CS 161
Intro to CS I

More About Functions:

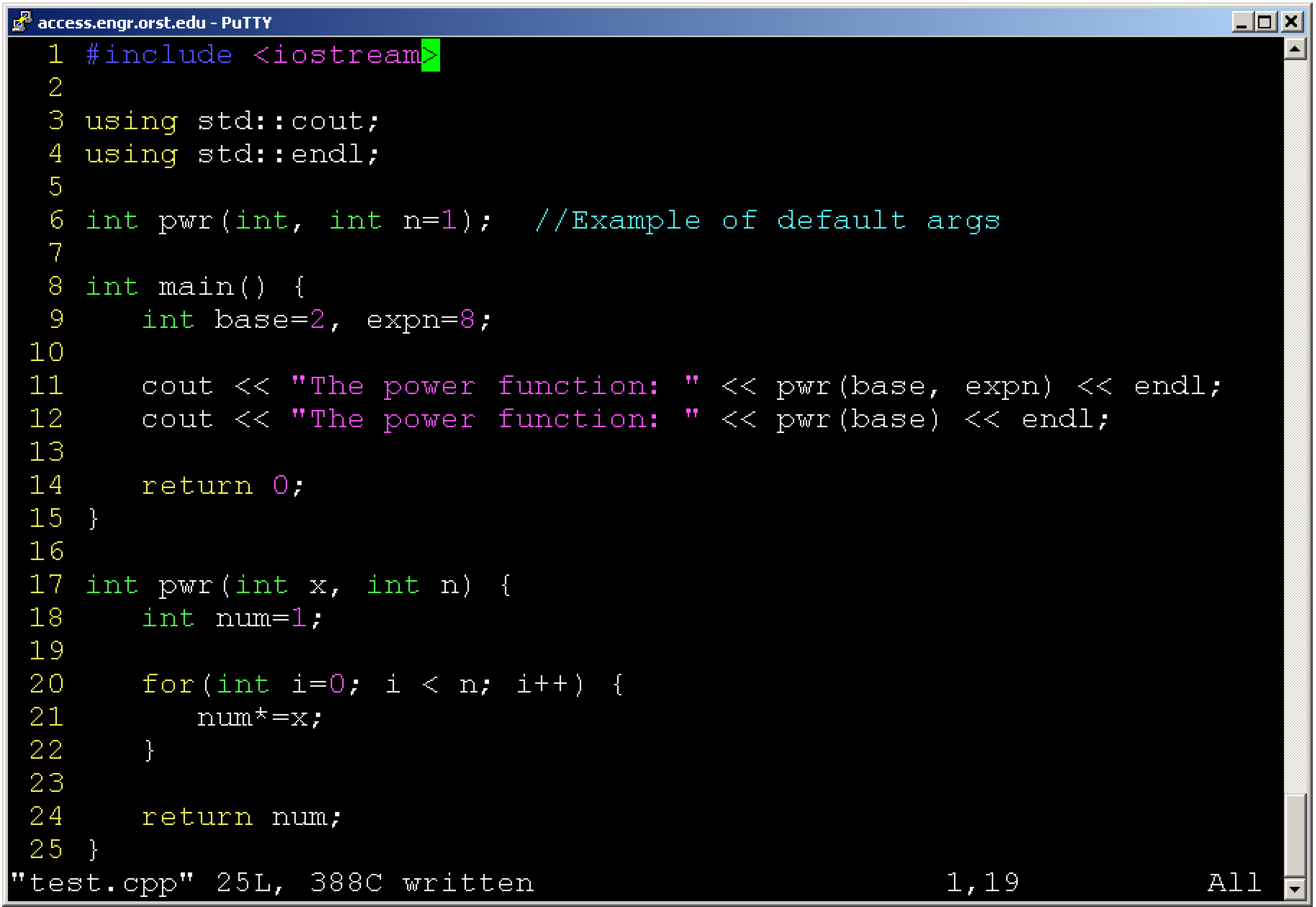
Default Values, Overloading, and References vs. Pointers

Oregon State University

Odds and Ends…

- Assignment #4 posted
- Demo Assignment #3
- Pythontutor.com (good way to visualize code)

Default Args



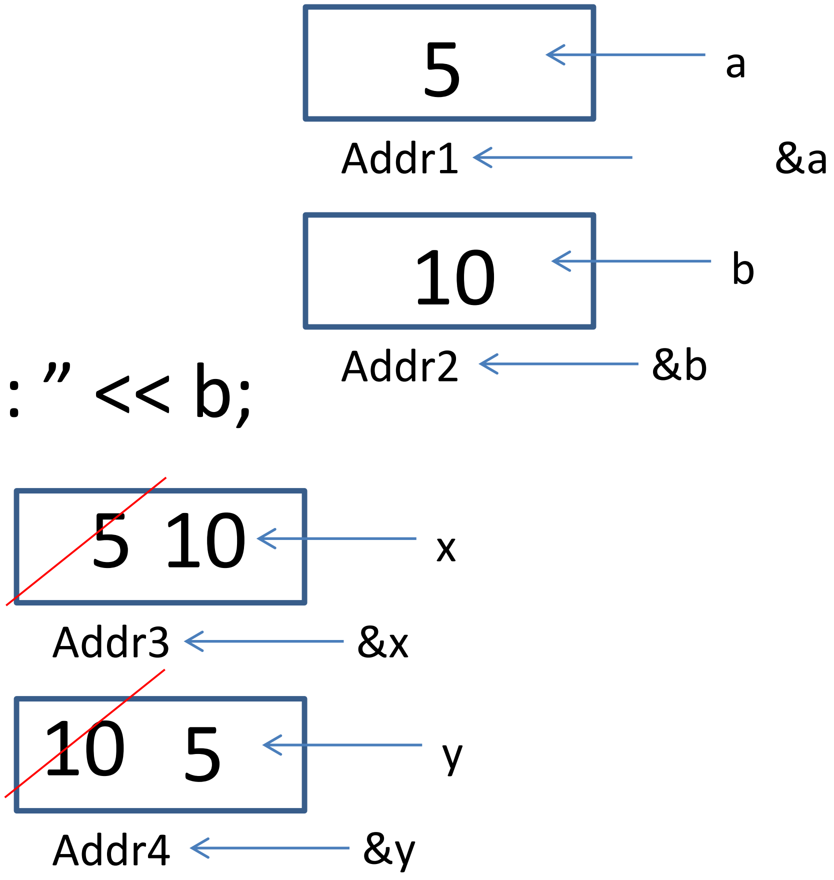
```cpp
#include <iostream>

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

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

int main() {
    int base=2, expn=8;

    cout << "The power function: " << pwr(base, expn) << endl;
    cout << "The power function: " << pwr(base) << endl;

    return 0;
}

int pwr(int x, int n) {
    int num=1;

    for(int i=0; i < n; i++) {
        num*=x;
    }

    return num;
}
"test.cpp" 25L, 388C written                                  1,19          All
```

C++ Function Overloading

- Multiple functions w/ same name
- Arguments determine function
- Default Args can be done w/ overloading
- Example: pow()
  - http://www.cplusplus.com/reference/cmath/pow/?kw=pow

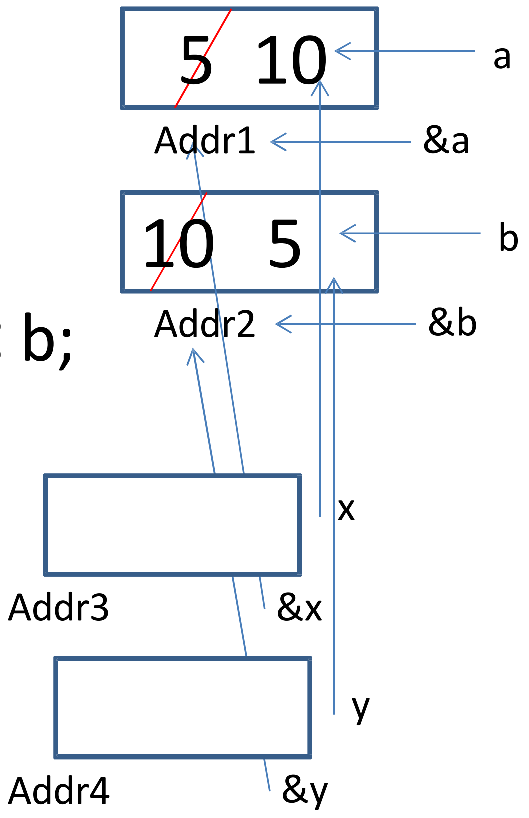
C++ Pass by Value

```
void swap(int, int);
int main() {
   int a=5, b=10;
   swap(a, b);
   cout << “a: ” << a << “b: ” << b;
}
void swap(int x, int y) {
   int temp = x;
   x = y;
   y = temp;
}
```



C++ Pass by Reference

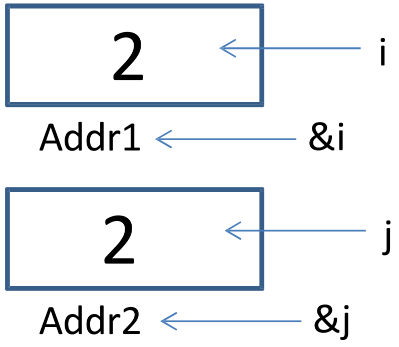
```
void swap(int &, int &);
int main() {
   int a=5, b=10;
   swap(a, b);
   cout << “a: ” << a << “b: ” << b;
}
void swap(int &x, int &y) {
   int temp = x;
   x = y;
   y = temp;
}
```



Variables vs. Pointers

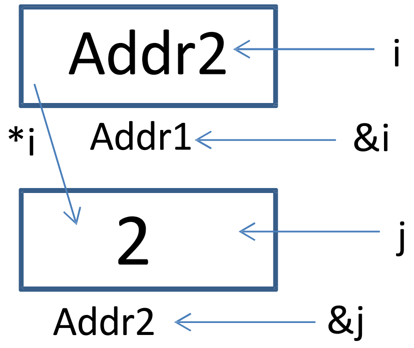
- Value Semantics
  - Values stored directly
  - Copy of value is passed

```
int i, j=2;
i=j;
```



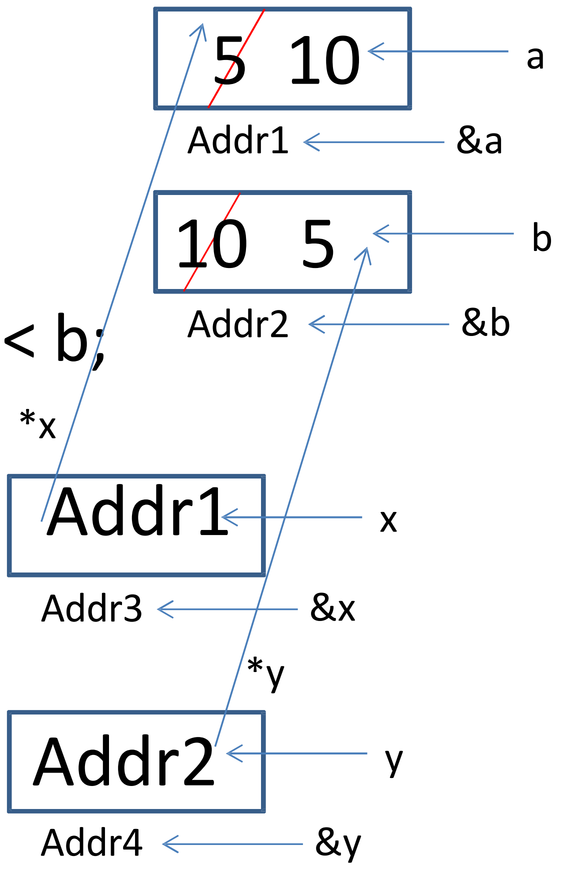
- Pointer Semantics
  - Address to variable is stored
  - Copy of address is passed

```
int *i, j=2;
i=&j;
```



C/C++ Pointers

```
void swap(int *, int *);
int main() {
   int a=5, b=10;
   swap(&a, &b);
   cout << “a: ” << a << “b: ” << b;
}
void swap(int *x, int *y) {
   int temp = *x;
   *x = *y;
   *y = temp;
}
```



OSU Oregon State University

Demo…

Pointer and References Cheat Sheet

- *
  - If used **in a declaration** (which includes function parameters), it **creates** the pointer.
    - Ex. int *p; //p will hold an address to where an int is stored
  - If used **outside a declaration**, it **dereferences** the pointer
    - Ex. *p = 3; //**goes to the address** stored in p and stores a value
    - Ex. cout << *p; //**goes to the address** stored in p and fetches the value
- &
  - If used **in a declaration** (which includes function parameters), it **creates and initializes** the reference.
    - Ex. void fun(int &p); //p will refer to an argument that is an int by implicitly using *p (dereference) for p
    - Ex. int &p=a; //p will refer to an int, a, by implicitly using *p for p
  - If used **outside a declaration**, it means **"address of"**
    - Ex. p=&a; //**fetches the address of** a (only used as rvalue!!!) and store the address in p.

More About Functions

- Do not use global variables!
- Function Headers
  - Description, Parameters, and Return Value
  - Preconditions
    - What is this?
  - Postconditions (look at Recitation Worksheet!)
    - What is this?