Programming Abstractions

CS106B

Instructor: Cynthia Bailey

Head TA: Yasmine Alonso

Today's Topics

pollev.com/cs106b

C++ intro, continued.

- Hamilton example (continued)
  - Writing good tests
- Parameter passing in C++
  - Pass by value semantics
  - Pass by reference
- A special C/C++ type: `struct`
- Important info for your first coding assignment that goes out today!
  - *Including:* Ethics discussion of C++ strings and representational harms

- **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 Yasmine**

Live Coding concept review

STYLE AND TESTING

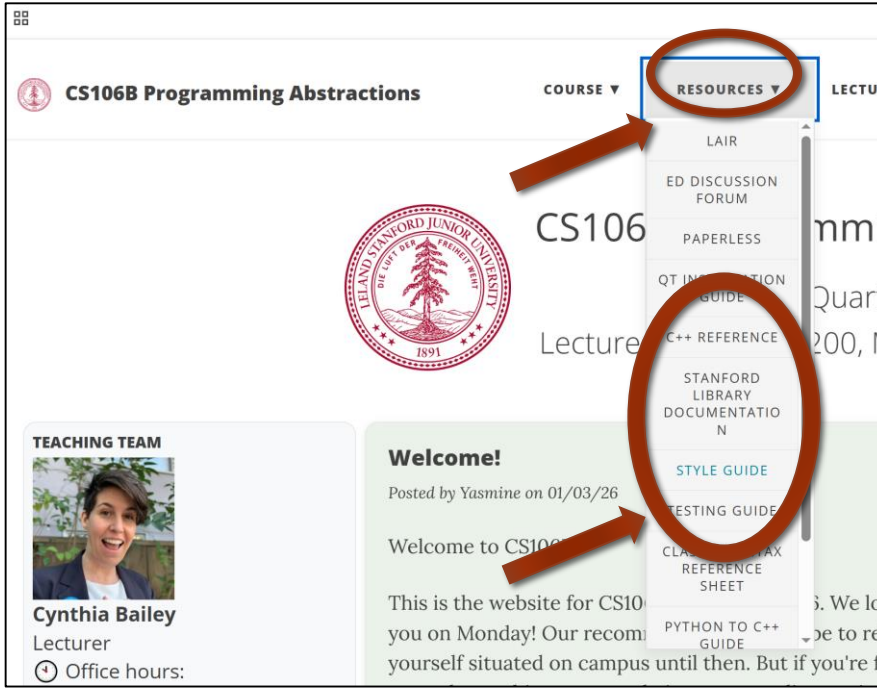
Hamilton Code Style Notes

Just as important as writing code that works is writing it well, and making it readable by other humans.

- **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`
  - 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

Important Resources for 106B

- More details about our expectations for code style on the website →
- Also:
  - List of C++ built-in libraries/functions that you can `#include`
  - List of Stanford libraries/functions that you can `#include`
  - Guide to writing tests in our testing software package (which we'll explore now!)



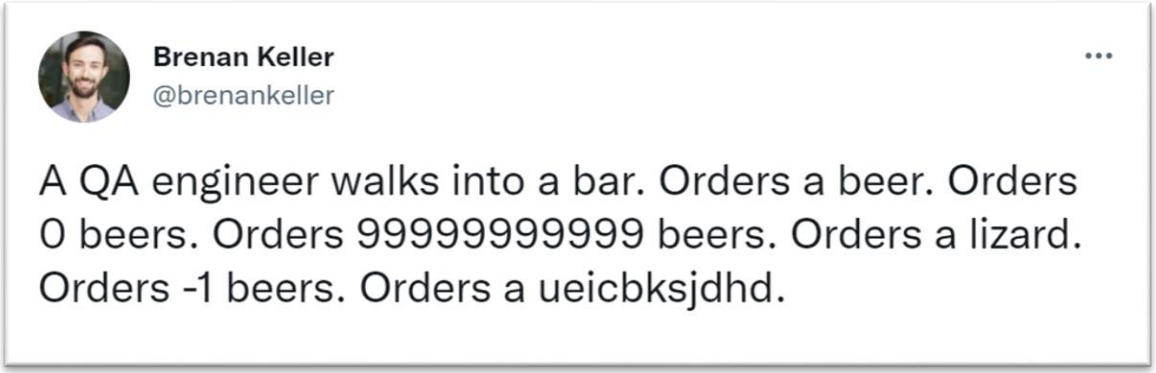
Writing Good Tests

- Testing is an essential part of software development.
  - "If you haven't tested it, it doesn't work."

- "Good" means **thorough**: covers **all code paths** with targeted 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*
- ***It's their job to think of creative ways to "break" things.***

CS106B Testing Framework

- We provide a framework for testing your code in this class

- **Quick version:** (more details on the website)
  - 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?**

pollev.com/cs106b

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++;
}
```

What is printed?

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

pollev.com/cs106b

"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?

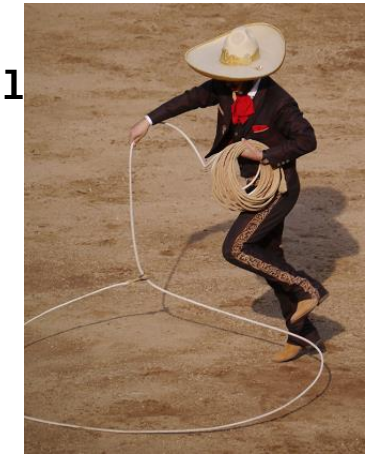
A: NO, this version also prints 5, because foo's variable is still a local copy only.

"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.

Extra practice problem (review after class if desired)

```
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;
}
```

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**!

C/C++ type: `struct`

SORT OF A VERY BASIC CLASS

A special type in C/C++: `struct`

- `struct` is like a very basic class
- It's a way to group a fixed number of pieces of data together for convenience
  - *As we've discussed before, C was invented before classes and objects—this was C's early attempt at something like a class*

- Example: `GPoint struct` in the Stanford libraries

```
GPoint loc;          // this struct type has 2 fields, x and y
loc.x = 5;           // like an object, use . to access fields
loc.y = -10;
```

Important info for your first coding assignment

ASSIGNMENT 1 GOES OUT TODAY, IS DUE A WEEK FROM MONDAY

Assignment Advice

- Start early!
- Refer to our Style Guide
- Take your time and really engage with each step of the process
  - Tip: don't be in too much of a rush to get past the warm-up steps to the "real" part—the warm-ups are very thoughtfully designed to help you
- Read the late policy on the course website
  - **Late days are to be used in case of emergencies, such as illness, injury, personal crisis; as well as mishaps like forgetting to submit even though you did finish**
  - We don't attempt to actively monitor that you don't use them for things like ski trips or because you were busy with another class' midterm, but those are **not approved uses**, and if you have a true emergency later in the quarter but no days left, that will become an issue at that point.
  - *Email Head TA Yasmine if you have a true emergency that consumes all your free allocation but you still need more*

Assignment Advice

- FAQ: Why aren't we allowed to use tools like Copilot, ChatGPT, or StackOverflow, or copy and adapt code, when professional engineers often do those exact things?
- Answer:
  - We have nothing against that per se. These are indeed good approaches (that we ourselves use!), *in the right context*.
- Analogy:
  - Many times a personal trainer will direct you to use motions that make a task harder
    - Keep body in a flat line when doing a push-up
    - Run from here to there but doing high knees

  - **Your turn:** Why do you think trainers impose these artificial conditions on people's motions, when that's not how you would do it in "real life"?

pollev.com/cs106b

Ethics in CS

ETHICS OF STRINGS

Ethics in CS106B

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

- **What we'll talk about this time:**
  - 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 this week
    - *That's right, even something as simple as strings has ethical concerns!*

Soundex project in Assignment 1

- Soundex is a phonetic algorithm used to identify and group words that sound the same.

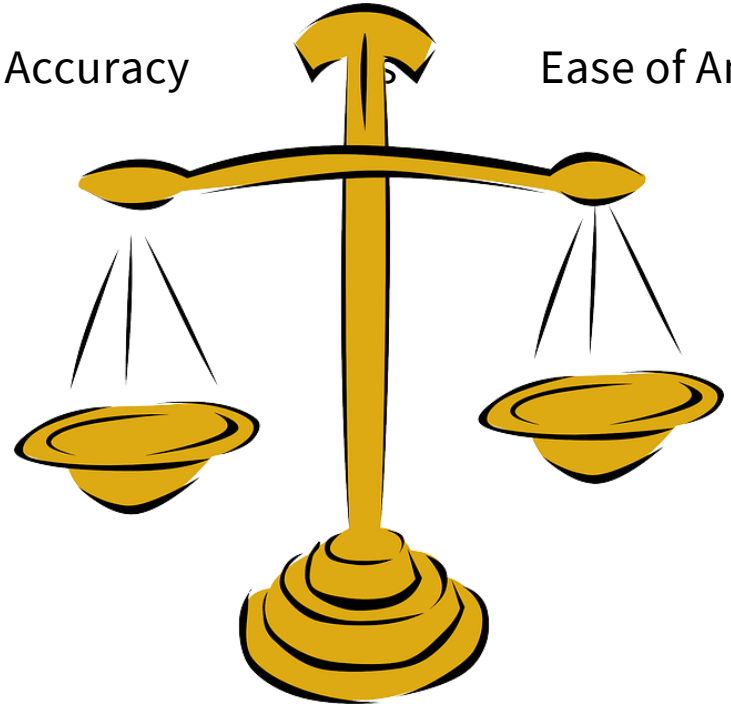
- Phonetic algorithms help us identify words with different spellings that have similar pronunciation, such as names in the U.S. Census.
  - Example: identify members of the same family in old census records, when their names may have slight spelling variations.

- Pretty cool that you're already implementing real algorithms that are really used in the real world!

- *Choosing an algorithm for social science research involves tradeoffs—choose wisely!*

Tradeoffs in algorithms like Soundex

Representational Accuracy

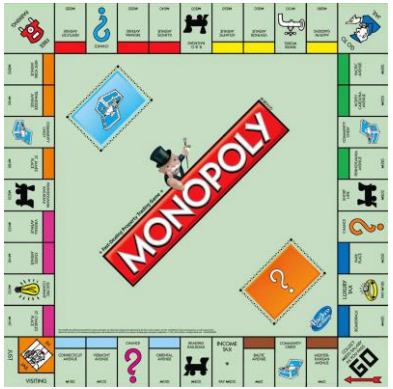


Ease of Analysis

What kind of harm is lack of representation?

Distributional Harms

- How should goods or outcomes in society be distributed?



Distributional Harms

- Equality of Opportunity:
  - *Everyone has the same opportunity to pursue the good thing*
  - *…but may result in unequal outcomes.*

Distributional Harms

- Equality of Outcome:
  - *Everyone gets the same good things, and the same responsibilities*

Distributional Harms

- Equality of Welfare:
  - *Everyone gets equal well-being/happiness, but not everyone may need the same amount of same resources to achieve that*

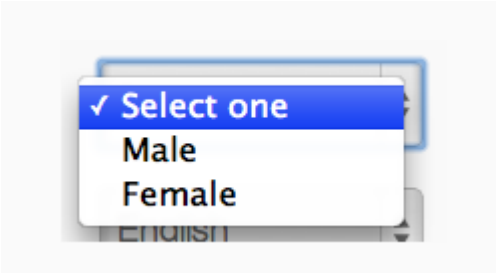
Back to representational harms

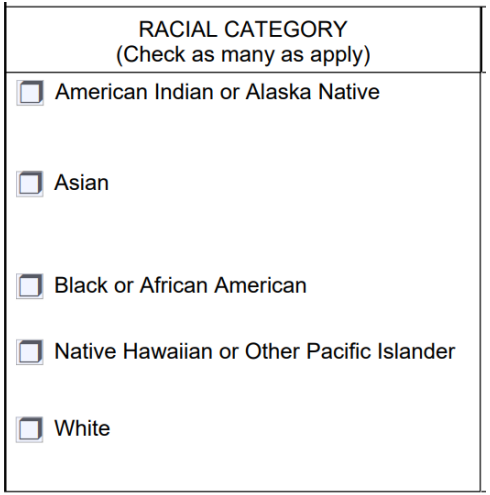
Representational Harms

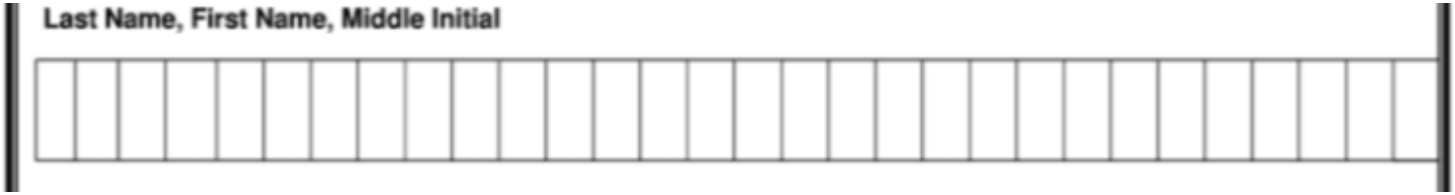
- Am I represented in this system?
- Can I express myself in it?
- Does this system recognize me, my culture, my language, or my self-expression?

Representational Harms

- Comes up ALL. THE. TIME. in website/app interface design and database design!

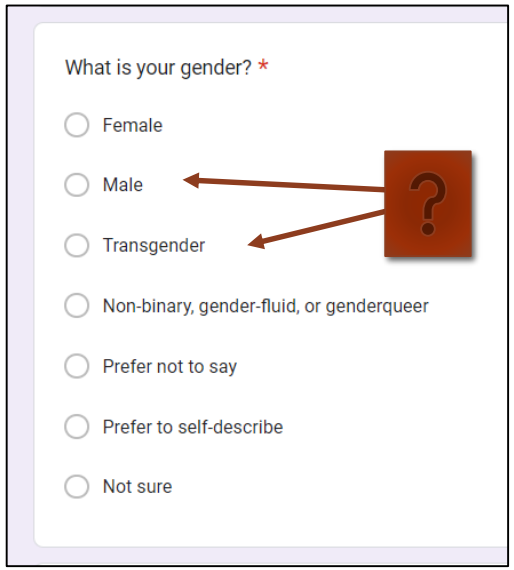






Representational Harms

- In fact, I just noticed before the quarter when preparing our assign0 that a question left over from previous years had a representational harm issue!
  - For example, a trans man may (or may not!) want to choose both “Male” and “Transgender,” but the form only allowed one selection



Representational Harms

- C originally had only ASCII code for its string type
- ASCII invented in 1963 by a guy from Michigan, Bob Bemer
- Only allows A-Z and a-z (plus digits, punctuation) as available characters
  - Can't do accents like Frédéric
  - Forget about Arabic, Korean, Chinese, etc, etc.
  - Also had to do old-timey "ASCII art" emoji like :-(instead of ☺

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040		Space	64	40	100	@	@	96	60	140	`	`
33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Representational Harms

- Unicode is a more modern option
  - As of Unicode version 16.0, there are **155,063 characters**
  - 168 modern and historical scripts
  - Also emoji and other symbols
  - Full coverage of 90 languages
  - Basic coverage of 200 languages