6/3/2015

Ressource Acquisition Is Initialization (RAII)

Sebastian J. Wirkert

dkfz.
DEUTSCHES KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

Willkommen im DKFZ!

dkfz.
DEUTSCHES
KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

50 Jahre – Forschen für
ein Leben ohne Krebs

Philosophical questions

- How do we manage a file?
- How do we manage lifetime of an object?

Examples


```
File f;
f.open("boo.txt");
…
loadFromFile(f);
f.close();

Dog* dog = new Daschund();
…
goToThePark(dog);
delete dog;

Lock* lock = getLock();
lock.aquire();
…
doSomething();
lock.release();
```

Solution RAII – Example File Handling


```
#include <string>
#include <cstdio>

class Datei {
    FILE* datei_;

public:
    Datei(const std::string& name)
    : datei_( std::fopen(name.c_str(), "w+") ) {} // Öffnen der Datei

    ~Datei() {
         std::fclose(datei_); // Schließen der Datei
    }

    void Ausgabe(const std::string& text) {
        if (datei_)
            std::fputs(text.c_str(), datei_);
    }
};

int main() {
    Datei datei("aufzeichnung.txt"); // Öffnen der Datei (Anfordern der Ressource)
    datei.Ausgabe("Hallo Welt!");

    // Mit dem Ende der Funktion endet auch der Gültigkeitsbereich (Scope)
    // des Objekts datei. Daher wird der Destruktor Datei::~Datei()
    // aufgerufen, der die Datei schließt → Freigabe der Ressource.
}
```

Solution RAII – Example new

nicht RAII:

```
void func(void)
{
    char* buffer = new char[3];
    buffer[0] = 'A';
    buffer[1] = 'B';
    buffer[2] = 'C';

    // buffer verwenden

    delete[] buffer;
}
```

RAII:

```
void func(void)
{
    std::vector<char> buffer = {'A', 'B', 'C'};

    // buffer verwenden
}
```

Solution Locks:


```
void write_to_file (const std::string & message) {
    // mutex to protect file access
    static std::mutex mutex;

    // lock mutex before accessing file
    std::lock_guard<std::mutex> lock(mutex);

    // try to open file
    std::ofstream file("example.txt");
    if (!file.is_open())
        throw std::runtime_error("unable to open file");

    // write message to file
    file << message << std::endl;

    // file will be closed 1st when leaving scope (regardless of exception)
    // mutex will be unlocked 2nd (from lock destructor) when leaving
    // scope (regardless of exception)
}
```

RAII

In RAII, holding a resource is tied to object lifetime: resource allocation (acquisition) is done during object creation (specifically initialization), by the constructor, while resource deallocation (release) is done during object destruction, by the destructor. If objects are destructed properly, resource leaks do not occur.

Benefits

- Encapsulation
- Exception safety
- Locality

Examples from the standard

- Smartpointer
- Container

Auf Wiedersehen im DKFZ!

dkfz.
DEUTSCHES KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

50 Jahre – Forschen für ein Leben ohne Krebs