

Shared Pointer Solutions

- Describe the C++ `shared_ptr`. How does it differ from `unique_ptr`?
 - `shared_ptr` is a “smart pointer” class which has a traditional pointer as a private member
 - Multiple instances can share the same memory allocation
 - Only one instance of `unique_ptr` can access the internal memory allocation at any one time
 - The memory allocation is automatically released when the last instance which uses it is destroyed

- What is a reference counter? How is it used in the C++ `shared_ptr` implementation?
 - A reference counter keeps track of the number of objects which refer to a given memory object
 - In `shared_ptr`, it counts the number of instances which have access to the memory allocation
 - The reference counter is incremented when an instance is copied, and decremented when a copy is destroyed
 - When the last copy is destroyed, the reference counter is equal to zero. The memory allocation is then deleted

- Describe what happens when
 - A new `shared_ptr` instance is created
 - Memory is allocated for the pointer member and for the control block containing the reference counter
 - If the `shared_ptr` is created using `make_shared()`, these are done in a single allocation
 - If the `shared_ptr` is created using `new`, these are done as two separate allocations
 - The reference counter is set to 1, indicating that one instance has access to the internal pointer member

- Describe what happens when
 - A shared_ptr instance is created as a copy of another instance
 - The new instance will have a pointer member and control block which have the same address as those in the old instance
 - The reference counter in the control block will be incremented

- Describe what happens when
 - A shared_ptr instance is assigned to another instance
 - If the two instances refer to the same memory allocation, the reference counter will be incremented
 - If they refer to different memory allocations, the assigned-to object will decrement its reference counter. A copy is then performed

- Give an example of a programming problem where a shared pointer could be useful
 - In applications which have many copies of the same data, save memory by having multiple references to a single copy
- Give a disadvantage of using shared pointers
 - Memory overhead (storage for control block)
 - Copying overhead (reference counter is incremented atomically to avoid “data races” in multi-threaded programs)
 - Can create circular dependencies which cause memory leaks

- Write a simple program which creates an instance of an initialized shared pointer and prints out its data
- Alter your program so that it creates another instance which is a copy of the first instance
- Alter your program so that it creates an uninitialized instance and assigns the first instance to it