1. std::unique\_ptr

使用方法：

std::unique\_ptr<int> ptr(new int(10)); // 自动删除的智能指针

特点：

1.保证其指向的对象在其生命周期内是独占的，即同一时间只有一个 std::unique\_ptr 可以指向某个对象。

2.拥有右值引用构造函数，可以方便地进行移动语义。

3.不能被复制，但可以被移动。

4.提供了 release 方法来释放其拥有的对象。

5.是最好的智能指针类型，适用于需要独占对象的场景。

1. std::shared\_ptr

使用方法：

std::shared\_ptr<int> ptr1 = std::make\_shared<int>(10); // 引用计数的智能指针

std::shared\_ptr<int> ptr2 = ptr1; // ptr1 和 ptr2 现在都指向同一个对象

特点：

1.通过引用计数来管理其所指向对象的生命周期。当最后一个指向对象的 std::shared\_ptr 被销毁或重置时，对象将被删除。

2.可以被复制和移动，通过引用计数确保多个指针共享同一个对象的生命周期。

3.提供了 use\_count 方法来查看引用计数。

4.适用于多个指针可能共享同一个对象的情况，例如，当需要将对象传递给多个函数或保存到容器中时。

1. std::weak\_ptr

使用方法：

std::shared\_ptr<int> sptr(new int(10)); // 创建共享指针

std::weak\_ptr<int> wptr = sptr; // 创建弱指针，它不增加引用计数

特点：

1.不会增加其所指向对象的引用计数。当对象被删除时，std::weak\_ptr 将变成“空”状态（可以再次获取到对象的指针）。

2.不能阻止其所指向的对象被删除。

3.可以与 std::shared\_ptr 配合使用，例如在观察者模式中，当多个观察者可能观察同一个对象时。它提供了对对象的非独占访问，而不会增加引用计数。

4.适用于需要观察对象而不影响其生命周期的场景。