Readings
This Week: No new readings. Consolidate!

(Reminder: Readings are absolutely vital for learning this stuff!)

Parameter Passing Exercises
Lecture 34

Labs and Tutorials
This Week: Lab #10
Labs are due at end of lab time! (Lab #10 is fairly short, but make sure to finish by the end of lab.)

Final Exam
- Wednesday, April 15, 7pm in SRC A
- This wasn’t a good room last year, but we’re stuck with the date, time, and room UBC assigns.
- Note: You are allowed to do scratch work on the exam paper! (But write your answer where indicated if the problem tells you to.)

Programming Assignment 3
- Assignment 3 is up on WebCT!
  - Click on the "Assignments" icon.
  - Assigned Sunday evening – sorry for delay!
  - Due at NOON, April 6 (Monday), via electronic hand in.
- Tips:
  - There is some Eclipse setup. Set-up ASAP!
  - Work in pairs. Some conceptual stuff.
  - Think carefully before coding. If concepts right, the coding is much much easier.

Survey #3
- Ben Yu's 3rd survey will be up on WebCT soon.
- This is an end-of-term survey, but we won’t have lectures for me to remind you.
- Keep an eye out on WebCT…
Learning Goals
By the end of class today you will be able to...

- Work through complicated examples of parameter passing and arrays. (Review)

- Apply some basic techniques to help get started when facing a new problem.

Review: Passing Parameters

In the object user (the caller):
```
  a.flatter("fabulous");
```

In the class definition (the callee):
```
public void flatter(String adjective) {
  System.out.println("Wow, you look "+
      adjective + "+!");
}
```

Parameter Passing Challenge #1
Consider the following method:
```
public void swap( int a, int b )
{
  int temp = a;
  a = b;
  b = temp;
}
```
and the following code from main that calls swap:
```
int x = 0;
int y = 5;
swap( x, y );
```
What values are stored in the x and y after the above code segment has executed?

Parameter Passing Challenge #2
Consider the following method:
```
public void process( int[][] arrA, int[][] arrB )
{
  int row, col;
  int[][] arrC = new int[][] { { 1, 1, 1 }, { 1, 1, 1 } };
  arrA = arrC;
  for( row = 0; row < arrB.length; row++ )
    for( col = 0; col < arrB[row].length; col++ )
      arrB[row][col] = row + col;
}
```
and the following code from main that calls process:
```
int[][] dataA = new int[][] { { 0, 0 }, { 0, 0 } };
int[][] dataB = new int[][] { { 0, 0 }, { 0, 0 } };
process( dataA, dataB );
```
What values are stored in the arrays dataA and dataB after the above code segment has executed?

Just as process is called
Just before process's first line

Just after arrC is initialized

Just after “arrA = arrC;”

Just as process completes

After returning to main

Parameter Passing Intuition
- Calling a method is like asking a friend to do something for you.
- Passing parameters is giving your friend the information needed to do the task.
- In Java, parameters are always “call by value” (sometimes “pass by value”):
  - You make a copy of the info for your friend.
  - Java makes a copy of the parameters (the value of the parameters) for the method call.
- Passing primitive type values vs. objects?
Real Life Analogy

- You show me a $20 bill. I make a photocopy. Can I spend your money? If I burn my photocopy, do I destroy your $20 bill?
- You show me your credit card. I make a photocopy. Can I spend your money? If I burn my photocopy, do I destroy your credit card?
- $20 bill is like a primitive type. It is the value.
- Credit card is like an object reference. It says how to find the value (your credit line).

Thoughts

In every case, the variables in main still stored the same values they did before. Java is “pass-by-value”; we only pass the values of arguments to the parameters of a method. So, we can never change those arguments.

But... if the variable is a reference variable, the object it points to can be changed! That’s what happened to dataB.

Why didn’t dataA change?

Parameter Passing Challenge #3

Consider the following method:

```java
public void swap( int[][] a, int[][] b )
{
    int[][] temp = a;
    a = b;
    b = temp;
}
```

and the following code that calls swap:

```java
int[][] x = new int[][]{{0, 0}, {0, 0}};
int[][] y = new int[][]{{1, 1}, {1, 1}};
swap( x, y );
```

What values are stored in the x and y after the above code segment has executed?

Just as swap is called

Just before swap’s first line

Just as swap completes
How do I get swap to work?

- Java doesn’t really have a good way to do this. How do I swap the contents of two pockets if you give me just copies of the contents?
- With object references, I can kind of do it:
  - I can’t swap your references, but I can follow the object references to the objects and change the contents of the objects.
- Other languages (like C++) let you create references to any variable (even primitive types).
  - If two of you give me (a copy of) the locations of your dorm rooms, I can swap the contents.
  - This is called “call by reference” or “pass by reference”.

Something like swap

```java
private static void swap(int[][] a, int[][] b) {
    assert a.length == b.length;
    int[][] temp = new int[a.length][];
    for (int i=0; i<a.length; i++) {
        temp[i] = a[i];
    }
    for (int i=0; i<b.length; i++)
        a[i] = b[i];
    b[i] = temp[i];
}
```

and the following code that calls swap:

```java
int[][] x = new int[][] {{0, 0}, {0, 0}};
int[][] y = new int[][] {{1, 1}, {1, 1}};
swap(x, y);
```

What values are stored in the x and y after the above code segment has executed?

Questions?

Problem Solving

- The ability to solve problems is one of the most important skills to acquire. (Not just in CPSC 111!)
- But it’s rarely taught explicitly:
  - Because it requires creative insight, there’s no cookbook recipe that always works.
  - Usual approach is to practice solving problems, until you get better at it.
- Again, there’s no recipe that always works
- Some basic techniques to get started:
  - Make sure you know what’s given and what kind of result you’re supposed to provide.
  - Try to solve the problem by hand, before programming
    - Work lots of small examples.
    - Think closely about what you did to solve the problem.
  - Think about what kinds of data to store, what kinds of operations need to be done to the data
    - This suggests objects and methods
  - Break things down into steps:
    - A sequence of steps becomes a sequence of statements
    - Look for repetition (loops), decisions (if)
    - Use helper methods to break out common sub-problems.
Example: Sudoku Solution Checker

- Sudoku is a popular logic puzzle:
  - The board is a 9x9 array of squares, with some squares filled in with digits from 1…9
  - The task is to fill in the rest of the squares with digits 1…9, so that:
    - Each row contains the numbers 1…9
    - Each column contains the numbers 1…9
    - Each 3x3 “zone” contains the numbers 1…9
- Our challenge is to write a program that checks whether a solution is valid.

Example: Sudoku Solution Checker

- Write a method checkSudoku() that takes a 9x9 array of int, and returns true if and only if the array is a legal Sudoku solution.

Example: A correct solution...

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>8</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>8</td>
<td>2</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>

Is this one OK?

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>9</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>4</td>
<td>8</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

How about this one?

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>6</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>9</td>
<td>8</td>
<td>6</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>9</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

And this one?

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>9</td>
<td>8</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
### Problem Solving

- Again, there’s no recipe that always works
- Some basic techniques to get started:
  - **Make sure you know what’s given and what kind of result you’re supposed to provide.**
  - **Try to solve the problem by hand, before programming**
    - Work lots of small examples.
    - Think closely about what you did to solve the problem.
  - **Think about what kinds of data to store, what kinds of operations need to be done to the data**
    - This suggests objects and methods
  - Break things down into steps:
    - A sequence of steps becomes a sequence of statements
    - Look for repetition (loops), decisions (if)
    - Use helper methods to break out common sub-problems.

### Problem Solving

- Again, there’s no recipe that always works
- Some basic techniques to get started:
  - **Make sure you know what’s given and what kind of result you’re supposed to provide.**
  - **Try to solve the problem by hand, before programming**
    - Work lots of small examples.
    - Think closely about what you did to solve the problem.
  - **Think about what kinds of data to store, what kinds of operations need to be done to the data**
    - This suggests objects and methods
  - Break things down into steps:
    - A sequence of steps becomes a sequence of statements
    - Look for repetition (loops), decisions (if)
    - Use helper methods to break out common sub-problems.

### Problem Solving

- Again, there’s no recipe that always works
- Some basic techniques to get started:
  - **Make sure you know what’s given and what kind of result you’re supposed to provide.**
  - **Try to solve the problem by hand, before programming**
    - Work lots of small examples.
    - Think closely about what you did to solve the problem.
  - **Think about what kinds of data to store, what kinds of operations need to be done to the data**
    - This suggests objects and methods
  - Break things down into steps:
    - A sequence of steps becomes a sequence of statements
    - Look for repetition (loops), decisions (if)
    - Use helper methods to break out common sub-problems.

### A Harder Example: Sudoku Solver

- This is a much harder problem. It’s worth thinking about if you want more of a challenge…
- It will be useful to declare a SudokuBoard class:
  - You may want bookkeeping info along with the main grid of numbers.
  - You can then have methods like:
    - applyMove(int row, int col, int digit)
    - getLegalDigits(int row, int col)
    - copyBoard()