内存块的数据结构：起始地址 长度 标记位

作业结构：id 大小 起始地址

管理类：（也是存储管理）

空闲区表，已分配区表 //管理

方法：

构造函数 分配 回收 合并 打印快照 首次适应策略 最佳适应策略 最坏适应策略

main：

输入作业

阈值--由用户指定

创建进程（作业申请内存--根据作业队列来）

结束进程（回收内存）

struct MemoryBlock {
  
 int startAddress;
  
 int length;
  
 int allocated;//0未分配，其他值--进程id
  
};
  
struct Job {
  
 int id;
  
 int size;
  
 int startAddress;
  
};
  
class MemoryManager {
  
private:
  
 std::vector<MemoryBlock> freeBlocks; // 空闲区表
  
 std::vector<MemoryBlock> allocatedBlocks; // 已分配区表
  
  
public:
  
 MemoryManager(int totalMemory) {
  
 // 初始化时，整个内存是一个大的空闲块
  
 freeBlocks.emplaceBack(0, totalMemory);
  
 }
  
  
 ~MemoryManager() {
  
 // 析构函数，可以用于清理资源
  
 }
  
  
 void allocate(int id, int length);//选择何种策略
  
 void deallocate(int id);//回收--根据分配区表来的
  
 void mergeFreeBlocks();
  
 void print();
  
  
 // 分配策略
  
 int firstFitAllocate(int id, int size);//返回空白块首地址
  
 int bestFitAllocate(int id, int size);
  
 int worstFitAllocate(int id, int size);
  
};
  
int main() {
  
 MemoryManager manager(5000); // 假设有5000个单位的内存
  
 /\*
  
 输入作业--id 大小
  
 阈值--由用户指定
  
 while(1){
  
 作业申请内存--根据作业队列来
  
 回收内存--输入id,回收哪个进程
  
 退出
  
 }
  
 \*/
  
 return 0;
  
}