Programming Abstractions

CS106B

Cynthia Bailey
Chris Gregg

Today's Topics

Introducing C++

pollev.com/cs106b

- Hamilton example
  - In QT Creator (the IDE for our class)
  - Function prototypes
  - cout
  - C++ characters and strings
  - *Next time:* Testing our code

- **For important announcements, be sure to see the weekly announcements post on the Ed Q&A board! https://edstem.org**
- **Also on Ed: live lecture Q&A with Chris & Jonathan**

Welcome to C++

LET'S START CODING!!

C++ variables and types (1.5-1.8)

- The C++ compiler is rather picky about *types* when it comes to variables.
- Types exist in languages like Python (see the two code examples at right), but you don't need to say much about them in the code. They just happen.
- The **first time** you introduce a variable in C++, you need to announce its type to the compiler (what kind of data it will hold).
  - After that, just use the variable name (don't repeat the type).
  - You won't be able to change the type of data later! C++ variables can only hold one kind of thing.

C++

```
int x = 42 + 7 * -5;
double pi = 3.14159;
char letter = 'Q';
bool done = true;

x = x - 3;
```

Python

```
x = 42 + 7 * -5
pi = 3.14159
letter = 'Q'
done = True

x = x - 3
```

Some C++ logistical details (2.2)

```
#include <libraryname>    // standard C++ library
#include "libraryname.h"  // local project library
```

- Attaches a library for use in your program

- Note the differences (common bugs):
  - `<>` vs `" "`
  - `.h` vs no `.h`

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"`
      - They are old C (pre-C++) style, but we still need to use them
    - 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` class 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);
}
```

Exercise:

Write a line of code that pulls out all but the first and last character of a string **`str`**.
(it's ok to assume string length is at least 3)

```
string middlePart = _____________________________________________________;
```

`s.substr(start, length)` or `s.substr(start)`	the next *length* characters beginning at *start* (inclusive); if *length* omitted, grabs till end of string

pollev.com/cs106b

Stanford library helpful string processing (*read* 3.7)

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

- Unlike the previous ones, these take the string as a parameter.
  - C++ string class example: `firstName.substr(0, 10);`
  - Stanford string library example: `trim(firstName);`
- That's because we here at Stanford wrote these functions, and they are not official C++ string class methods.

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

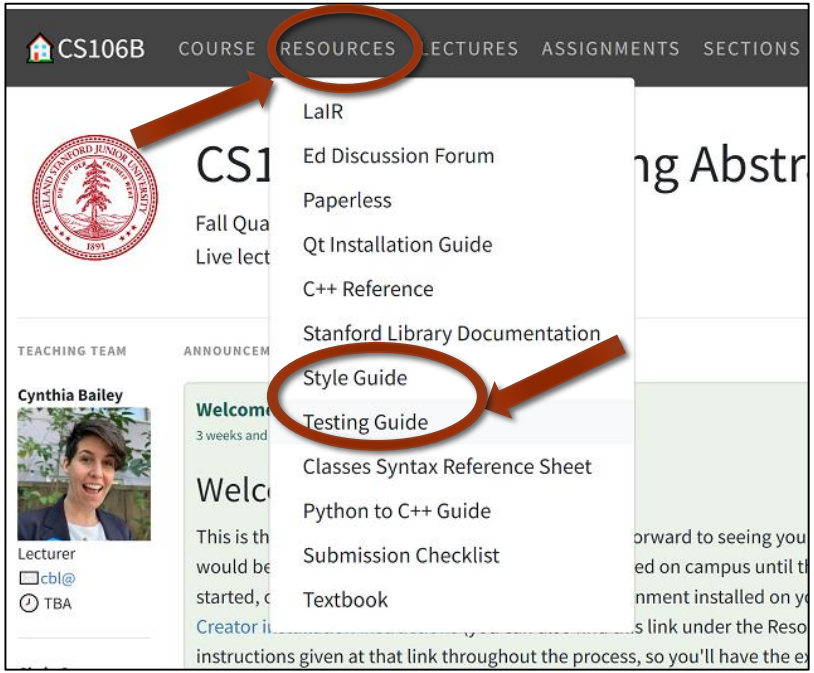
`equalsIgnoreCase(`***s1, s2***`)`	true if s1 and s2 have same chars, ignoring casing
`toLowerCase(`***str***`)` `toUpperCase(`***str***`)`	returns an upper/lowercase version of a string
`trim(`***str***`)`	returns string with surrounding whitespace removed

Live Coding concept review

STYLE AND TESTING

Code Quality in CS106B

- More details about our expectations on the website →



- **Take-home messages:**
  - Testing is an essential part of software development.
    - "If you haven't tested it, it doesn't work."
  - Just as important as writing code that works is writing it well, and making it readable by other humans.

Hamilton Code Style Notes

- Descriptive function and variable names
  - Even someone who doesn't know code would have a pretty good idea what a function called "generate lyrics" does!
- Proper indentation
  - *Even though* C++ relies on the {} and not indentation (!)
  - Pro tip: in Qt Creator, select all then do `CTRL-I` (PC) or `Cmd-I` (Mac)
- One space between operators and variables
  - Write `i < 3`, *not* `i<3`
  - Coders were social distancing before it was cool
  - Again, we do this even though C++ doesn't rely on it for parsing
- Define constants at the top of your file for any special values
  - Example: `const int DAT_FREQ = 3;`
  - Helps the reader understand what the value means or where it comes from
  - If you use the value in several places, only need to change it in one place

Next time: Testing our Code

CS106B TESTING FRAMEWORK