Habits for C++ Code Quality

Pham Anh Tuan

FT1 Team

Content

- What does this mean?
- C++
- General

What?

- It (software) is harder than anything else I’ve ever had to do. -- Donald Knuth --

- 2 main purposes:
  - Easier to read ⟶ code is for human first
  - Avoid potential unexpected things ⟶ sleep well

C++ Proficiency

Contribute

Invent

Quality

Efficient

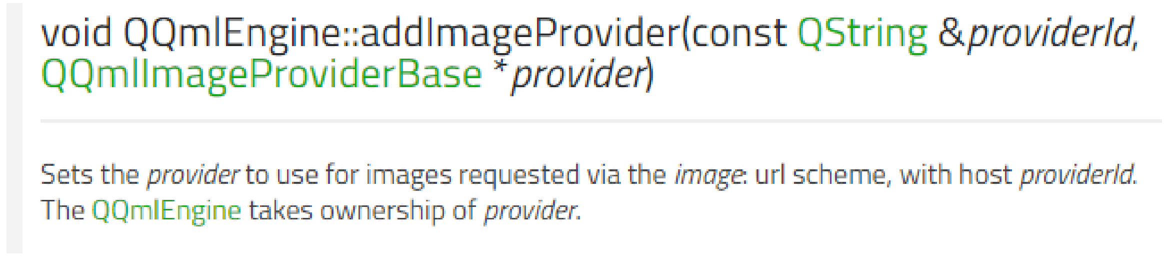
Right

Header file

- Avoid to include the same header file
- Using namespace
- Define name space
- Forward declaration

Pointer and Reference

- If you use pointer⟶ Know well ownership and life-cycle
  - Who is responsible
  - Set virtual destructor



void QQmlEngine::addImageProvider(const QString &*providerId*, QQmlImageProviderBase **provider*)

Sets the *provider* to use for images requested via the *image*: url scheme, with host *providerId*. The QQmlEngine takes ownership of *provider*.

- Use a const reference (avoid overhead):
  - Parameters of a function, return of a function
  - Left value

Resource acquisition is initialization (RAII)

C++ idioms

- A resource is tied to object lifetime
- Common case: Scope-based Resource Management
- Example:
  - shared_ptr, unique_ptr, etc.
  - lock_guard
  - QFile
  - Any other things require Open/Close, Acquire/Release

STL

- vector: operator [] or at() function
- Map: operator[] or value
- If you have to use it
  → Assert index value (like at() of std::vector)

Take advantage of STL

- Many are available:
  - Basic ones: max, min, swap, etc.
  - How many students got A → count_if
  - Show albums contain a song → find
- Names are meaningful
- Why use standard containers but ignore standard algorithms?

Utilize enum class

- enum is not int
- Prefer enum class

```
Traffic_light& operator++(Traffic_light& t)
// prefix increment: ++
{
    switch (t) {
    case Traffic_light::green:
        return t=Traffic_light::yellow;
    case Traffic_light::yellow:
        return t=Traffic_light::red;
    case Traffic_light::red:
        return t=Traffic_light::green;
    }
}
Traffic_light next = ++light;
// next becomes Traffic_light::green
```

→ encapsulation

const

- const bool isVisible();
- bool isVisible() const;

Warnings

- Warnings are not errors → but could become errors

Do not use C/Java style

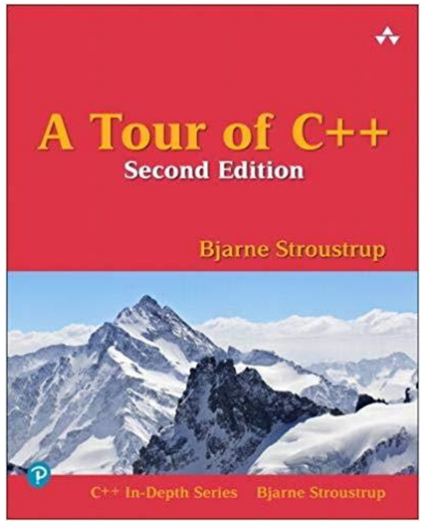
- char* $\longrightarrow$ std::string
- printf $\longrightarrow$ std::cout
- NULL or 0 $\longrightarrow$ nullptr
- Casting: const_cast, static_cast, dynamic_cast

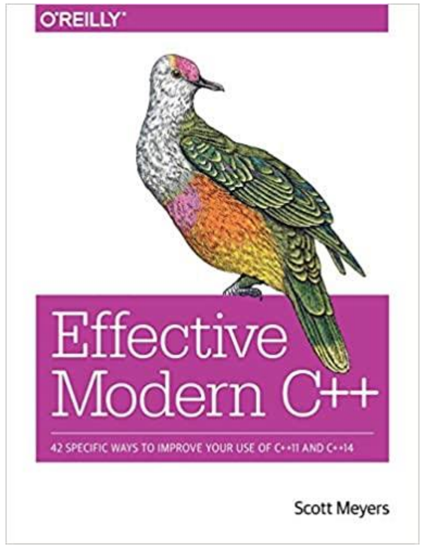
General rules

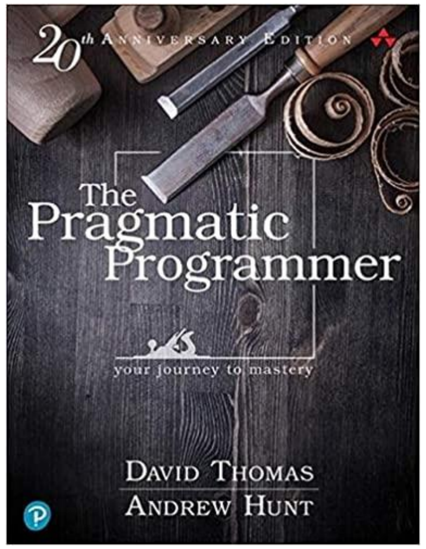
- Avoid code duplication
- Avoid create “redundant” new variables
- Line of code in a function
- Revise regularly → Things can always be improved
- TEST MORE!

Resources

- A Tour of C++
- Effective modern C++
- The Pragmatic Programmer







Questions