1.4 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
- 10 thousand undergrads at Penn
- 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 × 10^{23} particles in a mole

Many Variables of the Same Type

Goal: 10 variables of the same type

// tedious and error-prone
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 = 3.0;
a8 = 8.0;
double x = a4 + a8;

// easy alternative
double[] a = new double[10];
a[4] = 3.0;
a[8] = 8.0;
double x = a[4] + a[8];

Many Variables of the Same Type

Goal: 1 million variables of the same type

// scales to handle large arrays
double[] a = new double[1000000];
a[123456] = 3.0;
a[987654] = 8.0;
double x = a[123456] + a[987654];
Arrays in Java

Java has special language support for arrays

- To make an array: declare, create, and initialize it
- To access entry \(i\) of array named \(a\), use \(a[i]\)
- Array indices start at 0

```java
int N = 10; // size of array
double[] x = new double[N]; // declare the array
for (int i = 0; i < N; i++) // initialize the array
    x[i] = 0.0; // all to 0.0
```

## Compact alternative
- Declare, create, and initialize in one statement
- Default initialization: all numbers automatically set to zero

```java
int N = 10; // size of array
double[] x = new double[N]; // declare, create, init
```

Vector Dot Product

**Dot product** Given two vectors \(x[]\) and \(y[]\) of length \(N\), their dot product is the sum of the products of their corresponding components

```java
double[] x = {0.3, 0.6, 0.1};
double[] y = {0.3, 0.1, 0.4};
int N = x.length;
double sum = 0.0;
for (int i = 0; i < N; i++) {
    sum = sum + x[i]*y[i];
}
```

<table>
<thead>
<tr>
<th>(i)</th>
<th>(x[i])</th>
<th>(y[i])</th>
<th>(x[i]*y[i])</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.3</td>
<td>0.3</td>
<td>0.07</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.6</td>
<td>0.1</td>
<td>0.06</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>0.4</td>
<td>0.04</td>
<td>0.11</td>
</tr>
<tr>
<td>3</td>
<td>0.2</td>
<td>0.0</td>
<td>0.00</td>
<td>0.11</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td>0.3</td>
<td>0.03</td>
<td>0.14</td>
</tr>
</tbody>
</table>
```

Array-Processing Examples

- Create an array with random values
- Print the array values, one per line
- Find the maximum of the array values
- Compute the average of the array values
- Copy to another array
- Reverse the elements within an array

Shuffling a Deck

**Ex.** Print a random card

```java
String[] rank = {
    "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace"
};
String[] suit = {
    "Clubs", "Diamonds", "Hearts", "Spades"
};
int i = (int) (Math.random() * 13); // between 0 and 12
int j = (int) (Math.random() * 4); // between 0 and 3
System.out.println(rank[i] + " of " + suit[j]);
```
Setting Array Values at Run Time

**Example:** Create a deck of playing cards and print them out

```java
String[] deck = new String[52];
for (int i = 0; i < 52; i++)
    deck[i] = suit[i] + rank[i] + " of " + suit[j];
```

Q. In what order does it output them?

A. two of clubs  B. two of clubs
   two of diamonds  three of clubs
   two of hearts  four of clubs
   three of clubs  five of clubs
   ...  ...

Shuffling a Deck of Cards: Putting Everything Together

```java
public class Deck {
    public static void main(String[] args) { //build the deck
        int SUITS = 4; int RANKS = 13; // avoid "hardwired" constants
        String[] suit = {"Clubs", "Diamonds", "Hearts", "Spades"};
        String[] rank = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace"};
        int N = SUITS * RANKS;
        String[] deck = new String[N];
        for (int i = 0; i < N; i++) //shuffle
            String t = deck[i];
        for (int i = 0; i < N; i++) //print shuffled deck
            System.out.println(deck[i]);
    }
}
```

Shuffling a Deck of Cards

```java
public class Deck {
    public static void main(String[] args) { //build the deck
        int SUITS = 4; int RANKS = 13; // avoid "hardwired" constants
        String[] suit = {"Clubs", "Diamonds", "Hearts", "Spades"};
        String[] rank = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace"};
        int N = SUITS * RANKS;
        String[] deck = new String[N];
        for (int i = 0; i < N; i++) //shuffle
            String t = deck[i];
    }
}
```

War Story (PlanetPoker.com)

Texas hold’em poker. Software must shuffle electronic deck of cards

Coupon Collector

How we learned to cheat at online poker: a study in software security
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);
    }
}
```

**Coupon Collector: Debugging**

**Debugging** Add 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>F</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>F</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

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

**Coupon Collector: Mathematical 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.

**Multidimensional Arrays**
Two-Dimensional Arrays in Java

Array access: Use $a[i][j]$ to access entry in row $i$ and column $j$.

Zero-based indexing: Row and column indices start at 0.

```java
int M = 10;
int N = 3;
double[][] a = new double[M][N];
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        a[i][j] = 0.0;
    }
}
```

A 10-by-3 array

### Setting 2D Array Values at Compile Time

Initialize 2D array by listing values

```java
double[][] p = {
    {0.02, 0.02, 0.02, 0.02},
    {0.02, 0.02, 0.02, 0.02},
    {0.02, 0.02, 0.02, 0.02},
    {0.47, 0.47, 0.47, 0.47}};
```

### 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 $a[i][j] + b[i][j]$.

```java
double[][] c = new double[M][N];
for (int i = 0; i < M; 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 $a[i][j]$ and the $j$th column of $b[j][j]$.

```java
double[][] a = new double[M][N];
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        c[i][j] = a[i][j] * b[j][j];
    }
}
```

### Array Challenge

**Q.** How many scalar multiplications multiply two N-by-N matrices?

- A. $N^2$
- B. $N^3$
- C. $N^4$
- D. $N^6$

```java
double[][] c = new double[M][N];
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        c[i][j] = a[i][j] * b[j][j];
    }
}
```
Summary

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