گزارش کار

HW4

دانش عبداللهی

9723053

ابتدا کلاس UniquePtr را می‌نویسیم.

در ابتدا Constructor های کلاس را تعریف می‌کنیم. فقط برای اینکه برای Ovbejct این کلاس نتوان از Copy Constructor استفاده کرد ، Copy Constructor را برابر با delete قرار می‌دهیم. در این صورت در صورت استفاده از Copy Constructor برای Object ئی از این کلاس ، با Compile error مواجه می‌شویم.

```
UniquePtr(const UniquePtr&) = delete; // Copy Constructor
```

- تابع get()

یک تابع بدون ورودی و با خروجی اشاره‌گر است. این تابع متغیر _p ، Object را برمی‌گرداند.

- تابع reset()

تابعی بدون ورودی و خروجی است که حافظهٔ محتویات اشاره‌گر _p را به پاک می‌کند و سپس _p را برابر با Nullptr قرار می‌دهد.

- تابع reset(ptr)

تابعی با ورودی اشاره‌گر و بدون خروجی است که حافظهٔ محتویات اشاره‌گر _p را پاک می‌کند و سپس _p را برابر با ورودی قرار می‌دهد.

- **تابع ()release**

تابعی بدون ورودی و با خروجی اشاره‌گر است که Object _p را به عنوان خروجی برمی‌گرداند و دسترسی آنرا به محتویات فعلی آن قطع می‌کند. (آن را برابر با Nullptr قرار می‌دهد.)

- **تابع ()make_unique**

تابعی با ورودی یک متغیر از نوع دلخواه با مقدار معلوم است که خروجی آن از جنس UniquePtr است. در این با استفاده از Constructor ، یک Object از نوع UniquePtr به صورت R-Value می‌سازیم و آنرا برمی‌گردانیم. این کار باعث می‌شود که Compiler از RVO استفاده کند و از کپی کردن بی‌مورد جلوگیری شود. (در این صورت نیازی به استفاده از Copy Constructor نداریم و به Compile error بر نمی‌خوریم.)

- Operator*()
  محتویات اشاره‌گر p_ را برمی‌گرداند.
- Operator ->()
  اشاره‌گر p_ را برمی‌داند.
- Operator =
  همانند Copy Constuctor آنرا برابر با delete قرار می‌دهیم تا در صورت استفاده از آن با Compile error مواجه شویم.
- Operator bool()
  اگر اشاره‌گر p_ ، Nullptr باشد ، False و در غیر این صورت True را برمیگرداند در واقع با یک اپراتور Object از جنس UniquePtr را به متغیر Boolean تبدیل می‌کنیم.

سپس سراغ نوشتن کلاس SharedPtr می‌رویم.

در ابتدا Constructor های کلاس را تعریف می‌کنیم. در Constructor ، مقدار متغیر counter را برابر با 1 و در Default Contructor آنرا برابر با صفر قرار می‌دهیم.

همچنین در Copy Constuctor ، مقدار Conter ورودی را یکی زیاد می‌کنیم و آدرس آنرا در Conter ، Object جدید می‌ریزیم.

**در این کلاس یک متغیر جدید از جنس اشاره‌گر به int به نام counter تعریف می‌کنیم.

- تابع get()

یک تابع بدون ورودی و با خروجی اشاره‌گر است. این تابع متغیر p_ ، Object را برمی‌گرداند.

- تابع reset()

تابعی بدون ورودی و خروجی است که حافظهٔ محتویات اشاره‌گر های Object را پاک می‌کند و سپس آنها را برابر با Nullptr قرار می‌دهد.

- تابع reset(ptr)

تابعی با ورودی اشاره‌گر و بدون خروجی است که حافظهٔ محتویات اشاره‌گر p_ را به پاک می‌کند و سپس p_ را برابر با ورودی قرار می‌دهد. همچنین مقدار counter را برابر 1 قرار می‌دهد.

- تابع make_shared()

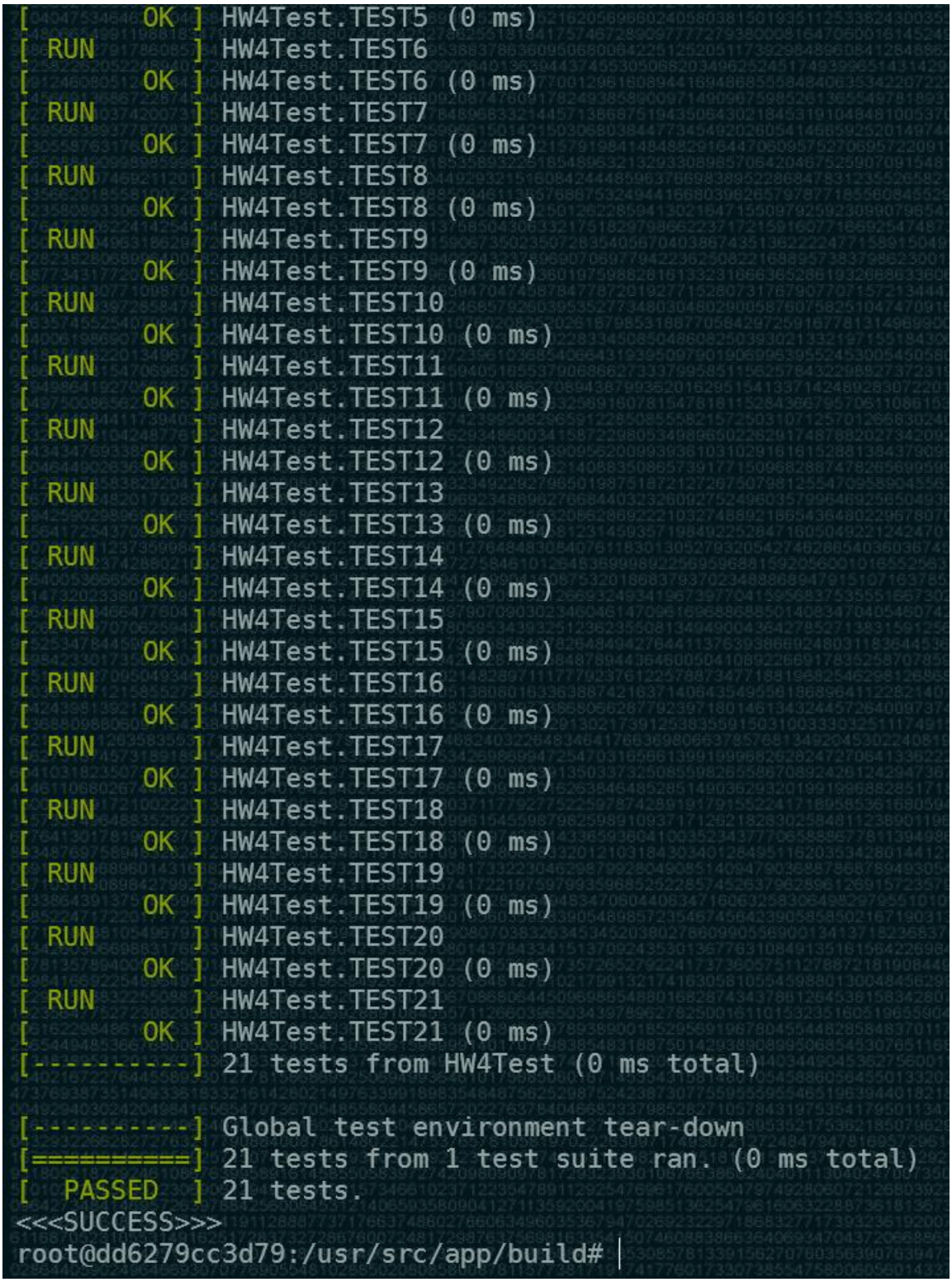
تابعی با ورودی یک متغیر از نوع دلخواه با مقدار معلوم است که خروجی آن از جنس SharedPtr است. در این با استفاده از Constructor ، یک Object از نوع

SharedPtr به صورت R-Value می‌سازیم و آنرا برمی‌گردانیم. این کار باعث می‌شود که Compiler از RVO استفاده کند و از کپی کردن بی‌مورد جلوگیری شود.

- Operator*()
  محتویات اشاره‌گر p_ را برمی‌گرداند.
- Operator ->()
  اشاره‌گر p_ را برمی‌داند.
- Operator =
  محتویات ورودی را در محتویات Object فعلی می‌ریزد و همچنین مقدار Counter آنها را به اضافهٔ 1 می‌کند.
- Operator bool()
  اگر اشاره‌گر p_ ، Nullptr باشد ، False و در غیر این صورت True را برمیگرداند در واقع با یک اپراتور Object از جنس SharedPtr را به متغیر Boolean تبدیل می‌کنیم.
- تابع use_count()
  اگر counter برابر Nullptr بود ، 0 و در غیر این صورت مقدار آنرا برمی‌گرداند. تابع بدون ورودی و با خروجی int است.
- Destructor

در Destructor این کلاس باید توجه کنیم که اگر Object ئی که Destructor آن صدا زده شده است ، مقدار counter آن غیر از 1 بود ، اشاره‌گر های آن را Delete نکنیم و فقط p_ آنرا برابر با Nullptr قرار دهیم و فقط در صورتی که مقدار Counter آن 1 بود ، اشاره‌گرهای آنرا delete کنیم و مقادیر آنها را برابر با Nullptr قرار دهیم. (اگر این کار را نکنیم به ارور Double free مواجه می‌شویم.)

لینک github


[OK] HW4Test.TEST5 (0 ms)
[RUN] HW4Test.TEST6
[OK] HW4Test.TEST6 (0 ms)
[RUN] HW4Test.TEST7
[OK] HW4Test.TEST7 (0 ms)
[RUN] HW4Test.TEST8
[OK] HW4Test.TEST8 (0 ms)
[RUN] HW4Test.TEST9
[OK] HW4Test.TEST9 (0 ms)
[RUN] HW4Test.TEST10
[OK] HW4Test.TEST10 (0 ms)
[RUN] HW4Test.TEST11
[OK] HW4Test.TEST11 (0 ms)
[RUN] HW4Test.TEST12
[OK] HW4Test.TEST12 (0 ms)
[RUN] HW4Test.TEST13
[OK] HW4Test.TEST13 (0 ms)
[RUN] HW4Test.TEST14
[OK] HW4Test.TEST14 (0 ms)
[RUN] HW4Test.TEST15
[OK] HW4Test.TEST15 (0 ms)
[RUN] HW4Test.TEST16
[OK] HW4Test.TEST16 (0 ms)
[RUN] HW4Test.TEST17
[OK] HW4Test.TEST17 (0 ms)
[RUN] HW4Test.TEST18
[OK] HW4Test.TEST18 (0 ms)
[RUN] HW4Test.TEST19
[OK] HW4Test.TEST19 (0 ms)
[RUN] HW4Test.TEST20
[OK] HW4Test.TEST20 (0 ms)
[RUN] HW4Test.TEST21
[OK] HW4Test.TEST21 (0 ms)
[----------] 21 tests from HW4Test (0 ms total)
[----------] Global test environment tear-down
[==========] 21 tests from 1 test suite ran. (0 ms total)
[PASSED] 21 tests.
<<<SUCCESS>>>
root@dd6279cc3d79:/usr/src/app/build#