

**课 程 设计 报 告**

**题目： 楼盘查询系统**

**专业班级：计算机科学与技术1607**

**学 号： U201614700**

**姓 名： 王亚宁**

**指导教师： 周时阳**

**报告日期： 2017年9月5日**

**计算机科学与技术学院**

目录

# 1任务书

**题目 楼盘查询系统**

⑴主要内容

建立楼盘楼盘信息系统，提供创建、编辑和综合查询等基本业务管理和服务。

⑵任务要求

收集与阅读相关文献资料，确定系统目标与范围，分析系统需求，确定系统功能；设计系统方案，完成系统实现；提交《课程设计报告》。

⑶参考文献

[1]曹计昌,卢萍,李开. C语言程序设计,北京：科学出版社,2013

[2]李开,卢萍,曹计昌. C语言实验与课程设计,北京：科学出版社,2011

[3][张引](http://www.amazon.cn/mn/searchApp?searchWord=%E5%BC%A0%E5%BC%95). C程序设计基础课程设计, 杭州: 浙江大学出版社,2007

[4][黄明](http://www.amazon.cn/mn/searchApp?searchWord=%E9%BB%84%E6%98%8E),[梁旭](http://www.amazon.cn/mn/searchApp?searchWord=%E6%A2%81%E6%97%AD),[万洪莉](http://www.amazon.cn/mn/searchApp?searchWord=%E4%B8%87%E6%B4%AA%E8%8E%89). C语言课程设计,北京: 电子工业出版社,2006

## 相关提示

⑴系统功能

系统主要是提供城市楼盘的快速查询、检索和统计。应该具有①按城区、按户型和按价格等查询信息；②组合条件检索楼盘；③按城区、按户型和按价格等统计房屋信息。

⑵系统数据

系统主要涉及“楼盘”、“楼栋”和“房屋”3类信息。 “楼盘”信息由楼盘编号、楼盘名称、楼盘地址、开发商和物业公司等数据项目组成，“楼栋”信息是由楼栋编号、单元和楼层等数据项组成，“房屋”信息由房屋编号、户型、面积、公摊率和价格等数据项目组成。

在磁盘上，建议采用3个文件分别存储“楼盘”、“楼栋”和“房屋”的数据。“楼栋”信息中增加楼盘编号，“房屋”信息中增加楼盘编号和楼栋编号。

# 2需求分析

## 2.1 需求分析

本题目属于系统设计题目，需要系统考虑题目要求以及真正需求是什么。楼盘查询系统，主要是给购房人士提供查询功能，给开发商和售楼人员提供查询楼盘信息，修改添加删除信息的功能，所以本系统主要的功能就是实现多功能查询，修改，删除，统计的功能。

对于购房人士，主要考虑的就是楼盘的位置信息，房屋面积，价格等信息，开发商需要的就是位置信息，周围楼盘分布，交通等信息，统计主要就是根据房屋所属地区，价格，面积进行分类，给开发商提供较全面的统计信息。

对于图形界面部分，用C语言结合GTK+也是能够写出图形界面的，不过不能够跨平台，在windows下面简直要不要太难用，使用Qt的话我感觉有作弊嫌疑，毕竟是为C++设计的框架，根据课设要求只能使用C语言，为了能够跨平台，程序中不能使用平台相关的文件等，最终还是选择使用简易的字符菜单。

3系统设计

## 3.1数据结构的设计

数据结构采用较为灵活的链表，使用指针和编号将这几个房屋组成部分连接起来。数据结构使用的链表分3个，一个是房屋的链表，还有楼栋的链表，楼盘的链表。

房屋所使用的结构体至少需要知道此房屋的面积信息，价格信息，户型等，为了便于构建链表，还要加入房屋所在的楼盘编号和楼栋编号，所以房屋的结构体大致如下：

**struct** House {  
 **int** house\_id; *// 编号* **char** model[100]; *// 户型* **float** area; *// 面积* **float** prize; *// 价格* **struct** House \*next; *// 指向下一个房屋的指针* **int** building\_id; *// 所在楼栋编号* **int** buildings\_id; *// 所在楼盘编号*}

对于楼栋的结构体，主要存储信息就是指向此楼栋里所有的房屋的头指针，还有此楼栋所处的楼盘编号，因此楼盘编号必须是唯一的，而楼栋和房屋的编号只需要在自己对应的楼盘和楼栋里唯一即可，因此楼栋的结构体如下：

**struct** OneBuilding {  
 **int** one\_building\_id; *// 楼栋编号* **int** floorCount; *// 楼栋层数* House \*head; *// 指向其拥有的房屋的头结点* **struct** OneBuilding \*next; *// 指向同一楼盘下一个楼栋的指针* **int** buildings\_id; *// 所在楼盘编号*}

楼盘结构体就需要存放较多信息，其中有楼盘编号，此楼盘中楼栋的头指针，楼盘的位置信息，楼盘的开发商等，具体如下：

**struct** Buildings {  
 **int** buildings\_id; *// 楼盘定义* **char** name[100]; *// 楼盘名称* **int** location\_id; *// 所在区域编号* **char** location[200]; *// 具体位置* **char** developers[100]; *// 开发商* OneBuilding \*head; *// 指向其所有楼栋的指针* **struct** Buildings \*next; *// 只指向下一个楼盘的指针*}

## 3.2 系统函数的设计

根据需求分析得出此程序需要有的功能主要有查询，添加，删除和统计，所以主要函数设定为图3-1：

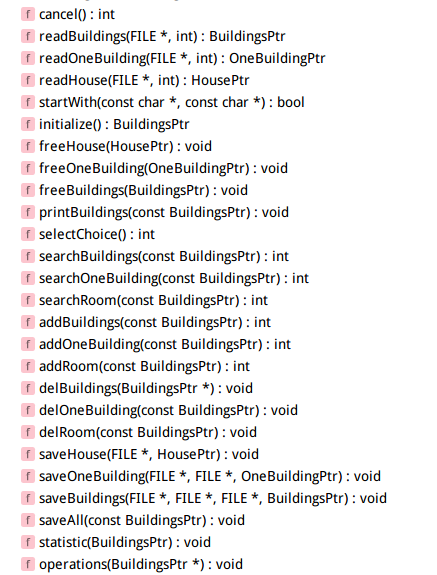


图3-1

其中的函数具体使用场景将一一解释。

cancel函数就是在程序运行中询问是否返回到主菜单，主要用途用于取消操作。readBuildings函数传入一个文件指针，还有一个楼盘编号，用于分配空间并读入相关信息，返回读入信息了的结构体指针，下面两个read\*函数为相同作用。startWith用于判断字符串是否以另一个字符串开头，在搜索房屋比较户型的时候会用到返回值为bool，表示是否以其开头。initialize函数用于初始化程序中使用到的指针，读入相关文件并存储在链表中，返回值为指向楼盘链表头的Buildings指针。下面3个free\*函数为程序结束时或者删除对应楼盘时释放内存使用，没有返回值。printBuildings函数用于检测是否读入成功，无返回值。selectChoice用于选择要进行的操作编号，返回值即为编号。下面几个search\*函数用于查询满足需求的楼盘及房屋，返回值为int，只是为了在函数中返回主函数方便，返回值无用可丢弃。add\*函数和del\*函数用于添加和删除楼盘信息。save\*函数用于保存变更了的更改，参数为打开的文件指针和要保存的结构体指针。statistic函数用于统计信息。operations函数为中转站，接受选择的操作编号跳转到指定位置并执行，无特殊作用。

其中的查询和添加删除操作均为遍历链表比较全部数据实现，可能效率会有点低，并且操作不当可能会导致内存泄露，保存的函数也是通过遍历实现，所以总体效率还是比较低的。

## 3.3 系统结构

系统的结构示意图如下

# **4系统测试**

## 4.1 查询测试

### 4.1.1 楼盘查询测试

楼盘查询，操作比较简单，只需要输入需要查询的楼盘的区域即可，输出一定数目程序会暂停输出等待指令是继续还是停止，测试如下：

首先测试一下正常输入，如图4-1

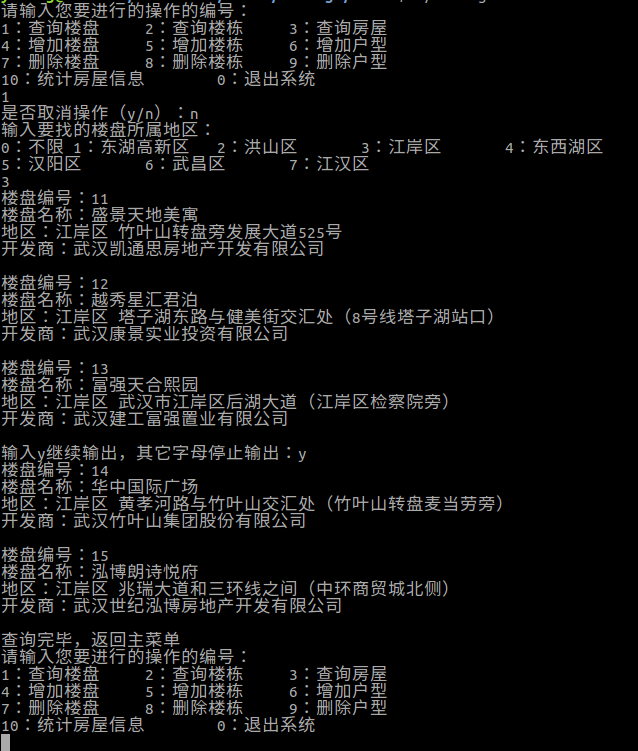


图4-1

输出无异常，而且可以看输出到3个楼盘的时候程序会等待输入指令，输入y就会继续输出，其他的字符会返回到主菜单，方便查询者阅读信息。再来测试输入一个错误的数字，如图4-2：

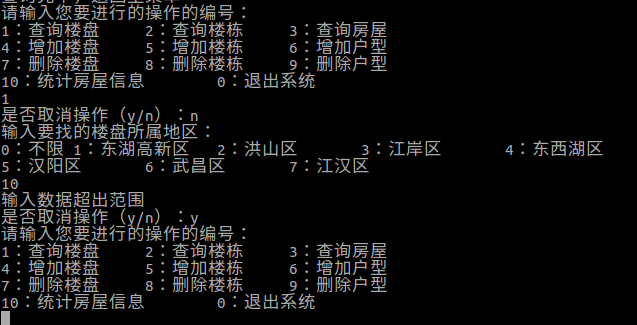


图4-2

系统检测到数字不合理可以拒绝访问，表明系统有一定的容错能力。

## 4.2 楼栋查询测试

楼栋查询主要涉及到楼栋所处的区域和楼栋高度等信息，查询的时候会要求输入楼栋所在区域和希望的楼栋层数，系统会输出所有符合要求的结果，和查询楼盘一样，在数据比较多的时候会进行分屏输出，便于阅读。查询例子见图4-3：

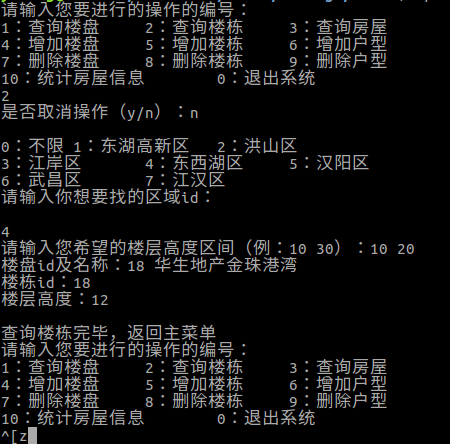


图4-3

由于数据量比较小，输出的时候没有进行分屏，系统输出了符合条件的楼栋及其所属于的楼盘编号和名称。

## 4.3 房屋查询测试

房屋查询就是题目中所指出的多条件组合查询，其中涉及到房屋的地区，价格，户型等因素，组合查询的一个测试例子见图4-4：

能够看到程序能够正确的读取用户的需求，并且按照需求去遍历得出符合要求的房屋，在数据比较多的时候也可以分屏输出方便阅读，较完美的达到了预设目的。

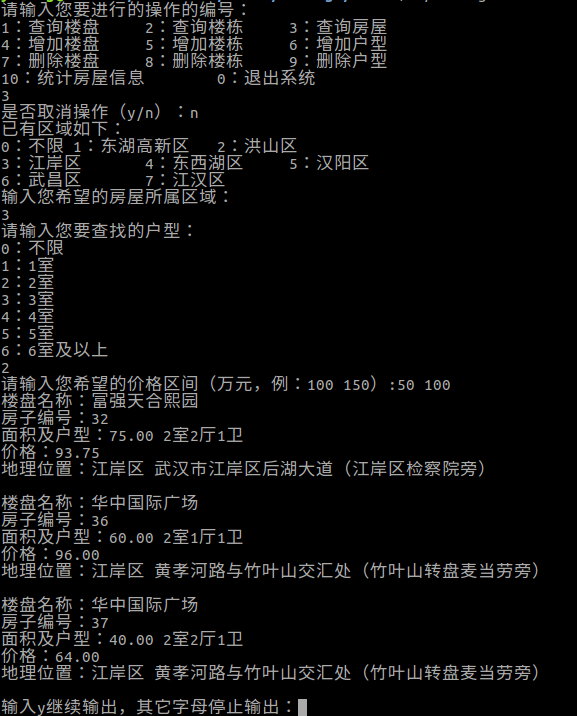


图4-4

## 4.4 插入楼盘测试

插入楼盘的时候程序会自动设置插入楼盘的编号，需要输入的只有楼盘的名称，地域信息，开发商信息等，插入的一个例子如下图4-5：

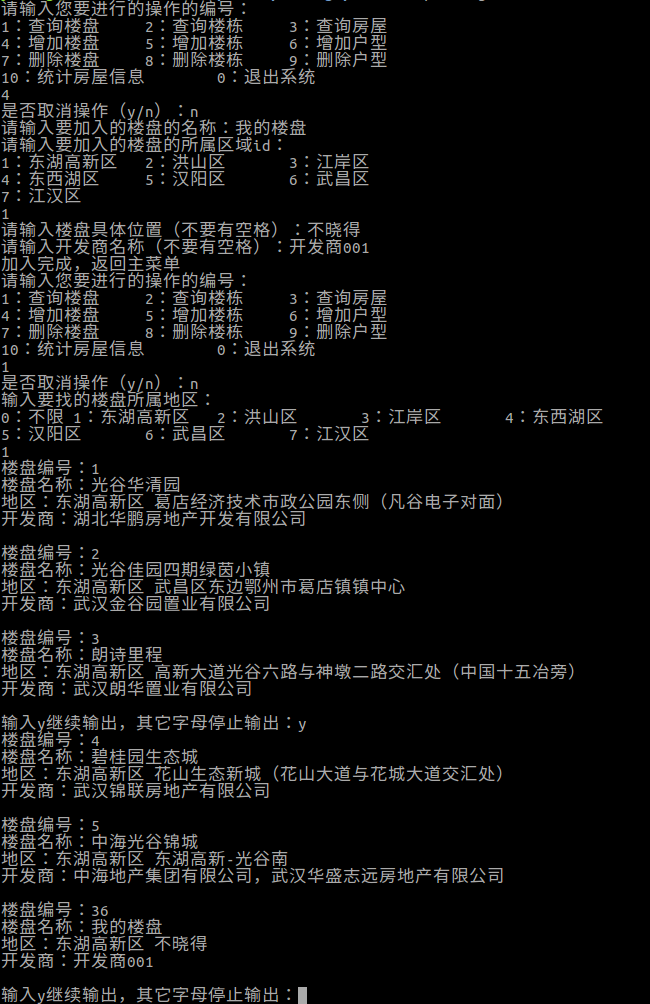


图4-5

# 

根据查询结果可知插入成功，此实验数据中楼盘最大编号为35，新增的楼盘编号为36，保证在链表中不会产生重复的楼盘编号，确保插入楼栋和房屋时不会出差错。

## 4.5 插入楼栋测试

插入楼栋和插入楼盘的操作类似，程序会在判断楼栋的编号，为新的楼栋自动分配编号，防止输入出错，插入测试在楼盘编号为01的楼盘，插入过程如下图4-6：

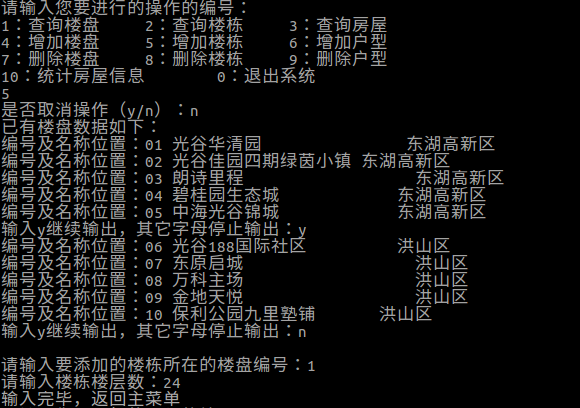


图4-6

插入过程也可以分屏输出，便于要加入的楼盘编号。插入后可以检测是否插入成功，检测的过程如下图4-7所示，可知插入成功，楼栋编号分配的是11。

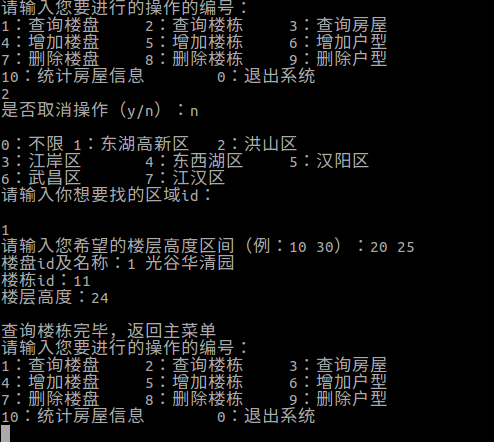


图4-7

## 4.6 插入房屋测试

插入房屋测试与上述的两个插入测试相同，只不过需要找到对应的楼盘和楼、栋，输入信息时系统会根据已有的房屋自动更新编号，这样就能够保证同一楼栋下不会出现重复的编号，具体插入房屋的测试见图4-8：

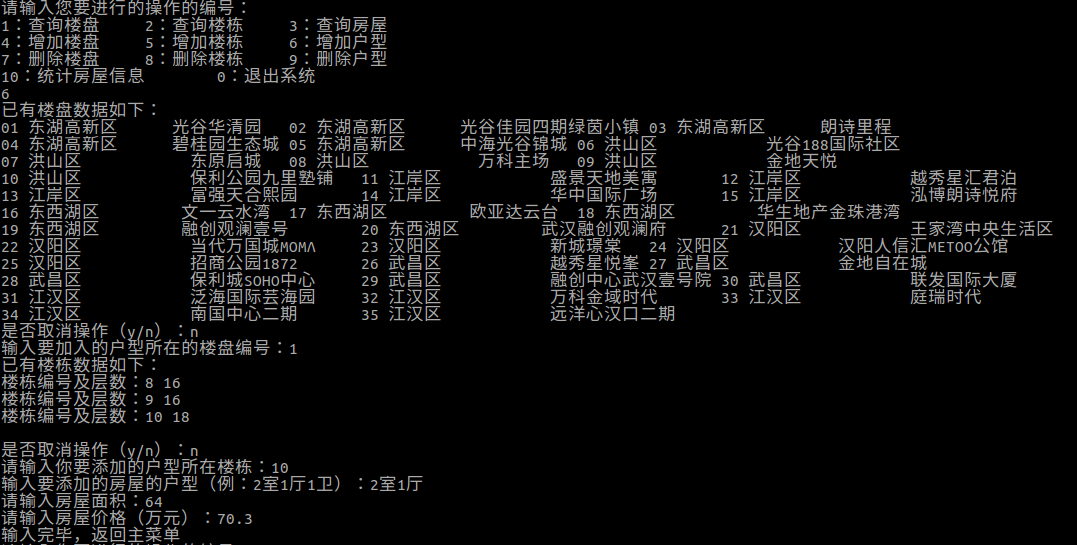


图4-8

插入过后检测是否插入成功可根据查询结果查看，见图4-9，第一个就是刚才插入的房屋，可知插入成功：

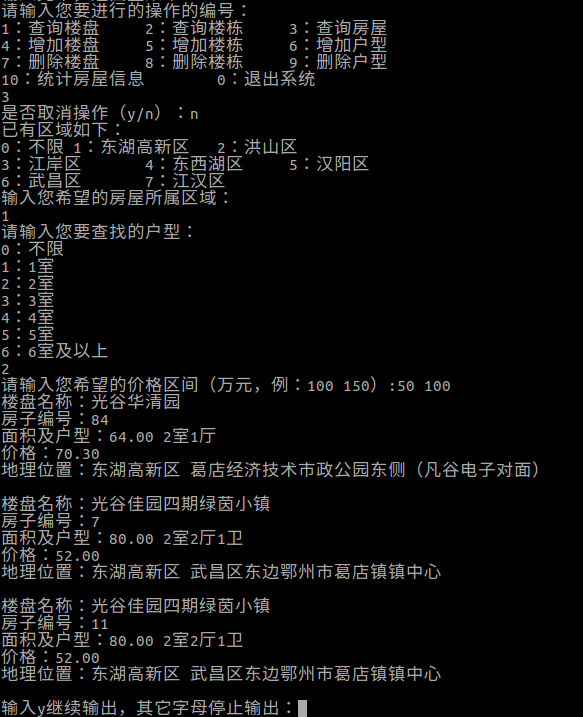


图4-9

## 4.7删除楼盘测试

现在测试删除函数是否可用，删除主要分两种情况，一是删除节点为根节点的，一个是删除节点为普通节点的，普通节点直接讲上一个节点的尾指针指向要删除节点的下一个即可，头结点直接修改头指针，释放对应空间即可。

具体两种删除的操作见图4-10，图4-12：

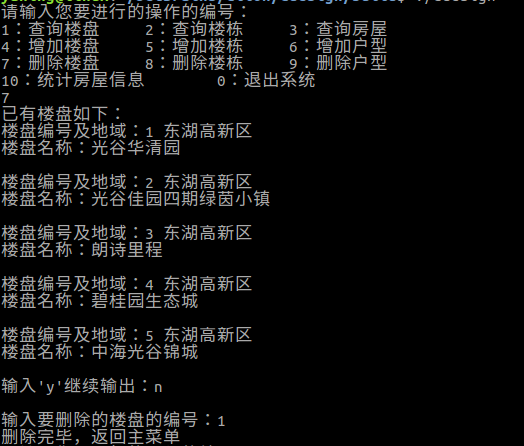


图4-10

删除后的检测，可知删除成功，首指针已改变：

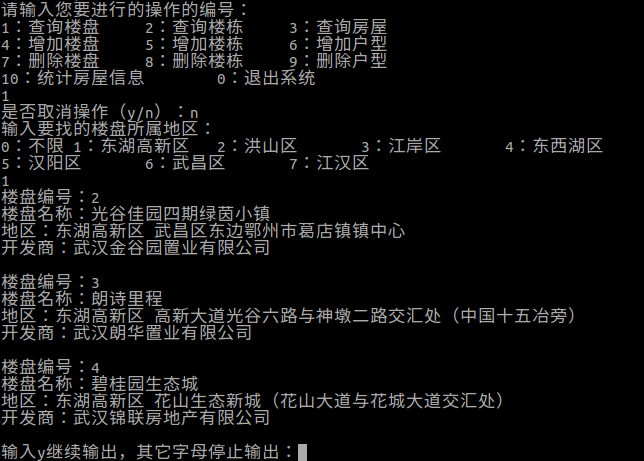


图4-11

由图4-12可见删除3号楼盘也是没有问题，

# 

图4-12

删除后的检测见图4-13，删除成功：

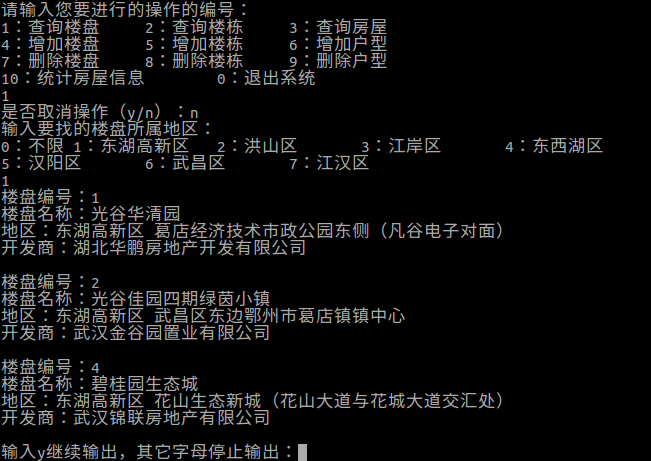


图4-13

## 4.8 删除楼栋测试

删除楼栋也分删除首个楼栋和普通楼栋，这里不再一一测试，直接删除首指针所指楼栋，见图4-14:

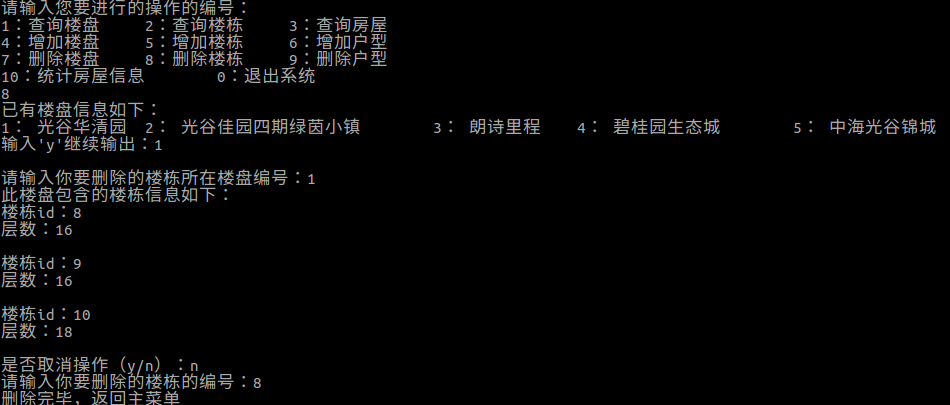


图4-14

删除后的检测如图4-15：

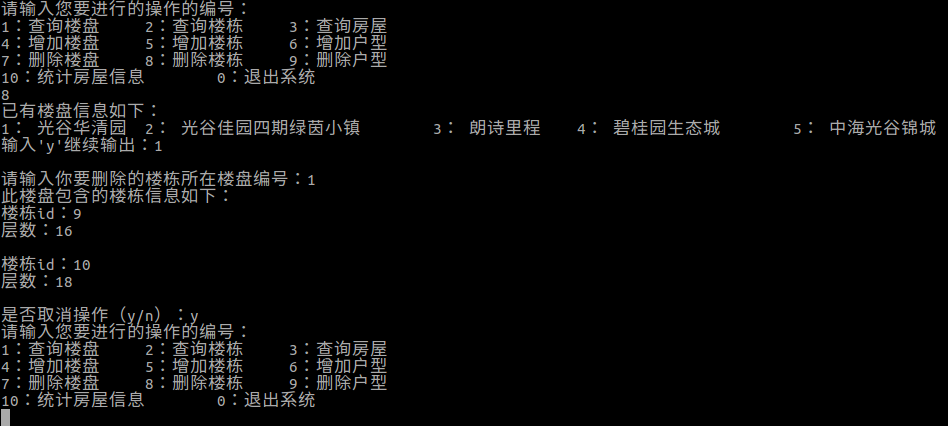


图4-15

可见编号为8的楼栋已被删除。删除楼栋部分工作正常。

## 4.9 删除房屋测试

删除房屋，和前面两个没有很大区别，只不过需要把楼盘和楼栋都指定，比较麻烦，具体删除的例子见图4-16和图4-17：

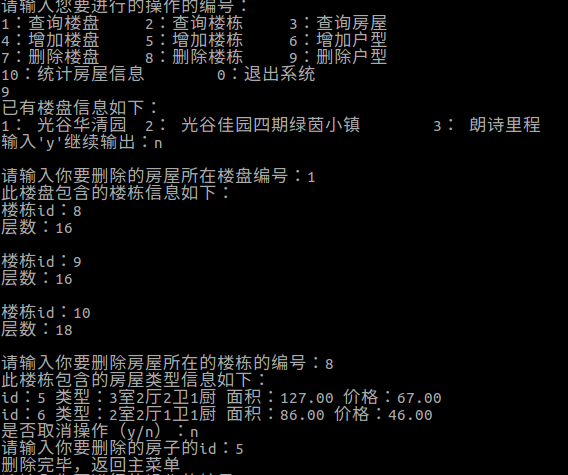


图4-16

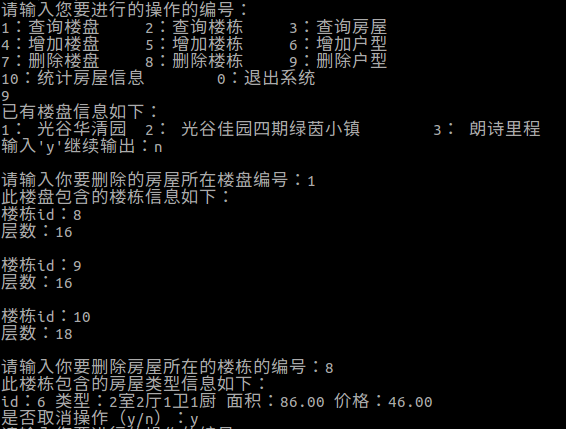


图4-17

删除后进行检测可以看到删除成功，删除房屋的函数工作正常。

## 4.10 统计函数测试

统计函数只需要将数据统计输出即可，功能不是太多，初始时运行函数得到结果见图4-18:

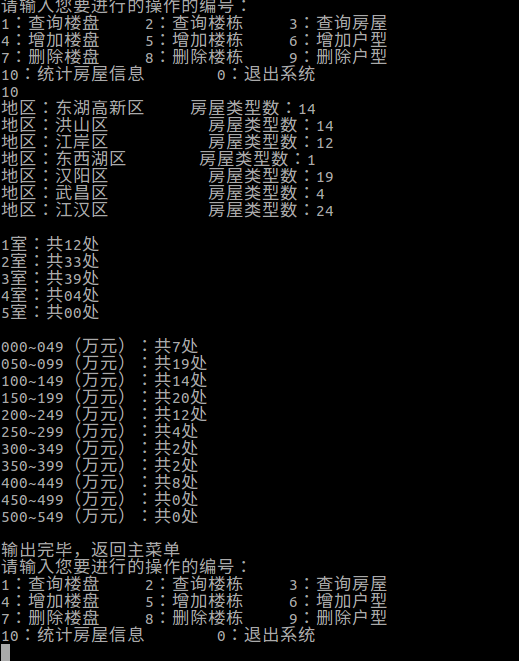


图4-18

这时进行删除操作，例如删除编号为01的楼盘，再次统计数据，结果见图4-19和图4-20：

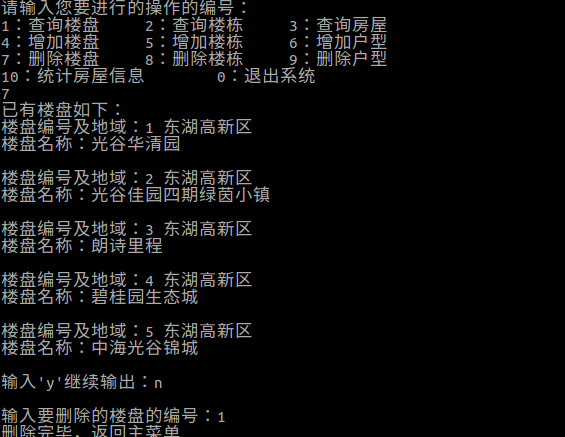


图4-19

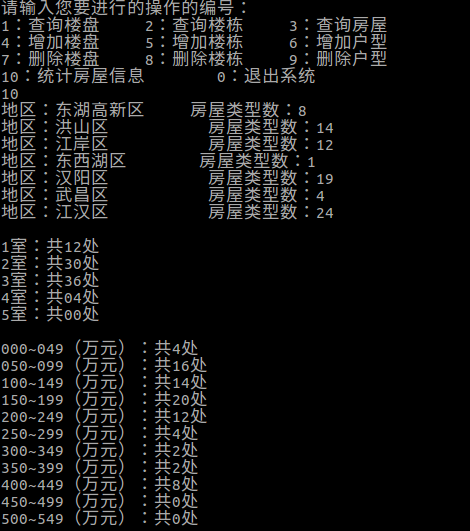


图4-20

数据统计发生了变化，表明删除成功，统计功能也正常运行。

## 4.11 保存更改测试

对数据进行修改后写入到磁盘中才能持久化存储，对于存盘的函数测试如下，先读取文件，添加一个楼盘后写入文件，查看结果即可知道保存函数是否运行正常。先查看buildings.dat中存储的最后一个楼盘数据，见图4-21：



图4-21

由图得出此时未进行操作时最后一个楼盘的编号为35，现在进行添加楼盘操作，操作见图4-22：

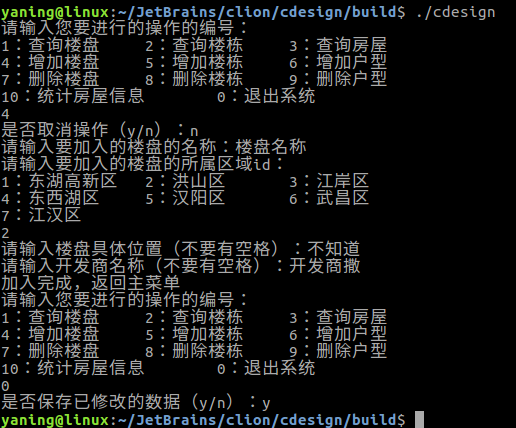


图4-22

操作过后的数据见图4-23，可知保存新建的楼盘成功。

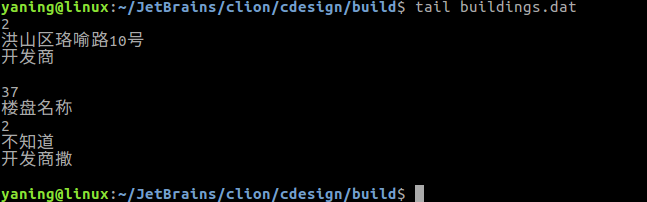


图4-23

# 5 系统实现

系统的具体实现见源代码文件main.c和Functions.h，编译可使用gcc，附带CMakeLists.txt可使用cmake和make进行编译，生成可执行文件为cdesign。