Programming Abstractions

CS106B

Cynthia Bailey Lee
Julie Zelenski

Today's Topics

Introducing C++

Go to pollev.com/cs106b to join class practice questions

Go to edstem.org/ to join live lecture Q&A with Julie

- Hamilton example
  - In QT Creator (the IDE for our class)
  - Function prototypes
  - <iostream> and cout
  - C++ characters and strings
  - Testing

- TODO this week:
  - Sign ups for section will **open on Thursday**, Sept. 29 at 5pm PT at cs198.stanford.edu**.** They will **close on Sunday**, Oct. 2 at 5pm PT. Section meetings start week 2
  - Assignment 0 is **due Friday**, Sept. 30 at 11:59pm
  - **Qt Installation Help Session** on 3rd floor of Durand Building on **Thursday**, Sept. 29 from 7-9pm

First C++ program (from Monday)

```
/*
 * hello.cpp
 * This program prints a welcome message
 * to the user.
 */
#include <iostream>
#include "console.h"
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

C++ math functions (2.1)

```
#include <cmath>
```

Function name	Description (returns)
`abs(`***value***`)`	absolute value
`ceil(`***value***`)`	rounds up
`floor(`***value***`)`	rounds down
`log10(`***value***`)`	logarithm, base 10
`max(`***value1, value2***`)`	larger of two values
`min(`***value1, value2***`)`	smaller of two values
`pow(`***base, exp***`)`	*base* to the *exp* power
`round(`***value***`)`	nearest whole number
`sqrt(`***value***`)`	square root

`sin(`***value***`)`
`cos(`***value***`)`
`tan(`***value***`)`
	sine/cosine/tangent of an angle in radians

Live coding in Qt

HAMILTON KING GEORGE EXAMPLE

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

- You must use function prototypes for your helper functions (if you want to keep **`main`** at the top, which is good style)
- You can write input/output with:
  - **`cout (<iostream>)`**
- cout uses the **`<<`** operator
  - Remember: the arrows point in the way the data is "flowing"
  - These aren't like HTML tags or C++ parentheses () or curly braces {} in that they don't need to "match"
- Good style: **`const int`** to make int constants
  - (in demo, not previous slides)
  - No "magic numbers"!
  - Works for other types too (**`const double`**)

Live Coding concept review

FUNCTION PROTOTYPES

A simple C++ program (ERROR)

simple.cpp

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

int main() {
   myFunction(); // compiler is unhappy with this line
   return 0;
}

void myFunction() {
   cout << "myFunction!!" << endl;
}
```

A simple C++ program (Fix option 1)

simple.cpp

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

void myFunction() {
   cout << "myFunction!!" << endl;
}

int main() {
   myFunction(); // compiler is happy with this line now
   return 0;
}
```

A simple C++ program (Fix option 2)

simple.cpp

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

void myFunction(); // this is called a function prototype

int main() {
   myFunction(); // compiler is happy with this line now
   return 0;
}

void myFunction() {
   cout << "myFunction!!" << endl;
}
```

A simple C++ program (Fix option 2)

simple.cpp

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

void myFunction(); // this is called a function prototype

int main() {
   myFunction(); // compiler initially ok with this line…
   return 0;
}

// …but sad when it realizes it was tricked and you
// never gave a definition of myFunction!!
```

Live Coding concept review

STRINGS AND CHARACTERS IN C++

Using cout and strings

```
int main(){
    string s = "ab";
    s = s + "cd";
    cout << s << endl;
    return 0;
}

int main(){
    string s = "ab" + "cd";
    cout << s << endl;
    return 0;
}
```

- This prints "abcd"
- The + operator concatenates strings in the way you'd expect.

- But…SURPRISE!…this one doesn't work.

String literals vs. C++ string objects

- In this class, we will interact with two types of strings:
  - **String literals** are just hard-coded string values:
    - `"hello!" "1234" "#nailedit"`
    - Even though old C style, we still need to use it to write string literals
    - They have no methods that do things for us
    - (object-oriented programming didn't exist back in the day of C)
  - **String objects** are objects with lots of helpful methods and operators:
    - `string s;`
    - `string piece = s.substr(0,3);`
    - `s.append(t); //or, equivalently: s += t;`

C++ standard `string` object member functions (3.2)

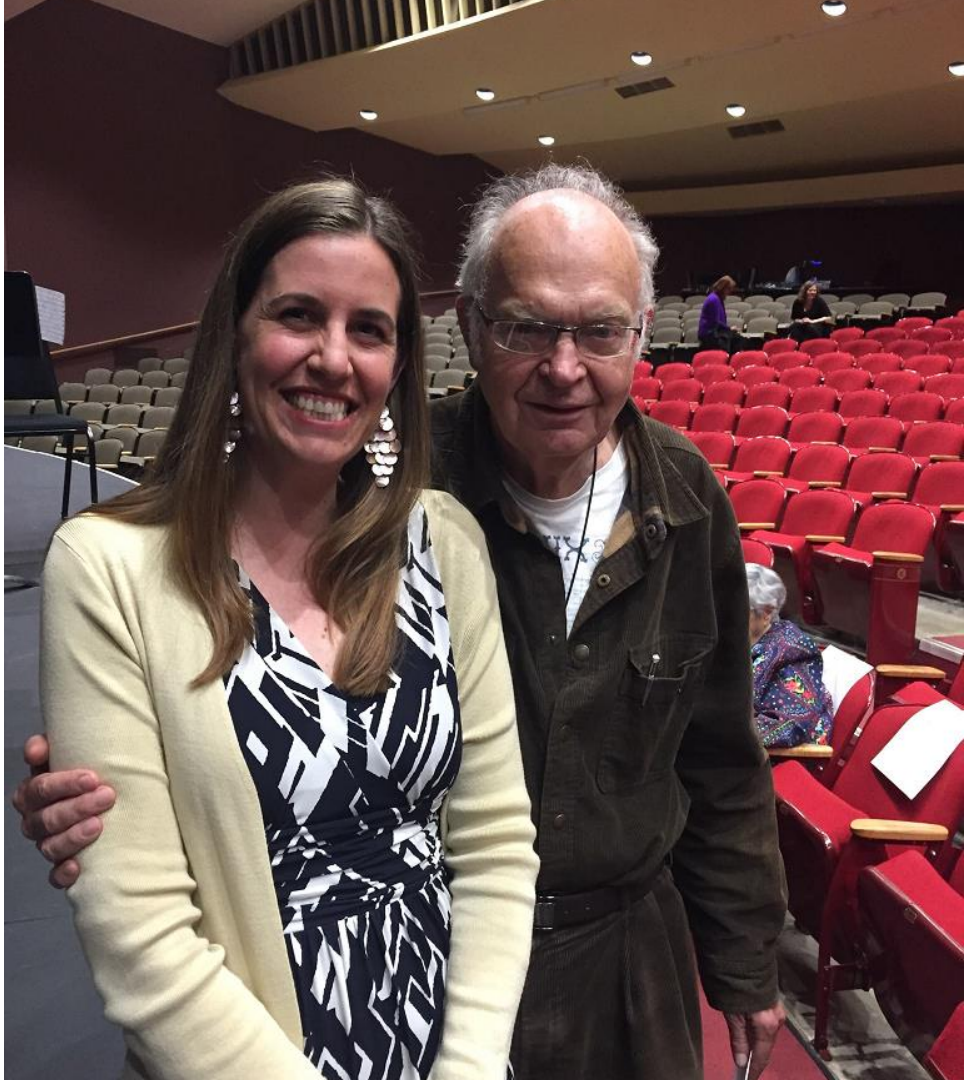
```
#include <string>
```

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(5, 6);
}
```

“Father of Algorithms”
“Yoda of Silicon Valley”
Donald Knuth

- Probably the most famous living computer scientist
- Stanford faculty (emeritus)
- Still lives on campus and comes to Gates building about once a week
- You’ll see him on his bike



BFFs!

C++ standard `string` object member functions (3.2)

```
#include <string>
```

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

Exercise: Write a line of code that pulls out the part of a string that is inside parentheses, assuming input variable `str` has the form "`(blahblah)`" where `blahblah` is any pattern of characters.

```
string insidePart = ____________________________________;
```

Exercise solutions:

Exercise: Write a line of code that pulls out the part of a string that is inside parentheses, assuming variable `str` has the form `"(blahblah)"` where `blahblah` is any pattern of characters.

```
string insidePart = ____________________________________________________;
```

Stanford library helpful string processing (*read* 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

Today's Topics

Introducing C++

- Hamilton example
  - In QT Creator (the IDE for our class)
  - Function prototypes
  - <iostream> and cout
  - C++ characters and strings
  - Testing

- TODO this week:
  - Sign ups for section will **open on Thursday**, Sept. 29 at 5pm PT at cs198.stanford.edu**.** They will **close on Sunday**, Oct. 2 at 5pm PT. Section meetings start week 2
  - Assignment 0 is **due Friday**, Sept. 30 at 11:59pm
  - **Qt Installation Help Session** on 3rd floor of Durand Building on **Thursday**, Sept. 29 from 7-9pm