**《计算机操作系统》实验报告**

**实验题目：请求页式存储管理**

**姓名：汪雨卿 学号：19120191 实验日期：2021.12.23**

**实验环境：**

Visual Studio 2019; C++

**实验目的：**

近年来，由于大规模集成电路（LSI）和超大规模集成电路（VLSI）技术的发展，使存储器的容量不断扩大，价格大幅度下降。但从使用角度看，存储器的容量和成本总受到一定的限制。所以，提高存储器的效率始终是操作系统研究的重要课题之一。虚拟存储技术是用来扩大内存容量的一种重要方法。学生应独立地用高级语言编写几个常用的存储分配算法，并设计一个存储管理的模拟程序，对各种算法进行分析比较，评测其性能优劣，从而加深对这些算法的了解。

**实验内容：**

（1） 编制和调试**示例**给出的请求页式存储管理程序，并使其投入运行。

（2） 增加 1～2 种已学过的淘汰算法，计算它们的页面访问命中率。试用各种算法的命中率加以比较分析。

**提示：**可选用 FIFO 方法，即先访问的页先淘汰，也可选用 LRU 方法中的其他方案。如在页表中设置标志位，按标志位值得变化来淘汰。也可用 LFU 方法，为页表中各页面设置访问计数器，淘汰访问频率最低的页（注意：当前访问的页不能淘汰）等等。

**操作过程：**

**1. 设计算法的流程图**

*图1请求页式存储管理逻辑过程*中展示了本程序的算法逻辑过程。

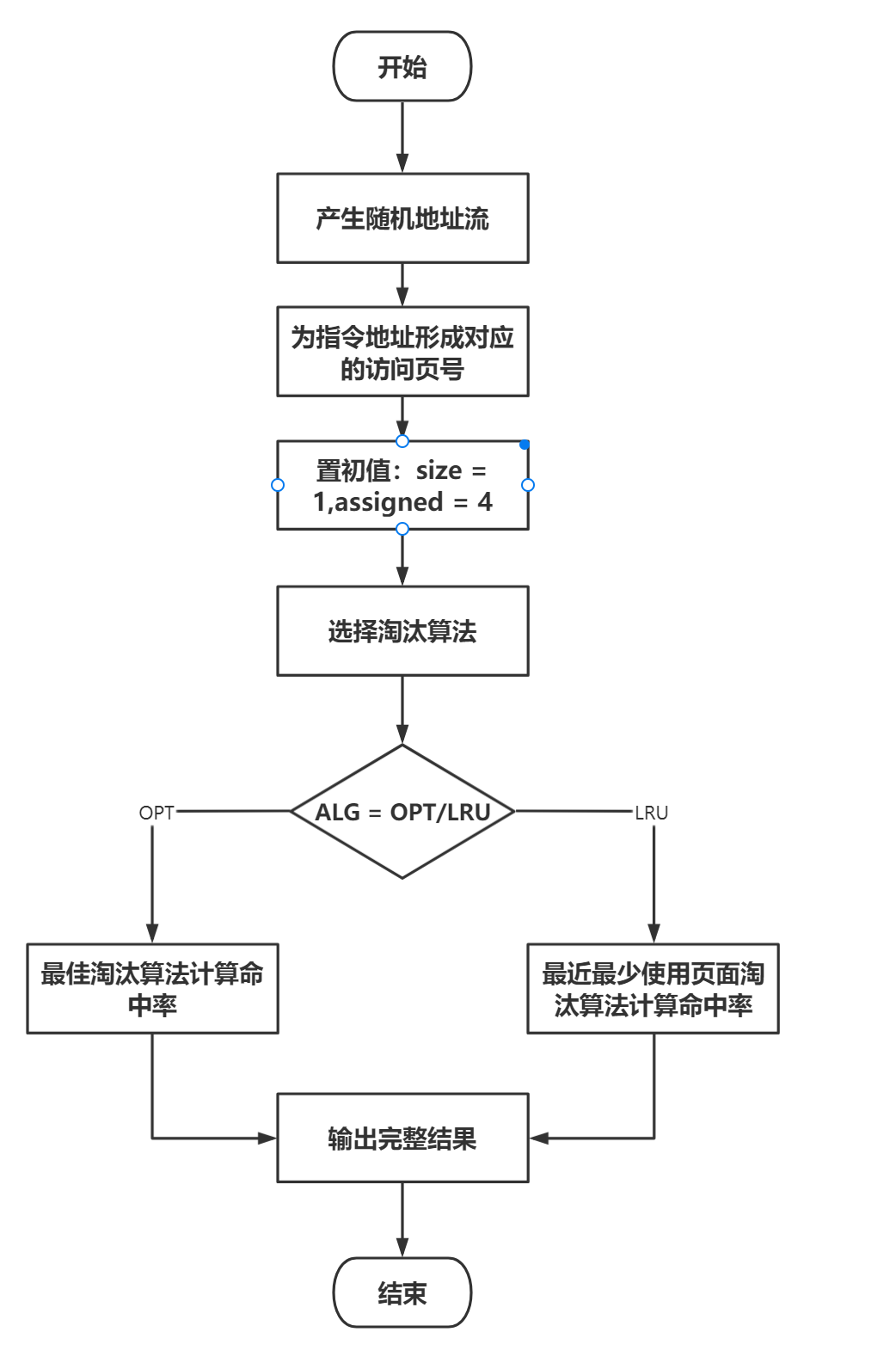


图1请求页式存储管理逻辑过程

**2. 设计全局变量**

**1. 全局变量**

**- \*random\_num 生成随机地址的数组**

**- \*page\_num 每个地址对应的页号**

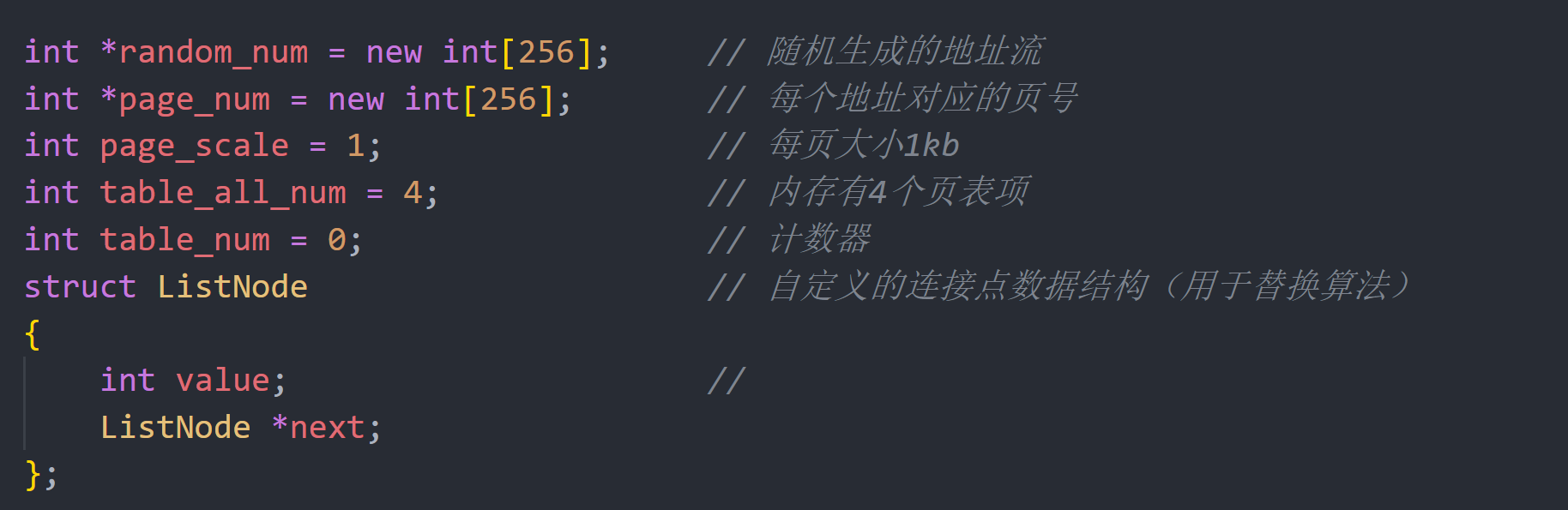
**- page\_scale 每页大小**

**- table\_all\_num 页表项个数**

**- table\_num 计数器**

**- ListNode 链表结点**

**图2全局变量 实现中展示了具体代码实现。**



**图2全局变量**

**3. 请求页式存储管理的具体实现**

**a. 生成随机地址流**

**get\_random\_addr()方法：利用rand()%(n-m+1)+m生成[m,n]范围内的一个随机数，模拟生成地址。利用循环，局部变量m的设置实现地址在前地址区，后地址区以及随机区域均匀的分布。**

**程序描述：**

**① 在[0，255]之间随机选取一条起始执行指令，其序号为m；**

**② 顺序执行下一条指令，即序号为m+1的指令；**

**③ 通过随机数，跳转到前地址部分[0，m-1]中的某条指令处，其序号为m1；**

**④ 顺序执行下一条指令，即序号为m1+1的指令；**

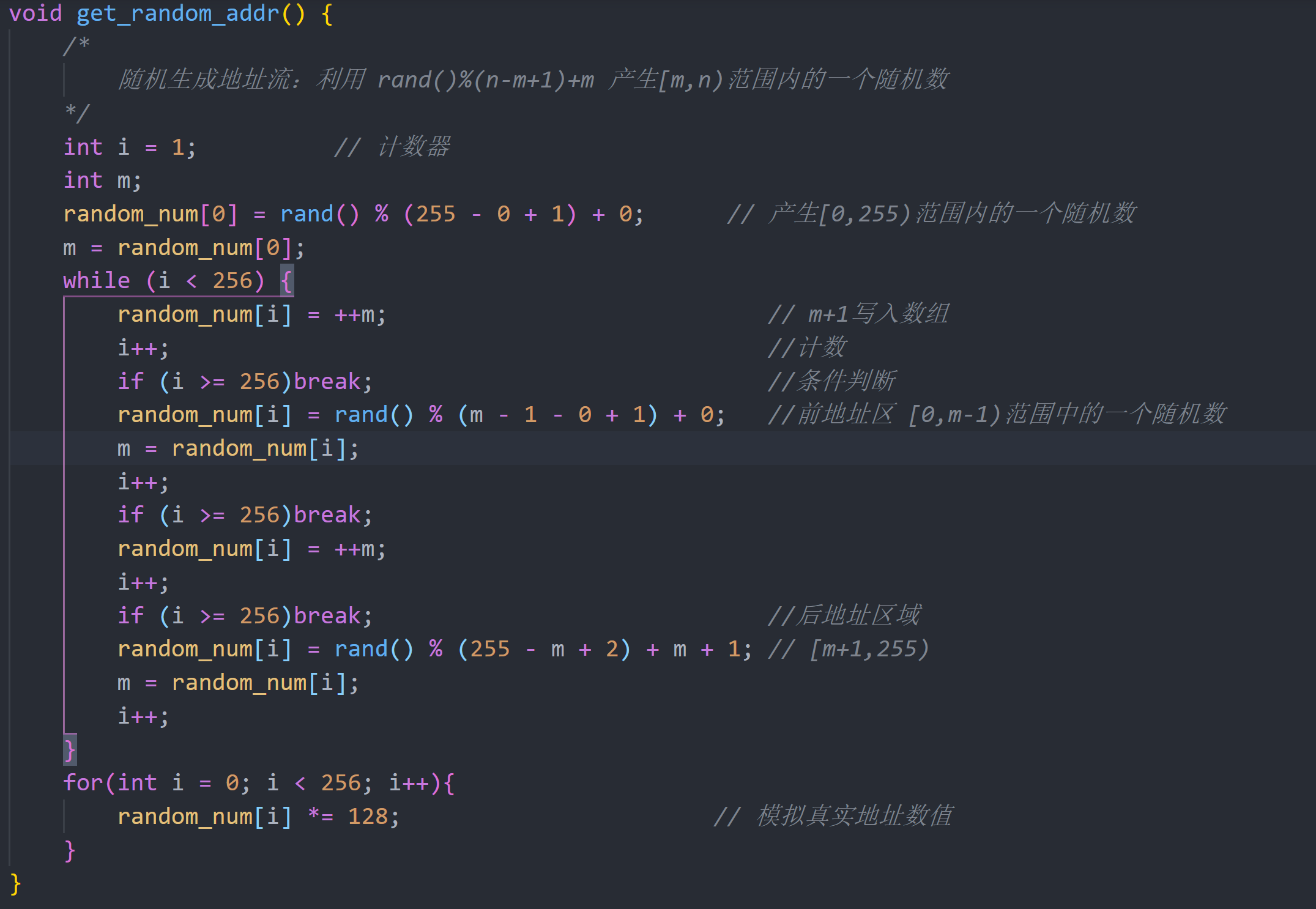
**⑤ 通过随机数，跳转到后地址部分[m1+2，255]中的某条指令处，其序号为m2；**

**⑥ 顺序执行下一条指令，即序号为m2+1的指令；**

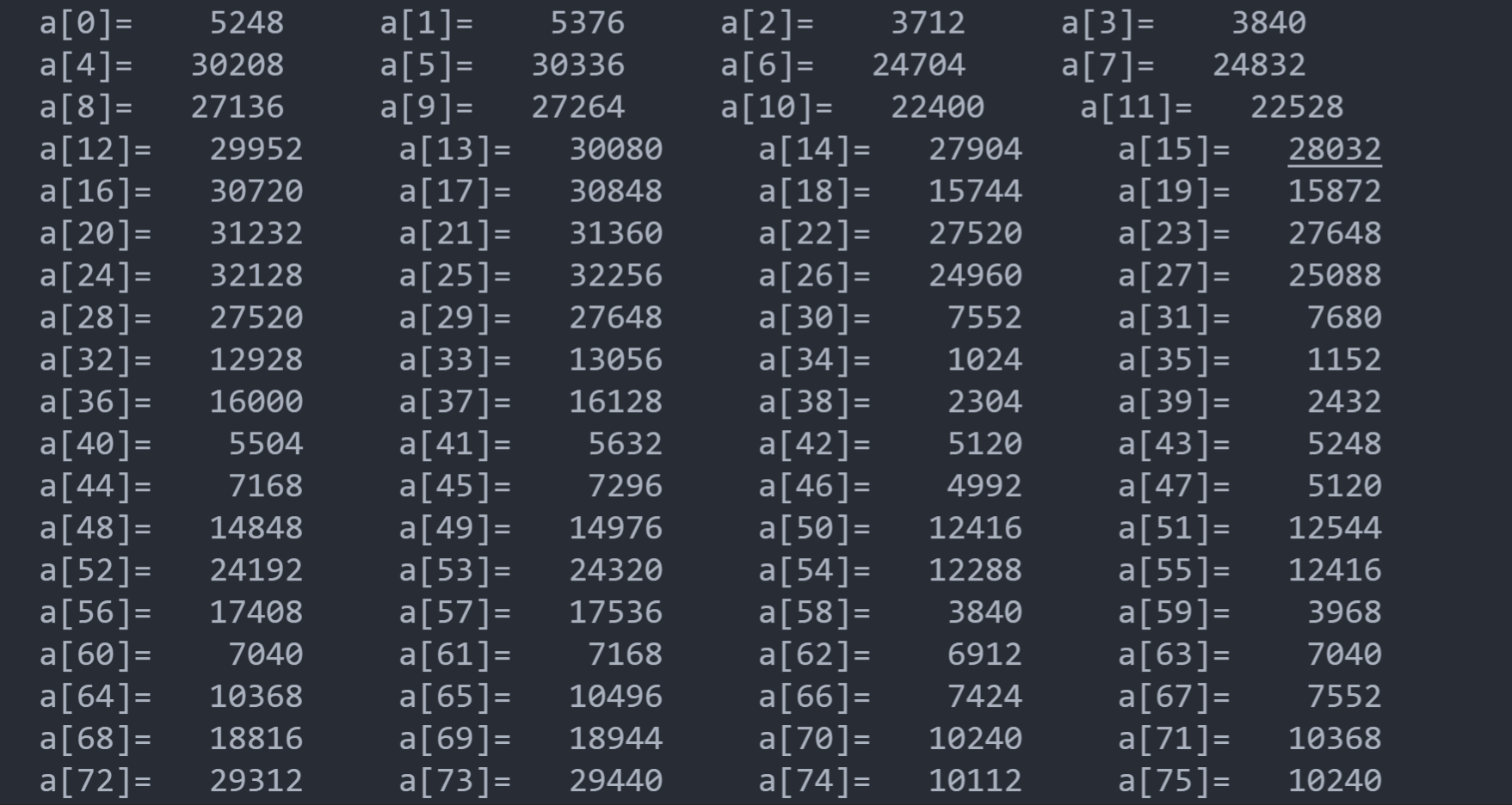
**⑦ 重复跳转到前地址部分、顺序执行、跳转到后地址部分、顺序执行的过程，直至执行255条指令**

***图3 get\_random\_addr()代码* 显示该程序的具体实现。**

***图4 随机生成地址流结果截图（部分）*显示某次执行该程序的结果输出，截图中只是生成地址中的前部分数据结果。**



**图3 get\_random\_addr()代码**

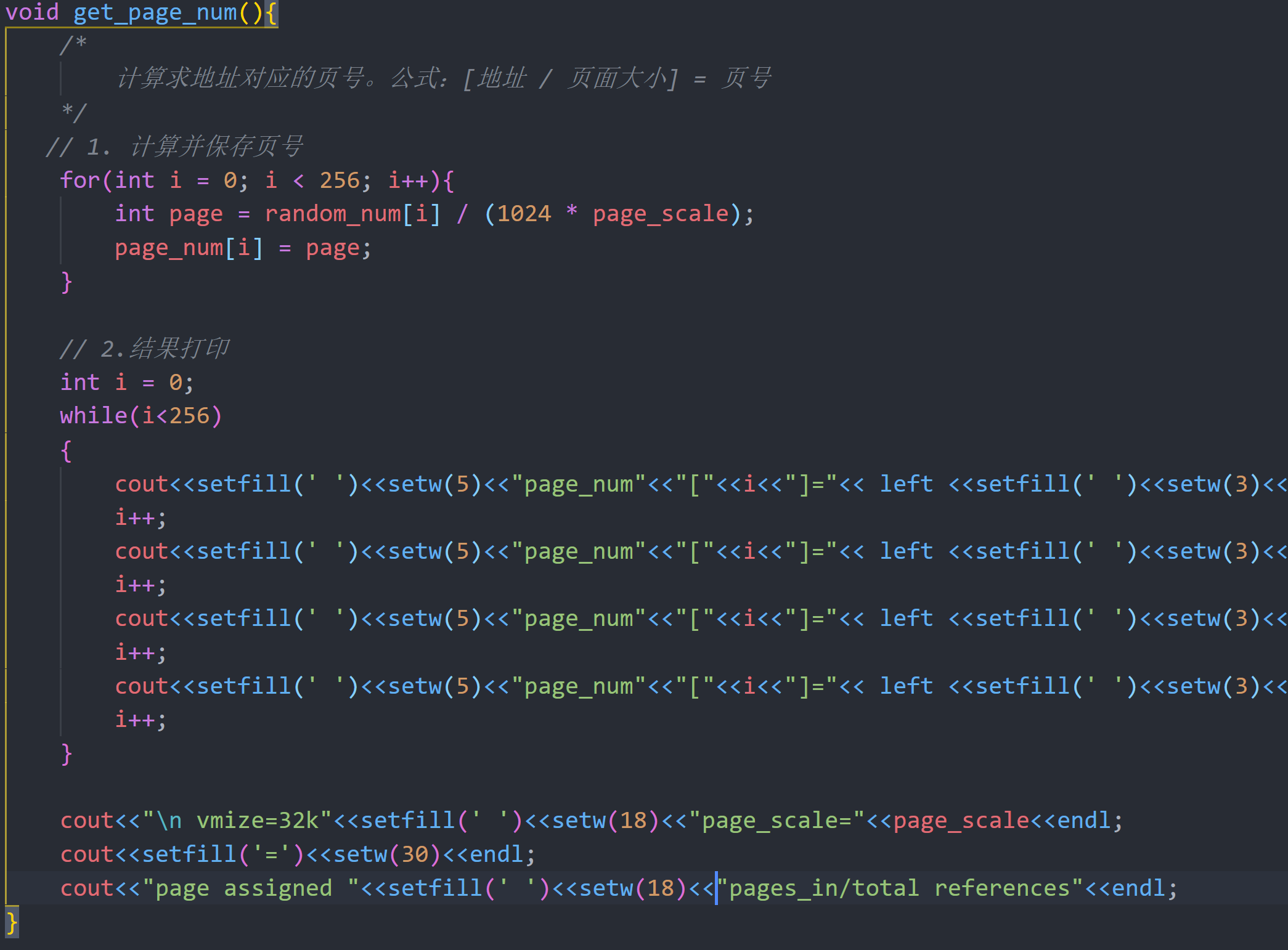


**图4 随机生成地址流结果截图（部分）**

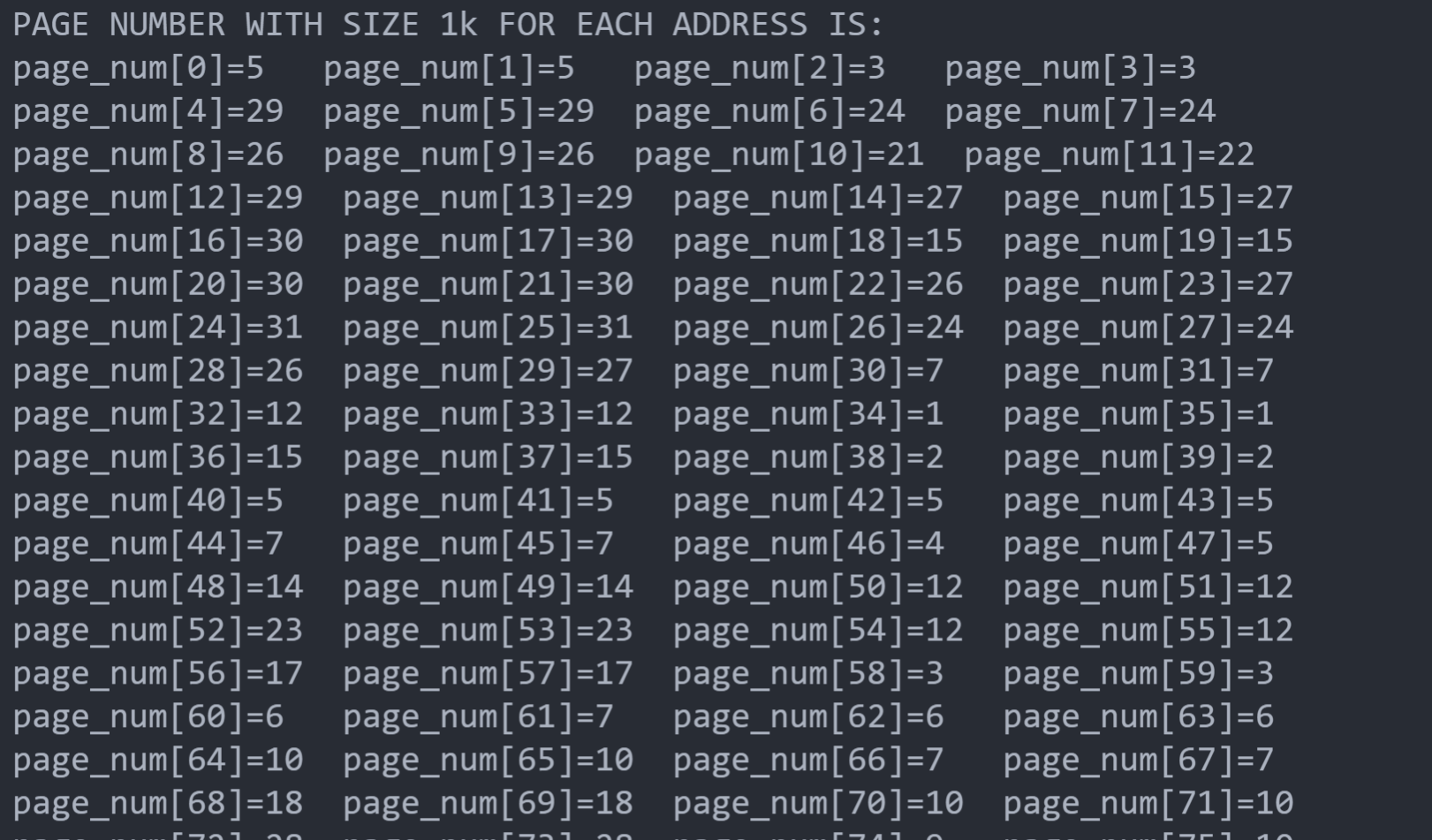
**b. 通过指令大小计算页号**

**设置一页的大小为1k。地址 / 1024 = 页号。**

**get\_page\_number()方法：利用以上公示，求出每一个地址的页号，并且存储在page\_num[]的数组中。**



**图5 get\_page\_num()代码**



**图6 地址对应页号计算结果局部截图**

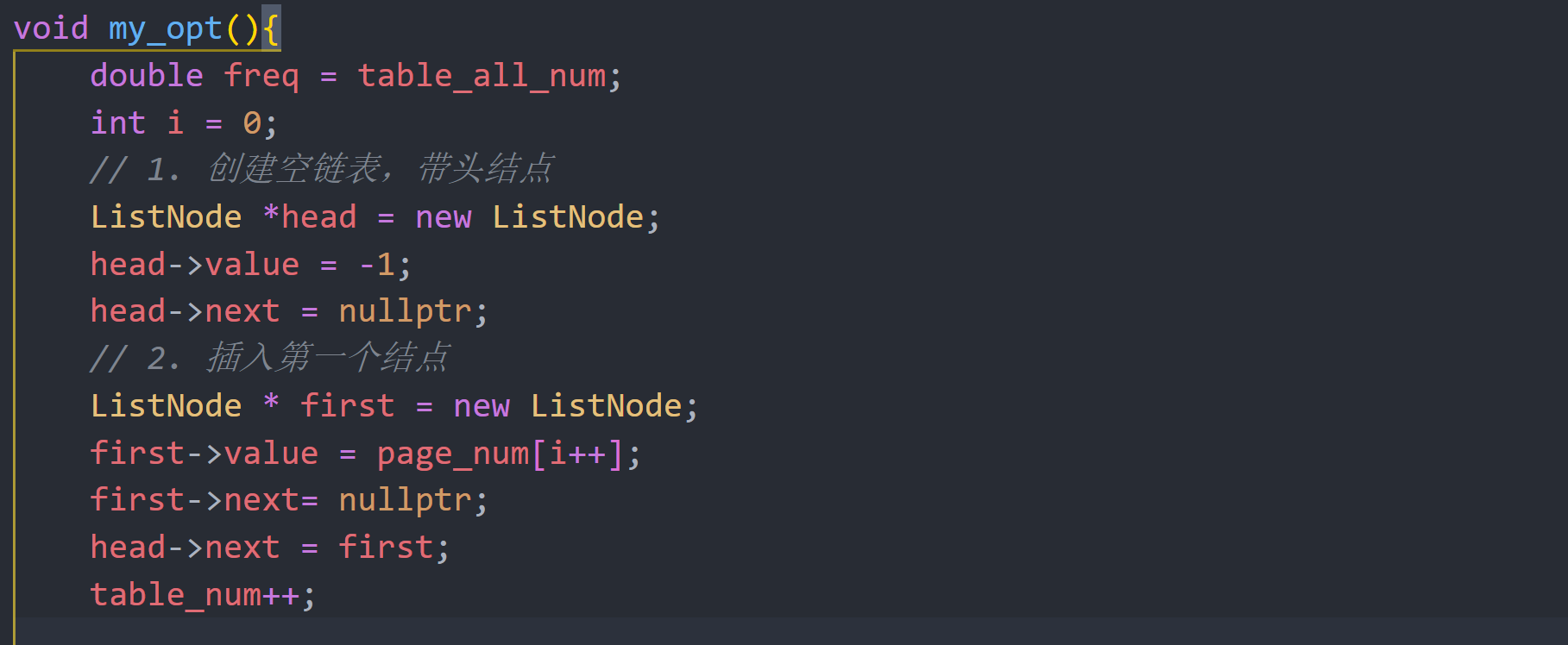
**c. OPT算法的实现**

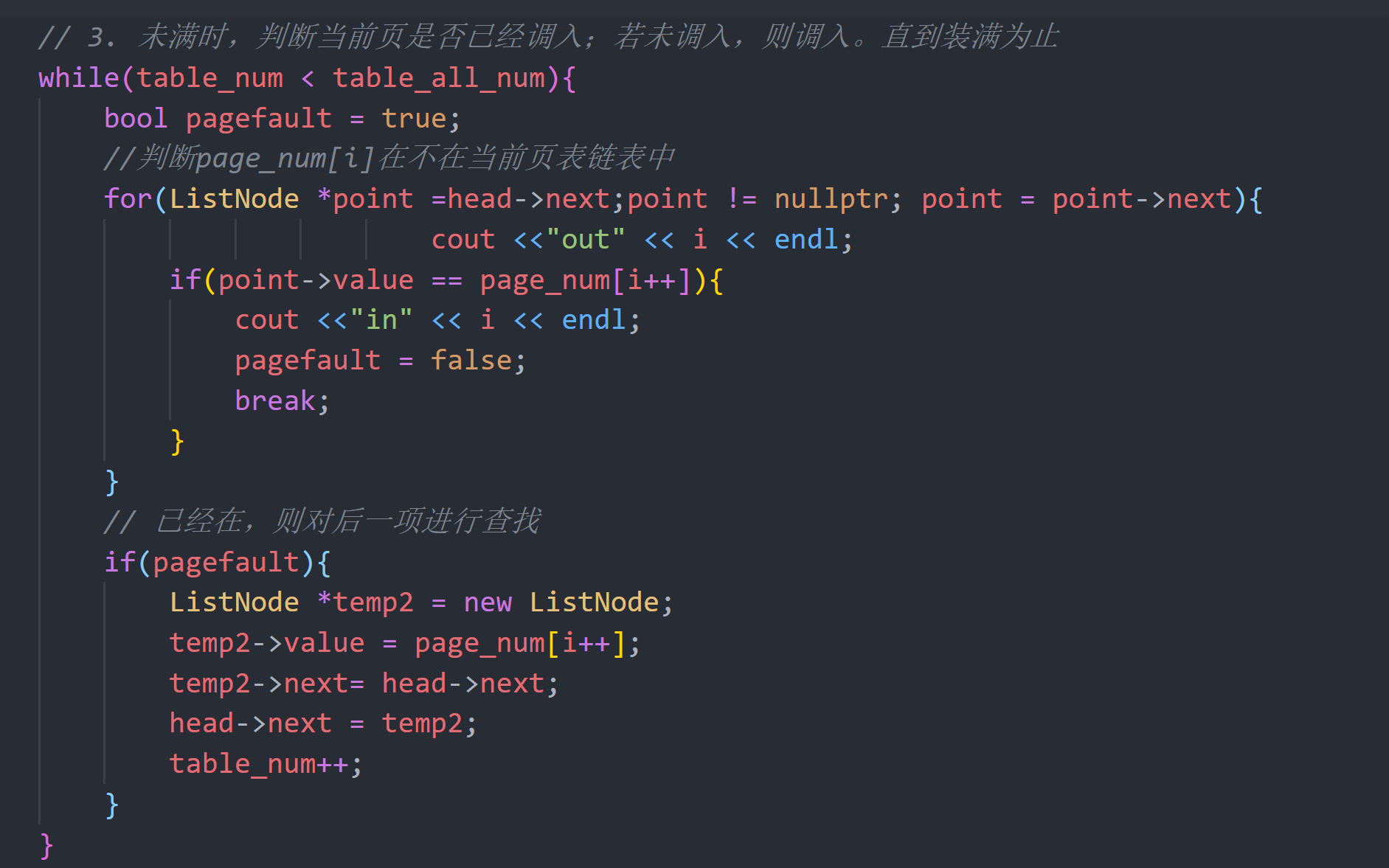
最佳置换算法，其所选择的被淘汰的页面将是以后永不使用的，或是在最长（未来）时间内不再被访问的页面。采用最佳置换算法通常可保证最低的缺页率。（OPT并不能真正的实现，但是可以通过类似的方法推测。常用于比较的标准）。

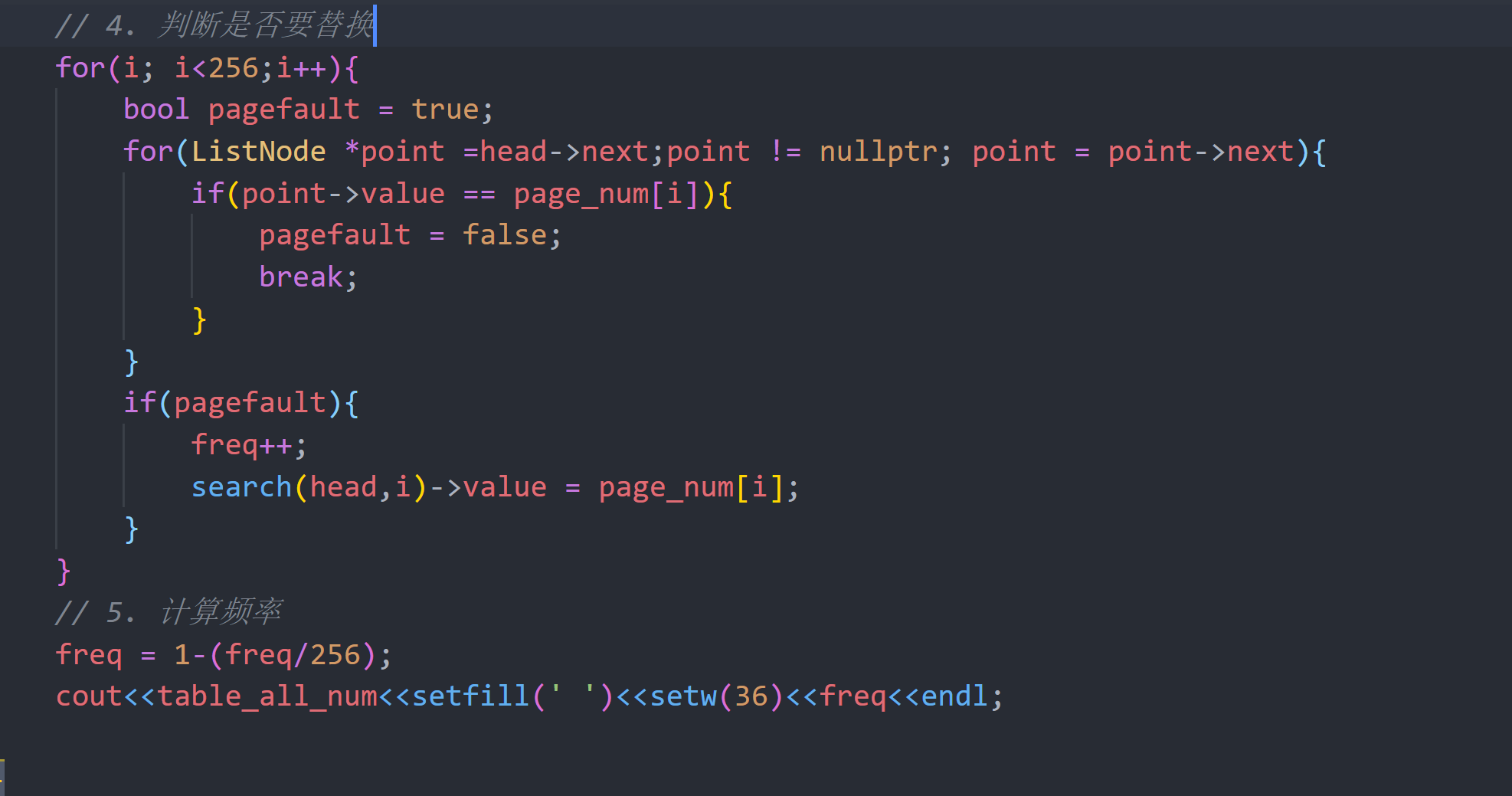
**算法描述：**

1. 将first——a[0]指针放进链表中。
2. 在页表项未满，也就是还有空闲页的时候，将当前页页调入内存（前提是当前页不在内存中）
3. 当页表项已满时，调用search函数计算出需要被调换的链表结点，调换出相应的页号。
4. 计算调换的次数，求出命中率。
5. 原理：使用链表结构，以队列的形式将一个个结点插入队列，然后当队列长度等于设置的最高队列长度时，采用OPT最晚出现置换规则置换，然后得到调换次数。

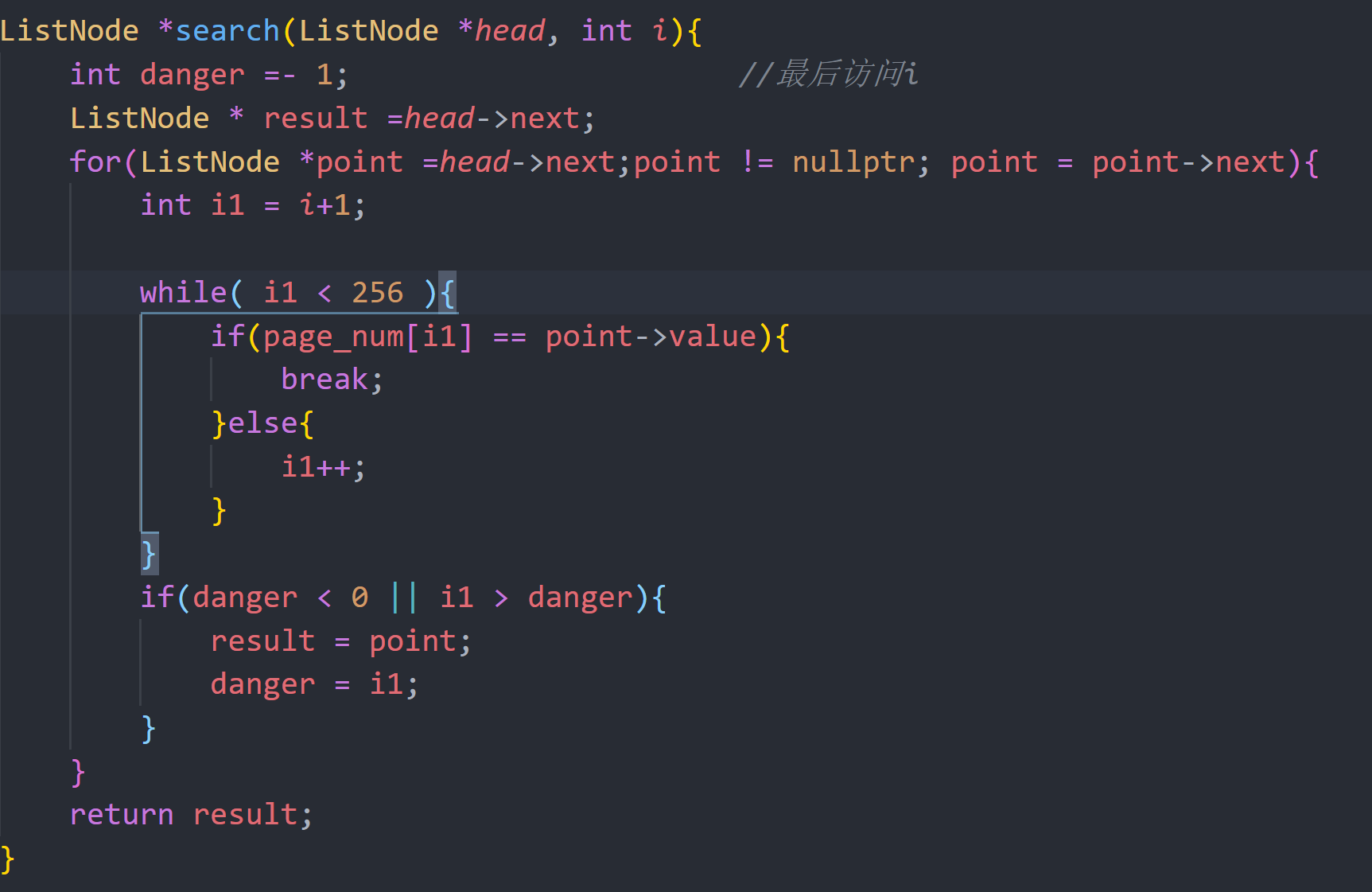
以下是具体的代码实现：







**\*\* search()方法原理：每一个链表结点的页都去计算出最迟出现的位置，从中挑选出最晚出现的结点返回，就是需要替换的页号。**



**d.lru算法**

**算法描述**

1.将first——a[0]指针放进链表中。

2.在页表项未满，也就是还有空闲页的时候，将页一页页调入内存（前提是当前页不在内存中）

3.每一次访问都将访问的页表结点都置于链表的尾部。

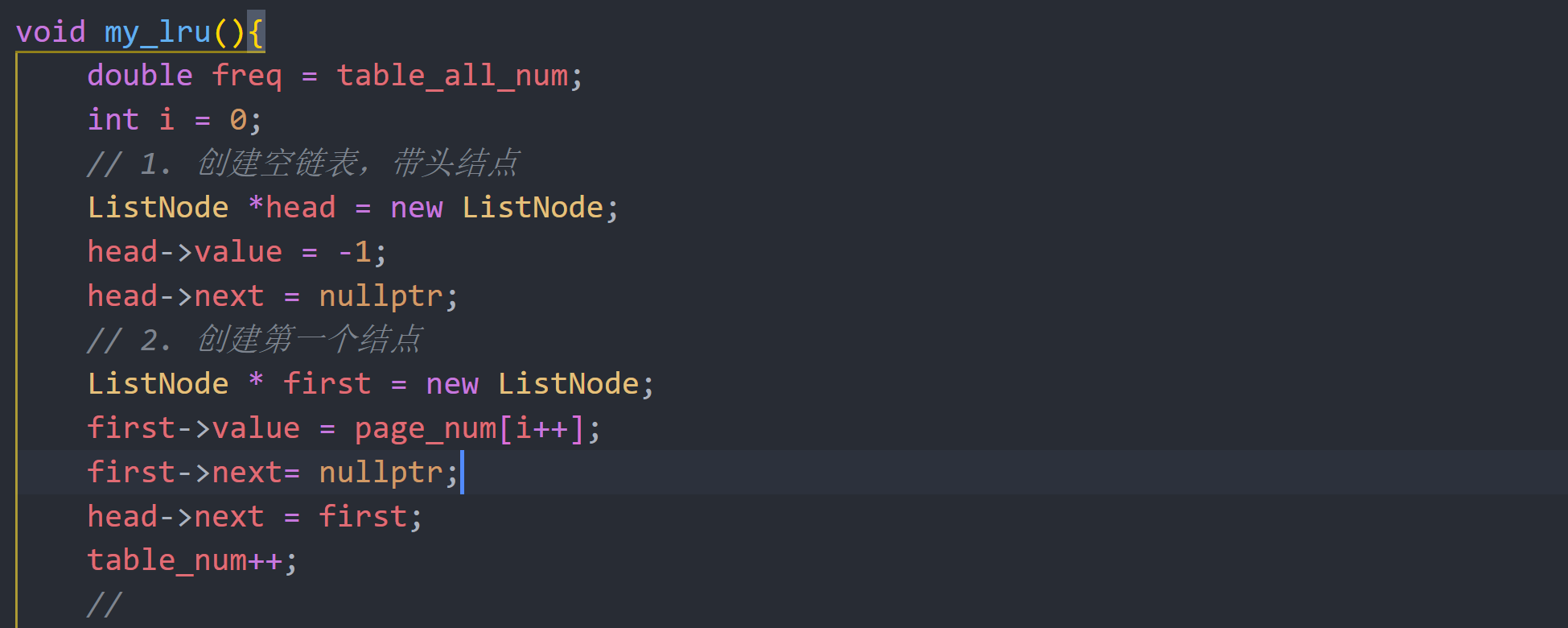
4.当链表已满时，

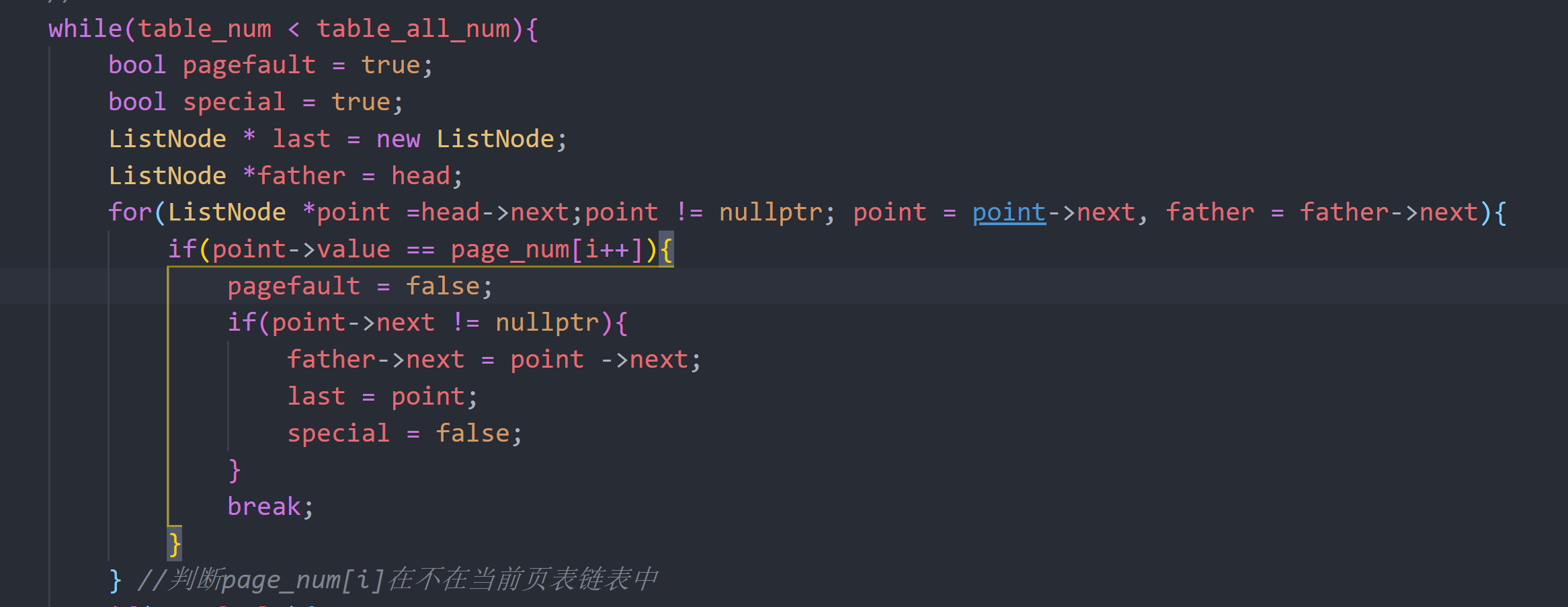
a没命中：将新页置于链表尾，然后删除第一个结点。

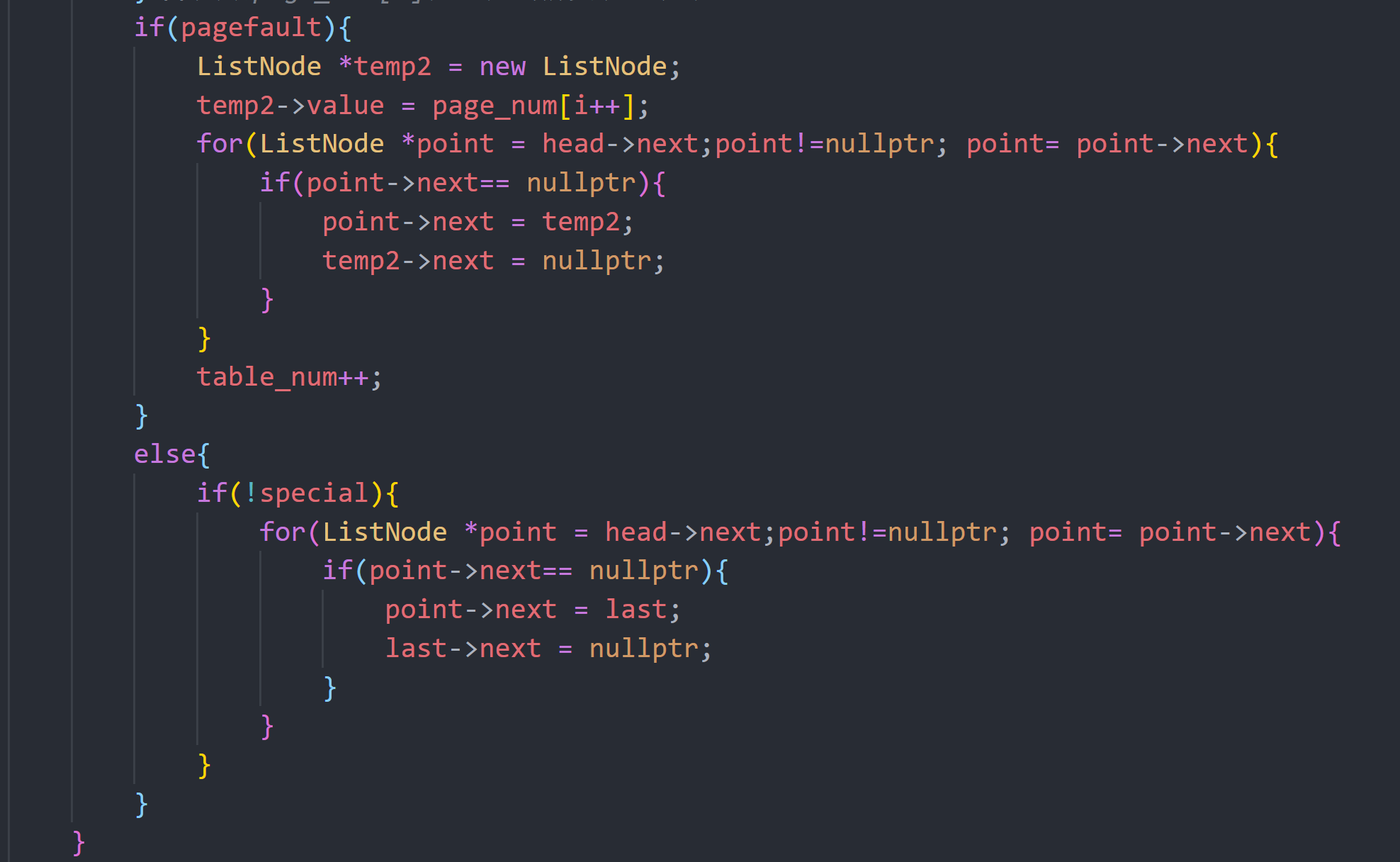
b命中：将命中的页置于链表尾

5.计算出没命中的次数。计算命中率。

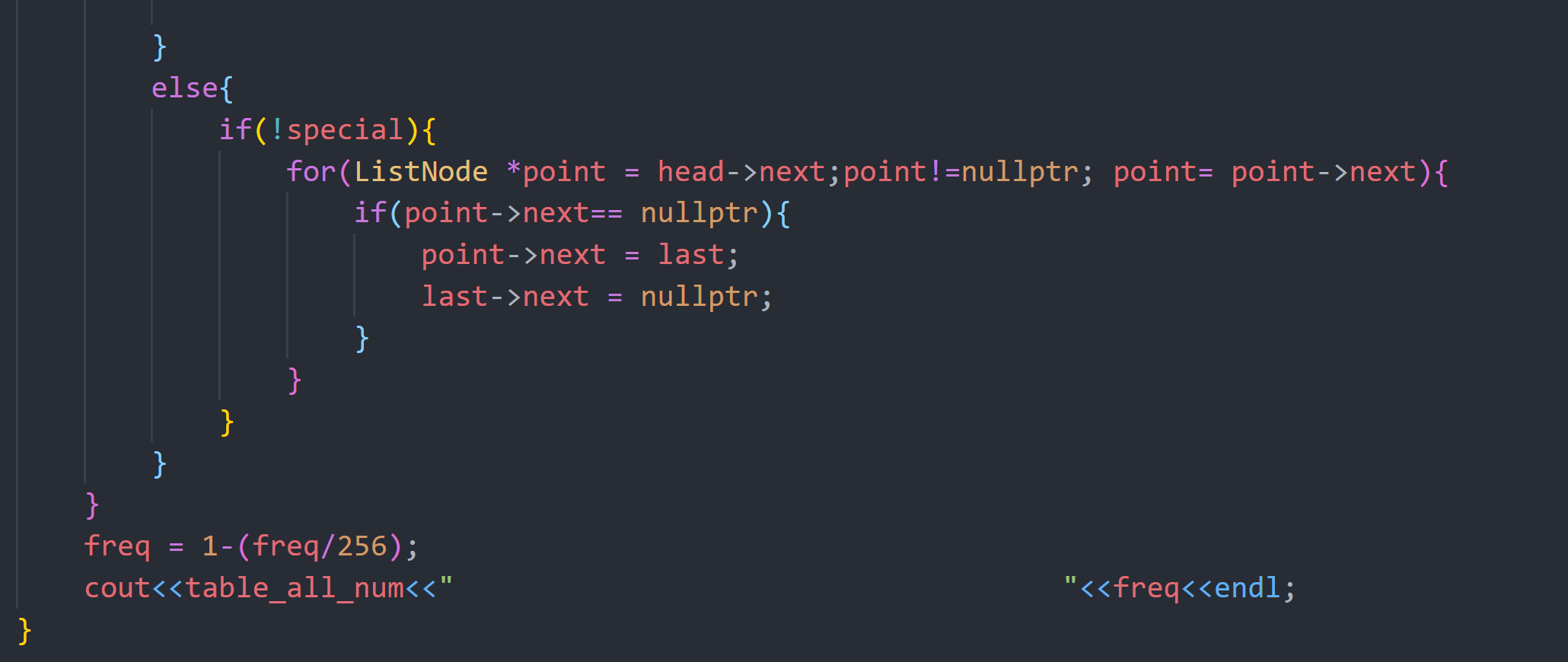
1. 原理：使用链表结构，以队列的形式将一个个结点插入队列，并根据LRU规则调整队列，然后当队列长度等于设置的最高队列长度时。淘汰队列最末尾的结点，并调整队列顺序。





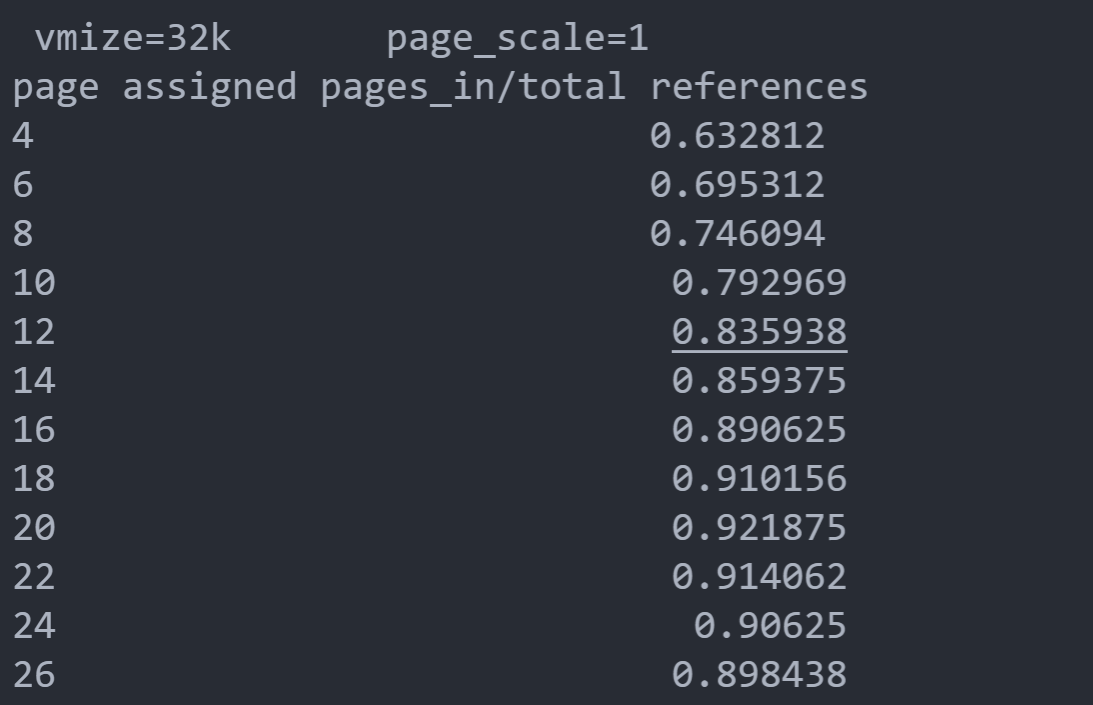




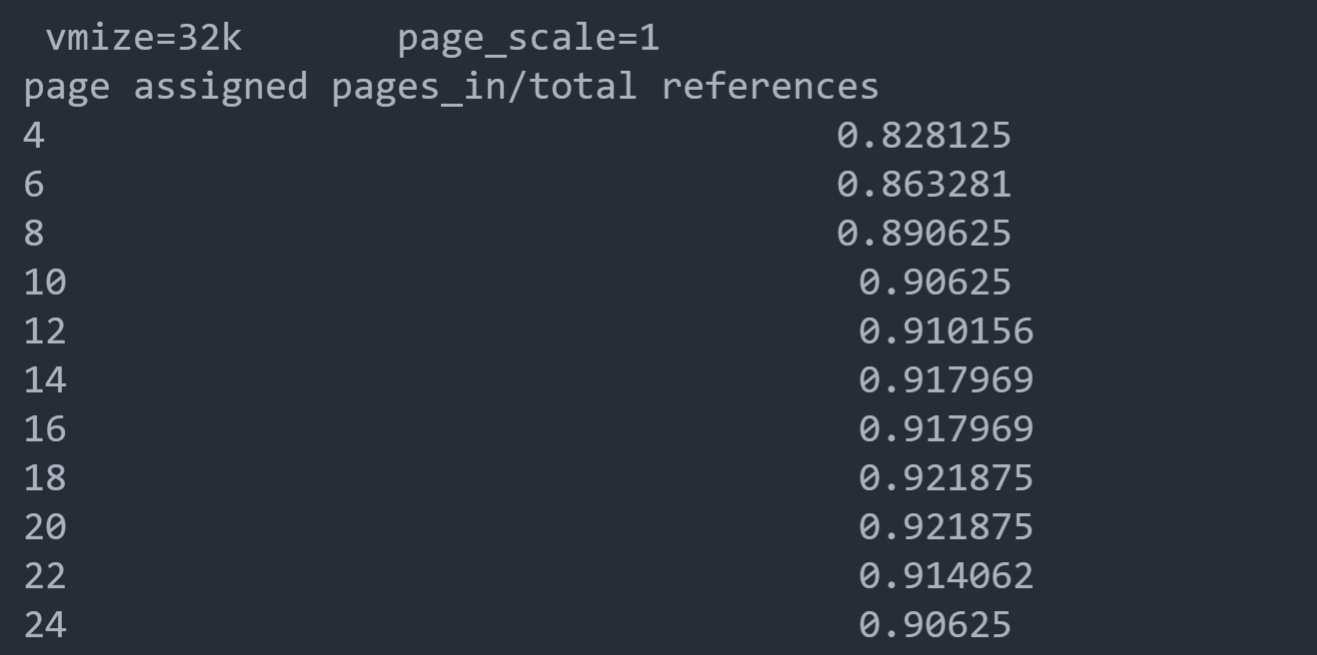


**结果：**

**OPT算法的运行结果**



**Lru算法的运行结果**



小结： OPT效率 > LRU效率

**思考题：**

（1）设计一个界地址存储管理的模拟系统，模拟界地址方式下存储区的分配和回收过程。

**提示：**必须设置一个内存分配表，按照分配表中有关信息实施存储区的分配，并不断根据存储区的分配和回收修改该表。算法有首次匹配法，循环首次匹配法和最佳匹配法等。可用各种方法的比较来充实实习内容。可使用碎片收集和复盖等技术。

答：界地址存储管理：

1）内存分配表 和 空闲区域表 array of(addr,size)

2）基址寄存器：保存运行进程的起始地址

限长寄存器：保存运行进程长度

3）地址映射：物理地址=逻辑地址+首址寄存器内容

分配算法：首次适应、循环首次适应、最佳适应、最坏适应、快速适应

回收：1）回收区与插入点的前一个空闲分区F1相邻接

2）回收区与插入嗲的后一个空闲分区F2相邻接

3）回收区同时与插入点的前、后两个空闲分区邻接

4）回收区与插入点F1、F2都不邻接

碎片收集：可采用将主存中的作业进行移动，碎片合并，使他们领接。把原来分散的多个小分区合并拼接成一个大分区。

覆盖技术： 将较大程序装入较小进程空间。只把全局代码和数据静态装入内存，其他部分动态装入。后装入的成分重复使用先装入成分所使用的存储区，以及覆盖先装入的成分。

**体会：**

本次实验模拟了存储管理系统中，请求页式管理系统。通过模拟虚拟地址流，请求页号等等流程。我更加清晰的认识到页表机制实现的过程。同时这次实验也帮助我更好的细化和区分了普通的页表调入进程方式和请求页式之间的区别。此外，这次实验让我深刻的意识到，高效的内存管理，配合和调度是一个极为复杂的过程。我以后也需要更加深入的学习相关内容，从而更好的掌握计算机操作系统相关的知识。

**源代码：**

