CS 161
Intro to CS I

Functions: Pass by Value, Pass by Reference, & Overloading
Chap. 4.1 - 4.2
Uh-Oh, What happened...

```cpp
#include <iostream>
using namespace std;

void zero_check(int d) {
    while (d==0) {
        cout << "Can't divide by zero!!!" << endl;
        cout << "Enter denominator: ";
        cin >> d;
    }
}

int main () {
    int num, denom;
    cout << "Enter numerator: ";
    cin >> num;
    cout << "Enter denominator: ";
    cin >> denom;
    zero_check(denom);
    cout << "Division: " << num/denom << endl;
    return 0;
}
```

One solution: Return a value...

```cpp
#include <iostream>
using namespace std;

int zero_check(int d) {
    while(d==0) {
        cout << "Can't divide by zero!!" << endl;
        cout << "Enter denominator: ";
        cin >> d;
    }
    return d;
}

int main () {
    int num, denom;

    cout << "Enter numerator: ";
    cin >> num;
    cout << "Enter denominator: ";
    cin >> denom;
    denom = zero_check(denom);
    cout << "Division: " << num/denom << endl;
    return 0;
}
```

"errors.cpp" 25L, 431C written
Scope (Visibility)

• Part of program in which a declaration is valid
• Local variable
  – Declared inside a function only accessible inside function
• Localizing variables
  – Declaring variable in innermost scope
Illegal access outside loops

```c++
for(x = 0; x < 10; x++) {
    int y = 10;
    cout << "The value of x * y is: " << x*y << endl;
}
cout << "The value of y is: " << y << endl; /*y outside scope*/
```

• How do we fix this?
• What about if/else blocks?
Illegal access in functions

```c
int main () {
    int x=2, y=3;
    compute_sum();
    sum = x+y; //error: sum hasn’t been declared
    return 0;
}

void compute_sum() {
    int sum = x+y; //error: x and y outside scope
}
```
#include <iostream>

int main() {

    // Don't declare x here if you want to use it outside of for
    for (int x=0; ;x++) {
        int y; // Don't declare here if you want to use outside loop!!
        std::cout << x << std::endl;
    }

    std::cout << x << ", " << y << std::endl; // scope error on x and y

    if (-1) {
        int i; // Same rule applies for if blocks
    }

    std::cout << i << std::endl; // scope error

    return 0;
}
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;
}
• What if we didn’t have temp?
C++ Pass by Reference

```cpp
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;
}
```
One solution: Pass by reference...

```cpp
#include <iostream>
using namespace std;

void zero_check(int &d) {
    while (d==0) {
        cout << "Can't divide by zero!!!" << endl;
        cout << "Enter denominator: ";
        cin >> d;
    }
}

int main () {
    int num, denom;

    cout << "Enter numerator: ";
    cin >> num;
    cout << "Enter denominator: ";
    cin >> denom;

    zero_check(denom);

    cout << "Division: " << num/denom << endl;
    return 0;
}
```
C++ Function Overloading

• Multiple functions w/ same name
• Arguments determine function
• Default Args can be done w/ overloading
• Example: pow()
Reading/Assignments

• Work on Assignment #3!!!
• Begin reviewing for test
Quiz #4

• Get into groups of 4 – 5.

• Discuss Assignment #3, design a solution, and provide your algorithm using pseudocode.
  • Where do you need a loop, and what kind of loop are you going to use?
  • How are you going to use decomposition?

• Write your own `int pwr(int base, int exp)` function that takes two integers as arguments and returns the integer result.