601.220 Intermediate Programming

Summer 2023, Meeting 16 (July 10th)

Today's agenda

- Review exercises 23 and 24
- Day 25 recap questions
- Exercise 25
- Day 26 recap questions
- Exercise 26

Reminders/Announcements

- HW5 is due **Thursday, July 13th**

Exercise 23 review

Read an input value into the variable `count`:

```
size_t count;

std::cin >> count;
```

Exercise 23 review

Make `vec` store `count` (pseudo-)random values:

```
std::vector< int > vec;

// ...

for (size_t i = 0; i < count; i++) {
  vec.push_back(::rand());
}
```

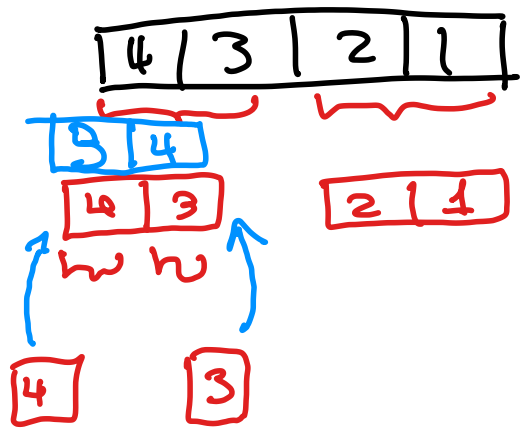
Note the `#include <cstdlib>` at the top of the source file. This shows how to include a C library header file in a C++ program.

Exercise 23 review

Pseudo code

```
void sort( std::vector< int > *values, int start, int end ) {
  int n = end - start;        // how many elements to sort?
  if (n < 2) { return; }      // base case
  int mid = start + n/2;
  sort(values, start, mid);
  sort(values, mid, end);
  merge(values, start, mid, end);
}

void sort( std::vector< int > *values ) {
  sort(values, 0, int(values->size()));
}
```



Exercise 23 review

Implementing `merge`: idea is to repeatly compare the smallest remaining elements from the left and right halves, and add the smaller element to the sorted result.

You'll need to use a temporary vector to hold the sorted result, and then copy it to overwrite the region being sorted.

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]++;
}
```

word => "hello"

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]++;
```

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

```
// 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

algorithms => they use iterators,

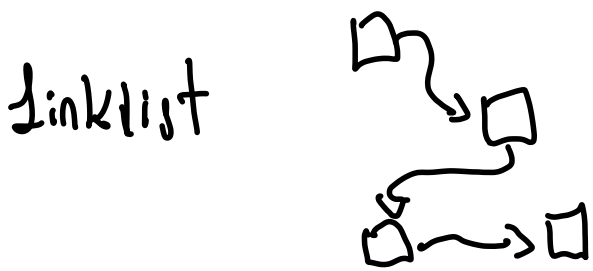
Part 7: the `std::sort` function is in the `<algorithm>` header:

```
#include <algorithm>
```

Using `std::sort` to sort the `vec2` vector:

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

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,

```
#include <sstream>
```

std::stringstream example

Stream modifiers.

```
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");
```

→ init stringstream.

cout << out.str();

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
- open file(s), cleaned up by closing

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

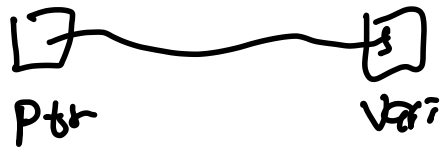
- Part 1: transform text by replacing sequences of vowels with `'`
- Part 2: classify tokens as integer, floating point, or non-numeric
- Part 3 (optional): determine letter frequencies of input text file
- Breakout rooms 1–10 are "social"
- Use Slack to let us know if you have a question

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

Day 26 recap questions

1. What is a C++ reference?
2. When should you use C++ references?
3. What is the difference between a pointer and a reference?
4. How do you dynamically allocate memory in C++?
5. How do you free memory in C++?

1. What is a C++ reference?



A reference is an alias (alternate name) for a variable or object.

In its lifetime, a reference can only refer to **one** variable or object.

Most common use: true reference parameters. E.g.:

```
void swap(int &a, int &b) {
  // ... a and b are aliases for the argument variables ...
}

// ...

int x = 2, y = 3;
swap(x, y); // the swap function can modify x and y
```

Const references

Another important use of references: `const` reference parameters. Very useful for passing a large object or collection to a function, since it avoids copying. E.g.:

```
// all elements from argument vector must be copied
// into a_vec, could be very slow
void myfunc1(vector<int> a_vec) {
  // ...
}

// a_vec is an alias for the argument vector,
// no copying required
void myfunc2(const vector<int> &a_vec) {
  // ...
}
```

2. When should you use C++ references?

Allowing a function to have an alias to an argument variable, so it can modify the argument variable.

Accepting a `const` reference to an object where copying would be slow.

Occasionally: capture a reference to a collection element so you can modify it. E.g.:

```
vector<int> myvec;

// ...
int &element = myvec[i];

element *= 2; // this modifies myvec[i]
```

vector<int> & vec = myvec;

3. What is the difference between a pointer and a reference?

Reference:

- Does not require explicit address-of (&) to create, or explicit dereference (*) to access the variable or object the reference refers to.
- Cannot be reassigned. (It can only refer to one variable or object during its lifetime.)

Pointer:

- Requires explicit address-of (&) to create, and explicit dereference (*) to access the variable the pointer points to.
- Can be reassigned. An assignment to a pointer variable changes what the pointer variable points to.

4. How do you dynamically allocate memory in C++?

`new` or `new[]`

Dynamically create one variable:

```
int *p = new int;
*p = 42;
```

Dynamically create an array:

```
int *p = new int[10];
for (int i = 0; i < 10; i++) {
  p[i] = i;
}
```

You should avoid using `malloc` in a C++ program.

in C world

→ malloc

→ calloc

5. How do you free memory in C++?

`delete` or `delete[]`

in C world
— free

Example:

```
int *p = new int;
*p = 42;
delete p;
```

Example:

```
int *p = new int[10];
for (int i = 0; i < 10; i++) { p[i] = i; }
delete[] p;
```

Exercise 26

- Given probability distribution for rolling weighted N-sided die, compute *cumulative distribution function* representing probability of rolling "i or less"
- "Naive" and "fast" functions to get an iterator positioned at last element in sorted vector less than or equal to v
  - Naive: use sequential search
  - Fast: use binary search
- Breakout rooms 1–10 are "social"
- Use Slack to let us know if you have a question!

Notes

Notes

Notes

Notes

Notes