのキーが押されたかの情報が保存されています。なお、矢印キーなどを押した場合には、システム変数 key には CODED という特別な値が保存され、どのキーが押されたかの情報はシステム変数 keyCode に保存されます。
キーボードが離された	keyReleased()	
キーボードが押された	keyTyped()	keyPressed 関数と異なり、1 回だけ呼び出されます。

システム変数 key には、押したキーの ASCII コードの値が保存されています。この方法では、日本語の入力が出来ません。また、矢印キーの処理には、2 段階の処理が必要となります。

これらのコールバック関数を利用したサンプルを示します。サンプル 8-7 は mouseClicked 関数を利用したものです。円の内部でマウスをクリックすると、円の描画色をランダムに変更するものです。描画色を color 型の fColor 変数に保存しておきます。マウスがクリックされた際のマウスカーソルの位置をしらべ、それが円の中であれば、fColor 変数の値を変更しています。サンプル 8-6 で作成した関数 inDisk を利用しています。

コールバック関数の利用例その 1 サンプル 8-7

```
color fColor;
void setup(){
  size(400,400);
  smooth();
  colorMode(HSB,359,99,99);
  fColor = color(random(360),99,99);
}
// サンプル 8-6' のものをそのまま利用
boolean inDisk(float x,float y,float cx,float cy,float r){
  float d = dist(x,y,cx,cy);
  return (d <= r);
}
void draw(){
  background(0,0,99);
  fill(fColor);
  stroke(fColor);
  ellipse(width/2,height/2,2*150,2*150);
}
// マウスがクリックされた際のコールバック関数
void mouseClicked(){
  if(inDisk(mouseX,mouseY,width/2,height/2,150)){
    fColor = color(random(360),99,99);
  }
}
```



サンプル 8-8 は、マウスボタンを押すとウインドウが黒くなり、マウスボタンを離すと徐々に色が白になるようなものです。描画色は変数 gray を使用して決めています。マウスボタンが押されると gray の値を 0 とし、マウスを動かすことにより、徐々に gray の値を大きくしていきます。ただし、255 より大きな値とすることができないので、constrain 関数を使って、255 よりも大きな値とならないようにしています。

コールバック関数の利用例その 2 サンプル 8-8

```
float gray=128;
void setup() {
  size(200, 200);
  smooth();
}
```

```
void draw() {
  stroke(gray);
  fill(gray);
  rect(0, 0, width, height);
}
// マウスを押したときの処理
void mousePressed() {
  gray = 0;
}
// マウスを移動させたときの処理
void mouseMoved() {
  gray = constrain(gray+1, 0, 255);
}
```

サンプル 8-9 は、マウスの移動とドラッグを組み合わせたサンプルです。

コールバック関数の利用例その 3 サンプル 8-9

```
boolean mustDraw = false;
color WHITE;
float diam=10;
void setup() {
  size(400, 400);
  smooth();
  colorMode(HSB,359,99,99);
  WHITE = color(0,0,99);
}
void draw() {
  fadeTo(WHITE);
  if(mustDraw){
    fill(random(360),99,99,150);
    float x = mouseX+random(-diam,diam);
    float y = mouseY+random(-diam,diam);
    ellipse(x,y,diam,diam);
    mustDraw = false;
  }
}
void fadeTo(color c){
  stroke(c,20);
  fill(c,20);
  rectMode(CORNER);
  rect(0,0,width,height);
}
void mouseMoved(){
  mustDraw = true;
  diam = random(10,20);
}
void mouseDragged(){
  mustDraw = true;
  diam = random(40,80);
}
```

プログラム中の fadeTo 関数は、ウインドウ全体を指定した色にフェードさせる関数です。色に不透明度の情報を付加して、実現しています。マウスをドラッグしているときには、少し大きな円を描画し、単にマウスを動かしている時には、小さな円を描画しています。boolean 型変数 mustDraw によって、円を描画する必要があるかどうかを判定しています。

サンプル 8-10 は、マウスのドラッグによる、物体の移動の例です。mouseDragged 関数は、マウスボタンを押しながらマウスを移動させると呼び出される関数です。一つ前のマウスの位置は pmouseX と pmouseY 変数に保存されています。つまり、mouseX-pmouseX の値は X 軸方向の移動距離を表してます。同様に、mouseY-pmouseY の値は Y 軸方向の移動距離を表しています。そこで、この 2 つの値を物体の位置に加えることにより、ドラッグ時の物体移動を再現できます。物体をつまんで動かしているような動作とするために、物体上でクリックした場合のみ移動するようになっています。

このサンプルでは、物体は円となっています。ですので、以前に作った inDisk 関数を利用しています。

コールバック関数の利用例その 4 サンプル 8-10

```
color fColor;
float diam=40;
float xBall,yBall;
void setup() {
  size(400, 400);
  smooth();
  colorMode(HSB,359,99,99);
  fColor = color(random(360),99,99);
  xBall = random(width);
  yBall = random(height);
}
void draw() {
  background(0,0,99);
  stroke(fColor);
  fill(fColor);
  ellipse(xBall,yBall,diam,diam);
}
void mouseClicked(){
  fColor = color(random(360),99,99);
  xBall = random(width);
  yBall = random(height);
}
void mouseDragged(){
  if(inDisk(mouseX,mouseY,xBall,yBall,diam/2)){
    xBall += (mouseX-pmouseX);
    yBall += (mouseY-pmouseY);
  }
}
boolean inDisk(float x,float y,float cx,float cy,float r){
  return (dist(x,y,cx,cy) <= r);
}
```

このプログラムには、一つ欠点があります。それは、マウスを早く動かすと、物体がついてこないことです。つまり、これを解決したものがサンプル 8-11 です。このサンプルでは、物体が移動中かどうかを示す boolean 型変数 moving を使っています。

1 つ前の状態からのマウスの移動量が円の半径より大きくなると、この現象が発生します。

コールバック関数の利用例その 4 サンプル 8-11

```
color fColor;
float diam=40;
float xBall,yBall;
boolean moving = false;
void setup() {
  size(400, 400);
  smooth();
  colorMode(HSB,359,99,99);
  fColor = color(random(360),99,99);
  xBall = random(width);
  yBall = random(height);
}
void draw() {
  background(0,0,99);
  stroke(fColor);
  fill(fColor);
  ellipse(xBall,yBall,diam,diam);
}
void mouseClicked(){
  fColor = color(random(360),99,99);
  xBall = random(width);
  yBall = random(height);
}
void mousePressed(){
   if(inDisk(mouseX,mouseY,xBall,yBall,diam/2)){
    moving = true;
  }
}
void mouseReleased(){
  moving = false;
}
void mouseDragged(){
  if(moving){
    xBall += (mouseX-pmouseX);
    yBall += (mouseY-pmouseY);
  }
}
boolean inDisk(float x,float y,float cx,float cy,float r){
  return (dist(x,y,cx,cy) <= r);
}
```

次にキーボードを使用したサンプルを示します。

コールバック関数の利用例その 5 サンプル 8-12

```
float xBall;

void setup(){
  size(400,200);
  smooth();
  xBall = width/2;
}

void draw(){
  background(255);
  stroke(0);
  fill(128);
  ellipse(xBall,height/2,30,30);
}

void keyPressed(){
  if(key == 'r'){
    xBall = constrain(xBall+random(2,4),0,width-1);
  }else if(key == 'l'){
    xBall = constrain(xBall-random(2,4),0,width-1);
  }else if(key == 'c'){
    xBall = width/2;
  }else if(key == CODED){
    if(keyCode == LEFT){
      xBall = constrain(xBall-1,0,width-1);
    }else if(keyCode == RIGHT){
      xBall = constrain(xBall+1,0,width-1);
    }
  }
}
```

r キーが押されたかどうかは、key == 'r' でわかります。通常は、key == ' 調べたいキー ' とすれば、指定したキーが押されたかどうかがわかります。直接 ASCII コードの値を書いてもかまいません。

システム変数 keyCoded は、以下のようなキーに対応しています。これ以外の BACKSPACE, TAB, ENTER, RETURN, ESC, DELETE キーは、通常のキーのように処理されます。つまり、ASCII コードで表されています。Processing では、この 6 つのキーの値は、BACKSPACE, TAB, ENTER, RETURN, ESC, DELETE という定数で定義されています。

Windows では ENTER キーが利用されますが、Mac では RETERN キーが利用されます。状況によっては、プログラムする際に注意が必要です。

表 8-7 keyCode が対応指定しているキー

キーの名称	keyCode の値	キーの名称	keyCode の値
上カーソルキー	UP	左カーソルキー	LEFT
下カーソルキー	DOWN	右カーソルキー	RIGHT
ALT キー	ALT	シフトキー	SHIFT
コントロールキー	CONTROL		

単純に keyPressed 関数を使った場合には、複数のキーが押されたかの処理ができません。しかし、少しプログラムを書くことで実現することができます。

最後にカーソルキーを利用して、円形の物体を動かすようなサンプルを示します。このサンプルにおける変数 x,y は円の中心座標を表しており、変数 vx,vy は物体の移動を表す速度ベクトルです。vx=vx=0 の時には、物体は移動しません。カーソルキーが押されたときに、コー

ルバック関数 keyPressed 内において適切な値を変数 vx,vy に設定します。カーソルキーが離されたときには、変数 vx,vy の値を 0 にすることで物体の移動を止めます。なお、キーボード上の C のキーが押されたときには、物体の位置をウインドウの中心に移動させます。

コールバック関数の利用例その 6 サンプル 8-13

```
float xBall,yBall; // 円の中心位置の座標
float vx,vy;       // 円の速度ベクトル
void setup(){
  size(600,600);
  smooth();
  xBall = width/2;
  yBall = height/2;
  vx = vy = 0; // この時には円は動かない
}
void draw(){
  background(255);
  fill(255,10,10);
  ellipse(x,y,20,20);
  xBall += vx;   // 速度ベクトルを加えることで、円を移動させる。
  yBall += vy;
}
void keyPressed(){
  if(key == 'c' || key == 'C'){
    xBall = width/2;
    yBall = height/2;
  }else if(key == CODED){
    if(keyCode == LEFT){
      vx = -1;
      vy = 0;
    }else if(keyCode == RIGHT){
      vx = 1;
      vy = 0;
    }else if(keyCode == UP){
      vy = -1;
      vx = 0;
    }else if(keyCode == DOWN){
      vy = 1;
      vx = 0;
    }
  }
}
```

```
void keyReleased(){
  if(key == CODED){
    if(keyCode == LEFT || keyCode == RIGHT ||
       keyCode == UP || keyCode == DOWN){
      vx = vy = 0;
    }
  }
}
```