CS 161
Intro to CS I

Finish Reference vs. Pointer

Begin Recursion
Odds and Ends...

• Design due Sunday at 5pm on Canvas.
• Critiques due by the end of next week.
```cpp
#include <iostream>

using namespace std;

//pass by reference
void f(int &x) {
    x = 20;  // implicitly has * for dereference
    cout << "val: " << x << " Addr: " << &x << endl;
}

//pass by pointer
void f(int *x) {
    cout << "val: " << x << " Addr: " << &x << endl;
    *x = 10;
}

int main() {
    int i = 3;
    cout << "val: " << i << " Addr: " << &i << endl;
    cout << "val: " << *(&i) << endl;
    f(&i);
    cout << "val: " << i << " Addr: " << &i << endl;
    f(i);  // implicitly has & for address of
    cout << "val: " << i << " Addr: " << &i << endl;
    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 << "*******" << endl;
        cout << "*       *" << endl;
        cout << "*       *" << endl;
        cout << "*******" << 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 * (n-1) * ... * (n-(n-1)) * 1 = n * (n-1)! ; n > 0
Iterative Factorial

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

\[
\text{factorial}(4)
\]

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

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

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

\[ \text{factorial}(4) \]

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

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

factorial(0) = 1
factorial(n) = n \times \text{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

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

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

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

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

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

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

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))))
  = 4 * (3 * (2 * (1 * 1)))
  = 4 * (3 * (2 * 1))
  = 4 * (3 * 2)
  = 4 * 6
Computing Factorial Recursively

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

• Pros
  – Readability

• Cons
  – Efficiency
  – Memory
Recursive Factorial

factorial(4) → Executes 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
- Step 9: return 24
Recursive Factorial

factorial(4)

return 4 * factorial(3)

Step 0: executes factorial(4)

Executes factorial(3)

return 24

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

Space Required for factorial(4)

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)

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
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

Executes factorial(1)

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

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 6: return 1
Step 7: return 2
Step 8: return 6

Executes factorial(0)
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)

return 1

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
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 5: return 1
Step 4: executes factorial(0)
return 1

returns factorial(0)

Stack

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

```java
return 1
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

```
returns factorial(1)
```

**Space Required for:**
- `factorial(4)`
- `factorial(3)`
- `factorial(2)`
- `factorial(1)`
- `factorial(0)`

**Stack:**
- Main method
- `factorial(4)`
- `factorial(3)`
- `factorial(2)`
- `factorial(1)`
- `factorial(0)`
- `return 1`
Recursive Factorial

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

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

OSU Oregon State University
Recursive Factorial

return 1
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
returns factorial(3)
Recursive Factorial

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 6: return 1

Step 7: return 2

Step 8: return 6

Step 9: return 24

returns factorial(4)