Smart pointers

- For pointer-based languages, smart pointers are meant to bridge the gap between pointers and references
- Provide safeguards against wild pointers, invalid pointers, dangling pointers, and memory leaks
- Usually implemented as a class (where supported), which provides methods or operators for safe/controlled access to the dynamically allocated item

Smart pointer use

- User declares a smart pointer of desired resource type
- User requests new instance of resource through the smart pointer
- User requests access to resource (methods/fields) through the smart pointer
- User can copy smart pointers (e.g. ptrA = ptrB)
- Resource de-allocation automatically handled by smart pointer

Smart pointer data content

- Smart pointer maintains internal pointer to the actual allocated resource, user never gets direct access
- Smart pointer also maintains internal pointer to an administrative object that tracks/controls access to the allocated resource
- The admin object is allocated when the resource is first allocated, and maintains a reference count (how many smart pointers have access to the resource instance)

Smart pointer operations

- The copy operation (e.g. ptrA = ptrB) for smart pointers copies both internal pointers (to the resource and the admin object) after decrementing the reference count for the resource ptrA used to point to, and incrementing the reference count for the resource ptrB points to
- If smart pointer variable goes out of scope the relevant reference count is also decremented
- If reference count ever hits 0 then the resource instance is deleted, internal pointer to it nullified
- Access operations (e.g. -> and *) are overloaded to check with the admin object to make sure the resource instance still exists before redirecting to the requested resource field/method

Safeguards provided

- Programmer doesn’t create any wild/invalid pointers because all “dumb” pointer operations handled internally by the smart pointer class
- Dangling pointers not an issue since resource kept alive as long as anything still refers to it
- Memory leaks not an issue since resource automatically deallocated as soon as last reference ends