**2023 12 06**

**std::weak\_ptr**

#include <memory>

#include <string>

**using** **namespace** std**;**

**int** main**()**

**{**

***/\****

***shared\_ptr ile oluşturduğumız weak\_ptr ile shared\_ptr'in kaynağının***

***sonlanıp sonlamadığıa bakabiliyoruz.***

***\*/***

shared\_ptr**<**string**>** sptr**(**mew string**{**"tamer dundar"**});**

cout **<<** "sptr.use\_count() = " **<<** sptr**.**use\_count**()** **<<**"\n"**;**

weak\_ptr wp **=** sptr**;**

***// sptr.use\_count() sabit kalır***

cout **<<** "sptr.use\_count() = " **<<** sptr**.**use\_count**()** **<<** "\n"**;**

weak\_ptr wp1 **=** sptr**;**

cout **<<** "wp.use\_count() = " **<<** wp**.**use\_count**()** **<<**"\n"**;**

cout **<<** "wp1.use\_count() = " **<<** wp1**.**use\_count**()** **<<**"\n"**;**

**auto** wp3 **=** wp2**;**

cout **<<** "wp3.use\_count() = " **<<** wp3**.**use\_count**()** **<<**"\n"**;**

***// hepsi 1 çıkar.***

**}**

***// weak\_ptr --expired and lock***

**int** main**()**

**{**

shared\_ptr**<**string**>** sptr**(**mew string**{**"tamer dundar"**});**

weak\_ptr wp **=** sptr**;**

***// true dönerse kaynak sonlamış, false verirse kaynak halen hayatta***

wp**.**expired**()**

***// kaynak hayattaysa shared\_ptr döndürür, hayatta değilse nullptr döndürür***

**auto** spx **=** wp**.**lock**()**

***// cpp 98s***

**if** **(**shared\_ptr**<**string**>** spx1 **=** wp**.**lock**();** spx **!=** **nullptr)**

**{**

**}**

shared\_ptr sp**(**wp**);** ***// kaynak sonlamışssa bad\_weak\_ptr throw eder***

**}**

**struct** A

**{**

std**::**shared\_ptr**<**B**>** bptr**;**

A**()**

**{**

std**::**cout **<<** "A default ctor this = " **<<** **this** **<<** "\n"**;**

**}**

**~**A**()**

**{**

std**::**cout **<<** "A default ctor this = " **<<** **this** **<<** "\n"**;**

**}**

**}:**

**struct** B

**{**

std**::**shared\_ptr**<**A**>** bptr**;**

B**()**

**{**

std**::**cout **<<** "B default ctor this = " **<<** **this** **<<** "\n"**;**

**}**

**~**B**()**

**{**

std**::**cout **<<** "B default ctor this = " **<<** **this** **<<** "\n"**;**

**}**

**}:**

**int** main**()**

**{**

**using** **namespace** std**;**

shared\_ptr**<**A**>** spa**(new** A**);**

shared\_ptr**<**B**>** spa**(new** B**);**

***// dtor çağrılmaz***

spa**->**bptr **=** spb**;**

spb**->**aptr **=** spa**;**

**}**

**CRTP (Curiously Recurring Template Pattern)**

***// CRTP (Curiously Recurring Template Pattern)***

**using** **namespace** std**;**

***//CRTP Base***

**template** **<typename** T**>**

**class** Base

**{**

***// taban sınıf türemiş sınıfa interface sağlar ve taban sınıf bu interfacete***

***// türemiş sınıfın fonksiyonlarını kullanabilir.***

**void** func**()**

**{**

**static\_cast<**Der**\*>(this).**foo**();**

**}**

**};**

**class** Der **:** **public** Base**<**Der**>**

**{**

**};**

**class** Nec**{};**

**int** main**()**

**{**

Nec **\***ptr **=** **new** Nec**;**

***// tanımsız davranış dangling pointer oluşturur***

shared\_ptr**<**Nec**>** sp1**(**ptr**);**

shared\_ptr**<**Nec**>** sp2**(**ptr**);**

cout **<<** sp1**.**use\_count**()** **<<** '\n'**;** ***// 1***

cout **<<** sp2**.**use\_count**()** **<<** '\n'**;** ***// 1***

***// tanımsız davranış yok***

shared\_ptr**<**Nec**>** sp3**(**sp2**);**

cout **<<** sp3**.**use\_count**()** **<<** '\n'**;** ***// 2***

**}**

Eğer bir sınıfın üye fonksiyonu içinde shared\_ptr ile hayatı kontrol edilien \*this nesnesini gösteren shared\_ptr'nin kopyasını çıkartmak isterseniz sınıfınızı CRTP örüntüsi ile kaltım yoluyla std::enable\_shared\_from\_this sınıfından elde etmelisiniz.

**class** Nec **:** **public** std**::**enable\_shared\_from\_this**<**Nec**>**

**{**

**public:**

**void** func**()**

**{**

***/\****

***reference count 2 olmaz hala 1 olur. tanımsız davranıştır***

***\*/***

shared\_ptr**<**Nec**>** spx**(this);**

cout **<<** spx**.**use\_count**()** **<<** "\n"**;**

**}**

**void** bar**()**

**{**

**auto** spx **=** shared\_from\_this**();**

cout **<<** "spx.use\_count() = " **<<** spx**.**use\_count**()** **<<** "\n"**;** ***// 2***

**}**

**};**

**int** main**()**

**{**

shared\_ptr**<**Nec**>** sptr**(new** Nec**);**

cout **<<** "sptr.use\_count() = " **<<** sptr**.**use\_count**()** **<<** "\n"**;** ***// 1***

sptr**->**func**();**

cout **<<** "sptr.use\_count() = " **<<** sptr**.**use\_count**()** **<<** "\n"**;** ***// 1***

**auto** p **=** **new** Nec**;**

**try{**

p**->**bar**();** ***// exception throw eder***

**}**

**catch(const** std**::**bad\_weak\_ptr**&** ex**)** **{**

std**::**cout **<<** "exception caught: " **<<** ex**.**what**()** **<<** "\n"**;**

**}**

**}**

***// container'da shared\_ptr tutma***

#include <list>

**using** **namespace** std**;**

**int** main**()**

**{**

list**<**shared\_ptr**<**Date**>>** mylist**;**

mylist**.**push\_back**(**make\_shared**<**Date**>(**2**,** 6**,** 1999**));**

mylist**.**push\_back**(**make\_shared**<**Date**>(**2**,** 6**,** 1993**));**

mylist**.**push\_back**(**make\_shared**<**Date**>(**21**,** 11**,** 1997**));**

mylist**.**push\_back**(**make\_shared**<**Date**>(**7**,** 10**,** 2004**));**

**for** **(auto** ptr **:** mylist**)**

**{**

cout **<<** **\***ptr **<<** "\n"**;**

**}**

vector**<**shared\_ptr**><**Date**>>** myvec**(**mylist**.**begin**(),** mylist**.**end**());**

***// vectordeki shared\_ptr değiştirsem listteki shared\_ptr değişir***

sort**(**myvec**.**begin**(),** myvec**.**end**(),** **[](auto** p1**,** **auto** p2**)**

**{**

**return** **\***p1 **<** **\***p2**;**

**}**

**for** **(auto** sp **:** myvec**)**

**{**

cout **<<** **\***sp **<<** '\n'**;**

**}**

**}**

**using** svector **=** std**::**vector**<**std**::**string**>**

**class** NameList

**{**

**public:**

NameList**()** **=** **default;**

NameList**(**std**::**initializer\_list**<**std**::**string**>** list**>** **:** sptr**(new** svector**{**list**}}**

**{**

**}**

**void** add**(const** std**::**string**&** name**)**

**{**

sptr**->**push\_back**(**name**);**

**}**

**void** remove**(const** std**::**string**&** name**)**

**{**

sptr**->**erase**(**std**::**remove**(**sptr**->**begin**(),** sptr**->**end**(),** name**),** sptr**->**end**());**

**}**

**size\_t** size**()const**

**{**

**return** sptr**->**size**();**

**}**

**void** print**()const**

**{**

**for(const** **auto&** s **:** **\***sptr**)**

**{**

std**::**cout **<<** s **<<** " "**;**

**}**

std**::**cout **<<** "\n"**;**

**}**

**void** sort**()**

**{**

std**::**sort**(**sptr**->**begin**(),** sptr**->**end**());**

**}**

**private:**

std**::**shared\_ptr**<**svector**>** sptr**;**

**}**

**int** main**()**

**{**

***// x, y ve z 'nin veri elemanı olam shared\_ptr aynı vektörü gösterir.***

NameList x**{** "ali"**,** "gul"**,** "eda"**,** "naz"**};**

NameList y **=** x**;**

NameList z **=** y**;**

x**.**add**(**"nur"**);**

y**.**add**(**"tan"**);**

std**::**cout **<<** "listede " **<<** x**.**size**()** **<<** "isim var\n"**;**

z**.**sort**()**

x**.**print**();**

y**.**remove**(**"gul"**);**

z**.**print**();**

**}**

***// Bir sınıf için dinamik bellek yönetimini özelleştirme***

**class** Myclass

**{**

Myclass**()**

**{**

std**::**cout **<<** "default ctor this : " **<<** **this** **<<** "\n;

**}**

**~**Myclass**()**

**{**

std**::**cout **<<** "dtor this : " **<<** **this** **<<** "\n"**;**

**}**

***// operator new ve delete static yazmasak bile static üyedir***

**void\*** **operator** **new(size\_t)**

**{**

std**::**cout **<<** "Myclass::operator new n:" **<<** n **<<** "\n"**;**

**auto** p **=** std**::**malloc**(**n**);**

**if** **(!**p**)**

**{**

**throw** std**::**bad\_alloc**{};**

**}**

std**::**cout **<<** "address of the allocated block is " **<<** p **<<** "\n"**;**

**}**

**void** **operator** **delete(void\*** vp**)noexcept**

**{**

std**::**cout **<<** "Myclass::operator delete vp:" **<<** vp **<<** "\n"**;**

std**::**free**(**vp**);**

**}**

**void** func**()**

**{**

std**::**cout **<<** "Myclass func() this = " **<<** **this** **<<** "\n"**;**

**}**

**private:**

**unsigned** **char** buf**[**512**]{};**

**};**

**int** main**()**

**{**

**auto** p **=** **new** Myclass**;**

**delete** p**;**

**}**