华中科技大学计算机科学与技术学院

C语言课程设计报告

题目：房屋出租信息管理系统

专　　业：计算机科学与技术专业

班　　级：CS1601

学　　号：U201614532

姓　　名：吕鹏泽

成　　绩：

指导教师：甘早斌

完成日期：2017年 09月 07日

目 录

一、系统需求分析

房屋出租信息管理系统用于房屋出租信息的管理，主要内容包括客房分类信息、客房基本信息、租客基本信息，以帮助管理人员及时了解各项基本情况，提高工作效率。

房屋出租信息管理系统要求实现以下几方面的基本功能：

1.基本信息的录入、修改和删除功能

房屋出租信息管理系统的基本信息主要包括以下三类：

(1) 房间分类信息：客房编号、客房类别。

(2) 房间基本信息：客房编号、电话号码、客房类别、客房面积、每月租金、是否有阳台、是否有客人入住

(3) 客人租房信息：身份证号、客人姓名、入住客房编号、入住时间、退房时间、入住月数、押金、应缴费用、实缴费用。

系统应实现以上三种基础信息的录入、修改和删除功能。在信息录入时，系统应提供尽量快捷和方便的录入方式，避免重复操作，降低数据冗余度；同事，还应提供自动数据校验功能，满足数据的正确性、合理性、有效性和依赖性的要求、尽量避免录入无用数据或非法数据。

具体实现为对客房分类信息， 客房基本信息， 及客人租房信息等三方面基本信息的数据维护功能，每个分类再次分为子类。如下：

1）客房分类信息维护

包括对客房分类信息的录入、修改和删除等功能。

2）客房基本信息维护

括对客房基本信息的录入、修改和删除等功能。

3）客人租房信息维护

包括对客人个人信息、入住信息、退房信息、押金等信息的录入、修改和删除等功能。

2.基本信息的查询功能

系统应实现对以上三种基础数据信息的查询功能，提供按多种条件分别进行查询的方式，具体分为三个子类：

1) 客房分类信息查询功能

以客房类别为条件来查找并显示满足条件的客房分类信息。例如，查找并显示客房类别为双人间的客房分类信息。

2) 客房基本信息查询功能

① 以客房编号为条件，查找并显示满足条件的客房基本信息。例如，查找并显示客房编号为“303”的客房基本信息。

② 以客房类别和每月租金为条件，查找并显示满足条件的客房基本信息。例如，查找并显示客房类别为单人间且每月租金 800 元的所有客房基本信息。

3) 客人租房信息查询功能

① 以客人身份证号为条件，查找并显示满足条件的客人租房信息。例如，查找并显示身份证号为“230101199200100 24”的客人所有租房信息。

② 以客人的姓或名及入住时间范围为条件，查找并显示满足条件的所有客人租房信息。例如，查找并显示姓张且在 2012 年 5 月 11 日至 2012 年 5 月 20 日之间入住的所有客人入住信息。

3.数据统计功能

在以上三种基础数据信息的基础上，提供多方面的数据统计功能，具体包括：

1) 统计每种类别的客房总数、入住数、未住数。例如：

统计时间：2015 年 2 月 14 日 15 时 20 分

客房类别	客房总数	已入住数	未入住数
单人间	3	3	0
双人间	2	1	1
三人间	3	3	0
合计	8	7	1

表格 1 客房入住情况统计表

2) 按类统计本年度各类客房的营业额。

年度：2016 年 计量单位：元人民币

客房编号	客房类别	营业额	入住月数	入住率
合计				

表格 2 本年度各类客房营业额 统计表

3) 输入年份，统计概念所有客房的营业额、入住月数、入住率。

年份：2017

月份	单人间	双人间	三人间
1			
…			
12			
合计			

表格 3 所有客房的营业额、入住月数、入住率统计表

4) 列出历年来本房屋出租月数最多的 10 个客人租房信息，按累计租房月数降序排序后输出。（身份证号、姓名、累计租房月数、应缴住宿费总额、实缴住宿费总额）。

身份证号	姓名	累计住宿月数	应缴费用总额	实缴费用总额

表格 4 累计最多入住客人信息统计表

5) 统计客房入住情况。

4.基础数据

基于以上数据以及功能的分析，部分基础数据做以下说明。

1) 客房分类信息

客房基本信息包括房间编号和客房类型

中文字段名	类型及长	举
客房类别	char	‘D’ 双人间，’S’ 单间，T 三个间，F 四人间
最多入住人数	int	2
客房套数	int	3
客房未住套数	int	2

表格 5 客房分类信息

2) 客房基本信息

客房基本信息包括：客房编号、电话号码、客房类别、客房面积、每月租金、是否有阳台、是否有客人入住。

中文字段名	类型及长	举
客房编号	char[4]	“302” 表示 3 楼的 2 号客房
电话号码	char[5]	“8302” 客房内部电话为 8+客房号
客房类别	char	‘D’ 双人间
客房面积	float	10.0 平方米
每月租金	float	20.0 元
是否有阳台	Char	1 表示有阳台，0 表示没有阳台
是否有客人入	char	‘y’ 已有客人入住 ‘n’ 未住

表格 6 客房基本信息

3) 客人基本信息

客人基本信息包括：身份证号、客人姓名、入住客房编号、入住时间、退房时间、入住月数、押金、应缴费用、实缴费用。

中文字段	类型及长	举
身份证号	char[20]	“230101198505050005”
客人姓名	char[20]	“张三”
入住客房编号	char[4]	“302” 三楼 2 号客房
入住时间	char[18]	“2015/03/05-13:00”
退房时间	char[18]	“” 空串表示在住
入住月数	float	0 表示在住
押金	float	
应缴费用	float	
实缴费用	float	

表格 7 客人基本信息

押金为客房一个月的月租金。

二、总体设计

1、功能结构设计

房屋出租信息管理系统由五大功能模块组成：文件模块，维护模块，查询模块，统计模块，帮助模块。如下图 1 所示：

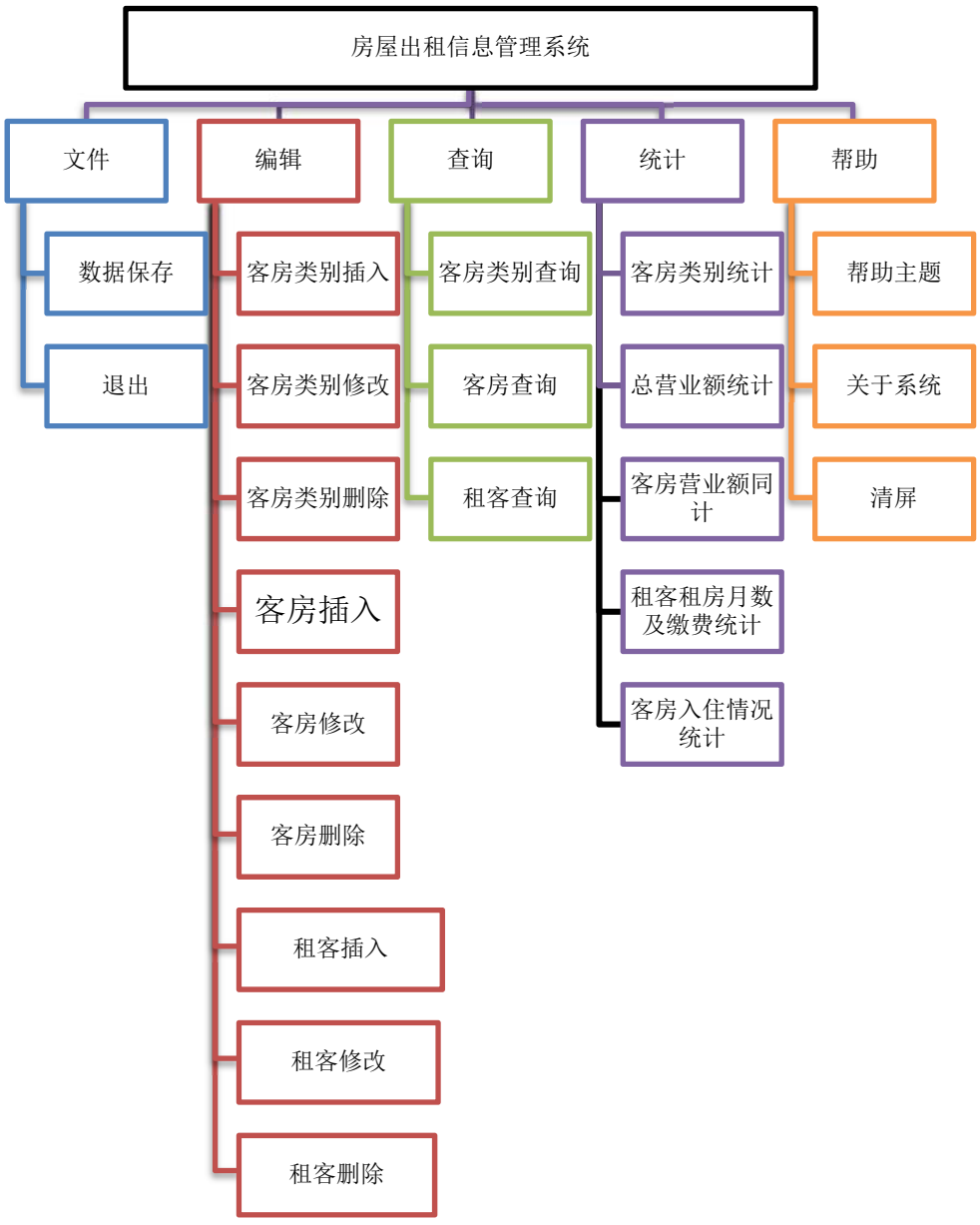


图 2.1.1 系统功能模块结构图

1、文件模块包括两个子模块：数据保存，退出；

2、维护模块包括九个子模块：客房类别插入，客房类别修改，客房类别删除，

客房插入，客房修改，客房删除，

租客插入，租客修改，租客删除；

3、查询模块包括三个子模块：客房类别查询，客房查询，租客查询；

4、统计模块包括五个子模块：客房类别统计，总营业额统计，客房营业额统计，

租客租房月数及缴费统计，客房入住情况统计；

5、帮助模块包括三个子模块：帮助主题，关于系统，清屏。

三、数据结构设计

1、客房分类信息数据结构(CLASS_NODE)

```
typedef struct class_node {
    char type;              /**< 客房类别*/
    int pnum;               /**< 最多入住人数*/
    int rnum;               /**< 客房套数*/
    int emptyr;             /**< 客房未住套数*/
    struct class_node *next;/**指向下一节点*/
    struct room_node *rnext;/**指向客房基本信息节点*/
} CLASS_NODE;
```

中文字段名	类型及长度	举例
客房类别	char	‘D’ 双人间，’S’ 单间，T 三个间，F 四人间
最多入住人数	int	2
客房套数	int	3
客房未住套数	int	2

2、客房基本信息数据结构(ROOM_NODE)

```
typedef struct room_node {
    char room_id[4];        /**< 客房编号*/
    char tel[5];            /**< 电话号码,8+客房号*/
    char type;              /**< 客房类别*/
    float area;             /**< 客房面积*/
    float deposit;          /**< 每月租金*/
    char balcony;           /**< 是否有阳台，1 有，0 无*/
    char in;                /**< y 有客人，n 无客人*/
    struct ROOM_NODE *rnext;  /**< 指向租客基本信息支链的指针*/
    struct room_node *next;   /**< 指向下一结点的指针*/
} ROOM_NODE;
```

中文字段名	类型及长	举
客房编号	char[4]	“302” 表示 3 楼的 2 号客房
电话号码	char[5]	“8302” 客房内部电话为 8+客房号
客房类别	char	‘D’ 双人间
客房面积	float	10.0 平方米
每月租金	float	20.0 元

是否有阳台	Char	1 表示有阳台，0 表示没有阳台
是否有客人入	char	‘y’ 已有客人入住 ‘n’ 未住

3、客人租房信息数据结构(RENTER_NODE)

```
typedef struct renter_node {
    char id_card[20];          /**< 身份证号*/
    char name[20];             /**< 姓名*/
    char room_id[4];           /**< 入住客房编号*/
    char date_in[18];          /**< 入住时间*/
    char date_out[18];         /**< 退房时间*/
    float months;              /**< 入住月数*/
    float deposit;             /**< 押金*/
    float shouldpay;           /**< 应缴费用*/
    float realpay;             /**< 实缴费用*/
    struct renter_node *next;  /**< 指向下一结点的指针*/
} RENTER_NODE;
```

中文字段	类型及长	举
身份证号	char[20]	“230101198505050005”
客人姓名	char[20]	“张三”
入住客房编号	char[4]	“302” 三楼 2 号客房
入住时间	char[18]	“2015/03/05-13:00”
退房时间	char[18]	“” 空串表示在住
入住月数	float	0 表示在住
押金	float	
应缴费用	float	
实缴费用	float	

4、客人租房月数及缴费统计数据结构(RENTER_INFO)

```
typedef struct renter_info{
    char id_card[20];/**身份证号*/
    char name[20];/**姓名*/
    float allmonths;/**入住总月数*/
    float shouldpay;/**应付总金额*/
    float realpay;/**实付总金额*/
    struct renter_info *next;/**指向下一个节点*/
```

}RENTER_INFO;

身份证号	姓名	累计住宿月数	应缴费用总额	实缴费用总额

5、总营业额统计结构(INCOME_NODE)

```
typedef struct income {
    char type;//客房类别
    float sale[12];//记录 1-12 月的收入
    struct income *next;          /**< 指向下一结点的指针*/
} INCOME_NODE;
```

月份	单人间	双人间	三人间
1			
…			
12			
合计			

6、客房营业额统计结构(ROOM_INFO)

```
typedef struct room_info{
    char room_id[4];//客房编号
    char type;//客房类别
    float income;//客房收入
    float months;//客人在本年入住该客房总月数
    float in_rate;//客房入住率
    struct room_info *next;//下一节点
}ROOM_INFO;
```

客房编号	客房类别	营业额	入住月数	入住率
合计				

7、屏幕窗口信息链数据结构(LAYER_NODE)

```
typedef struct layer_node {
    char LayerNo;              /*弹出窗口层数*/
```

```
    SMALL_RECT  rcArea;   /*弹出窗口区域坐标*/
    CHAR_INFO *pContent;  /*弹出窗口区域字符单元原信息存储缓冲区*/
    char *pScrAtt;        /*弹出窗口区域字符单元原属性值存储缓冲区*/
    struct layer_node *next;   /*指向下一结点的指针*/
} LAYER_NODE;
```

8、标签束信息结构(LABEL_BUNDLE)

```
typedef struct label_bundle {
        char **ppLabel;      /*标签字符串数组首地址*/
        COORD *pLoc;         /*标签定位数组首地址*/
        int num;             /*标签个数*/
} LABEL_BUNDLE;
```

标签束信息包括标签字符串数组内容，标签位置，标签个数。用一个字符型的二重指针变量 ppLabel 指向标签字符串数组内容，用一个 COORD 类型的字符指针变量 pLoc 指向标签串数组输出时的首位置坐标，用整型变量 num 表示标签的个数。

9、热区信息结构(HOT_AREA)及用法

```
typedef struct hot_area {
        SMALL_RECT *pArea;   /*热区定位数组首地址*/
        char *pSort;         /*热区类别(按键、文本框、选项框)数组首地址*/
        char *pTag;          /*热区序号数组首地址*/
        int num;             /*热区个数*/
} HOT_AREA;
```

热区信息包括热区的位置，类别，序号及个数。因此用一个 SMALL_RECT 结构类型的指针变量 pArea 指向热区的定位范围；用字符指针变量 pSort 指向热区的类别类型，其中数字'0'表示按钮型热区，'1'表示文本框热区，'2'表示选项框热区；用字符指针变量 pTag 指向热区的序号，热区编号一般为 1,2,3,4,5……这些自然数表示多个热区的排列顺序，整型变量 num 表示热区个数。

10、三方向识字交叉链表结构

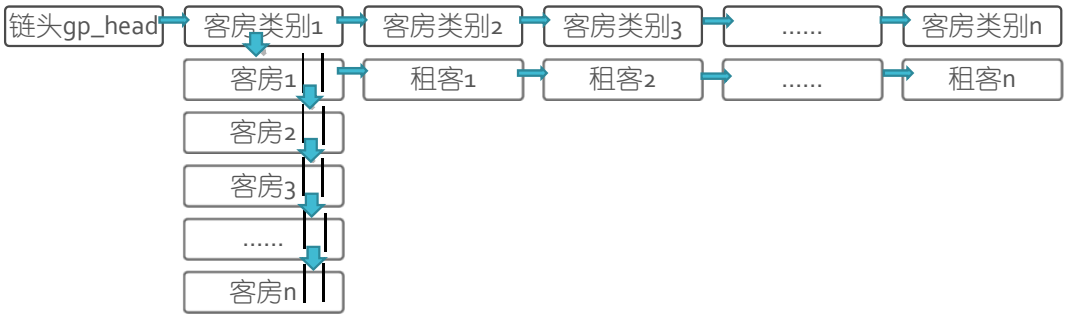
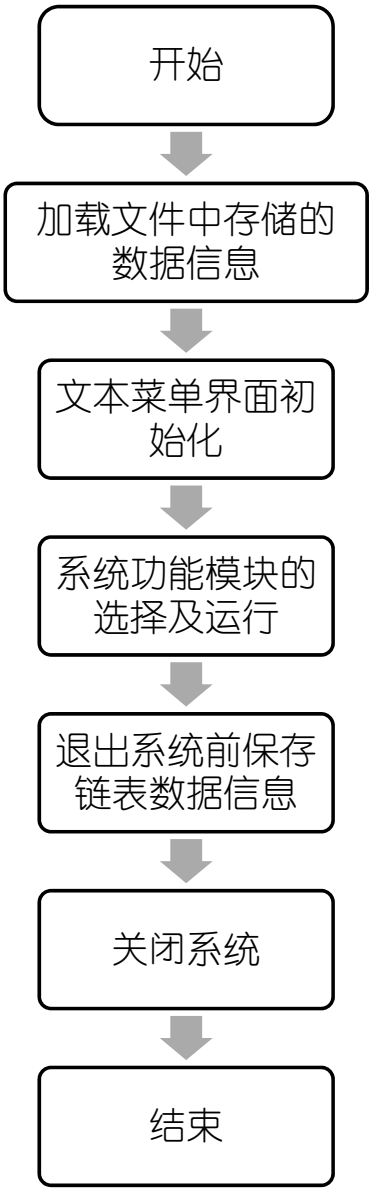


图 3.6.1 三方向十字交叉链表

四、详细设计

1、主程序运行流程

双击 main.exe 运行程序，首先加载数据，然受初始化界面，用户选择相应的功能运行，退出前保存数据。



2、数据加载流程

声明一个外部变量头指针 gp_head，始终指向链头，方便之后部分函数的调用。

1）创建客房分类信息主链

打开客房分类信息数据文件，依次读取每条记录。同时将记录存放在动态创建的结点中，用循环方式建立链表，最后关闭文件。如果客房分类信息数据文件打开失败，则返回相关的错误信息。

2）创建客房基本信息支链

打开客房基本信息数据文件，每次读取一条记录，并将记录存放在动态创建的结点中，在主链上搜索找到与该客房对应的客房分类信息，将结点插入主链结点的客房基本信息支链上，如果没有找到对应的客房分类信息，则释放结点存储区，并从数据文件读取下一条记录，直到读完数据文件中的所有记录，关闭数据文件。

3）创建客人租房信息支链

打开租客信息数据文件，每次读取一条记录，并将记录存放在动态创建的结点中，依次查找每个主链结点上的客房基本信息支链，如果找到该结点对应的客房基本信息，将结点插入客房基本信息结点的租客信息支链上，如果没有找到对应的客房基本信息，则释放结点存储区，并从数据文件读取下一条记录，直到读完数据文件中的所有记录，关闭数据文件。

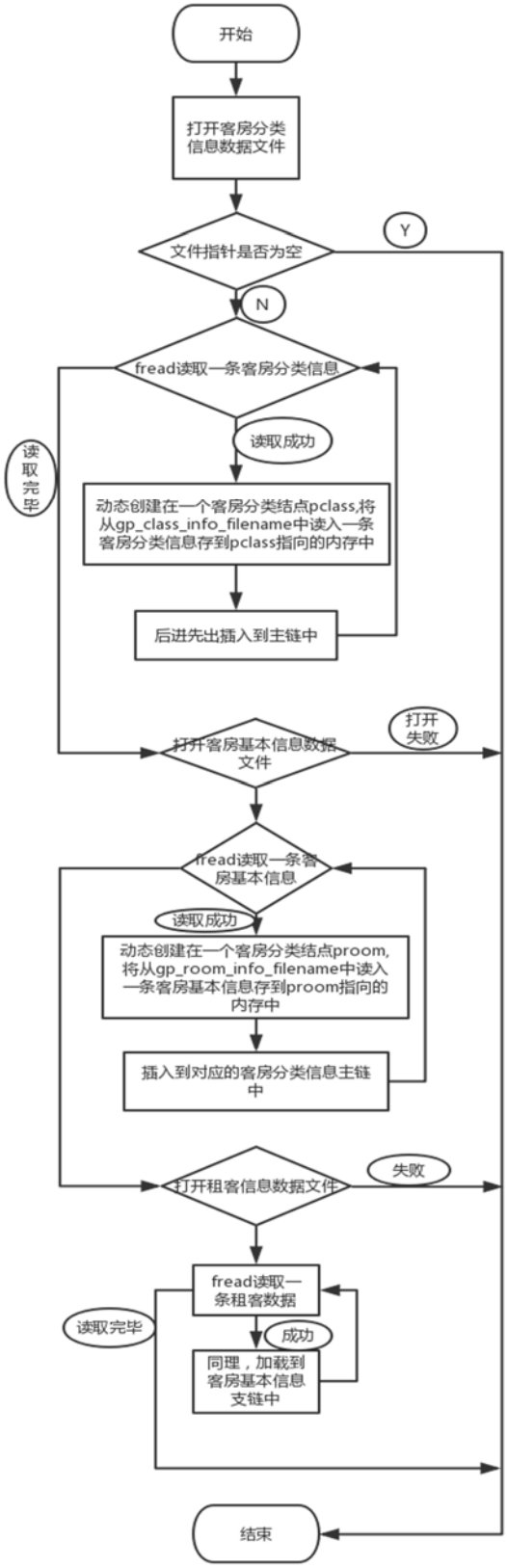


图 4.2.1

3、数据保存流程

●保存客房分类信息：以“wb”方式打开客房分类信息数据文件，循环遍历客房分类信息主结点，将各结点的分类信息用 fwrite 函数写入客房分类信息文件。

●保存客房基本信息：以“wb”方式打开客房基本信息数据文件，二重循环遍历到商品基本信息结点，将各结点的基本信息用 fwrite 函数写入客房基本信息文件。

●保存客人租房信息：以“wb”方式打开客人租房信息数据文件，三重循环遍历到客人租房信息节点，将各结点的基本信息用 fwrite 函数写入客人租房信息文件。

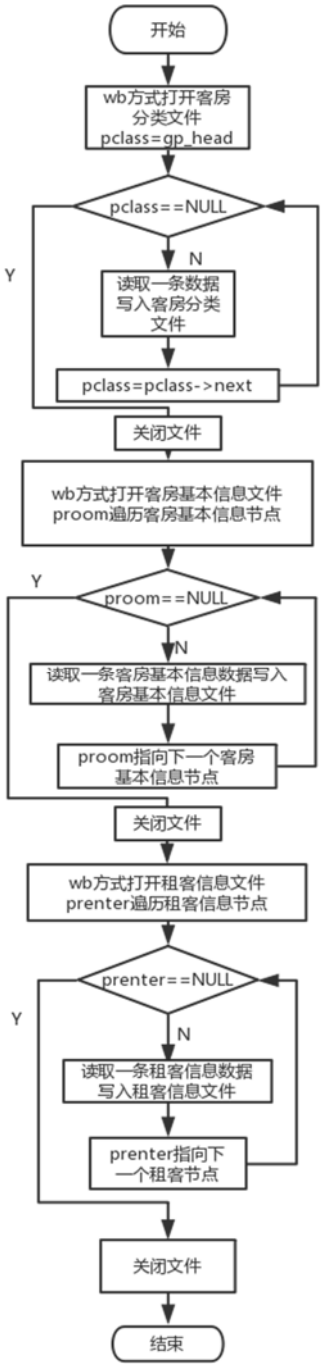


图 4.3.1

4、数据插入流程

三类节点的插入均可表述为：创建一个动态节点，读取用户输入为该节点赋值，确认保存后插入到对应的主链或支链上。

1) 客房基本信息插入

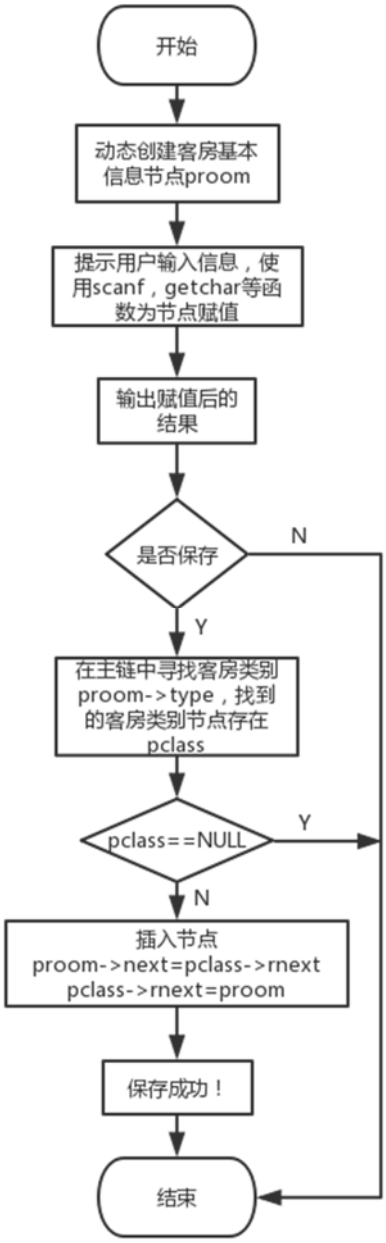


图 4.4.1

2）客房分类信息插入

同客房基本信息插入，即录入信息，确认保存后寻找节点，插入到链中

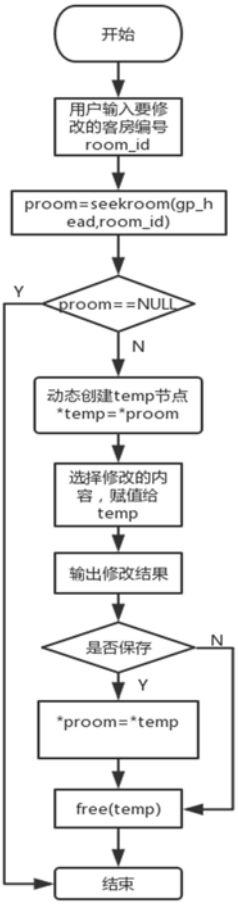
3）租客信息插入

同客房基本信息插入，即录入信息，确认保存后寻找节点，插入到链中

5、数据修改流程

选择要修改的节点类型，动态创建新节点 p1，读取用户输入查找要修改的节点 p2，将 p2 数据复制给 p1，用 switch 确认用户需要修改的数据，使用输入输出函数更改 p1 的数据，确认修改后将 p1 的数据赋值给 p2。

1.修改客房基本信息



2.修改客房分类信息

同修改客房基本信息

3.修改客人租房信息

同修改客房基本信息

6、数据删除流程

1.删除客房分类信息

读取用户输入查找要删除的节点 p1，使用 free()函数二重循环遍历删除 p1 下的一级支链及二级支链节点，修该 p1 的前一个节点的指针指向，最后删除 p1。

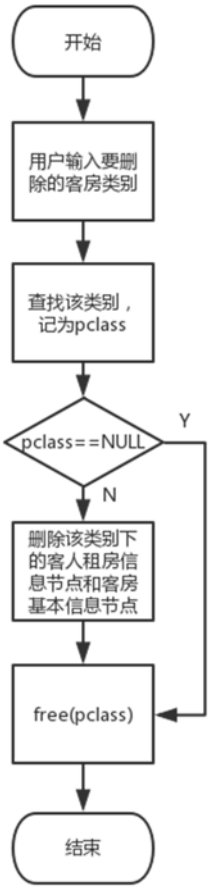


图 4.6.1

2.删除客房基本信息

读取用户输入查找要删除的节点 p1，使用 free()函数一重循环遍历删除 p1 下二级支链节点，修该 p1 的前一个节点的指针指向，最后删除 p1。

3.删除客人租房信息

读取用户输入查找要删除的节点 p1，修该 p1 的前一个节点的指针指向，使用 free()函数一重循环遍历删除 p1。

7、数据查询流程

1.客房分类信息查询

读取用户输入确认查找的客房分类，遍历客房分类信息主链，依次比较客房类型，找到后输出客房分类信息，未找到输出提示。

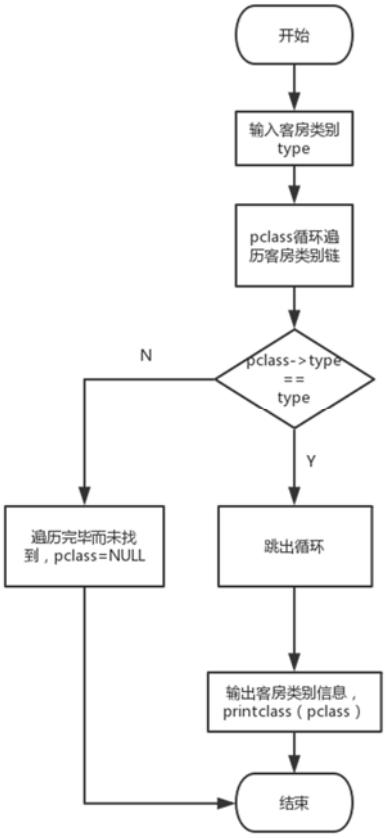


图 4.7.1

2.客房基本信息查询

1）按照客房编号查询

读取用户输入确认查找的客房编号，二重循环遍历客房分类信息主链，依次比较客房编号，找到后输出客房基本信息，未找到输出提示。

2）按照客房分类及租金查询

读取用户输入确认查找的客房分类及租金，二重循环遍历客房分类信息主链，依次比较客房分类及租金，找到输出客房基本信息，未找到输出提示。

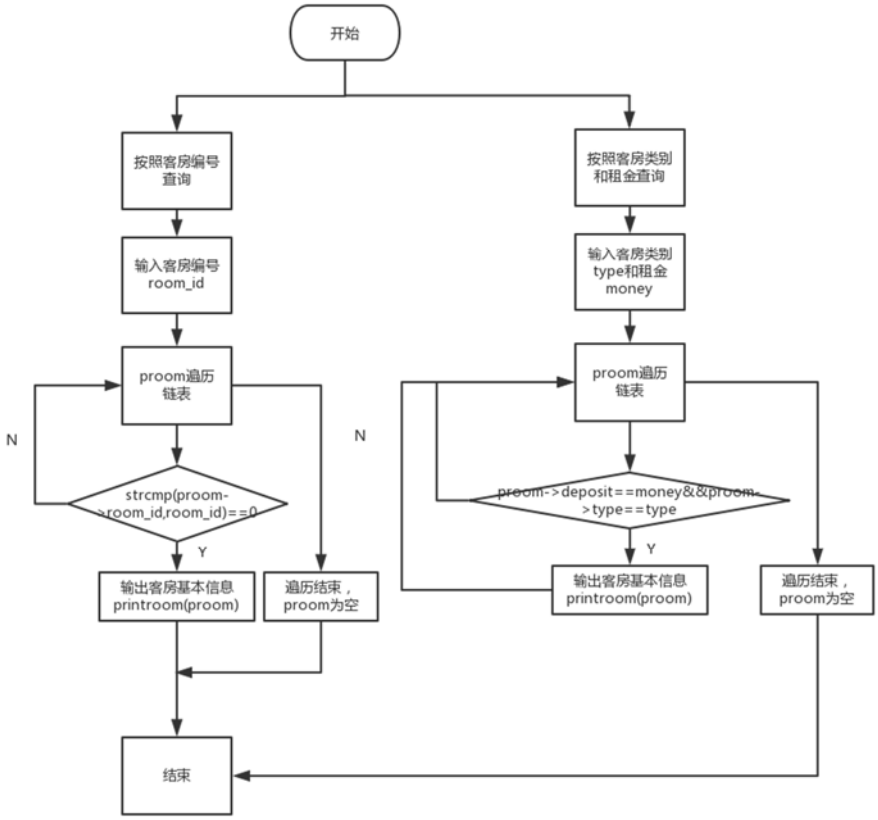


图 4.7.2

3.客人租房信息查询

1）按照租客身份证号查询

读取用户输入确认查找的租客身份证号，三重循环遍历客房分类信息主链，依次比较客身份证号，找到后输出客人租房信息，未找到输出提示。

2）按照租客姓氏及日期查询

读取用户输入确认查找的租客姓氏及日期，三重循环遍历客房分类信息主链，依次比较租客姓氏及日期，找到后输出客人租房信息，未找到输出提示。

3）按照租客名字及日期查询

读取用户输入确认查找的租客名字及日期，三重循环遍历客房分类信息主链，依次比较租客名字及日期，找到后输出客人租房信息，未找到输出提示。

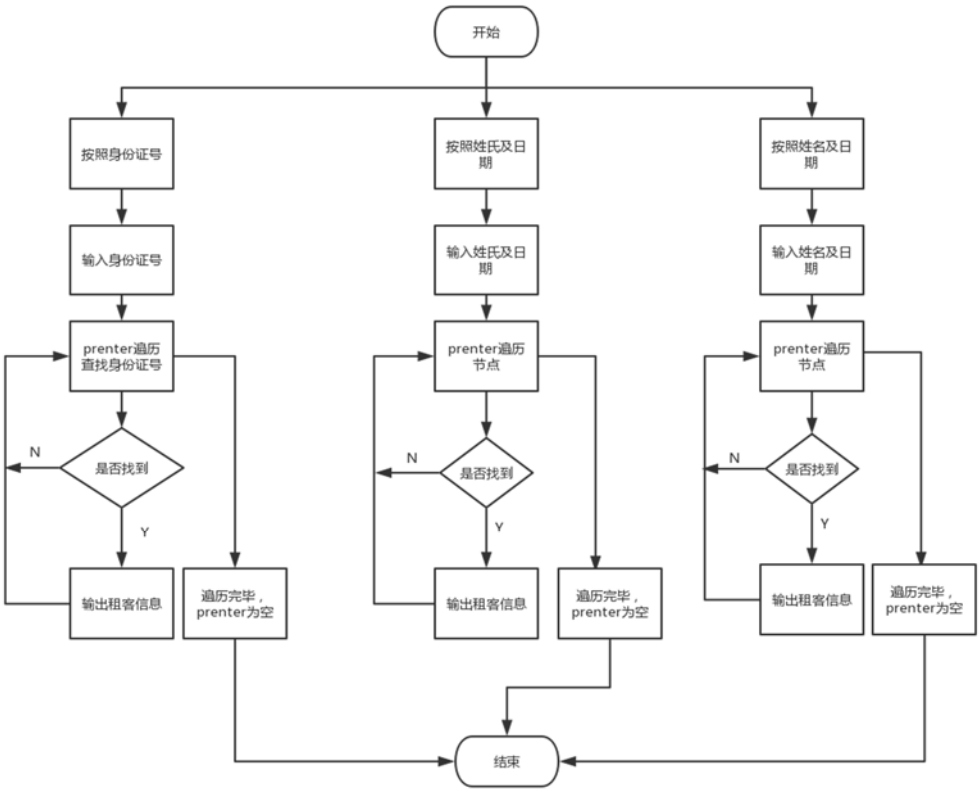


图 4.7.3

8、数据统计流程

1. 客房类别统计

二重循环遍历客房基本信息节点，每遍历一个客房 allroom++，如果该客房下无人入住 emptyroom++，将数据记录在以客房类别创建的统计链表中，遍历完毕后输出统计信息。

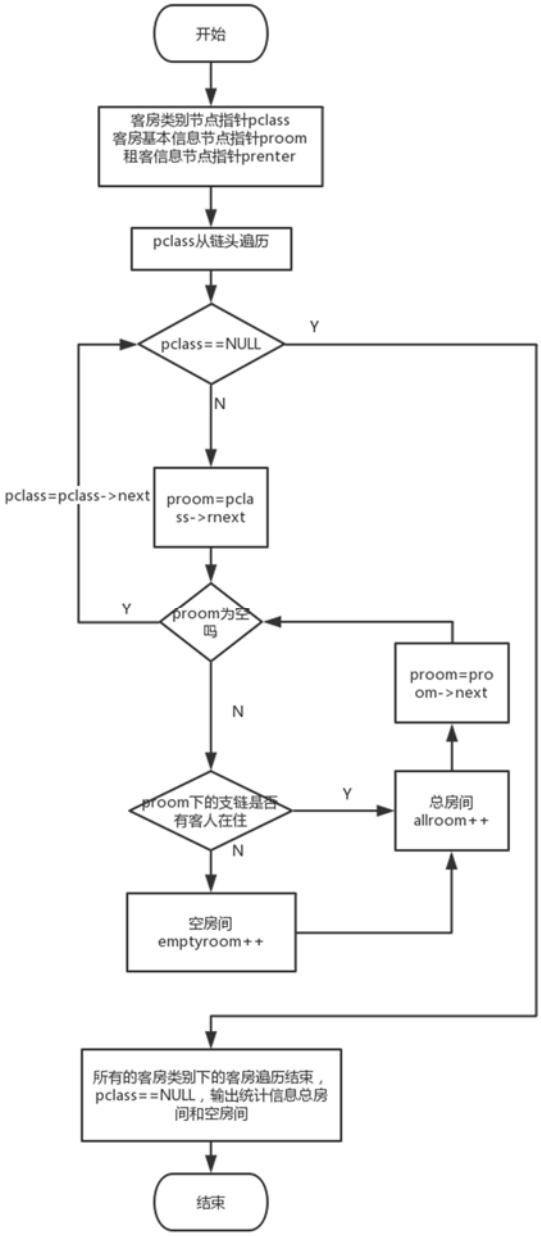


图 4.8.1

2. 总营业额统计

统计链表节点结构是一个 float[12]的数组和 next 指针，其中 i 月的营业额可存储在 float[i-1]中。

以客房类别为节点创建统计链表，三重循环遍历到租客信息节点，判断租客退房日期是否在给定范围内，若在，以租客退房日期缴费为准，将租客的实缴金额加到相应统计节点的月数中，遍历结束后输出统计信息。

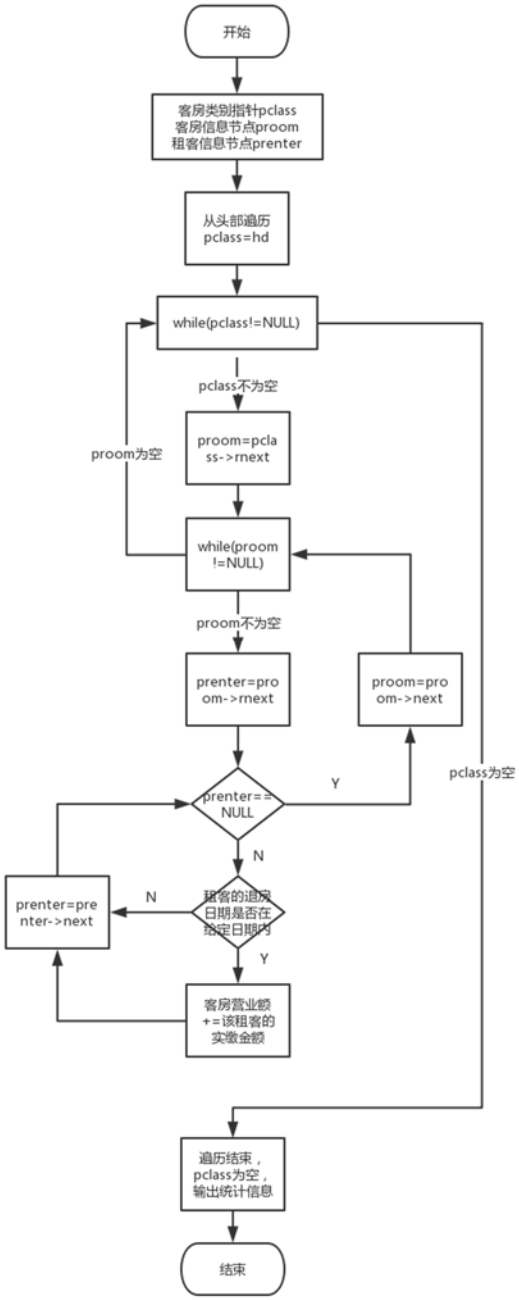


图 4.8.2

3．客房出租信息及营业额统计

以客房编号为准创建统计链表。

三重循环遍历客房分类信息链，判断租客信息节点的租客退房日期是否在给定范围内，若在，将租客的实缴金额的信息加到租客入住的客房编号标识的统计信息链的节点中。统计完毕后输出统计信息。

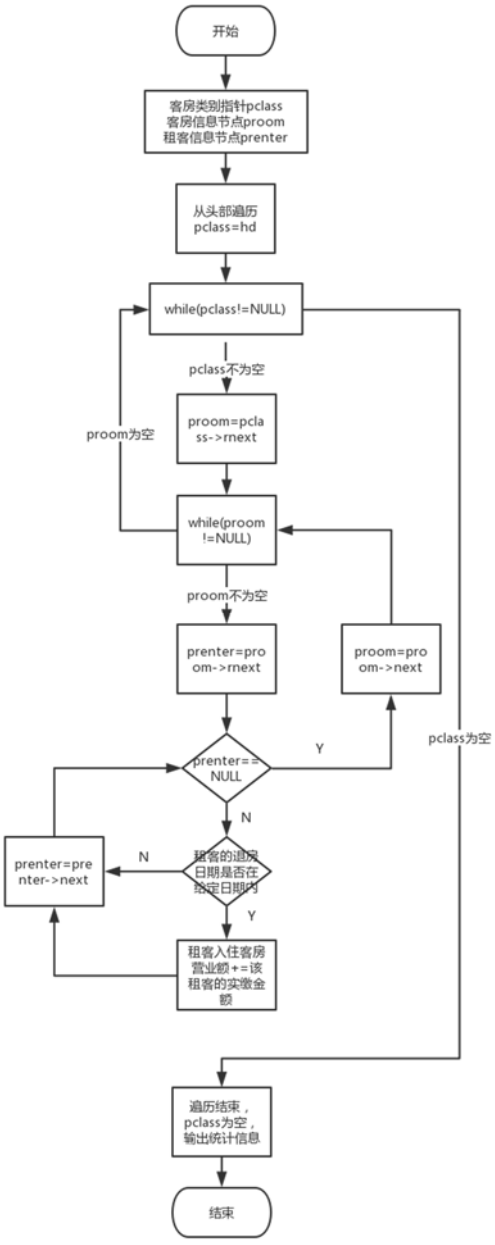


图 4.8.3

4. 租客租房月数及缴费统计

以租客身份证号为准创建统计信息链

三重循环遍历客房分类信息主链，查找租客的身份证号是否创建了统计节点，如果有，将信息加到身份证号标识的统计节点中；如果没有，创建新的统计节点并赋值。统计链创建完毕后使用选择排序法进行排序，交换两节点的数据时采用交换数据域保留指针域的方法，最后从头结点到尾节点依次输出统计数据。

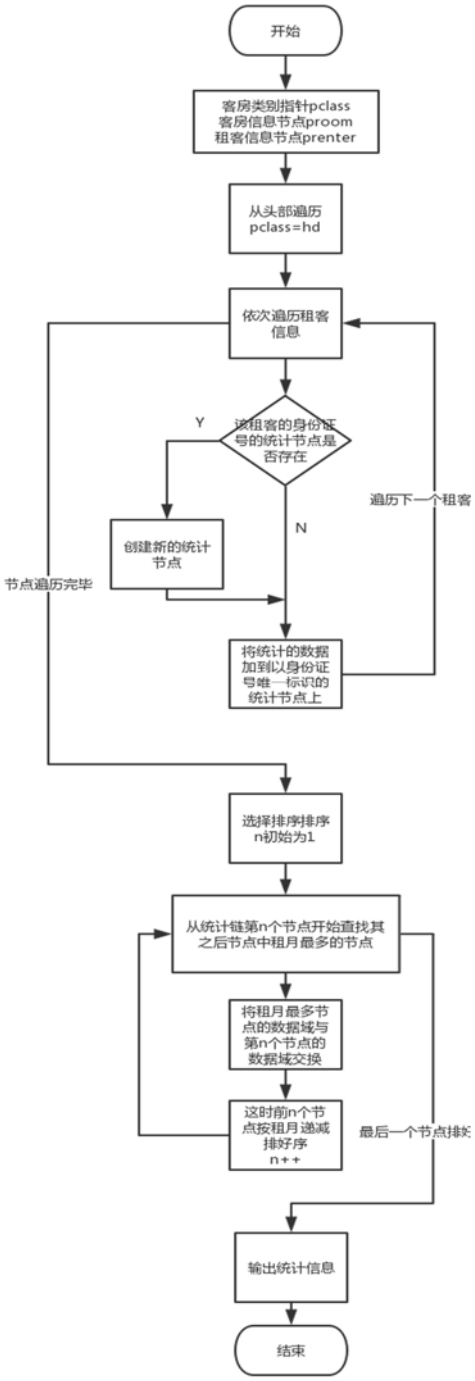


图 4.8.4

5. 客房入住情况统计

二重循环遍历客房分类信息主链，判断客房基本信息节点下是否有客人在住，每遍历一个客房基本信息节点则输出一个节点的入住及租金信息，直至遍历完所有的客房。

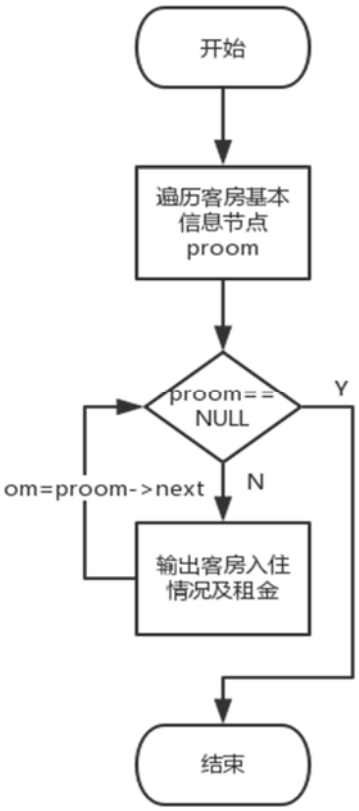


图 4.8.5

9、自定义函数流程

1. 判断日期是否在两日期之间

int Judgedate(char *date_in,char *date_out,char *date);

判断 date 是否在 date_in 与 date_out 之间，是返回 1，不是返回 0.

以公元 0 年为 0 天，计算 date_in, date_out, date 的总天数，比较 date 的天数是否在 date_in 和 date_out 的天数之间，在返回 1，不在返回 0.

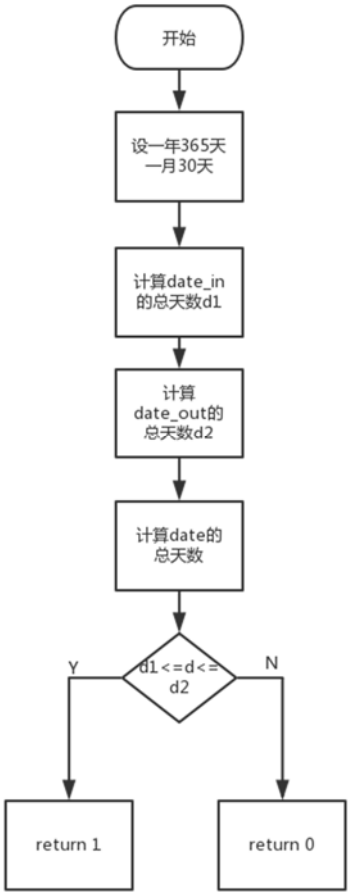


图 4.9.1

2. 统计两个日期之间的月数

float Maintain_month(char * date_in,char * date_out);

粗略估计 date_in 与 date_out 的月数

计算 date_in, date_out 之间的天数 days，days/31 即可粗略算出两日期间的月数，采用四舍五入法保留整月数。

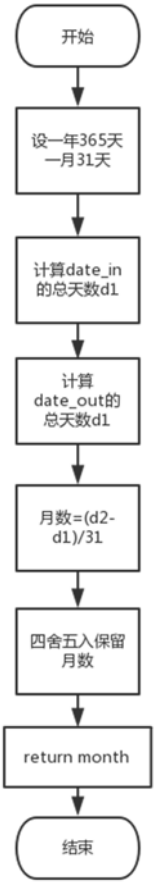


图 4.9.2

五、系统实现

1.MAIN.C

```
#include "rent.h"

unsigned long ul;

int main()
{
    COORD size = { SCR_COL, SCR_ROW };              /*窗口缓冲区大小*/

    gh_std_out = GetStdHandle(STD_OUTPUT_HANDLE); /* 获取标准输出设备句柄*/
    gh_std_in = GetStdHandle(STD_INPUT_HANDLE);   /* 获取标准输入设备句柄*/

    SetConsoleTitle(gp_sys_name);                 /*设置窗口标题*/
```

```
    SetConsoleScreenBufferSize(gh_std_out, size); /*设置窗口缓冲区大小80*25*/

    LoadData();                 /*数据加载*/
    InitInterface();            /*界面初始化*/
    RunSys(&gp_head);           /*系统功能模块的选择及运行*/
    CloseSys(gp_head);          /*退出系统*/

    return 0;
}

/**
* 函数名称: LoadData
* 函数功能: 将代码表和三类基础数据从数据文件载入到内存缓冲区和十字链表中.
* 输入参数: 无
* 输出参数: 无
* 返 回 值: BOOL类型, 功能函数中除了函数ExitSys的返回值可以为FALSE外,
*           其他函数的返回值必须为TRUE.
*
* 调用说明: 为了能够以统一的方式调用各功能函数, 将这些功能函数的原型设为
*           一致, 即无参数且返回值为BOOL. 返回值为FALSE时, 结束程序运行.
*/
BOOL LoadData()
{
    /*加载数据并创建链表*/
    CreatList(&gp_head);
    getchar();
    return TRUE;

}


/**
* 函数名称: CreatList
* 函数功能: 从数据文件读取基础数据, 并存放到所创建的十字链表中.
* 输入参数: 无
* 输出参数: phead 主链头指针的地址, 用来返回所创建的十字链.
* 返 回 值: int型数值, 表示链表创建的情况.
*
* 调用说明:
*/
int CreatList(CLASS_NODE **phead)
{
    CLASS_NODE *hd = NULL, *pclass, tmp1;
    ROOM_NODE *proom, tmp2;
```

```
RENTER_NODE *prenter, tmp3;
FILE *pFile;
int find;
int re = 0;

if ((pFile = fopen(gp_class_info_filename, "rb")) == NULL)
{
    printf("客房分类信息数据文件打开失败!\n");
    return re;
}
printf("客房分类信息数据文件打开成功!\n");

/*从数据文件中读客房分类信息数据，存入以后进先出方式建立的主链中*/
while (fread(&tmp1, sizeof(CLASS_NODE), 1, pFile) == 1)
{
    pclass = (CLASS_NODE *)malloc(sizeof(CLASS_NODE));
    *pclass = tmp1;
    pclass->rnext = NULL;
    pclass->next = hd;
    hd = pclass;
}
fclose(pFile);
if (hd == NULL)
{
    printf("客房分类信息数据文件加载失败!\n");
    return re;
}
printf("客房分类信息数据文件加载成功!\n");
*phead = hd;
re += 4;

if ((pFile = fopen(gp_room_info_filename, "rb")) == NULL)
{
    printf("客房基本信息数据文件打开失败!\n");
    return re;
}
printf("客房基本信息数据文件打开成功!\n");
re += 8;

/*从数据文件中读取学生基本信息数据，存入主链对应结点的学生基本信息支链中*/
while (fread(&tmp2, sizeof(ROOM_NODE), 1, pFile) == 1)
{
    /*创建结点，存放从数据文件中读出的学生基本信息*/
    proom = (ROOM_NODE *)malloc(sizeof(ROOM_NODE));
```

```
        *proom = tmp2;
        proom->rnext = NULL;

        /*在主链上查找该学生所住宿舍楼对应的主链结点*/
        pclass = hd;
        while (pclass != NULL
            && pclass->type != proom->type)
        {
            pclass = pclass->next;
        }
        if (pclass != NULL) /*如果找到，则将结点以后进先出方式插入学生信息支链*/
        {
            proom->next = pclass->rnext;
            pclass->rnext = proom;
        }
        else  /*如果未找到，则释放所创建结点的内存空间*/
        {
            free(proom);
        }
    }
    fclose(pFile);

    if ((pFile = fopen(gp_renter_info_filename, "rb")) == NULL)
    {
        printf("租客信息数据文件打开失败!\n");
        return re;
    }
    printf("租客信息数据文件打开成功!\n");
    re += 16;

    /*从数据文件中读取学生缴费信息数据，存入学生基本信息支链对应结点的缴费支链中*/
    while (fread(&tmp3, sizeof(RENTER_NODE), 1, pFile) == 1)
    {
        /*创建结点，存放从数据文件中读出的学生缴费信息*/
        prenter = (RENTER_NODE *)malloc(sizeof(RENTER_NODE));
        *prenter = tmp3;

        /*查找学生信息支链上对应学生信息结点*/
        pclass = hd;
        find = 0;
        while (pclass != NULL && find == 0)
        {
            proom = pclass->rnext;
            while (proom != NULL && find == 0)
```

```
            {
                if (strcmp(proom->room_id, prenter->room_id) == 0)
                {
                    find = 1;
                    break;
                }
                proom = proom->next;
            }
            pclass = pclass->next;
        }
        if (find)  /*如果找到，则将结点以后进先出方式插入学生缴费信息支链中*/
        {
            prenter->next = proom->rnext;
            proom->rnext = prenter;
        }
        else /*如果未找到，则释放所创建结点的内存空间*/
        {
            free(prenter);
        }
    }
    fclose(pFile);


    return re;
}

/**
* 函数名称: InitInterface
* 函数功能: 初始化界面.
* 输入参数: 无
* 输出参数: 无
* 返 回 值: 无
*
* 调用说明:
*/
void InitInterface()
{
    WORD att = FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_INTENSITY
        ;  /*黄色前景和蓝色背景*/

    SetConsoleTextAttribute(gh_std_out, att);  /*设置控制台屏幕缓冲区字符属性*/

    ClearScreen();  /* 清屏*/
```

```
                    /*创建弹出窗口信息堆栈，将初始化后的屏幕窗口当作第一层弹出窗口*/
    gp_scr_att = (char *)calloc(SCR_COL * SCR_ROW, sizeof(char));/*屏幕字符属性*/
    gp_top_layer = (LAYER_NODE *)malloc(sizeof(LAYER_NODE));
    gp_top_layer->LayerNo = 0;      /*弹出窗口的层号为0*/
    gp_top_layer->rcArea.Left = 0;  /*弹出窗口的区域为整个屏幕窗口*/
    gp_top_layer->rcArea.Top = 0;
    gp_top_layer->rcArea.Right = SCR_COL - 1;
    gp_top_layer->rcArea.Bottom = SCR_ROW - 1;
    gp_top_layer->pContent = NULL;
    gp_top_layer->pScrAtt = gp_scr_att;
    gp_top_layer->next = NULL;

    ShowMenu();     /*显示菜单栏*/
    ShowState();    /*显示状态栏*/

    return;
}

/**
* 函数名称: ClearScreen
* 函数功能: 清除屏幕信息.
* 输入参数: 无
* 输出参数: 无
* 返 回 值: 无
*
* 调用说明:
*/
void ClearScreen(void)
{
    CONSOLE_SCREEN_BUFFER_INFO bInfo;
    COORD home = { 0, 0 };
    unsigned long size;

    GetConsoleScreenBufferInfo(gh_std_out, &bInfo);/*取屏幕缓冲区信息*/
    size = bInfo.dwSize.X * bInfo.dwSize.Y; /*计算屏幕缓冲区字符单元数*/

                                            /*将屏幕缓冲区所有单元的字符属性设置为
当前屏幕缓冲区字符属性*/
    FillConsoleOutputAttribute(gh_std_out, bInfo.wAttributes, size, home, &ul);

    /*将屏幕缓冲区所有单元填充为空格字符*/
    FillConsoleOutputCharacter(gh_std_out, ' ', size, home, &ul);
    fflush(stdin);
```

```
    return;
}

/**
* 函数名称: ShowMenu
* 函数功能: 在屏幕上显示主菜单, 并设置热区, 在主菜单第一项上置选中标记.
* 输入参数: 无
* 输出参数: 无
* 返 回 值: 无
*
* 调用说明:
*/
void ShowMenu()
{
    CONSOLE_SCREEN_BUFFER_INFO bInfo;
    CONSOLE_CURSOR_INFO lpCur;
    COORD size;
    COORD pos = { 0, 0 };
    int i, j;
    int PosA = 2, PosB;
    char ch;

    GetConsoleScreenBufferInfo(gh_std_out, &bInfo);
    size.X = bInfo.dwSize.X;
    size.Y = 1;
    SetConsoleCursorPosition(gh_std_out, pos);
    for (i = 0; i < 5; i++) /*在窗口第一行第一列处输出主菜单项*/
    {
        printf("  %s  ", ga_main_menu[i]);
    }

    GetConsoleCursorInfo(gh_std_out, &lpCur);
    lpCur.bVisible = FALSE;
    SetConsoleCursorInfo(gh_std_out, &lpCur);  /*隐藏光标*/

                                                /*申请动态存储区作为存放菜单条屏幕
区字符信息的缓冲区*/
    gp_buff_menubar_info = (CHAR_INFO *)malloc(size.X * size.Y * sizeof(CHAR_INFO));
    SMALL_RECT rcMenu = { 0, 0, size.X - 1, 0 };

    /*将窗口第一行的内容读入到存放菜单条屏幕区字符信息的缓冲区中*/
    ReadConsoleOutput(gh_std_out, gp_buff_menubar_info, size, pos, &rcMenu);

    /*将这一行中英文字母置为红色, 其他字符单元置为白底黑字*/
```

```
    for (i = 0; i<size.X; i++)
    {
        (gp_buff_menubar_info + i)->Attributes = BACKGROUND_BLUE | BACKGROUND_GREEN
            | BACKGROUND_RED;
        ch = (char)((gp_buff_menubar_info + i)->Char.AsciiChar);
        if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z'))
        {
            (gp_buff_menubar_info + i)->Attributes |= FOREGROUND_RED;
        }
    }

    /*修改后的菜单条字符信息回写到窗口的第一行*/
    WriteConsoleOutput(gh_std_out, gp_buff_menubar_info, size, pos, &rcMenu);
    COORD endPos = { 0, 1 };
    SetConsoleCursorPosition(gh_std_out, endPos);  /*将光标位置设置在第2行第1列*/

                                                   /*将菜单项置为热区，热区编号为
菜单项号，热区类型为0(按钮型)*/
    i = 0;
    do
    {
        PosB = PosA + strlen(ga_main_menu[i]);  /*定位第i+1号菜单项的起止位置*/
        for (j = PosA; j<PosB; j++)
        {
            gp_scr_att[j] |= (i + 1) << 2; /*设置菜单项所在字符单元的属性值*/
        }
        PosA = PosB + 4;
        i++;
    } while (i<5);

    TagMainMenu(gi_sel_menu);  /*在选中主菜单项上做标记，gi_sel_menu初值为1*/

    return;
}

/**
* 函数名称: ShowState
* 函数功能: 显示状态条.
* 输入参数: 无
* 输出参数: 无
* 返 回 值: 无
*
* 调用说明: 状态条字符属性为白底黑字, 初始状态无状态信息.
*/
```

```
void ShowState()
{
    CONSOLE_SCREEN_BUFFER_INFO bInfo;
    COORD size;
    COORD pos = { 0, 0 };
    int i;

    GetConsoleScreenBufferInfo(gh_std_out, &bInfo);
    size.X = bInfo.dwSize.X;
    size.Y = 1;
    SMALL_RECT rcMenu = { 0, bInfo.dwSize.Y - 1, size.X - 1, bInfo.dwSize.Y - 1 };

    if (gp_buff_stateBar_info == NULL)
    {
        gp_buff_stateBar_info = (CHAR_INFO *)malloc(size.X * size.Y *
sizeof(CHAR_INFO));
        ReadConsoleOutput(gh_std_out, gp_buff_stateBar_info, size, pos, &rcMenu);
    }

    for (i = 0; i<size.X; i++)
    {
        (gp_buff_stateBar_info + i)->Attributes = BACKGROUND_BLUE | BACKGROUND_GREEN
            | BACKGROUND_RED;
        /*
        ch = (char)((gp_buff_stateBar_info+i)->Char.AsciiChar);
        if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z'))
        {
        (gp_buff_stateBar_info+i)->Attributes |= FOREGROUND_RED;
        }
        */
    }

    WriteConsoleOutput(gh_std_out, gp_buff_stateBar_info, size, pos, &rcMenu);

    return;
}

/**
* 函数名称: TagMainMenu
* 函数功能: 在指定主菜单项上置选中标志.
* 输入参数: num 选中的主菜单项号
* 输出参数: 无
* 返 回 值: 无
*
```

```
* 调用说明:
*/
void TagMainMenu(int num)
{
    CONSOLE_SCREEN_BUFFER_INFO bInfo;
    COORD size;
    COORD pos = { 0, 0 };
    int PosA = 2, PosB;
    char ch;
    int i;

    if (num == 0) /*num为0时，将会去除主菜单项选中标记*/
    {
        PosA = 0;
        PosB = 0;
    }
    else  /*否则，定位选中主菜单项的起止位置：PosA为起始位置，PosB为截止位置*/
    {
        for (i = 1; i<num; i++)
        {
            PosA += strlen(ga_main_menu[i - 1]) + 4;
        }
        PosB = PosA + strlen(ga_main_menu[num - 1]);
    }

    GetConsoleScreenBufferInfo(gh_std_out, &bInfo);
    size.X = bInfo.dwSize.X;
    size.Y = 1;

    /*去除选中菜单项前面的菜单项选中标记*/
    for (i = 0; i<PosA; i++)
    {
        (gp_buff_menubar_info + i)->Attributes = BACKGROUND_BLUE | BACKGROUND_GREEN
            | BACKGROUND_RED;
        ch = (gp_buff_menubar_info + i)->Char.AsciiChar;
        if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z'))
        {
            (gp_buff_menubar_info + i)->Attributes |= FOREGROUND_RED;
        }
    }

    /*在选中菜单项上做标记，黑底白字*/
    for (i = PosA; i<PosB; i++)
    {
```

```
            (gp_buff_menubar_info + i)->Attributes = FOREGROUND_BLUE | FOREGROUND_GREEN
                | FOREGROUND_RED;
        }

        /*去除选中菜单项后面的菜单项选中标记*/
        for (i = PosB; i<bInfo.dwSize.X; i++)
        {
            (gp_buff_menubar_info + i)->Attributes = BACKGROUND_BLUE | BACKGROUND_GREEN
                | BACKGROUND_RED;
            ch = (char)((gp_buff_menubar_info + i)->Char.AsciiChar);
            if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z'))
            {
                (gp_buff_menubar_info + i)->Attributes |= FOREGROUND_RED;
            }
        }

        /*将做好标记的菜单条信息写到窗口第一行*/
        SMALL_RECT rcMenu = { 0, 0, size.X - 1, 0 };
        WriteConsoleOutput(gh_std_out, gp_buff_menubar_info, size, pos, &rcMenu);

        return;
    }

    /**
    * 函数名称: CloseSys
    * 函数功能: 关闭系统.
    * 输入参数: hd 主链头指针
    * 输出参数: 无
    * 返 回 值: 无
    *
    * 调用说明:
    */
    void CloseSys(CLASS_NODE *hd)
    {
        CLASS_NODE *pDormNode1 = hd, *pDormNode2;
        ROOM_NODE *pStuNode1, *pStuNode2;
        RENTER_NODE *pChargeNode1, *pChargeNode2;

        while (pDormNode1 != NULL) /*释放十字交叉链表的动态存储区*/
        {
            pDormNode2 = pDormNode1->next;
            pStuNode1 = pDormNode1->rnext;
            while (pStuNode1 != NULL) /*释放学生基本信息支链的动态存储区*/
            {
```

```
            pStuNode2 = pStuNode1->next;
            pChargeNode1 = pStuNode1->rnext;
            while (pChargeNode1 != NULL) /*释放缴费信息支链的动态存储区*/
            {
                pChargeNode2 = pChargeNode1->next;
                free(pChargeNode1);
                pChargeNode1 = pChargeNode2;
            }
            free(pStuNode1);
            pStuNode1 = pStuNode2;
        }
        free(pDormNode1);  /*释放主链结点的动态存储区*/
        pDormNode1 = pDormNode2;
    }

    ClearScreen();        /*清屏*/

                          /*释放存放菜单条、状态条、性别代码和学生类别代码等信息动态
存储区*/
    free(gp_buff_menubar_info);
    free(gp_buff_stateBar_info);

    /*关闭标准输入和输出设备句柄*/
    CloseHandle(gh_std_out);
    CloseHandle(gh_std_in);

    /*将窗口标题栏置为运行结束*/
    SetConsoleTitle("运行结束");

    return;
}

/**
* 函数名称: RunSys
* 函数功能: 运行系统, 在系统主界面下运行用户所选择的功能模块.
* 输入参数: 无
* 输出参数: phead 主链头指针的地址
* 返 回 值: 无
*
* 调用说明:
*/
void RunSys(CLASS_NODE **phead)
{
    INPUT_RECORD inRec;
```

```
DWORD res;
COORD pos = { 0, 0 };
BOOL bRet = TRUE;
int i, loc, num;
int cNo, cAtt;      /*cNo:字符单元层号, cAtt:字符单元属性*/
char vkc, asc;      /*vkc:虚拟键代码, asc:字符的ASCII码值*/

while (bRet)
{
    /*从控制台输入缓冲区中读一条记录*/
    ReadConsoleInput(gh_std_in, &inRec, 1, &res);

    if (inRec.EventType == MOUSE_EVENT) /*如果记录由鼠标事件产生*/
    {
        pos = inRec.Event.MouseEvent.dwMousePosition;  /*获取鼠标坐标位置*/
        cNo = gp_scr_att[pos.Y * SCR_COL + pos.X] & 3; /*取该位置的层号*/
        cAtt = gp_scr_att[pos.Y * SCR_COL + pos.X] >> 2;/*取该字符单元属性*/
        if (cNo == 0) /*层号为0，表明该位置未被弹出子菜单覆盖*/
        {
            /* cAtt > 0 表明该位置处于热区(主菜单项字符单元)
            * cAtt != gi_sel_menu 表明该位置的主菜单项未被选中
            * gp_top_layer->LayerNo > 0 表明当前有子菜单弹出
            */
            if (cAtt > 0 && cAtt != gi_sel_menu && gp_top_layer->LayerNo > 0)
            {
                PopOff();             /*关闭弹出的子菜单*/
                gi_sel_sub_menu = 0; /*将选中子菜单项的项号置为0*/
                PopMenu(cAtt);        /*弹出鼠标所在主菜单项对应的子菜单*/
            }
        }
        else if (cAtt > 0) /*鼠标所在位置为弹出子菜单的菜单项字符单元*/
        {
            TagSubMenu(cAtt); /*在该子菜单项上做选中标记*/
        }

        if (inRec.Event.MouseEvent.dwButtonState
            == FROM_LEFT_1ST_BUTTON_PRESSED) /*如果按下鼠标左边第一键*/
        {
            if (cNo == 0) /*层号为0，表明该位置未被弹出子菜单覆盖*/
            {
                if (cAtt > 0) /*如果该位置处于热区(主菜单项字符单元)*/
                {
                    PopMenu(cAtt);    /*弹出鼠标所在主菜单项对应的子菜单*/
                }
```

```
                    /*如果该位置不属于主菜单项字符单元，且有子菜单弹出*/
                    else if (gp_top_layer->LayerNo > 0)
                    {
                        PopOff();               /*关闭弹出的子菜单*/
                        gi_sel_sub_menu = 0; /*将选中子菜单项的项号置为0*/
                    }
                }
                else /*层号不为0，表明该位置被弹出子菜单覆盖*/
                {
                    if (cAtt > 0) /*如果该位置处于热区(子菜单项字符单元)*/
                    {
                        PopOff(); /*关闭弹出的子菜单*/
                        gi_sel_sub_menu = 0; /*将选中子菜单项的项号置为0*/

                                                    /*执行对应功能函数:gi_sel_menu主菜单
项号,cAtt子菜单项号*/
                        bRet = ExeFunction(gi_sel_menu, cAtt);
                    }
                }
            }
            else if (inRec.Event.MouseEvent.dwButtonState
                == RIGHTMOST_BUTTON_PRESSED) /*如果按下鼠标右键*/
            {
                if (cNo == 0) /*层号为0，表明该位置未被弹出子菜单覆盖*/
                {
                    PopOff();               /*关闭弹出的子菜单*/
                    gi_sel_sub_menu = 0; /*将选中子菜单项的项号置为0*/
                }
            }
        }
        else if (inRec.EventType == KEY_EVENT  /*如果记录由按键产生*/
            && inRec.Event.KeyEvent.bKeyDown) /*且键被按下*/
        {
            vkc = inRec.Event.KeyEvent.wVirtualKeyCode; /*获取按键的虚拟键码*/
            asc = inRec.Event.KeyEvent.uChar.AsciiChar; /*获取按键的ASC码*/

                                                        /*系统快捷键的处理*/
            if (vkc == 112) /*如果按下F1键*/
            {
                if (gp_top_layer->LayerNo != 0) /*如果当前有子菜单弹出*/
                {
                    PopOff();               /*关闭弹出的子菜单*/
                    gi_sel_sub_menu = 0; /*将选中子菜单项的项号置为0*/
                }
```

```
        bRet = ExeFunction(5, 1);  /*运行帮助主题功能函数*/
    }
    else if (inRec.Event.KeyEvent.dwControlKeyState
        & (LEFT_ALT_PRESSED | RIGHT_ALT_PRESSED))
    { /*如果按下左或右Alt键*/
        switch (vkc)  /*判断组合键Alt+字母*/
        {
        case 88:  /*Alt+X 退出*/
            if (gp_top_layer->LayerNo != 0)
            {
                PopOff();
                gi_sel_sub_menu = 0;
            }
            bRet = ExeFunction(1, 4);
            break;
        case 70:  /*Alt+F*/
            PopMenu(1);
            break;
        case 77: /*Alt+M*/
            PopMenu(2);
            break;
        case 81: /*Alt+Q*/
            PopMenu(3);
            break;
        case 83: /*Alt+S*/
            PopMenu(4);
            break;
        case 72: /*Alt+H*/
            PopMenu(5);
            break;
        }
    }
    else if (asc == 0) /*其他控制键的处理*/
    {
        if (gp_top_layer->LayerNo == 0) /*如果未弹出子菜单*/
        {
            switch (vkc) /*处理方向键(左、右、下)，不响应其他控制键*/
            {
            case 37:
                gi_sel_menu--;
                if (gi_sel_menu == 0)
                {
                    gi_sel_menu = 5;
                }
```

```
            TagMainMenu(gi_sel_menu);
            break;
        case 39:
            gi_sel_menu++;
            if (gi_sel_menu == 6)
            {
                gi_sel_menu = 1;
            }
            TagMainMenu(gi_sel_menu);
            break;
        case 40:
            PopMenu(gi_sel_menu);
            TagSubMenu(1);
            break;
        }
    }
    else  /*已弹出子菜单时*/
    {
        for (loc = 0, i = 1; i<gi_sel_menu; i++)
        {
            loc += ga_sub_menu_count[i - 1];
        }  /*计算该子菜单中的第一项在子菜单字符串数组中的位置(下标)*/
        switch (vkc) /*方向键(左、右、上、下)的处理*/
        {
        case 37:
            gi_sel_menu--;
            if (gi_sel_menu < 1)
            {
                gi_sel_menu = 5;
            }
            TagMainMenu(gi_sel_menu);
            PopOff();
            PopMenu(gi_sel_menu);
            TagSubMenu(1);
            break;
        case 38:
            num = gi_sel_sub_menu - 1;
            if (num < 1)
            {
                num = ga_sub_menu_count[gi_sel_menu - 1];
            }
            if (strlen(ga_sub_menu[loc + num - 1]) == 0)
            {
                num--;
```

```
            }
            TagSubMenu(num);
            break;
        case 39:
            gi_sel_menu++;
            if (gi_sel_menu > 5)
            {
                gi_sel_menu = 1;
            }
            TagMainMenu(gi_sel_menu);
            PopOff();
            PopMenu(gi_sel_menu);
            TagSubMenu(1);
            break;
        case 40:
            num = gi_sel_sub_menu + 1;
            if (num > ga_sub_menu_count[gi_sel_menu - 1])
            {
                num = 1;
            }
            if (strlen(ga_sub_menu[loc + num - 1]) == 0)
            {
                num++;
            }
            TagSubMenu(num);
            break;
        }
    }
}
else if ((asc - vkc == 0) || (asc - vkc == 32)) {  /*按下普通键*/
    if (gp_top_layer->LayerNo == 0)  /*如果未弹出子菜单*/
    {
        switch (vkc)
        {
        case 70: /*f或F*/
            PopMenu(1);
            break;
        case 77: /*m或M*/
            PopMenu(2);
            break;
        case 81: /*q或Q*/
            PopMenu(3);
            break;
        case 83: /*s或S*/
```

```
                PopMenu(4);
                break;
            case 72: /*h或H*/
                PopMenu(5);
                break;
            case 13: /*回车*/
                PopMenu(gi_sel_menu);
                TagSubMenu(1);
                break;
            }
        }
        else /*已弹出子菜单时的键盘输入处理*/
        {
            if (vkc == 27) /*如果按下ESC键*/
            {
                PopOff();
                gi_sel_sub_menu = 0;
            }
            else if (vkc == 13) /*如果按下回车键*/
            {
                num = gi_sel_sub_menu;
                PopOff();
                gi_sel_sub_menu = 0;
                bRet = ExeFunction(gi_sel_menu, num);
            }
            else /*其他普通键的处理*/
            {
                /*计算该子菜单中的第一项在子菜单字符串数组中的位置(下标)*/
                for (loc = 0, i = 1; i<gi_sel_menu; i++)
                {
                    loc += ga_sub_menu_count[i - 1];
                }

                /*依次与当前子菜单中每一项的代表字符进行比较*/
                for (i = loc; i<loc + ga_sub_menu_count[gi_sel_menu - 1];
i++)
                {
                    if (strlen(ga_sub_menu[i])>0 && vkc ==
ga_sub_menu[i][1])
                    { /*如果匹配成功*/
                        PopOff();
                        gi_sel_sub_menu = 0;
                        bRet = ExeFunction(gi_sel_menu, i - loc + 1);
                    }
```

```
                        }
                    }
                }
            }
        }
    }
}

/**
* 函数名称: PopMenu
* 函数功能: 弹出指定主菜单项对应的子菜单.
* 输入参数: num 指定的主菜单项号
* 输出参数: 无
* 返 回 值: 无
*
* 调用说明:
*/
void PopMenu(int num)
{
    LABEL_BUNDLE labels;
    HOT_AREA areas;
    SMALL_RECT rcPop;
    COORD pos;
    WORD att;
    char *pCh;
    int i, j, loc = 0;

    if (num != gi_sel_menu)       /*如果指定主菜单不是已选中菜单*/
    {
        if (gp_top_layer->LayerNo != 0) /*如果此前已有子菜单弹出*/
        {
            PopOff();
            gi_sel_sub_menu = 0;
        }
    }
    else if (gp_top_layer->LayerNo != 0) /*若已弹出该子菜单，则返回*/
    {
        return;
    }

    gi_sel_menu = num;    /*将选中主菜单项置为指定的主菜单项*/
    TagMainMenu(gi_sel_menu); /*在选中的主菜单项上做标记*/
    LocSubMenu(gi_sel_menu, &rcPop); /*计算弹出子菜单的区域位置, 存放在rcPop中*/
```

```
                                        /*计算该子菜单中的第一项在子菜单字符串数组中的
位置(下标)*/
    for (i = 1; i<gi_sel_menu; i++)
    {
        loc += ga_sub_menu_count[i - 1];
    }
    /*将该组子菜单项项名存入标签束结构变量*/
    labels.ppLabel = ga_sub_menu + loc;   /*标签束第一个标签字符串的地址*/
    labels.num = ga_sub_menu_count[gi_sel_menu - 1]; /*标签束中标签字符串的个数*/
    COORD aLoc[labels.num];/*定义一个坐标数组，存放每个标签字符串输出位置的坐标*/
    for (i = 0; i<labels.num; i++) /*确定标签字符串的输出位置，存放在坐标数组中*/
    {
        aLoc[i].X = rcPop.Left + 2;
        aLoc[i].Y = rcPop.Top + i + 1;
    }
    labels.pLoc = aLoc; /*使标签束结构变量labels的成员pLoc指向坐标数组的首元素*/
                        /*设置热区信息*/
    areas.num = labels.num;       /*热区的个数，等于标签的个数，即子菜单的项数*/
    SMALL_RECT aArea[areas.num];                    /*定义数组存放所有热区位置*/
    char aSort[areas.num];                         /*定义数组存放所有热区对应类别*/
    char aTag[areas.num];                           /*定义数组存放每个热区的编号*/
    for (i = 0; i<areas.num; i++)
    {
        aArea[i].Left = rcPop.Left + 2;  /*热区定位*/
        aArea[i].Top = rcPop.Top + i + 1;
        aArea[i].Right = rcPop.Right - 2;
        aArea[i].Bottom = aArea[i].Top;
        aSort[i] = 0;       /*热区类别都为0(按钮型)*/
        aTag[i] = i + 1;           /*热区按顺序编号*/
    }
    areas.pArea = aArea;/*使热区结构变量areas的成员pArea指向热区位置数组首元素*/
    areas.pSort = aSort;/*使热区结构变量areas的成员pSort指向热区类别数组首元素*/
    areas.pTag = aTag;   /*使热区结构变量areas的成员pTag指向热区编号数组首元素*/

    att = BACKGROUND_BLUE | BACKGROUND_GREEN | BACKGROUND_RED;  /*白底黑字*/
    PopUp(&rcPop, att, &labels, &areas);
    DrawBox(&rcPop);  /*给弹出窗口画边框*/
    pos.X = rcPop.Left + 2;
    for (pos.Y = rcPop.Top + 1; pos.Y<rcPop.Bottom; pos.Y++)
    { /*此循环用来在空串子菜项位置画线形成分隔，并取消此菜单项的热区属性*/
        pCh = ga_sub_menu[loc + pos.Y - rcPop.Top - 1];
        if (strlen(pCh) == 0) /*串长为0，表明为空串*/
        {   /*首先画横线*/
            FillConsoleOutputCharacter(gh_std_out, '-', rcPop.Right - rcPop.Left - 3,
```

```
pos, &ul);
            for (j = rcPop.Left + 2; j<rcPop.Right - 1; j++)
            {   /*取消该区域字符单元的热区属性*/
                gp_scr_att[pos.Y*SCR_COL + j] &= 3; /*按位与的结果保留了低两位*/
            }
        }

    }
    /*将子菜单项的功能键设为白底红字*/
    pos.X = rcPop.Left + 3;
    att = FOREGROUND_RED | BACKGROUND_BLUE | BACKGROUND_GREEN | BACKGROUND_RED;
    for (pos.Y = rcPop.Top + 1; pos.Y<rcPop.Bottom; pos.Y++)
    {
        if (strlen(ga_sub_menu[loc + pos.Y - rcPop.Top - 1]) == 0)
        {
            continue;  /*跳过空串*/
        }
        FillConsoleOutputAttribute(gh_std_out, att, 1, pos, &ul);
    }
    return;
}

/**
* 函数名称: PopUp
* 函数功能: 在指定区域输出弹出窗口信息, 同时设置热区, 将弹出窗口位置信息入栈.
* 输入参数: pRc 弹出窗口位置数据存放的地址
*           att 弹出窗口区域字符属性
*           pLabel 弹出窗口中标签束信息存放的地址
pHotArea 弹出窗口中热区信息存放的地址
* 输出参数: 无
* 返 回 值: 无
*
* 调用说明:
*/
void PopUp(SMALL_RECT *pRc, WORD att, LABEL_BUNDLE *pLabel, HOT_AREA *pHotArea)
{
    LAYER_NODE *nextLayer;
    COORD size;
    COORD pos = { 0, 0 };
    char *pCh;
    int i, j, row;

    /*弹出窗口所在位置字符单元信息入栈*/
    size.X = pRc->Right - pRc->Left + 1;     /*弹出窗口的宽度*/
```

```
    size.Y = pRc->Bottom - pRc->Top + 1;     /*弹出窗口的高度*/
                                             /*申请存放弹出窗口相关信息的动态存储区
*/
    nextLayer = (LAYER_NODE *)malloc(sizeof(LAYER_NODE));
    nextLayer->next = gp_top_layer;
    nextLayer->LayerNo = gp_top_layer->LayerNo + 1;
    nextLayer->rcArea = *pRc;
    nextLayer->pContent = (CHAR_INFO *)malloc(size.X*size.Y * sizeof(CHAR_INFO));
    nextLayer->pScrAtt = (char *)malloc(size.X*size.Y * sizeof(char));
    pCh = nextLayer->pScrAtt;
    /*将弹出窗口覆盖区域的字符信息保存，用于在关闭弹出窗口时恢复原样*/
    ReadConsoleOutput(gh_std_out, nextLayer->pContent, size, pos, pRc);
    for (i = pRc->Top; i <= pRc->Bottom; i++)
    {   /*此二重循环将所覆盖字符单元的原先属性值存入动态存储区，便于以后恢复*/
        for (j = pRc->Left; j <= pRc->Right; j++)
        {
            *pCh = gp_scr_att[i*SCR_COL + j];
            pCh++;
        }
    }
    gp_top_layer = nextLayer;  /*完成弹出窗口相关信息入栈操作*/
                               /*设置弹出窗口区域字符的新属性*/
    pos.X = pRc->Left;
    pos.Y = pRc->Top;
    for (i = pRc->Top; i <= pRc->Bottom; i++)
    {
        FillConsoleOutputAttribute(gh_std_out, att, size.X, pos, &ul);
        pos.Y++;
    }
    /*将标签束中的标签字符串在设定的位置输出*/
    for (i = 0; i<pLabel->num; i++)
    {
        pCh = pLabel->ppLabel[i];
        if (strlen(pCh) != 0)
        {
            WriteConsoleOutputCharacter(gh_std_out, pCh, strlen(pCh),
                pLabel->pLoc[i], &ul);
        }
    }
    /*设置弹出窗口区域字符单元的新属性*/
    for (i = pRc->Top; i <= pRc->Bottom; i++)
    {   /*此二重循环设置字符单元的层号*/
        for (j = pRc->Left; j <= pRc->Right; j++)
        {
```

```
                gp_scr_att[i*SCR_COL + j] = gp_top_layer->LayerNo;
            }
        }

        for (i = 0; i<pHotArea->num; i++)
        {   /*此二重循环设置所有热区中字符单元的热区类型和热区编号*/
            row = pHotArea->pArea[i].Top;
            for (j = pHotArea->pArea[i].Left; j <= pHotArea->pArea[i].Right; j++)
            {
                gp_scr_att[row*SCR_COL + j] |= (pHotArea->pSort[i] << 6)
                    | (pHotArea->pTag[i] << 2);
            }
        }
        return;
}

/**
* 函数名称: PopOff
* 函数功能: 关闭顶层弹出窗口, 恢复覆盖区域原外观和字符单元原属性.
* 输入参数: 无
* 输出参数: 无
* 返 回 值: 无
*
* 调用说明:
*/
void PopOff(void)
{
    LAYER_NODE *nextLayer;
    COORD size;
    COORD pos = { 0, 0 };
    char *pCh;
    int i, j;

    if ((gp_top_layer->next == NULL) || (gp_top_layer->pContent == NULL))
    {   /*栈底存放的主界面屏幕信息，不用关闭*/
        return;
    }
    nextLayer = gp_top_layer->next;
    /*恢复弹出窗口区域原外观*/
    size.X = gp_top_layer->rcArea.Right - gp_top_layer->rcArea.Left + 1;
    size.Y = gp_top_layer->rcArea.Bottom - gp_top_layer->rcArea.Top + 1;
    WriteConsoleOutput(gh_std_out, gp_top_layer->pContent, size, pos,
&(gp_top_layer->rcArea));
    /*恢复字符单元原属性*/
```

```
    pCh = gp_top_layer->pScrAtt;
    for (i = gp_top_layer->rcArea.Top; i <= gp_top_layer->rcArea.Bottom; i++)
    {
        for (j = gp_top_layer->rcArea.Left; j <= gp_top_layer->rcArea.Right; j++)
        {
            gp_scr_att[i*SCR_COL + j] = *pCh;
            pCh++;
        }
    }
    free(gp_top_layer->pContent);    /*释放动态存储区*/
    free(gp_top_layer->pScrAtt);
    free(gp_top_layer);
    gp_top_layer = nextLayer;
    gi_sel_sub_menu = 0;
    return;
}

/**
* 函数名称: DrawBox
* 函数功能: 在指定区域画边框.
* 输入参数: pRc 存放区域位置信息的地址
* 输出参数: 无
* 返 回 值: 无
*
* 调用说明:
*/
void DrawBox(SMALL_RECT *pRc)
{
    char chBox[] = { '+','-','|' };  /*画框用的字符*/
    COORD pos = { pRc->Left, pRc->Top };  /*定位在区域的左上角*/

    WriteConsoleOutputCharacter(gh_std_out, &chBox[0], 1, pos, &ul);/*画边框左上角*/
    for (pos.X = pRc->Left + 1; pos.X < pRc->Right; pos.X++)
    {   /*此循环画上边框横线*/
        WriteConsoleOutputCharacter(gh_std_out, &chBox[1], 1, pos, &ul);
    }
    pos.X = pRc->Right;
    WriteConsoleOutputCharacter(gh_std_out, &chBox[0], 1, pos, &ul);/*画边框右上角*/
    for (pos.Y = pRc->Top + 1; pos.Y < pRc->Bottom; pos.Y++)
    {   /*此循环画边框左边线和右边线*/
        pos.X = pRc->Left;
        WriteConsoleOutputCharacter(gh_std_out, &chBox[2], 1, pos, &ul);
        pos.X = pRc->Right;
        WriteConsoleOutputCharacter(gh_std_out, &chBox[2], 1, pos, &ul);
```

```
    }
    pos.X = pRc->Left;
    pos.Y = pRc->Bottom;
    WriteConsoleOutputCharacter(gh_std_out, &chBox[0], 1, pos, &ul);/*画边框左下角*/
    for (pos.X = pRc->Left + 1; pos.X < pRc->Right; pos.X++)
    {   /*画下边框横线*/
        WriteConsoleOutputCharacter(gh_std_out, &chBox[1], 1, pos, &ul);
    }
    pos.X = pRc->Right;
    WriteConsoleOutputCharacter(gh_std_out, &chBox[0], 1, pos, &ul);/*画边框右下角*/
    return;
}

/**
* 函数名称: TagSubMenu
* 函数功能: 在指定子菜单项上做选中标记.
* 输入参数: num 选中的子菜单项号
* 输出参数: 无
* 返 回 值: 无
*
* 调用说明:
*/
void TagSubMenu(int num)
{
    SMALL_RECT rcPop;
    COORD pos;
    WORD att;
    int width;

    LocSubMenu(gi_sel_menu, &rcPop);  /*计算弹出子菜单的区域位置, 存放在rcPop中*/
    if ((num<1) || (num == gi_sel_sub_menu) || (num>rcPop.Bottom - rcPop.Top - 1))
    {   /*如果子菜单项号越界, 或该项子菜单已被选中, 则返回*/
        return;
    }

    pos.X = rcPop.Left + 2;
    width = rcPop.Right - rcPop.Left - 3;
    if (gi_sel_sub_menu != 0) /*首先取消原选中子菜单项上的标记*/
    {
        pos.Y = rcPop.Top + gi_sel_sub_menu;
        att = BACKGROUND_BLUE | BACKGROUND_GREEN | BACKGROUND_RED;  /*白底黑字*/
        FillConsoleOutputAttribute(gh_std_out, att, width, pos, &ul);
        pos.X += 1;
        att |= FOREGROUND_RED;/*白底红字*/
```

```
        FillConsoleOutputAttribute(gh_std_out, att, 1, pos, &ul);
    }
    /*在制定子菜单项上做选中标记*/
    pos.X = rcPop.Left + 2;
    pos.Y = rcPop.Top + num;
    att = FOREGROUND_BLUE | FOREGROUND_GREEN | FOREGROUND_RED;  /*黑底白字*/
    FillConsoleOutputAttribute(gh_std_out, att, width, pos, &ul);
    gi_sel_sub_menu = num;  /*修改选中子菜单项号*/
    return;
}

/**
* 函数名称: LocSubMenu
* 函数功能: 计算弹出子菜单区域左上角和右下角的位置.
* 输入参数: num 选中的主菜单项号
* 输出参数: rc 存放区域位置信息的地址
* 返 回 值: 无
*
* 调用说明:
*/
void LocSubMenu(int num, SMALL_RECT *rc)
{
    int i, len, loc = 0;

    rc->Top = 1; /*区域的上边定在第2行，行号为1*/
    rc->Left = 1;
    for (i = 1; i<num; i++)
    {   /*计算区域左边界位置，同时计算第一个子菜单项在子菜单字符串数组中的位置*/
        rc->Left += strlen(ga_main_menu[i - 1]) + 4;
        loc += ga_sub_menu_count[i - 1];
    }
    rc->Right = strlen(ga_sub_menu[loc]);/*暂时存放第一个子菜单项字符串长度*/
    for (i = 1; i<ga_sub_menu_count[num - 1]; i++)
    {   /*查找最长子菜单字符串，将其长度存放在rc->Right*/
        len = strlen(ga_sub_menu[loc + i]);
        if (rc->Right < len)
        {
            rc->Right = len;
        }
    }
    rc->Right += rc->Left + 3;  /*计算区域的右边界*/
    rc->Bottom = rc->Top + ga_sub_menu_count[num - 1] + 1;/*计算区域下边的行号*/
    if (rc->Right >= SCR_COL)  /*右边界越界的处理*/
    {
```

```
            len = rc->Right - SCR_COL + 1;
            rc->Left -= len;
            rc->Right = SCR_COL - 1;
        }
        return;
}

/**
* 函数名称: DealInput
* 函数功能: 在弹出窗口区域设置热区, 等待并相应用户输入.
* 输入参数: pHotArea
*           piHot 焦点热区编号的存放地址, 即指向焦点热区编号的指针
* 输出参数: piHot 用鼠标单击、按回车或空格时返回当前热区编号
* 返 回 值:
*
* 调用说明:
*/
int DealInput(HOT_AREA *pHotArea, int *piHot)
{
    INPUT_RECORD inRec;
    DWORD res;
    COORD pos = { 0, 0 };
    int num, arrow, iRet = 0;
    int cNo, cTag, cSort;/*cNo:层号, cTag:热区编号, cSort: 热区类型*/
    char vkc, asc;       /*vkc:虚拟键代码, asc:字符的ASCII码值*/

    SetHotPoint(pHotArea, *piHot);
    while (TRUE)
    {    /*循环*/
        ReadConsoleInput(gh_std_in, &inRec, 1, &res);
        if ((inRec.EventType == MOUSE_EVENT) &&
            (inRec.Event.MouseEvent.dwButtonState
                == FROM_LEFT_1ST_BUTTON_PRESSED))
        {
            pos = inRec.Event.MouseEvent.dwMousePosition;
            cNo = gp_scr_att[pos.Y * SCR_COL + pos.X] & 3;
            cTag = (gp_scr_att[pos.Y * SCR_COL + pos.X] >> 2) & 15;
            cSort = (gp_scr_att[pos.Y * SCR_COL + pos.X] >> 6) & 3;

            if ((cNo == gp_top_layer->LayerNo) && cTag > 0)
            {
                *piHot = cTag;
                SetHotPoint(pHotArea, *piHot);
                if (cSort == 0)
```

```
            {
                iRet = 13;
                break;
            }
        }
    }
    else if (inRec.EventType == KEY_EVENT && inRec.Event.KeyEvent.bKeyDown)
    {
        vkc = inRec.Event.KeyEvent.wVirtualKeyCode;
        asc = inRec.Event.KeyEvent.uChar.AsciiChar;;
        if (asc == 0)
        {
            arrow = 0;
            switch (vkc)
            {  /*方向键(左、上、右、下)的处理*/
            case 37: arrow = 1; break;
            case 38: arrow = 2; break;
            case 39: arrow = 3; break;
            case 40: arrow = 4; break;
            }
            if (arrow > 0)
            {
                num = *piHot;
                while (TRUE)
                {
                    if (arrow < 3)
                    {
                        num--;
                    }
                    else
                    {
                        num++;
                    }
                    if ((num < 1) || (num > pHotArea->num) ||
                        ((arrow % 2) && (pHotArea->pArea[num - 1].Top
                            == pHotArea->pArea[*piHot - 1].Top)) || ((!(arrow %
2))
                                && (pHotArea->pArea[num - 1].Top
                                    != pHotArea->pArea[*piHot - 1].Top)))
                    {
                        break;
                    }
                }
                if (num > 0 && num <= pHotArea->num)
```

```
                {
                    *piHot = num;
                    SetHotPoint(pHotArea, *piHot);
                }
            }
        }
        else if (vkc == 27)
        {  /*ESC键*/
            iRet = 27;
            break;
        }
        else if (vkc == 13 || vkc == 32)
        {  /*回车键或空格表示按下当前按钮*/
            iRet = 13;
            break;
        }
    }
}
return iRet;
}

void SetHotPoint(HOT_AREA *pHotArea, int iHot)
{
    CONSOLE_CURSOR_INFO lpCur;
    COORD pos = { 0, 0 };
    WORD att1, att2;
    int i, width;

    att1 = FOREGROUND_BLUE | FOREGROUND_GREEN | FOREGROUND_RED;  /*黑底白字*/
    att2 = BACKGROUND_BLUE | BACKGROUND_GREEN | BACKGROUND_RED;  /*白底黑字*/
    for (i = 0; i<pHotArea->num; i++)
    {  /*将按钮类热区置为白底黑字*/
        pos.X = pHotArea->pArea[i].Left;
        pos.Y = pHotArea->pArea[i].Top;
        width = pHotArea->pArea[i].Right - pHotArea->pArea[i].Left + 1;
        if (pHotArea->pSort[i] == 0)
        {  /*热区是按钮类*/
            FillConsoleOutputAttribute(gh_std_out, att2, width, pos, &ul);
        }
    }

    pos.X = pHotArea->pArea[iHot - 1].Left;
    pos.Y = pHotArea->pArea[iHot - 1].Top;
    width = pHotArea->pArea[iHot - 1].Right - pHotArea->pArea[iHot - 1].Left + 1;
```

```
    if (pHotArea->pSort[iHot - 1] == 0)
    {  /*被激活热区是按钮类*/
        FillConsoleOutputAttribute(gh_std_out, att1, width, pos, &ul);
    }
    else if (pHotArea->pSort[iHot - 1] == 1)
    {  /*被激活热区是文本框类*/
        SetConsoleCursorPosition(gh_std_out, pos);
        GetConsoleCursorInfo(gh_std_out, &lpCur);
        lpCur.bVisible = TRUE;
        SetConsoleCursorInfo(gh_std_out, &lpCur);
    }
}

/**
* 函数名称: ExeFunction
* 函数功能: 执行由主菜单号和子菜单号确定的功能函数.
* 输入参数: m 主菜单项号
*           s 子菜单项号
* 输出参数: 无
* 返 回 值: BOOL类型, TRUE 或 FALSE
*
* 调用说明: 仅在执行函数ExitSys时, 才可能返回FALSE, 其他情况下总是返回TRUE
*/
BOOL ExeFunction(int m, int s)
{
    BOOL bRet = TRUE;
    /*函数指针数组，用来存放所有功能函数的入口地址*/
    BOOL(*pFunction[ga_sub_menu_count[0] + ga_sub_menu_count[1] + ga_sub_menu_count[2]
+ ga_sub_menu_count[3] + ga_sub_menu_count[4]])(void);
    int i, loc;

    /*将功能函数入口地址存入与功能函数所在主菜单号和子菜单号对应下标的数组元素*/
    pFunction[0] = SaveData;
    pFunction[1] = ExitSys;
    /*=======================================*/
    pFunction[2] = Insertclass;
    pFunction[3] = Maintainclass;
    pFunction[4] = Delclass;

    pFunction[5] = Insertroom;
    pFunction[6] = Maintainroom;
    pFunction[7] = Delroom;

    pFunction[8] = Insertrenter;
```

```
    pFunction[9] = Maintainrenter;
    pFunction[10] = Delrenter;
    /*=====================================*/
    pFunction[11] = Queryclass;
    pFunction[12] = Queryroom;
    pFunction[13] = Queryrenter;

    pFunction[14] = Query;
    pFunction[15] = QueryTypeInfo;
    /*=====================================*/
    pFunction[16] = Staclass;
    pFunction[17] = Staincome;
    pFunction[18] = Staroom_info;
    pFunction[19] = Starenter_info;
    pFunction[20] = Sta_in;
    /*=====================================*/
    pFunction[21] = HelpTopic;
    pFunction[22] = AboutDorm;
    pFunction[23] = Clear;

    for (i = 1, loc = 0; i<m; i++)  /*根据主菜单号和子菜单号计算对应下标*/
    {
        loc += ga_sub_menu_count[i - 1];
    }
    loc += s - 1;

    if (pFunction[loc] != NULL)
    {
        bRet = (*pFunction[loc])();  /*用函数指针调用所指向的功能函数*/
    }

    return bRet;
}
/*=====================================*/
/**插入客房分类信息*/
BOOL Insertclass()
{
    char *plabel_name[] = { "主菜单项：数据维护",
        "子菜单项：插入客房分类信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
```

```
    CLASS_NODE *pclass;
    CLASS_NODE *pc;
    pclass = (CLASS_NODE *)malloc(sizeof(CLASS_NODE));
    /*录取客房分类信息*/
    printf("               ***************请输入客房类别（S,D,T）   ***************");
    printf("                    \n               ***************    ");
    pclass->type = toupper(getchar());
    getchar();
    printf("               ***************请输入最多入住人数（输入数字）
****************");
    printf("                    \n               ***************    ");
    scanf("%d", &pclass->pnum);
    getchar();
    printf("               ***************请输入客房套数（输入一位数字）
****************");
    printf("                    \n               ***************    ");
    scanf("%d", &pclass->rnum);
    getchar();
    printf("               ***************请输入未住套数（输入一位数字）
****************");
    printf("                    \n               ***************    ");
    scanf("%d", &pclass->emptyr);
    getchar();
    printclass(pclass);
    printf("               ***************是否保存？（y或n）   ***************");
    printf("                    \n               ***************    ");
    /*是否保存*/
    char c = getchar();
    getchar();
    if (c == 'y' || c == 'Y')
    {
        /*判断客房类别是否存在，存在则覆盖，不存在创建新节点*/
        for (pc = gp_head; pc != NULL; pc = pc->next)
            if (pc->type == pclass->type)
            {
                //存在执行这一部
                printf("客房类别已存在，修改成功！\n");
                /*修改数据域*/
                pclass->next = pc->next;
                pclass->rnext = pc->rnext;
                *pc = *pclass;
                free(pclass);
                return TRUE;
            }
```

```
        //不存在执行这一步
        pclass->next = gp_head;
        pclass->rnext = NULL;
        gp_head = pclass;
        printf("保存成功！\n");
        return TRUE;
    }
    //不保存
    printf("保存失败！\n");
    free(pclass);
    putchar('\n');
    putchar('\n');

    return TRUE;
}
/**查询客房分类信息*/
BOOL Queryclass()
{
    char *plabel_name[] = { "主菜单项：数据查询",
        "子菜单项：查询客房分类信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    Maintain_in();
    printf("               ***************请输入客房类别（S,D,T）   ***************");
    printf("                    \n                ***************    ");
    /*获取用户输入*/
    char c = getchar();
    c = toupper(c);
    getchar();
    CLASS_NODE *pclass;
    pclass = gp_head;
    //遍历客房分类主链，查找目标节点*/
    while (pclass != NULL&&pclass->type != c)
    {
        pclass = pclass->next;
    }
    if (pclass == NULL)//没有找到
        printf("无查询结果！\n");
    else printclass(pclass);//找到
    putchar('\n');
    return TRUE;
```

```
}
/**修改客房分类信息*/
BOOL Maintainclass()
{
    char *plabel_name[] = { "主菜单项：数据维护",
        "子菜单项：修改客房分类信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    printf("               ***************请输入要修改的客房类别（S,D,T）
***************");
    printf("                    \n               ***************    ");
    char type = getchar();
    getchar();
    type = toupper(type);
    CLASS_NODE *pclass;
    CLASS_NODE *pclass1;
    pclass = (CLASS_NODE *)malloc(sizeof(CLASS_NODE));
    for (pclass1 = gp_head; pclass1 != NULL&&pclass1->type != type; pclass1 =
pclass1->next);
    if (pclass1 == NULL)
    {//客房类别不存在，结束函数
        printf("客房类别不存在！");
        return TRUE;
    }
    *pclass = *pclass1;
    printf("               ===============修改最多入住人数请按“1”
===============\n");
    printf("               ===============修改客房套数请按“2”
===============\n");
    printf("               ***************     ");
    char key = getchar();
    while (key != '1'&&key != '2')
    {
        printf("输入错误！\n");
        key = getchar();
    }
    getchar();
    /*修改客房类别信息*/
    switch (key)
    {
    case '1':
```

```
    {
        printf("                ==============请重新输入最多入住人数,当前为%d
==============\n", pclass->pnum);
        printf("                ==============");
        scanf("%d", &pclass->pnum);
        getchar();
        break;

    }
    case '2':
    {
        printf("                ==============请重新输入客房套数,当前为%d
==============\n", pclass->rnum);
        printf("                ==============");
        scanf("%d", &pclass->rnum);
        getchar();
        break;
    }
    default:
    {
        printf("输入错误！\n");
        break;
    }

    }
    printf("修改后的结果为：\n");
    printclass(pclass);
    printf("是否确认?（y或者n）\n");
    char ch = getchar();
    ch = toupper(ch);
    getchar();
    if (ch == 'Y')//进行保存
    {
        *pclass1 = *pclass;
        printf("修改成功！\n");
        printclass(pclass1);
    }
    //不保存
    free(pclass);
    return TRUE;
}
/**删除客房分类信息*/
BOOL Delclass()
{
```

```
    char *plabel_name[] = { "主菜单项：数据维护",
        "子菜单项：删除客房分类信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    printf("\n\n                ***************警告！删除客房分类信息会删除该类别下的客房基本信息和租客信息！！请谨慎操作");
    printf("                    \n                ***************    ");
    printf("退出请输入'n',输入其他键继续");
    char c;
    printf("                    \n                ***************    ");
    c = getchar();
    if (c == 'n')
        return TRUE;
    fflush(stdin);
    printf("                    \n                ***************    ");
    printf("请输入要删除的客房类别(S,D,T)：");
    printf("                    \n                ***************    ");
    char type = getchar();
    type = toupper(type);
    getchar();
    CLASS_NODE *pclass;
    CLASS_NODE *pb;
    ROOM_NODE *proom;
    RENTER_NODE *prenter;
    pclass = gp_head;
    for (pclass = gp_head; (pclass != NULL) && (pclass->type != type); pb = pclass, pclass = pclass->next);
    if (pclass == gp_head)//如果节点位于链头
        gp_head = pclass->next;
    else pb->next = pclass->next;//如果节点位于链中
    if (pclass == NULL)//如果节点不存在
    {
        printf("客房类别不存在！");
        return TRUE;
    }
    /*释放客房分类信息下的支链*/
    proom = pclass->rnext;
    while (proom != NULL)
    {
        prenter = proom->rnext;
        while (prenter != NULL)
```

```
        {
            free(prenter);
            prenter = prenter->next;
        }
        free(proom);
        proom = proom->next;
    }
    free(pclass);//释放完毕
    printf("               ***************    ");
    printf("删除成功！\n");
    return TRUE;
}
/*=======================================*/
/*=======================================*/
/**插入客房基本信息*/
BOOL Insertroom()
{
    char *plabel_name[] = { "主菜单项：数据维护",
        "子菜单项：插入客房基本信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    CLASS_NODE *pclass;
    CLASS_NODE *hd;
    hd = gp_head;
    /*录入客房进本信息*/
    ROOM_NODE *proom, *pr;
    proom = (ROOM_NODE *)malloc(sizeof(ROOM_NODE));
    printf("               ***************请输入客房编号
***************");
    printf("                    \n               ***************    ");
    gets(proom->room_id);
    printf("               ***************请输入电话号码
***************");
    printf("                    \n               ***************    ");
    gets(proom->tel);
    printf("               ***************请输入客房类别
***************");
    printf("                    \n               ***************    ");
    scanf("%c", &proom->type);
    proom->type = toupper(proom->type);
    getchar();
```

```
    printf("               **********请输入客房面积
**********");
    printf("                    \n              **********    ");
    scanf("%f", &proom->area);
    getchar();
    printf("               **********请输入每月租金
**********");
    printf("                    \n              **********    ");
    scanf("%f", &proom->deposit);
    getchar();
    printf("               **********请输入是否有阳台(1有，0无)
**********");
    printf("                    \n              **********    ");
    scanf("%c", &proom->balcony);
    getchar();
    printf("               **********请输入是否有客人入住(y有，n无)
**********");
    printf("                    \n              **********    ");
    scanf("%c", &proom->in);
    getchar();
    printroom(proom);
    proom->rnext = NULL;
    /*是否保存*/
    printf("               **********是否保存？(y或n)
**********");
    printf("                    \n              **********    ");
    char c = getchar();
    getchar();
    if (c == 'y' || c == 'Y')
    {//保存
        for (pclass = hd; pclass != NULL&&proom->type != pclass->type; pclass =
pclass->next);
        if (pclass == NULL)
        {
            printf("客房类别不存在！");
            return TRUE;
        }
        /*查找客房编号是否存在*/
        pr = seekroom(gp_head, proom->room_id);
        if (pr != NULL)
        {//存在修改原来的客房信息，然后退出*/
         /*修改数据域*/
            proom->next = pr->next;
            proom->rnext = pr->rnext;
```

```
            *pr = *proom;
            free(proom);
            printf("客房编号已存在，修改成功！");
            return TRUE;
        }
        proom->next = pclass->rnext;
        pclass->rnext = proom;

        printf("保存成功！\n");
        return TRUE;
    }
    //不保存
    printf("保存失败！\n");
    free(proom);
    putchar('\n');
    putchar('\n');

}
/**修改客房基本信息*/
BOOL Maintainroom()
{
    char *plabel_name[] = { "主菜单项：数据维护",
        "子菜单项：修改客房基本信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    printf("              **************请输入要修改的客房编号  **************");
    printf("                    \n              **************    ");
    char roomid[8];
    gets(roomid);

    ROOM_NODE *proom;
    ROOM_NODE *temp;
    temp = (ROOM_NODE *)malloc(sizeof(ROOM_NODE));
    proom = seekroom(gp_head, roomid);

    *temp = *proom;
    printroom(proom);
    printf("              ==============修改客房编号请按“1”
==============\n");
    printf("              ==============修改客房电话请按“2”
==============\n");
```

```
    printf("              ==============修改客房类别请按“3”
==============\n");
    printf("              ==============修改客房面积请按“4”
==============\n");
    printf("              ==============修改每月租金请按“5”
==============\n");
    printf("              ==============修改客房阳台请按“6”
==============\n");
    printf("              ***************     ");
    char key = getchar();
    while (key<'1' || key>'6')
    {
        printf("输入错误，请重新输入\n");
        key = getchar();
    }
    getchar();
    switch (key)
    {
    case '1':
    {
        printf("              ==============请重新输入客房编号
==============\n");
        printf("              ==============");
        gets(temp->room_id);
        break;

    }
    case '2':
    {
        printf("              ==============请重新输入客房电话
==============\n");
        printf("              ==============");
        gets(temp->tel);
        break;
    }
    case '3':
    {
        printf("              ==============请重新输入客房类别
==============\n");
        printf("              ==============");
        temp->type = getchar();
        getchar();
        break;
    }
```

```
    case '4':
    {
        printf("                ==============请重新输入客房面积
==============\n");
        printf("                ==============");
        scanf("%f", &temp->area);
        getchar();
        break;

    }
    case '5':
    {
        printf("                ==============请重新输入每月租金
==============\n");
        printf("                ==============");
        scanf("%f", &temp->deposit);
        getchar();
        break;

    }
    case '6':
    {
        printf("                ==============请重新输入是否有阳台(有输入1，无输入0)
==============\n");
        printf("                ==============");
        temp->balcony = getchar();
        getchar();
        break;

    }
    default:
    {
        printf("输入错误！\n");
        break;
    }

    }
    printf("修改后的结果为：\n");
    printroom(temp);
    printf("是否确认?（y或者n）\n");
    char ch = getchar();
    ch = toupper(ch);
    getchar();
    if (ch == 'Y')
```

```
    {
        *proom = *temp;
        printf("修改成功！\n");
        printroom(proom);
    }
    free(temp);
    return TRUE;
}
/**查询客房基本信息*/
BOOL Queryroom()
{
    char *plabel_name[] = { "主菜单项：数据查询",
        "子菜单项：查询客房基本信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    Maintain_in();
    CLASS_NODE *pclass;
    ROOM_NODE *proom;
    printf("               ==============根据客房编号查询请按“1”
==============\n");
    printf("               ==============根据客房类别和押金查询请按“2”
==============\n");
    printf("               ***************     ");
    char key = getchar();
    while (key != '1'&&key != '2')
    {
        printf("输入错误，请重新输入\n");
        key = getchar();
    }
    getchar();
    switch (key)
    {
    case '1':
    {
        printf("               ***************     请输入客房编号\n");
        printf("               ***************     ");
        char r_id[10];
        gets(r_id);
        for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
        {
            proom = pclass->rnext;
```

```
            while (proom != NULL)
            {
                if (strcmp(proom->room_id, r_id) == 0)
                {
                    printroom(proom);
                    printf("\n\n");
                    return TRUE;
                }
                proom = proom->next;
            }
        }
        printf("无结果!\n");
        printf("\n\n");

        return TRUE;

    }
    case '2':
    {
        printf("               ***************     请输入客房类别\n");
        printf("               ***************     ");
        char type = getchar();
        type = toupper(type);
        getchar();
        for (pclass = gp_head; pclass != NULL&&pclass->type != type; pclass =
pclass->next);
        if (pclass == NULL)
        {
            printf("               ***************     ");
            printf("无结果!\n\n");
            return TRUE;
        }
        printf("               ***************     请输入客房租金\n");
        float money;
        int find = 0;
        printf("               ***************     ");
        scanf("%f", &money);
        getchar();
        proom = pclass->rnext;
        while (proom != NULL)
        {
            if (fabs(money - proom->deposit)<1)
            {
                printroom(proom);
```

```
                find = 1;
            }
            proom = proom->next;
        }
        if (find == 0)
        {
            printf("              ***************     ");
            printf("没有找到\n");
        }
        printf("\n\n");
        return TRUE;


    }
    default:
    {
        printf("输入错误！\n");
        break;
    }

    }

}
/**删除客房基本信息*/
BOOL Delroom()
{
    char *plabel_name[] = { "主菜单项：数据维护",
        "子菜单项：删除客房基本信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    printf("\n\n              ***************警告！删除客房基本信息会删除该类别下租客
信息！！请谨慎操作");
    printf("                    \n              ***************    ");
    printf("退出请输入'n',输入其他键继续");
    char c;
    c = getchar();
    if (c == 'n')
        return TRUE;
    fflush(stdin);
    printf("                    \n              ***************    ");
    printf("请输入要删除的客房编号：");
```

```
    printf("                    \n              ***************    ");
    char room_id[4];
    gets(room_id);
    CLASS_NODE *pclass;
    ROOM_NODE *proom;
    ROOM_NODE *pr;//记录proom指向节点的前一个节点
    RENTER_NODE *prenter;
    for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
    {
        proom = pclass->rnext;
        while (proom != NULL)
        {
            if (strcmp(proom->room_id, room_id) == 0)//如果找到,改变链表指向关系
            {
                if (proom == pclass->rnext)//如果节点位于头部
                {
                    pclass->rnext = proom->next;
                }
                else //如果节点位于链中
                {
                    pr->next = proom->next;
                }
                /*删除节点*/
                prenter = proom->rnext;
                while (prenter != NULL)
                {
                    free(prenter);
                    prenter = prenter->next;
                }
                printroom(proom);
                free(proom);
                printf("删除成功！");
                return TRUE;
            }
            pr = proom;
            proom = proom->next;
        }

    }
    printf("未找到!\n");
    return TRUE;

}
/*=======================================*/
```

```
/*=====================================*/
/**插入租客信息*/
BOOL Insertrenter()
{
    char *plabel_name[] = { "主菜单项：数据维护",
        "子菜单项：插入租客信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    ROOM_NODE *proom;
    RENTER_NODE *prenter, *pr;
    prenter = (RENTER_NODE *)malloc(sizeof(RENTER_NODE));
    /*录入租客信息*/
    printf("               **********请输入身份证号（18位）  **********");
    printf("                    \n              **********    ");
    gets(prenter->id_card);
    while (strlen(prenter->id_card) != 18)
    {
        printf("                    \n              **********    ");
        printf("身份证号错误，请重新输入！\n");
        printf("                    \n              **********    ");
        gets(prenter->id_card);
    }
    printf("               **********请输入客人姓名  **********");
    printf("                    \n              **********    ");
    gets(prenter->name);
    printf("               **********请输入入住客房编号  **********");
    printf("                    \n              **********    ");
    gets(prenter->room_id);
    printf("               **********请输入入住时间(如2015/03/05-13:00)
**********");
    printf("                    \n              **********    ");
    gets(prenter->date_in);
    printf("               **********请输入退房时间(在住输入回车)
**********");
    printf("                    \n              **********    ");
    gets(prenter->date_out);
    prenter->months = Maintain_month(prenter->date_in, prenter->date_out);
    proom = seekroom(gp_head, prenter->room_id);
    if (proom == NULL)
        return TRUE;
    prenter->deposit = proom->deposit;
```

```
if (prenter->date_out[0] != '\0')//如果客人已经退房
{
    prenter->shouldpay = proom->deposit*prenter->months;
    printf("                    \n              ***************    ");
    printf("入住月数为%.1f\n", prenter->months);
    printf("                    \n              ***************    ");
    printf("应缴费用为%.1f\n", prenter->shouldpay);
    printf("                    \n              ***************    ");
    printf("请输入实缴费用\n");
    printf("                    \n              ***************    ");
    scanf("%f", &prenter->realpay);
    getchar();
}
else {
    prenter->months = prenter->shouldpay = prenter->realpay = 0;
}
printrenter(prenter);
printf("               ***************是否保存？(y或n)   ***************");
printf("                    \n              ***************    ");
char c = getchar();
getchar();
if (c == 'y' || c == 'Y')
{
    proom = seekroom(gp_head, prenter->room_id);
    if (proom == NULL)
    {
        printf("未找到客房！");
        return TRUE;
    }
    pr = seekrenter(gp_head, prenter->date_in, prenter->id_card);
    /*判断租客信息是否已经存在*/
    if (pr != NULL&&strcmp(pr->room_id, prenter->room_id) == 0)
    {//若租客个人信息，入住日期，入住客房编号相同则租客信息存在
     /*修改数据域*/
        prenter->next = pr->next;
        *pr = *prenter;
        free(prenter);
        printf("客人信息已存在,修改成功！");
        return TRUE;
    }
    prenter->next = proom->rnext;
    proom->rnext = prenter;
    printf("保存成功！\n");
    proom->in = 'y';
```

```
        fflush(stdin);
        return TRUE;
    }
    printf("保存失败！\n");
    free(prenter);
    putchar('\n');
    putchar('\n');
    putchar('\n');

    return TRUE;
}
/**查询租客信息*/
BOOL Queryrenter()
{
    char *plabel_name[] = { "主菜单项：数据查询",
        "子菜单项：查询租客信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    CLASS_NODE *pclass;
    ROOM_NODE *proom;
    RENTER_NODE *prenter;
    /*获取用户输入*/
    printf("               ===============根据租客身份证号查询请按“1”
===============\n");
    printf("               ===============根据客人的姓和入住时间查询请按“2”
===============\n");
    printf("               ===============根据客人的名字和入住时间查询请按“3”
===============\n");
    printf("               ***************     ");
    char key = getchar();
    while (key != '1'&&key != '2'&&key != '3')
    {
        printf("输入错误，请重新输入\n");
        key = getchar();
    }
    getchar();
    int find = 0;
    switch (key)
    {
    case '1':
    {
```

```
        printf("              **************     请输入租客身份证号\n");
        printf("              **************     ");
        char id_card[20];
        gets(id_card);
        while (strlen(id_card) != 18)
        {
            printf("                    \n              **************     ");
            printf("身份证号错误，请重新输入！\n");
            printf("                    \n              **************     ");
            gets(id_card);
        }
        for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
        {
            proom = pclass->rnext;
            while (proom != NULL)
            {
                prenter = proom->rnext;
                while (prenter != NULL)
                {
                    if (strcmp(prenter->id_card, id_card) == 0)//如果找到该身份证号
                    {
                        printrenter(prenter);//输出租客信息
                    }
                    prenter = prenter->next;
                }
                proom = proom->next;
            }
        }


        return TRUE;

    }
    case '2':
    {
        /*获取搜索信息*/
        printf("              **************     请输入客人的姓\n");
        printf("              **************     ");
        char firstname[3];
        gets(firstname);
        printf("              **************     请输入入住开始时间(时间格式为
2015/05/07)\n");
        printf("              **************     ");
        char date1[18];
```

```
        gets(date1);
        printf("               **************     请输入入住结束时间(时间格式为
2015/05/07)\n");
        printf("               **************     ");
        char date2[18];
        gets(date2);
        for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
        {

            for (proom = pclass->rnext; proom != NULL; proom = proom->next)
            {

                for (prenter = proom->rnext; prenter != NULL; prenter =
prenter->next)
                {

                    if (strncmp(prenter->name, firstname, 2) == 0 && Judgedate(date1,
date2, prenter->date_in) == 1)//如果客人的姓符合并且日期在给定范围之内
                        printrenter(prenter);//输出租客信息
                }
            }
        }

        return TRUE;

    }
    case '3':
    {
        int find = 0;
        printf("               **************     请输入客人的姓名\n");
        printf("               **************     ");
        char name[3];
        gets(name);
        printf("               **************     请输入入住开始时间(时间格式为
2015/05/07)\n");
        printf("               **************     ");
        char date1[18];
        gets(date1);
        printf("               **************     请输入入住结束时间(时间格式为
2015/05/07)\n");
        printf("               **************     ");
        char date2[18];
        gets(date2);
        for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
```

```
        {

            for (proom = pclass->rnext; proom != NULL; proom = proom->next)
            {

                for (prenter = proom->rnext; prenter != NULL; prenter =
prenter->next)
                {

                    if (strcmp(prenter->name, name) == 0 && Judgedate(date1, date2,
prenter->date_in) == 1)//如果客人的姓名符合并且日期在给定范围之内
                    {
                        printrenter(prenter); find = 1;
                    }
                }
            }
        }
        if (find == 0)
            printf("未找到该租客\n");
        return TRUE;
    }
    default:
    {
        printf("输入错误！\n");
        break;
    }

    }

}
/**修改租客信息*/
BOOL Maintainrenter()
{
    char *plabel_name[] = { "主菜单项：数据维护",
        "子菜单项：修改租客信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    printf("              **************请输入租客身份证号  **************");
    printf("                    \n              **************    ");
    char id_card[20];
    gets(id_card);
```

```
    while (strlen(id_card) != 18)
    {
        printf("                    \n              ***************    ");
        printf("身份证号错误，请重新输入！\n");
        printf("                    \n              ***************    ");
        gets(id_card);
    }
    printf("              ***************请输入租客入住日期（精确到日，如2015/05/06,修
改在住可输入回车）  ***************");
    printf("                    \n              ***************    ");
    char date[18];
    gets(date);
    while (strlen(date) != 10)
    {
        printf("                    \n              ***************    ");
        printf("日期输入错误，请重新输入！\n");
        printf("                    \n              ***************    ");
        gets(date);
    }
    CLASS_NODE *pclass;
    ROOM_NODE *proom;
    RENTER_NODE *prenter;
    RENTER_NODE *temp;
    temp = (RENTER_NODE *)malloc(sizeof(RENTER_NODE));

    prenter = seekrenter(gp_head, date, id_card);
    if (prenter == NULL)
    {
        return TRUE;
    }
    printrenter(prenter);
    *temp = *prenter;
    printf("              ==============修改身份证号请按“1”
==============\n");
    printf("              ==============修改客人姓名请按“2”
==============\n");
    printf("              ==============修改入住客房编号请按“3”
==============\n");
    printf("              ==============修改入住时间请按“4”
==============\n");
    printf("              ==============修改退房时间请按“5”
==============\n");
    printf("              ==============修改押金请按“6”
==============\n");
```

```
    printf("              ==============修改应缴费用请按“7”
==============\n");
    printf("              ==============修改实缴费用请按“8”
==============\n");
    printf("              ***************     ");
    char key = getchar();
    while (key<'1' || key>'9')
    {
        printf("输入错误，请重新输入\n");
        key = getchar();
    }
    getchar();
    switch (key)
    {
    case '1':
    {
        printf("              ==============请重新输入身份证号
==============\n");
        printf("              ==============");
        gets(temp->id_card);
        break;

    }
    case '2':
    {
        printf("              ==============请重新输入客人姓名
==============\n");
        printf("              ==============");
        gets(temp->name);
        break;
    }
    case '3':
    {
        printf("              ==============请重新输入客房编号
==============\n");
        printf("              ==============");
        gets(temp->room_id);
        break;
    }
    case '4':
    {
        printf("              ==============请重新输入入住时间
==============\n");
        printf("              ==============");
```

```
        gets(temp->date_in);
        break;

    }
    case '5':
    {
        printf("              ==============请输入退房时间      ==============\n");
        printf("              ==============");
        gets(temp->date_out);
        temp->months = Maintain_month(temp->date_in, temp->date_out);
        temp->months = Maintain_month(temp->date_in, temp->date_out);
        break;
    }
    case '6':
    {
        printf("              ==============请重新输入押金      ==============\n");
        printf("              ==============");
        scanf("%f", &temp->deposit);
        getchar();
        break;
    }
    case '7':
    {
        printf("              ==============请重新输入应缴费用
==============\n");
        printf("              ==============");
        scanf("%f", &temp->shouldpay);
        getchar();

    }
    case '8':
    {
        printf("              ==============请重新输入实缴费用
==============\n");
        printf("              ==============");
        scanf("%f", &temp->realpay);
        getchar();
        break;
    }

    default:
    {
        printf("输入错误！\n");
        break;
```

```
    }

    }
    printf("修改后的结果为：\n");
    printrenter(temp);
    printf("是否确认?（y或者n）\n");
    char ch = getchar();
    ch = toupper(ch);
    getchar();
    if (ch == 'Y')
    {
        if (key == '3')
        {
            proom = seekroom(gp_head, prenter->room_id);
            proom->in = 'n';
            proom = seekroom(gp_head, temp->room_id);
            proom->in = 'y';
        }
        if (key == '5'&&prenter->date_out[0] == '\0')
        {
            proom = seekroom(gp_head, prenter->room_id);
            proom->in = 'n';
        }
        *prenter = *temp;
        printf("修改成功！\n");
        printrenter(prenter);
        return TRUE;
    }
    free(temp);
    printf("保存失败\n");
    return TRUE;
}
/**删除租客信息*/
BOOL Delrenter()
{
    char *plabel_name[] = { "主菜单项：数据维护",
        "子菜单项：删除租客信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    printf("                    \n              **********    ");
    printf("请输入要删除的租客身份证号：");
```

```
    printf("                    \n              ***************    ");
    char id_card[20];
    gets(id_card);
    while (strlen(id_card) != 18)
    {
        printf("                    \n              ***************    ");
        printf("身份证号错误，请重新输入！\n");
        printf("                    \n              ***************    ");
        gets(id_card);
    }
    printf("                    \n              ***************    ");
    printf("请输入租客入住日期(如2015/06/05)");
    printf("                    \n              ***************    ");
    char date[18];
    gets(date);
    ROOM_NODE *proom;
    RENTER_NODE *prenter;
    RENTER_NODE *pb;
    prenter = seekrenter(gp_head, date, id_card);
    proom = seekroom(gp_head, prenter->room_id);
    if (prenter == proom->rnext)
    {//如果节点位于支链头部
        proom->rnext = prenter->next;
        free(prenter);
        return TRUE;
    }
    /*节点位于支链内部*/
    prenter = proom->rnext;
    pb = proom->rnext;
    while (prenter != NULL)
    {

        if (strncmp(prenter->date_in, date, 10) == 0 && strcmp(prenter->id_card,
id_card) == 0)
        {//释放内存空间
            pb->next = prenter->next;
            free(prenter);
            printf("删除成功！\n");
            return TRUE;
        }
        pb = prenter;
        prenter = prenter->next;
    }
    printf("未找到");
```

```
    return TRUE;
}
/*=======================================*/
/*=======================================*/
/**统计统计每种类别的客房总数、入住数、未住数*/
BOOL Staclass()
{
    char *plabel_name[] = { "主菜单项：数据统计",
        "子菜单项：统计统计每种类别的客房总数、入住数、未住数",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    Maintain_in();//更新客房入住情况
    int allroom, inroom, emptyroom;//分别为客房总数，已入住数，未入住数
    int totalroom, totalinroom, totalemptyroom;//合计的客房客房总数，已入住数，未入住数
    totalroom = totalinroom = totalemptyroom = 0;
    CLASS_NODE *pclass;

    for (pclass = gp_head; pclass != NULL&&pclass->type != 'S'; pclass =
pclass->next);//遍历主链，pclass指向单人间客房类别
    allroom = pclass->rnum;
    emptyroom = pclass->emptyr;
    inroom = allroom - emptyroom;
    totalroom += allroom;
    totalinroom += inroom;
    totalemptyroom += emptyroom;
    printf("+-------------------------------------------------------------------------+\n");
    printf("+        客房类别        客房总数        已入住数        未入住数  +\n");
    printf("+-------------------------------------------------------------------------+\n");
    printf("+        单人间  %16d%16d%16d  +\n", allroom, inroom, emptyroom);

    for (pclass = gp_head; pclass != NULL&&pclass->type != 'D'; pclass =
pclass->next);//遍历主链，pclass指向双人间客房类别
    allroom = pclass->rnum;
    emptyroom = pclass->emptyr;
    inroom = allroom - emptyroom;
    totalroom += allroom;
    totalinroom += inroom;
    totalemptyroom += emptyroom;
    printf("+-------------------------------------------------------------------------+\n");
    printf("+        双人间  %16d%16d%16d  +\n", allroom, inroom, emptyroom);
    printf("+-------------------------------------------------------------------------+\n");
```

```
    for (pclass = gp_head; pclass != NULL&&pclass->type != 'T'; pclass =
pclass->next);//遍历主链，pclass指向三人间客房类别
    allroom = pclass->rnum;
    emptyroom = pclass->emptyr;
    inroom = allroom - emptyroom;
    totalroom += allroom;
    totalinroom += inroom;
    totalemptyroom += emptyroom;
    printf("+        三人间  %16d%16d%16d  +\n", allroom, inroom, emptyroom);
    printf("+-------------------------------------------------------------------------+\n");
    printf("+        合计    %16d%16d%16d  +\n", totalroom, totalinroom,
totalemptyroom);
    printf("+-------------------------------------------------------------------------+\n");

    return TRUE;
}
/**按月统计本年度各类客房的营业额*/
BOOL Staincome()
{
    char *plabel_name[] = { "主菜单项：数据统计",
        "子菜单项：按月统计本年度各类客房的营业额",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    INCOME_NODE *sroom, *droom, *troom;//单人间，双人间，三人间
                                       /*动态分配存储空间*/
    sroom = (INCOME_NODE *)malloc(sizeof(INCOME_NODE));
    droom = (INCOME_NODE *)malloc(sizeof(INCOME_NODE));
    troom = (INCOME_NODE *)malloc(sizeof(INCOME_NODE));
    /*将收入结构中个sale数组置为0*/
    int i;
    for (i = 0; i<12; i++)
    {
        sroom->sale[i] = 0;
        droom->sale[i] = 0;
        troom->sale[i] = 0;
    }
    sroom->next = droom;
    droom->next = troom;
    troom->next = NULL;
```

```
    /*处理用户输入*/
    printf("请输入将要统计的年份");
    char year[18];
    gets(year);

    /*开始统计收入*/
    CLASS_NODE *pclass;
    ROOM_NODE *proom;
    RENTER_NODE *prenter;
    /*统计单人间*/
    for (pclass = gp_head; pclass != NULL&&pclass->type != 'S'; pclass =
pclass->next);//pclass定位单人间类别
    if (pclass != NULL)//如果找到单人间类别
    {
        for (proom = pclass->rnext; proom != NULL; proom = proom->next)
            for (prenter = proom->rnext; prenter != NULL; prenter = prenter->next)
            {
                /**如果年份相同，进行统计，租客退房时收取费用，押金不计入收入*/
                if (strncmp(year, prenter->date_out, 4) == 0)
                {
                    /*提取日期中的月数*/
                    char months[3];
                    months[0] = prenter->date_out[5];
                    months[1] = prenter->date_out[6];
                    months[2] = '\0';
                    int month = atoi(months);//将月份从字符转换为数字
                    sroom->sale[month - 1] += prenter->realpay;//将收入加到对应的月中
                }
            }
    }
    /*统计双人间*/
    for (pclass = gp_head; pclass != NULL&&pclass->type != 'D'; pclass =
pclass->next);//pclass定位双人间类别
    if (pclass != NULL)//如果找到双人间类别
    {
        for (proom = pclass->rnext; proom != NULL; proom = proom->next)
            for (prenter = proom->rnext; prenter != NULL; prenter = prenter->next)
            {
                /**如果年份相同，进行统计，租客退房时收取费用，押金不计入收入*/
                if (strncmp(year, prenter->date_out, 4) == 0)
                {
                    /*提取日期中的月数*/
                    char months[3];
                    months[0] = prenter->date_out[5];
```

```
                    months[1] = prenter->date_out[6];
                    months[2] = '\0';
                    int month = atoi(months);//将月份从字符转换为数字
                    droom->sale[month - 1] += prenter->realpay;//将收入加到对应的月中
                }
            }
    }
    /*统计三人间*/
    for (pclass = gp_head; pclass != NULL&&pclass->type != 'T'; pclass =
pclass->next);//pclass定为三人间类别
    if (pclass != NULL)//如果找到三人间类别
    {
        for (proom = pclass->rnext; proom != NULL; proom = proom->next)
            for (prenter = proom->rnext; prenter != NULL; prenter = prenter->next)
            {
                /**如果年份相同，进行统计，租客退房时收取费用，押金不计入收入*/
                if (strncmp(year, prenter->date_out, 4) == 0)
                {
                    /*提取日期中的月数*/
                    char months[3];
                    months[0] = prenter->date_out[5];
                    months[1] = prenter->date_out[6];
                    months[2] = '\0';
                    int month = atoi(months);//将月份从字符转换为数字
                    sroom->sale[month - 1] += prenter->realpay;//将收入加到对应的月中
                }
            }
    }
    /*输出结果*/
    char *p[4];
    p[0] = "月份";
    p[1] = "单人间";
    p[2] = "双人间";
    p[3] = "三人间";
    printf("    %10s    %10s    %10s    %10s    \n", p[0], p[1], p[2], p[3]);
    for (i = 0; i<12; i++)
    {
        printf("    %10.1d    %10.1f    %10.1f    %10.1f    \n", i + 1,
sroom->sale[i], droom->sale[i], troom->sale[i]);
    }
    /*释放内存*/
    free(sroom);
    free(droom);
    free(troom);
```

```
    return TRUE;
}
/**输入年份，统计该年所有客房的营业额、入住月数、入住率*/
BOOL Staroom_info()
{
    char *plabel_name[] = { "主菜单项：数据统计",
        "子菜单项：统计一年所有客房的营业额、入住月数、入住率",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    ROOM_INFO *gp_room_info = NULL;                    /*客房信息统计链链头指针*/
    CLASS_NODE *pclass;
    ROOM_NODE *proom;
    RENTER_NODE *prenter;
    ROOM_INFO *proominfo;

    /*处理用户输入*/
    printf("请输入将要统计的年份");
    char year[18];
    gets(year);

    for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
    {
        for (proom = pclass->rnext; proom != NULL; proom = proom->next)
        {
            /*创建统计链表节点*/
            proominfo = (ROOM_INFO *)malloc(sizeof(ROOM_INFO));
            proominfo->next = gp_room_info;


            /*输出统计信息*/
            char *pt[5];
            pt[0] = "客房编号";
            pt[1] = "客房类别";
            pt[2] = "营业额  ";
            pt[3] = "入住月数";
            pt[4] = "入住率  ";
            printf("    %-10s    %-10s    %-10s    %-10s    %-10s    \n", pt[0],
pt[1], pt[2], pt[3], pt[4]);
            for (proominfo = gp_room_info; proominfo != NULL; proominfo =
proominfo->next)
                printf("    %-10s    %-10c    %-10.1f    %-10.1f    %-10.2f\n",
```

```
proominfo->room_id, proominfo->type,
                    proominfo->income, proominfo->months, proominfo->in_rate);
            /*释放统计节点*/
            for (proominfo = gp_room_info; proominfo != NULL; proominfo =
proominfo->next)
                free(proominfo);
            return TRUE;
        }
        /**统计出租月数最多的 10 个客人租房信息*/
        BOOL Starenter_info()
        {
            char *plabel_name[] = { "主菜单项：数据统计",
                "子菜单项：统计出租月数最多的 10 个客人租房信息",
                "确认"
            };

            ShowModule(plabel_name, 3);
            Clear();
            CLASS_NODE *pclass;
            ROOM_NODE *proom;
            RENTER_NODE *prenter;
            RENTER_INFO *gp_renter_head, *prenter_info;
            gp_renter_head = NULL;
            for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
            {
                for (proom = pclass->rnext; proom != NULL; proom = proom->next)
                {
                    for (prenter = proom->rnext; prenter != NULL; prenter =
prenter->next)
                    {
                        if (prenter->date_out[0] != '\0')//在住租客不参与统计
                        {

                            if ((prenter_info = search_renter_info(gp_renter_head,
prenter->id_card)) == NULL)//如果该身份证号未在统计链表中
                            {
                                prenter_info = (RENTER_INFO
*)malloc(sizeof(RENTER_INFO));//那么新创建一个节点

                /*赋值节点*/
                                prenter_info->allmonths = prenter->months;
                                prenter_info->realpay = prenter->realpay;
                                prenter_info->shouldpay = prenter->shouldpay;
                                strcpy(prenter_info->id_card, prenter->id_card);
```

```
                    strcpy(prenter_info->name, prenter->name);
                    /*插入节点*/
                    prenter_info->next = gp_renter_head;
                    gp_renter_head = prenter_info;
                }
                else//如果该身份证号已经存在于统计链表中
                {
                    /*统计*/
                    prenter_info->allmonths += prenter->months;
                    prenter_info->realpay += prenter->realpay;
                    prenter_info->shouldpay += prenter->shouldpay;
                }
            }
        }
    }
}//统计完毕
 /*按照租房总月数降序排序*/
sortrenter_info(gp_renter_head);
/*输出累计租房月数排名前十的统计结果*/
int amount;
char *p[5];
p[0] = " 身份证号 ";
p[1] = "姓名";
p[2] = "累计住宿月数";
p[3] = "应缴费用总额";
p[4] = "实缴费用总额";
printf("%-20s    %-15s    %-15s    %-15s    %-15s    \n", p[0], p[1],
p[2], p[3], p[4]);
for (amount = 1, prenter_info = gp_renter_head; amount <= 10 &&
prenter_info != NULL; amount++, prenter_info = prenter_info->next)
{
    printf("%-20s    %-15s    %-15.1f    %-15.1f    %-15.1f    \n",
prenter_info->id_card, prenter_info->name,
        prenter_info->allmonths, prenter_info->shouldpay,
prenter_info->realpay);
}
return TRUE;
}
/**统计当前所有客房租金及入住情况*/
BOOL Sta_in()
{
    char *plabel_name[] = { "主菜单项：数据统计",
        "子菜单项：统计当前所有客房租金及入住情况",
        "确认"
```

```
        };

        ShowModule(plabel_name, 3);
        Clear();
        CLASS_NODE *pclass;
        ROOM_NODE *proom;

        char *p[4];
        p[0] = "客房类型";
        p[1] = "客房编号";
        p[2] = "客房租金";
        p[3] = "入住情况";
        char *ch[2];
        ch[0] = "有人";
        ch[1] = "未入住";
        char *pcode[3];
        pcode[0] = "单人间";
        pcode[1] = "双人间";
        pcode[2] = "三人间";
        int key;
        printf("    %-10s    %-10s    %-10s    %-10s    \n", p[0], p[1], p[2],
p[3]);
        for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
        {
            for (proom = pclass->rnext; proom != NULL; proom = proom->next)
            {
                if (proom->type == 'S')
                    key = 0;
                if (proom->type == 'D')
                    key = 1;
                if (proom->type == 'T')
                    key = 2;
                if (proom->in == 'y')
                    printf("    %-10s    %-10s    %-10.1f    %-10s    \n",
pcode[key], proom->room_id, proom->deposit, ch[0]);
                else printf("    %-10s    %-10s    %-10.1f    %-10s    \n",
pcode[key], proom->room_id, proom->deposit, ch[1]);

            }
        }
        return TRUE;
    }
    /*=======================================*/
    BOOL SaveData()
```

```
{
    BOOL bRet = TRUE;
    char *plabel_name[] = { "主菜单项：文件",
        "子菜单项：数据保存",
        "确认"
    };

    ShowModule(plabel_name, 3);
    CLASS_NODE * pclass;
    ROOM_NODE *proom;
    RENTER_NODE * prenter;
    FILE * pfout;

    Maintain_in();
    pfout = fopen(gp_class_info_filename, "wb");
    for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
    {
        /*保存客房分类信息*/
        fwrite(pclass, sizeof(CLASS_NODE), 1, pfout);
    }
    fclose(pfout);
    //printf("1");

    pfout = fopen(gp_room_info_filename, "wb");
    for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
    {/*保存客房基本信息*/
        proom = pclass->rnext;
        while (proom != NULL)
        {
            fwrite(proom, sizeof(ROOM_NODE), 1, pfout);
            proom = proom->next;
        }
    }
    fclose(pfout);
    //printf("2");

    pfout = fopen(gp_renter_info_filename, "wb");
    for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
    {

        proom = pclass->rnext;
        while (proom != NULL)
        {
            //printf("3") ;
```

```
            prenter = proom->rnext;

            while (prenter != NULL)
            {
                fwrite(prenter, sizeof(RENTER_NODE), 1, pfout);
                prenter = prenter->next;
            }
            proom = proom->next;
        }

    }
    fclose(pfout);
    printf("数据保存成功！\n");

    return bRet;
}
/**查找客房节点*/
ROOM_NODE * seekroom(CLASS_NODE *hd, char * r_id)
{
    CLASS_NODE * pclass;
    ROOM_NODE *proom;
    ROOM_NODE *proom1;
    int flat = 1;//用于退出循环，找到后置为0.
    for (pclass = hd; pclass != NULL&&flat; pclass = pclass->next)
    {
        proom = pclass->rnext;
        while (proom != NULL&&flat)
        {
            proom1 = proom;
            if (strcmp(r_id, proom->room_id) == 0)
                flat = 0;
            proom = proom->next;

        }
    }
    if (flat)
    {
        printf("未找到！");
        return NULL;
    }
    return proom1;
}
/**查找租客节点*/
RENTER_NODE * seekrenter(CLASS_NODE *hd, char *date, char * id_card)
```

```
        {
            CLASS_NODE *pclass;
            ROOM_NODE *proom;
            RENTER_NODE *prenter;
            for (pclass = hd; pclass != NULL; pclass = pclass->next)
            {
                proom = pclass->rnext;
                while (proom != NULL)
                {
                    prenter = proom->rnext;
                    while (prenter != NULL)
                    {
                        if ((strncmp(prenter->date_in, date, 10) == 0 || (date[0] ==
'\0'&&prenter->date_out[0] == '\0')) && strcmp(prenter->id_card, id_card) == 0)
                        {
                            return prenter;
                        }
                        prenter = prenter->next;
                    }
                    proom = proom->next;
                }

            }
            //printf("没有找到该租客");
            return NULL;
        }
        /**查找prenter_info指向链表中是否有id_card的信息，存在返回id_card所在的节点，
不存在返回NULL**/
        RENTER_INFO * search_renter_info(RENTER_INFO * hd, char *id_card)
        {
            RENTER_INFO *prenter_info;
            for (prenter_info = hd; prenter_info != NULL; prenter_info =
prenter_info->next)
            {//遍历链表
                if (strncmp(prenter_info->id_card, id_card, 18) == 0)
                    return prenter_info;//如果找到，返回该节点
            }
            return NULL;//遍历完而未找到，返回空指针
        }
        /**将租房月数统计链表按租房月数降序排序*/
        void sortrenter_info(RENTER_INFO *hd)
        {
            RENTER_INFO *prenter_info, *pr;//prenter_info遍历外层循环，pr遍历内层循环
            RENTER_INFO temp;
```

```
            RENTER_INFO *p1, *p2, *p3, *p4;//p1指向租房月数最大的节点,p2指向要交换的节点,p3,p4临时保存p1,p2的next的值

            float maxmonths;//最大租房月数
                                /*选择排序法排序，链表指向不变，每次选取最大租房月数的节点，交换其与前面节点的数据域*/
            for (prenter_info = hd; prenter_info != NULL; prenter_info = prenter_info->next)
            {
                maxmonths = prenter_info->allmonths;
                p1 = prenter_info;
                for (pr = prenter_info; pr != NULL; pr = pr->next)
                {
                    if (pr->allmonths>maxmonths)//如果pr节点的月数更大
                    {
                        p1 = pr;//p1指向月数最大的节点
                        maxmonths = pr->allmonths;
                    }

                }
                if (p1 != prenter_info)//如果p1的值变了，进行交换
                {
                    p2 = prenter_info;
                    p3 = p1->next;
                    p4 = p2->next;//保留指针
                                    /*交换p1,p2的数据域*/
                    temp = *p1;
                    *p1 = *p2;
                    *p2 = temp;
                    /*恢复指针指向*/
                    p1->next = p3;
                    p2->next = p4;
                }
            }
        }
        BOOL ExitSys(void)
        {
            LABEL_BUNDLE labels;
            HOT_AREA areas;
            BOOL bRet = TRUE;
            SMALL_RECT rcPop;
            COORD pos;
            WORD att;
            char *pCh[] = { "确认退出系统吗？", "确定    取消" };
```

```
    int iHot = 1;

    pos.X = strlen(pCh[0]) + 6;
    pos.Y = 7;
    rcPop.Left = (SCR_COL - pos.X) / 2;
    rcPop.Right = rcPop.Left + pos.X - 1;
    rcPop.Top = (SCR_ROW - pos.Y) / 2;
    rcPop.Bottom = rcPop.Top + pos.Y - 1;

    att = BACKGROUND_BLUE | BACKGROUND_GREEN | BACKGROUND_RED;  /*白底黑字*/
    labels.num = 2;
    labels.ppLabel = pCh;
    COORD aLoc[] = { { rcPop.Left + 3, rcPop.Top + 2 },
    { rcPop.Left + 5, rcPop.Top + 5 } };
    labels.pLoc = aLoc;

    areas.num = 2;
    SMALL_RECT aArea[] = { { rcPop.Left + 5, rcPop.Top + 5,
        rcPop.Left + 8, rcPop.Top + 5 },
        { rcPop.Left + 13, rcPop.Top + 5,
        rcPop.Left + 16, rcPop.Top + 5 } };
    char aSort[] = { 0, 0 };
    char aTag[] = { 1, 2 };
    areas.pArea = aArea;
    areas.pSort = aSort;
    areas.pTag = aTag;
    PopUp(&rcPop, att, &labels, &areas);

    pos.X = rcPop.Left + 1;
    pos.Y = rcPop.Top + 4;
    FillConsoleOutputCharacter(gh_std_out, '-', rcPop.Right - rcPop.Left - 1,
pos, &ul);

    if (DealInput(&areas, &iHot) == 13 && iHot == 1)
    {
        bRet = FALSE;
    }
    else
    {
        bRet = TRUE;
    }
    PopOff();

    return bRet;
```

```
}
/**查询客房基本信息字符信息*/
BOOL Query(void)
{
    char *plabel_name[] = { "主菜单项：数据查询",
        "子菜单项：查询客房基本信息字符信息",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();
    BOOL bRet = TRUE;
    printf("               ***************    0 无阳台\n");
    printf("               ***************    1 有阳台\n");
    printf("               ***************    y 有客人入住\n");
    printf("               ***************    n 无客人入住\n");
    return bRet;
}
/**查询客房类别代码*/
BOOL QueryTypeInfo(void)
{
    char *plabel_name[] = { "主菜单项：数据查询",
        "子菜单项：查找客房类别代码",
        "确认"
    };

    ShowModule(plabel_name, 3);
    Clear();

    BOOL bRet = TRUE;
    printf("               ***************    S 单人间\n");
    printf("               ***************    D 双人间\n");
    printf("               ***************    T 三人间\n");
    return bRet;
}
/**帮助*/
BOOL HelpTopic(void)
{
    BOOL bRet = TRUE;
    char *plabel_name[] = { "主菜单项：帮助",
        "子菜单项：帮助主题",
        "确认"
    };
```

```
    ShowModule(plabel_name, 3);
    Clear();
    printf("\n");
    printf("《系统快捷操作指南》\n");
    printf(" 1、Alt + F  弹出\"文件(F)\"的下拉菜单;\n");
    printf(" 2、Alt + E  弹出\"编辑(E)\"的下拉菜单;\n");
    printf(" 3、Alt + I  弹出\"查询(I)\"的下拉菜单;\n");
    printf(" 4、Alt + S  弹出\"统计(S)\"的下拉菜单;\n");
    printf(" 5、Alt + H  弹出\"帮助(H)\"的下拉菜单;\n");
    printf(" 6、ESC  键  关闭弹出窗口,回到主菜单栏;\n");
    printf(" 7、Enter键  执行相应的功能函数;\n");


    return bRet;
}
/**关于*/
BOOL AboutDorm(void)
{
    BOOL bRet = TRUE;
    char *plabel_name[] = { "主菜单项：帮助",
        "子菜单项：关于...",
        "确认",

    };
    ShowModule(plabel_name, 3);
    Clear();
    printf("              ***************    作者：  吕鹏泽\n");

    return bRet;
}

BOOL ShowModule(char **pString, int n)
{
    LABEL_BUNDLE labels;
    HOT_AREA areas;
    BOOL bRet = TRUE;
    SMALL_RECT rcPop;
    COORD pos;
    WORD att;
    int iHot = 1;
    int i, maxlen, str_len;

    for (i = 0, maxlen = 0; i<n; i++) {
        str_len = strlen(pString[i]);
```

```
        if (maxlen < str_len) {
            maxlen = str_len;
        }
    }

    pos.X = maxlen + 6;
    pos.Y = n + 5;
    rcPop.Left = (SCR_COL - pos.X) / 2;
    rcPop.Right = rcPop.Left + pos.X - 1;
    rcPop.Top = (SCR_ROW - pos.Y) / 2;
    rcPop.Bottom = rcPop.Top + pos.Y - 1;

    att = BACKGROUND_BLUE | BACKGROUND_GREEN | BACKGROUND_RED;  /*白底黑字*/
    labels.num = n;
    labels.ppLabel = pString;
    COORD aLoc[n];

    for (i = 0; i<n; i++) {
        aLoc[i].X = rcPop.Left + 3;
        aLoc[i].Y = rcPop.Top + 2 + i;

    }
    str_len = strlen(pString[n - 1]);
    aLoc[n - 1].X = rcPop.Left + 3 + (maxlen - str_len) / 2;
    aLoc[n - 1].Y = aLoc[n - 1].Y + 2;

    labels.pLoc = aLoc;

    areas.num = 1;
    SMALL_RECT aArea[] = { { aLoc[n - 1].X, aLoc[n - 1].Y,
        aLoc[n - 1].X + 3, aLoc[n - 1].Y } };

    char aSort[] = { 0 };
    char aTag[] = { 1 };

    areas.pArea = aArea;
    areas.pSort = aSort;
    areas.pTag = aTag;
    PopUp(&rcPop, att, &labels, &areas);

    pos.X = rcPop.Left + 1;
    pos.Y = rcPop.Top + 2 + n;
    FillConsoleOutputCharacter(gh_std_out, '-', rcPop.Right - rcPop.Left - 1,
pos, &ul);
```

```
        DealInput(&areas, &iHot);
        PopOff();

        return bRet;

    }
    /**打印客房分类信息*/
    void printclass(CLASS_NODE *pclass)
    {
        printf("              ***************    ");
        printf("客房类别    最多入住人数    客房套数    客房未住个数\n");
        printf("              ***************    ");
        printf("%-12c%-16d%-12d%d", pclass->type, pclass->pnum, pclass->rnum,
pclass->emptyr);
        putchar('\n');
    }
    /**打印客房基本信息*/
    void printroom(ROOM_NODE * proom)
    {
        /*打印租客信息*/
        printf("              ***************");
        printf("客房编号 电话号码 客房类别 客房面积 每月租金 是否有阳台 是否有客人
入住\n");
        printf("              ***************");
        char *pc[2];
        pc[0] = "无";
        pc[1] = "有";
        //printf("%s %s",pc[0],pc[1]);
        int key;
        if (proom->in == 'y')
            key = 1;
        else key = 0;
        //printf("\n%d\n",key);
        printf("%-8s %-8s %-8c %-8.1f %-8.1f %-10s %s\n", proom->room_id,
proom->tel, proom->type, proom->area, proom->deposit, pc[proom->balcony - '0'],
pc[key]);

    }
    /**打印租客基本信息*/
    void printrenter(RENTER_NODE *prenter)
    {
        char * p[9];
        p[0] = "身份证号";
```

```
            p[1] = "客人姓名";
            p[2] = "客房编号";
            p[3] = "入住时间";
            p[4] = "退房时间";
            p[5] = "入住月数";
            p[6] = "押金";
            p[7] = "应缴费用";
            p[8] = "实缴费用";
            printf("%-20s %-8s %-8s %-20s %-20s %-8s %-8s %-8s %-8s\n", p[0], p[1],
p[2], p[3], p[4], p[5], p[6], p[7], p[8]);
            printf("%-20s %-8s %-8s %-20s %-20s %-8.1f %-8.1f %-8.1f %-8.1f\n",
prenter->id_card, prenter->name, prenter->room_id,
                prenter->date_in, prenter->date_out, prenter->months,
prenter->deposit, prenter->shouldpay, prenter->realpay);
            putchar('\n');
        }
        /**清屏*/
        BOOL Clear()
        {
            InitInterface();
            return TRUE;
        }
        /**更新房屋入住情况*/
        void Maintain_in(void)
        {
            CLASS_NODE *pclass;
            ROOM_NODE *proom;
            RENTER_NODE *prenter;
            int key = 1;
            int allroom, emptyroom;
            allroom = emptyroom = 0;
            /*遍历每个房屋下的支链，如果有租客没退房，则该房间有人居住，否则无人居住*/
            for (pclass = gp_head; pclass != NULL; pclass = pclass->next)
            {
                for (proom = pclass->rnext, allroom = emptyroom = 0; proom != NULL;
proom = proom->next, key = 1)
                {

                    prenter = proom->rnext;
                    proom->in = 'n';
                    while (prenter != NULL&&key)
                    {
                        if (prenter->date_out[0] == '\0')
                        {
```

```
                    key = 0;
                    proom->in = 'y';
                }
                prenter = prenter->next;
            }
            /*没遍历一个房间，房间总数++，没找到一个空房间，空房间数++*/
            allroom++;
            if (proom->in == 'n')
                emptyroom++;
        }
        /*记录总房间数和空房间数*/
        pclass->rnum = allroom;
        pclass->emptyr = emptyroom;
    }
}
/**统计两个日期之间的月数*/
float Maintain_month(char * date_in, char * date_out)
{//日期的格式为2015/03/06
    char year[5];
    char month[3];
    char day[3];
    int total_in, total_out;
    int i;
    total_in = total_out = 0;
    /*计算第一个日期的总天数*/
    for (i = 0; i <= 3; i++)
    {
        year[i] = date_in[i];
    }
    year[4] = '\0';
    total_in = atoi(year) * 365;
    month[0] = date_in[5];
    month[1] = date_in[6];
    month[2] = '\0';
    total_in += atoi(month) * 31;
    day[0] = date_in[8];
    day[1] = date_in[9];
    day[2] = '\0';
    total_in += atoi(day);

    /*如果date_out为空串，即客人在住，返回0*/
    if (date_out[0] == '\0')
        return 0;
```

```
        /*计算第二个日期的总天数*/
        for (i = 0; i <= 3; i++)
        {
            year[i] = date_out[i];
        }
        year[4] = '\0';
        total_out = atoi(year) * 365;
        month[0] = date_out[5];
        month[1] = date_out[6];
        month[2] = '\0';
        total_out += atoi(month) * 31;
        day[0] = date_out[8];
        day[1] = date_out[9];
        day[2] = '\0';
        total_out += atoi(day);
        /*计算两个日期相差的天数，进而可以计算出月数*/
        float months;
        months = (total_out - total_in) / 31.0;
        months = (months - (int)months>0.5) ? (int)months + 1 : (int)months;//四舍五入法计算月数
        return months;

    }
    /**判断日期是否在指定范围之内，是返回1，不是返回0*/
    int Judgedate(char *date_in, char *date_out, char *date)
    {
        /*计算出三个日期的总天数，通过天数判断date是否在date_in和date_out之间*/
        int total_in, total_out, total;
        total = total_in = total_out = 0;
        char year[5];
        char month[3];
        char day[3];
        int i;

        /*计算date_in的总天数*/
        for (i = 0; i <= 3; i++)
        {
            year[i] = date_in[i];
        }
        year[4] = '\0';
        total_in = atoi(year) * 365;
        month[0] = date_in[5];
        month[1] = date_in[6];
        month[2] = '\0';
```

```
    total_in += atoi(month) * 31;
    day[0] = date_in[8];
    day[1] = date_in[9];
    day[2] = '\0';
    total_in += atoi(day);

    /*计算date_out的总天数*/
    for (i = 0; i <= 3; i++)
    {
        year[i] = date_out[i];
    }
    year[4] = '\0';
    total_out = atoi(year) * 365;
    month[0] = date_out[5];
    month[1] = date_out[6];
    month[2] = '\0';
    total_out += atoi(month) * 31;
    day[0] = date_out[8];
    day[1] = date_out[9];
    day[2] = '\0';
    total_out += atoi(day);

    /*计算date的总天数*/
    for (i = 0; i <= 3; i++)
    {
        year[i] = date[i];
    }
    year[4] = '\0';
    total = atoi(year) * 365;
    month[0] = date[5];
    month[1] = date[6];
    month[2] = '\0';
    total += atoi(month) * 31;
    day[0] = date[8];
    day[1] = date[9];
    day[2] = '\0';
    total += atoi(day);
    /*判断日期是否在范围之内*/
    if (total >= total_in&&total <= total_out)
        return 1;
    else return 0;
}
```

2.RENT.H

```
#pragma once
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <wincon.h>
#include <conio.h>
#include <string.h>
#include <io.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <ctype.h>
#include <time.h>
#include <math.h>


#ifndef TYPE_H_INCLUDED
#define TYPE_H_INCLUDED

#define SCR_ROW 25              /*屏幕行数*/
#define SCR_COL 80              /*屏幕列数*/

/**
*客人租房信息链结点结构
*/
typedef struct renter_node {
    char id_card[20];           /**< 身份证号*/
    char name[20];              /**< 姓名*/
    char room_id[4];            /**< 入住客房编号*/
    char date_in[18];           /**< 入住时间*/
    char date_out[18];          /**< 退房时间*/
    float months;               /**< 入住月数*/
    float deposit;              /**< 押金*/
    float shouldpay;            /**< 应缴费用*/
    float realpay;              /**< 实缴费用*/
    struct renter_node *next;   /**< 指向下一结点的指针*/
} RENTER_NODE;

/**
*客房基本信息链结点结构
*/
typedef struct room_node {
    char room_id[4];            /**< 客房编号*/
    char tel[5];                /**< 电话号码,8+客房号*/
```

```
    char type;            /**< 客房类别*/
    float area;       /**< 客房面积*/
    float deposit;        /**< 每月租金*/
    char balcony;             /**< 是否有阳台，1有，0无*/
    char in;              /**< y有客人，n无客人*/
    struct ROOM_NODE *rnext;  /**< 指向租客基本信息支链的指针*/
    struct room_node *next;  /**< 指向下一结点的指针*/
} ROOM_NODE;

/**
*客房分类信息链结点结构
*/
typedef struct class_node {
    char type;            /**< 客房类别*/
    int pnum;              /**< 最多入住人数*/
    int rnum;                  /**< 客房套数*/
    int emptyr;          /**< 客房未住套数*/
    struct class_node *next;/**指向下一节点*/
    struct room_node *rnext;/**指向客房基本信息节点*/

} CLASS_NODE;

/**
*客房营业额统计结构
*/
typedef struct income {
    char type;//客房类别
    float sale[12];//记录1-12月的收入
    struct income *next;      /**< 指向下一结点的指针*/
} INCOME_NODE;
/**
*客房营业额统计结构
*/
typedef struct room_info {
    char room_id[4];//客房编号
    char type;//客房类别
    float income;//客房收入
    float months;//客人在本年入住该客房总月数
    float in_rate;//客房入住率
    struct room_info *next;//下一节点
}ROOM_INFO;
/**
*租客入住情况统计结构
*/
```

```
typedef struct renter_info {
    char id_card[20];/**身份证号*/
    char name[20];/**姓名*/
    float allmonths;/**入住总月数*/
    float shouldpay;/**应付总金额*/
    float realpay;/**实付总金额*/
    struct renter_info *next;/**指向下一个节点*/
}RENTER_INFO;
/**
*屏幕窗口信息链结点结点结构
*/
typedef struct layer_node {
    char LayerNo;           /**< 弹出窗口层数*/
    SMALL_RECT rcArea;      /**< 弹出窗口区域坐标*/
    CHAR_INFO *pContent;    /**< 弹出窗口区域字符单元原信息存储缓冲区*/
    char *pScrAtt;          /**< 弹出窗口区域字符单元原属性值存储缓冲区*/
    struct layer_node *next; /**< 指向下一结点的指针*/
} LAYER_NODE;

/**
*标签束结构
*/
typedef struct label_bundle {
    char **ppLabel;         /**< 标签字符串数组首地址*/
    COORD *pLoc;            /**< 标签定位数组首地址*/
    int num;                /**< 标签个数*/
} LABEL_BUNDLE;

/**
*热区结构
*/
typedef struct hot_area {
    SMALL_RECT *pArea;      /**< 热区定位数组首地址*/
    char *pSort;            /**< 热区类别(按键、文本框、选项框)数组首地址*/
    char *pTag;             /**< 热区序号数组首地址*/
    int num;                /**< 热区个数*/
} HOT_AREA;

LAYER_NODE *gp_top_layer = NULL;                /*弹出窗口信息链链头*/
CLASS_NODE *gp_head = NULL;                      /*主链头指针*/


char *gp_sys_name = "房屋出租信息管理系统";     /*系统名称*/
char *gp_room_info_filename = "room.dat";         /*客房信息数据文件*/
```

```
char *gp_renter_info_filename = "renter.dat";  /*租客信息数据文件*/
char *gp_class_info_filename = "class.dat";      /*分类信息数据文件*/


char *ga_main_menu[] = { "文件(F)",              /*系统主菜单名*/
"数据维护(M)",
"数据查询(Q)",
"数据统计(S)",
"帮助(H)"
};

char *ga_sub_menu[] = { "[S] 数据保存",          /*系统子菜单名*/
"[X] 退出    Alt+X",//

"插入客房分类信息",
"修改客房分类信息",
"删除客房分类信息",
"插入客房基本信息",
"修改客房基本信息",
"删除客房基本信息",
"插入客人租房信息",
"修改客人租房信息",
"删除客人租房信息",            /*空串用来在弹出菜单中分隔子菜单项，下同*/

"客房分类信息查询功能",
"客房基本信息查询功能",
"客人租房信息查询功能",
"[S] 客房基本信息代码",
"[T] 客房类别代码",

"统计每种类别的客房总数、入住数、未住数",
"按月统计本年度各类客房的营业额",
"输入年份，统计该年所有客房的营业额、入住月数、入住率",
"统计出租月数最多的 10 个客人租房信息",
"统计当前所有客房租金及入住情况",

"[T] 帮助主题",
"[A] 关于...",
"清空屏幕"
};

int ga_sub_menu_count[] = { 2, 9, 5, 5, 3 };  /*各主菜单项下子菜单的个数*/
int gi_sel_menu = 1;                          /*被选中的主菜单项号,初始为1*/
int gi_sel_sub_menu = 0;                      /*被选中的子菜单项号,初始为0,表示未选中*/
```

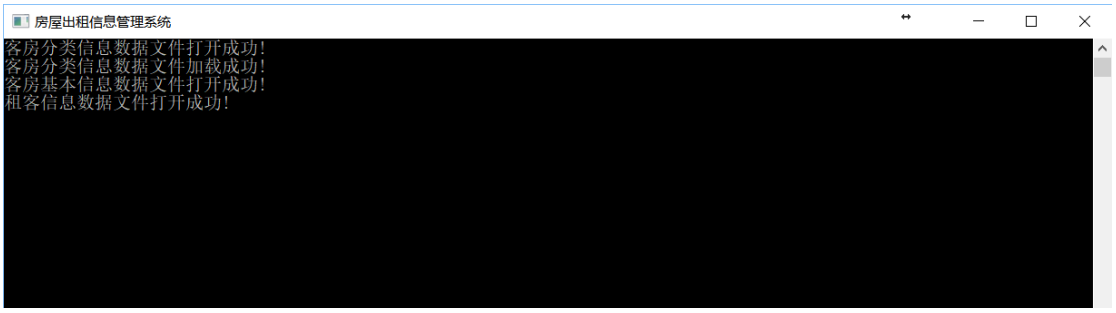
```
CHAR_INFO *gp_buff_menubar_info = NULL;      /*存放菜单条屏幕区字符信息的缓冲区*/
CHAR_INFO *gp_buff_stateBar_info = NULL;     /*存放状态条屏幕区字符信息的缓冲区*/

char *gp_scr_att = NULL;     /*存放屏幕上字符单元属性值的缓冲区*/
char gc_sys_state = '\0';    /*用来保存系统状态的字符*/


HANDLE gh_std_out;           /*标准输出设备句柄*/
HANDLE gh_std_in;            /*标准输入设备句柄*/

int LoadCode(char *filename, char **ppbuffer);  /*代码表加载*/
int CreatList(CLASS_NODE **pphead);              /*数据链表初始化*/
void InitInterface(void);                   /*系统界面初始化*/
void ClearScreen(void);                          /*清屏*/
void ShowMenu(void);                             /*显示菜单栏*/
void PopMenu(int num);                           /*显示下拉菜单*/
void PopUp(SMALL_RECT *, WORD, LABEL_BUNDLE *, HOT_AREA *);  /*弹出窗口屏幕信息维护*/
void PopOff(void);                               /*关闭顶层弹出窗口*/
void DrawBox(SMALL_RECT *parea);                 /*绘制边框*/
void LocSubMenu(int num, SMALL_RECT *parea);     /*主菜单下拉菜单定位*/
void ShowState(void);                            /*显示状态栏*/
void TagMainMenu(int num);                       /*标记被选中的主菜单项*/
void TagSubMenu(int num);                        /*标记被选中的子菜单项*/
int DealConInput(HOT_AREA *phot_area, int *pihot_num);  /*控制台输入处理*/
void SetHotPoint(HOT_AREA *phot_area, int hot_num);     /*设置热区*/
void RunSys(CLASS_NODE **pphd);                  /*系统功能模块的选择和运行*/
BOOL ExeFunction(int main_menu_num, int sub_menu_num);  /*功能模块的调用*/
void CloseSys(CLASS_NODE *phd);                  /*退出系统*/
BOOL ShowModule(char **pString, int n);

BOOL LoadData(void);            /*数据加载*/
BOOL SaveData(void);            /*保存数据*/
BOOL ExitSys(void);             /*退出系统*/
BOOL HelpTopic(void);           /*帮助主体*/
BOOL AboutDorm(void);           /*关于系统*/


BOOL Query(void);       /*查询客房基本信息字符信息*/


                        /**查找客房节点*/
ROOM_NODE * seekroom(CLASS_NODE *hd, char * r_id);
/**查找租客节点*/
```

```
RENTER_NODE * seekrenter(CLASS_NODE *hd, char *date, char * id_card);
/**查找prenter_info指向链表中是id_card，存在返回id_card所在的节点，不存在返回NULL**/
RENTER_INFO * search_renter_info(RENTER_INFO * hd, char *id_card);
/**将租房月数统计链表按租房月数降序排序*/
void sortrenter_info(RENTER_INFO *hd);
/**查询客房类别代码*/
BOOL QueryTypeInfo(void);
/**更新房屋入住情况*/
void Maintain_in(void);
/**统计两个日期之间的月数*/
float Maintain_month(char * date_in, char * date_out);
/**判断日期是否在指定范围之内，是返回1，不是返回0*/
int Judgedate(char *date_in, char *date_out, char *date);


/**输出客房分类信息*/
void printclass(CLASS_NODE *pclass);
/**插入客房分类信息*/
BOOL Insertclass();
/**修改客房分类信息*/
BOOL Maintainclass();
/**删除客房分类信息*/
BOOL Delclass();
/**查询客房分类信息*/
BOOL Queryclass();

/**输出客房基本信息*/
void printroom(ROOM_NODE *proom);
/**插入客房基本信息*/
BOOL Insertroom();
/**修改客房基本信息*/
BOOL Maintainroom();
/**删除客房基本信息*/
BOOL Delroom();
/**查询客房基本信息*/
BOOL Queryroom();

/**输出租客信息*/
void printrenter(RENTER_NODE *prenter);
/**插入租客信息*/
BOOL Insertrenter();
/**修改租客信息*/
BOOL Maintainrenter();
/**删除租客信息*/
```

```
BOOL Delrenter();
/**查询租客信息*/
BOOL Queryrenter();

/**统计每种类别的客房总数、入住数、未住数*/
BOOL Staclass();
/**按月统计本年度各类客房的营业额*/
BOOL Staincome();
/**输入年份，统计该年所有客房的营业额、入住月数、入住率*/
BOOL Staroom_info();
/**统计出租月数最多的 10 个客人租房信息*/
BOOL Starenter_info();
/**统计当前所有客房租金及入住情况*/
BOOL Sta_in();

BOOL Clear();

#endif /**< TYPE_H_INCLUDED*/
```

六、运行测试与结果分析

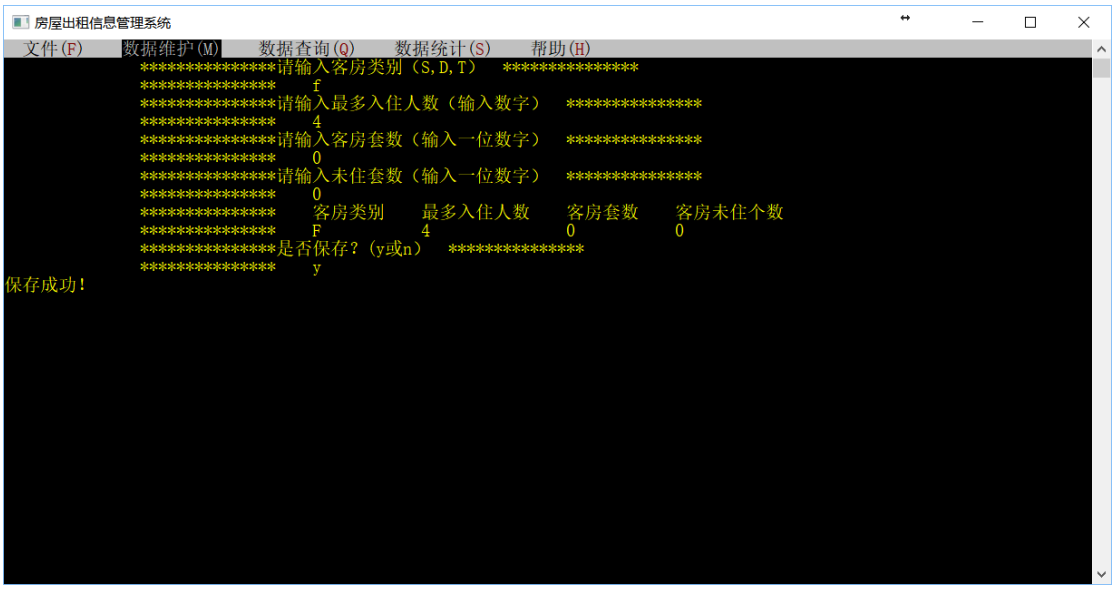
1.数据加载



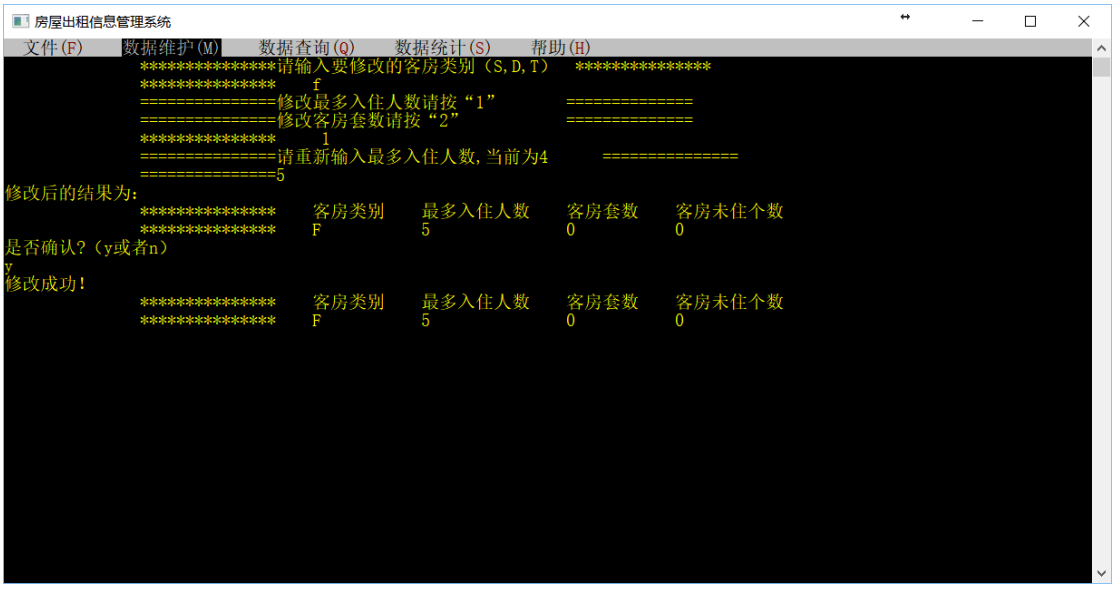
打开系统后进行数据统计，查询，得到合理且一致的数据。

2.客房分类信息操作

1.客房分类节点插入



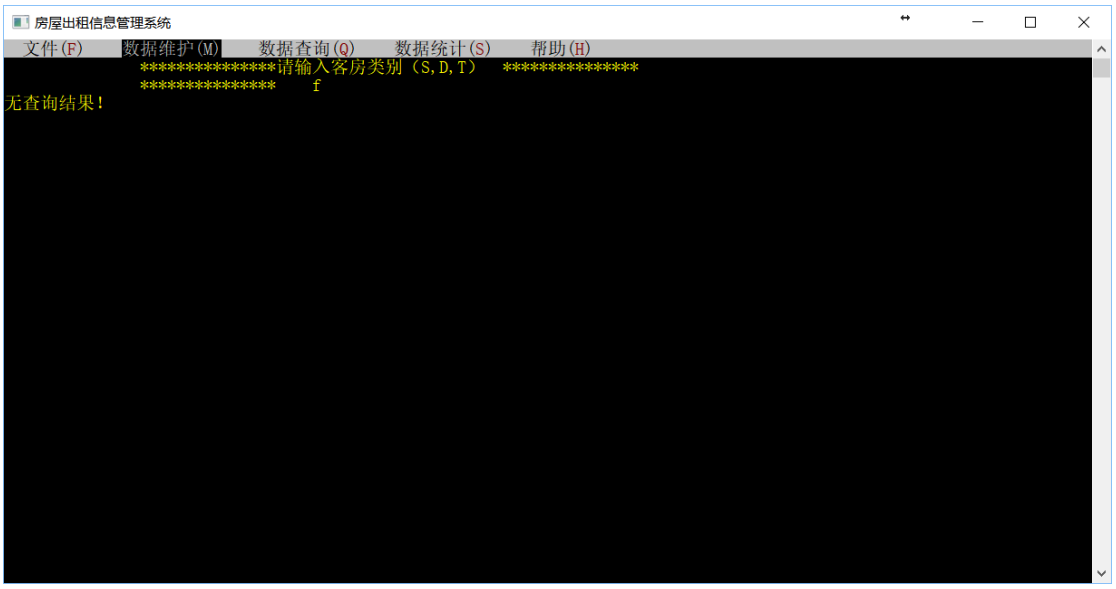
2.客房分类节点修改



3.客房分类节点删除

4.客房分类信息节点查询

插入前

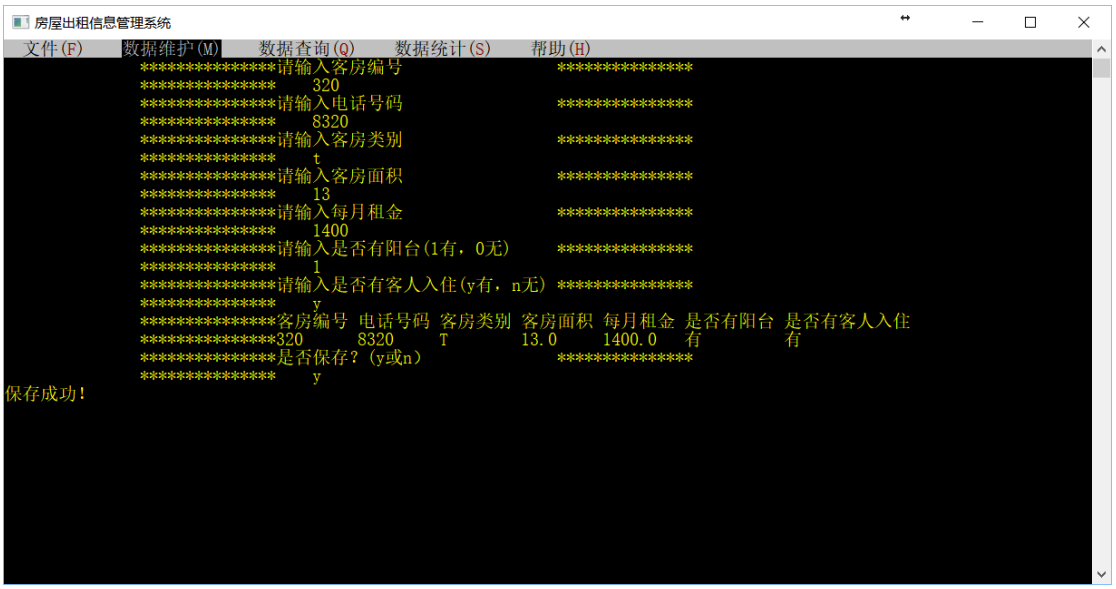


插入后

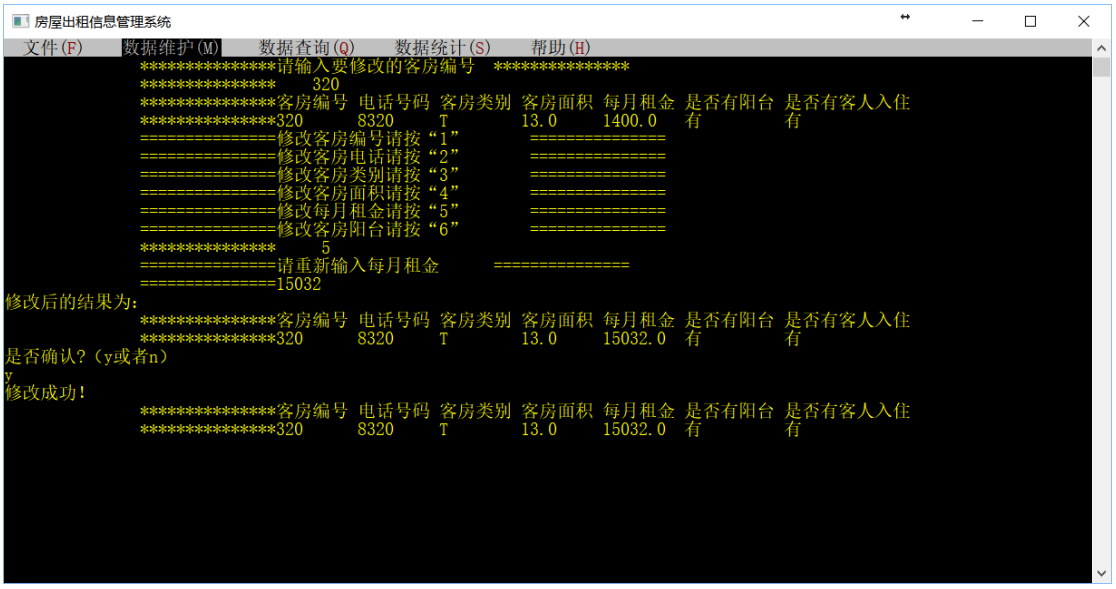
修改后

3.客房基本信息操作

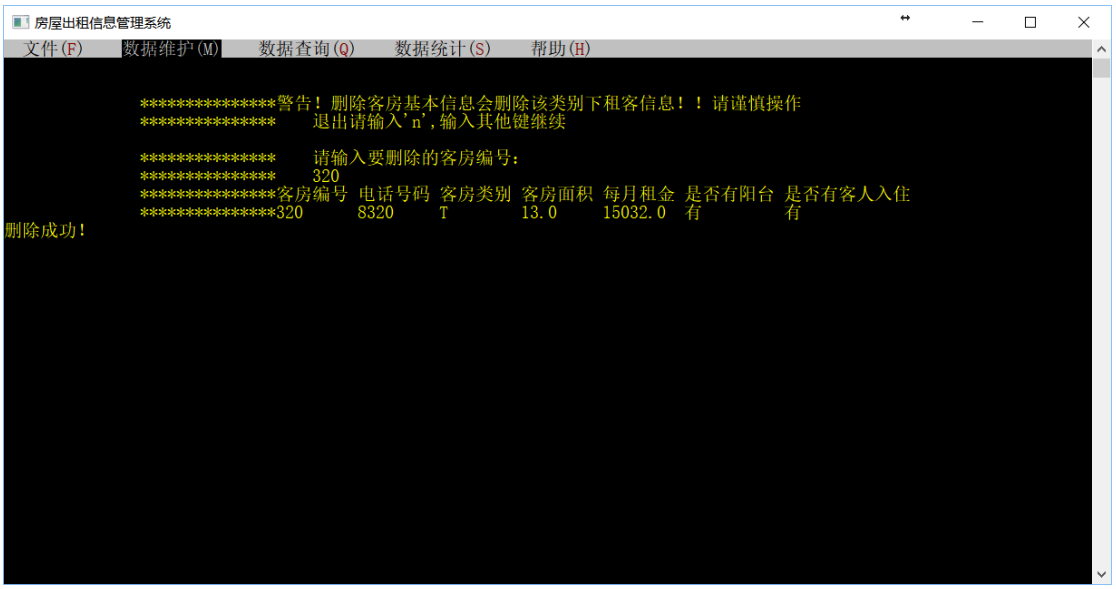
1.客房基本信息插入



2.客房基本信息修改

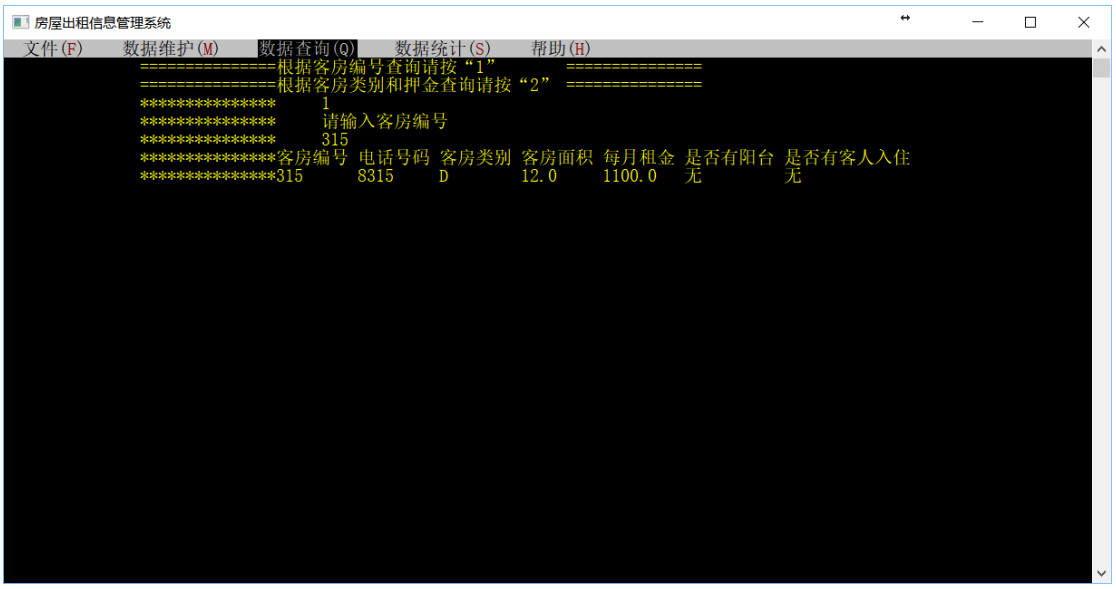


3.客房基本信息删除

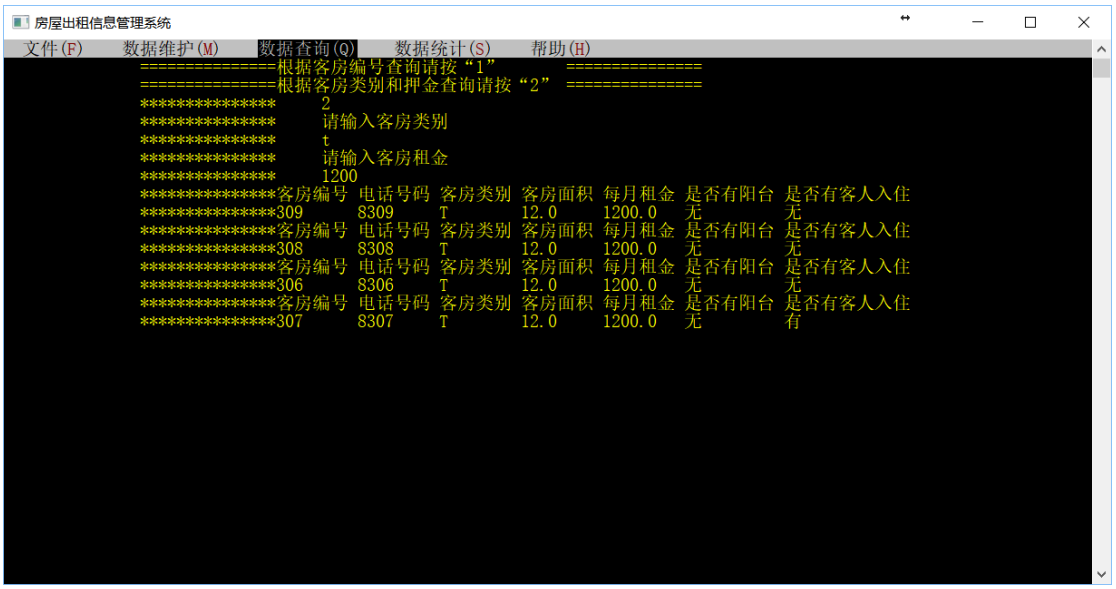


4.客房基本信息查询

1）按客房编号查询

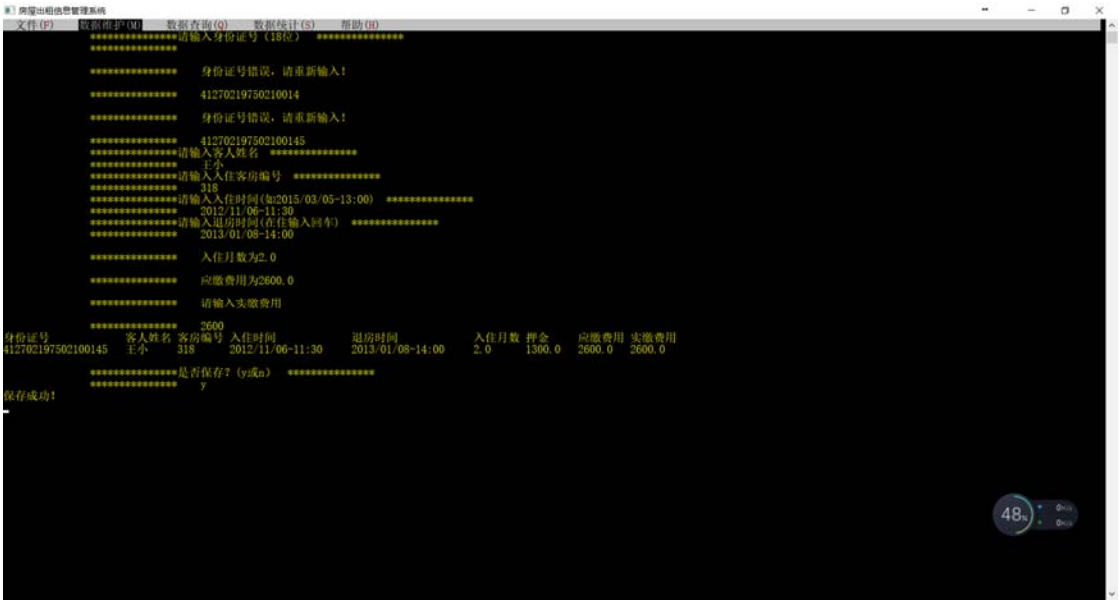


2）按客房类别及租金查询

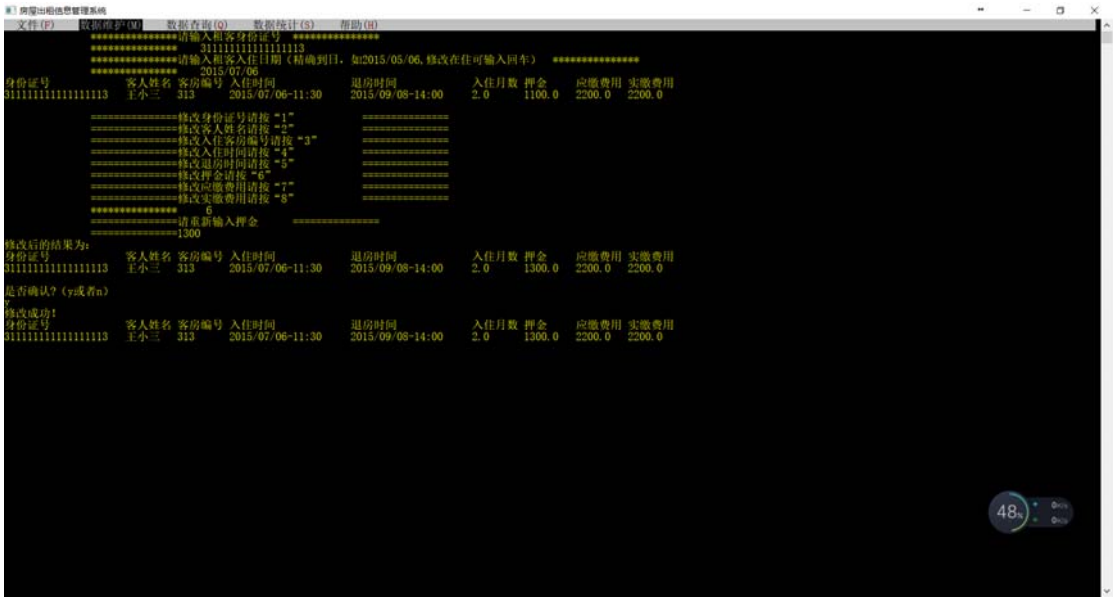


4.客人租房信息操作

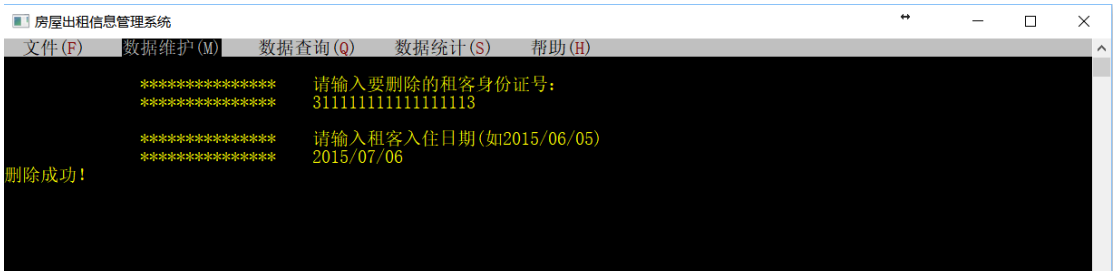
1.租客信息节点插入



2. 租客信息节点修改

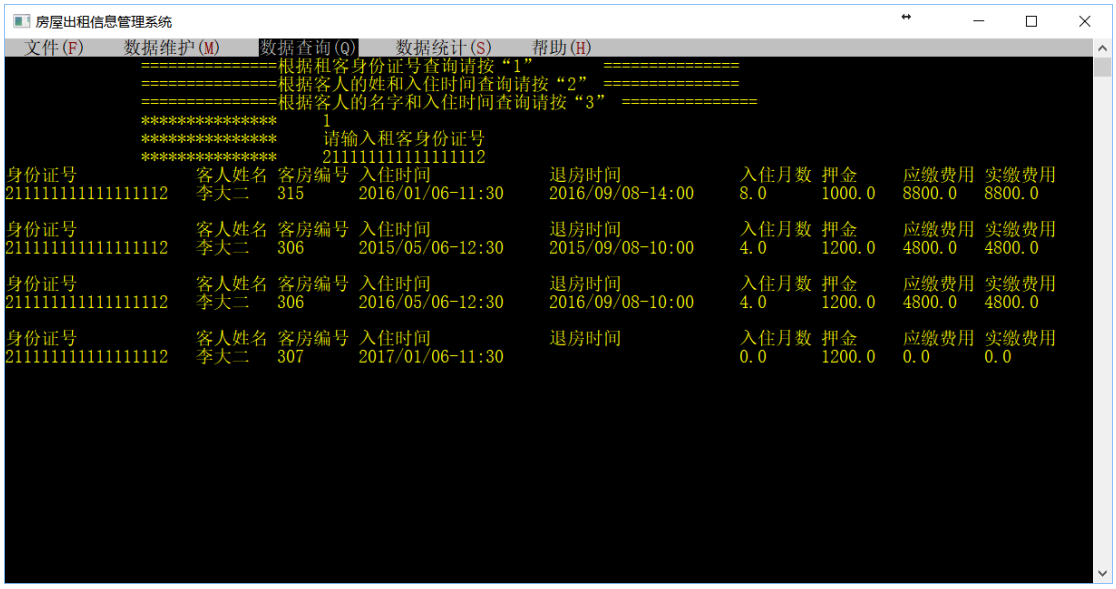


3. 租客信息节点删除

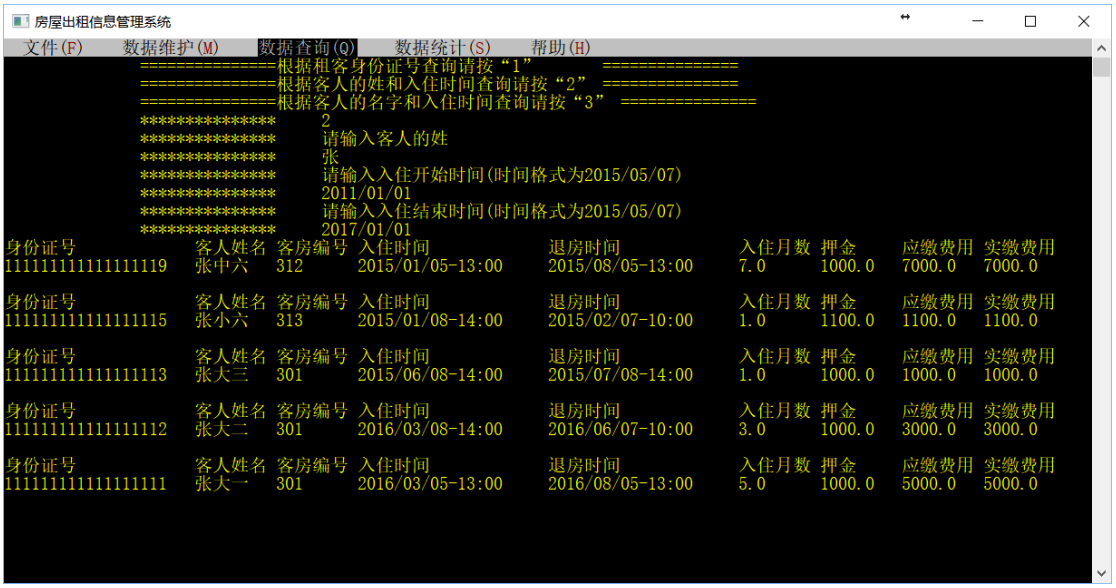


4. 租客信息节点查询

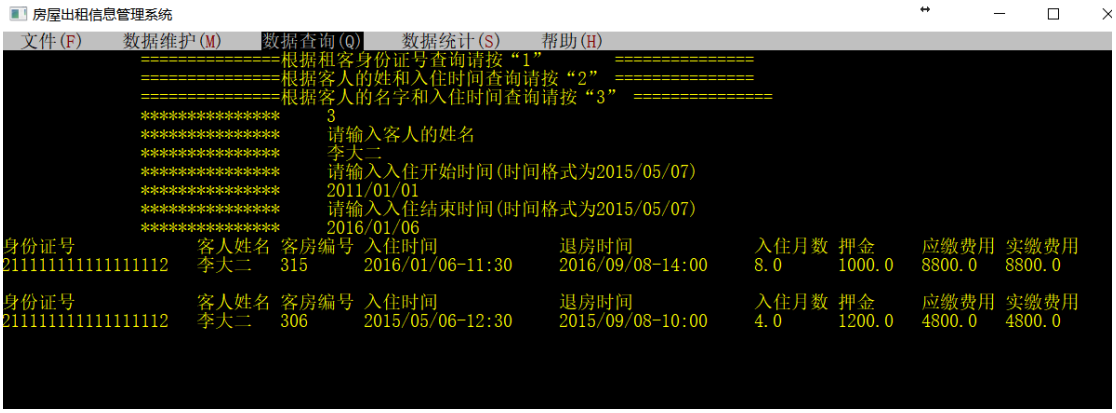
1）根据身份证号查询



2）根据客人的姓和入住时间查询

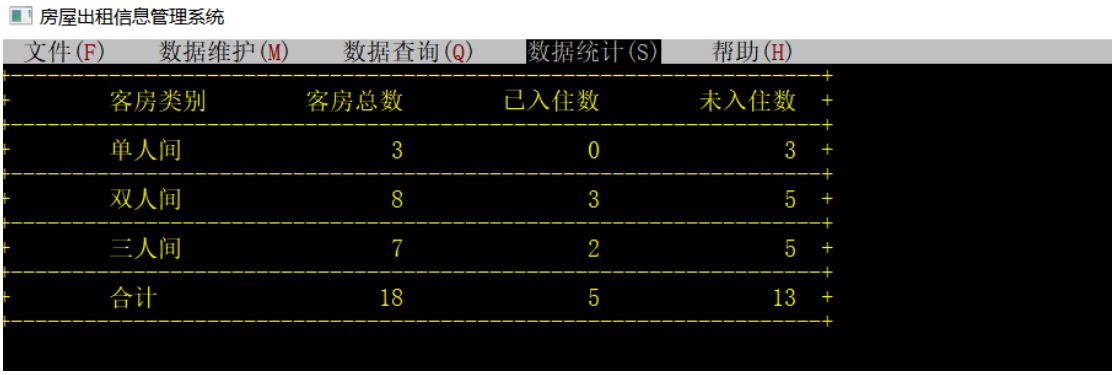


3）根据客人的名字和入住时间查询

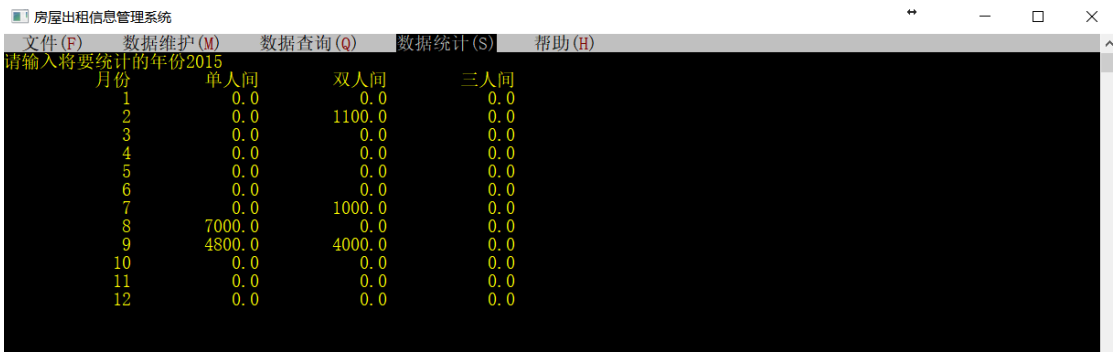


5.数据统计

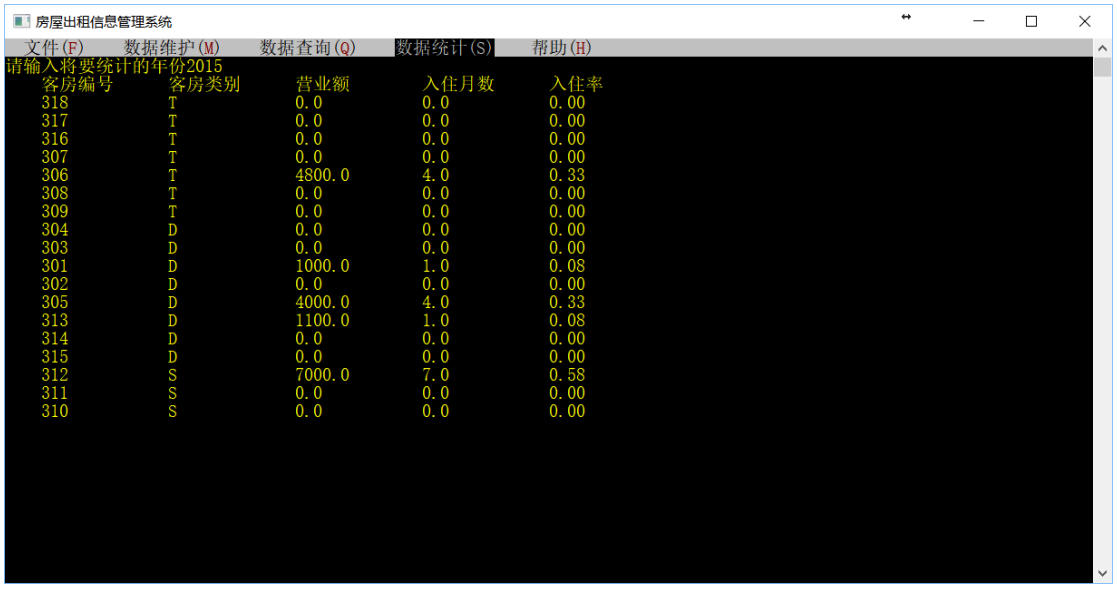
1) 统计每种类别的客房总数、入住数、未住数



2) 统计每种类别的客房总数、入住数、未住数

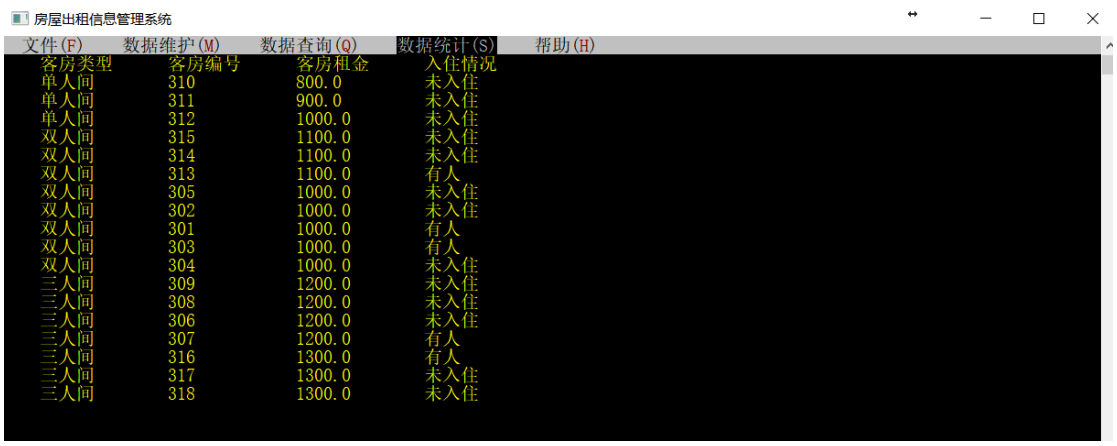


3) 输入年份，统计该年所有客房的营业额、入住月数、入住率



4) 统计出租月数最多的 10 个客人租房信息

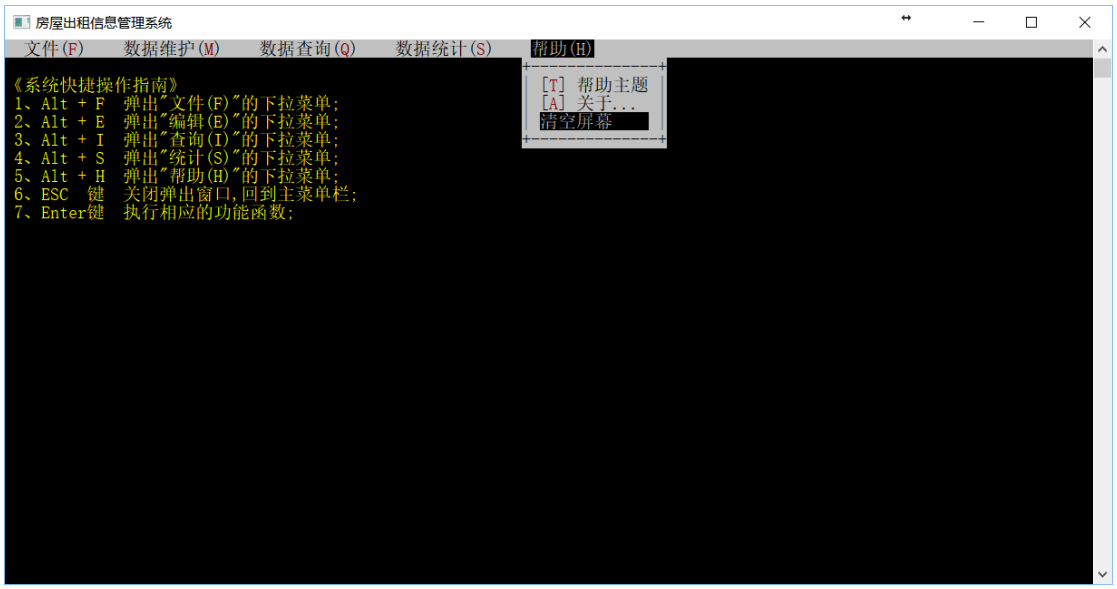
5) 统计当前所有客房租金及入住情况



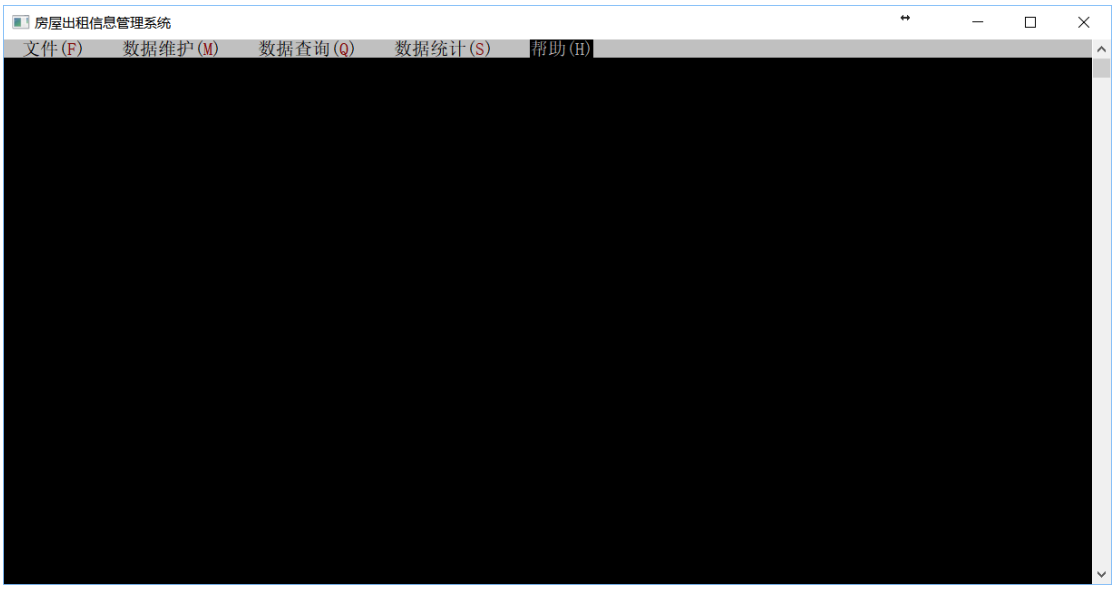
6.帮助

1.清屏:

清屏前:



清屏后:



七、总结

暑假开始的课程设计，刚开始我看到题目时一头雾水，不知该从何处下手，因为平时学的 C 根本没有提及控制台程序，根本不知该用哪方面的知识来实现文本菜单界面，因为平时只做过一些输入输出的普通小程序，根本没有界面的概，所以拖了久才开始做。刚开始没有什么进展，零零碎碎看了一些书上和老师给的样例，但是没有从整体上理解整个程序，所以导致一直不能全面地思考程序的流程和方向。后来因为时间的紧迫性，终于开始下手写写程序。按照书上和老师给的样例进行修改，终于慢慢地理解了这个较为复杂和庞大的系统的主线。因为一直不敢下手，导致没有进度，当我动手去写了几行代码时，才知道并没有想象中那么高深莫测，什么东西都得自己亲自实践过才能了解，才会理解和学到东西。

整个系统的核心就是链表操作，通过阅读 C 语言课程设计及课设要求，明白了各功能函数之间没有什么联系，因此我把大任务分成了许多小任务，从主链开始编写，每次编写一个功能，调试成功后再编写下一个功能，然后编写二级支链、三级支链，最后完成统计函数。一步一个脚印，稳扎稳打的书写方式保证了函数的正确率及程序的稳定性，为我节省了大量调试函数的时间。

程序的数据结构实际上就是一个三方向十字交叉链表，主链为客房分类信息，分类信息结点上有指向分类信息对应的客房基本信息支链的指针，而客房基本信息结点上又保存着指向客人租房信息的指针。这样一个三级链表保存着整个系统的三类基础信息。数据的增删，修改，查询和统计都基于这个十字链表。其中数据的统计需要再构建链表来保存统计后的数据。

完成课设的这段时间里我体会到调试员真还得过硬的基础知识，能敏锐觉察出代码中的错误，而且还得有足够的精力和耐力。通过这次课程设计，我对很多的函数有了新的认识，也学会了运用多种函数，初步掌握了编写软件的基本流程和基本方法，明白了数据结构的重要性。写软件的过程中我又温习了曾经自己掌握不熟练函数的用法，特别是文件的开启和关闭，读与写。这次课程设计课总的来说让我受益匪浅，学到了很多知识。