1.3 Conditionals and Loops

A Foundation for Programming

any program you might want to write
- objects
- functions and modules
- graphics, sound, and image I/O
- arrays
- conditionals and loops
- Math
- text I/O
- primitive data types
- assignment statements

Control Flow

Control flow
- Sequence of statements that are actually executed in a program
- Conditionals and loops: enable us to choreograph control flow

Conditionals

If Statement

The if statement
- A common branching structure
  - Evaluate a boolean expression
  - If true, execute some statements
  - If false, execute other statements

```
if (boolean expression) {
  statement T;
} else {
  statement F;
}
```
### If Statement

**The if statement** A common branching structure

- Evaluate a boolean expression
  - If true, execute some statements
  - If false, execute other statements

```java
if (x < 0) x = -x;
int t = x;
x = y;
y = t;
```

**If Statement Examples**

<table>
<thead>
<tr>
<th>Boolean expression (true)</th>
<th>Boolean expression (false)</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (x &lt; 0) x = -x;</td>
<td>if (x &gt; 0) max = x; max = y;</td>
</tr>
<tr>
<td>int t = x;</td>
<td>x = y;</td>
</tr>
<tr>
<td>x = y;</td>
<td>y = t;</td>
</tr>
<tr>
<td>if (x &gt; y) max = x;</td>
<td>if (x &lt; y) max = x; max = y;</td>
</tr>
<tr>
<td>else</td>
<td>else</td>
</tr>
<tr>
<td>System.out.println(&quot;Heads&quot;);</td>
<td>System.out.println(&quot;Tails&quot;);</td>
</tr>
<tr>
<td>System.out.println(&quot;Heads&quot;);</td>
<td>System.out.println(&quot;Tails&quot;);</td>
</tr>
</tbody>
</table>

### The For Loop

**The for loop** A common repetition structure

- Execute initialization statement
- Evaluate a boolean expression
  - If true, execute some statements
  - And then the increment statement
- Repeat

```java
for (init; boolean expression; increment) {
    statement 1;
    statement 2;
}
```

**Anatomy of a For Loop**

- Initialize loop control variable
- Evaluate loop continuation condition
- Increment loop control variable
- Repeat

Q. What does it print?

A.
For Loop: Powers of Two

Ex. Print powers of 2 that are $\leq 2^N$
- Increment $i$ from 0 to $N$
- Double $v$ each time

```java
int v = 1;
for (int i = 0; i <= N; i++) {
    System.out.println(i + " " + v);
    v = 2 * v;
}
```

<table>
<thead>
<tr>
<th>$i$</th>
<th>$v$</th>
<th>$i \leq N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>false</td>
</tr>
</tbody>
</table>

$N = 6$

Click for demo

For Loops: Subdivisions of a Ruler

Create subdivision of a ruler
- Initialize ruler to ""
- For each value $i$ from 1 to $N$:
  sandwich two copies of ruler on either side of $i$

```java
public class RulerN {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        String ruler = " ";
        for (int i = 1; i <= N; i++) {
            ruler += i;
            ruler += i;
        }
        System.out.println(ruler);
    }
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>ruler</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1 2 1</td>
</tr>
<tr>
<td>3</td>
<td>1 2 1 3 1 2 1</td>
</tr>
<tr>
<td>4</td>
<td>1 2 1 3 1 2 1 4 1 2 1 3 1 2 1</td>
</tr>
<tr>
<td>5</td>
<td>1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1</td>
</tr>
<tr>
<td>6</td>
<td>1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 6 1 2 1 3 1 2 1</td>
</tr>
<tr>
<td>7</td>
<td>1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 6 1 2 1 3 1 2 1 7 1 2 1 3 1 2 1</td>
</tr>
</tbody>
</table>

Exception in thread "main"
java.lang.OutOfMemoryError

Observation: Loops can produce a huge amount of output!

The While Loop

The while loop. Another common repetition structure
- Evaluate a boolean expression
- If true, execute some statements
- Repeat

```java
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i += 1;
    v *= 2;
}
```

<table>
<thead>
<tr>
<th>$i$</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

$N = 6$

Click for demo

While Loop: Powers of Two

Ex. Print powers of 2 that are $\leq 2^N$
- Increment $i$ from 0 to $N$
- Double $v$ each time

```java
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i += 1;
    v *= 2;
}
```

<table>
<thead>
<tr>
<th>$i$</th>
<th>$v$</th>
<th>$i \leq N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>true</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>true</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>false</td>
</tr>
</tbody>
</table>

$N = 6$

Click for demo
While Loop Challenge

Q. Anything wrong with the following code for printing powers of 2?

```java
public class PowersOfTwo {
    public static void main(String[] args) {
        // last power of two to print
        int N = Integer.parseInt(args[0]);
        int i = 0; // loop control counter
        while (i <= N) {
            System.out.println(i + " " + v);
            i = 1 + i;
        }
    }
}
```

A. Need curly braces around statements in while loop; otherwise it enters an infinite loop, printing "0 1".

Moment of panic. How to stop infinite loop?

While Loops: Square Root

Goal. Implement `Math.sqrt()`.

Newton-Raphson method to compute the square root of c:
- Initialize \( t_0 = c \).
- Repeat until \( t_i = c / t_i \) up to desired precision:
  - set \( t_{i+1} \) to be the average of \( t_i \) and \( c / t_i \).

| \( t_i \) | 2.0 |
| \( t_1 \) | \( \frac{t_0 + \frac{c}{t_0}}{2} \) | 1.5 |
| \( t_2 \) | \( \frac{t_1 + \frac{c}{t_1}}{2} \) | 1.4166666666666665 |
| \( t_3 \) | \( \frac{t_2 + \frac{c}{t_2}}{2} \) | 1.4142556178849052 |
| \( t_4 \) | \( \frac{t_3 + \frac{c}{t_3}}{2} \) | 1.4142135623730952 |
| \( t_5 \) | \( \frac{t_4 + \frac{c}{t_4}}{2} \) | 1.4142135623730952 |

Technical conditions. \( f(x) = x^2 - c \) to compute \( x \):
- Goal: find root of any function \( f(x) \).
- Start with estimate \( t_0 \).
- Draw line tangent to curve at \( x = t_1 \).
- Set \( t_0 \) to be \( x \)-coordinate where line hits \( x \)-axis
- Repeat until desired precision
Loop Examples

\[\text{int v = 1;}\]
\[\text{while (v < N(2))} \]
\[\text{v = 2*v;}\]
\[\text{System.out.println(v);}\]

\[\text{int sum = 0;}\]
\[\text{for (int i = 1; i <= N; i++)} \]
\[\text{sum = sum + i;}\]
\[\text{System.out.println(sum);}\]

\[\text{int product = 1;}\]
\[\text{for (int i = 1; i <= N; i++)} \]
\[\text{product = product * i;}\]
\[\text{System.out.println(product);}\]

\[\text{for (int i = 0; i <= N; i++)} \]
\[\text{System.out.println(i = "\^" + Math.PI*i/M);}\]

Ex: Pay a certain tax rate depending on income level

<table>
<thead>
<tr>
<th>Income</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 47,450</td>
<td>22%</td>
</tr>
<tr>
<td>47,450 - 114,650</td>
<td>25%</td>
</tr>
<tr>
<td>114,650 - 174,700</td>
<td>28%</td>
</tr>
<tr>
<td>174,700 - 311,950</td>
<td>33%</td>
</tr>
<tr>
<td>311,950 - 35%</td>
<td></td>
</tr>
</tbody>
</table>

Nested If Statements

Use nested if statements to handle multiple alternatives

```java
if (income < 47450) rate = 0.22;
else if (income < 114650) rate = 0.25;
else if (income < 174700) rate = 0.28;
else if (income < 311950) rate = 0.33;
else rate = 0.35;
```

Nested If Statement Challenge

Q. What's wrong with the following for income tax calculation?

<table>
<thead>
<tr>
<th>Income</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 47,450</td>
<td>22%</td>
</tr>
<tr>
<td>47,450 - 114,650</td>
<td>25%</td>
</tr>
<tr>
<td>114,650 - 174,700</td>
<td>28%</td>
</tr>
<tr>
<td>174,700 - 311,950</td>
<td>33%</td>
</tr>
<tr>
<td>311,950 - 35%</td>
<td></td>
</tr>
</tbody>
</table>

Wrong graduated income tax calculation
Monte Carlo Simulation

Gambler’s Ruin

Gambler’s Ruin
Gambler starts with $stake and places $1 fair bets until going broke or reaching $goal

- What are the chances of winning?
- How many bets will it take?

One approach: Monte Carlo simulation
- Flip digital coins and see what happens
- Repeat and compute statistics

Digression: Simulation and Analysis

Fact
- Probability of winning = stake / goal

Ex. 20% chance of turning $500 into $2500, but expect to make one million $1 bets

Fact
- Expected number of bets = stake * desired gain

Remark: Both facts can be proved mathematically; for more complex scenarios, computer simulation is often the best (only) plan of attack

Control Flow Summary

Control flow
- Sequence of statements that are actually executed in a program
- Conditionals and loops: enable us to choreograph the control flow

<table>
<thead>
<tr>
<th>Control Flow</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>straight-line programs</td>
<td>all statements are executed in the order given</td>
<td></td>
</tr>
<tr>
<td>conditionals</td>
<td>certain statements are executed depending on the values of certain variables</td>
<td>if, if-else</td>
</tr>
<tr>
<td>loops</td>
<td>certain statements are executed repeatedly until certain conditions are met</td>
<td>while, for-do-while</td>
</tr>
</tbody>
</table>