Review Sheet 1b

CS 70: Data Structures and Program Development

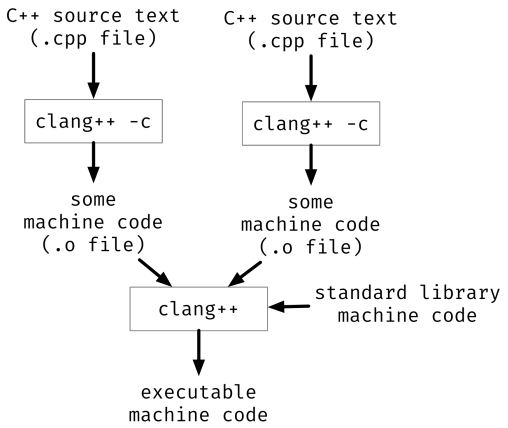
Thursday, January 23, 2020

Learning Targets

- I can explain the steps to compile multi-file C++ code.
- I can contrast the design goals of Java and C++.
- I can identify code with bad style.
- I can write readable and elegant C++ code.

Many Files, Separate Compiliation

1. Compiling C++ (multiple files, better!)



2. Suppose our program has three `.cpp` source files. To get a runnable program, how many times should `clang++` run?

   Suppose now we change one definition in one `.cpp` files. To get an updated runnable program, how many times should `clang++` run?

3. ```
   #include <iostream>
   #include <string>

   int main() {
     std::string message = "Hello, World!";
     std::cout << message << "\n";
    return 0;
   }
   ```

4. ```
   ----------------- exclaim.hpp ----------------
   #include <string>

   // Adds an !
   std::string exclaim(std::string sentence);

   ---------------------main.cpp-----------------
   #include <iostream>

   #include "exclaim.hpp"

   int main() {
     std::cout << exclaim("wow") << std::endl;
   }
   ```

   ```
   ------------------ exclaim.cpp ----------------
   #include "exclaim.hpp"
   #include <string>

   std::string exclaim(std::string sentence) {
     return sentence + "!";
   }
   ```

Programming Language Design Principles

1. How do C++ and Java differ in their goals?
2. C++ or Java?
   a. A `long` integer is a 64-bit number.
   b. A `long` integer has some number of bits (perhaps the same number as in a CPU register).
   c. If you try to access index 100 of a 10-element array of integers, an error (exception) will be reported.
   d. If you try to access index 100 of a 10-element array of integers, anything could happen (but you'll most likely get back some bits taken from memory past the end of the array, interpreted as an integer.)

Coding Style

1. What were the big ideas in the style readings?
2. CS 70 Naming Conventions
   - Variable names and functions: camelCase
     - `count`, `i`, `activeTask`, `launchMissiles()`
   - Data members (fields): camelCase + trailing underscore
     - `front_`, `currentCapacity_`
   - Class names: Capitalized CamelCase
     - `Gene`, `StudentTranscript`
   - Constants: All caps, underscore between words
     - `VERSION`, `MAX_STUDENTS`
3. Idiomatic Loops

   ```
   const size_t NUM_LETTERS = 26
   std::string alphabet;
   for (size_t i = 0; i < NUM_LETTERS; ++i) {
       alphabet += ('a' + i)
   }

   for (size_t i = 0; i < alphabet.size(); ++i) {
       std::cout << alphabet[i] << " " << i << "\n";
   }
   ```

1. Consider this code example.

   - What, specifically, is wrong with it?
   - Why, specifically, is this a problem?
   - How would you suggest fixing the problem(s)?

```
// MUST be set to 1!
Params.ParentalLevel = 3; // QA now insists this be 2
```

2. Consider this code example.

   - What, specifically, is wrong with it?
   - Why, specifically, is this a problem?
   - How would you suggest fixing the problem(s)?

```
// if ((typec!="20") && (typec!="13") && (typec!="5") && (typec!="4"))
if (typec !="20") {
    if (typec != "13") {
        if (typec != "5") {
            if (typec != "4") {
                selectType("ALLOC");
                return;
            }
        }
    }
}
```

3. Use this space to list some ways in which code can exhibit bad style, and explain exactly how each can cause trouble.