Expressions & Conditionals
CSE 120 Winter 2019

Instructor: Justin Hsia
Teaching Assistants: Ann Shan, Eunia Lee, Pei Lee Yap, Sam Wolfson, Travis McGaha

Delivery Robot Engulfed in Flames, Honored on Campus With Candlelight Vigil

“The University of California, Berkeley, lost a beloved member of their campus last week, when a delivery robot was engulfed in flames outside the student union.

“The courier rovers started delivering food to students about two years ago. The service seems to have been operating relatively smoothly until… a Kiwibot spontaneously combusted while on the job.

“After an investigation, the company revealed … ‘a defective battery was put in place of a functioning one. This caused an exceedingly rare occurrence of the battery experiencing thermal runaway.’ ”

Administrivia

- Assignments:
  - Animal Functions due tonight (1/28)
  - Reading Check 4 due Thursday @ 3:30 (1/31)
  - Jumping Monster due Friday (2/1) (significantly harder!)

- “Big Ideas” this week: Digital Distribution
Outline

- **Expressions & Operators**

- **Conditionals**
  - if
  - else
  - else if
Expressions

“An expression is a combination of one or more values, constants, variables, operators, and functions that the programming language interprets and computes to produce another value.”


Expressions are evaluated and resulting value is used

- Assignment: \( x \leftarrow x + 1; \)
- Assignment: \( x\_pos = \min(x\_pos + 3, 460); \)
- Argument: \( \text{ellipse}(50+x, 50+y, 50, 50); \)
- Argument: \( \text{drawMouse}(rowX+4*50, rowY, rowC); \)
Operators

- Built-in “functions” in Processing that use special symbols:
  - Multiplicative: * mult, / div, % modulus
  - Additive: + add, - sub
  - Relational: < less than, > greater, <= less than or equal to, >= greater than or equal to
  - Equality: == equal to, != not equal to
  - Logical: && and, || or, ! not

- Operators can only be used with certain data types and return certain data types
  - Multiplicative/Additive: 1+2 give numbers, get number (3)
  - Relational: 1 < 5 give numbers, get Boolean (true)
  - Logical: true && true give Boolean, get Boolean (true)
  - Equality: color(0) == color(255) give same type, get Boolean (false)
Operators

- **Built-in “functions” in Processing that use special symbols:**
  - **Multiplicative:** *mult, / div, % modulus*
  - **Additive:** + add, - sub
  - **Relational:** < less than, > greater than, <= less than or equal to, >= greater than or equal to
  - **Equality:** == equal to, != not equal to
  - **Logical:** && and, || or, ! not

- **Logical operators use Boolean values (true, false)**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x &amp;&amp; y</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

| x  | y  | x || y |
|----|----|-------|
| false | false | false |
| false | true  | true  |
| true | false | true  |
| true | true  | true  |

<table>
<thead>
<tr>
<th>x</th>
<th>!x</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>
Operators

- Built-in “functions” in Processing that use special symbols:
  - Multiplicative: * mult, / div, % modulus
  - Additive: + add, - sub
  - Relational: < less than, > greater than, <= less than or equal to, >= greater than or equal to
  - Equality: == equal to, != not equal to
  - Logical: && and, || or, ! not

- In expressions, use parentheses for evaluation ordering and readability
  - e.g. \( x + (y \times z) \) is the same as \( x + y \times z \), but easier to read
  - \( (x + y) \times z \) is required if you want addition to happen first.
Modulus Operator: %

- \( x \mod y \) is read as “\( x \) mod \( y \)” and returns the remainder after \( y \) divides \( x \)
  - For short, we say “mod” instead of modulus

Example Uses:

- **Parity:** Number \( n \) is even if \( n \mod 2 == 0 \)
- **Leap Year:** Year \( year \) is a leap year if \( year \mod 4 == 0 \) divisible by 4 (e.g. 2016, 2020)
- **Chinese Zodiac:** \( year_1 \) and \( year_2 \) are the same animal if \( year_1 \mod 12 == year_2 \mod 12 \)
Conditionals Worksheet

- Work just on Page 1 (Questions 1-6)

- Operators:
  - Arithmetic: +, -, *, /, %
  - Relational: <, >, <=, >=
  - Equality: ==, !=
  - Logical: &&, ||, !

- Data Types:
  - Arithmetic: give numbers, get number
  - Relational: give numbers, get Boolean
  - Logical: give Boolean, get Boolean
  - Equality: give same type, get Boolean
Modulus Example in Processing

- Use mod to “wrap around”
  - Replace \texttt{min/max} function to “connect” edges of drawing canvas

\begin{align*}
\text{x\_pos} & = 459; \\
\text{x\_pos} & = \text{min}(\text{x\_pos} + 3, 460); \quad \text{// stores 460} \\
\text{x\_pos} & = (\text{x\_pos} + 3) \mod 460; \quad \text{// stores 2}
\end{align*}
Control Flow

- The order in which instructions are executed
- We typically say that a program is executed in sequence from top to bottom, but that’s not always the case:
  - Function calls and \texttt{return} calls
  - Conditional/branching statements
  - Loops
- Curly braces \{\} are used to group statements
  - Help parse control flow
  - Remember to use indentation!
Outline

- Expressions & Operators
- Conditionals
  - if
  - else
  - else if
If-Statements

- Sometimes you don’t want to execute *every* instruction
  - Situationally-dependent

- **Conditionals** give the programmer the ability to make decisions
  - The next instruction executed depends on a specified *condition*
    - The condition must evaluate to a **boolean** *(i.e. true or false)*
    - Sometimes referred to as “branching”
  - This generally lines up well with natural language intuition
If-Statements

- Basic form:
  ```
  if(condition) {
    // "then"
    // statements
  }
  ```

- Example conditions:
  - Variable:  `if( done == true )`
  - Variable:  `if( done )`
  - Expression:  `if( x_pos > 460 )`
  - Expression:  `if( x_pos > 100 && y_pos > 100 )`
If-Statements

- With `else` clause:

```java
if (condition) {
    // "then"
    // statements
} else {
    // "otherwise"
    // statements
}
```
If-Statements

- With else if clause:

```c
if (cond1) {
    // "then"
    // statements
} else if (cond2) {
    // "otherwise if"
    // statements
}
```

Diagram:
- Start
- Cond1?
  - True: branch 1
    - "Then" Statements
  - False: Cond2?
    - True: branch 2
      - "Otherwise if" Statements
    - False: branch 3
- End
If-Statements

- Notice that conditionals *always* go from Start to End
  - Choose one of many *branches*
  - A conditional must have a single `if`, as many `else if` as desired, and at most one `else`

- Can nest and combine in interesting ways:

```
if (cond1) {
    if (cond2) {
        // statement1
    } else {
        // statement2
    }
} else if (cond1) {
    // statement2
}
```
Practice Question

Which value of $x$ will get the following code to print out "Maybe"?

A. 1  No
B. 3  Maybe
C. 5  Yes
D. 7  No
E. We’re lost...

Think for a minute, then discuss with your neighbor(s)

- Vote at http://PollEv.com/justinh
Conditionals Worksheet

- Work on Page 2 (Questions 7-9)

```java
if (cond1) {
    // "then"
} else if (cond2) {
    // "otherwise if"
} else {
    // "else"
}
```
Processing Demo: Drawing Dots

```java
void draw() {
    if(mousePressed) {
        fill(0, 0, 255); // blue if mouse is pressed
    } else {
        fill(255, 0, 0); // red otherwise
    }
    ellipse(mouseX, mouseY, 5, 5); // draw circle
}
```
Jumping Monster

- Using *expressions* and *conditionals* in conjunction with *variables* and *user input* (Wed) to control what is drawn as well as motion: