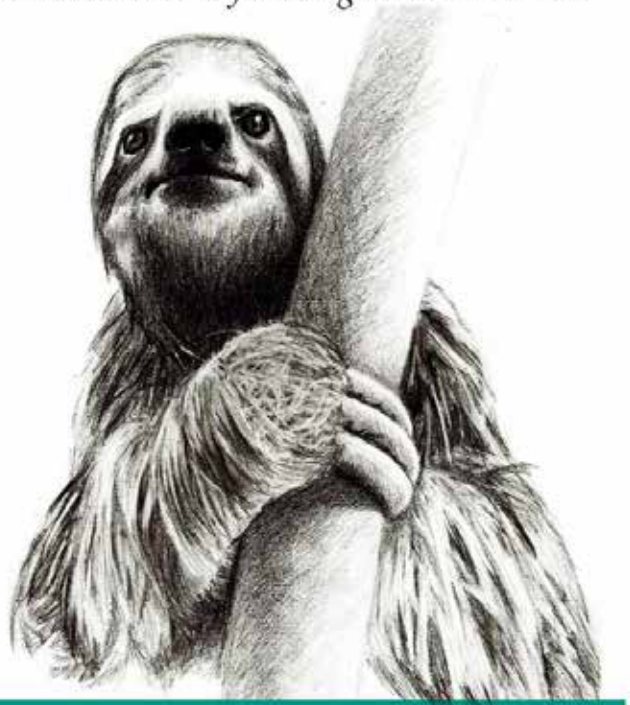
Final Exam Review

CS 16: Solving Problems with Computers I
Lecture #18

Ziad Matni
Dept. of Computer Science, UCSB

Cutting corners to meet arbitrary management deadlines



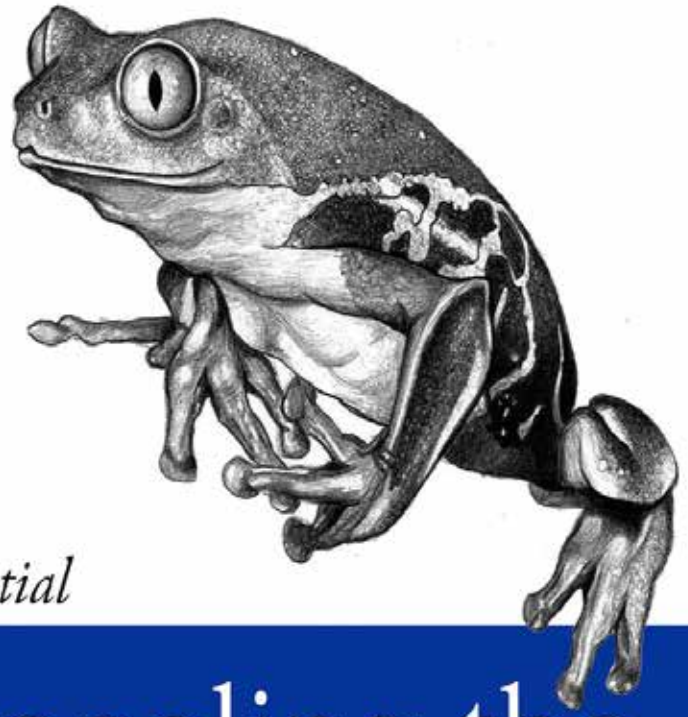
Essential

Copying and Pasting from Stack Overflow

O'REILLY®

The Practical Developer
@ThePracticalDev

The internet will make those bad words go away



Essential

Googling the Error Message

O RLY?

The Practical Developer
@ThePracticalDev

Announcements

- Lab #9 is due on the last day of classes: **Friday, 12/2**
- Teacher evaluations at the end of class

- And also… you have a final exam on Tuesday, 12/6…

FINAL IS COMING!

- Material: ***Everything***!
  - Except Constructors
- Homework, Labs, Lectures, Textbook
- **Tuesday, 12/6** in this classroom
- **Starts at 4:00pm **SHARP****
- **Seating will be assigned to you!**
- Duration: **2 hours long**
- Closed book: no calculators, no phones, no computers
- Only 1 sheet (***double***-sided is ok) of written notes
  - Must be no bigger than 8.5" x 11"
  - You have to turn it in with the exam
- **You will write your answers on the exam sheet itself.**

Concepts You Will Have To Know

The Basics

Lecture 2

- What does a CPU do?
- What does an OS do?
- What are compilers? Linkers?
- What's an algorithm and how is it different from a program?
- How do we solve problems with and without computers?
  - Including knowledge of the SDLC
- Variables and their operations in C++

- SKIP:
  - The historical stuff

Concepts You Will Have To Know

Programming Basics Lectures 3-6

- **cin** and **cout**
- **if/else** statements
- Boolean operations and logic
- Rules and precedence of operations in C++
  - Including different ways to do increments
- Loops in C++
  - **while**, **do-while**, **for**
  - Controlling statements
  - Infinite loops
  - Multiway branches
- **switch/case** statements
- Global vs. local variables
- Type casting

Concepts You Will Have To Know

Functions

Lectures 7, 8, 9

- Function declaration
- Function definition
- Function calling
- Placing of all of these
- Return statements
- "Black Box" Abstraction
- Block scope of variables
- Overloading functions in C++
- **void** functions
- **main ( )** function in C++
- Call-by-value vs. Call-by-reference
- Functions calling functions
- How do we best design a program using functions?

Concepts You Will Have To Know
Number Conversions

Lecture 9

- Positional Notation
- Binary to Octal
- Binary to Hex
- Binary to Decimal
- Any-base to Decimal

Concepts You Will Have To Know

Testing

Lecture 10

- Driver test programs
- Stubs
- Debug techniques and practices
- Using **assert**

Concepts You Will Have To Know

I/O Streams (1) Lectures 10, 11

- File I/O and Stream Variables
- **ifstream** and **ofstream** libraries
  - Variable/object declarations
  - Use of file names
  - Using **.open( )** and **.close( )** member functions
  - Use of the **>>** and **<<** operators
  - How to handle errors in File I/O: **fail( )** and **exit( )**
  - How to append data to an output file
- Formatting outputs
  - Using member functions like **.setf( )** and **.precision( )**
  - Using manipulators like **setw( )** and **setprecision( )**
- Stream names as arguments in a function
- Detecting the end of an input file
  - Using **(in_stream.eof()**) vs. **(in_stream >> next)**

Concepts You Will Have To Know

I/O Streams (2)

Lecture 11

- Formatting outputs
  - Using member functions like **.setf( )** and **.precision( )**
  - Using manipulators like **setw( )** and **setprecision( )**
- Stream names as arguments in a function
- Detecting the end of an input file
  - Using **(in_stream.eof()**) vs. **(in_stream >> next)**
- Reading characters and strings
  - **get(char)**, **put (char)** and **putback(char)**
  - **getline(string)**
- Character functions
  - **toupper( ), tolower( ), isspace( ), isalpha( ), isdigit( )**

Concepts You Will Have To Know

Strings

Lecture 11

- Basics
  - The **+** , **+=** operators
  - The use of [ ] to look at one character in a string
- Built-in string manipulators
  - Search functions
    - **find**, **rfind**, **find_first_of**, **find_first_not_of**
  - Descriptor functions
    - **length**, **size**
  - Content changers
    - **substr**, **replace**, **append**, **insert**, **erase**

Concepts You Will Have To Know

C-Strings

Lecture 14

- C-Strings vs. C++ strings
- C-String declaration & initialization
- C-String assignments, comparisons
  - Using **strncpy** and **strcmp**
- Using **getline( )** to use custom stopping point
- Use of **.ignore( )** member function with **cin**

Concepts You Will Have To Know

Arrays

Lectures 12, 13

- Basics
  - What are arrays? What types can they be?
  - How do we declare them? Initialize them?
  - Indexing use and index vs. size
- Using arrays in loops
- Using arrays in functions
  - Passing an array
  - The **const** modifier
  - Returning an array
- How are arrays stored in computer memory?
- Partially-filled arrays
- Searching arrays
- Sorting arrays
- Multi-dimensional arrays

Concepts You Will Have To Know

Combining Multiple Files Lecture 14

- Why bother? (the 4 reasons)
- Compiling with g++
- Using make

Concepts You Will Have To Know

Vectors

Lecture 14

- Basics
  - How to use them, initialize them
  - Accessing elements
- Using **push_back( )**
- Size of a vector
  - Using the **.size( )** member function
- Vector efficiency, capacity
  - And other advantages over arrays

Concepts You Will Have To Know

Pointers

Lecture 15, 16

- Basics
  - What are they? Why do we care?
  - How do we declare them? Initialize them?
- Use of the **&** and ***** operators
- The **new** and **delete** operators
- The freestore or heap
- Dangling pointers
- Automatic variables
- Using **typedef**
- Dynamic Arrays
  - Creating them and managing them
  - Multidimensional dynamic arrays
- Linked Lists
  - Definition

Concepts You Will Have To Know

Recursive Functions

Lecture 16

- Recursive functions
  - How to build them from a repeating series
- How to track them
- Ending recursive calls
  - The stopping case and why it's important
- Infinite recursion
- The "stack" concept and LIFO data structures
- Stack overflow
- Recursion vs. Loop Iteration
- Recursive functions that return something vs. void ones
- The 3 rules for thinking recursively & checking to see if it works
  - Check for infinite recursion; check stopping case; check all returned values
- The binary search example

Concepts You Will Have To Know

Structures & Classes

Lecture 17

- Basics
  - What is a class? An object? A structure?
  - Examples of pre-defined classes
  - What is a member variable? A member function?
- Structures
  - Definition, initialization and use
  - The dot operator
  - Structures in functions, in structures
- Classes
  - Definition and use
  - Creating and defining member functions
  - The scope resolution operator
  - Public vs. Private use
- SKIP:
  - Constructors

SAMPLE PROBLEMS

What is the output of this code?

```
char s1[10] = “Hello”;
char s2[10] = {'H', 'e', 'l', 'l', 'o', '\0'};
char s3[10] = {'H', 'e', '\0', 'l', 'l', 'o', '\0'};
cout << s1 << “, ” << s2 << “, ” << s3 << endl;
```

Hello, Hello, He

What is the output of this code?

```
vector<int> v;
v.push_back(5);
v.push_back(20);
cout << v[0] << “, ” << v[1] << “, ” << v.size()
```

5, 20, 2

Convert the hexadecimal number 3E8 into decimal.

$$3E8 =$$
$$3 \times 16^2$$
$$+ 14 \times 16^1$$
$$+ 8 \times 16^0$$

$$= 3 \times 256$$
$$+ 14 \times 16$$
$$+ 8 \times 1$$

$$= 1000$$

Show all the outputs:

```
int *p1, *p2;
p1 = new int;
p2 = new int;
*p1 = 10;
*p2 = 20;
cout << *p1 << endl;
cout << *p2 << endl;
*p1 = *p2;
*p2 = 30;
cout << *p1 << endl;
cout << *p2 << endl;
p1 = p2;
cout << (*p1 + *p2) << endl;
```

10

20

20

30

60

From Homework #15, Question #6:

Write a recursive function program to find the *n*th element in the following arithmetic numerical sequence: **3, 11, 27, 59, 123, …**

Hint: You first have to figure out what is the recursive pattern. You also have to identify the base case. A correct example output would look like this:

Which element of the sequence would you like to know?
4
Element number 4 in the sequence is 59.

```
int main( ) {
    int number(0);
    cout << "Enter an integer number: ";
    cin >> number;
    cout << "Element #" << number << " is: "
        << formula(number) << endl;
    return 0;
}

int formula(int n) {
    if (n == 1) return 3;
    else return (2*formula(n - 1) + 5);
}
```

WHAT IS THE SERIES DOING?

$a_1 = 3$, $a_2 = 11$, $a_3 = 27$, $a_4 = 59$, etc…

Differences: 8, 16, 32, etc…
But this involves powers… maybe something easier in the form of:
$a_n = c.\ a_{n-1} + d$?

Note that I make c = 2,
Then $a_2 = (2 \times 3) + 5$ and
Then $a_3 = (2 \times 11) + 5$, etc…

The recursive formula is:
$a_n = 2\ a_{n-1} + 5$

WHAT IS THE STOPPING CASE?

$a_1 = 3$

</LECTURE>