CS 161
Intro to CS I
Review Functions & Recursion
Chap. 4.1-4.2 & 13
Revisit Overload/Default Args

• How’s Assignment 3 going?
• What is the difference?
• Let’s look at pwr() again...
Overloading vs. Default Args

```cpp
#include <iostream>

using std::cout;
using std::endl;

//int pwr(int, int n=1);  //Example of default args
double pwr(double, double);  //Example of overloading

int main() {

    int base=2, expn=8;
    double dbase=2.2, dexp=8.0;

    //cout << "The power function: " << pwr(base, expn) << endl;
    //cout << "The power function: " << pwr(base) << endl;
    cout << "The power function: " << pwr(dbase) << endl;
    //cout << "Pow with int and double: " << pwr(2, 2.2) << endl;
    return 0;

}

double pwr(double x, double n = 1) {
    double num=1.0;
```
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 << "******\n";
        cout << "*         *\n";
        cout << "******\n"
        cout << "******\n\n";
    }
}
```
Example: Drawing Rectangles

• Recursive Solution

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

- Definition
  
  0! = 1;
  
  n! = n * (n-1) * ... * 2 * 1 = n * (n-1)! ; n > 0
Iterative Factorial

factorial(0) = 1;
factorial(n) = n \cdot n-1 \cdot n-2 \cdot \ldots \cdot 2 \cdot 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;
factorial(n) = n*factorial(n-1);

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

\[
\text{factorial}(0) = 1; \\
\text{factorial}(n) = n \times (n-1) \times ... \times 2 \times 1;
\]

\text{factorial}(4)
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

\[
\text{factorial}(n) = n \times (n-1) \times \ldots \times 2 \times 1;
\]

\[
\text{factorial}(4) = 4 \times 3
\]
\[
= 12 \times 2
\]
\[
= 24 \times 1
\]
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(0) = 1;
factorial(n) = n * factorial(n-1);

factorial(4)
Computing Factorial Recursively

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

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

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

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

factorial(4) = 4 \times \text{factorial}(3)
\quad = 4 \times (3 \times \text{factorial}(2))
\quad = 4 \times (3 \times (2 \times \text{factorial}(1)))

\text{factorial}(0) = 1;
\text{factorial}(n) = n \times \text{factorial}(n-1);
Computing Factorial Recursively

\[
factorial(0) = 1; \\
factorial(n) = n \times factorial(n-1);
\]

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

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

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)))
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))
Computing Factorial Recursively

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

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

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

factorial(0) = 1;
factorial(n) = n*factorial(n-1);
Differences

• Pros
  – Readability

• Cons
  – Efficiency
  – Memory
Recursive Factorial

Executes factorial(4)

factorial(4)
Recursive Factorial

factorial(4)

Step 0: executes factorial(4)

return 4 * factorial(3)

Executes factorial(3)

Step 4: executes factorial(0)

Step 5: return 1

Step 6: return 1

Step 7: return 2

Step 8: return 6

Space Required for factorial(4)

Stack

Space Required for factorial(3)

Space Required for factorial(2)

Space Required for factorial(1)

Main method
Recursive Factorial

```
factorial(4)
return 4 * factorial(3)
return 3 * factorial(2)
return 2 * factorial(1)
return 1 * factorial(0)
```

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
Recursive Factorial

factorial(4)

return 4 * factorial(3)

Step 0: executes factorial(4)

return 3 * factorial(2)

Step 1: executes factorial(3)

return 2 * factorial(1)

Step 2: executes factorial(2)

return 1 * factorial(0)

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

Space Required for factorial(4)

Space Required for factorial(3)

Space Required for factorial(2)

Space Required for factorial(1)

Main method

Stack
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

Step 7: return 2

Step 8: return 6

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

Stack

Space Required for factorial(0)

Space Required for factorial(1)

Space Required for factorial(2)

Space Required for factorial(3)

Main method

OSU Oregon State University
Recursive Factorial

```
factorial(4)  
return 4 * factorial(3)  
return 3 * factorial(2)  
return 2 * factorial(1)  
return 1 * factorial(0)  
```

Step 0: executes factorial(4)

Step 1: executes factorial(3)

Step 2: executes factorial(2)

Step 3: executes factorial(1)

Step 5: return 1

Stack

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

Main method

3

Space Required for factorial(3)

Space Required for factorial(2)

Space Required for factorial(1)

Stack
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)
return 1
Step 5: return 1
Step 6: return 1

returns factorial(1)
Recursive Factorial

factorial(4) 
return 4 * factorial(3) 
return 3 * factorial(2) 
return 2 * factorial(1) 
return 1 * factorial(0) 
Step 5: return 1 
Step 6: return 1 
Step 7: return 2 
Step 8: return 6 
Step 4: executes factorial(0) 

returns factorial(2)

Space Required 
for factorial(3) 
Space Required 
for factorial(2) 
Space Required 
for factorial(1) 
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)

Step 4: executes factorial(0)

return 1

Step 5: return 1

Step 6: return 1

Step 7: return 2

Step 8: return 6

Space Required for factorial(4)

Space Required for factorial(3)

Space Required for factorial(2)

Space Required for factorial(1)

Stack

Main method
Recursive Factorial

Main method

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

Step 9: return 24

Space Required for factorial(3)
Space Required for factorial(2)
Space Required for factorial(4)
Space Required for factorial(3)
Space Required for factorial(2)
Space Required for factorial(1)

Stack