Introduction to Programming

Lecture 2: functions, flow control and more Processing

Ben Ruijl

Nikhef Amsterdam and Leiden University

October 11, 2016

Conditional execution

Sometimes you want to execute code only under certain conditions

- If the light is green then cross the street else stand still
- If number b is not zero then divide a by b
- If the ghost is close to pacman then run else collect dots
- If you are at a Justin Bieber concert then run away

In code

We can write this condition like this:

```
1 if (condition) {
2     // do something if condition is met
3 }
```

In code

Optionally, we can include an else block:

```
if (condition) {
    // do something if condition is met
} else {
    // if not met, do this
}
```

In code

The condition is a test, that is either `true` or `false`

```
1 int a = 5;
2 if (a < 7) {
3     a++;
4 } else {
5     a--;
6 }
```

Booleans

- A `boolean` is a type that is either `true` or `false`
- We can do logical tests on booleans:

Name	Explanation	Command	Example
eq	equals	==	`true == false`
neq	not equals	!=	`true != false`
and	both should be true	&&	`true && false`
or	one or both should be true	||	`true || false`
not	true → false, false → true	!	`!true`

Table: Some logical tests

Example

```
1 int a = 10, b = 4;
2 boolean c = a < 7;
3 boolean d = c || (b < 5);
```

What is c and d?

Exercise: exclusive or

How do we make exclusive or (just *a* or *b* but not both)?

Exercise: exclusive or

How do we make exclusive or (just *a* or *b* but not both)?

```
1 (a && !b) || (!a && b)
```

Or (surprisingly enough):

```
1 a != b
```

Exercise: leap year

Leap year

A leap year is a year that is divisble by 4, except for century years (e.g., 1900) that are not divisible by 400

- 2004: leap year
- 2000: leap year
- 1900: no leap year

Exercise: leap year

Leap year

A leap year is a year that is divisble by 4, except for century years (e.g., 1900) that are not divisible by 400

- 2004: leap year
- 2000: leap year
- 1900: no leap year

```
1 boolean leapyear = y % 4 == 0 &&
2                    (y % 100 != 0 || y % 400 == 0);
```

While

- Sometimes we want to do something more than once, until a condition is met
- We do this with a `while` loop:

```
1 while(condition) {
2
3 }
```

While

```
while(there are crisps left) {
  eat crisps;
}
```

While

```
1 int i = 0;
2 while(i < 3) {
3   println("Hi, for the " + i + "th time!");
4   i++;
5 }
```

```
Hi, for the 0th time!
Hi, for the 1th time!
Hi, for the 2th time!
```

For statement

- Sometimes, you have a start condition, a stop condition, and a transformation each time.
- Use a `for` loop for this:

```
for(initial ; condition ; step) {
  // some code
}
```

For statement

```
1 for(int i = 0; i < 3; i++) {
2   println("Hi, for the " + i + "th time!");
3 }
```

```
Hi, for the 0th time!
Hi, for the 1th time!
Hi, for the 2th time!
```

Example: growing circles

Easy to do multiple things in a few lines!

```
1 noFill();
2 for(int i = 1; i < 4; i++) {
3   ellipse(320, 320, i * 10, i * 10);
4 }
```

Loops in loops

Loops can be nested:

```
for(int i = 0; i < 2; i++) {
  for(int j = 5; j < 7; j++) {
    println(i + " " + j);
  }
}
```

```
0 5
0 6
1 5
1 6
```

Functions

- A function is a statement that takes arguments and returns a value.
- You can define a function as follows:

```
1 type functioname(type arg1, type arg2, ...) {
2   // something the function does
3   return somevariable;
4 }
```

- type can be any of the defined types: `int`, `boolean`, ...
- `return` returns the expression that is given afterwards.

Examples

```
1 int plusone(int num) {
2   return num + 1;
3 }
4
5 println(plusone(3));
```

Examples

Functions can have no arguments, nor return anything (`void`)

```
1 void drawface() {
2   ellipse(320, 320, 600, 600);
3   ellipse(200, 200, 100, 100);
4   ellipse(440, 200, 100, 100);
5   arc(320, 320, 400, 400, 0.1*PI, 0.9*PI);
6 }
```

Copying and shadowing

- Variable arguments are local (copied)!

```
1 int num = 3;
2
3 int plusone(int num) {
4   num++;
5   return num;
6 }
7
8 println(plusone(num)); // 4
9 println(num); // 3
```

Coding practice

The golden rule is to never repeat yourself!

Structure

Use functions to structure your code and avoid repetition.

Examples

This function draws a face at position x, y

```
1  void drawface(float x, float y) {
2    ellipse(x, y, 600, 600); // outline
3    ellipse(x - 120, y - 120, 100, 100); // left eye
4    ellipse(x + 120, y - 120, 100, 100); // right eye
5    arc(x, y, 400, 400, 0.1*PI, 0.9*PI); // mouth
6  }
7
8  // draw 5 faces side by side
9  for (int i = 0; i < 5; i++) {
10   drawface(100 + i * 100, 100);
11 }
```

Mouse

Check for mouse press:

```
1 void draw() {
2   if (mousePressed) {
3     println("Mouse pressed!");
4   }
5 }
```

Keyboard

Check for key press:

```
1 void draw() {
2   if (keyPressed) {
3     println("Pressing key: " + key);
4   }
5 }
```

- Similarly, `keyReleased`
- Read up on `key` and `keyCode` to check for specific key

Functions

Most of these checks also exist as seperate functions:

```
1 void draw() {
2 }
3
4 void mousePressed() {
5   println("Mouse pressed!");
6 }
```

Similarly for `keyPressed()`