Programming Abstractions

CS106B

Cynthia Bailey Lee
Julie Zelenski

Today's Topics

Introducing C++

- Finish in-class string exercise
- Hamilton example (continued)
  - Style, defining constants
  - Testing
- Parameter passing in C++
  - Pass by value semantics
  - Pass by reference
  - const

NEW! Go to pollev.stanford.edu, join class "cs106b"

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

- TODO this week:
  - Sign ups for section are open at cs198.stanford.edu**.** They will **close on Sunday**, Oct 2nd at 5PM PT. Section meetings start week 2.
  - Assignment 0 is **due today, Friday**, Sept 30th at 11:59PM. There is a 48-hour grace period for assignment 0.
  - Assignment 1 will go out today and be due in 1 week.

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 = ____________________________________________________;
```

Respond at
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: `str.substr(0, 2);`
  - Stanford string library example: `endsWith(".jpg");`
- 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

Hamilton Code (continued): Style and Testing

JUST AS IMPORTANT AS WRITING THE CODE IS WRITING IT WELL AND WRITING GOOD TESTS

Hamilton Code Style Notes

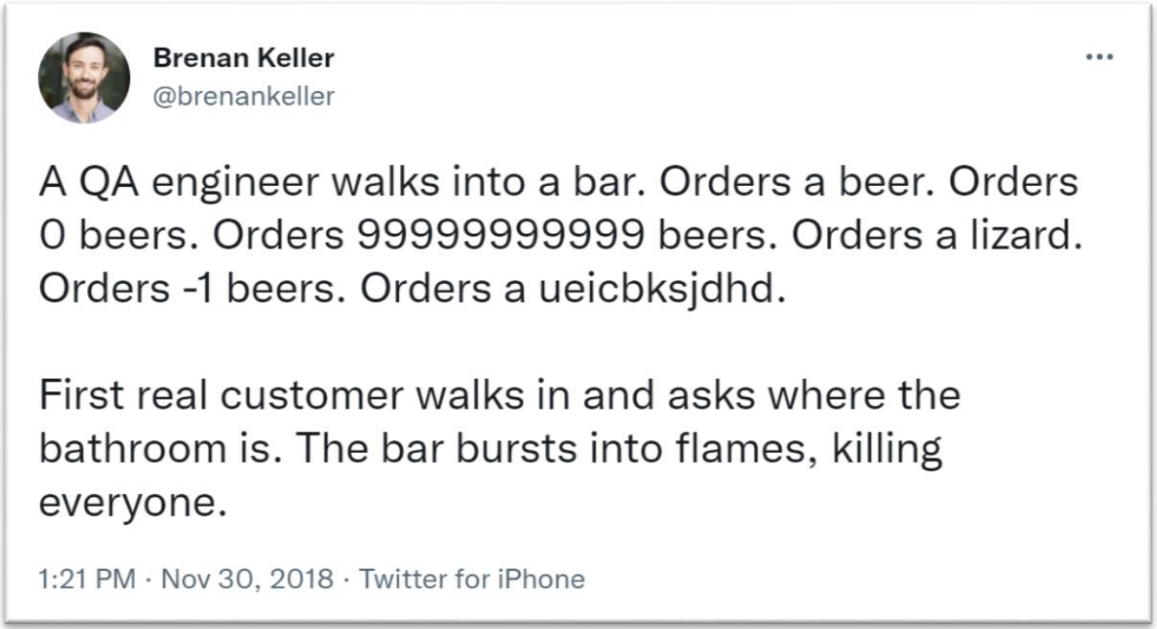
- Descriptive function and variable names
  - Even someone who doesn't know code would have a pretty good idea what a function called "generate lyrics" does!
- Proper indentation
  - *Even though* C++ relies on the {} and not indentation (!)
  - Pro tip: in Qt Creator, select all then do `CTRL-I` (PC) 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

Writing Good Tests

- "Good" means thorough: covers all code paths and cases
- But don't just add loads of tests for the sake of having many—each should have a purpose

- Be extra attentive to unusual circumstances
- These will vary, specific to the function you are testing, but common examples include:
  - Integer inputs: negative numbers, zero, very large numbers
  - String inputs: very short strings (length 0 or 1), very long strings

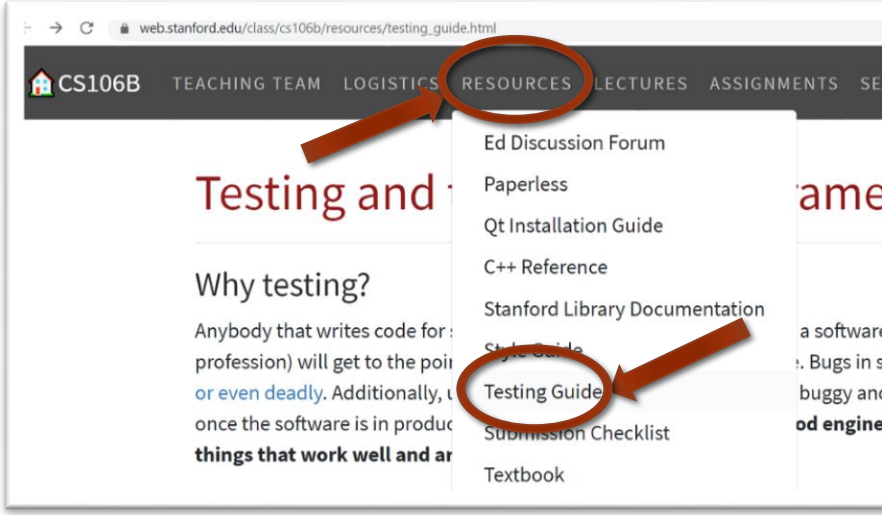
Writing Good Tests



- *A QA engineer is a software developer who specializes in writing tests and finding bugs in other engineers' code*

CS106B Testing Framework

- We provide a framework for testing your code in this class
- More details on the website →

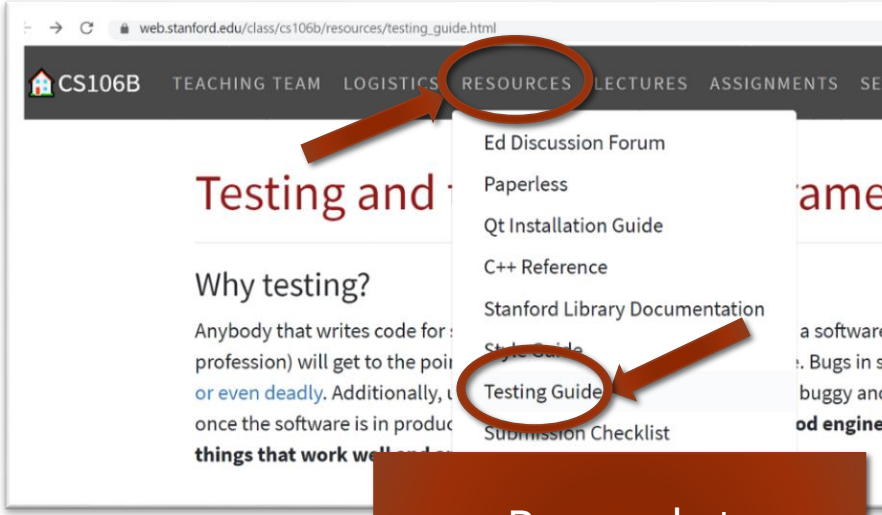


- **Quick version:**
- In `main()`, write:
  - `runSimpleTests(SELECTED_TESTS);`
- Write tests as:
  - `EXPECT_EQUAL(`***`functionBeingTested`***`(`***`input`***`), `***`expectedOutput`***`);`
  - `EXPECT_EQUAL(generateLyrics(2), "Da Da ");`

- **Your Turn: What are some good test cases for our Hamilton code?**

CS106B Testing Framework

- We provide a framework for testing your code in this class
- More details on the website →



- **Quick version:**
- In `main()`, write:
  - `runSimpleTests(SELECTED_TESTS);`
- Write tests as:
  - `EXPECT_EQUAL(functionBeingTested(input), expectedOutp`
  - `EXPECT_EQUAL(generateLyrics(2), "Da Da ");`

Respond at pollev.com/cs106b

- **Your Turn: What are some good test cases for our Hamilton code?**

C++ Parameter Passing

TWO PARADIGMS:
PASS BY VALUE
PASS BY REFERENCE

"Pass by value"
(default behavior of parameters)

```
#include <iostream>
void foo(int n);

int main(){
    int num = 5;
    foo(num);
    cout << num << endl;
    return 0;
}

void foo(int n) {
     n++;
}
```

Respond at pollev.com/cs106b

What is printed?

A. 5
B. 6
C. Error or something else

"Pass by value"
(default behavior of parameters)

```
#include <iostream>
void foo(int n);

int main(){
    int num = 5;
    foo(num);
    cout << num << endl;
    return 0;
}

void foo(int n) {
    n++;
}
```

What is printed?

A. 5
B. 6
C. Error or something else

Correct answer: 5
The function `foo` takes the value of `main`'s variable `num` as input, but the change in `foo` only happens to a local copy named `n`.

"Pass by value"
(default behavior of parameters)

```
#include <iostream>
void foo(int n);

int main(){
    int num = 5;
    foo(num);
    cout << num << endl;
    return 0;
}

void foo(int n) {
    n++;
}
```

```
#include <iostream>
void foo(int n);

int main(){
    int num = 5;
    foo(num);
    cout << num << e
    return 0;
}

void foo(int num) {
    num++;
}
```

Q: Does the answer change if our variable in `foo` is called `num` also?

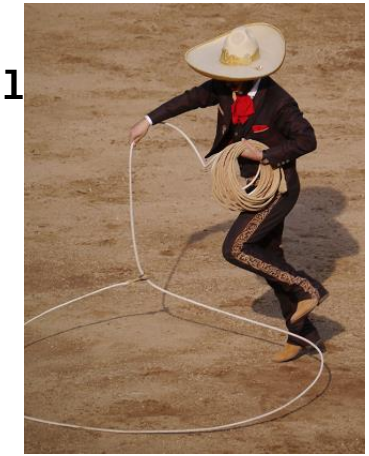
A: NO, this version also prints 5, because foo's variable is still a local copy only.

"Pass by reference"

```
#include <iostream>
void foo(int &num);

int main(){
    int num = 5;
    foo(num);
    cout << num << endl
    return 0;
}

void foo(int &n) {
    n++;
}
```



- This one prints 6!
- I like to think of the `&` as a rope lasso that grabs the input parameter and drags it into the function call directly, rather than making a copy of its value and then leaving it in place.

Your turn!

```
void mystery(int c, int& a, int b) {
    cout << b << " + " << c << " = " << a << endl;
    a++;
    b--;
}

int main() {
    int a = 4;
    int b = 7;
    int c = -2;

    mystery(b, a, c);
    mystery(c, b, 3);
    mystery(b, c, b + a);
    return 0;
}
```

What does this print?

Respond at pollev.com/cs106b

Why though??

- We've looked at the *how* of pass-by-reference, but we haven't yet discussed the *why*.
- We'll see some examples of when this feature comes especially in handy next week when we learn about containers for data!

Ethics in CS106B

ETHICAL DECISION-MAKING FRAMEWORKS

ETHICS OF STRINGS!

Ethics in CS106B

- This will be a recurring series throughout the quarter, and will tie in to your homework assignments

- **What to watch for in your Assignment 1 ethics video:**
  - **Meet your guide, Katie Creel!** Dr. Creel has degrees in computer science, moral philosophy, and history of science in society.
  - Learn about some philosophical **frameworks for making ethical decisions**, which we will be a formal guide for our thinking throughout the quarter
  - Consider the **ethical implications of C++ variable types `char` and `string`**, which you just learned about
    - *That's right, even something as simple as strings has ethical concerns!*