C++ Core Guidelines

Quick 10 minutes talk

By Shalom Kramer @kramerpeace

I.8: Prefer Ensures() for expressing postconditions

- *assert()* - but more expressive
- Postconditions of the form "this resource must be released" are best expressed by RAII
- Will morph into the not-yet-formulated Contract Design

```
void push(queue &q)
    [[expects: !q.full()]]
    [[Ensures: !q.empty()]]
{
    ...
    [[assert: q->is_ok()]]
}
```

RAII

- Resource Acquisition Is Initiazation

```
class Port {
public:
    Port();
    void init(string p);
    void write();
    void close();
};

void write() {
    Port p;
    p.init("COM3");
    p.write();
    p.close();
}
```

```
class Port {
public:
    Port(string p) { init(p); }
    ~Port() { close(); }
    void init(string p);
    void write();
    void close();
};

void write() {
    Port p("COM3");
    p.write();
}
```

C.31: All resources acquired by a class must be released by the class's destructor

- Not just dynamic memory, all resources!
- Prevent leaked resources
- RAII conformity
- What happens if a resource refuses to close? - no real solution

C.35: A base class destructor should be either public and virtual, or protected and nonvirtual

- Prevent destructing only the base class without calling the derived class destructor
- This is **really** important with RAII

I.3: Avoid singletons

- Reason: Singletons are basically complicated global objects in disguise.
- It’s hard to destroy a singleton.
- - WeakSingleton
- What happens when singleton A uses singleton B?

CP.110: Do not write your own double-checked locking for initialization

```
static std::once_flag
my_once_flag;
    std::call_once(my_once_flag, []()
    {
        // do this only once
    });
// Assuming the compiler is compliant with C++11
static My_class my_object; // Constructor called only once
```

CP.111: Use a conventional pattern if you really need double-checked locking

```
mutex action_mutex;
atomic<bool> action_needed = true;

if (action_needed) {
    std::lock_guard<std::mutex> lock(action_mutex);
    if (action_needed) {
        take_action();
        action_needed = false;
    }
}
```

Questions?