Part I. RAII and Memory Management

RAII stands for Resource Acquisition Is Initialization. The idea behind this idiom: for any resource acquired, an object should be initialized that will own that resource and close it in the destructor. Smart pointers are a prominent example of RAII. They help avoid memory leaks. The following libraries provide smart pointers and other tools to help you manage memory more easily.

- Boost.SmartPointers defines smart pointers. Some of them are provided by the C++11 standard library. Others are only available in Boost.
- Boost.PointerContainer defines containers to store dynamically allocated objects – objects that are created with `new`. Because the containers from this library destroy objects with `delete` in the destructor, no smart pointers need to be used.
- Boost.ScopeExit makes it possible to use the RAII idiom for any resources. While Boost.SmartPointers and Boost.PointerContainer can only be used with pointers to dynamically allocated objects, with Boost.ScopeExit no resource-specific classes need to be used.
- Boost.Pool has nothing to do with RAII, but it has a lot to do with memory management. This library defines numerous classes to provide memory to your program faster.

Table of Contents