

# The STL Laboratory:

## Bringing It All Together

# Game Plan



- Mid-Quarter Survey
- The Big STL Recap
- Let's Put It All Together!

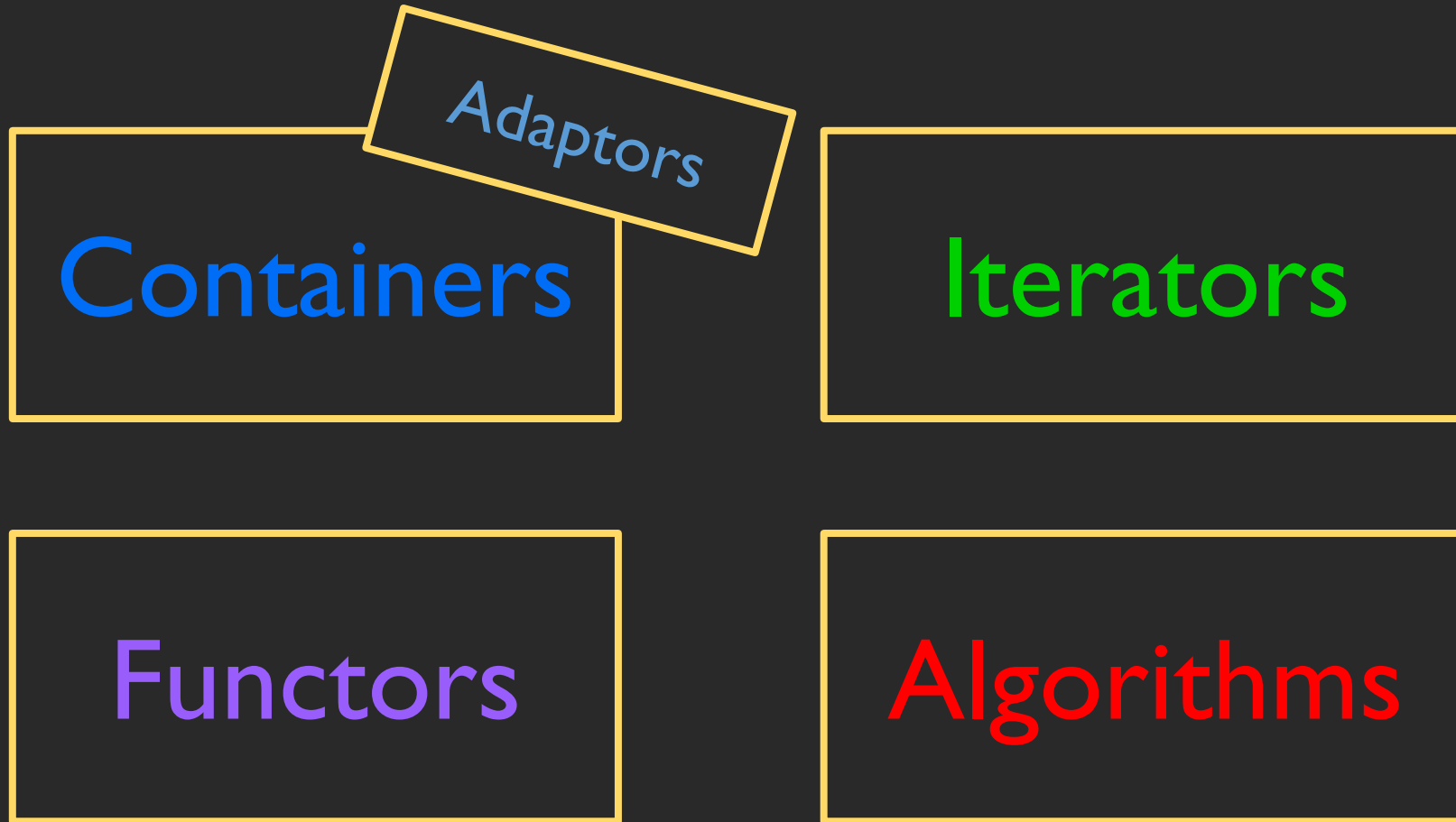
# Mid-Quarter Survey

due May 11th (1 week from now)

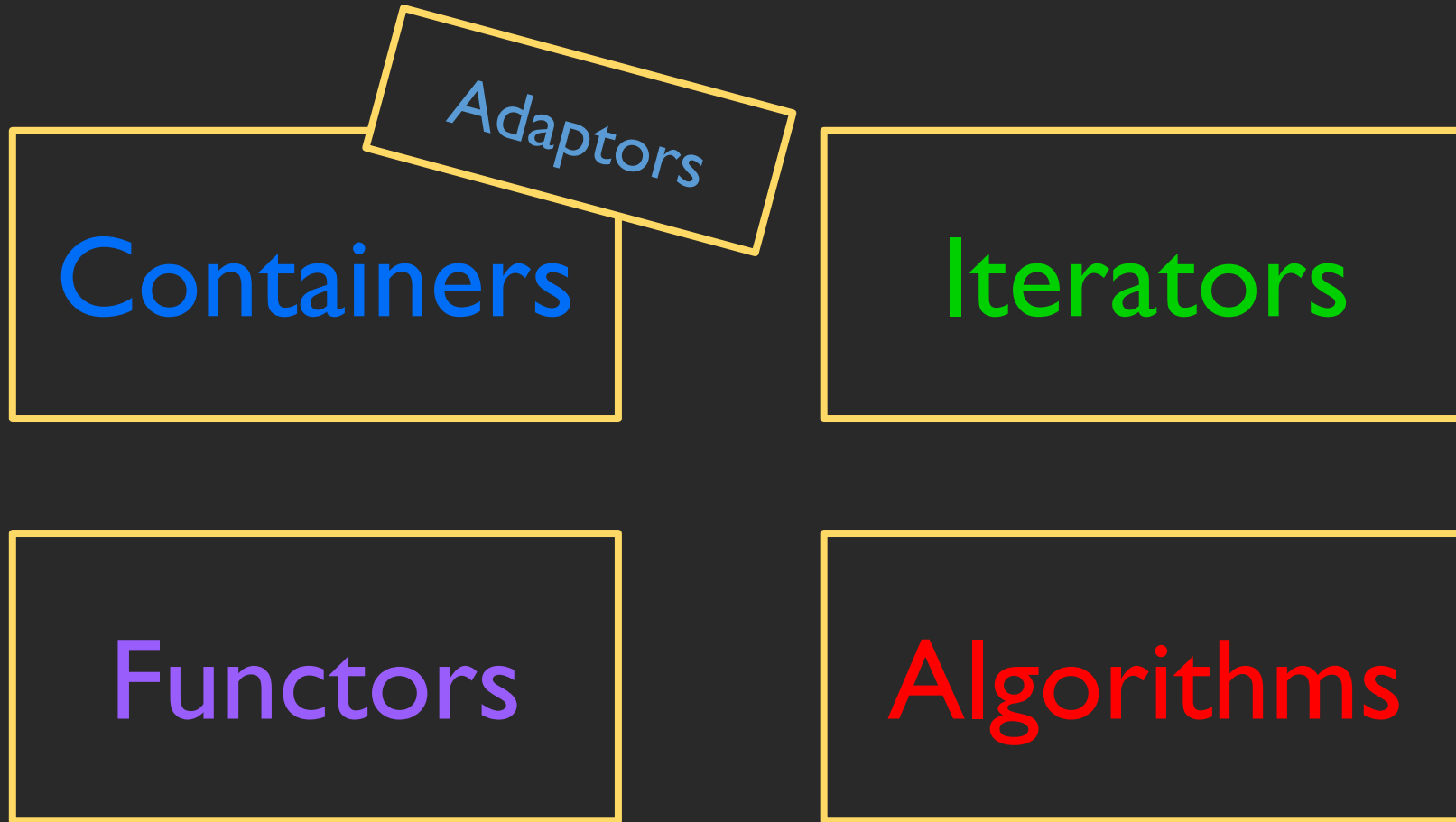
<https://forms.gle/L8HwaVFzjGjZVf6a9>

+1 late day!

# Overview of STL



You've now seen it all!



# 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):

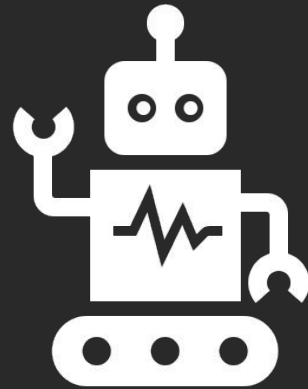
- `cin >> myString >> myInt;`
- `getline(stream, line);`

```
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)
- list, forward\_list

## 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)
- list, forward\_list

## vectors / deques:

- `vector<int> v{7, 8, 9};`
- `v.push_back(10);`
- `v.pop_back();`
- `v[3]` fails silently! `v.at(3)` throws error
- deques only: `push_front(6)`, `pop_front()`

## Container Adaptors:

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

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
- multimap
- multiset  
(allows repeated keys)

(hashed, fast for single elems:)

- unordered\_map
- unordered\_set
- unordered\_multimap
- unordered\_multiset



# Containers Recap Pt. 2

## Associative Containers:

(sorted, fast for range:)

- map
- set
- multimap
- multiset

(allows repeated keys)

(hashed, fast for single elems:)

- unordered\_map
- unordered\_set
- unordered\_multimap
- unordered\_multiset

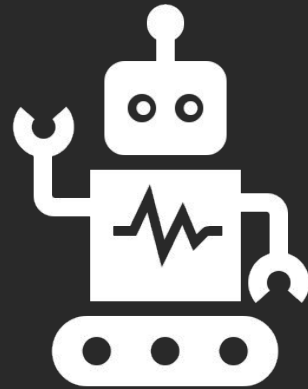
map:

- `map<int, string> m{{5, "Hi"}, {80, "Bye"}};`
- `m[106] = "C++";`
- `m.count(106);`
- `m.at(99) = "Hey";` // throws error
- `m[99] = "Hey";` // creates new entry

set:

- `set<int> s{1, 0, 6};`
- really a map to 0/1, without `.at()` and `[]`

# Challenge #2: Containers



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

# Iterators Recap

An iterator is like “the claw”.

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

# 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
- Maps return iterators to pairs:
  - `auto iter = map.begin(); auto key = iter->first;`
  - `auto [key, val] = map.begin()`

# 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:

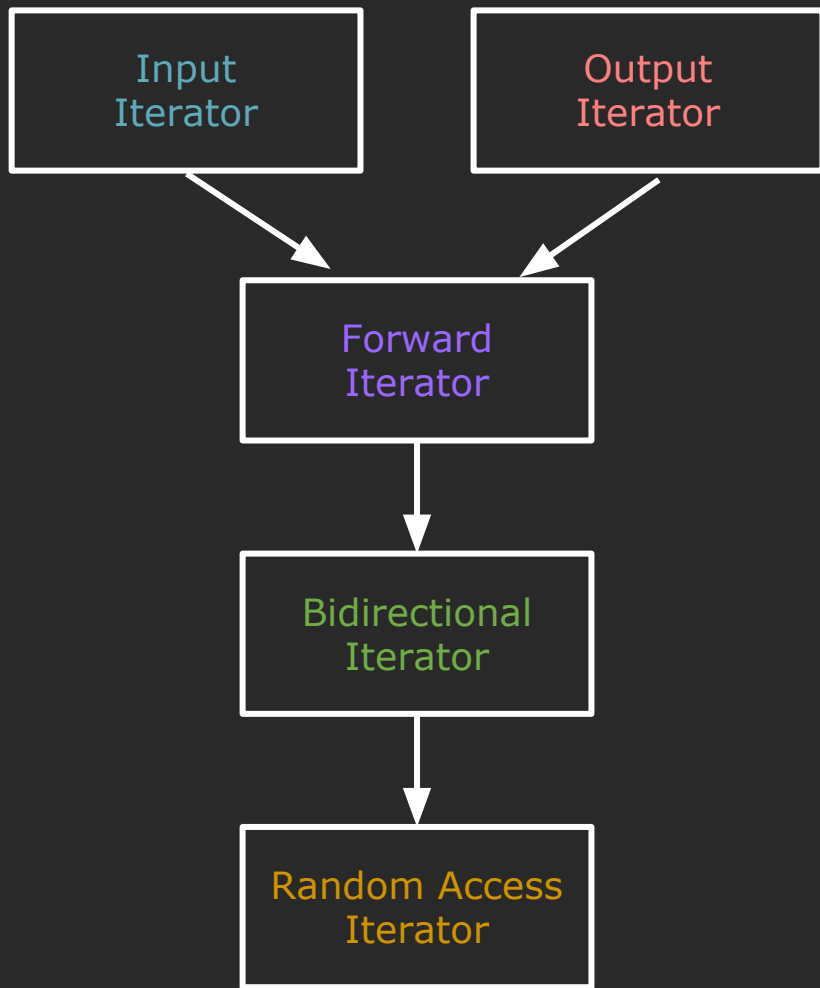
- `begin()` - iterator to the first element
- `end()` - iterator **one past** the last element
- Maps return iterators to pairs:
  - `auto iter = map.begin(); auto key = iter->first;`
  - `auto [key, val] = map.begin()`

for-each loop is just iterators!

```
for (int i : s)
    vs.
for (auto it = s.begin(); it != s.end(); ++it)
```

# Iterators Recap

Types:



An iterator is like “the claw”.

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

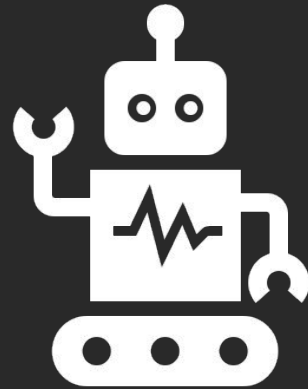
STL containers support:

- `begin()` - iterator to the first element
- `end()` - iterator **one past** the last element
- Maps return iterators to pairs:
  - `auto iter = map.begin(); auto key = iter->first;`
  - `auto [key, val] = map.begin()`

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)
```

# 2-min stretch break!

Bad Dad Joke of the Day:

- Today, my son asked "Can I have a book mark?" and I burst into tears. 11 years old and he still doesn't know my name is Brian.

Creds: Clayton



### BONUS Bad Dad Joke:

- Why do Python programmers wear glasses?
- They can't C#!

Creds: Harrison

## 2-min stretch break!

### Bad Dad Joke of the Day:

- Today, my son asked "Can I have a book mark?" and I burst into tears. 11 years old and he still doesn't know my name is Brian.

Creds: Clayton

# Templates Recap

# Templates Recap

Declares the next  
function is a template.

Specifies T is some  
arbitrary type.

List of template  
arguments.

```
template <typename T>  
pair<T, T> my_minmax(T a, T b) {  
    if (a < b) return {a, b};  
    else return {b, a};  
}
```

Scope of template  
argument T limited to  
function.

# Templates Recap

Declares the next  
function is a template.

Specifies T is some  
arbitrary type.

List of template  
arguments.

```
template <typename T>  
pair<T, T> my_minmax(T a, T b) {  
    if (a < b) return {a, b};  
    else return {b, a};  
}
```

Scope of template  
argument T limited to  
function.

Explicit instantiation:

- my\_minmax<string>("Avery",  
"Anna");

# Templates Recap

Declares the next function is a template.

Specifies T is some arbitrary type.

List of template arguments.

```
template <typename T>
pair<T, T> my_minmax(T a, T b) {
    if (a < b) return {a, b};
    else return {b, a};
}
```

Scope of template argument T limited to function.

Explicit instantiation:

- my\_minmax<string>("Avery", "Anna");

Implicit instantiation:

- my\_minmax(3, 6);
- my\_minmax("Avery", "Anna");  
won't do as you expect! Will deduce C-strings

# Templates Recap

Declares the next function is a template.

Specifies T is some arbitrary type.

List of template arguments.

```
template <typename T>  
pair<T, T> my_minmax(T a, T b) {  
    if (a < b) return {a, b};  
    else return {b, a};  
}
```

Scope of template argument T limited to function.

Explicit instantiation:

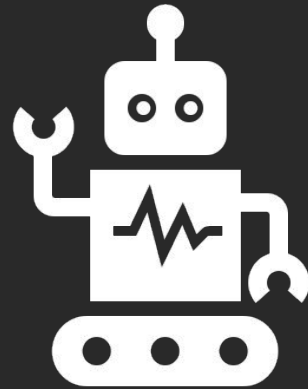
- my\_minmax<string>("Avery", "Anna");

Implicit instantiation:

- my\_minmax(3, 6);
- my\_minmax("Avery", "Anna");  
won't do as you expect! Will deduce C-strings

Concept (C++20) = turns the implicit assumptions that your code is making, into explicit requirements

# Challenge #4: Templates



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

# Lambdas Recap



# Lambdas Recap

We don't know the type, ask compiler.

capture clause,  
gives access to outside  
variables

parameter list,  
can use auto!

return type,  
optional



The diagram illustrates the syntax of a C++ lambda expression with four annotations and arrows pointing to specific parts of the code. The code is enclosed in large square brackets. The annotations are: 'We don't know the type, ask compiler.' pointing to 'auto'; 'capture clause, gives access to outside variables' pointing to '[limit]'; 'parameter list, can use auto!' pointing to '(auto val)'; and 'return type, optional' pointing to 'bool'.

```
[ auto isLessThanLimit = [limit](auto val) -> bool {  
    return val < limit;  
} ]
```

Scope of lambda limited to capture  
clause and parameter list.

# Lambdas Recap

We don't know the type, ask compiler.

capture clause,  
gives access to outside  
variables

parameter list,  
can use auto!

return type,  
optional

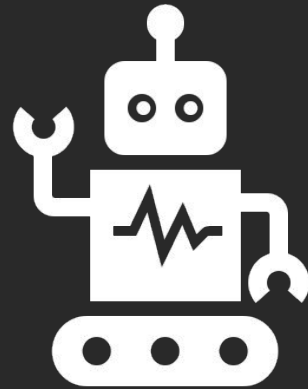
```
[ auto isLessThanLimit = [limit](auto val) -> bool {  
    return val < limit;  
} ]
```

Scope of lambda limited to capture  
clause and parameter list.

Capture clause:

- [=, &obj] → captures everything by value, except obj by reference
- [&, limit] → captures everything by reference, except limit by value

# Challenge #5: Lambdas



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

# Algorithms Recap

# Algorithms Recap

Algorithms we've seen:

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

# Algorithms Recap

## Algorithms we've seen:

- `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, ", "));`

# Algorithms Recap

## Algorithms we've seen:

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

## Special iterators:

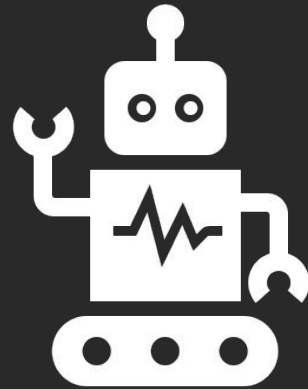
- `back_inserter`
  - `std::copy(vec.begin(), vec.end(), std::back_inserter(newVec));`
- `ostream_iterator`
  - `std::copy(vec.begin(), vec.end(), std::ostream_iterator<int>(cout, ", "));`

## Erase-remove idiom using algorithms\*:

```
std::erase(
    std::remove_if(v.begin(), v.end(), pred),
    v.end()
);
```

\*many containers will define their own erase function which does this for you - this only applies if you use the STL erase/remove algorithms

# Challenge #6: Algorithms



```
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. *M. Mady*

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. *W. Madison*

This work will be printed on a fine Paper  
and good Type, in one handsome Volume duo-  
decimo, and delivered to subscribers at the  
moderate price of one dollar. A few copies  
will be printed on superfine royal writing pa-  
per, 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, Septem-  
ber 17th, 1787.

PHILO-PUBLIUS



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. *W. Madison*

This work 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.

PHILO-PUBLIUS

The FÆDERALIST, No. 10.

*To the People of the State of New-York.*

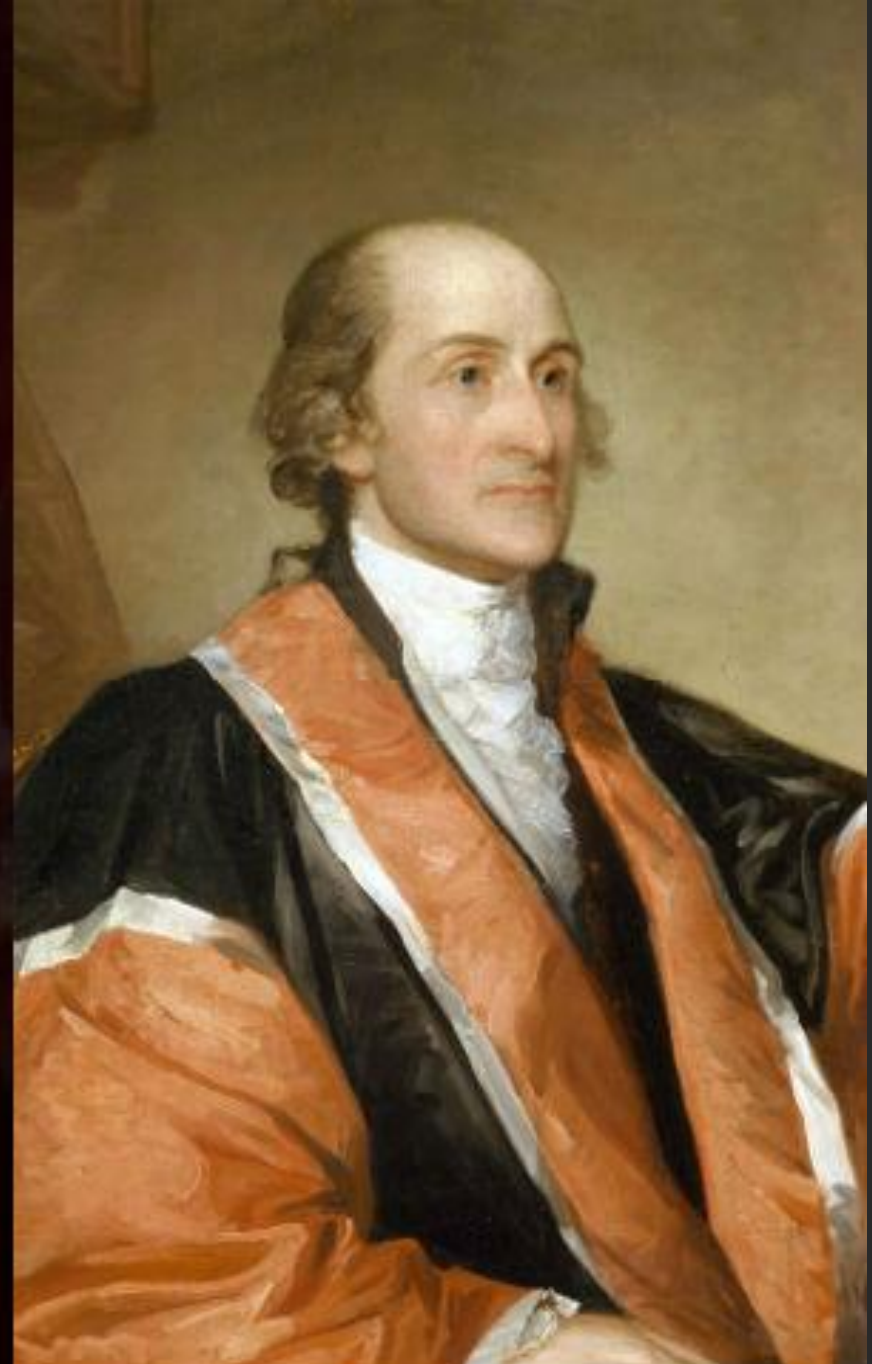
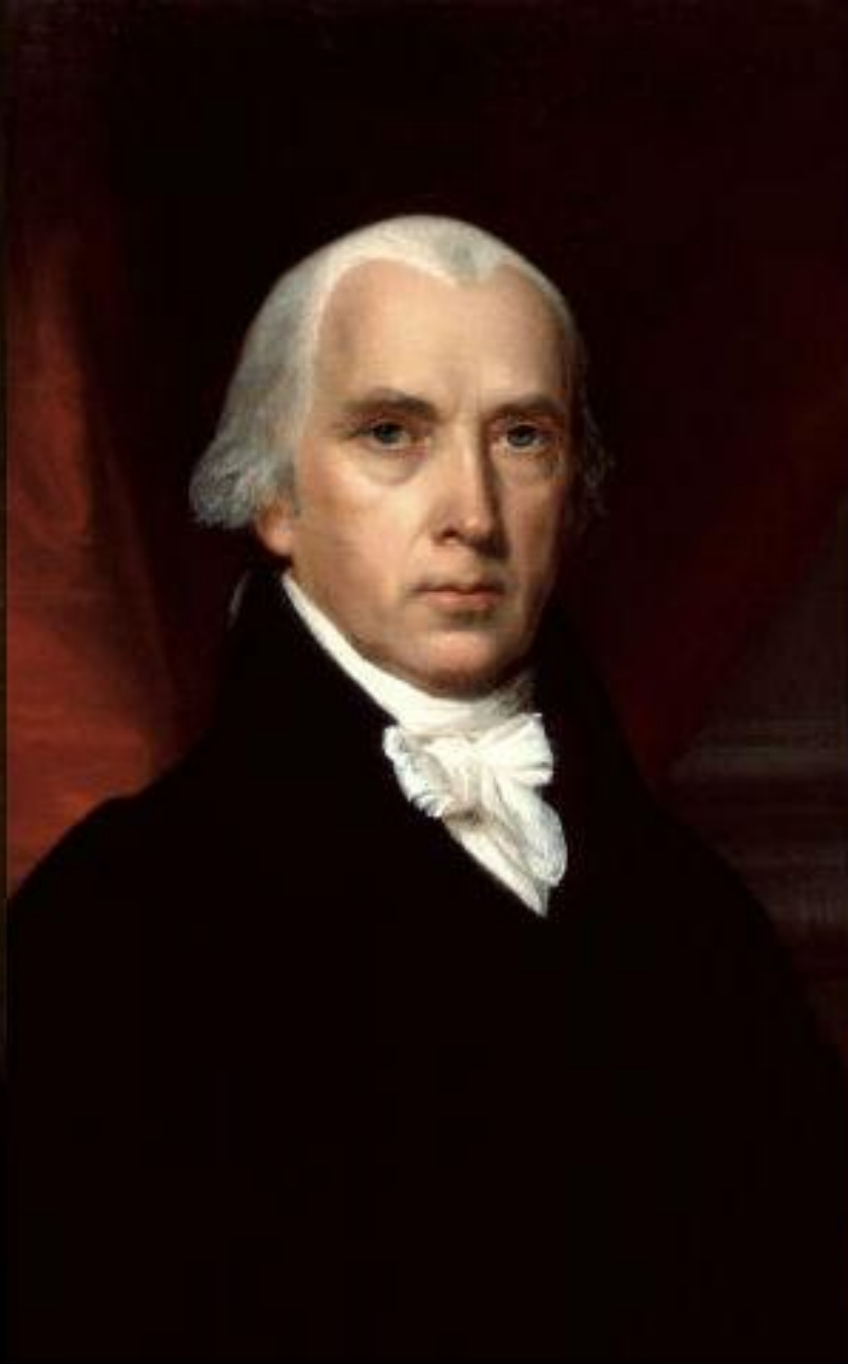
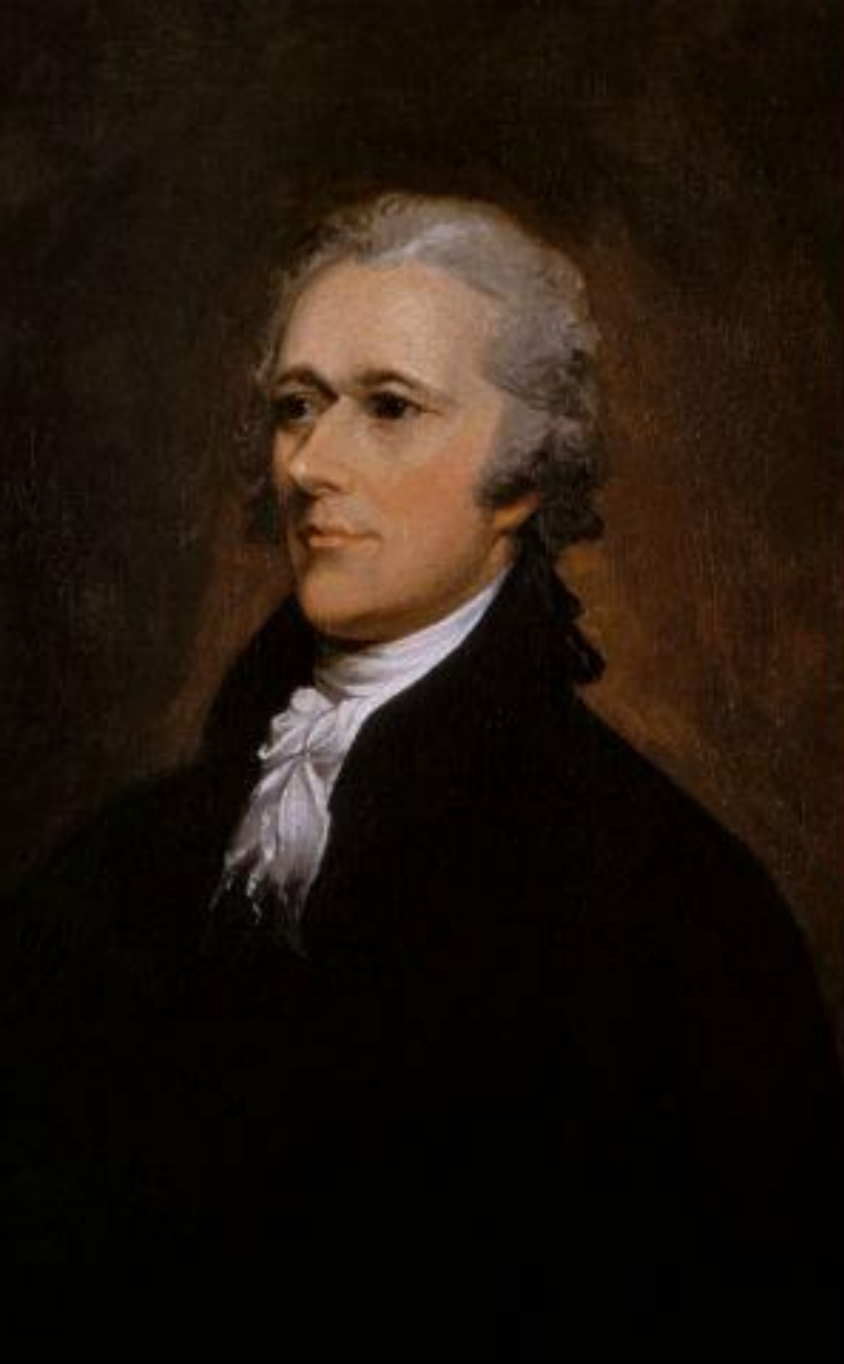
**A**MONG the numerous advantages promised by a well constructed Union, none deserves to be more accurately developed than its tendency to break and control the violence of faction. The friend of popular governments, never finds himself so much alarmed for their character and fate, as when he contemplates 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 possibly

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 Federalists.

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

[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

[ 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]

[ 0 , 0 , 0 ]

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

[ 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 ]

[ 0 , 0 , 0 ]

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

[ 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 ]

[ 0 , 0 , 0 ]

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

[ 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 , 0 , 0 ]

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

[ 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 , 0 , 0 ]

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

[ 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 , 0 , 0 ]

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

[ 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 , 0 ]

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

[ 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 , 0 ]

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

[ 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 , 0 ]

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

[ 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]

[ 1 , 0 , 1 ]

[ 4 , 1 , 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

[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

[ 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

[ 1 , 0 , 1 ]

[ 4 , 1 , 1 ]

# The Idea

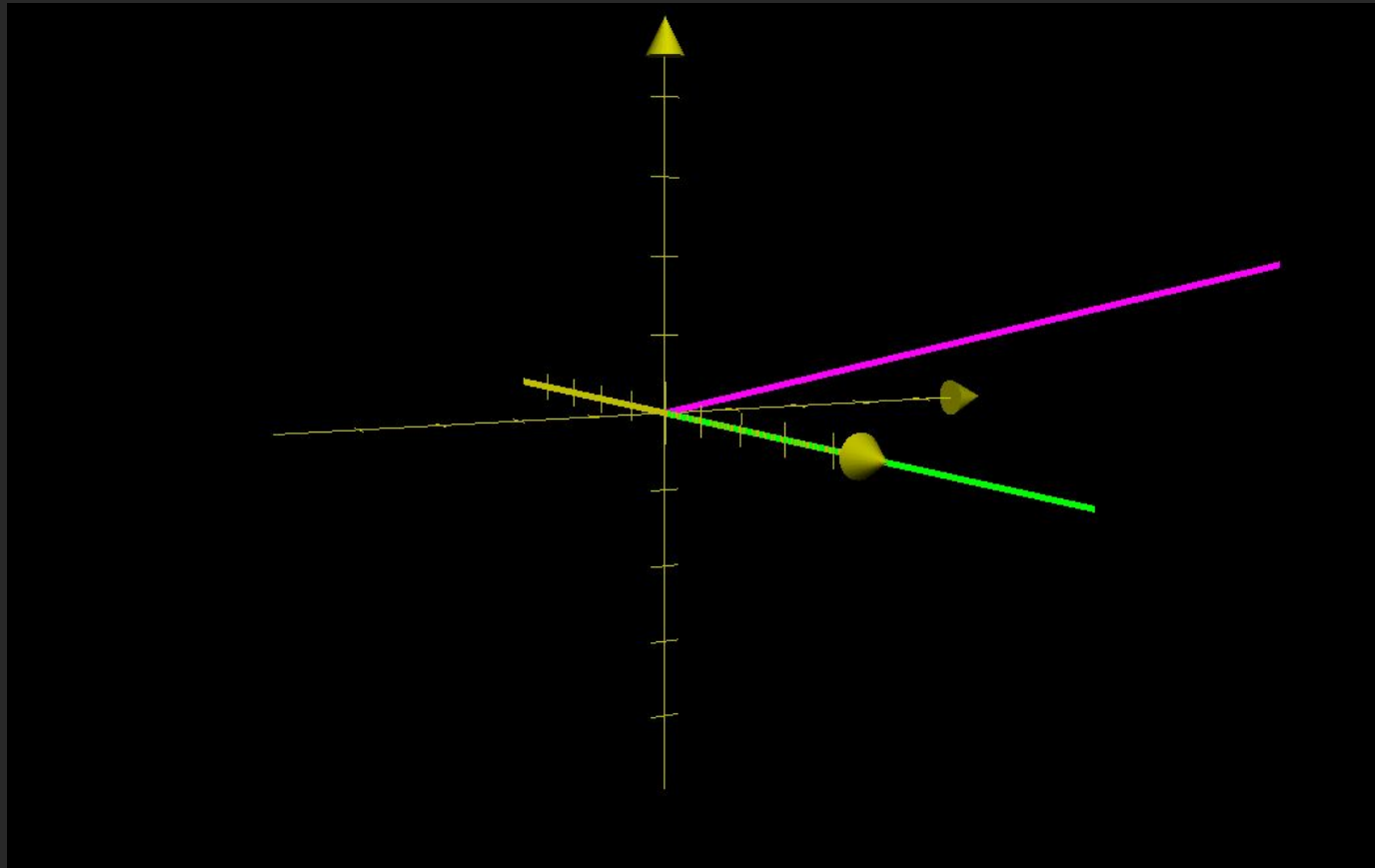
[ 1 , 0 , 1 ]

[ 4 , 1 , 1 ]

# The Idea

[ 1 , 0 , 1 ]

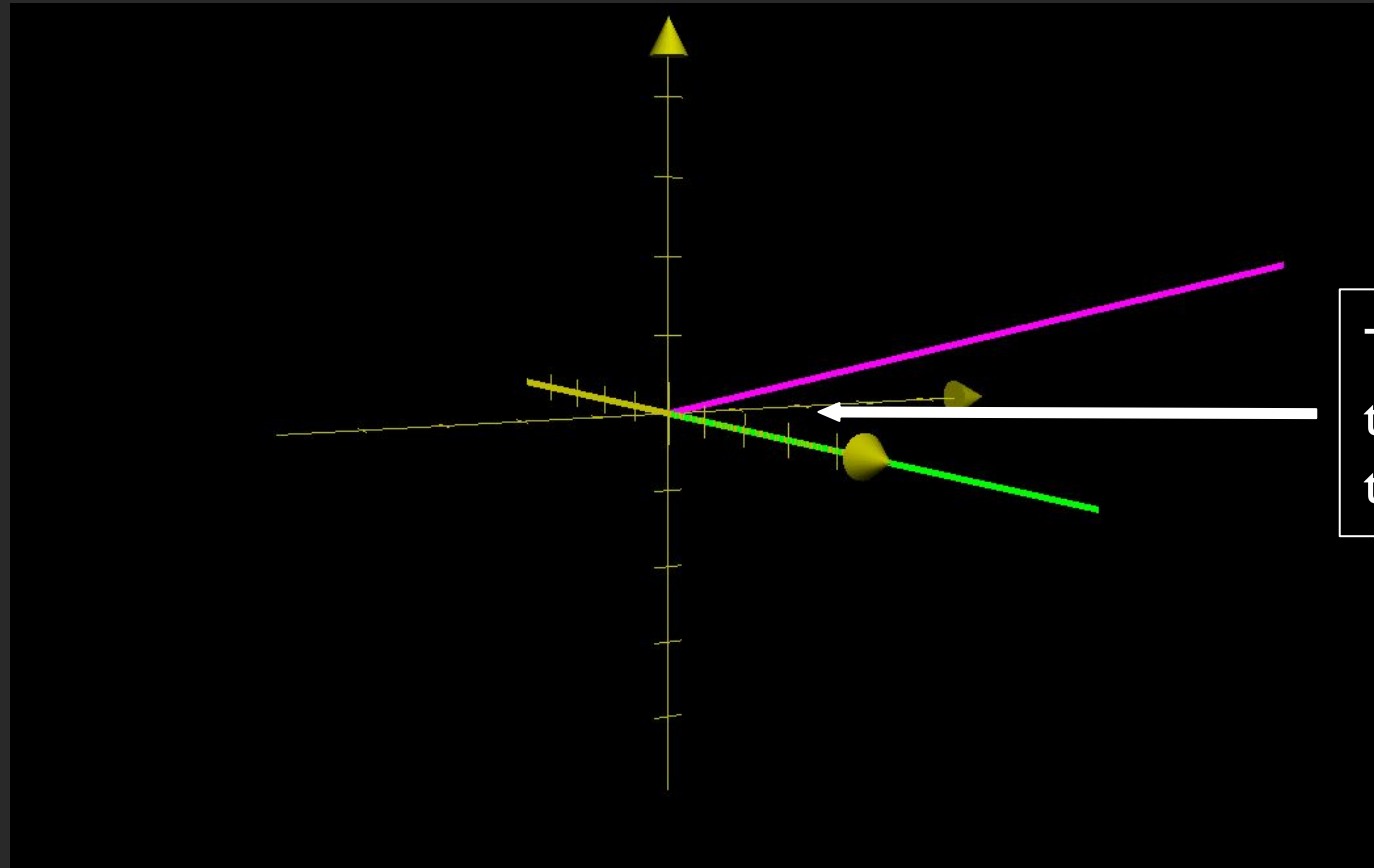
[ 4 , 1 , 1 ]



# The Idea

[ 1 , 0 , 1 ]

[ 4 , 1 , 1 ]



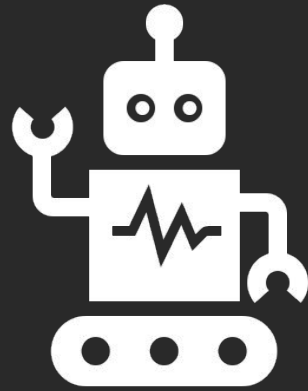
The closer this angle,  
the more similar the  
texts

# The Idea

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

Let's get coding!





# Example

## Stylometry

# Closing Notes

# Closing Notes

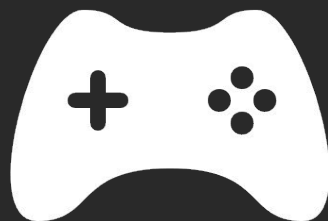
The code for getting the word count (i.e. `countOccurrences`)  
will be really useful for the first part of assignment 2.

# 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

Starting a new unit: Object-Oriented Programming  
(in Modern C++)!