Lecture 9: STL Summary

CS 106L, Fall '20

Today's Agenda

- Big STL Recap
- Optional exercise to try on your own

Overview of STL

Containers

Adaptors

Iterators

Functions

Algorithms

You've now seen it all!

Adaptors

Containers

Iterators

Functions

Algorithms

THE BIG STL RECAP

The Big STL Recap

1. Streams
2. Containers
3. Iterators
4. Templates
5. Lambdas
6. Algorithms

Streams Recap

Streams Recap

console &
keyboard

a string

files

Streams Recap

console & keyboard

a string

files

output streams (cout, ofstream):

- `cout << "5 + 2 = " << 7 << endl;`

input streams (cin, ifstream):

- `cin >> myString >> myInt;`
- `getline(stream, line);`

Streams Recap

console & keyboard

a string

files

State bits:

- good, fail, eof, bad
- fail fails silently!

output streams (cout, ofstream):

- `cout << "5 + 2 = " << 7 << endl;`

input streams (cin, ifstream):

- `cin >> myString >> myInt;`
- `getline(stream, line);`

```
while (getline(stream, temp)) {
    do_something(temp);
}
```

Streams Recap

console & keyboard

a string

files

State bits:

- good, fail, eof, bad
- fail fails silently!

output streams (cout, ofstream):

- `cout << "5 + 2 = " << 7 << endl;`

input streams (cin, ifstream):

- `cin >> myString >> myInt;`
- `getline(stream, line);`

```
while (getline(stream, temp)) {
    do_something(temp);
}
```

filestreams:

- `fstream fs(filename);`
- Don't need to call `fs.open()` or `fs.close()`!

Challenge #1: Streams

```
string fileToString(ifstream& file)
```

Containers Recap Pt. 1

Sequence Containers:

- vector

  (fast access of middle)

- deque

  (fast insert begin/end)

Container Adaptors:

- stack → secretly a deque
- queue → secretly a deque

Containers Recap Pt. 1

Sequence Containers:

- vector

  (fast access of middle)

- deque

  (fast insert begin/end)

Container Adaptors:

- stack → secretly a deque
- queue → secretly a deque

vectors / deques:

- `vector<int> v{7, 8, 9};`
- `v.push_back(10);`
- `v.pop_back();`
- deques only: `push_front(6), pop_front()`

stack: `stack<int> s{1, 0, 6}; s.push(5); s.pop();`

queue: `queue<int> q{1, 0, 6}; q.push(5); q.pop();`

Containers Recap Pt. 2

Associative Containers:

(sorted, fast for range:)

- map
- set

(allows repeated keys)

(hashed, fast for single elems:)

- unordered_map
- unordered_set

Containers Recap Pt. 2

Associative Containers:

(sorted, fast for range:)

- map
- set

(allows repeated keys)

(hashed, fast for single elems:)

- unordered_map
- unordered_set

map:

```
- map<int, string> m{{5, “Hi”}, {80, “Bye”}};
- m[106] = “C++”;
- m.count(106);
```

set:

```
- set<int> s{1, 0, 6};
```

Challenge #2: Containers (if time)

```
vector<int> createCountVec(const string& text)
```

Iterators Recap

An iterator allows us to iterate over any container.

- Copyable (`iter = another_iter`)
- Retrieve current element (`*iter`)
- Advance iterator (`++iter`)
- Equality comparable (`iter != container.end()`)

Iterators Recap

An iterator allows us to iterate over any container.

- Copyable (`iter = another_iter`)
- Retrieve current element (`*iter`)
- Advance iterator (`++iter`)
- Equality comparable (`iter != container.end()`)

STL containers support:

- begin() - iterator to the first element
- end() - iterator one past the last element

Iterators Recap

An iterator is like “the claw”.

- Copyable (`iter = another_iter`)
- Retrieve current element (`*iter`)
- Advance iterator (`++iter`)
- Equality comparable (`iter != container.end()`)

STL containers support:

- begin() - iterator to the first element
- end() - iterator one past the last element

for-each loop is just iterators!

```
for (int i : s) vs.

for (auto it = s.begin(); it != s.end(); ++it)
```

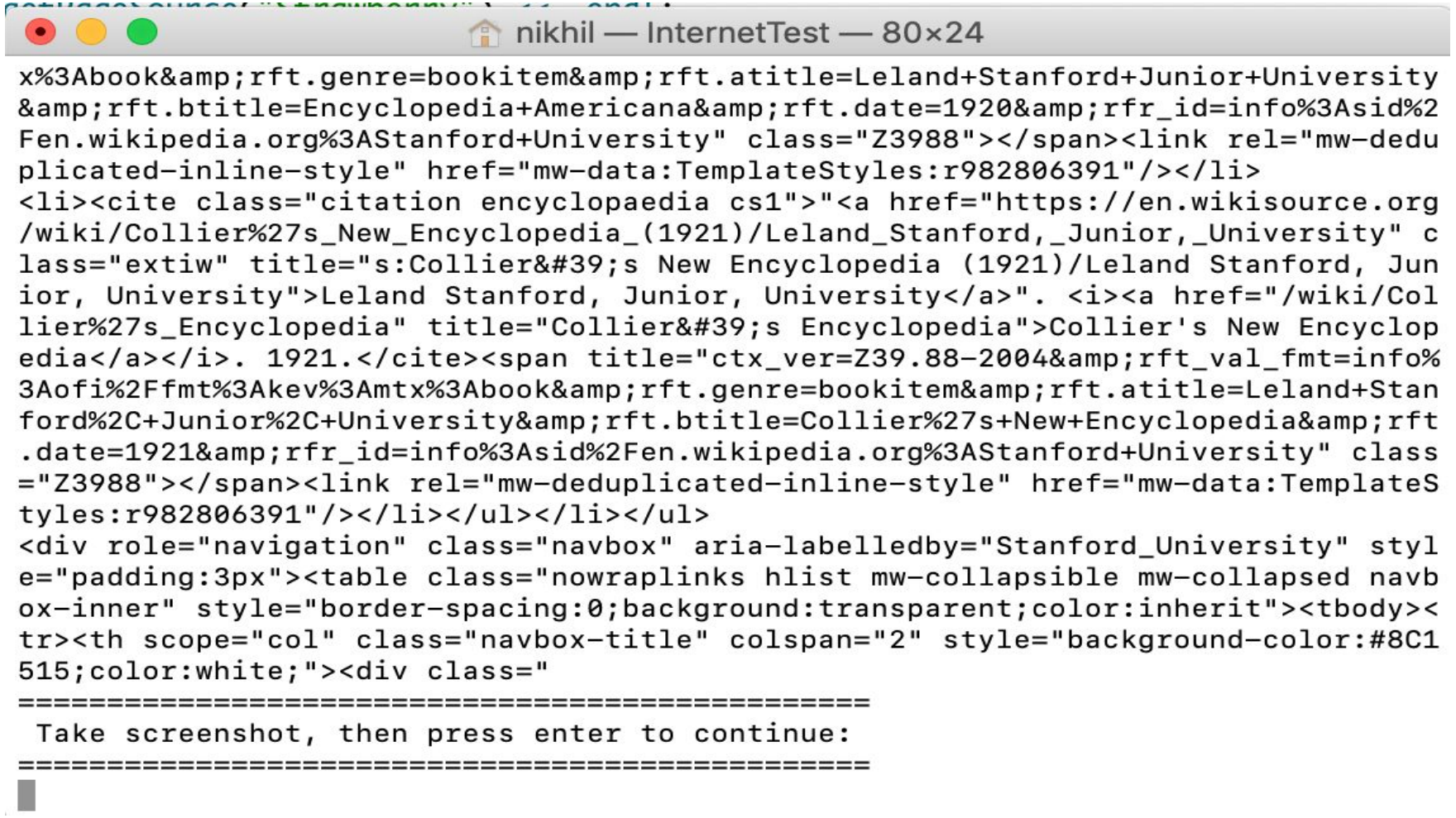
Challenge #3: Iterators

```
int countOccurrences(const string& text, const string& feature)
```

Announcements

Thank you for for completing the InternetTest exercise!

- Please confirm your screenshots look something like this:



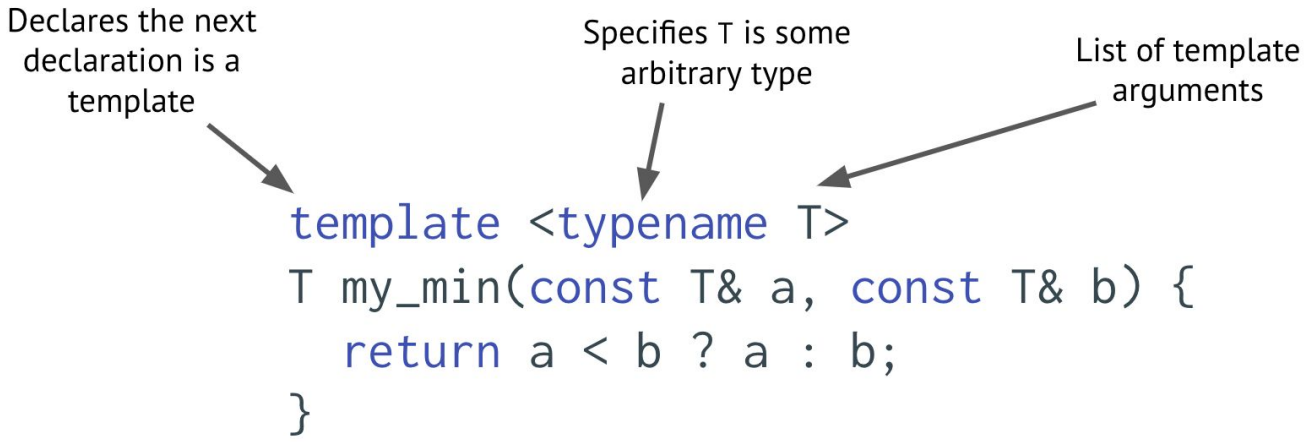
Mid-quarter survey!

- +1 late day!
- https://forms.gle/KjUSQzmU1f8LG1tV8

Return to STL

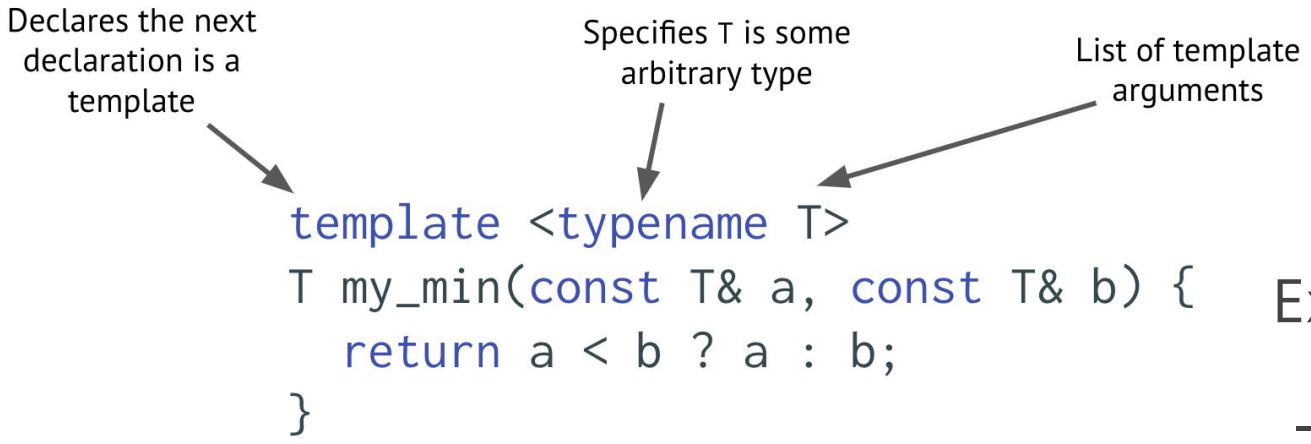
Templates Recap

Templates Recap



Note: Scope of template argument T is limited to this one function!

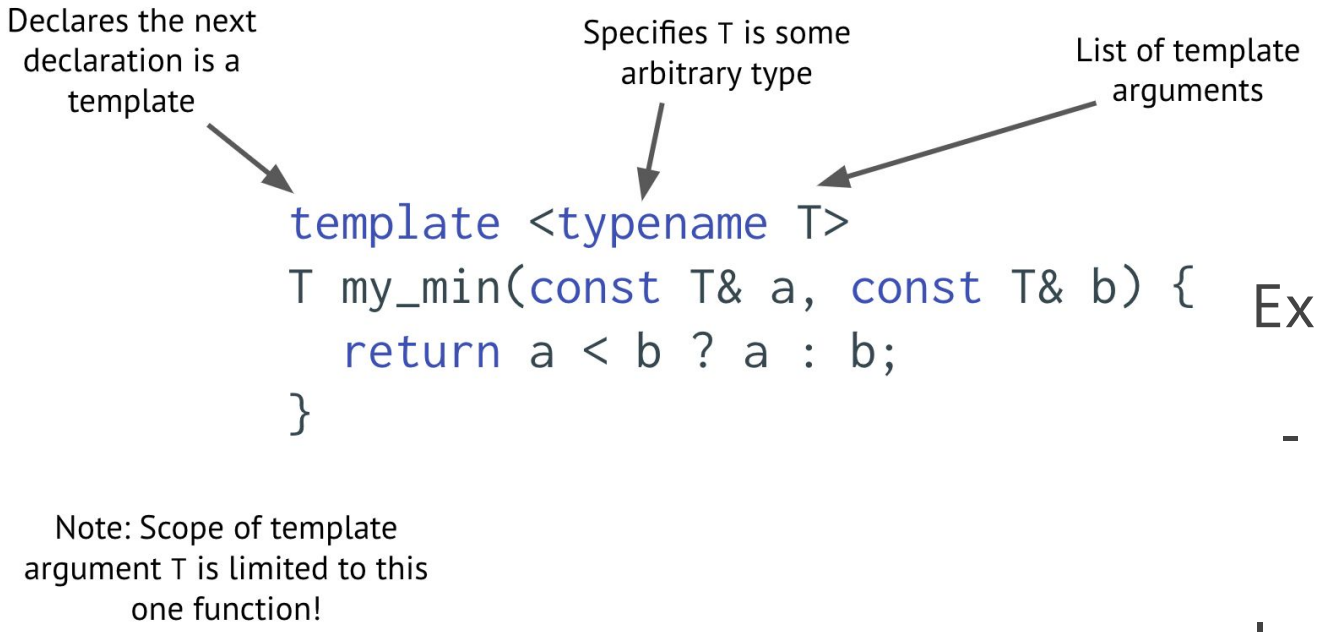
Templates Recap



Explicit instantiation:

- `my_minmax<string>("Nikhil", "Ethan");`

Templates Recap



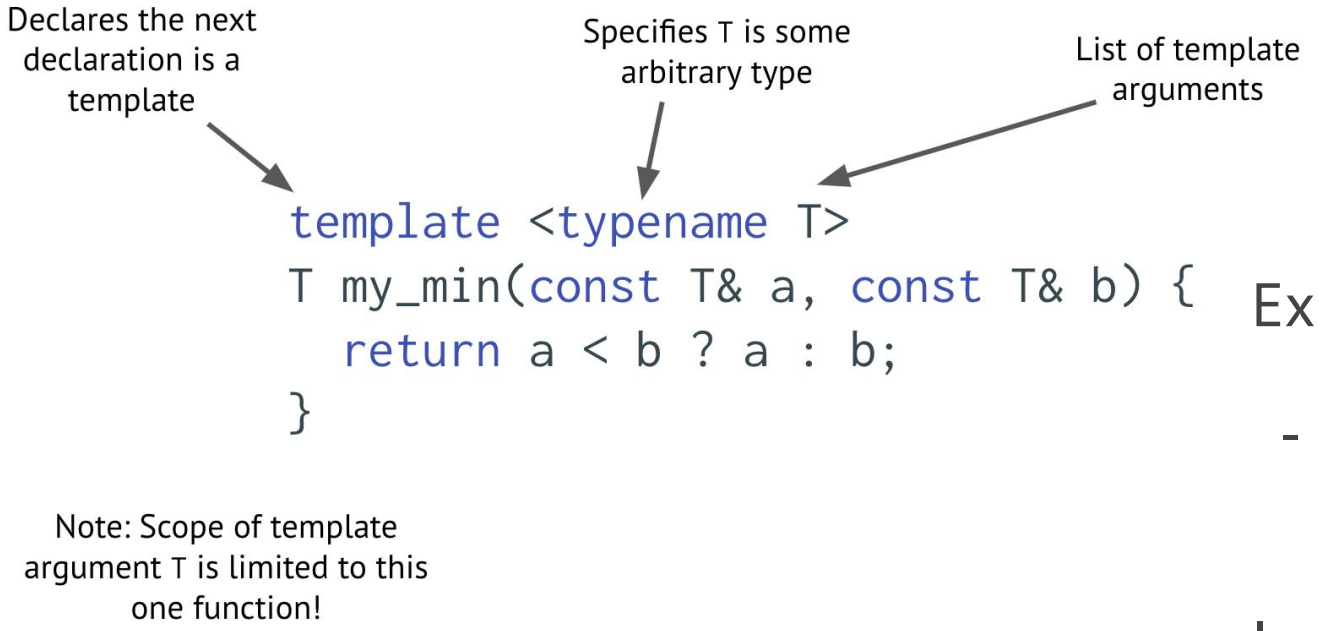
Explicit instantiation:

- `my_minmax<string>("Nikhil", "Ethan");`

Implicit instantiation:

- my_minmax(3, 6);
- my_minmax("Nikhil", "Ethan"); won't do as you expect! Will deduce C-strings

Templates Recap



Explicit instantiation:

- `my_minmax<string>("Nikhil", "Ethan");`

Implicit instantiation:

- my_minmax(3, 6);
- my_minmax("Nikhil", "Ethan"); won't do as you expect! Will deduce C-strings

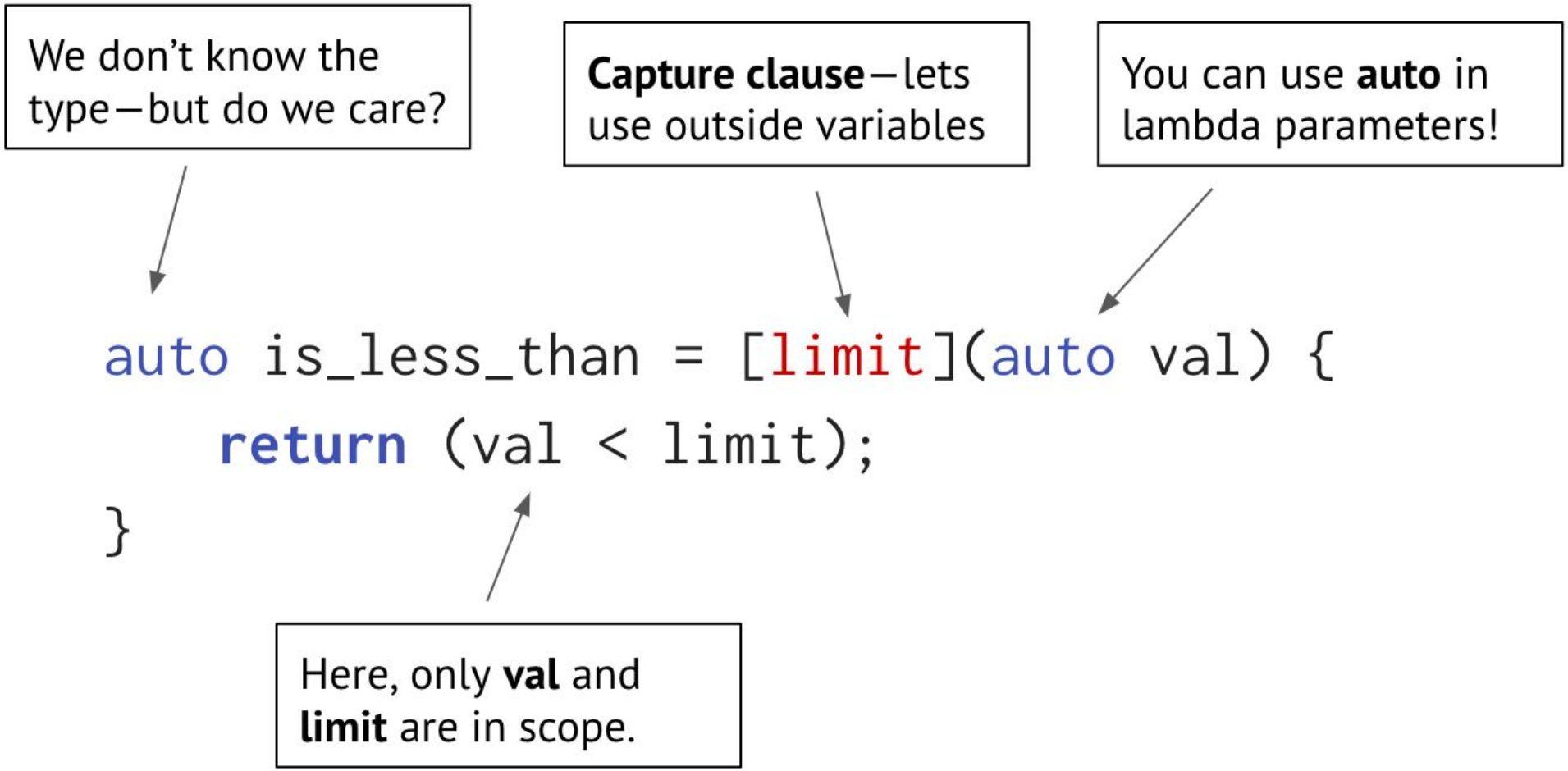
Notice how our template code makes implicit assumptions about the template arguments (e.g., they are comparable via <)

Challenge #4: Templates (if time)

```
int countOccurrences(const string& text, const string& feature)
```

Lambdas Recap

Lambdas Recap



Challenge #5: Lambdas

```
string fileToString(ifstream& file)
```

Algorithms Recap

Algorithms Recap (and brief intro to new stuff)

Example algorithms:

- `std::sort`
- `std::find`
- `std::count`
- `std::nth_element`
- `std::stable_partition`
- `std::copy`
- `std::copy_if`
- `std::remove_if`
- and more!

Algorithms Recap (and brief intro to new stuff)

Example algorithms:

- `std::sort`
- `std::find`
- `std::count`
- `std::nth_element`
- `std::stable_partition`
- `std::copy`
- `std::copy_if`
- `std::remove_if`
- and more!

Special iterators:

- back_inserter

```
std::copy(vec.begin(), vec.end(), std::back_inserter(newVec));
```

- stream_iterator

```
std::copy(vec.begin(), vec.end(),
        std::ostream_iterator<int>(cout,","));
```

Challenge #6: Algorithms (if time)

```
int dotProduct(const vector<int>& v1, const vector<int>& v2)
```

STL Wrap-Up:
Let's put it all together!

THE

FEDERALIST:

A COLLECTION OF

ESSAYS,

WRITTEN IN FAVOUR OF THE

NEW CONSTITUTION,

AS AGREED UPON BY THE

FEDERAL CONVENTION,

SEPTEMBER 17, 1787.

IN TWO VOLUMES.
VOL. I.

NEW-YORK:
PRINTED AND SOLD BY JOHN TIEBOUT,
No. 358 PEARL-STREET.
1799.

THE

FEDERALIST:

A COLLECTION OF

ESSAYS,

WRITTEN IN FAVOUR OF THE

NEW CONSTITUTION,

AS AGREED UPON BY THE

FEDERAL CONVENTION,

SEPTEMBER 17, 1787.

IN TWO VOLUMES.
VOL. I.

NEW-YORK:
PRINTED AND SOLD BY JOHN TIEBOUT,
No. 358 PEARL-STREET.
1799.

...will be printed on a fine Paper and good Type, in one handsome Volume duodecimo, and delivered to subscribers at the moderate price of one dollar. A few copies will be printed on superfine royal writing paper, price ten shillings.

No money required till delivery.

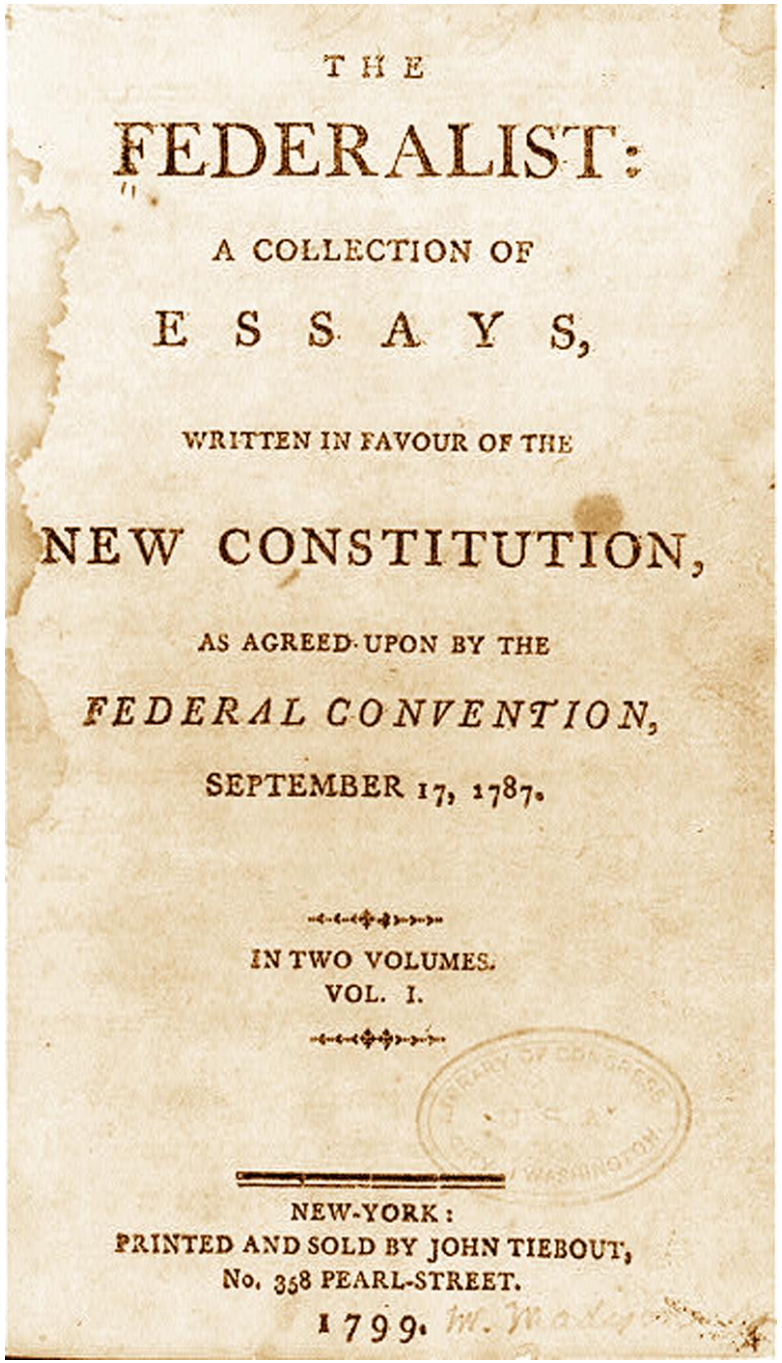
To render this work more complete, will be added, without any additional expence,

PHILO-PUBLIUS,

AND THE

Articles of the Convention,

As agreed upon at Philadelphia, September 17th, 1787.



THE

FEDERALIST:

A COLLECTION OF

ESSAYS,

WRITTEN IN FAVOUR OF THE

NEW CONSTITUTION,

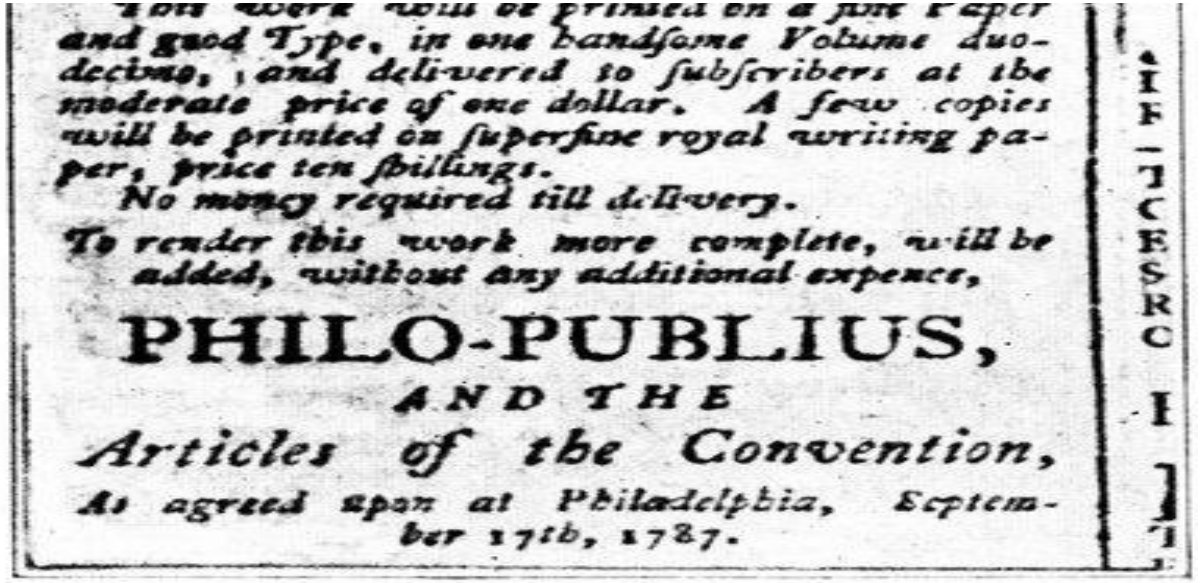
AS AGREED UPON BY THE

FEDERAL CONVENTION,

SEPTEMBER 17, 1787.

IN TWO VOLUMES.
VOL. I.

NEW-YORK:
PRINTED AND SOLD BY JOHN TIEBOUT,
No. 358 PEARL-STREET.
1799.



and good Type, in one handſome Volume duodecimo, and delivered to ſubſcribers at the moderate price of one dollar. A few copies will be printed on ſuperfine royal writing paper, price ten ſhillings.

No money required till delivery.

To render this work more complete, will be added, without any additional expence,

PHILO-PUBLIUS,

AND THE

Articles of the Convention,

As agreed upon at Philadelphia, September 17th, 1787.

The FŒDERALIST, No. 10.

To the People of the State of New-York.

AMONG the numerous advantages promiſed by a well conſtructed Union, none deſerves to be more accurately developed than its tendency to break and control the violence of faction. The friend of popular governments, never finds himſelf ſo much alarmed for their character and fate, as when he contemplates their propenſity to this dangerous vice. He will not fail therefore to ſet a due value on any plan which, without violating the principles to which he is attached, provides a proper cure for it. The inſtability, injuſtice and confuſion introduced into the public councils, have in truth been the mortal diſeaſes under which popular governments have every where periſhed; as they continue to be the favorite and fruitful topics from which the adverſaries to liberty derive their moſt ſpecious declamations. The valuable improvements made by the American Conſtitutions on the popular models, both ancient and modern, cannot certainly

The influence of factious leaders may kindle a flame within their particular States, but will be unable to ſpread a general conflagration through the other States: A religious ſect, may degenerate into a political faction in a part of the confederacy; but the variety of ſects diſperſed over the entire face of it, muſt ſecure the national Councils againſt any danger from that ſource: A rage for paper money, for an abolition of debts, for an equal diviſion of property, or for any other improper or wicked project, will be leſs apt to pervade the whole body of the Union, than a particular member of it; in the ſame proportion as ſuch a malady is more likely to taint a particular county or diſtrict, than an entire State.

In the extent and proper ſtructure of the Union, therefore, we behold a republican remedy for the diſeaſes moſt incident to republican Government. And according to the degree of pleaſure and pride, we feel in being Republicans, ought to be our zeal in cheriſhing the ſpirit and ſupporting the character of Fœderaliſts.

PUBLIUS.

Can we discover an author's identity from their writing?

Can we discover an author's identity from their writing?

stylometry noun

sty·lom·e·try | \ stī'lämə·trē, -tri\

plural -es

Definition of *stylometry*

: the study of the chronology and development of an author's work based especially on the recurrence of particular turns of expression or trends of thought

The Idea

Authors have an underlying writing style.

Subconsciously writers tend to write in a consistent manner.

...

The Idea

Authors have an underlying writing style.

Subconsciously writers tend to write in a consistent manner.

...

Could we use these tendencies as a literary fingerprint?

The Idea

We need a writer invariant.

The Idea

We need a writer invariant.

Function words:

- Syntactic glue of a language
- E.g. *the, I, he, she, do, from, because...*

The Idea

Let's imagine our language only has 3 function words:

```
[I, the, there]
```

Deep into that darkness peering, long I stood there, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

We can create a fingerprint vector for the two texts.

`[I, the, there]`

Deep into that darkness peering, long I stood there, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

```
[I, the, there]

[ 0 ,  0 ,  0 ]
```

Deep into that darkness peering, long I stood there, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

```
[I, the, there]

[ 0 ,  0 ,  0 ]
```

Deep into that darkness peering, long I stood there, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

```
[I, the, there]

[ 0 ,  0 ,  0 ]
```

Deep into that darkness peering, long I stood there, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

```
[I, the, there]

[ 1 ,  0 ,  0 ]
```

Deep into that darkness peering, long I stood there, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

[I, **the**, there]

[1 , 0 , 0]

Deep into that darkness peering, long I stood there, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

[I, the, **there**]

[1 , 0 , 0]

Deep into that darkness peering, long I stood **there**, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

[I, the, **there**]

[1 , 0 , **0**]

Deep into that darkness peering, long I stood **there**, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

```
[I, the, there]

[ 1 ,  0 ,  1 ]
```

Deep into that darkness peering, long I stood there, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

We can repeat this procedure for our other text, too

```
[I, the

[ 1 ,
```

Deep into that darkne
there, wondering, fea
dreams no mortal eve

- Edgar Allan Poe

ny wife and I split up. I had
s that I won't bother to talk
ning to do with the
y feeling that everything
there was dead.

- Jack Kerouac

The Idea

[I, the, there]

[1 , 0 , 1]

Deep into that darkness peering, long I stood there, wondering, fearing, doubting, dreaming dreams no mortal ever dared to dream before.

- Edgar Allan Poe

[I, the, there]

[4 , 1 , 1]

I first met Dean not long after my wife and I split up. I had just gotten over a serious illness that I won't bother to talk about, except that it had something to do with the miserably weary split-up and my feeling that everything there was dead.

- Jack Kerouac

The Idea

```
[I, the, there]          [I, the, there]
[ 1 ,  0 ,  1 ]          [ 4 ,  1 ,  1 ]
```

The Idea

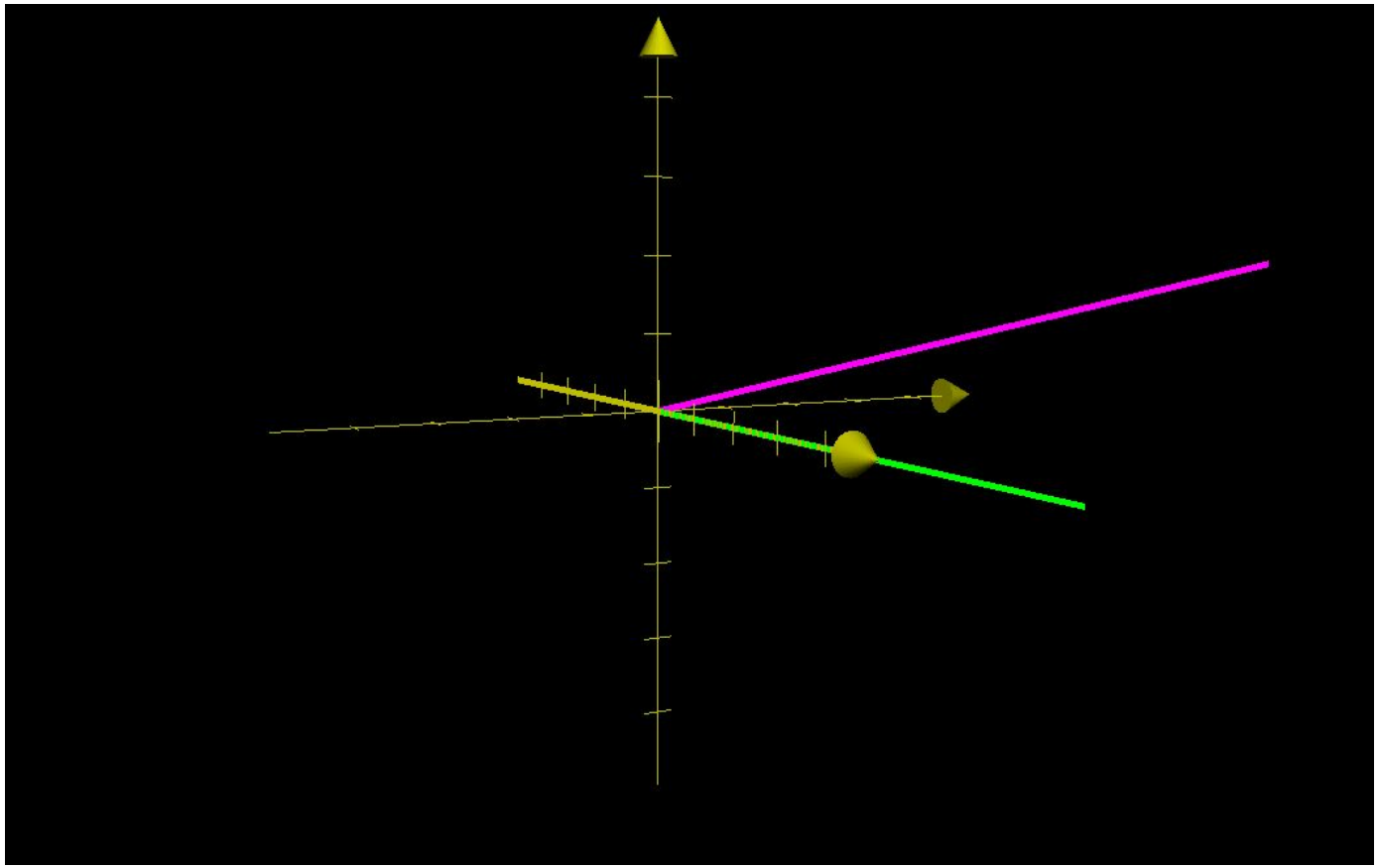
```
[I, the, there]          [I, the, there]
[ 1 ,  0 ,  1 ]          [ 4 ,  1 ,  1 ]
```

Now that we have vector representations of the frequencies of our function words in the excerpts, can we use math to compute how similar the two texts are to each other?

The Idea

```
[ 1 , 0 , 1 ]
```

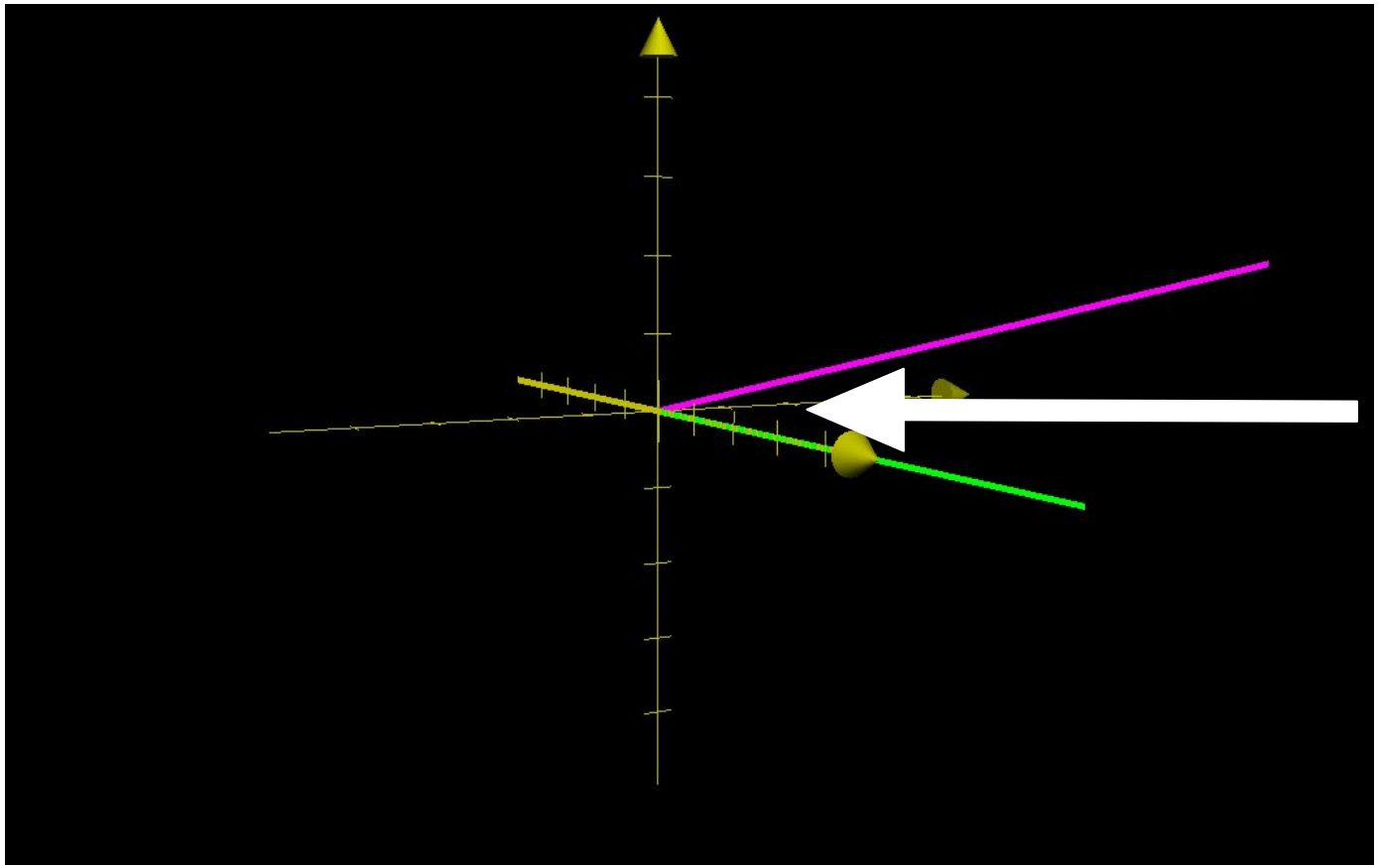
```
[ 4 , 1 , 1 ]
```



The Idea

```
[ 1 , 0 , 1 ]
```

```
[ 4 , 1 , 1 ]
```



The closer this angle, the more similar the texts

The Idea

$$\cos\theta = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|}$$

Let's call our two vectors u and v. A big value of cosine using this equation means that the texts are very similar, and a small value of cosine means the texts are different.

How can we leverage these principles to compute who wrote an unknown Federalist paper? Open up the "Stylometry" project and see if you can code up a solution using the hints in main.cpp!

Congratulations!

"As mathematicians learned to lift theorems into their most general setting, so I wanted to lift algorithms and data structures."

— *Alex Stepanov, inventor of the STL*

Next time:

Template Classes