Chapter 2: Recursion: The Mirrors

Vocabulary

Write out a description of each vocabulary term for reference later.

Term	**Description**
Recursion	
Base case	
Iteration	
Recursion	

Concepts

Example: Recursion in math

A factorial is a good way to show an example of recursion. Let's say we have the equation that needs to be simplified…:

$$\frac{100!}{(100-2)!}$$

This can be further simplified to:

$$\frac{100!}{98!}$$

We can simplify this fraction further without having to calculate 100! and 95! ; if we break down these factorials, we can write them as:

$$\frac{100\cdot 99\cdot 98\cdot 97\cdot 96\cdot \ldots \cdot 3\cdot 2\cdot 1}{98\cdot 97\cdot 96\cdot \ldots \cdot 3\cdot 2\cdot 1}$$

In the denominator, everything can be canceled out. In the numerator, this would leave just 100 x 99. Perhaps by writing out the factorial like this, you can see that *100!* is equivalent to:

100 x 99 x 98!

The result of any *n!* , *n*-factorial, will be *n* times *(n-1)!*

Example: Recursion and the file system

Let's say that you're writing a program that will search through the hard drive searching for a file. You might start off first by implementing something simpler…

- Search the current directory for the file.

And, once that is finished, you can expand it to work past the current directory…

- For every folder in the directory…
    - Search inside that subfolder

The "Search inside that subfolder" routine will perform the same functionality as "Search the current directory for the file", except instead of "current directory", we're looking at "subfolder *x*". At this point, we should generalize our "search the directory for file" functionality to work with any folder passed in…

```
bool IsFileFound( Directory searchHere, string fileName )
```

While you might look at each file in an iterative manner to try to find your file, it doesn't make sense to try to *iterate* through all folders on the hard-drive. Instead, you can "walk" through all the folders by using recursion.

```
bool IsFileFound( Directory searchHere, string fileName )
   for each item in the directory searchHere…
      if item is a directory…
         call IsFileFound( item, fileName )
      else, if item is a file…
         if this item matches fileName, return true.
         else, keep going.
```

The highlighted line is the *recursive case*, calling the same function again. If we've found that `item` is a directory (not a file), then we tell the function to begin searching in *that* directory. Once that directory is done being searched, we will return back to the original execution of this function.

Examples

Recursion can be a difficult concept to grasp; for many people it is just easier to design a solution using an iterative approach (with a loop), while doing the same task can be challenging using recursion.

Here are some examples of functions that achieve the same thing, both iteratively and recursively:

Counting up

Counts from *start* to *end*…

Function definition – iterative

```
void CountUp_Iter( int start, int end )
{
    for ( int i = start; i <= end; i++ )
    {
        cout << i << "\t";
    }
}
```

Function definition – recursive

```
void CountUp_Rec( int start, int end )
{
    cout << start << "\t";

    // Terminating case
    if ( start == end )
        return;

    // recursive case
    CountUp_Rec( start+1, end );
}
```

Factorial

Calculate *n!*…

Function definition – iterative

```
int Factorial_Iter( int n )
{
    for ( int i = n-1; i > 0; i-- )
    {
        n *= i;
    }
    return n;
}
```

Function definition – recursive

```
int Factorial_Rec( int n )
{
    // Terminating case
    if ( n == 0 )
    {
        return 1;
    }

    // Recursive case
    return n * Factorial_Rec( n-1 );
}
```