Programming Abstractions

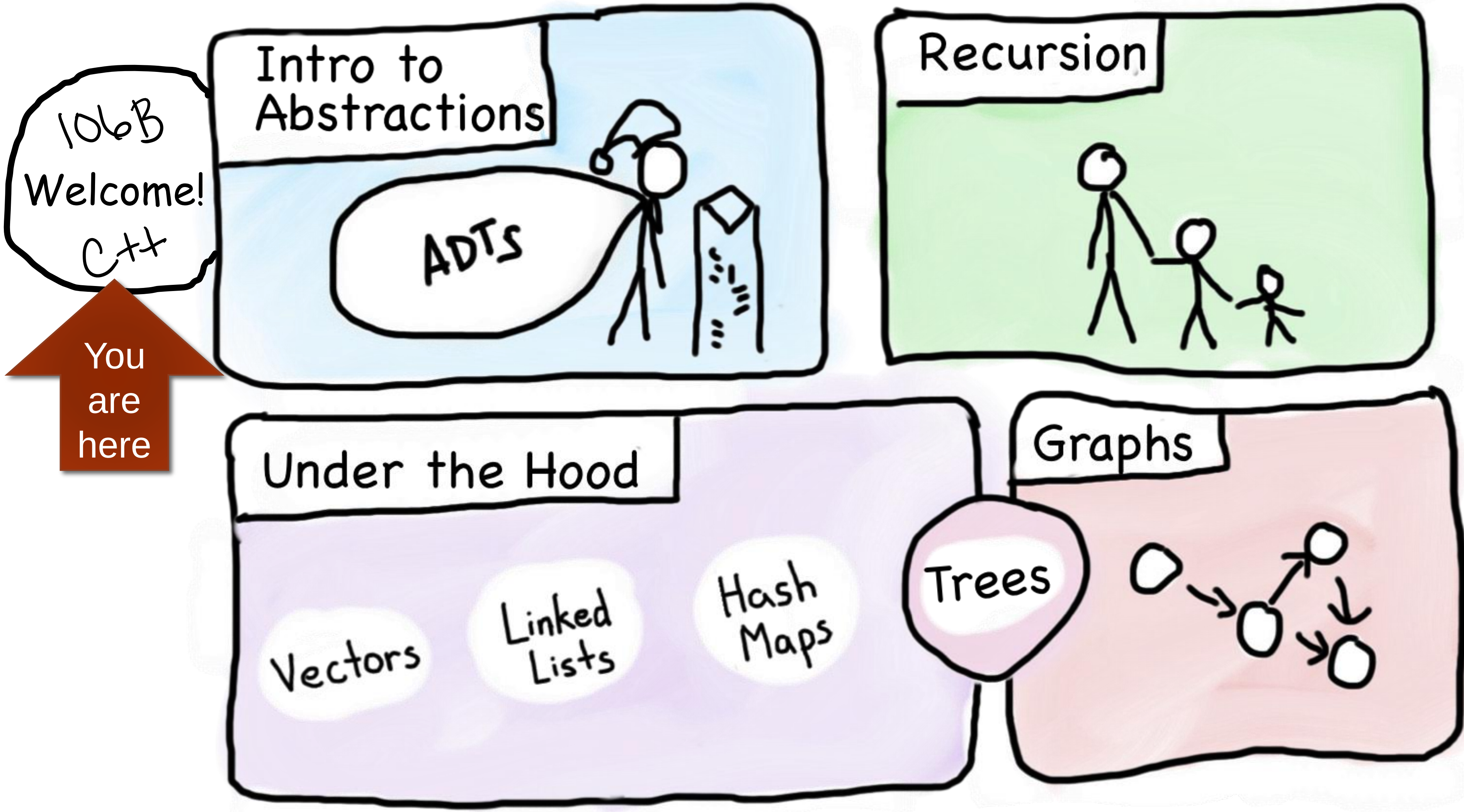
CS106B

Cynthia Lee

Today's Topics

Introducing C++ from the Java Programmer's Perspective

- firstprogram.cpp
  - Function prototypes
  - <iostream> and cout
  - "simpio.h" and getLine()
- Hamilton example
  - C++ strings and streams


106B
Welcome!
C++
You are here
Intro to Abstractions
ADTs
Recursion
Under the Hood
Vectors
Linked Lists
Hash Maps
Trees
Graphs

C++ from the Java Programmer's Perspective

(But it's ok if you don't know java!)

A first C++ program (Error)

firstprogram.cpp

```
#include <iostream>
#include "console.h"
using namespace std;

int main(){
    cout << "|-5| = "
      << absoluteValue(-5)
      << endl;
    return 0;
}
```

```
int absoluteValue(int n) {
    if (n<0){
        return -n;
    }
    return n;
}
```

A first C++ program (Fixed #1)

firstprogram.cpp

```
#include <iostream>
#include "console.h"
using namespace std;

int absoluteValue(int n) {
    if (n<0){
        return -n;
    }
    return n;
}
```

```
int main(){
    cout << "|-5| = "
      << absoluteValue(-5)
      << endl;
    return 0;
}
```

A first C++ program (Fixed #2)

firstprogram.cpp

```
#include <iostream>
#include "console.h"
using namespace std;

int absoluteValue(int n);

int main(){
    cout << "|-5| = "
      << absoluteValue(-5)
      << endl;
    return 0;
}
```

```
int absoluteValue(int n) {
    if (n<0){
        return -n;
    }
    return n;
}
```

Design Question: Why does C++ have the function prototype syntax?

In other words, why not just have a rule that you must set up the ordering so you define your functions before using them, as in the "FIXED 1" example?

A. C++ could have done that, but such a rule would be *too cumbersome* for programmers to follow.

B. C++ could have done that, but good programming *style* dictates "top-down" approach that logically puts main() first and helper functions it calls to follow.

C. C++ could *not* have done that, because sometimes there is *no way* to order the functions so that all functions are defined before being used.

D. Other/none/more than one of the above

All of the above!

Stanford University

Which code comes first, the chicken or the egg?

(this code is just for fun, for now—we'll cover recursion in depth in a few weeks!)

```
#include<iostream>
#include "console.h"
using namespace std;

void go();
void stanford();

int main(){
    go();
    return 0;
}
```

```
void go() {
    cout << "Go!" << endl;
    stanford();
}

void stanford() {
    cout << "Stanford!" << endl;
    go();
}
```

What does this code do??

Stanford University

Streams in C++

Hamilton Example
iostream (C++ Standard)
simpio (Stanford)

HamDaDaDa Code Demo:
What essential skills did we just see?

- You can read and write input/output with:
  - `cout`, `cin` (`<iostream>`)
  - `getInteger()`, `getLine()`, etc (`"simpio.h"`) print a message before waiting for input
- cin and cout use the >> and << operators, respectively
  - Remember: the arrows point in the way the data is “flowing”
  - These aren’t like HTML tags <b> or Java/C++ parentheses () or curly braces {} in that they don’t need to “match”
- Good style: “`static const int`” to make int constants
  - No “magic numbers”!
  - Works for other types too (`"static const double"`)

Strings in C++

String literal vs string class

Concatenation

String class methods

Stanford University

Using cout and strings

```
int main(){
   int n = absoluteValue(-5);
   string s = "|-5|";
   s += " = ";
   cout << s << n << endl;
   return 0;
}
int absoluteValue(int n) {
   if (n < 0){
      n = -n;
   }
   return n;
}
```

- This prints |-5| = 5
- The + operator concatenates strings, and += works in the way you'd expect.

Using cout and strings

```
int main(){
   int n = absoluteValue(-5);
   string s = "|-5|" + " = ";
   cout << s << n << endl;
   return 0;
}

int absoluteValue(int n) {
   if (n<0){
      n = -n;
   }
   return n;
}
```

But SURPRISE!…this one doesn’t work.

C++ string objects and string literals

- In this class, we will interact with two types of strings:
  - String <u>literals</u> are just hard-coded string values:
    - `"hello!"  "1234"  "#nailedit"`
    - They have <u>no methods</u> that do things for us
    - Think of them like integer literals: you can't do `"4.add(5);" //no`
  - String <u>objects</u> are objects with lots of helpful methods and operators:
    - `string s;`
    - `string piece = s.substr(0,3); //yes`
    - `s.append(t);                     //or, equivalently: s+= t;`

String object member functions (3.2)

Member function name	Description
s`.append(`***str***`)`	add text to the end of a string
s`.compare(`***str***`)`	return -1, 0, or 1 depending on relative ordering
s`.erase(`***index, length***`)`	delete text from a string starting at given index
s`.find(`***str***`)` ***s***`.rfind(`***str***`)`	first or last index where the start of ***str*** appears in this string (returns `string::npos` if not found)
s`.insert(`***index, str***`)`	add text into a string at a given index
s`.length()` or ***s***`.size()`	number of characters in this string
s`.replace(`***index, len, str***`)`	replaces *len* chars at given index with new text
s`.substr(`***start, length***`)` or ***s***`.substr(`***start***`)`	the next *length* characters beginning at *start* (inclusive); if *length* omitted, grabs till end of string

```
string name = "Donald Knuth";
if (name.find("Knu") != string::npos) {
    name.erase(7, 5);               // "Donald"
}
```

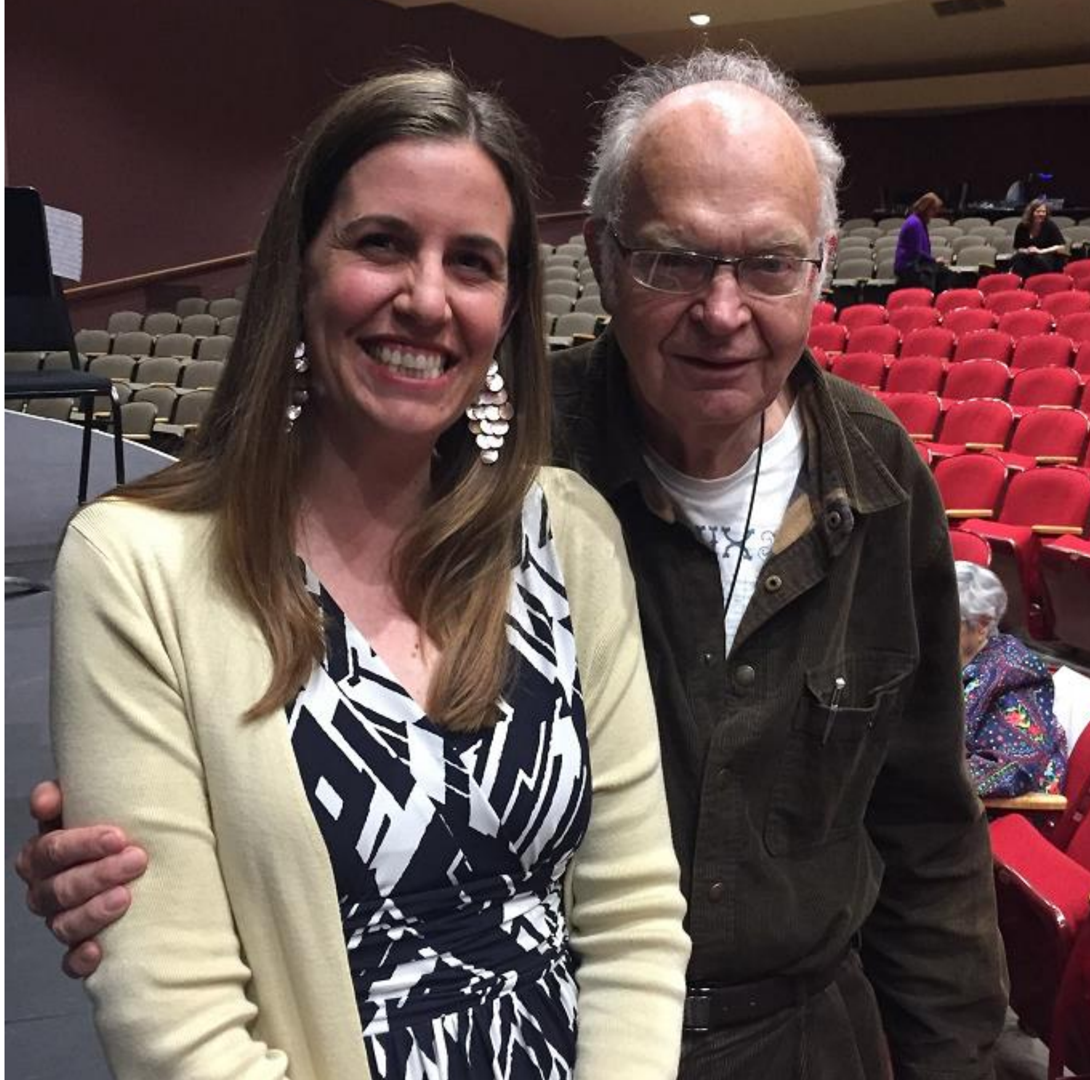
Stanford University

Aside: Donald Knuth

Emeritus (*i.e.*, retired) faculty in our dept.

Legend of computer science

If you're lucky, you'll still see him around campus from time to time!



Recap: C++ string objects and string literals

- Even though they are different types, you can mix them as long as there is a string object around to be the "brains" of the operation:
  - Yes:

```
string s;
s = "hello!"    //string knows how to convert literal
s = s + "1234"; //string has + defined as concatenation
char ch = ‘A’;  //a single ASCII character with ‘ not "
s += ch;        //string knows how to interact with char
s += ‘A’;       //and char literal
```

  - No:

```
"hello!" + " " + "bye!"; //literal not 'smart' enough to
                         //do concat with +
"hello!".substr(0);      //literal has no methods
```

Stanford University

Practice: C++ strings

```
int main(){
    string s = "1776. ";
    s += "Aaron ";
    s += s + "Burr, " + "sir. ";
    s.append("Sure, ");
    s.append("sir.");
    cout << s + 'A' << endl;
    cout << "1776" + 'B' << endl;
    return 0;
}
```

How many of these lines would NOT work?

A. 0
B. 1
C. 2
D. 3
E. More than 3

When discussing:

- Make sure you agree not only on how many but which
- Talk about the "why" for each

Stanford library (3.7)

```
#include "strlib.h"
```

- Unlike the previous ones, these take the string as a parameter.

Function name	Description

`endsWith(`***`str, suffix`***`)`
`startsWith(`***`str, prefix`***`)`
	returns true if the given string begins or ends with the given prefix/suffix text

`integerToString(`***`int`***`)`
`realToString(`***`double`***`)`
`stringToInteger(`***`str`***`)`
`stringToReal(`***`str`***`)`
	returns a conversion between numbers and strings

`equalsIgnoreCase(`***`s1, s2`***`)`	true if s1 and s2 have same chars, ignoring casing

`toLowerCase(`***`str`***`)`
`toUpperCase(`***`str`***`)`
	returns an upper/lowercase version of a string

`trim(`***`str`***`)`	returns string with surrounding whitespace removed

```
if (startsWith(name, "Mr.")) {
    name += integerToString(age) + " years old";
}
```