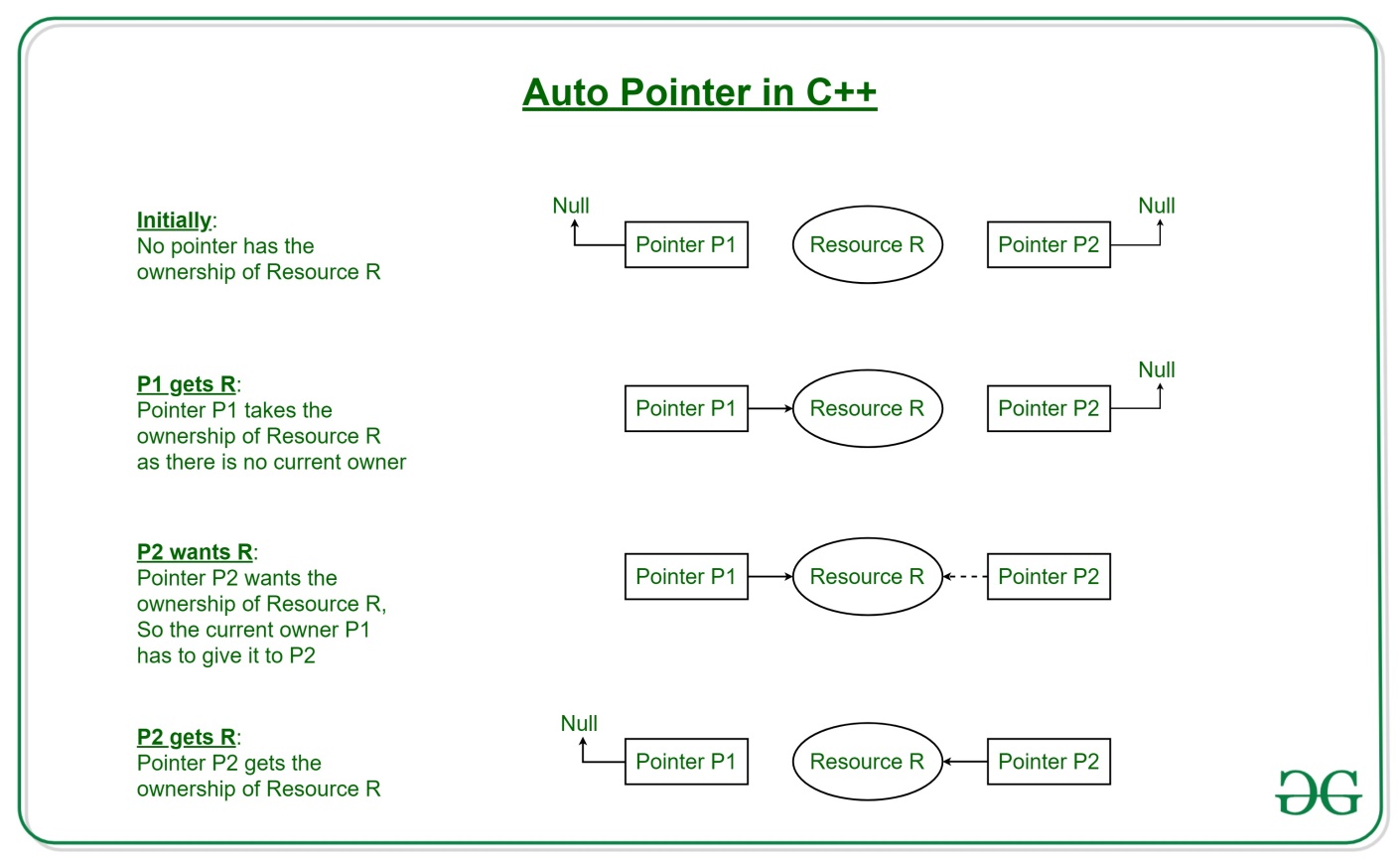
**auto\_ptr**

**auto\_ptr** - unikal\_ptr ilə oxşar funksionallığı olan, lakin təkmilləşdirilmiş təhlükəsizliyi olan yeni bir vasitədir. auto\_ptr özü silindikdə həmin obyekti silən , əldə edilmiş obyekti idarə edən smart pointerdir. Obyekt auto\_ptr sinifindən istifadə edərək təsvir edildikdə, o, əhatə dairəsindən kənara çıxanda işarə etdiyi obyektin avtomatik silinməsini təmin edən vahid ayrılmış obyektə göstərici saxlayır. O, eksklüziv sahiblik modelinə əsaslanır, yəni eyni tipli iki göstərici eyni vaxtda eyni mənbəyə işarə edə bilməz. Aşağıdakı proqramda göstərildiyi kimi, göstəricilərin surətinin çıxarılması və ya təyin edilməsi mülkiyyət hüququnu dəyişir, yəni mənbə göstəricisi təyinat göstəricisinə sahiblik verməlidir.

copy constructor və assignment operator əslində saxlanılan göstəricini kopyalamır, əvəzində ilk auto\_ptr obyektini boş qoyurlar.



Bu, istənilən vaxt yalnız bir auto\_ptr obyektinin göstəriciyə sahib ola bilməsi üçün eksklüziv sahiblik modelinəinin həyata keçirməyin bir yolu idi ki, , yəni copy semantikasının lazım olduğu yerlərdə auto\_ptr istifadə edilməməlidir.

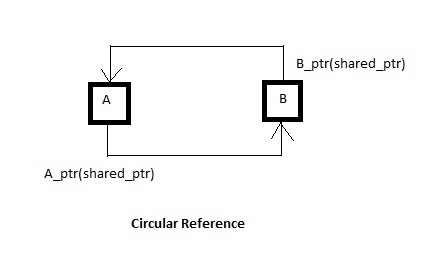
**auto\_ptr niyə köhnəlmişdir?**

O, göstəriciyə elə sahiblik edir ki, heç bir iki göstərici eyni obyekti göstərməsin. Sahibliyi köçürür və avtomatik dəyər göstəricisini boş göstəriciyə sıfırlayır. Beləliklə, yuxarıda qeyd olunan copy mümkün olmadığı üçün STL konteynerlərində istifadə edilə bilməz.

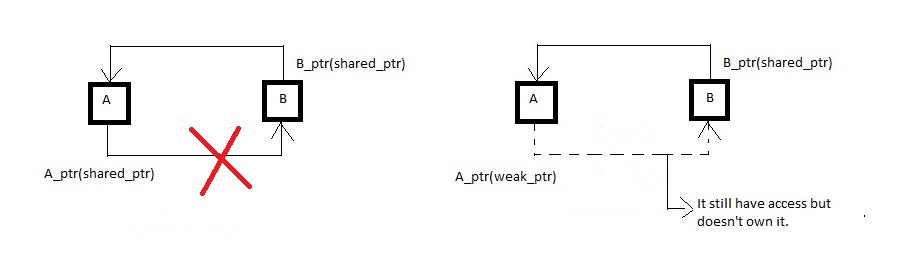
**weak\_ptr**

**weak\_ptr**  , shared\_ptr -nin surəti kimi yaradılır. O, bir və ya daha çox shared\_ptr-ə məxsus olan, lakin göstəricilərin hesablanmasında iştirak etməyən obyektə girişi təmin edir. weak\_ptr -nin mövcudluğu və ya silinməsi shared\_ptr və ya onun digər nüsxələrinə heç bir təsiri yoxdur. Bəzi hallarda shared\_ptr nümunələri arasında dairəvi istinadları pozmaq tələb olunur.

Cyclic Dependency (Tsiklik Asılılıq) ( shared\_ptr ilə bağlı problemlər): Gəlin bir ssenarini nəzərdən keçirək ki, bizdə iki sinif A və B var, hər ikisinin də digər siniflərə göstəriciləri var. Beləliklə, həmişə A-nın B-yə, B isə A-ya işarə edir. Beləliklə, use\_count heç vaxt sıfıra çatmayacaq və heç vaxt silinməyəcək.



weak\_ptr - lərdən stifadə etməmizin səbəbi budur, çünki onlar istinad sayılmır. Beləliklə weak\_ptr –də mülkiyyət paylaşılmır, lakin bu obyektlərə giriş əldə edə bilərlər.



Beləliklə, dövrü asılılıq səbəbindən share\_ptr-in use\_count-u heç vaxt sıfıra çatmır, bu da weak\_ptr istifadə edərək qarşısı alınır, bu problem A\_ptr-ni weak\_ptr elan etməklə bu problemi aradan qaldırır, beləliklə, A sinfi ona sahib deyil, yalnız ona daxil ola bilər və biz də obyektin etibarlılığını yoxlamalıyıq, çünki əhatə dairəsindən kənara çıxa bilər. Ümumiyyətlə, bu, dizayn problemidir.

weak\_ptr nə vaxt istifadə edilməlidir?

Obyektinizə bir neçə yerdən müraciət etmək istədiyiniz zaman – diqqətdən kənarda qalmağın və yerinin dəyişdirilməsinin düzgün olmadığı istinadlar üçün.