小明的智能指针

小明的智能指针

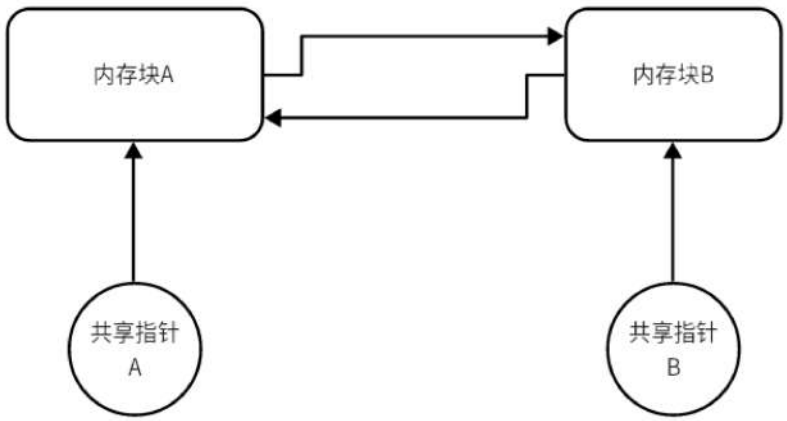
刷新

题目描述

小明在实现许多需要使用指针的数据结构时，常常因为内存泄露或者访问非法内存而导致~~小明~~(程序)崩溃。后来，小明了解到了智能指针的存在，可是小明实在是太懒了，请你帮他实现一下简单的智能指针吧。

事实上，C++的标准模板库中提供了 `shared_ptr` ， `unique_ptr` 和 `weak_ptr` 等智能指针模板类，然而今天我们只需要实现一个简单版本的 `shared_ptr` 和 `weak_ptr` 即可（让小明自己去写 `unique_ptr` 罢）。其中， `shared_ptr` 指向一块内存区域，该内存资源具有共享性，即多个 `shared_ptr` 可以指向同一资源。当所有指向该内存的共享指针全部析构后，释放该内存。而 `weak_ptr` 的行为类似于 `shared_ptr` ，不过**不影响该内存的生命周期（即被 `weak_ptr` 指向不会增加引用计数）**。

使用共享指针进行内存管理时，你可能会遇到循环引用的问题。如图，考虑一个长度为2的双向链表：



内存块A和B之间的相互联系用共享指针实现，因此A和B的引用计数均为2。当共享指针A和B释放后，内存块A和B的引用计数为1，内存并未得到释放，从而造成内存泄漏。

通常采用的解决方法是，将内存块A和B内部的共享指针中的一个换为弱指针，这样A和B的引用计数分别为1和2，从而可以正确的释放内存。事实上，本题也是这么做的。

你需要编写两个模板类 `MySharedPtr` 和 `MyWeakPtr` ，来实现 `shared_ptr` 和 `weak_ptr` 的基本功能，具体包括：

1. 内存管理：你的 `MySharedPtr` 应该可以管理一块内存，**并且在所有指向这个内存区域的 `MySharedPtr` 全部析构后释放该内存**。你可以通过拷贝构造函数和拷贝赋值运算符来管理同一块内存，通过移动构造函数和移动赋值运算符来接管另一块内存。
2. 重载运算符：你的 `MySharedPtr` 需要支持指针的两种操作 `*` 和 `->` 。
3. 共享指针和弱指针的转换：**弱指针不可以直接操作内存**，需要将其通过 `lock()` 方法转换为 `MySharedPtr` 后进行操作。（注意：如果该 `MyWeakPtr` 指向的内存已经被释放， `lock()` 方法应该返回一个空的 `MySharedPtr` ）
4. 计数：你的 `MySharedPtr` 和 `MyWeakPtr` 需要能够通过 `use_count()` 方法查看指向该内存的** `MySharedPtr` 的数量**。

提示：你可能会用到如下类型转换运算符

```
operator bool() { return data != nullptr; }
```

`main.cpp` 已经为你写好，你仅需要完成 `MyPtr.h` 来实现题目要求的功能。

下载地址：点击下载 (/staticdata/2016.VsaOuh7nxDhnZBYr.pub/TyrKXbmLumsqSsDX.download.zip/download.zip)

测试点分为4个子任务：

1. 测试内存管理和引用计数
2. 测试构造和赋值
3. 测试运算符重载和弱指针
4. 综合测试

你可以实现不同的子任务来得到不同的分数。我们在上述下载文件中提供了4个子任务公开测试点的正确输出（不包含隐藏测试点）。

输入格式

本题无输入，请参考下载文件中的main.cpp

输出格式

当 `MySharedPtr` 被构造或析构时，你需要输出一行形式为 `my_sptr constructed.` 或 `my_sptr destructed.` 的信息。

当 `MyWeakPtr` 被构造或析构时，你需要输出一行形式为 `my_wptr constructed.` 或 `my_wptr destructed.` 的信息。

当调用赋值运算符时，你需要输出一行形式为 `my_sptr updated.` 或 `my_wptr updated.` 的信息。

当你管理的内存被真正释放(`delete p` ， `p` 不为空)时，你需要输出一行形式为 `data released.` 的信息。

例如子任务2的输出：

```
my_sptr constructed.
my_sptr constructed.
my_sptr constructed.
my_sptr updated.
my_sptr updated.
data released.
my_sptr constructed.
my_sptr updated.
my_sptr updated.
3330
my_sptr destructed.
my_sptr destructed.
my_sptr destructed.
my_sptr destructed.
data released.
Accepted!
```

部分解释：首先构造3个 `MySharedPtr` ，当运行完 `p3 = p2;` 和 `p1 = p2;` 后， `p1` 所管理的内存引用计数为0，故释放该内存区域。

提示

本题分为了4个 subtask，每个 subtask 各占25分。你可以分别实现来得到部分分数，除了提供的 `main.cpp` 里的测试点，每个子任务还各有一个隐藏测试点，你需要都通过才能获得这个子任务的分数。四个 subtask 对应的测试点编号分别为：(1,5)，(2,6)，(3,7)，(4,8)。

要求

1. 不能修改 `main.cpp` 和 `Makefile` ，即使你修改了，也会被覆盖掉:(
2. 修改 `MyPtr.h` ，编写 `MySharedPtr` 和 `MyWeakPtr` 模板类来实现题目要求的功能，你可以借助 `MyPtr.h` 中已经写好的部分，也可以全部自己实现。

提交格式

请将你的文件打包成一个 zip 格式的压缩包并上传。

注意：你的文件应该在压缩包的根目录下，而不是压缩包的一个子文件夹下，换而言之，解压你提交的压缩包后，应该直接得到一系列 cpp 文件、h 文件等代码文件，而不是一个包含它们的文件夹。评测时，OJ会将提供的文件贴入你的目录下进行编译并执行。

资源限制

时间：1000ms；内存：512MB

语言和编译选项

#	名称	编译器	额外参数	代码长度限制
0	oop_custom	`make`		65536 B

递交历史

#	状态	时间

递交答案

语言和编译选项 oop_custom

1

提交

文件请拖入编辑器中，或 上传文件