实验1：缓冲池管理器实现

⼀、实验⽬的

1. 掌握Rucbase缓冲池⻚⾯替换策略的实现⽅法。

2. 掌握Rucbase缓冲池管理器的实现⽅法。

二、相关知识

1. 缓冲池的组成

2. 缓冲池的⻚⾯替换策略

3. 缓冲池访问请求的处理⽅法

三、实验内容

本实验包括2项任务。

**任务1：缓冲池⻚⾯替换策略实现**

补全 LRUReplacer 类，实现最近最少使⽤（Least Recently Used, LRU）⻚⾯替换策略。 LRUReplacer 类继承了 Replacer 类。当缓冲池没有空闲⻚⾯时，缓冲池管理器需要使⽤ Replacer 类实现的⻚⾯替换策略选择⼀个⻚⾯进⾏淘汰。

**1.1理解 LRUReplacer 类的设计，并回答下列问题：**

1. LRUlist\_ 的作⽤是什么？
2. LRUhash\_ 的作⽤是什么？

3. LRUlist\_ 和 LRUhash\_ 的关系是什么？

**1.2在这一部分，你需要列举并介绍你实现的全部方法。你需要按照方法所在的类进行组织。在介绍每个方法的具体实现时，需要包含以下内容：**

1. 方法的声明。给出方法的声明（注意：是方法声明，不是方法定义）。如果这个方法是你自己声明的，请说明它的功能以及为何要声明这个方法，何时调用这个方法。
2. 方法实现思路。根据方法实现的难度，可以采用不同的介绍形式。对于简单的方法，简要介绍方法的实现思路即可。对于复杂的方法，如果执行过程非常复杂，可以借助流程图或伪代码进行介绍。
3. 方法实现难点。如果你在实现这个方法的过程中遇到了较大的困难，不妨介绍一下你遇到的是什么困难，你最终的解决办法是什么。

Eg:

1.2.1：

void DiskManager::write\_page(int fd, page\_id\_t page\_no, const char \*offset, int num\_bytes);

**1.方法声明：**

方法名：write\_page

返回类型：void

功能：在对应文件的对应页面写入规定数量的数据

参数列表：

|  | **fd** | **page\_no** | **\*offset** | **num\_bytes** |
| --- | --- | --- | --- | --- |
| 类型 | int | page\_id\_t ：int32\_t | const char | int |
| 含义 | 文件描述符 | 页号 | 存放数据 | 数据的bit数 |

**2.方法实现思路：**

lseek()定位到文件头，通过(fd,page\_no)可以定位指定页面及其在磁盘文件中的偏移量

调用write()函数

如果写入到数据数量和num\_bytes不等，注意处理异常

**3.遇到困难….**

**任务2：缓冲池管理器实现**

补全 BufferPoolManager 类，实现Rucbase缓冲池管理.

**2.1理解 Page 和 BufferPoolManager 类的设计，并回答下列问题：**

1. Page::is\_dirty\_ 的作⽤是什么？

2. Page::pin\_count\_ 的作⽤是什么？

3.BufferPoolManager::page\_table\_ 的作⽤是什么？

4. BufferPoolManager::free\_list\_ 的作⽤是什么？

**2.2:同1.2**

四、实验总结