ECE 3574: Applied Software Design

Design Patterns and Idioms

Spring 2017

Some Announcements

- My office hours for 2/20 are cancelled.
- There is an error in the initial repository for project 2. You need to edit the ‘`scripts/coverage.sh` file and change `./test_main_window` to `./test_gui`. Commit this change to your repository.
- There is an anonymous survey on Canvas (under Quizzes) about your perceptions of the course. Please provide constructive feedback.

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

Another simple C++ idiom: erase-remove

What does the following print?

```
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! To *really* remove you use the "erase-remove" idiom.

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

Example: RAII

RAII stands for Resource Acquisition Is Initialization.
See example code

Example: Copy/Swap

We can remove the code duplication and the self assignment test in the copy-assignment operator using the copy-swap idiom.
See example code

Example: Move semantics in C++11

C++11 defines *move semantics* that add to RAII and the copy-swap idiom

See example code

Example: Copy-on-Write (COW)

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

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.
Can be usefull for abstracting platform differences without headers full of macros.
Qt uses the Pimpl pattern extensively.
See example code.

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.

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