ECE 3574: Design Patterns and Idioms

Changwoo Min
Design Patterns and Idioms

• Today we will discuss the use of design patterns and common idioms used to write canonical C++ code.

• Common C++ idioms
  • Example: RAII
  • Example: Copy/Swap
  • Example: COW

• Design Patterns
  • Example: Iterator
Common C++ Idioms

- All programming languages are equivalent in the sense that they are Turing complete.
- However, programming languages (or more properly the community of programmers) develop *idioms*, common ways of expressing ideas that leverages the semantics of that language.
Common C++ Idioms

- Simple example in C++: removing excess storage from a container, (e.g. a `std::vector`)

  ```cpp
  std::vector<int> c;
  // ... Lots of push_backs and then lots of remove on c.
  // ...
  // Prior to C++11
  std::vector<int>(c).swap(c);
  //With C++11 (technically it is still a “non-binding request”)
  c.shrink_to_fit();
  ```

- Reference: `std::vector::swap`, Clear-and-minimize
Another simple C++ idiom: erase-remove

• What does the following print?

```cpp
std::list<int> mylist;
mylist.push_back(0);
mylist.push_back(12);
mylist.push_back(31);
std::cout << mylist.size() << std::endl;
std::remove(mylist.begin(), mylist.end(), 12);
std::cout << mylist.size() << std::endl;
```

• Remove actually does not actually remove!
Another simple C++ idiom: erase-remove

- To really remove you use the “erase-remove” idiom.

```
mylist.erase(std::remove(mylist.begin(), mylist.end(), 12), mylist.end());
```

- See example code: `shrink.cpp`
Another simple C++ idiom: erase-remove

- Erase–remove idiom
- std::remove
- std::vector::erase
Example: RAII

• RAII stands for Resource Acquisition Is Initialization.
  • Resource acquisition is initialization
  • RAII
• See example code: raii.cpp
Example: Copy/Swap

- We can remove the code duplication and the self assignment test in the copy-assignment operator using the copy-swap idiom.
  - Copy-and-Swap Idiom in C++
  - Copy-and-swap
- See example code: copyswap.cpp
Example: Move semantics in C++11

- C++11 defines *move semantics* that add to RAII and the copy-swap idiom
  - Rvalue reference
  - Move semantics
  - std::move
- See example code: `copyswap11.cpp`
A big difference between most `std::string` implementations and `QString` is the latter uses COW.

COW is an optimization that lets objects share the same data as long as neither tries to change it, at which time a copy is made.

Note: Matlab uses this for Matrices.

COW has problems with concurrency, as we will see in a couple of weeks.

See example code: `cow.cpp`
Design Patterns

• Design patterns are similar to Idioms but are less language specific.
• They are patterns in the sense of higher-order abstractions of code design.
• See the book *Design Patterns: Elements of Reusable Object-Oriented Software*
• There are many online compendium of patterns.
Example Design Pattern: PIMPL: Pointer-to-Implementation

- Pimpl decouples the definition and implementation of a class stronger than via private and public.
  - Pointer To Implementation
  - PImpl
- Can be usefull for abstracting platform differences without headers full of macros.
- Qt uses the Pimpl pattern extensively.
- See example code: pimpl/*
Example Design Pattern: Iterators

- Iterators are used throughout the standard library for accessing and manipulating containers.
- They are an abstraction of pointers.
- See example code: iterators/*
Criticisms of Design Patterns

• To some extent the patterns are ways of expressing things not naturally found in the language.

• Some people consider this a limitation of the programming language in question.

• It is easy to go overboard. Some patterns are overused (in my opinion), Singleton for example.
Next Actions and Reminders

• Read about the Factory and Model-View Pattern
• Start Project 3
  1. Before staring Project 3, you should first finish Project 2.
  2. Read the Project 3 specification and test the `sldraw` in your repository.
  3. Integrate your Project 2 source code to Project 3
  4. Extend your `slisp` for Project 3
  5. Add the initial GUI, which just have the same layout with the spec.
  6. Make your GUI can call your interpreter. :-(