CS 101 Lecture 21

Flow Control:
boolean expressions,
“if” selection statements
(Alice In Action, Ch 4)

8 November 2012

Slides Credit: Joel Adams, Alice in Action

Flow Control
• Flow: sequence of steps for performing a user story
• Flow control statement: structure for managing flow
• Flow control statements used in previous chapters
  – doInOrder: produces a sequential execution
  – doTogether: produces a parallel execution
  – methods and functions: name a block of statements

FIGURE 4-1 The flows produced by the doInOrder and doTogether statements
Flow Control

- Control statements introduced in today
  - **if**: directs program flow along one of two paths

![Flow through an if statement](image)

Objectives

- Use the **Boolean** type and its basic operations

- Use **Boolean** variables, expressions and functions to control **if** statements (and later **while** statements)

- Use the **if** statement to perform some statements while skipping others
Boolean Variables

- Boolean values: true, false

- Boolean Variables: are of the Boolean type
  - Used to store a value of true or false
  - Can be used in condition for if or while statement
  - How to create a Boolean variable
    - Click create new variable (or parameter) button
    - Specify Boolean as variable (or parameter) type

Boolean Functions

- Return a value of true or false
- Can act as a condition in an if or while statement

- Alice: many predefined Boolean functions refer to an object’s bounding box
  - Example: obj.isBehind(obj2)
    - true, if obj’s position is beyond obj2’s rear face
    - false, otherwise
Boolean Expressions

- Boolean expression
  - Producing a value of true or false, which can be
    - Stored in a boolean variable
    - Returned from a boolean function
    - Used in if or while statement, as the description of a condition, and the basis for decision making

  - E.g. if (temperature > 95) && (notRain == true)
    goToSwim()

Relational Operators

- Produce true or false values
- Six relational operators: ==, !, <, <=, >, >=
- Located in functions pane of world's details area
- Most often used to compare Number values
- Example: hoursWorked > 40
  - hoursWorked is a Number variable
  - true when more than 40 hours have been worked
Relational Operators (continued)

<table>
<thead>
<tr>
<th>Relational Operator</th>
<th>Name</th>
<th>Value Produced</th>
</tr>
</thead>
<tbody>
<tr>
<td>val1 == val2</td>
<td>equality</td>
<td>true, if val1 and val2 have the same value; false, otherwise.</td>
</tr>
<tr>
<td>val1 != val2</td>
<td>inequality</td>
<td>true, if val1 and val2 have different values; false, otherwise.</td>
</tr>
<tr>
<td>val1 &lt; val2</td>
<td>less-than</td>
<td>true, if val1 is less than val2; false, otherwise.</td>
</tr>
<tr>
<td>val1 &lt;= val2</td>
<td>less-than-or-equal</td>
<td>true, if val1 is less than or equal to val2; false, otherwise.</td>
</tr>
<tr>
<td>val1 &gt; val2</td>
<td>greater-than</td>
<td>true, if val1 is greater than val2; false, otherwise.</td>
</tr>
<tr>
<td>val1 &gt;= val2</td>
<td>greater-than-or-equal</td>
<td>true, if val1 is greater than or equal to val2; false, otherwise.</td>
</tr>
</tbody>
</table>

FIGURE 4-3 The relational operators

Boolean Operators

- Used to modify or combine relational operations
- Three Boolean operators: **AND**, **OR**, **NOT**
- Located in functions pane of world’s details area
- Example: `age > 12 && age < 20`
  - `age` is a Number variable
  - Teen number compared to condition returns true
Boolean Operators (continued)

<table>
<thead>
<tr>
<th>Boolean Operation</th>
<th>Name</th>
<th>Value Produced</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>val1 &amp;&amp; val2</code></td>
<td>AND</td>
<td>true, if <code>val1</code> and <code>val2</code> are both true; false, otherwise.</td>
</tr>
<tr>
<td>`val1</td>
<td></td>
<td>val2`</td>
</tr>
<tr>
<td><code>!val1</code></td>
<td>NOT</td>
<td>true, if <code>val1</code> is false; false, if <code>val1</code> is true.</td>
</tr>
</tbody>
</table>

FIGURE 4-4 The boolean operators

| `var1` | `var2` | `var1 && var2` | `var1 || var2` | `!var1` |
|--------|--------|---------------|---------------|--------|
| true   | true   | true          | true          | false  |
| true   | false  | false         | true          | false  |
| false  | true   | false         | true          | true   |
| false  | false  | false         | false         | true   |

This is called a truth table.
Introducing Selective Flow Control

• Example: a scene with a princess and a dragon
  – Princess meets a mute dragon and asks questions
  – Dragon shakes its head to respond yes or no

• Objective: write a `shakeHead()` method
  – Parameter: `yesOrNo`, a `String`
  – If `yesOrNo == “yes”`, dragon shakes head up and down
  – If `yesOrNo == “no”`, dragon shakes head sideways
  – Use an `if` statement to produce conditional behavior

if Statement

• Structure of an `if` statement:
  – `if (Condition )`
    {
      \[Statements_1\]
    } else {
      \[Statements_2\]
    }

• Value of a condition determines direction of flow
  – If `Condition` is `true`, \[Statements_1\] are selected
  – If `Condition` is `false`, \[Statements_2\] are selected
**if Statement Mechanics (continued)**

If statement behavior is also called **selective flow or selection.**

![Diagram](image)

**Building if Statement Conditions**

- The `if` control structure is at bottom of edit area
- After dragging it to the edit area, you get

![](image)

Note that it's ok to leave the `else` block as `Do Nothing`. 

---

8

15
Building if Statement Conditions

- Coding the condition of the if statement
  - Click on the yesOrNo parameter
  - Drag parameter into the editing area
  - Drop the parameter onto the condition’s placeholder
  - Choose other and then type “yes”

![Drag parameter to condition](image)

Building if Statement Conditions

- Overview for coding the remainder of shakeHead()
  - Add headMovement variable for amount of turn
  - Add turn() statements for up and down motion
  - Add turn() statements for sideways motion
Building if Statement Conditions (continued)

- Building a scene method that uses `shakeHead()`
  - `princess` greets `dragon` using a `say()` message
  - `princess` asks four questions
  - `shakeHead()` is called in response to each question
- Click the `Play` button to test the program

![Image](image.png)

**FIGURE 4-12** Testing `shakeHead()` in `playScene3()`
Building if Statement Conditions (continued)

The `wait()` Statement

- Pauses a program for specified number of seconds
- Form of `wait()` statement: `wait(numSecs);`
- Use of `wait()` scene with dragon and princess
  - Inserted between princess’s first and second lines
USS Yorktown

On 21 September 1997, while on maneuvers off the coast of Cape Charles, Virginia, a crew member entered a zero into a database field causing a divide by zero error in the ship's Remote Data Base Manager which brought down all the machines on the network, causing the ship's propulsion system to fail. Yorktown was "dead in the water" for 2 hours and 45 minutes.


Question: How do we prevent this?

Validating Parameter Values

• **if** statement can be used to guard set of statements
  – Flow enters only if parameter values are valid
• Example: check distance value passed to jump()
  – Check for positive value with condition distance > 0
  – Check jump length with distance < MAX_DISTANCE
  – Combine two conditions with the AND (&&) operator
    • distance > 0 && distance <= MAX_DISTANCE
• How to incorporate validating logic using if structures
  – Place original jump() logic onto true path (outer if)
  – Place validating logic in the false path (nested if)
• Nested logic
Validating Parameter Values (continued)

![Image of code snippet]

FIGURE 4.14 Validating a parameter’s value with nested if statements

Validating Parameter Values (continued)

![Image of fish image with text]

FIGURE 4.15 Asking the fish to jump a negative distance
Validating Parameter Values (continued)

Summary

- Flow control statement: controls the flow of statement execution
- if statement: directs flow along one of two paths based on evaluation of a condition
Summary (continued)

- Condition: a Boolean entity producing a true or false value
- Boolean variable: holds value of true or false
- Boolean function: returns true or false value
- Boolean expression: produces true or false value
  - Relational operators: ==, !=, <, <=, >, >=
  - Boolean operators: &&, ||, !