1. **需求分析：**

实现一个简单的存储缓冲管理器，缓冲区采用的替换策略为LRU算法，利用符合Zipf分布的500000个数据进行页面访问的模拟。页面访问结束后将缓冲区中dirty=1的页面写回磁盘存储器。最终结果输出显示将dirty=1的页面写回磁盘存储器前的磁盘I/O数和命中数hit，以及将dirty=1的页面写回磁盘存储器后的磁盘I/O数和命中数hit。

1. **算法设计：**

(1)建立磁盘管理模拟，类名为DSMgr，详细类设计见类设计图，下同。创建50w个页面，采用基于目录的结构来组织数据库文件，该目录文件命名为data.dbf，将50w个页面的指针写入data.dbf中；

(2)建立LRU队列管理；

(3)建立缓冲区管理；

(4)执行记录驱动文件验证存储缓冲管理其的正确性，算法如下：

#include<iostream>

#include<fstream>

#include"BMgr.h"

#include"DSMgr.h"

using std::ifstream;

using std::cout; using std::endl;

int main() {

BMgr\* bufferMgr=new BMgr();

int dirty = 0, page\_id,frame\_id,count=0;

char c;

int cnt = 0;

ifstream in("data.txt");

if (!in) std::cerr << "open file error" << endl;

while (in >> dirty >>c>>page\_id) {

frame\_id = bufferMgr->fixPage(page\_id);

if (dirty == 1)

bufferMgr->setDirty(frame\_id);

else

bufferMgr->unsetDirty(frame\_id);

}

cout << "before write dirtys:" << endl;

cout << "hit: " << bufferMgr->getHit() << "\n"

<< "IO: " << bufferMgr->getIO() << endl;

bufferMgr->writeDirtys();

cout << "after write dirtys:" << endl;

cout << "hit: " << bufferMgr->getHit() << "\n"

<< "IO: " << bufferMgr->getIO() << endl;

//cout << cnt << endl;

return 0;

}

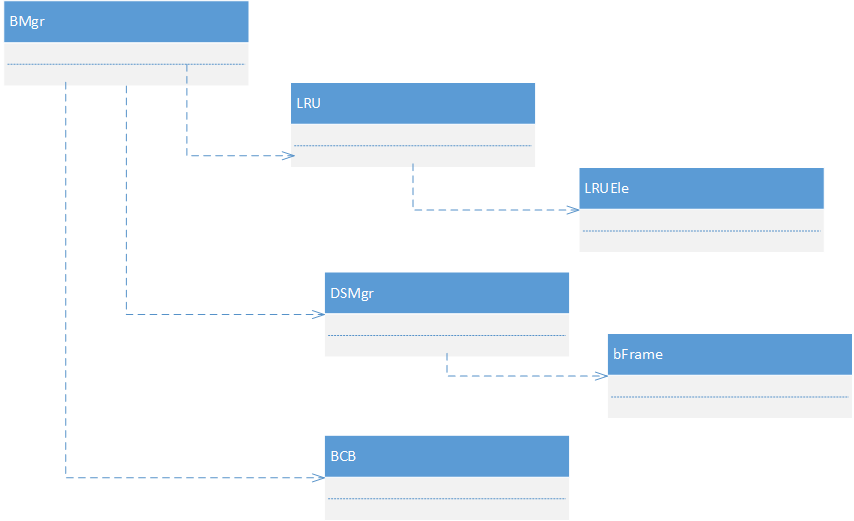
1. **类图设计：**





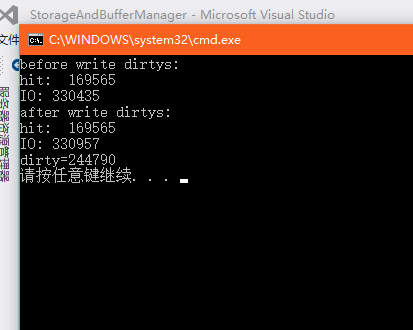
1. **类图关系：**

类图之间的主要为依赖关系，如下

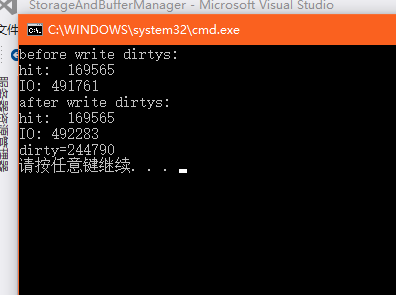


1. **结果及分析：**

代码运行后，结果如下



图一



图二

为验证程序的正确性，先不算上在选择victim时将victim缓冲块写回磁盘的io，那么在将

dirty=1的缓冲块写回磁盘前会有等式：*hit+io=500000*

图一在将脏位写入磁盘前，未计算将victim以及脏位写回的io次数为330435，命中次数hit为169565，hit+io=500000正好与测试数据集大小相同，命中率为33.9%，之后将缓冲区中的dirty=1的页面写入磁盘，此时io=330957，说明程序即将结束缓冲区中dirty=1的页面数n=522个。第二张图片是程序中加上在选择victim时将victim页面写回磁盘的真实io=491761，总io=492283。