Intermediate Level C++

C++ Smart Pointers

Yong Zhang

Objectives

In this chapter you will learn:

- Pointers in C++
- unique_ptr
- shared_ptr
- Guidelines of Using Smart Pointers

Pointers in C++

- **Old Techniques: raw or naked pointer**
  - T* new, delete
  - auto_ptr
- **New Techniques: smart pointers**
  - unique_ptr
  - shared_ptr
  - weak_ptr

unique_ptr

- Exclusively owns object to which it points
- Can't be copied
- Can be moved
- Should use make_unique instead of new
- Demo: UniquePointers

shared_ptr

- Shares ownership of object to which it points
- Non-invasive reference counting
- Can be copied
- Can be moved
- Should use make_shared instead of new
- Demo: SharedPointer

weak_ptr

- **share_ptr has two pointers**
  - One points to the managed object
  - The other points to the use count
- **Circular reference (reference loop):**
  - If two share_ptr objects refer to each other they will never get deleted
- **weak_ptr**
  - points to a shared_ptr (the object a shared_ptr points to)
  - does not increase its use count.
- **Demo: WeakPointerDemo**

Guidelines of Using Smart Pointers

- **Always use smart pointer instead of raw pointers, if possible.**
- **If you return an object whose lifetime is to be managed by the caller, return unique_ptr.**
- **If you know the object that maintains the resource dies before the lifetime of the resource, use shared_ptr to handle pointed-to resources.**
- **weak_ptr is rarely needed.**