601.220 Intermediate Programming

Spring 2023, Day 25 (March 27th)

Today's agenda

- Review exercise 24
- Day 25 recap questions
- Exercise 25

Reminders/Announcements

- HW5 is due **Friday, March 31st**
- Midterm project: you may submit by 11pm this evening

Exercise 24 review

```
// Part 3:
// map to store the occurrence count for each word
map<string, int> counters;

// read each word in the input, and update
// the occurrence counts
string word;
while (cin >> word) {
  counters[word]++;
}
```

Why does this work?

Exercise 24 review

In a map, each key/vaue pair is represented by a `std::pair` object.

For a `std::map<std::string, int>` collection, the pair type is `std::pair<std::string, int>`.

When using the subscript operator to refer to a key not currently in the map, a new `std::pair` object is created for that key.

The new pair's key is the one specified as the subscript. The new pair's value is created using the value type's *default constructor*.

Exercise 24 review

Primitive types (int, char, double, etc.) do have a default
constructor! Its behavior is to produce the value zero. So, when the
code does

```
counters[word]++;
```

→ *creates new pair
if key doesn't exist yet*

if the word does not yet exist as a key, a `pair` is created with the
`int` value set to 0, which is then incremented to 1.

Exercise 24 review

```
// Part 4
// create a map of occurrence counts to vectors of words
// with that occurrence count

map<int, vector<string>> words_by_freq;

for (map<string, int>::const_iterator i = counters.cbegin();
     i != counters.cend();
     ++i) {
  words_by_freq[i->second].push_back(i->first);
}
```

Again, this works because when a new `pair` is added to the
`words_by_freq` map, its `vector` is initialized using the default
constructor, so the `vector` is initially empty.

Exercise 24 review

```cpp
// Part 5
for (map<int, vector<string>>::const_iterator i =
       words_by_freq.cbegin();
     i != words_by_freq.cend();
     ++i) {
  cout << "Frequency: " << i->first << "\n";
  for (vector<string>::const_iterator j = i->second.cbegin();
       j != i->second.cend();
       ++j) {
    cout << *j << "\n";
  }
}
```

In the body of the outer loop, `i->first` is the occurrence count,
and `i->second` is the `vector` of `strings` representing the input
words with that occurrence count.

Exercise 24 review

Part 7: the std::sort function is in the <algorithm> header:

```
#include <algorithm>
```

Using std::sort to sort the vec2 vector:

```
std::sort(vec2.begin(), vec2.end());
```

vector
arrays

Note that std::sort requires random-access iterators, so it can't be used with collections with sequential iterators, such as std::list.

Exercise 24 review

Part 8: When I tried it:

```
$ ./sort
Enter the count: 10000000
Your sort time = 13301(ms)
STL's sort time = 2105(ms)
```

Merge sort is asymptotically optimal, but has relatively high per-element overhead due to the copying of data between the vector being sorted and the temporary vector (or array) used to hold the merged elements.

Day 25 recap questions

① How do you read and write files in C++?
② What is a stringstream in C++?
③ How do you extract the contents of a stringstream?
④ What does a constructor do?
⑤ What does a destructor do?

1. How do you read and write files in C++?

Read a file: std::ifstream

```
std::ifstream in("input.txt");
if (!in.is_open()) { /* error, couldn't open */ }

// ...use to read input, works just like any ifstream
//    (such as std::cin)..
```

Write a file: std::ofstream

```
std::ofstream out("output.txt");
if (!out.is_open()) { /* error, couldn't open */ }

// ...use to write output, works just like any ofstream
//    (such as std::cout)...
```

2. What is a stringstream in C++?

A `std::stringstream` allows you to read formatted input from a `std::string` or write formatted output to a `std::string`.

To use,

```cpp
#include <sstream>
```

std::stringstream example

```
std::string s = "foo bar 123", tok1, tok2;
int n;

std::stringstream in(s);
in >> tok1 >> tok2 >> n;
assert(tok1 == "foo");
assert(tok2 == "bar");
assert(n == 123);

std::stringstream out;
out << "Hello, n=" << n;
std::string s2 = out.str();
assert(s2 == "Hello, n=123");
```

3. How do you extract the contents of a stringstream?

Use the `.str()` member function (see previous slide.)

It returns the string data in the `stringstream` as a `std::string` value.

4. What does a constructor do?

A constructor initializes the member variables (a.k.a. fields) of a newly-constructed object.

"Object" = "instance of a class or struct type"

Every object is initialized by a call to a constructor when its lifetime begins

The call to the constructor happens before the object is used by the program.

5. What does a destructor do?

A destructor "cleans up" an object whose lifetime is ending.

The primary purpose of a destructor is de-allocating dynamic resources associated with the object.

Examples of resources requiring cleanup:

- dynamically-allocated memory, cleaned up by freeing the memory (e.g. string, vector, map, etc...)
- open file(s), cleaned up by closing (e.g. ifstream, ofstream)

The compiler will automatically invoke a destructor for any object declared as a local variable, when the function returns.

RAII
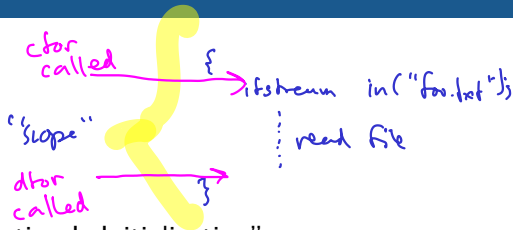
RAII = "Resource Allocation Is Initialization"

This is the principle that dynamic resources should be managed by an object. As long as the object's destructor is called (which happens automatically for all objects except dynamically-allocated ones), the programmer doesn't need to write special code to clean up resources.

Exercise 25

- Part 1: transform text by replacing sequences of vowels with `'`
- Part 2: classify tokens as integer, floating point, or non-numeric
- Part 3 (optional): determine letter frequencies of input text file
- Talk to us if you have a question!

✳ Note that for part 2, it might be helpful to create multiple
  `stringstream` objects to use to recognize a particular input token.

Notes

Notes

Notes

Notes

Notes