

CS 106B, Lecture 9

Recursive Data

Plan for Today

- More recursion practice!
- Learning goals for today
 - See examples of recursively structured data.

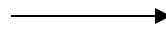
Recap: Recursion Tips

- Look for *self-similarity*
- Make the problem *simpler* by doing the least amount of work possible
- *Trust* the recursion
- Find a stopping point (*base case*)

reverseLines exercise

- Write a recursive function `reverseLines` that accepts a file input stream and prints the lines of that file in reverse order.
 - Example input file:

```
Roses are red,  
Violets are blue.  
All my base  
Are belong to you.
```



Expected console output:

```
Are belong to you.  
All my base  
Violets are blue.  
Roses are red,
```

Reversal pseudocode

- Reversing the lines of a file:
 - Read a line L from the file.
 - Print the rest of the lines in reverse order.
 - Print the line L.
- If only we had a way to reverse the rest of the lines of the file....

reverseLines solution

```
void reverseLines(ifstream& input) {  
    string line;  
    if (getline(input, line)) {  
        // recursive case  
        reverseLines(input);  
        cout << line << endl;  
    }  
}
```

- Where is the base case?

Stanford C++ files

```
#include "filelib.h"
```

Function	Description
<code>createDirectory(<i>name</i>)</code>	creates a new directory with given path name
<code>deleteFile(<i>name</i>)</code>	removes file from disk
<code>fileExists(<i>name</i>)</code>	whether this file exists on the disk
<code>getCurrentDirectory()</code>	returns directory the current C++ program runs in
<code>getExtension(<i>name</i>)</code>	returns file's extension, e.g. "foo.cpp" → ".cpp"
<code>getHead(<i>name</i>), getTail(<i>name</i>)</code>	separate a file path into the directory and file part; for "a/b/c/d.txt", head is "a/b/c", tail is "d.txt"
<code>isDirectory(<i>name</i>)</code>	returns whether this file name represents a directory
<code>isFile(<i>name</i>)</code>	returns whether this file name represents a regular file
<code>listDirectory(<i>name</i>)</code>	returns a <code>Vector<string></code> with the names of all files contained in the given directory
<code>readEntireFile(<i>name</i>, <i>v</i>)</code>	reads lines of the given file into a vector of strings
<code>renameFile(<i>old</i>, <i>new</i>)</code>	changes a file's name

crawl exercise

- Write a function `crawl` accepts a file name as a parameter and prints information about that file.
 - If the name represents a normal file, just print its name.
 - If the name represents a directory, print its name and information about every file/directory inside it, indented.

```
course
  handouts
    syllabus.doc
    lecture-schedule.xls
  homework
    1-gameoflife
      life.cpp
      life.h
      GameOfLife.pro
```

- **recursive data**: A directory can contain other directories.

Optional parameters

- We cannot vary the indentation without an extra parameter:

```
void crawl(string filename, string indent) {
```

- Often the parameters we need for our recursion do not match those the client will want to pass.

One solution is to use a *default parameter* value:

```
void crawl(string filename, string indent = "");
```

- The client can call `crawl` passing only one parameter.
- The recursive calls can pass the second parameter to indent.

crawl solution

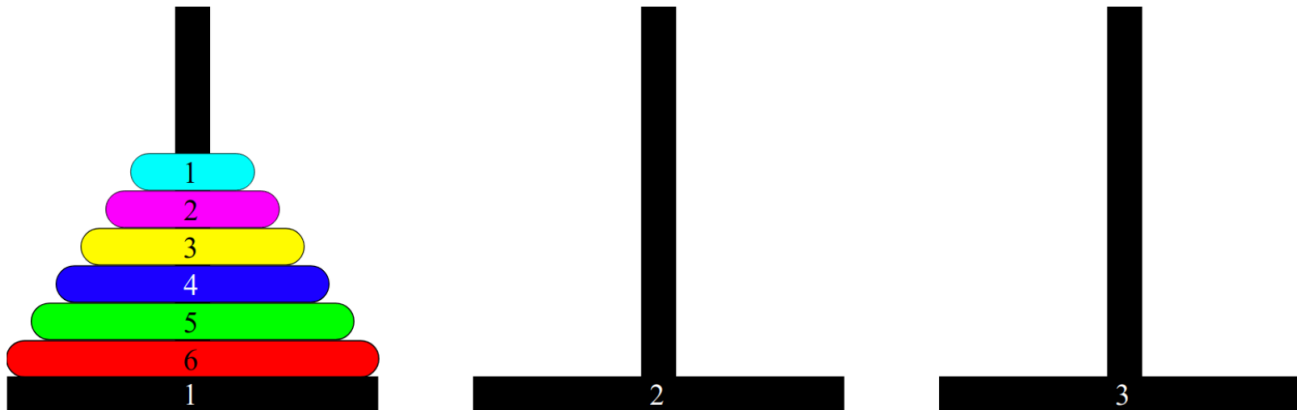
```
// Prints information about this file,  
// and (if it is a directory) any files inside it.  
void crawl(string filename, string indent = "") {  
    cout << indent << getTail(filename) << endl;  
    if (isDirectory(filename)) {  
        // recursive case; print contained files/dirs  
        Vector<string> filelist;  
        listDirectory(filename, filelist);  
        for (string subfile : filelist) {  
            crawl(filename + "/" + subfile,  
                  indent + "    ");  
        }  
    }  
}
```

Announcements

- My OH tomorrow are cancelled 😞
 - Come by today for OH!
 - LaIR is still open tonight and tomorrow
 - Can also post on Piazza for help
- HW2 is due tomorrow at 5PM

Towers of Hanoi

- The Towers of Hanoi puzzle asks a player to move a stack of discs from one peg to another, moving one disc at a time.
 - A disc cannot sit on top of a smaller disc.
- Write a recursive function `moveDiscs` with three parameters: number of discs, start peg, end peg, that moves that many discs from the start peg to the end peg.



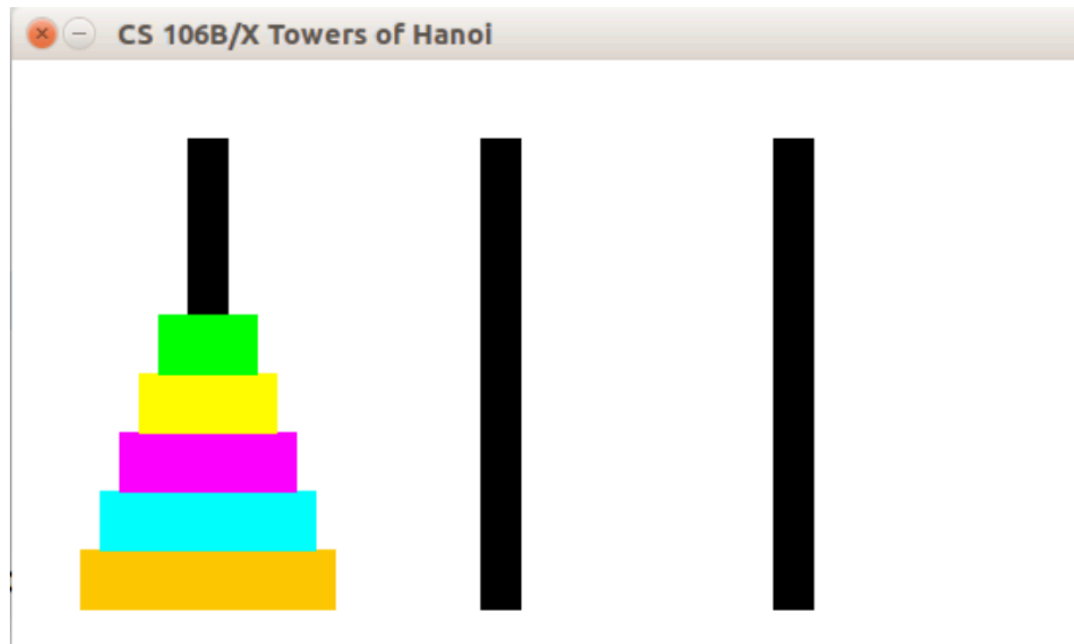
Towers of Hanoi

- Assume we have a HanoiGui with the following functions:

```
void initialize(int numDiscs)
```

```
void moveOneDisc(int startPeg, int endPeg)
```

```
int thirdPegNumber(int peg1, int peg2)
```



Hanoi solution

```
// Moves the given number of discs from the given
// starting peg to the given ending peg.
void moveDiscs(int numDiscs, int startPeg, int endPeg) {
    if (numDiscs > 0) {
        // move rest of discs
        int thirdPeg = HanoiGui::thirdPegNumber(startPeg,
            endPeg);
        moveDiscs(numDiscs - 1, startPeg, thirdPeg);
        // move remaining bottom disc
        HanoiGui::moveOneDisc(startPeg, endPeg);
        // move rest of discs
        moveDiscs(numDiscs - 1, thirdPeg, endPeg);
    }
    // else, implicit base case: do nothing
}
```

evenDigits exercise

- Write a recursive function `evenDigits` that accepts an integer and returns a new number containing only the even digits, in the same order. If there are no even digits, return 0.
 - Example: `evenDigits(8342116)` returns 8426
 - Example: `evenDigits(40109)` returns 400
 - Example: `evenDigits(8)` returns 8
 - Example: `evenDigits(-163505)` returns -60
 - Example: `evenDigits(35179)` returns 0

evenDigits solution

```
// Returns a new integer containing only the even-valued  
// digits from the given integer, in the same order.  
// Returns 0 if there are no even digits.
```

```
int evenDigits(int n) {  
    if (n < 0) {  
        return -evenDigits(-n);  
    } else if (n == 0) {  
        return 0;  
    } else if (n % 2 == 0) {  
        return 10 * evenDigits(n / 10) + n % 10;  
    } else {  
        return evenDigits(n / 10);  
    }  
}
```