Programming Abstractions

CS106B

Cynthia Bailey Lee
Julie Zelenski
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](http://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

Go to [pollev.com/cs106b](http://pollev.com/cs106b) to join class practice questions

Go to [edstem.org/](http://edstem.org/) to join live lecture Q&A with Julie
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)

```cpp
#include <cmath>
```

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description (returns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(value)</td>
<td>absolute value</td>
</tr>
<tr>
<td>ceil(value)</td>
<td>rounds up</td>
</tr>
<tr>
<td>floor(value)</td>
<td>rounds down</td>
</tr>
<tr>
<td>log10(value)</td>
<td>logarithm, base 10</td>
</tr>
<tr>
<td>max(value1, value2)</td>
<td>larger of two values</td>
</tr>
<tr>
<td>min(value1, value2)</td>
<td>smaller of two values</td>
</tr>
<tr>
<td>pow(base, exp)</td>
<td>base to the exp power</td>
</tr>
<tr>
<td>round(value)</td>
<td>nearest whole number</td>
</tr>
<tr>
<td>sqrt(value)</td>
<td>square root</td>
</tr>
<tr>
<td>sin(value)</td>
<td>sine/cosine/tangent of</td>
</tr>
<tr>
<td>cos(value)</td>
<td>an angle in radians</td>
</tr>
<tr>
<td>tan(value)</td>
<td></td>
</tr>
</tbody>
</table>
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>`) uses the `<<` operator
    - Remember: the arrows point in the way the data is “flowing”
    - These aren’t like HTML tags `<b></b>` 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)

```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)

```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)

```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)

```
#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)

```cpp
#include <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>

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

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

\begin{verbatim}
string insidePart = ____________________________;
\end{verbatim}
#include "strlib.h"

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

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endsWith(<em>str</em>, <em>suffix</em>)</td>
<td>returns true if the given string begins or ends with the given prefix/suffix text</td>
</tr>
<tr>
<td>startsWith(<em>str</em>, <em>prefix</em>)</td>
<td></td>
</tr>
<tr>
<td>integerToString(<em>int</em>)</td>
<td>returns a conversion between numbers and strings</td>
</tr>
<tr>
<td>realToString(<em>double</em>)</td>
<td></td>
</tr>
<tr>
<td>stringToInteger(<em>str</em>)</td>
<td></td>
</tr>
<tr>
<td>stringToReal(<em>str</em>)</td>
<td></td>
</tr>
<tr>
<td>equalsIgnoreCase(<em>s1</em>, <em>s2</em>)</td>
<td>true if s1 and s2 have same chars, ignoring casing</td>
</tr>
<tr>
<td>toLowerCase(<em>str</em>)</td>
<td>returns an upper/lowercase version of a string</td>
</tr>
<tr>
<td>toUpperCase(<em>str</em>)</td>
<td></td>
</tr>
<tr>
<td>trim(<em>str</em>)</td>
<td>returns string with surrounding whitespace removed</td>
</tr>
</tbody>
</table>
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