**一个B+树仿真器程序的系统设计报告**

## B+树的概念

B+树的定义：

有M颗子树的结点中含有M个关键字

所有的叶子结点中包含了全部关键字的信息，及指向含这些关键字记录的指针，且叶子结点本身依关键字的大小自小而大顺序链接

所有的非终端结点可以看成是索引部分，结点中仅含有其子树(根结点)中的最大或最小关键字

## B+树的逻辑结构

一个B+树结点具有以下结构：

关键字数量 KeyNum

关键字向量 Key[M]

指向父结点指针 \*Parent

指向孩子结点指针 \*Child[M]

叶子结点判断标志 Isleaf

指向下一记录的指针 \*next

B+树结构：

叶结点头指针 \*data

根结点指针 \*root

## 用一个大小为 40Bytes 的内存单元模拟一个外部存储块，规定关键字大小为 4Bytes，地址大小为 4Bytes，记录信息数据大小为 8Bytes。确定上述B+树的M值（用于内部M-路搜索树）和L值（用于每个叶子块存储的记录数目）

B+树的M值为5，L值也为5

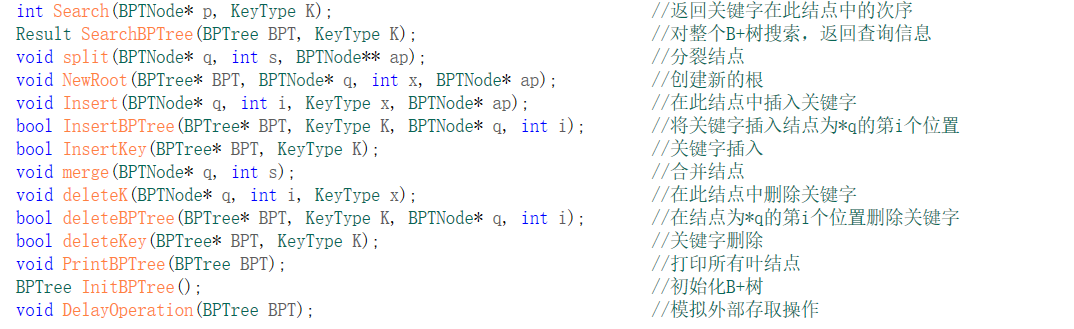
之所以确定L值等于M值，有利于后续理解和减少一定代码量和方便理解

## 存储上述B+树的数据结构设计 （程序设计语言描述）

代码在附件

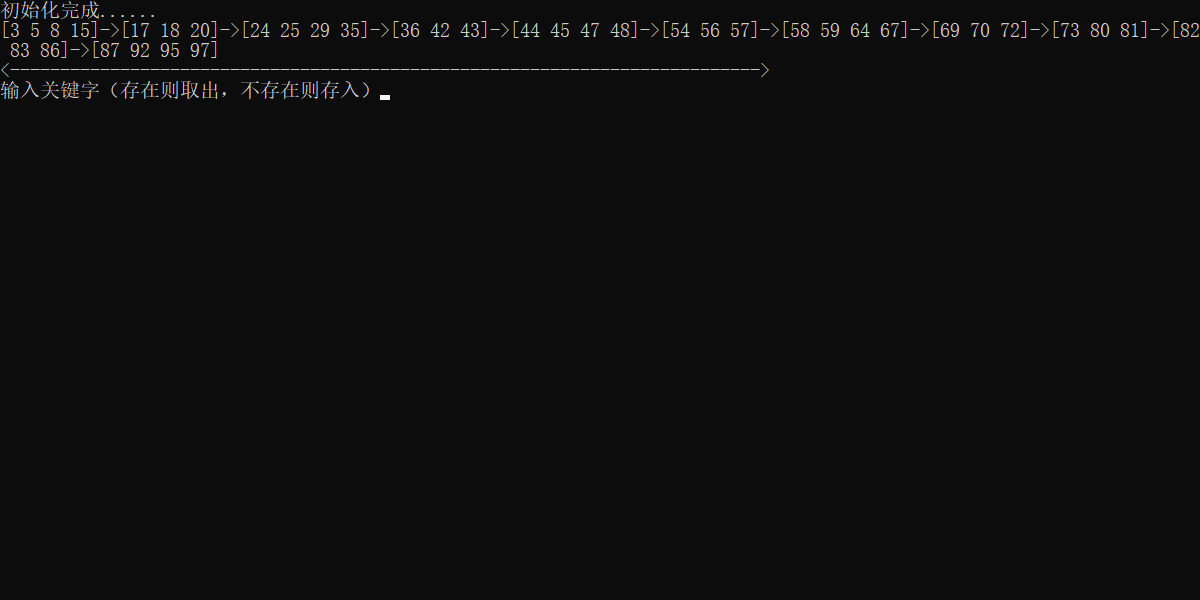
## B+树用于记录查找、插入和删除的算法流程，包含一个自行设计的20ms延时器模拟一次外部存取的时间延迟

## 源代码各个主模块命名清单（不需要代码清单）

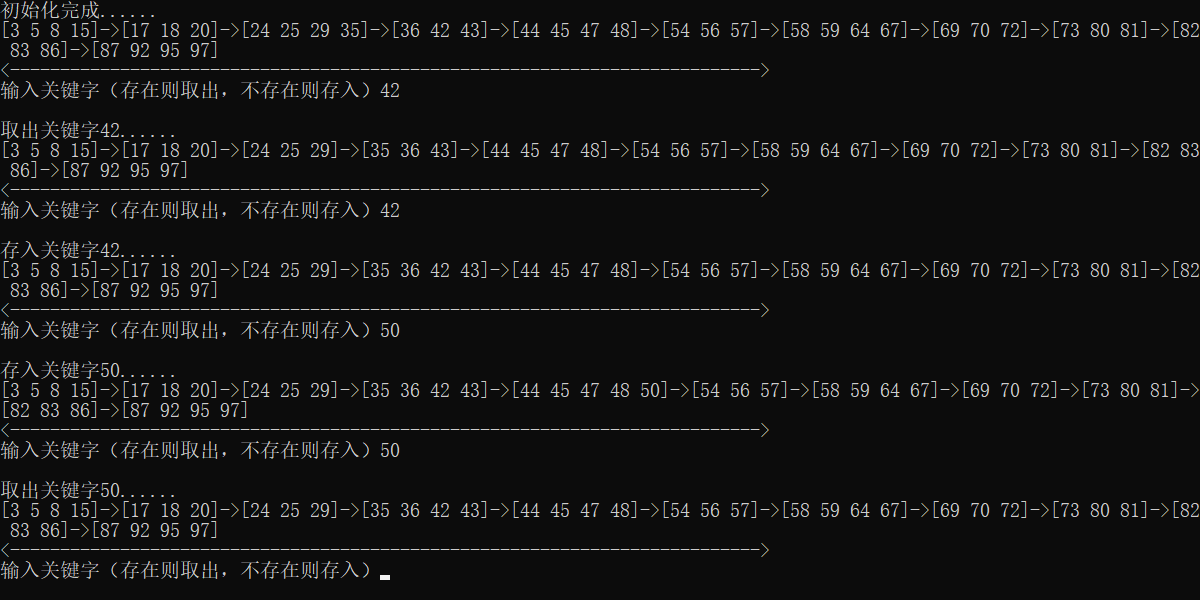


## 测试用例设计（初始化至少包含50个记录数据）

使用了rand()函数产生50个随机数(记录)并插入到B+树中，但产生的随机数并不独立，最终插入到树中的数(记录)不足50个。



## 运行结果分析，包括外部存取延时统计



## Binary Search Trees, AVL Trees 和 B+Trees 在处理相同的记录信息数据量时各自的表现和计算效率的比较