Lab 2 - File I/O, Vectors, References

CS 251, Fall 2024

Copyright Notice

Learning Goals

By the end of this lab, you'll:

- Trace code that uses references in various ways
- Debug an algorithm that manipulates a vector
- Read from and write to files

Starter Code

lab02-starter.zip

Tasks

Starting this week, labs will include a mix of non-programming tasks (theory, code-reading, code-tracing) and programming tasks (debugging, code-writing). We have multiple kinds of tasks to help you practice for both exams and projects. You can work on the tasks in any order.

As usual, we strongly encourage collaboration in labs. Work with your peers, and ask your TAs questions!

(1) References by Request

Trace through the following code. What does it print? Why does each function do what it does? (Don't run it until you've made an attempt!)

```
void printVec(vector<int>& v) {
  for (int i : v) {
    cout << i << " ";
  }
```

```
  cout << endl;
}

void maui(vector<int> v) {
  for (size_t i = 0; i < v.size(); i++) {
    v[i] += 2;
  }
}

void sina(vector<int>& v) {
  for (size_t i = 0; i < v.size(); i++) {
    v[i] += 2;
  }
}

void moana(vector<int>& v) {
  for (auto i : v) {
    i += 2;
  }
}

void heihei(vector<int>& v) {
  for (auto& i : v) {
    i += 2;
  }
}

vector<int> teFiti(vector<int>& v) {
  vector<int> result;
  for (int i : v) {
    result.push_back(i + 2);
  }
  return result;
}
```

```
int main() {
  vector<int> v = {4, 2, 5};

  maui(v);
  printVec(v);

  sina(v);
  printVec(v);

  moana(v);
  printVec(v);

  heihei(v);
  printVec(v);

  teFiti(v);
  printVec(v);

  return 0;
}
```

(2) Duplicate Debugging

Consider the following function, which accepts a ***sorted*** `vector<string>` as input. It tries to modify it by removing duplicate elements. Since the input is sorted, we know duplicate elements are adjacent.

```
void deduplicate(vector<string> vec) {
  for (int i = 0; i < vec.size(); i++) {
    if (vec.at(i) == vec.at(i + 1)) {
      vec.erase(vec.begin() + i);
    }
  }
```

```
}
```

We'd like to be able to use the function like this:

```
vector<string> drinks = {
    "coffee",
    "coffee",
    "coffee",
    "hot chocolate",
    "tea",
    "tea",
};

deduplicate(drinks);
// drinks should now contain {"coffee", "hot chocolate", "tea"}
```

Unfortunately, this implementation does not work correctly. It has **3** bugs!

First, identify the bugs and explain what the problems are. Then, fix the code in `dedup.cpp` so that it works according to our expectations. You can use `make run_dedup` to execute and see the effect of your changes.

(3) Concat Files

For this task, you'll be implementing some functions from scratch in `concat_files.cpp`. We've given you automated tests.

- Implement the `readlines` function, which reads each line in the given file into a vector.
- Implement the `concatFiles` function, which takes two filenames. The first file contains a filename on each line. Concatenate all lines of each of these files in order into the output file.

Use the commands `make run_concat` to run your program's `main` (not required, but useful for manual testing), and `make test_concat` to run the test suite.

Deliverables

To record your attendance at the lab, please show a TA your group's answers and tests. You will be asked questions about your work!

TAs will discuss solutions in the last 15 minutes of the lab. Once you finish, you can work on the project until then. If you aren't interested, feel free to leave early once you've been checked off.

Acknowledgements

Content from Sean Szumlanski et. al, revised and curated by Ethan Ordentlich