I/O (cont) and Program Development

A Foundation for Programming

Standard Audio

Sound. Perception of the vibration of molecules in our eardrums.

Concert A. Sine wave, scaled to oscillate at 440Hz.

Other notes. 12 notes on chromatic scale, divided logarithmically.

<table>
<thead>
<tr>
<th>note</th>
<th>i</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>440.00</td>
</tr>
<tr>
<td>A#</td>
<td>1</td>
<td>466.16</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>493.88</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>523.25</td>
</tr>
<tr>
<td>C#</td>
<td>4</td>
<td>554.37</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>587.33</td>
</tr>
<tr>
<td>D#</td>
<td>6</td>
<td>622.25</td>
</tr>
<tr>
<td>E</td>
<td>7</td>
<td>659.26</td>
</tr>
<tr>
<td>F</td>
<td>8</td>
<td>698.46</td>
</tr>
<tr>
<td>F#</td>
<td>9</td>
<td>739.99</td>
</tr>
<tr>
<td>G</td>
<td>10</td>
<td>783.99</td>
</tr>
<tr>
<td>G#</td>
<td>11</td>
<td>830.61</td>
</tr>
<tr>
<td>A</td>
<td>12</td>
<td>880.00</td>
</tr>
</tbody>
</table>

Notes, numbers, and waves
Digital Audio

**Sampling.** Represent curve by sampling it at regular intervals.

\[ y(i) = \sin \left( \frac{2\pi i \cdot 440}{44,100} \right) \]

Standard Audio

**Standard audio.** Library for playing digital audio.

```java
public class StdAudio {
  void play(String file) { ... }
  void play(double[] a) { ... }
  void save(String file, double[] a) { ... }
  void double[] read(String file) { ... }
}
```

Play That Tune

**Goal.** Read in pitches and durations from standard input, and play using standard audio.

```java
public class Tone {
  public static void main(String[] args) {
    int sps = 44100;
    double hz = Double.parseDouble(args[0]);
    double duration = Double.parseDouble(args[1]);
    int N = (int) (sps * duration);
    double[] a = new double[N+1];
    for (int i = 0; i <= N; i++)
      a[i] = Math.sin(2 * Math.PI * i * hz / sps);
    StdAudio.play(a);
  }
}
```

Warmup: Musical Tone

**Musical tone.** Create a music tone of a given frequency and duration.

```java
public class Tone {
  public static void main(String[] args) {
    int sps = 44100;
    double hz = Double.parseDouble(args[0]);
    double duration = Double.parseDouble(args[1]);
    int N = (int) (sps * duration);
    double[] a = new double[N+1];
    for (int i = 0; i <= N; i++)
      a[i] = Math.sin(2 * Math.PI * i * hz / sps);
    StdAudio.play(a);
  }
}
```
Play That Tune

**Goal.** Read in pitches and durations from standard input, and play using standard audio.

```java
public class PlayThatTune {
    public static void main(String[] args) {
        int sps = 44100;
        while (!StdIn.isEmpty()) {
            int pitch = StdIn.readInt();
            double duration = StdIn.readDouble();
            double hz = 440 * Math.pow(2, pitch / 12.0);
            int N = (int) (sps * duration);
            double[] a = new double[N+1];
            for (int i = 0; i <= N; i++)
                a[i] = Math.sin(2 * Math.PI * i * hz / sps);
            StdAudio.play(a);
        }
    }
}
```

95% of Program Development

**Program development.** Creating a program and putting it to good use.

**Def.** A **bug** is a mistake in a computer program.

Programming is primarily a **process** of finding and fixing bugs.

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.

“As soon as we started programming, we found out to our surprise that it wasn’t as easy to get programs right as we had thought. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs.” — Maurice Wilkes

“If I had eight hours to chop down a tree, I would spend six hours sharpening an axe.” — Abraham Lincoln
Debugging Example

Factor. Given an integer $N > 1$, compute its prime factorization.

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

$$98 = 2 \times 7^2$$

$$17 = 17$$

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


Brute-force algorithm. For each putative factor $i = 2, 3, 4, \ldots$, check if $N$ is a multiple of $i$, and if so, divide it out.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$N$</th>
<th>output</th>
<th>$i$</th>
<th>$N$</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3757208</td>
<td>2 2 2</td>
<td>9</td>
<td>67093</td>
<td>16 397</td>
</tr>
<tr>
<td>3</td>
<td>469651</td>
<td>11 11 11</td>
<td>10</td>
<td>67093</td>
<td>17 397</td>
</tr>
<tr>
<td>4</td>
<td>67093</td>
<td>17</td>
<td>11</td>
<td>67093</td>
<td>18 397</td>
</tr>
<tr>
<td>5</td>
<td>67093</td>
<td>17</td>
<td>12</td>
<td>67093</td>
<td>19 397</td>
</tr>
<tr>
<td>6</td>
<td>67093</td>
<td>17</td>
<td>13</td>
<td>67093</td>
<td>20 397</td>
</tr>
<tr>
<td>7</td>
<td>67093</td>
<td>17</td>
<td>14</td>
<td>397</td>
<td>397</td>
</tr>
<tr>
<td>8</td>
<td>67093</td>
<td>17</td>
<td>15</td>
<td>397</td>
<td>397</td>
</tr>
</tbody>
</table>

Debugging: 95% 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++) {
            while (N % i == 0) {
                System.out.print(i + " ");
                N = N / i;
            }
        }
    }
}
```

this program has many 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 (int i = 0; i < N; i++) {
            while (N % i == 0) {
                System.out.print(i + " ");
                N = N / i;
            }
        }
    }
}
```

% javac Factors.java
Factors.java:4: ';' expected
for (i = 0; i < N; i++)
  ^
1 error the first error
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 + " ");
        }
    }
}

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

Semantic error. Legal but wrong Java program.
- Run program to identify problem.
- Add print statements if needed to produce trace.
Success. Program factors $98 = 2 \times 7^2$.

- But that doesn’t mean it works for all 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++) {
            while (N % i == 0) {
                System.out.print(i + " ");
                N = N / i;
            }
        }
        System.out.println();
    }
}
```

Success. Program now seems to work.

```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();
    }
}
```

Success. Program factors $98 = 2 \times 7^2$.

- But that doesn’t mean it works for all 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++) {
            while (N % i == 0) {
                System.out.print(i + " ");
                N = N / i;
            }
        }
        System.out.println();
    }
}
```

Performance error. Correct program, but too slow.

```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();
    }
}
```
Debugging: Performance Error

Performance error. Correct program, but too slow.

Solution. Improve or change underlying algorithm.

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

```
% java Factors 98
2 7
% java Factors 1111111
11 11 101
% java Factors 1111111111111111
2071723
```

need special case to print biggest factor (unless it occurs more than once)

```
% java Factors 3757208
2 2 2 7 13 13 397
% java Factors 920111116975555703
9201111169755555703
```

Q. How large an integer can I factor?

```
Note. Can’t break RSA this way (experts are still trying).
```

Caveat. Optimizing your code tends to introduce bugs.

Lesson. Don’t optimize until it’s absolutely necessary.

```java
if (N > 1) System.out.println(N);
else System.out.println();
```

need special case to print biggest factor (unless it occurs more than once)

```
% java Factors 1111111
11 11 101 137
% javaFactors 9201111169755555703
9201111169755555703
```

Program Development: Analysis

<table>
<thead>
<tr>
<th>largest factor</th>
<th>(i &lt;= N)</th>
<th>(i &lt;= N/i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>digits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>instant</td>
<td>instant</td>
</tr>
<tr>
<td>6</td>
<td>0.15s</td>
<td>instant</td>
</tr>
<tr>
<td>9</td>
<td>0.77s</td>
<td>instant</td>
</tr>
<tr>
<td>12</td>
<td>21hrs</td>
<td>0.16s</td>
</tr>
<tr>
<td>15</td>
<td>2.4yrs</td>
<td>2.7s</td>
</tr>
<tr>
<td>18</td>
<td>2.4mill</td>
<td>92s</td>
</tr>
</tbody>
</table>
```

after a few minutes of computing...

Note. Can’t break RSA this way (experts are still trying).

Debugging

Programming. A process of finding and fixing mistakes.

1. Create the program.
2. Compile it.
   Compiler says: That’s not a legal program.
   Back to step 1 to fix syntax errors.
3. Execute it.
   Result is bizarrely (or subtly) wrong.
   Back to step 1 to fix semantic errors.
4. Enjoy the satisfaction of a working program!
5. Too slow? Back to step 1 to try a different algorithm.
U.S.S. Grace Murray Hopper