**Debugging: Success?**

**Success?** Program seems to work.
- Remove trace to try larger inputs.
- [stay tuned].

```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();
    }
}
```

**Debugging: Performance Errors**

**Performance error.** Correct program, but too slow.
- Are all iterations of inner loop necessary?
- 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++)
        {
            // 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();
    }
}
```

**Program Development: Analysis**

**Q.** How large an integer can I factor?

```java
% java Factors 3757208
2 2 7 13 13 397
% java Factors 9201111167975555703
9201111167975555703
```

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

5. Too slow? Back to step 1 to try a different algorithm.

1.4 Arrays

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
- assignment statements
- primitive types

Arrays

This lecture. Store and manipulate huge quantities of data.

Array. Indexed sequence of values of the same type.

Examples.
- 52 playing cards in a deck.
- 5 thousand undergrads at Princeton.
- 1 million characters in a book.
- 10 million audio samples in an MP3 file.
- 4 billion nucleotides in a DNA strand.
- 73 billion Google queries per year.
- 50 trillion cells in the human body.
- \(6.02 \times 10^{23}\) particles in a mole.
// Tedious and error-prone code.
double a0, a1, a2, a3, a4, a5, a6, a7, a8, a9;
a0 = 0.0;
a1 = 0.0;
a2 = 0.0;
a3 = 0.0;
a4 = 0.0;
a5 = 0.0;
a6 = 0.0;
a7 = 0.0;
a8 = 0.0;
a9 = 0.0;
...
a4 = ... 
...
a8 = ... 
...
double x = a4 + a8;

double a[10];
...
a[4] = ... 
...
a[8] = ... 

double x = a[4] + a[8];

double a[1000];
...
a[432] = ... 
...
a[811] = ... 
...
double x = a[432] + a[811];
Arrays in Java

Java has special language support for arrays.
- To make an array: declare, create, and initialize it.
- Array indices start at 0.
  ```java
  int N = 1000;
  double[] a;
  a = new double[N];
  for (int i = 0; i < N; i++)
    a[i] = 0.0;
  // all to 0.0
  ```

Compact alternative: Declare, create, and initialize in one statement.
- Version 1: all entries automatically set to 0 at run time.
  ```java
  int N = 1000;
  double[] a = new double[N];
  ```
- Version 2: entries initialized to given literal values at compile time.
  ```java
  double[] x = { 0.3, 0.6, 0.1 };  
  ```

Array Processing Examples

- create an array with N random values
  ```java
  double[] a = new double[N];
  for (int i = 0; i < N; i++)
    a[i] = Math.random();
  ```

- find the maximum of the array values
  ```java
  double max = Double.NEGATIVE_INFINITY;
  for (int i = 0; i < N; i++)
    if (a[i] > max) max = a[i];
  ```

- print the array values, one per line
  ```java
  for (int i = 0; i < N; i++)
    System.out.println(a[i]);
  ```

- compute the average of the array values
  ```java
  double sum = 0.0;
  double average = sum / N;
  ```

- copy to another array
  ```java
  double[] b = new double[N];
  for (int i = 0; i < N; i++)
    b[i] = a[i];
  ```

- reverse the elements within the array
  ```java
  for (int i = 0; i < N/2; i++)
    { double temp = b[i];
      b[i] = b[N-1-i];
      b[N-i-1] = temp;
    }
  ```

Dot product. Given two vectors x[] and y[] of length n, their dot product is the sum of the products of their corresponding components.

<table>
<thead>
<tr>
<th>i</th>
<th>x[i] y[i] x[i]*y[i] sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.30 0.50 0.15 0.15</td>
</tr>
<tr>
<td>1</td>
<td>0.60 0.10 0.06 0.21</td>
</tr>
<tr>
<td>2</td>
<td>0.10 0.40 0.04 0.25</td>
</tr>
</tbody>
</table>

Shuffling a Deck
Shuffling

**Goal.** Given an array, rearrange its elements in random order.

**Shuffling algorithm.**

- In iteration `i`, pick random card from `deck[i]` through `deck[N-1]`, with each card equally likely.
- Exchange it with `deck[i]`.

```java
int N = deck.length;
for (int i = 0; i < N; i++)
    { int r = i + (int) (Math.random() * (N-i));
      String t = deck[r];
      deck[r] = deck[i];
      deck[i] = t;
    }
```

**Array Challenge 1**

The following code sets array values to the 52 card values and prints them.

What order are they printed?

```
String[] deck = new String[52];
for (int i = 0; i < 13; i++)
    for (int j = 0; j < 4; j++)
        deck[4*i + j] = rank[i] + " of " + suit[j];
for (int i = 0; i < 52; i++)
    System.out.println(deck[i]);
```

**A.** 2 of clubs  
2 of diamonds  
2 of hearts  
2 of spades  
3 of clubs

**B.** 2 of clubs  
3 of clubs  
4 of clubs  
5 of clubs  
6 of clubs

**Shuffling a Deck of Cards**

```java
public class Deck
    {
        public static void main(String[] args)
        {
            String[] suit = { "Clubs", "Diamonds", "Hearts", "Spades" };
            String[] rank = { "2", "3", "4", "5", "6", "7", "8", "9",
            "10", "Jack", "Queen", "King", "Ace" };
            int SUITS = suit.length;
            int RANKS = rank.length;  \ **avoid "hardwired" constants like 52, 4, and 13.
            int N = SUITS * RANKS;

            String[] deck = new String[N];
            build the deck
            for (int i = 0; i < RANKS; i++)
                for (int j = 0; j < SUITS; j++)
                    deck[SUITS*i + j] = rank[i] + " of " + suit[j];

            shuffle
            for (int i = 0; i < N; i++)
                { int r = i + (int) (Math.random() * (N-i));
                  String t = deck[r];
                  deck[r] = deck[i];
                  deck[i] = t;
                }

            print shuffled deck
            for (int i = 0; i < N; i++)
                System.out.println(deck[i]);
        }
    }
```
Coupon Collector Problem

**Coupon collector problem.** Given \( N \) different card types, how many do you have to collect before you have (at least) one of each type?

**Simulation algorithm.** Repeatedly choose an integer \( i \) between 0 and \( N-1 \). Stop when we have at least one card of every type.

**Q.** How to check if we’ve seen a card of type \( i \)?

**A.** Maintain a boolean array so that \( \text{found}[i] \) is true if we’ve already collected a card of type \( i \).

Coupon Collector: Java Implementation

```java
public class CouponCollector {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int cardcnt = 0; // number of cards collected
        int valcnt = 0; // number of distinct cards

        // Do simulation.
        boolean[] found = new boolean[N];
        while (valcnt < N) {
            int val = (int) (Math.random() * N);
            cardcnt++;
            if (!found[val]) {
                valcnt++;
                found[val] = true;
            }
        }

        // all N distinct cards found
        System.out.println(cardcnt);
    }
}
```
**Debugging.** Need code to print contents of all variables.

<table>
<thead>
<tr>
<th>val</th>
<th>found</th>
<th>valcnt</th>
<th>cardcnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>F</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>T</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>T</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>T</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>T</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

**Challenge.** Debugging with arrays requires tracing many variables.

**Coupon Collector: Scientific Context**

**Q.** Given a sequence from nature, does it have same characteristics as a random sequence?

**A.** No easy answer - many tests have been developed.

**Coupon collector test.** Compare number of elements that need to be examined before all values are found against the corresponding answer for a random sequence.
Two Dimensional Arrays

Two dimensional arrays.

- Table of data for each experiment and outcome.
- Table of grades for each student and assignments.
- Table of grayscale values for each pixel in a 2D image.

Mathematical abstraction. Matrix.
Java abstraction. 2D array.

Setting 2D Array Values at Compile Time

Initialize 2D array by listing values.

double[][] p =
{  
  { .02, .92, .02, .02, .02 },  
  { .02, .02, .32, .32, .32 },  
  { .92, .02, .02, .02, .02 },  
  { .47, .02, .47, .02, .02 },
};

Matrix Addition

Matrix addition. Given two N-by-N matrices a and b, define c to be the N-by-N matrix where c[i][j] is the sum a[i][j] + b[i][j].

double[][] c = new double[N][N];
for (int i = 0; i < N; i++)
  for (int j = 0; j < N; j++)
    c[i][j] = a[i][j] + b[i][j];
Matrix Multiplication

Matrix multiplication. Given two N-by-N matrices $a$ and $b$, define $c$ to be the N-by-N matrix where $c[i][j]$ is the dot product of the $i^{th}$ row of $a$ and the $j^{th}$ row of $b$.

Application: 2D Random Walks

Array Challenge 2

How many multiplications to multiply two N-by-N matrices?

A. $N$
B. $N^2$
C. $N^3$
D. $N^4$

Application: Self-Avoiding Walks
**Self-Avoiding Walk**

**Model.**
- N-by-N lattice.
- Start in the middle.
- Randomly move to a neighboring intersection, avoiding all previous intersections.
- Two possible outcomes: escape and dead end

**Applications.** Polymers, statistical mechanics, etc.

**Q.** What fraction of time will you escape in a 5-by-5 lattice?
**Q.** In an N-by-N lattice?
**Q.** In an N-by-N-by-N lattice?

```java
public class SelfAvoidingWalk {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);  // lattice size
        int T = Integer.parseInt(args[1]);  // number of trials
        int deadEnds = 0;  // trials ending at dead end

        for (int t = 0; t < T; t++) {
            boolean[][] a = new boolean[N][N];  // intersections visited
            int x = N/2, y = N/2;  // current position

            while (x > 0 && x < N-1 && y > 0 && y < N-1) {
                if (a[x+1][y] && a[x-1][y] && a[x][y+1] && a[x][y-1]) {
                    deadEnds++;  break; }
                a[x][y] = true;  // mark as visited
                double r = Math.random();
                if (r < 0.25) { if (!a[x+1][y]) x++; }
                else if (r < 0.50) { if (!a[x-1][y]) x--; }
                else if (r < 0.75) { if (!a[x][y+1]) y++; }
                else if (r < 1.00) { if (!a[x][y-1]) y--; }
            }
        }
        System.out.println(100*deadEnds/T + " % dead ends");
    }
}
```

**Arrays.**
- Organized way to store huge quantities of data.
- Almost as easy to use as primitive types.
- Can directly access an element given its index.

**Ahead.** Reading in large quantities of data from a file into an array.