Brief Intro to Arrays

CS 110

Arrays

- A special kind of variable that holds not one, by many data items of a given type.
- Declared like variables, only type is followed by a pair of brackets.

```
float[] sx;
```

- Can be initialized using a special syntax involving the `new` keyword, the type, and a *size* in brackets.

```
int[] diameter = new int[10]; // Ten diameters
```

Arrays

- Individual data items are accessed with an index and square brackets.
    - **Indexes start at 0!**

- The length of an array can be determined using its `length` property.
    - The length of an array is one greater than the last valid index.

- Arrays can be passed to, and returned from functions.

Built-in Array Functions

append(*array, item*)

- returns a new array expanded by one and add item to end

expand(*array, newSize*)

- returns a new array with size increased to newSize

shorten(*array*)

- returns a new array shortened by one

concat(*array1, array2*)

- returns a new array that is the concatenation of array1 and array2

subset(*array, offset [, length]*)

- returns a subset of array starting at offset and proceeding for length (or end)

splice(*array, value|array2, index*) or

- returns a new array with value or array2 inserted at index

sort(*array*)

- returns a new array sorted numerically or alphabetically

reverse(*array*)

- returns a new array with all elements reversed in order

```
// arrays1
String[] names = new String[5];

void setup() {
  size(500, 500);
  background(200);
  names[0] = "Chococat";
  names[1] = "Cinnamoroll";
  names[2] = "Landry";
  names[3] = "Pekkle";
  names[4] = "Purin";
}

void draw() {
  fill(0);
  int n = int( random(-0.5, names.length) );
  float x = random(0,width);
  float y = random(0, height);
  text( names[n], x, y );
}

void mousePressed() {
  names = shorten(names);
  background(200);
}
```

Example: Functions + Arrays + Loops

```
// gaussian
float[] bins = new float[500];

void setup() {
  size(500, 500);
  stroke(255);
  frameRate(15);
}

void draw() {

  for (int i=0; i<50; i++) {        // Add 50 new numbers
    int idx = int( map( gaussian(), -2.0, 2.0, 0, width ));
    bins[idx]++;
  }

  background(0);                    // Redraw all
  for (int i=0; i<bins.length; i++) {
    line(i, height, i, height-bins[i]);
  }
}

// Gaussian distribution sampler
float gaussian() {
  // Polar form of Box-Muller transformation
  // Converts uniform in [-1,1] to gaussian w/ mean=1.0, sd=1.0
  float x1, x2, w, y1, y2;

  while (true) {
    x1 = random(-1.0, 1.0);
    x2 = random(-1.0, 1.0);
    w = x1 * x1 + x2 * x2;
    if (w < 1.0) break;    // Try again if w >= 1.0
  }

  // Generate two samples
  w = sqrt( (-2.0 * log( w ) ) / w );
  y1 = x1 * w;
  y2 = x2 * w;

  return y1;
}
```