CS 161
Intro to CS I

Recursive Functions Chap. 13
Odds and Ends...

• Assignment #4
  – Extension: Due 11/5, Midnight
  – Error Handling

• Exam #1 extra points
  – Due: 11/2, 11am
```cpp
#include <iostream>
#include <cstring>

using std::cout;
using std::cin;
using std::string;

int main() {
    int i=1;
    string c;
    string str_num;

    cout << "Give me a binary number: ";
    cin >> str_num;
    c=str_num.substr(0,1); //start at 0 and get 1 char

    //A string is NULL terminated, see if we are the end
    while(!(c=="\0")) {
        if(!(c=="0"|| c=="1") ) {
            cout << "Not a valid binary number";
            cout << "Give me a binary number: ";
            cin >> str_num;
            i=0;
        }
        c=str_num.substr(i,1); //Now get the next char
        i++;
    }
    cout << str_num;
    return 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\n";
    }
}
```
Example: Drawing Rectangles

- Recursive Solution

```c
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

$0! = 1$

$n! = n \times (n-1) \times \ldots \times 2 \times 1 = n \times (n-1)! ; n > 0$
Iterative Factorial

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

factorial(4)
Computing Factorial Iteratively

factorial(4) = 4 * 3

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

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

\[
\text{factorial}(4) = 4 \times 3 \\
= 12 \times 2
\]
Computing Factorial Iteratively

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

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

\[
\text{factorial}(4) = 4 \times 3 \\
= 12 \times 2 \\
= 24 \times 1 \\
= 24
\]

\[
\text{factorial}(0) = 1; \\
\text{factorial}(n) = n \times (n-1) \times \ldots \times 2 \times 1;
\]
Computing Factorial Recursively

\begin{align*}
\text{factorial}(0) &= 1; \\
\text{factorial}(n) &= n \times \text{factorial}(n-1);
\end{align*}

\text{factorial}(4)
Computing Factorial Recursively

factorial(4) = 4 * factorial(3)

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

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

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

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

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

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

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

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

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

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

• Pros
  – Readability

• Cons
  – Efficiency
  – Memory
Recursive Factorial

Executes factorial(4)

factorial(4)

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)

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

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
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

Main method

Space Required
for factorial(3)
Space Required
for factorial(2)
Space Required
for factorial(4)
Space Required
for factorial(0)

Stack

Executes factorial(1)
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)

Step 5: return 1

Step 6: return 1

Step 7: return 2

Step 8: return 6

Main method

Space Required for factorial(4)

Space Required for factorial(3)

Space Required for factorial(2)

Space Required for factorial(1)

Stack

Main method

Space Required for factorial(4)

Space Required for factorial(3)

Space Required for factorial(2)

Space Required for factorial(1)
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

Main method

Step 5: return 1

Step 6: return 1

Step 7: return 2

Step 8: return 6

Step 4: executes factorial(0)

returns 1

Space Required for factorial(4)

Space Required for factorial(3)

Space Required for factorial(2)

Space Required for factorial(1)

Space Required for factorial(0)

Main method
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

Step 4: executes factorial(0)

returns factorial(0)

OSU Oregon State University
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)

Space Required for factorial(4)
5

Space Required for factorial(3)
4

Space Required for factorial(2)
3

Space Required for factorial(1)
2

Main method

Recursive Factorial

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

return factorial(2)
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

return factorial(4)

return 4 * factorial(3)

return 3 * factorial(2)

return 2 * factorial(1)

return 1 * factorial(0)

returns factorial(3)
Recursive Factorial

Step 9: return 24

Step 8: return 6

Step 7: return 2

Step 6: return 1

Step 5: return 1

Step 4: executes factorial(0)

Step 3: executes factorial(1)

Step 2: executes factorial(2)

Step 1: executes factorial(3)

Step 0: executes factorial(4)

returns factorial(4)