Bad Dad Joke of the Day:
- How did the hamburger introduce his wife?
- Meat patty.

Creds: Julie
Game Plan

- Assignment 2 Preview
- BIG STL RECAP and Mystery Activity
- Let’s Put It All Together!
Preview of Assignment 2

Part A: due Friday, February 7 (no more than 2 hours)
Part B: due Thursday, February 13

Overview of STL

Containers

Iterators

Functors

Algorithms

Adaptors
You’ve now seen it all!
THE BIG STL RECAP,
aka the BSTLR*
THE BIG STL RECAP,
aka the BSTLR*

*not an official C++ acronym
Feedback Survey #1

Thanks for filling out the feedback survey!

We’re still processing the results, but one common theme for improvement was:

- “recaps”
- “coding exercises in class”
- “a bit too fast”
Feedback Survey #1

Thanks for filling out the feedback survey!

We’re still processing the results, but one common theme for improvement was:

- “recaps”
- “coding exercises in class”
- “a bit too fast”... and “good pace”
Feedback Survey #1

Thanks for filling out the feedback survey!

We’re still processing the results, but one common theme for improvement was:

- “recaps”
- “coding exercises in class”
- “a bit too fast”
Feedback Survey #1

Thanks for filling out the feedback survey!

We’re still processing the results, but one common theme for improvement was:

- “recaps”
- “coding exercises in class”
- “a bit too fast”

The BSTLR!
The BSTLR

1. Streams
2. Sequence containers + container adaptors
3. Associative containers
4. Iterators
5. Templates
6. Lambdas
7. Algorithms
Streams Recap

console & keyboard
files

a string
Streams Recap

- stringstream ss("Hello", stringstream::ate);
- ss << 106;
- string myString;
  ss >> myString;
Streams Recap

stringstream:
- stringstream ss("Hello", stringstream::ate);
- ss << 106;
- string myString; ss >> myString;

filestream:
- fstream fs(filename);
- string line; getline(fs, line);
Streams Recap

- stringstream:
  - stringstream ss(“Hello”, stringstream::ate);
  - ss << 106;
  - string myString; ss >> myString;

- filestream:
  - fstream fs(filename);
  - string line; getline(fs, line);

- cout/cin:
  - cout << “Hello” << endl;
  - cin >> myInt >> myString;
Streams Recap

stringstream:
- stringstream ss("Hello", stringstream::ate);
- ss << 106;
- string myString; ss >> myString;

filestream:
- fstream fs(filename);
- string line; getline(fs, line);

cout/cin:
- cout << "Hello" << endl;
- cin >> myInt >> myString;

State bits:
- good, fail, eof, bad
- fail fails silently!
Challenge #1: Streams

string fileToString(ifstream& file)
Sequence Containers + Adaptors Recap
Sequence Containers + Adaptors Recap

Sequence Containers:
- vector
  (fast access of middle)
- deque
  (fast insert begin/end)
- list, forward_list
Sequence Containers + Adaptors Recap

Sequence Containers:
- vector
  (fast access of middle)
- deque
  (fast insert begin/end)
- list, forward_list

Container Adaptors:
- stack
  ➔ secretly a vector or deque
- queue
  ➔ secretly a deque
Sequence Containers + Adaptors Recap

Sequence Containers:
- **vector**
  (fast access of middle)
- **deque**
  (fast insert begin/end)
- **list, forward_list**

Container Adaptors:
- **stack**
  ➔ secretly a vector or deque
- **queue**
  ➔ secretly a deque

vector:
- `vector<int> v{1, 0, 6};`
- `v.push_back(-100);`
- `v.pop_back();`
- `v[3]` fails silently! `v.at(3)` throws error
Sequence Containers + Adaptors Recap

Sequence Containers:
- **vector**
  (fast access of middle)
- **deque**
  (fast insert begin/end)
- list, forward_list

Container Adaptors:
- **stack**
  ➔ secretly a vector or deque
- **queue**
  ➔ secretly a deque

vector:
- vector<int> v[1, 0, 6];
- v.push_back(-100);
- v.pop_back();
- v[3] fails silently! v.at(3) throws error

deque:
- deque<int> d[1, 0, 6];
- same as a vector
- d.push_front(42);
- d.pop_front();
Sequence Containers + Adaptors Recap

Sequence Containers:
- vector
  (fast access of middle)
- deque
  (fast insert begin/end)
- list, forward_list

Container Adaptors:
- stack
  ➔ secretly a vector or deque
- queue
  ➔ secretly a deque

vector:
- vector<int> v{1, 0, 6};
- v.push_back(-100);
- v.pop_back();
- v[3] fails silently! v.at(3) throws error

deque:
- deque<int> d{1, 0, 6};
- same as a vector
- d.push_front(42);
- d.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();
Associative Containers Recap
Associative Containers Recap

Associative Containers:

(sorted, fast for range:)
- map
- set
- multimap
- multiset
  (allows repeated keys)
Associative Containers Recap

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
Associative Containers Recap

Associative Containers:

(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

Associative Containers:

(map: [sorted, fast for range:]
- map
- set
- multimap
- multiset
  (allows repeated keys)

(ordered, fast for single elems:)
- unordered_map
- unordered_set
- unordered_multimap
- unordered_multiset)
Associative Containers Recap

Associative Containers:

(map, set, multimap, multiset)

(map, set)

(map, set)

(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
Iterators Recap

Types:
- input
  (one-pass, read-only)
- output
  (one-pass, write-only)
- forward
  (multi-pass, read and write)
- bidirectional
  (multi-pass, read and write, can decrement)
- random access
  (multi-pass, read and write, can incr/decr by arbitrary amounts)
Iterators Recap

Types:
- input
  (one-pass, read-only)
- output
  (one-pass, write-only)
- forward
  (multi-pass, read and write)
- bidirectional
  (multi-pass, read and write, can decrement)
- random access
  (multi-pass, read and write, can incr/decr by arbitrary amounts)

Basic syntax:
- set<int> s{1, 0, 6};
- set<int>::iterator it = s.begin();
- auto it2 = s.end();
- ++it;
- *it = 3;
- if (it != it2) ...
- map<int, string> m{{1, “Hi”}, {6, “Bye”}};
- auto [key, val] = m.begin();
- (m.begin())->first = 3
Iterators Recap

Types:
- **input**
  - (one-pass, read-only)
- **output**
  - (one-pass, write-only)
- **forward**
  - (multi-pass, read and write)
- **bidirectional**
  - (multi-pass, read and write, can decrement)
- **random access**
  - (multi-pass, read and write, can incr/decr by arbitrary amounts)

Basic syntax:
- `set<int> s{1, 0, 6};`
- `set<int>::iterator it = s.begin();`
- `auto it2 = s.end();`
- `++it;`
- `*it = 3;`
- `if (it != it2) ...`
- `map<int, string> m{{1, “Hi”}, {6, “Bye”}};`
- `auto [key, val] = m.begin();`
- `(m.begin())->first = 3`

for-each loop:
- `for (int i : s) ...`
  - is implemented as
- `for (auto it = s.begin(); it != s.end(); ++it) ...`
Challenge #3: Iterators

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

Declares the next function is a template.

Specifies T is some arbitrary type.

List of template arguments.

```cpp
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

Explicit instantiation:
- `my_minmax<string>("Avery", "Anna");`

Scope of template argument T limited to function.

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

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

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

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

Scope of template argument T limited to function.

Declares the next function is a template.

Specifies T is some arbitrary type.

List of template arguments.
Challenge #4: Templates

```cpp
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

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

Scope of lambda limited to capture clause and parameter list.
Lambdas Recap

Capture clause:
- `[=, &obj]` → captures everything by value, except obj by reference
- `[]={, limit]` → captures everything by reference, except limit by value

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

Scope of lambda limited to capture clause and parameter list.
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
  - e.g., std::copy(vec.begin(), vec.end(), std::back_inserter(newVec));
- stream_iterator
  - e.g., 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
  - e.g., std::copy(vec.begin(), vec.end(),
    std::back_inserter(newVec));
- stream_iterator
  - e.g., std::copy(vec.begin(), vec.end(),
    std::ostream_iterator<int>(cout, " "));

Erase-remove idiom using algorithms*:
std::erase(
    std::remove(v.begin(), v.end()), 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

```cpp
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. 338 PEARL-STREET.
1799.

This work will be printed on a fine paper and good type, in one handsome volume double-decked, 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 expense, 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. 338 PEARL-STREET.
1799.

This work will be printed on a fine paper
and good type, in one handsome volume du-
dedicating, 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 expense,
PHILO-PUBLIUS,
AND THE
Articles of the Convention,
As agreed upon at Philadelphia, Septem-
ber 17th, 1787.

The FŒDERALIST, No. 10.
To the People of the State of New-York.

Among the numerous advantages promised by
a well-contructed 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 dan-
gerous 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 pro-
er cure for it. The infallibility, injustice and con-
fusion 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, are mainly

The influence of disinterested leaders may kindle a
flame within their particular States, but will be un-
able to spread a general conflagration through the
other States. A religious feeling, may degenerate into
a political passion 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 projec-
ts, will be leis 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 prudence and pru-
ence 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

/stɪˈlæmətri, -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
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
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
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
The Idea

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

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

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
The Idea

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

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
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
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
The Idea

[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

\[[1, 0, 1] \quad [4, 1, 1]\]
The Idea

\[ [1, 0, 1] \quad \quad \quad \quad \quad \quad [4, 1, 1] \]
The Idea

\[ [1, 0, 1] \quad [4, 1, 1] \]
The Idea

\[ [1, 0, 1] \quad [4, 1, 1] \]

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

Class Design with Iterators

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