1.3 Conditionals and Loops

Conditionals and loops enable us to choreograph control flow. They are used in any program you might want to write. The basic structure of a program includes:

- **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 includes:
- **Sequence of statements** that are actually executed in a program.
- **Conditionals and loops** allow us to choreograph control flow.
Conditionals

If Statement

The if statement. A common branching structure.
- Check boolean condition.
- If true, execute some statements.
- If false, execute other statements.

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

can be any sequence of statements

Ex. Take different action depending on value of variable.

```java
class Flip{
  public static void main(String[] args)
  {
    if (Math.random() < 0.5)
      System.out.println("Heads");
    else System.out.println("Tails");
  }
}
```

% java Flip
Heads
% java Flip
Heads
% java Flip
Tails
% java Flip
Heads
If Statement Examples

| absolute value | if \( x < 0 \) \( x = -x; \)
| put x and y into sorted order | if \( x > y \)
| | \{ \n| | \hspace{1cm} \text{int} \ t = x; \n| | \hspace{1cm} y = x; \n| | \hspace{1cm} x = t; \n| | \} \n| maximum of x and y | if \( x > y \) max = x; \n| | else max = y; \n| error check for division operation | if \( \text{den} == 0 \) System.out.println("Division by zero"); \n| | else System.out.println("Quotient = \( \frac{\text{num}}{\text{den}} \); \n| error check for quadratic formula | double discriminant = \( b^2 - 4.0c \); \n| | if (discriminant < 0.0) \n| | \{ System.out.println("No real roots"); \n| | \} \n| | else \n| | \{ System.out.println((-b + Math.sqrt(discriminant))/2.0); \n| | System.out.println((-b - Math.sqrt(discriminant))/2.0); \n| |

While Loop

The while loop. A common repetition structure.
- Check a boolean expression.
- Execute a sequence of statements.
- Repeat.

```
while (boolean expression)
    { statement 1; \n    statement 2; \n    } \n```
While Loop Example: Powers of Two

Ex. Print first \( n \) powers of 2.
- Increment \( i \) from 1 to \( n \).
- Double \( v \) each time.

<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>

\( n = 6 \)

While Loop Challenge

Anything wrong with the following code?

```java
public class PowersOfTwo {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int i = 0; // loop control counter
        int v = 1; // current power of two
        while (i <= N) {
            System.out.println(v);
            i = i + 1;
            v = 2 * v;
        }
    }
}
```

While Loop Example: 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 \( c / t_i \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>( t_i )</th>
<th>( 2/t_i )</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.0</td>
<td>1.0</td>
<td>1.5</td>
</tr>
<tr>
<td>1</td>
<td>1.5</td>
<td>1.333333</td>
<td>1.416667</td>
</tr>
<tr>
<td>2</td>
<td>1.416667</td>
<td>1.4117647</td>
<td>1.4142157</td>
</tr>
<tr>
<td>3</td>
<td>1.4142157</td>
<td>1.4142114</td>
<td>1.4142136</td>
</tr>
<tr>
<td>4</td>
<td>1.4142136</td>
<td>1.4142136</td>
<td>1.4142136</td>
</tr>
</tbody>
</table>

computing the square root of 2 to seven places

“A wonderful square root. Let’s hope it can be used for the good of mankind.”

Copyright 2004, Sidney Harris
http://www.sciencecartoonsplus.com
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 $c / t_i$.

```java
public class Sqrt {
    public static void main(String[] args) {
        double EPS = 1E-15;
        double c = Double.parseDouble(args[0]);
        double t = c;
        while (Math.abs(t - c/t) > t*EPS) {
            t = (c/t + t) / 2.0;
        }
        System.out.println(t);
    }
}
```

% java Sqrt 2.0 1.414213562373095

15 decimal digits of accuracy in 5 iterations
Anatomy of a For Loop

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

Observation. Loops can produce a huge amount of output!

For Loops: Subdivisions of a Ruler

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

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

Loop Examples

<table>
<thead>
<tr>
<th>i</th>
<th>ruler</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot; 1 &quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot; 1 2 1 &quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot; 1 2 1 3 1 2 1 &quot;</td>
</tr>
</tbody>
</table>

\% java Ruler 1
1
\% java Ruler 2
1 2 1
\% java Ruler 3
1 2 1 3 1 2 1
\% java Ruler 4
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
\% java Ruler 5
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
\% java Ruler 100
Exception in thread "main"
java.lang.OutOfMemoryError
Nesting

**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-Else Statements**

Need all those braces? Not always:

```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;
```

is shorthand for

```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;
  }
}
```

but BE CAREFUL when nesting if-else statements (see Q&A p. 75).
If-Else Statement Challenge

Anything wrong with the following code?

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

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.

Nesting Example: Gambler’s Ruin Simulation

```java
public class Gambler
{
    public static void main(String[] args)
    {
        // Get parameters from command line.
        int stake = Integer.parseInt(args[0]);
        int goal = Integer.parseInt(args[1]);
        int trials = Integer.parseInt(args[2]);

        int wins = 0;
        for (int i = 0; i < trials; i++)
        {
            // Do one gambler's ruin experiment.
            int t = stake;
            while (t > 0 && t < goal)
            {
                // flip coin and update
                if (Math.random() < 0.5) t++;
                else t--;
            }
            if (t == goal) wins++;
        }
        System.out.println(wins + " wins of " + trials);
    }
}
```

Digression: Simulation and Analysis

```java
% java Gambler 5 25 1000
191 wins of 1000
% java Gambler 5 25 1000
203 wins of 1000
% java Gambler 500 2500 1000
197 wins of 1000
```

Fact. Probability of winning = stake / goal.

Fact. Expected number of bets = stake \times desired gain.

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

Remark. Both facts can be proved mathematically.
For more complex scenarios, computer simulation is often the best plan of attack.
Control Flow Summary

Control flow.
- Sequence of statements that are actually executed in a program.
- Conditionals and loops: enables 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>

99% of program development

Debugging. Cyclic process of editing, compiling, and fixing errors.
- Always a logical explanation.
- What would the machine do?
- Explain it to the teddy bear.

You will make many mistakes as you write programs. It’s normal.

Good news: Can use computer to test program.
Bad news: Conditionals/loops open up huge number of possibilities.
Really bad news: Cannot use computer to automatically find all bugs.

Debugging Example

**Factor.** Given an integer \( N > 1 \), compute its prime factorization.

\[
3,757,208 = 2^3 \times 7 \times 13^2 \times 397
\]

\[
98 = 2 \times 7^2
\]

\[
17 = 17
\]

11,111,111,111,111 = 2,071,723 \times 5,363,222,357

Note: 1 is not prime. (else it would have to be in every factorization)

**Application.** Break RSA cryptosystem (factor 200-digit numbers).
Debugging: 99% of Program Development

**Programming.** A process of finding and fixing mistakes.
- Compiler error messages help locate syntax errors.
- Run program to find semantic and performance errors.

```java
public class Factors {
    public static void main(String[] args) {
        long N = Long.parseLong(args[0]);
        for (int i = 0; i < N; i++) {
            if (N % i == 0) {
                System.out.print(i + " ");
                N = N / i;
            }
        }
    }
}
```

This program has bugs!

Debugging: Syntax Errors

**Syntax error.** Illegal Java program.
- Compiler error messages help locate problem.
- Goal: no errors and a file named `Factors.class`.

```java
public class Factors {
    public static void main(String[] args) {
        long N = Long.parseLong(args[0]);
        for (i = 0; i < N; i++) {
            while (N % i == 0)
                System.out.print(i + " ");
                N = N / i;
        }
    }
}
```

% javac Factors.java
Factors.java:6: ';' expected
for (i = 2; i < N; i++)
^ 1 error ← the FIRST error
Debugging: Semantic Errors

Semantic error. Legal but wrong Java program.
- Run program to identify problem.
- Add print statements if needed to produce trace.

```java
public class Factors {
    public static void main(String[] args) {
        long N = Long.parseLong(args[0]);
        for (int i = 0; i < N; i++) {
            while (N % i == 0) {
                System.out.print(i + " ");
                N = N / i;
            }
        }
    }
}
```

```
% javac Factors.java
% java Factors
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0 at Factors.main(Factors.java:5)
```

oops, need argument

...you will see this message!

Debugging: Semantic Errors

Semantic error. Legal but wrong Java program.
- Run program to identify problem.
- Add print statements if needed.

```java
public class Factors {
    public static void main(String[] args) {
        long N = Long.parseLong(args[0]);
        for (int i = 0; i < N; i++) {
            while (N % i == 0) {
                System.out.print(i + " ");
                N = N / i;
            }
        }
    }
}
```

```
% javac Factors.java
% java Factors 98
Exception in thread "main" java.lang.ArithmeticException: / by zero at Factors.main(Factors.java:8)
```

need to start at 2 since 0 and 1 cannot be factors

Debugging: Semantic Errors

Semantic error. Legal but wrong Java program.
- Run program to identify problem.
- Add print statements if needed.

```java
public class Factors {
    public static void main(String[] args) {
        long N = Long.parseLong(args[0]);
        for (int i = 1; i < N; i++) {  // Start at 2
            while (N % i == 0) {
                System.out.print(i + " ");
                N = N / i;
            }
        }
    }
}
```

```
% javac Factors.java
% java Factors 98
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

??? infinite loop
Debugging: Semantic Errors

Semantic error. Legal but wrong Java program.

- Run program to identify problem.
- Add print statements if needed.

```java
public class Factors {
    public static void main(String[] args) {
        long N = Long.parseLong(args[0]);
        for (int i = 2; i < N; i++) {
            while (N % i == 0) {
                System.out.print(i + " ");
                N = N / i;
            }
        }
        System.out.println("TRACE " + N);
    }
}
```

Semantic (run-time) error: indents do not imply braces

Success? Program seems to work.

- Time to try it for other inputs.
- Add trace to find and fix (minor) problems.

```java
public class Factors {
    public static void main(String[] args) {
        long N = Long.parseLong(args[0]);
        for (int i = 2; i < N; i++) {
            System.out.println(i + " ");
            N = N / i;
        }
        if (N > 1) System.out.println(N);
        else System.out.println();
    }
}
```

Corner case: print largest factor (and new line)

Debugging: The Beat Goes On

Success? Program factors 98 = 2 2 7.

- Time to try it for other inputs.
- Add trace to find and fix (minor) problems.

```java
public class Factors {
    public static void main(String[] args) {
        long N = Long.parseLong(args[0]);
        for (int i = 2; i < N; i++) {
            // Check whether i is a factor.
            while (N % i == 0) {
                // If so, print and divide.
                System.out.print(i + " ");
                N = N / i;
            }
        }
        if (N > 1) System.out.println(N);
        else System.out.println();
    }
}
```

Success? Program seems to work.

- Remove trace to try larger inputs.
- [stay tuned].

```java
% javac Factors.java
% java Factors 5
TRACE 2 5
TRACE 3 5
TRACE 4 5
% java Factors 6
2
% java Factors 98
2 7 7
% java Factors 3757208
2 2 2 7 13 13 397
```

AHA!

Print out N after for loop (if it is not 1)

```java
% javac Factors.java
% java Factors 5
2 7 7 % need newline
% java Factors 6
2 3
% java Factors 98
2 7 7
% java Factors 3757208 2 2 2 7 13 13 397
```

% forgot to recompile

```java
% java Factors 5
% java Factors 6
% java Factors 98
% java Factors 3757208
% java Factors 5
% java Factors 6
% java Factors 98
% java Factors 3757208
```

???

Forgot to recompile

Time to add comments (if not earlier).
Debugging Your Program

1. Create the program.
2. Compile it.
   Compiler says: That's not a legal program.
   Back to step 1 to fix your errors of syntax.
3. Execute it.
   Result is bizarrely (or subtly) wrong.
   Back to step 1 to fix your errors of semantics.
4. Enjoy the satisfaction of a working program!
   [but stay tuned for more debugging]