Lecture 9: STL Summary

CS 106L, Winter ‘21
Today’s Agenda

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

- 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
- files
- a string
output streams (cout, ofstream):
- `cout << "5 + 2 = " << 7 << endl;`

input streams (cin, ifstream):
- `cin >> myString >> myInt;`
- `getline(stream, line);`
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);
}

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);`
Challenge #1: Streams

string fileToString(ifstream& file)
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

[hashed, fast for single elems:]

- unordered_map
- unordered_set
Containers Recap Pt. 2

Associative Containers:

(sorted, fast for range:)
- map
- set

[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)
An iterator allows us to iterate over any sequence.

- Copyable (iter = another_iter)
- Retrieve current element (*iter)
- Advance iterator (++iter)
- Equality comparable (iter != container.end())
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
Challenge #3: Iterators

```c
int countOccurrences(const string& text, const string& feature)
```
Announcements
InternetTest exercise due tonight!

- Please see linked troubleshooting doc if you encounter any issues
- Please confirm your screenshots look something like this:

![Image of navigation screenshot](https://example.com/image.png)

Take screenshot, then press enter to continue:
Mid-quarter survey!

● +1 late day!
● https://forms.gle/eEE2UAVnNwDMibN79
● Form is anonymous, and you’ll receive a link to another form to be recorded for your late day
No class on Thursday!

- Enjoy your day off :)
- If you’re in CS 106B, try to come to class on Tuesday caught up on 106B lectures
Return to STL
Templates Recap
Templates Recap

Declares the next declaration is a template

```
template <typename T>
T my_min(const T& a, const T& b) {
    return a < b ? a : b;
}
```

Specifies T is some arbitrary type

List of template arguments

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

Declares the next declaration is a template

```
template <typename T>
T my_min(const T& a, const T& b) {
    return a < b ? a : b;
}
```

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

Explicit instantiation:

- `my_min<string>`(“Nikhil”, “Ethan”);
Templates Recap

- **Explicit instantiation:**
  - `my_min<string>("Nikhil", "Ethan");`

- **Implicit instantiation:**
  - `my_min(3, 6);`

---

template <typename T>
T my_min(const T& a, const T& b) {
  return a < b ? a : b;
}

Declares the next declaration is a template

Specifies T is some arbitrary type

List of template arguments

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

Explicit instantiation:
- `my_min<string>`(“Nikhil”, “Ethan”);

Implicit instantiation:
- `my_min(3, 6);`

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

---

```cpp
template <typename T>
T my_min(const T& a, const T& b) {
    return a < b ? a : b;
}
```
Challenge #4: Templates (if time)

```cpp
int countOccurrences(const string& text, const string& feature)
```
Lambdas Recap
Lambdas Recap

- Usually, all functions are created by the compiler when we compile our code
- Lambda functions allow us to create functions during runtime
Lambdas Recap

We don’t know the type—but do we care?

Capture clause—lets use outside variables

You can use `auto` in lambda parameters!

```cpp
auto is_less_than = [limit](auto val) {
    return (val < limit);
};
```

Here, only `val` and `limit` are in scope.
Challenge #5: Lambdas

string fileToString(ifstream& file)
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!
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)

```c++
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. 338 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. 328 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.

NEW-YORK:
PRINTED AND SOLD BY JOHN TIEBOUT,
No. 328 PEARL-STREET.
1799.

PHILO-PUBLIUS,
AND THE Articles of the Convention,
As agreed upon at Philadelphia, September 17, 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 let a due value on any plan which, without violating the principles to which he is attached, provides a proper cure for it. The infallibility, 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 supply the

The influence of fœdrous 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 country; but the variety of fields 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 projects, 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 madness is more likely to take 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 profit 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*  
/stɪˈlɒmətrɪ, -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
Authors have an underlying **writing style**.

Subconsciously writers tend to write in a **consistent** manner.

...
The Idea

Authors have an underlying \textit{writing style}.

Subconsciously writers tend to write in a \textit{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...*
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
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
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
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
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
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
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
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
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
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
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

We can repeat this procedure for our other text, too

[1, 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

\[
[I, \text{ the, there}] \\
[1, 0, 1]
\]

\[
[I, \text{ the, there}] \\
[4, 1, 1]
\]
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

\[ [1, 0, 1] \quad [4, 1, 1] \]
The closer this angle, the more similar the texts
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.

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