Smart Pointers in C++
#include <memory>

shared_ptr and unique_ptr

smart pointers - automatically (in most cases) will deallocate the object that they point at when that object can no longer be referenced

Krishna Kumar

shared_ptr

- If you are concerned about freeing of resource/memory AND if you have more than one function that could be using the object AT DIFFERENT times, then go with shared_ptr.
- Allows for multiple pointers to point at a given resource.
- When the very last shared_ptr to a resource is destroyed, the resource will be deallocated.

```
// create an instance of a registered class
std::shared_ptr<Course> Create(std::string coursename);
```

shared_ptr (cont...)

- shared_ptr<T> myPtr(new T);
  - // Okay
- shared_ptr<T> myOtherPtr = myPtr;

  // Sure! Now have two pointers to the resource.
  - It is both copyable and movable

auto_ptr

- Deprecated - C++11 (Avoid at all costs)
- unique_ptr is a new facility with a similar functionality, but with improved security (no fake copy assignments), added features (deleters) and support for arrays.

unique_ptr

- If all you are concerned is freeing memory, and the access to object is SEQUENTIAL, then go for unique_ptr.
- By SEQUENTIAL, I mean, at any point object will be accessed from one context.
- is a smart pointer which owns an object exclusively.
- there can be at most one unique_ptr pointing at any one resource
- When that unique_ptr is destroyed, the resource is automatically reclaimed

std::unique_ptr (cont...)

Kind of assignments supported by unqiue_ptr

- move assignment
- assign null pointer
- type-cast assignment

unique_ptr (cont...)

- unique_ptr<T> myPtr(new T);   // Okay
- unique_ptr<T> myOtherPtr = myPtr;

  // Error: Can't copy unique_ptr
- unique_ptr<T> myOtherPtr = std::move(myPtr);

  // Okay, resource now stored in myOtherPtr

References

- https://stackoverflow.com/questions/6876751/differences-between-unique-ptr-and-shared-ptr
- https://stackoverflow.com/questions/3697686/what-is-the-problem-with-auto-ptr
- http://www.cplusplus.com/reference/memory/
- http://www.careerride.com/C++-what-are-shallow-and-deep-copy.aspx