EECS 161
Intro to Programming I

Functions: Pass by Reference, Overloading, and Default Args
Review Chap. 3 & Chap. 4.1-4.2
Programming Errors

• Syntax errors
  – Misuse of C++ language
  – How are they caught?

• Logic errors
  – Doesn’t perform task correctly (aka. bugs)
  – How are they caught?

• Runtime errors
  – Stops your program from running
  – How are they caught?
Syntax Error Examples

• Missing main function
• Use of identifier not declared
• Misspelled Words
• Forget a Semicolon
• Forget Required Keyword
• Missing quote, curly brace, and parenthesis
• Use of single quotes instead of double
Logic Error Examples

• Poorly written programs
  – Add instead of subtract (incorrect operation)
  – Using last two digits for date
  – Same error message for different errors
  – Program that never ends
  – Add one to the largest integer (could be syntax)
Runtime Error Examples

• Open a file that doesn’t exist
• Segmentation fault
  – Infinite loop that eats memory
  – Divide by variable that is zero
Debugging Errors

• Syntax:
  – **READ compiler errors** (pay attention to line #)
  – Use **google** to search for error

• Logic/Runtime
  – Use **std::cout** to find where the code is breaking
    • Print variable values
    • Print indicator messages
  – **Trace** through the code
  – **Comment** out code
Error Handling

• What can we do to prevent these errors?
  – Overflow
  – Divide by zero
  – Getting a number with too many digits!!!
Decomposition

• Divide Problem (task) Into Subtasks
  – Procedural Decomposition
  – Examples: cooking, cleaning, etc.

• Incremental Programming
  – Iterative Enhancement (Stepwise Refinement)

• Examples: Replicating Code
Procedural Decomposition

• Functions
  – int main() { }  
  – User defined
    void draw_box() { }  

• Function Call
  – draw_box();
Procedural Decomposition

```cpp
#include <iostream>
using std::cout;

void draw_box();  // Declare function

int main() {  
    draw_box();    // Use function
    draw_box();
    return 0;
}

void draw_box() {  // Define function
    cout << "+-------+\n";
    cout << "|       |\n";
    cout << "+-------+\n";
    cout << "|       |\n";
    return 0;
}
```

```cpp
#include <iostream>
using std::cout;

void draw_box();  // Declare function

int main() {  
    draw_box();    // Use function
    draw_box();
    return 0;
}

void draw_box() {  // Define function
    cout << "+-------+\n";
    cout << "|       |\n";
    cout << "+-------+\n";
    cout << "|       |\n";
    return 0;
}
```
Functions Calling Other Functions

#include <iostream>

void draw_box();
void draw_top_bottom();
void draw_sides();

int main() {
    draw_box();
    return 0;
}

void draw_box() {
    draw_top_bottom();
    draw_sides();
    draw_top_bottom();
}

void draw_top_bottom() {
    std::cout << “+--------+\n”;
}

void draw_sides() {
    std::cout << “|       |\n”;
}
Functions

• What is a function?
  – Block of code to perform action/subroutine

• When have we seen functions already?
  – Predefined

• What is the purpose?
  – Reduce
  – Reuse
  – Readability
Generalization

• Does a function make a task more specific or more general?
  – Justification
  – Examples
Predefined Functions

- sqrt()
- pow()
- abs()
- rand()
- srand()
- What is the difference b/w srand() and others?
void Functions

• Doesn’t return a value
• Still has arguments/parameters

• Can we write a `void check_num_digits()`?
• Is it more useful to return a value?
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

```cpp
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
}
```
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;
}
```
```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(dbase, dexp) << endl;
    return 0;
}

double pwr(double x, double n) {
    double num=1.0;
    for(int i=0; i < n; i++) {
        num*=x;
    }
    return num;
}
```
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?

OSU Oregon State University
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 &, int &)
    int temp = x;
    x = y;
    y = temp;
}
Programming Demo
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][1]

[1]: http://www.cplusplus.com/reference/cmath/pow/?kw=pow
denom not visible in zero_check()
Even though it works,
DO NOT USE GLOBAL VARIABLES!!!
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;
}
```

"errors.cpp" 25L, 412C written
Conditionals, Functions, Program...

- What is the difference?
  break;
  return;
  exit(1);