**Raw pointers drawbacks**

* Declarations doesn’t indicate whether it points to a single object or to an array
* There is no way to tell whether use delete or different destruction mechanism to destroy a pointer points to
* Not possible to know whether to use the single-object (delete) form or the array (delete[]) form to delete
* Dangling pointers arise when objects are destroyed while pointers still point to them

Smart pointers are one way to address these issues.

Smart pointers are wrappers around raw pointers that act much like the raw pointers they wrap

Four smart pointer in C++11

auto\_ptr, unique\_ptr, shared\_ptr, weak\_ptr

**auto\_ptr** is deprecated leftover from C++98. It was replaced with unique\_ptr

**unique\_ptr**:

* exclusive-ownership resource management
* ownership transfer from source to destination pointer. Source ptr set to null
* copying not allowed
* its move-only type

Used as a factory function return type for objects in a hierarchy

[Program: factoryMethod\_uniquePtr.cpp]