EECS 161
Intro to Programming I

More Functions and Recursion
Chap. 13.1
Recursion

• What is it?
  – Function that calls itself 1 or more times
  – Has a base case for stopping
Example: Drawing Rectangles

• Iterative Solution:

```cpp
void draw_rect(int i) {
    for( ; i > 0; i--){
        cout << "******" << endl;
        cout << "*       *
            " << endl;
        cout << "******" << endl << endl;
    }
}
```
Example: Drawing Rectangles

• Recursive Solution

```cpp
void draw_rect(int i) {
    if(i>0){  // Base case
        draw_rect(--i);  // Recursive call
        cout << "*****" << endl;
        cout << "* *" << endl;
        cout << "*****" << endl << endl;
    }
}
```
What is different when we call after?

• Recursive Solution

```cpp
void draw_rect(int i) {
    if(i>0){  //Base case
        cout << "******" << endl;
        cout << "*    *" << endl;
        cout << "******" << endl
        << endl;
        draw_rect(--i);  //Recursive call
    }
}
```
Example: Factorial

• Definition

\[ 0! = 1; \]
\[ n! = n \times (n-1) \times \ldots \times (n-(n-1)) \times 1 = n \times (n-1)! \quad ; \quad n > 0 \]
Iterative Factorial

factorial(0) = 1;
factorial(n) = n*n-1*n-2*...*n-(n-1)*1;

long factorial(int n)
{
    long fact;
    if(n==0)
        fact=1;
    else
        for(fact=n; n > 1; n--)
            fact=fact*(n-1);
    return fact;
}
Recursive Factorial

factorial(0) = 1; // Base case
factorial(n) = n*factorial(n-1);

long factorial(int n) {
    if (n == 0) return 1; // Base case
    else
        return n * factorial(n - 1);
}
Computing Factorial Iteratively

factorial(4)

factorial(0) = 1;
factorial(n) = n * (n-1) * ... * 2 * 1;
Computing Factorial Iteratively

factorial(4) = 4 * 3

factorial(0) = 1;
factorial(n) = n*(n-1)*...*2*1;
Computing Factorial Iteratively

factorial(4) = 4 * 3
= 12 * 2

factorial(0) = 1;
factorial(n) = n*(n-1)*...*2*1;
Computing Factorial Iteratively

factorial(4) = 4 * 3
  = 12 * 2
  = 24 * 1

factorial(0) = 1;
factorial(n) = n*(n-1)*...*2*1;
Computing Factorial Iteratively

factorial(4) = 4 * 3
    = 12 * 2
    = 24 * 1
    = 24

factorial(0) = 1;
factorial(n) = n*(n-1)*...*2*1;
Computing Factorial Recursively

factorial(4)

factorial(0) = 1;
factorial(n) = n * factorial(n-1);
Computing Factorial Recursively

\[
\text{factorial}(4) = 4 \times \text{factorial}(3)
\]

\[
\begin{align*}
\text{factorial}(0) &= 1; \\
\text{factorial}(n) &= n \times \text{factorial}(n-1);
\end{align*}
\]
Computing Factorial Recursively

\[ \text{factorial}(4) = 4 \times \text{factorial}(3) \]
\[ = 4 \times (3 \times \text{factorial}(2)) \]
Computing Factorial Recursively

factorial(4) = 4 * factorial(3)
  = 4 * ( 3 * factorial(2))
  = 4 * ( 3 * (2 * factorial(1)))

factorial(0) = 1;
factorial(n) = n*factorial(n-1);
Computing Factorial Recursively

factorial(4) = 4 * factorial(3)
= 4 * ( 3 * factorial(2))
= 4 * ( 3 * (2 * factorial(1)))
= 4 * ( 3 * ( 2 * (1 * factorial(0))))
Computing Factorial Recursively

factorial(4) = 4 * factorial(3)
= 4 * (3 * factorial(2))
= 4 * (3 * (2 * factorial(1)))
= 4 * (3 * (2 * (1 * factorial(0))))
= 4 * (3 * (2 * (1 * 1)))

factorial(0) = 1;
factorial(n) = n*factorial(n-1);
Computing Factorial Recursively

\[
\begin{align*}
\text{factorial}(4) &= 4 \times \text{factorial}(3) \\
&= 4 \times (3 \times \text{factorial}(2)) \\
&= 4 \times (3 \times (2 \times \text{factorial}(1))) \\
&= 4 \times (3 \times (2 \times (1 \times \text{factorial}(0)))) \\
&= 4 \times (3 \times (2 \times (1 \times 1))) \\
&= 4 \times (3 \times (2 \times 1))
\end{align*}
\]
Computing Factorial Recursively

factorial(4) = 4 * factorial(3)
= 4 * (3 * factorial(2))
= 4 * (3 * (2 * factorial(1)))
= 4 * (3 * (2 * (1 * factorial(0))))
= 4 * (3 * (2 * (1 * 1)))
= 4 * (3 * (2 * 1))
= 4 * (3 * 2)

factorial(0) = 1;
factorial(n) = n*factorial(n-1);
Computing Factorial Recursively

\[
\text{factorial}(4) = 4 \times \text{factorial}(3) \\
= 4 \times (3 \times \text{factorial}(2)) \\
= 4 \times (3 \times (2 \times \text{factorial}(1))) \\
= 4 \times (3 \times (2 \times (1 \times \text{factorial}(0)))) \\
= 4 \times (3 \times (2 \times (1 \times 1))) \\
= 4 \times (3 \times (2 \times 1)) \\
= 4 \times (3 \times 2) \\
= 4 \times 6
\]
Computing Factorial Recursively

\[
\text{factorial}(4) = 4 \times \text{factorial}(3)
= 4 \times (3 \times \text{factorial}(2))
= 4 \times (3 \times (2 \times \text{factorial}(1)))
= 4 \times (3 \times (2 \times (1 \times \text{factorial}(0))))
= 4 \times (3 \times (2 \times (1 \times 1)))
= 4 \times (3 \times (2 \times 1))
= 4 \times (3 \times 2)
= 4 \times 6
= 24
\]

\[
\text{factorial}(0) = 1;
\text{factorial}(n) = n \times \text{factorial}(n-1);
\]
Differences

For recursion

• Pros
  – Readability

• Cons
  – Efficiency
  – Memory
Recursive Factorial

factorial(4) → Executes factorial(4)

Stack

Main method
Recursive Factorial

factorial(4)

Step 0: executes factorial(4)

return 4 * factorial(3)

Executes factorial(3)

Stack

Space Required for factorial(4)
Main method
Recursive Factorial

factorial(4)

Step 0: executes factorial(4)

return 4 * factorial(3)

Step 1: executes factorial(3)

return 3 * factorial(2)

Executes factorial(2)

Space Required for factorial(3)

Space Required for factorial(4)

Main method
Recursive Factorial

Step 0: executes factorial(4)
return 4 * factorial(3)

Step 1: executes factorial(3)
return 3 * factorial(2)

Step 2: executes factorial(2)
return 2 * factorial(1)

Executes factorial(1)

Stack

- Space Required for factorial(2)
- Space Required for factorial(3)
- Space Required for factorial(4)
- Main method

OSU Oregon State University
Recursive Factorial

factorial(4)
Step 0: executes factorial(4)
return 4 * factorial(3)
Step 1: executes factorial(3)
return 3 * factorial(2)
Step 2: executes factorial(2)
return 2 * factorial(1)
Step 3: executes factorial(1)
return 1 * factorial(0)

Executes factorial(0)

Stack
- Space Required for factorial(1)
- Space Required for factorial(2)
- Space Required for factorial(3)
- Space Required for factorial(4)
- Main method
Recursive Factorial

Stack

Space Required for factorial(0)
Space Required for factorial(1)
Space Required for factorial(2)
Space Required for factorial(3)
Space Required for factorial(4)
Main method
Recursive Factorial

Step 0: executes factorial(4)
return 4 * factorial(3)

Step 1: executes factorial(3)
return 3 * factorial(2)

Step 2: executes factorial(2)
return 2 * factorial(1)

Step 3: executes factorial(1)
return 1 * factorial(0)

Step 4: executes factorial(0)

Step 5: return 1

returns factorial(0)

Main method

Stack
Space Required for factorial(1)
Space Required for factorial(2)
Space Required for factorial(3)
Space Required for factorial(4)
Main method
Recursive Factorial

Factorial(4)

Step 0: executes Factorial(4)

return 4 * Factorial(3)

Step 1: executes Factorial(3)

return 3 * Factorial(2)

Step 2: executes Factorial(2)

return 2 * Factorial(1)

Step 3: executes Factorial(1)

return 1 * Factorial(0)

Step 4: executes Factorial(0)

return 1

Step 5: return 1

Step 6: return 1

Stack

Space Required for Factorial(0)
Space Required for Factorial(1)
Space Required for Factorial(2)
Space Required for Factorial(3)
Main Method
Recursive Factorial

Main method

Space Required for factorial(4)
Space Required for factorial(3)
Main method

Stack

factorial(4)

Step 0: executes factorial(4)
return 4 * factorial(3)

Step 1: executes factorial(3)
return 2 * factorial(2)

Step 2: executes factorial(2)
return 2 * factorial(1)

Step 3: executes factorial(1)
return 1 * factorial(0)

Step 4: executes factorial(0)
return 1

Step 5: return 1

Step 6: return 1

Step 7: return 2
returns factorial(2)

return 1
Recursive Factorial

factorial(4)

returns factorial(3)

Step 0: executes factorial(4)

return 4 * factorial(3)

Step 1: executes factorial(3)

return 3 * factorial(2)

Step 2: executes factorial(2)

return 2 * factorial(1)

Step 3: executes factorial(1)

return 1 * factorial(0)

Step 4: executes factorial(0)

return 1

Step 5: return 1

Step 6: return 1

Step 7: return 2

Step 8: return 6

Stack

Space Required for factorial(4)

Main method

OSU Oregon State University
Recursive Factorial

Step 0: executes factorial(4)

Step 1: executes factorial(3)

Step 2: executes factorial(2)

Step 3: executes factorial(1)

Step 4: executes factorial(0)

Step 5: return 1

Step 6: return 1

Step 7: return 2

Step 8: return 6

Step 9: return 24

returns factorial(4)
Reading/Assignments

• Work on Assignment #3
• Read Chap. 13.1-13.2
• Start Reading Chap. 9