CS 161
Intro to CS I

Decomposition/Begin Functions
Odds and Ends...

• Exam I – Friday, 10/20
• Keep working on Assignment #3!!!
  – Design due Sunday on Canvas!!!

• Don’t miss Demo, and be patient waiting
• READ, READ, READ!!!
• Ask TA pointed questions
• Just THINK! KISS😊
Finish cin...

• What happens when we remove the cin >> s; before the getline(cin, s);?
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

• Segmentation fault or Core dump
  – Read a file that doesn’t exist
  – Go outside of memory bounds
  – 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
```cpp
#include <iostream>
#include <string>  // c++ strings
#include <cstdlib>  // atoi()

using namespace std;

int main() {
    int x;
    float f;
    string s;  // create a string object
    bool bad;  // create a flag to indicate bad or good data

    // cin and getline difference: cin stops reading at any whitespace
    // and stays at the whitespace where it left off reading, getline
    // reads until a newline is encountered and moves past the newline
    cout << "enter a string: " << endl;
    // Having both cin and getline will create a runtime error with string
    // input not including any spaces and ends with newline because the
    // cin stops at \n, but getline reads until seeing \n, which is where
    // it left off from the cin. This means getline reads nothing and we
    // can access the first character if there isn't one there!!!
    cin >> s;  // read a string of chars from the user until whitespace
    getline(cin, s);  // read chars from the user until a newline
    cout << "length: " << s.length() << endl;
    cout << "First char: " << s.at(0) << endl;  // 0 is 1st location
    cout << "The whole string: " << s << endl;
}```
Runtime error caused by s.at(0) when there is a string with 0 chars/length

```cpp
flip3 ~/cs161/private 154% g++ loops.cpp
flip3 ~/cs161/private 155% a.out
enter a string: jen
length: 0
terminate called after throwing an instance of 'std::out_of_range'
  what(): basic_string::at
Abort (core dumped)
flip3 ~/cs161/private 156%  
```
Decomposition

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

• Incremental Programming
  – Iterative Enhancement (Stepwise Refinement)

• Examples: Replicating Code
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
Predefined Functions

• `sqrt()`
• `pow()`
• `abs()`
• `rand()`
• `srand()`

What is the difference between `srand()` and others?
Procedural Decomposition

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

• Function Call
  – draw_box();
Procedural Decomposition

```cpp
#include <iostream>
using std::cout;
int main() {
    cout << "+--------+
    cout << "|           |
    cout << "+--------+
    return 0;
}
```

```cpp
#include <iostream>
using std::cout;
void draw_box(); //Declare function
int main() {
    draw_box(); //Use function
    draw_box();
    return 0;
}
```
#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 << "|           |
";
Generalization

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