****

**算法设计与分析报告**

|  |  |
| --- | --- |
| **题目** | **宾馆管理系统** |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **学生姓名** | **李晓明** |  |
|  | **学生学号** |  |  |
|  | **指导教师** | **章元** |  |
|  | **导师职称** | **副教授** |  |
|  | **所在系部** | **数据科学与大数据技术** |  |
|  | **年级专业** | **2018数据科学与大数据技术** |  |

2021 年 01 月 10 日

**目录**

[一、概述 2](#_Toc59988071)

[二、需求分析 3](#_Toc59988072)

[三、设计 4](#_Toc59988073)

[3.1总体框架 4](#_Toc59988074)

[3.2初始化模块 4](#_Toc59988075)

[3.3开房模块 5](#_Toc59988076)

[3.4查找模块 6](#_Toc59988077)

[3.5删除模块 7](#_Toc59988078)

[3.6修改模块 7](#_Toc59988079)

[3.7排序模块 8](#_Toc59988080)

[3.8统计模块 9](#_Toc59988081)

[3.9展示模块 9](#_Toc59988082)

[四、开发 9](#_Toc59988083)

[4.1主函数模块 9](#_Toc59988084)

[4.2初始化模块 11](#_Toc59988085)

[4.3录入模块&删除模块&修改模块 12](#_Toc59988086)

[4.4查找模块 16](#_Toc59988087)

[4.5排序模块 21](#_Toc59988088)

[4.6统计模块 25](#_Toc59988089)

[4.7展示所有信息模块 28](#_Toc59988090)

[五、测试 29](#_Toc59988091)

[六、心得体会 35](#_Toc59988092)

# 一、概述

随着计算机技术的飞速发展，信息时代的到来，信息改变了我们这个社会。各类行业在日常经营管理各个方面也在悄悄地走向规范化和信息化。宾馆管理的信息化程度体现在将计算机及网络与信息技术应用于经营与管理，以现代化工具代替传统手工作业。无疑，使用网络信息化管理使宾馆管理更先进，更高效，更科学，信息交流更迅速。

宾馆管理系统是宾馆经营管理中不可缺少的部分，它的内容对于经营决策者和管理者来说都至关重要，因此宾馆管理系统，信息管理系统应该能够为用户提供充分的信息和快捷的查询手段。但是一直以来人们使用传统的人工的方式管理文件档案，这种管理方式存在着许多弊端，如：效率低，保密性差，容易出现差错等，且对于查询空房间及已定房间等极为不方便。在当今时代，这些完全能够改用计算机来代替人的手工操作。

本项目旨在利用算法为基础，在保证用到一些常用算法的前提下设计并开发出满足对宾馆管理系统的需求的程序。其中用到了诸如枚举法，冒泡排序，插入排序，选择排序，希尔排序，二分法查找，插值法查找，顺序法查找等算法。

# 二、需求分析

本系统设计实现的目标是对宾馆管理信息，客户的信息进行管理，既要满足客户选购不同类型房间的需求，又需要方便管理员查询入住情况，同时满足管理员对顾客姓名，身份证号，入住天数，以及预计住房费用等信息的录入，查找，删除，既登记顾客信息，查找顾客信息，修改顾客信息，对不同数据进行排序，退房并清除顾客信息等功能，考虑到宾馆的特殊性，需留存一份开退房记录以配合公安局等部门的检查，要求顾客开放退房修改信息的记录都有留存，不会被程序内部删除掉。

功能清单：

1. 开房功能
2. 能够通过选择不同房间号的房间，以及不同类型的房间
3. 当输入的房间已经有人入住时，要返回提示
4. 当输入的房间号有误时，要返回提示
5. 系统能够录入顾客身份证号，姓名，入住天数
6. 顾客身份证号应在五位数以上（假设），一个身份证可开多个房间
7. 入住天数需在一天以上，否则返回提示信息，重新录入
8. 系统可以根据顾客入住天数以及入住的房间类型自动计算房费
9. 开房记录需保存留档
10. 查找功能
11. 系统能够通过房间号，身份证号，和姓名三种方式查询
12. 当输入以上三种信息输入有误，不符合规范时，需返回提示
13. 当输入以上三种信息符合规范，但系统中未找到时，需返回提示
14. 退房功能
15. 系统仅能支持通过房间号退房
16. 退房成功与否都需要返回提示
17. 若所退房间为空房间需返回提示信息
18. 退房记录需保存留档，不能因退房而被系统删除
19. 修改信息功能
20. 修改功能仅能对已经入住的房间中顾客信息修改，否则返回提示
21. 修改功能中对身份证号，入住天数等数据规范要求与开房功能中要求一致
22. 修改记录也需要保存留档，该记录永久保存，不能被修改，删除
23. 排序功能
24. 支持通过对房间号，入住天数，以及房费三种数据进行排序
25. 排序展示要求能清晰看到排序后结果，最好可以对比排序前后结果
26. 统计功能
27. 统计功能需统计空房间，已入住房间，以及豪华房，双人房，标间三种类型房间入住情况
28. 统计功能需要展示出空房间的房间号

# 三、设计

设计思路

围绕功能清单的六个主要功能，对其中查找功能和排序功能设计使用算法，算法详细介绍在开发模块介绍，此部分仅介绍设计思路。

### 3.1总体框架

### 3.2初始化模块

设计宾馆有10间标间(黄色部分)，5间豪华房(蓝色部分)，5间双人间（绿色部分）且不同类型房间价格不同。

|  |  |  |  |
| --- | --- | --- | --- |
| 房间价格表 | | | |
| 房间类型 | 标间 | 豪华房 | 双人间 |
| 价格 | 100 | 150 | 200 |

房间示意图：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 |
| 1001 | 1002 | 1003 | 1004 | 1005 | 1006 | 1007 | 1008 | 1009 | 1010 |

应需求设计初始化房间应该只执行一次，因为宾馆房间类型和房间数一般是固定的，初始化后基本信息不再变动。且要保证在后续对入住房间的顾客信息进行增删改查，但对房间信息不造成影响。

部分房间举例：

未登记入住前：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 房间号 | 顾客姓名 | 身份证号 | 入住天数 | 房间类型 | 预计房费 |
| 1001 | Null | 0 | 0 | 豪华房 | 0 |
| 1006 | Null | 0 | 0 | 双人房 | 0 |
| 2001 | Null | 0 | 0 | 标间 | 0 |

当有顾客入住时：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 房间号 | 顾客姓名 | 身份证号 | 入住天数 | 房间类型 | 预计房费 |
| 1001 | 李晓明 | 180374118 | 5 | 豪华房 | 1000 |
| 1006 | Null | 0 | 0 | 双人房 | 0 |
| 2001 | Null | 0 | 0 | 标间 | 0 |

退房后：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 房间号 | 顾客姓名 | 身份证号 | 入住天数 | 房间类型 | 预计房费 |
| 1001 | Null | 0 | 0 | 豪华房 | 0 |
| 1006 | Null | 0 | 0 | 双人房 | 0 |
| 2001 | Null | 0 | 0 | 标间 | 0 |

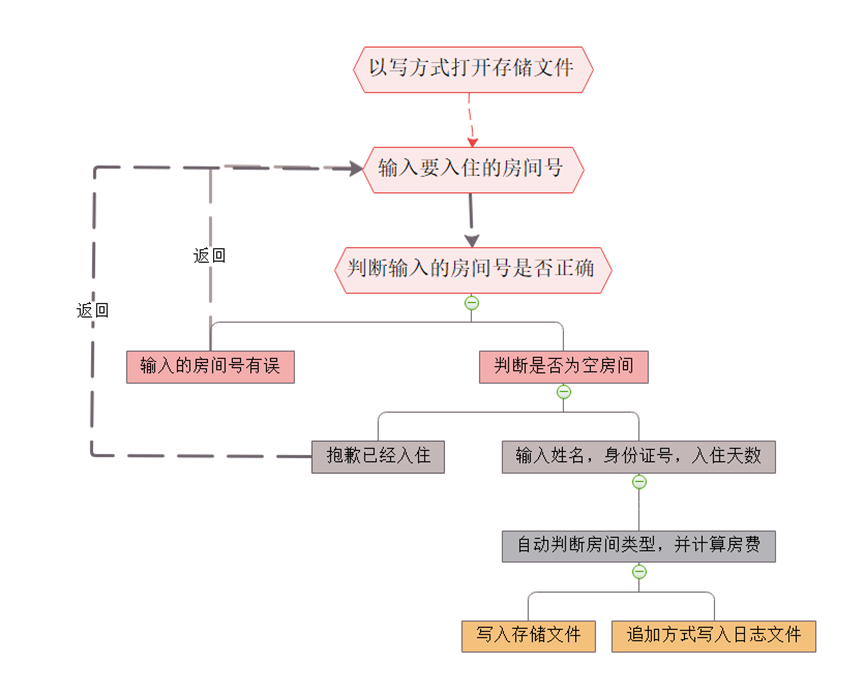
既退房前后仅变动顾客信息，而对房间号房间类型不改变。

所以我们的设计思路就是，初始化一个6\*20的矩阵，矩阵的[i][0]列和[i][4]列，房间号房间类型是定值，开房时，既通过房间号，找到矩阵对应的行，然后对该行数据进行修改，同理退房时，将该行的数据重新变为初始数据。

### 3.3开房模块

录入顾客信息时，因为考虑需求中方便公安机关等部门查阅开房记录，所以特保存一份日志文件供查询记录使用，此文件内容只有保存功能，不可被程序内部修改。房间号和身份证只能为数字类型，否则会返回提醒，该部分用异常值做，主动抛出异常。

设计规范：房间号和身份证号码为数字，若输入字母等返回提示信息，身份证号码应在五位数以上，入住天数应该大于等于1天



### 3.4查找模块

该模块用顺序法，插值法和二分法等算法实现通过房间号，身份证号和姓名三种方法对顾客信息进行查找。(算法的详细介绍看开发部分)，选择序号123分别对应三种查询方法，因为部分查询算法需要先排序，所以设计一个排序函数供需要先排序的算法调用。

插值法

通过房间号

排序

结果

二分法

查询

通过身份证号

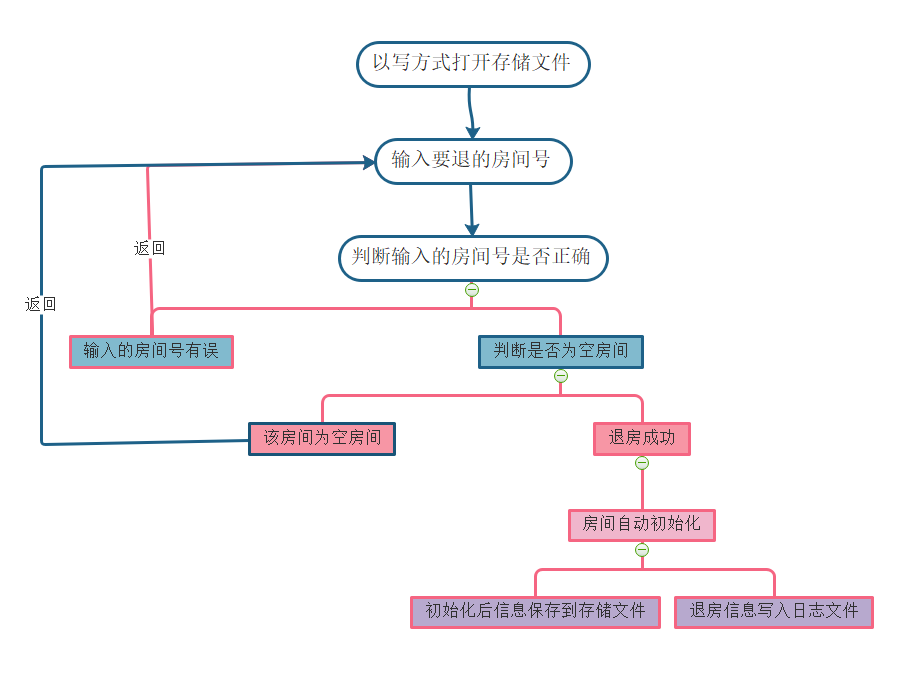
顺序法

通过姓名

### 3.5删除模块

删除模块设计思路和原理上还是属于修改信息，与前面开房模块一样，但是删除时仅能通过房间号进行删除。若是空房间，不能退房。退房后信息变成初始化状态的信息。

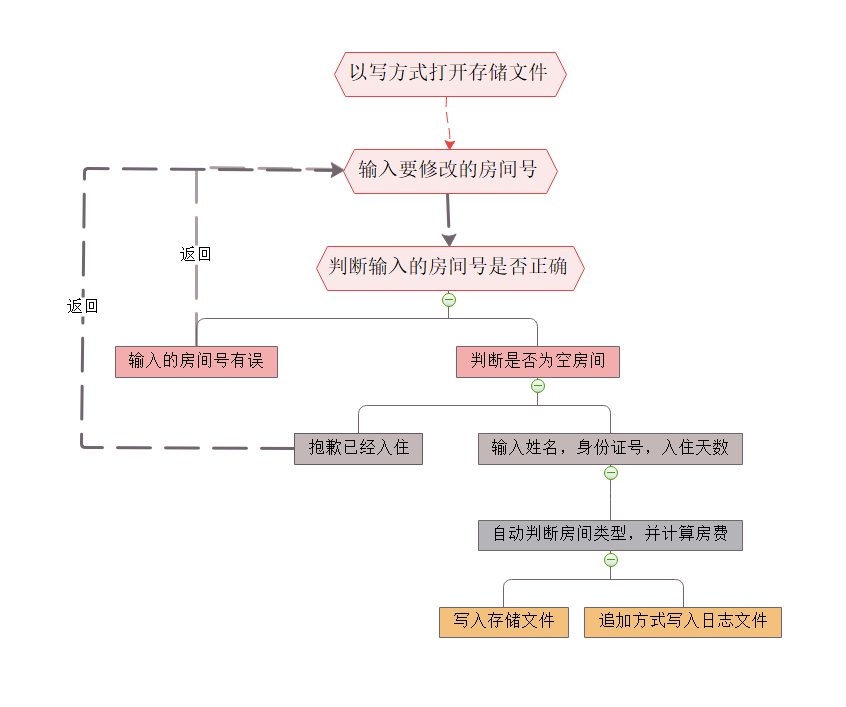
设计规范：房间号和身份证号码为数字，若输入字母等返回提示信息，身份证号码应在五位数以上，入住天数应该大于等于1天



### 3.6修改模块

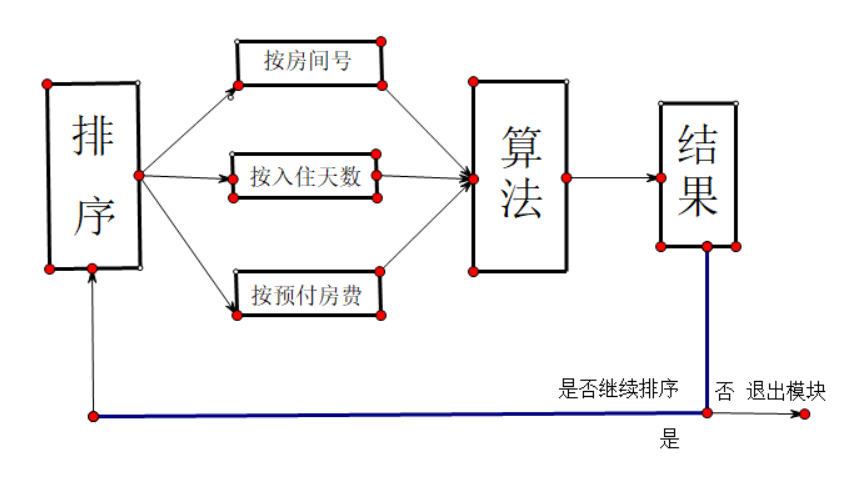
修改模块，设计为仅能通过查找房间号模式修改，若为空房间则不能修改。且修改的信息也应符合规范。

设计规范：房间号和身份证号码为数字，若输入字母等返回提示信息，身份证号码应在五位数以上，入住天数应该大于等于1天



### 3.7排序模块

排序模块应用四种算法，冒泡排序用两次，分别升序一次，降序一次，房间号排序用冒泡排序，入住天数用希尔排序，预付房费用冒泡，插入，选择三种排序方式。（详细介绍见开发文档），在通过12345序号选择对应的排序方法后，开始排序，排序结果调用展示模块，并将排序前后结果进行对比。



### 3.8统计模块

统计模块根据需求需要设计统计空房间，已入住房间，以及豪华房，双人房，标间三种类型房间入住情况五种情况。来满足各个场景需要。

### 3.9展示模块

该模块专门供其他各模块调用，例如排序后的结果展示，查找后的结果展示，所有信息展示

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 房间号 | 顾客姓名 | 身份证号 | 入住天数 | 房间类型 | 预计房费 |
| 1001 | Null | 0 | 0 | 豪华房 | 0 |
| 1002 | Null | 0 | 0 | 豪华房 | 0 |
| 1003 | Null | 0 | 0 | 豪华房 | 0 |

# 四、开发

## 4.1主函数模块

主函数中先查看是否已经创建用来存储数据的binguan.txt文件，如果已经创建了则调用菜单模块，供操作人员选择相应的功能，选择相应的序号后，则调用相应的模块。

代码：

**def main**():  
 **if not** os.path.exists(filename):  
 chushihua() #初始化房间信息  
 ctrl = **True** # 标记是否退出系统  
 **while** (ctrl):  
 menu() # 显示菜单  
 option = input("请选择：") # 选择菜单项  
 option\_str = re.sub("\D", "", option) # 提取数字  
 **if** option\_str **in** ['8', '1', '2', '3', '4', '5', '6', '7']:  
 option\_int = int(option\_str)  
 **if** option\_int == 8: # 退出系统  
 print('您已退出宾馆管理系统！')  
 ctrl = **False  
 elif** option\_int == 1: # 录入学生成绩信息  
 insert()  
 **elif** option\_int == 2: # 查找学生成绩信息  
 search()  
  
 **elif** option\_int == 3: # 删除学生成绩信息  
 delete()  
 **elif** option\_int == 4: # 修改学生成绩信息  
 modify()  
 **elif** option\_int == 5: # 排序  
 sort()  
 **elif** option\_int == 6: # 统计学生总数  
 total()  
 **elif** option\_int == 7: # 显示所有学生信息  
 show()

## 4.2初始化模块

初始化模块用于构建宾馆的数据结构初形，将宾馆信息与顾客信息构建成一个6\*20的矩阵，一共20个列表，每个列表中存储6维数据，数据由字典构成。共构建10间标间，5间豪华房，5间双人房。构建后调用存储模块将数据保存在文本文件中。

代码：

**def chushihua**():  
 stdentList = []  
 **for** i **in** range(10):  
 stdent = {"id": 2010-i, "name": "null", "IdCard":0,"days": 0, "hometype": "标间","yu\_money": 0} # 将输入的学生信息保存到字典  
 stdentList.append(stdent) # 将学生字典添加到列表中  
 **for** i **in** range(5):  
 stdent = {"id": 1010-i, "name": "null", "IdCard":0,"days": 0, "hometype": "双人房","yu\_money": 0} # 将输入的学生信息保存到字典  
 stdentList.append(stdent) # 将学生字典添加到列表中  
 **for** i **in** range(5):  
 stdent = {"id": 1005-i, "name": "null", "IdCard":0,"days": 0, "hometype": "豪华房","yu\_money": 0} # 将输入的学生信息保存到字典  
 stdentList.append(stdent) # 将学生字典添加到列表中  
 save(stdentList) # 将学生信息保存到文件  
 print("顾客信息录入完毕！！！")

## 4.3录入模块&删除模块&修改模块

录入模块与删除模块以及修改模块主体是一样的所以放在一块介绍，因为需求和设计都要求能够对顾客信息进行增删改查，在设计的时候设计为开房和退房，后房间信息不得被改编，因此，无论录入顾客信息或者删除顾客信息，本质上都是对矩阵中数据进行修改。录入顾客信息就是矩阵中的空值修改为顾客信息，删除顾客信息就是将房间内信息重新初始化为空值。因此这就是我们的设计思路，所以开发时也按这个思路来进行。

程序先调用os模块判断存储信息的文本文件是否存在，若存在则打开文件，读取文件中信息，将信息存在列表中。

然后程序以写模式打开文件，然后读取输入的房间号，以顺序查找方式，找到对应的房间号，然后程序判断该房间是否为空房间，若不是空房间则提醒顾客，然后重新操作，若是空房间，则开始逐条录入顾客信息，最后根据房间号，判断房间类型，再根据入住天数自动录入预付房费。在进行存储时，一份写入日志文件，一份写入存储的文本文件。最后输入是否继续录入信息。

同理修改模块与删除模块也是类似流程，开发思想一致。

代码：

**def insert**():  
 show()  
  
 **if** os.path.exists(filename) : # 判断文件是否存在  
 **with** open(filename, 'r') **as** rfile: # 打开文件  
 binguan\_old = rfile.readlines() # 读取全部内容  
 **else**:  
 **return  
 while True**:  
 #防止输入的房间号为非数字类型数据而导致程序报错  
 **try**:  
 print("您好，提醒您：1001-1005为豪华房，1006-1010为双人房，2001-2010为标间")  
 binguanid = int(input("请输入要入住的房间号ID："))  
 **if** binguanid <1001 **or** binguanid>2010 **or** 1010<binguanid<2001:  
 print("您输入的房间号有误请核对后再输入")  
 **continue  
 except**:  
 print("您输入的房间号不规范，请核对")  
 **else**:  
 **break  
  
 with** open(filename,"w") **as** wfile:# 以写模式打开文件  
 **for** binguan **in** binguan\_old:  
  
 d = dict(eval(binguan)) # 字符串转字典  
  
 **if** d["id"] == binguanid:  
 **if** d["name"] =="null":  
 **while True**:  
 **try**:  
 d["name"] = input("请输入姓名：")  
 d["IdCard"] = int(input("请输入身份证号："))  
 **if** d["IdCard"]<10000 :  
  
 print("请输入正确的身份证号")  
 **continue** d["days"] = int(input("请输入入住天数："))  
 **if** d["days"]<1 :  
 print("请输入规范的天数")  
 **continue  
 if** d["hometype"]=="标间":  
 d["yu\_money"]=d["days"]\*biao\_money  
 **elif** d["hometype"]=="双人房":  
 d["yu\_money"]=d["days"]\*shuang\_money  
 **elif** d["hometype"]=="豪华房":  
 d["yu\_money"]=d["days"]\*hhf\_money  
  
  
 **except**:  
 print("您的输入有误，请重新输入")  
 **else**:  
 **break** # 跳出循环  
 **with** open(filename2,"a") **as** wfile2:  
 wfile2.write("【开房信息】：\n 时间：{},房间号:{}，顾客:{}，身份证号：{},入住天数：{},预付金额：{}\n".format(time.strftime("%Y-%m-%d %H:%M:%S ", time.localtime()),d["id"],d["name"],d["IdCard"],d["days"],d["yu\_money"]))  
 wfile2.close()  
 binguan = str(d) # 将字典转换为字符串  
 wfile.write(binguan + "\n") # 将修改的信息写入到文件  
 print("登记完毕！")  
 **else**:  
 print("抱歉，该房间已经有客人了，请您核对房间号")  
 wfile.write(binguan)  
  
 **else**:  
  
 wfile.write(binguan) # 将未修改的信息写入到文件  
 #print("您输入的房间号有误，请检查正确的房间号后重新录入!")  
 mark = input("是否继续登记其他顾客信息？（输入y继续，输入其他默认退出）：")  
 **if** mark == "y":  
 insert() # 重新执行修改操作

## 4.4查找模块

查找模块按需求和设计需要实现通过房间号和身份证号还有姓名查找三种查询方法，我分别使用插值法来实现通过房间号查询需求，通过二分法实现通过身份证号查询需求，通过顺序法来实现通过姓名查询的需求。而由于部分查询方法需要查找的数据是已经排好序的，所以我先在该模块内部写入冒泡排序算法，若需要排序时，可先调用排序算法。

4.4.1.排序处理

因部分查找算法需求先排序，所以将排序函数写入查找模块内部，供部分查找算法调用。

排序代码：

**def paixu**(x):  
 datas=student\_list  
 lens=len(datas)  
 **for** i **in** range(lens-1,-1,-1):  
 **for** j **in** range(i):  
 **if** int(datas[j][x])>int(datas[j+1][x]):  
 datas[j],datas[j+1]=datas[j+1],datas[j]  
 **return** datas

* + 1. 二分法查找

算法时间复杂度： 

算法介绍：二分查找是一种基于比较目标值和数组中间元素的教科书式算法。如果目标值等于中间元素，则找到目标值。

如果目标值较小，继续在左侧搜索。

如果目标值较大，则继续在右侧搜索。

速记打油诗（来源LeetCode）

搜索左右边界时，搜索区间要阐明。

左闭右开最常见，其余逻辑便自明:

while要用小于号，这样才能不漏掉。

if相等别返回，利用mid锁边界。

mid加一或减一? 要看区间开或闭。

while结束不算完，因为你还没返回。

索引可能出边界，if 检查保平安。

左闭右开最常见，难道常见就合理?

有人真是不信邪，偏要改成两端闭

搜索区间记于心，或开或闭有何异?

二分搜索三变体，逻辑统一容易记。

一套框架改两行，胜过千言和万语。

**相关代码：**

id = int(input("请输入身份证号："))  
nums=paixu("IdCard")#二分查找先排序  
left, right = 0, len(nums) - 1  
**while** left <= right:  
 pivot = left + (right - left) // 2  
 **if** nums[pivot]["IdCard"] == id:  
 student\_query.append(nums[pivot])  
 show\_student(student\_query)  
 #show\_student(nums[pivot])  
 #print(nums[pivot])  
 **break  
 if** id < nums[pivot]["IdCard"]:  
 right = pivot - 1  
 **else** :  
 left = pivot + 1

**代码分析：**

初始化指针 left = 0, right = n - 1。

当 left <= right：

比较中间元素 nums[pivot] 和目标值 target 。

如果 target = nums[pivot]，返回 pivot。

如果 target < nums[pivot]，则在左侧继续搜索 right = pivot - 1。

如果 target > nums[pivot]，则在右侧继续搜索 left = pivot + 1。

图解(由于身份证号字段太长，所以此图解用其他数据演示)：

红色代表目标值，绿色为中间值，蓝色为左指针位置，黄色为右指针位置，刚开始时，left=0,right=9,pivot= left + (right - left) // 2=4,如下图：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **左指针** |  |  | **目标值** | **中间值** |  |  |  |  | **右指针** |
| **↓** |  |  |  | **↓** |  |  |  |  | **↓** |
| 11 | 23 | 31 | 37 | 43 | 46 | 52 | 57 | 61 | 72 |

此时目标值小于中间值，既target < nums[pivot]，所以执行right = pivot – 1=3， pivot= left + (right - left) // 2=1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **左指针** | **中间值** |  | **目标值与右指针重合** |  |  |  |  |  |  |
| **↓** | **↓** |  | **↓** |  |  |  |  |  |  |
| 11 | 23 | 31 | 37 | 43 | 46 | 52 | 57 | 61 | 72 |

此时目标值大于中间值，既target > nums[pivot]，所以执行left = pivot + 1=2， pivot= left + (right - left) // 2=2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **左指针** | **中间值** | **目标值与右指针重合** |  |  |  |  |  |  |
|  | **↓** | **↓** | **↓** |  |  |  |  |  |  |
| 11 | 23 | 31 | 37 | 43 | 46 | 52 | 57 | 61 | 72 |

此时目标值大于中间值，既target > nums[pivot]，所以执行left = pivot + 1=3， pivot= left + (right - left) // 2=3

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **中间值=目标值** |  |  |  |  |  |  |
|  |  |  | **↓** |  |  |  |  |  |  |
| 11 | 23 | 31 | 37 | 43 | 46 | 52 | 57 | 61 | 72 |

4.4.3. 插值法查找

算法时间复杂度：

插值查找是二分查找的改进    对于插值查找，就是对于二分查找的优化，将二分查找中的mid = (low + high) / 2改为mid = low + (high - low) \* (key - a[low]) / (a[high] - a[low])。插值查找是根据查找关键子key与查找表中最大最小记录关键字比较后的查找方法，核心在于插值计算公式(key-a[low])/(a[high] - a[low])。

代码：

key = int(input("请输入房间号："))  
list=paixu("id")#调用排序函数  
length = len(list)  
low = 0  
high = length - 1  
**while** low <= high:  
 mid = low + int((high - low) \* (key - list[low]["id"]) / (list[high]["id"] - list[low]["id"]))# 计算mid值是插值算法的核心代码  
 **if** key < list[mid]["id"]:  
 high = mid - 1  
 **elif** key > list[mid]["id"]:  
 low = mid + 1  
 **else**:  
 student\_query.append(list[mid])  
 show\_student(student\_query)  
 #print(list[mid])  
 **break**

4.4.4. 插值法查找

算法时间复杂度：

顺序查找算法简单粗暴，从表的一端开始，顺序扫描线性表，依次将扫描到的结点关键宇和给定值K相比较。若当前扫描到的结点关键字与K相等，则查找成功；若扫描结束后，仍未找到关键字等于K的结点，则查找失败。

顺序查找的优点：算法简单，且对表的结构无任何要求，无论是用向量还是用链表来存放结点，也无论结点之间是否按关键字有序，它都同样适用。

顺序查找的缺点：查找效率低，因此，当n较大时不宜采用顺序查找。

代码：

#顺序查找法  
name = ""  
name =input("请输入您要查找的姓名")  
#print(name)  
**if** name == "":  
 print("查无此人")  
**else**:  
 **for** i **in** student\_list:  
 **if** name == i["name"]:  
 student\_query.append(i)  
 show\_student(student\_query)

## 4.5排序模块

排序模块可以实现排序的需求，可以通过冒泡排序法实现按房间号进行排序，通过希尔排序法实现按入住天数进行排序，通过冒泡排序，插入排序，选择排序三种不同的排序方式实现按预付房费进行排序。

开发过程，先读取文本文件，将其中的内容保存在列表中，将列表中每一列，数据转换为字典格式，然后选择下面4个算法对三个需求进行排序。最后将排序结果调用展示模块进行展示。

算法介绍：

4.5.1冒泡排序

时间复杂度：

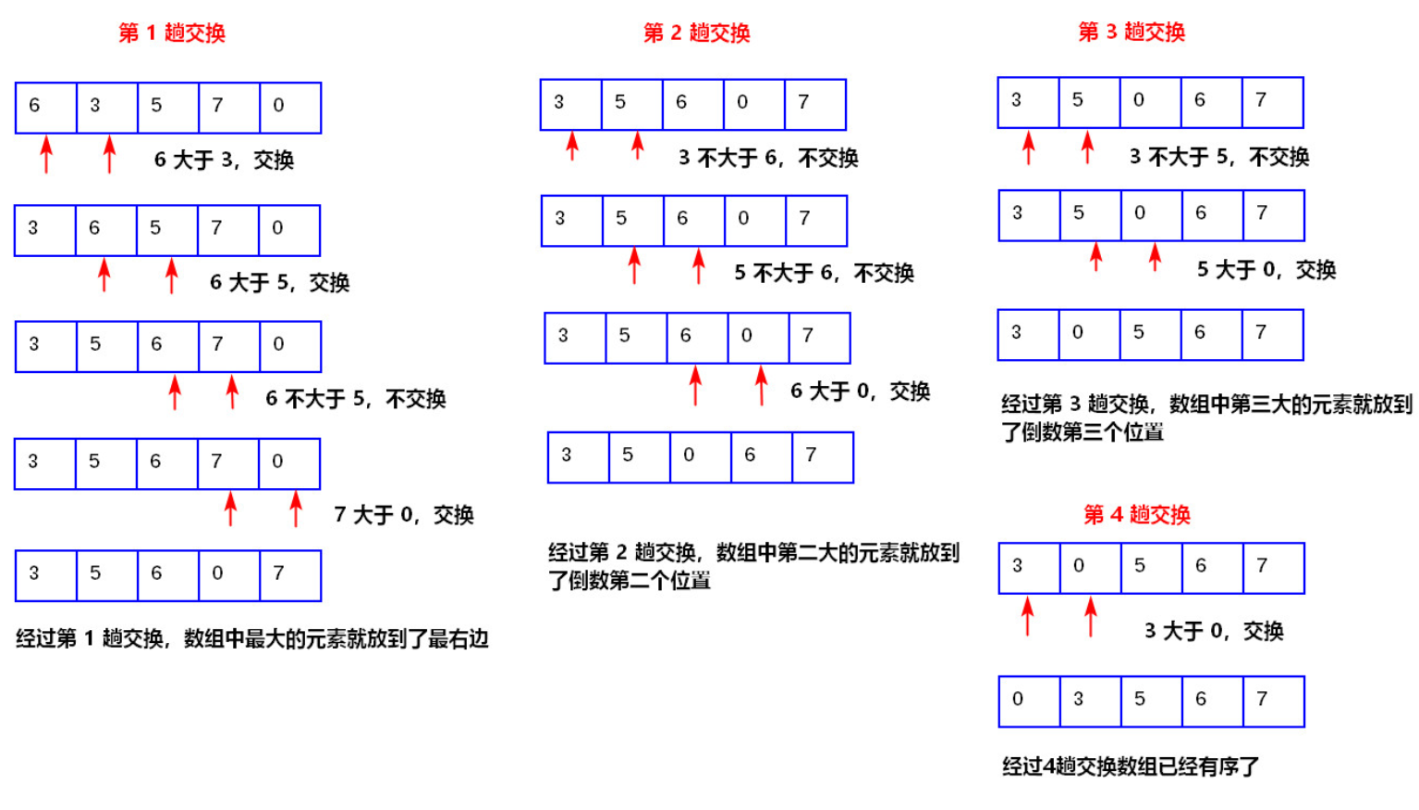
冒泡排序算法的原理如下：

1. 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
2. 对每一对相邻元素做同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。
3. 针对所有的元素重复以上的步骤，除了最后一个。
4. 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

代码：

datas=student\_new  
lens=len(datas)  
**for** i **in** range(lens-1,-1,-1):  
 **for** j **in** range(i):  
 **if** int(datas[j]["id"])>int(datas[j+1]["id"]):  
 datas[j],datas[j+1]=datas[j+1],datas[j]

冒泡排序图解：

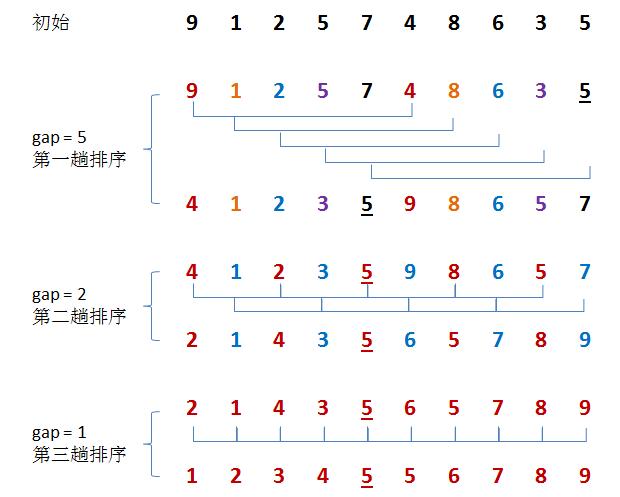


4.5.2.希尔排序

时间复杂度：

希尔排序是希尔（Donald Shell）提出的一种排序方法，也属于插入排序，但是简单插入排序的高效版本，也称为缩小增量排序。基本思想是将待排序元素进行增量分组，然后在分组组内进行插入排序，随着增量的减少，每个分组组内的元素越来越多，直至增量减至1时，所有元素都分到同一个组内，执行插入排序后完成整个排序操作。

图解：



* + 1. 插入排序

插入排序，一般也被称为直接插入排序。对于少量元素的排序，它是一个有效的算法。插入排序是一种最简单的排序方法，它的基本思想是将一个记录插入到已经排好序的有序表中，从而一个新的、记录数增1的有序表。在其实现过程使用双层循环，外层循环对除了第一个元素之外的所有元素，内层循环对当前元素前面有序表进行待插入位置查找，并进行移动

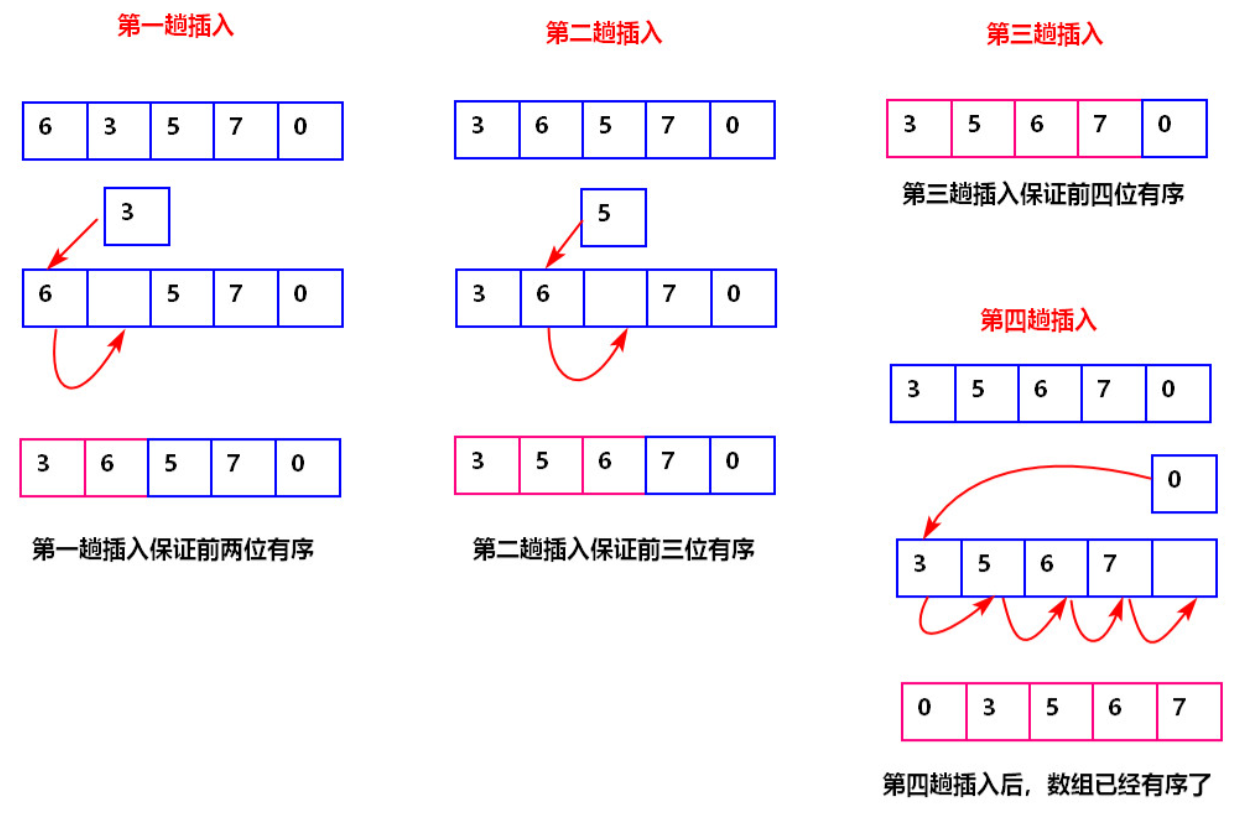
第1趟插入：将第2个元素插入前面的有序子序列，此时前面只有一个元素，当然是有序的

第2趟比较：将第3个元素插入前面的有序子序列，前面的2个元素是有序的

......

第n-1趟比较：将第n个元素插入前面的有序子序列，前面n-1个元素是有序的

图解：



4.5.4.选择排序

选择排序是一种简单直观的排序算法。它的工作原理是：第一次从待排序的数据元素中选出最小（或最大）的一个元素，存放在序列的起始位置，然后再从剩余的未排序元素中寻找到最小（大）元素，然后放到已排序的序列的末尾。以此类推，直到全部待排序的数据元素的个数为零。选择排序是不稳定的排序方法。

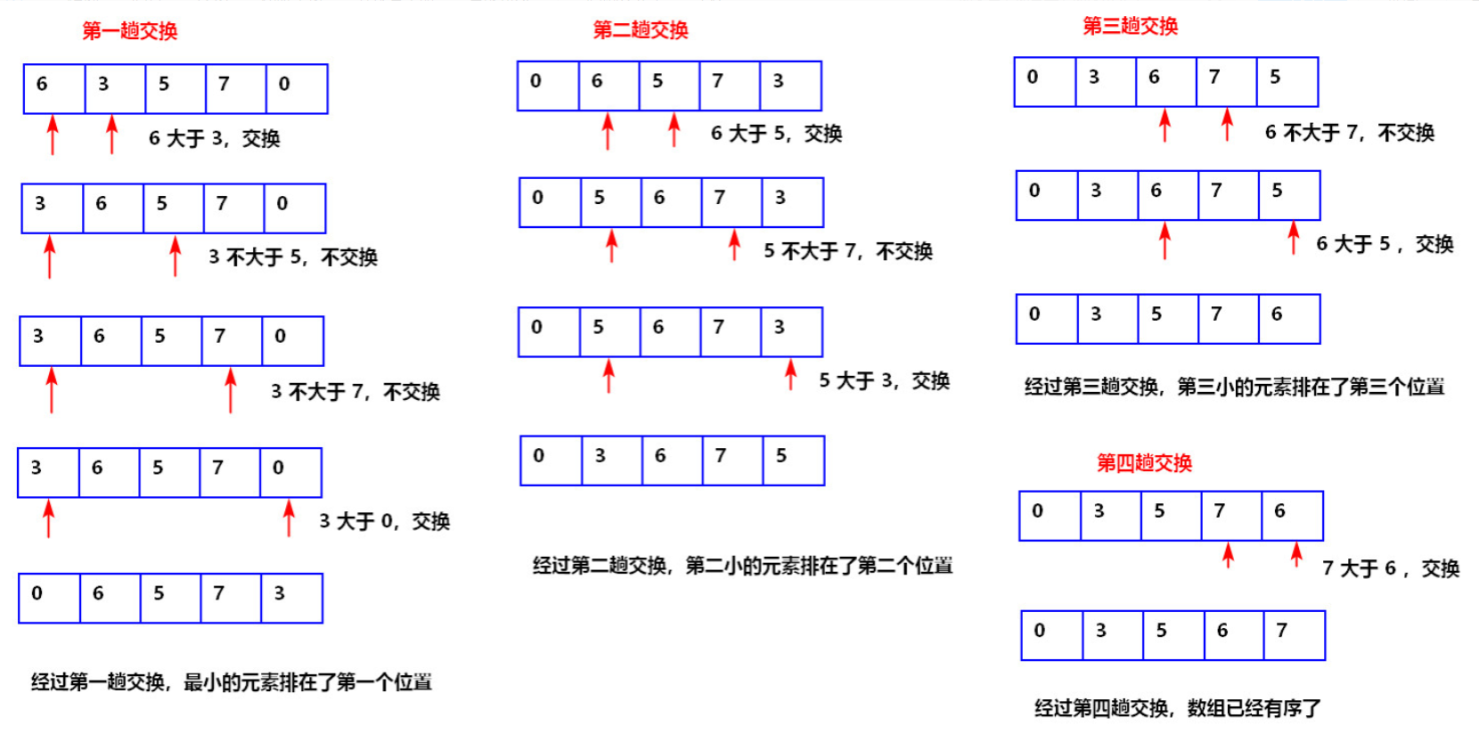
第1趟比较：拿第1个元素依次和它后面的每个元素进行比较，如果第1个元素大于后面某个元素，交换它们，经过第1趟比较，数组中最小的元素被选出，它被排在第一位

第2趟比较：拿第2个元素依次和它后面的每个元素进行比较，如果第2个元素大于后面某个元素，交换它们，经过第2趟比较，数组中第2小的元素被选出，它被排在第二位

......

第n-1趟比较：第n-1个元素和第n个元素作比较，如果第n-1个元素大于第n个元素，交换它们

图解：



## 4.6统计模块

统计模块可以对各种类型房间进行统计，并将结果展示出来，主要开发思路为：定义多个变量用于保存不同类型房间数量，然后遍历列表，不同类型空房间+1。管理员可以选择显示不同类型房间剩余数量，①.统计空房间，②.统计已入住房间，③.统计豪华房入住情况。④.统计双人房入住情况。⑤.统计标间入住情况。

代码：

**def total**():  
 empty\_room=0  
 empty\_room\_list=[]  
 full\_room=0  
 full\_room\_list=[]  
 empty\_hao\_room=0  
 empty\_hao\_room\_list=[]  
 empty\_biao\_room=0  
 empty\_biao\_room\_list=[]  
 empty\_shuang\_room=0  
 empty\_shuang\_room\_list=[]  
 filename = "binguan.txt"  
  
 **if** os.path.exists(filename): # 判断文件是否存在  
 **with** open(filename, 'r') **as** rfile: # 打开文件  
 binguan\_old = rfile.readlines() # 读取全部内容  
  
  
 **for** binguan **in** binguan\_old:  
 #顺序法查找并统计  
 d = dict(eval(binguan))  
 **if** d["days"] ==0:  
 empty\_room +=1  
 empty\_room\_list.append(d["id"])  
 **if** d["hometype"] == "标间":  
 empty\_biao\_room +=1  
 empty\_biao\_room\_list.append(d["id"])  
 **elif** d["hometype"]=="双人房":  
 empty\_shuang\_room +=1  
 empty\_shuang\_room\_list.append(d["id"])  
 **elif** d["hometype"]=="豪华房":  
 empty\_hao\_room +=1  
 empty\_hao\_room\_list.append(d["id"])  
 **else**:  
 full\_room +=1  
 full\_room\_list.append(d["id"])  
  
 print("""  
 & & & & & & & & & & & & & & & &  
 & &  
 & ①.统计空房间 &  
 & ②.统计已入住房间 &  
 & ③.统计豪华房入住情况 &  
 & ④.统计双人房入住情况 &  
 & ⑤.统计标间入住情况 &  
 & &  
 & & & & & & & & & & & & & & & &  
 """ )  
 mode = input("请输入序号选择您要统计的信息")  
 **if** mode =="1":  
 print("一共有{}间空房间".format(empty\_room))  
 print("以下为空房间的房间号：\n",empty\_room\_list)  
 **elif** mode =="2":  
 print("有客的房间一共有{}间".format(full\_room))  
 print("以下为已经入住的房间号:\n",full\_room\_list)  
 **elif** mode =="3":  
 print("豪华房一共有{}间空房\n 以下为空房房号：{}\n".format(empty\_hao\_room,empty\_hao\_room\_list))  
 **elif** mode =="4":  
 print("双人房一共有{}间空房\n 以下为空房房号：{}\n".format(empty\_shuang\_room,empty\_shuang\_room\_list))  
 **elif** mode =="5":  
 print("标间一共有{}间空房\n 以下为空房房号：{}\n".format(empty\_biao\_room,empty\_biao\_room\_list))  
 **else**:  
 print("请输入正确的序号")  
 mark = input("是否继续统计？（输入y继续，输入其他默认退出）：")  
 **if** mark == "y":  
 total() # 重新执行修改操作

## 4.7展示所有信息模块

此模块供其他各模块调用，例如排序，例如查找，最后的结果展示时调用该模块

**def show\_binguan**(binguanList):  
 **if not** binguanList:  
 print("无数据信息\n")  
 **return** format\_title = "{:^6}{:^12}\t{:^8}\t{:^6}\t{:^10}{:^6}"  
 print(format\_title.format("ID", "名字", "身份证号", "入住天数", "房间类型","预付金额"))  
 format\_data = "{:^6}{:^12}\t{:^8}\t{:^14}\t{:<10}{:<6}"  
 **for** info **in** binguanList:  
 print(format\_data.format(str(info.get("id")), info.get("name"),str(info.get("IdCard")), str(info.get("days")), info.get("hometype"), str(info.get("yu\_money"))))

# 五、测试

针对不同模块进行测试，此部分仅做简单用例测试

测试模块：初始化模块

设计需求：要求只初始化一次

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试模块 | 测试输入 | 预计结果 | 实际结果 |
| A001 | 初始化 | 删除binguan.txt文件 | 自动创建binguan.txt文件并完成初始化 | 符合预计结果 |
| A002 | 初始化 | 不删除binguan.txt文件 | 跳过初始化，进入菜单 | 符合预计结果 |

符合需求与设计，测试通过

测试模块：录入顾客信息模块

模块需求：

1. 能够通过选择不同房间号的房间，以及不同类型的房间

2. 当输入的房间已经有人入住时，要返回提示

3. 当输入的房间号有误时，要返回提示

4. 系统能够录入顾客身份证号，姓名，入住天数

5. 顾客身份证号应在五位数以上（假设），一个身份证可开多个房间

6. 入住天数需在一天以上，否则返回提示信息，重新录入

7. 系统可以根据顾客入住天数以及入住的房间类型自动计算房费

8. 开房记录需保存留档

测试思路，选取正常输入作为对照组，再进行边界值，等价类法测试

有效等价类：1001<=id<=1010 或 2001<=id<=2010 d[“days”]>=1，d[“IdCard”]>=10000

无效等价类: id>2010,或id<1001,或1010<id<2001 d[“days”]<1, d[“IdCard”]<10000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试模块 | 测试输入 | 预计结果 | 实际结果 |
| B001 | 开房模块-房间号，姓名，身份证号 | 1001，李明，180374118，5 | 打开文本文binguan.txt，里面有刚刚输入进去的信息 | 符合预计结果 |
| B002 | 开房模块-房间号 | 1000 | 返回提示：您输入的房间号有误，请检查正确的房间号后重新录入!然后重新录入 | 不符合预计结果，未返回提示信息，直接重新录入 |
| B003 | 开房模块-房间号 | 2011 | 返回提示：您输入的房间号有误，请检查正确的房间号后重新录入!然后重新录入 | 不符合预计结果，未返回提示信息，直接重新录入 |
| B004 | 开房模块-房间号 | 1011 | 返回提示：您输入的房间号有误，请检查正确的房间号后重新录入!然后重新录入 | 不符合预计结果，未返回提示信息，直接重新录入 |
| B005 | 开房模块-房间号 | 1001（重复录入同一房间号） | 返回提示：该房间已经有客人了，请您核对房间号 | 符合预期结果 |
| B006 | 开房模块-入住天数 | -1 | 返回提示：您输入的天数数据不符合规范，请重新输入 | 不符合预计，能直接录入不规范数据 |
| B007 | 开房模块-入住天数 | 1.123 | 返回提示：您输入的天数数据不符合规范，请重新输入 | 不符合预期，直接退出循环 |
| B008 | 开房模块-身份证号 | Abcdef | 返回提示：您的输入有误，请重新输入 | 符合预期 |
| B009 | 开房模块-身份证号 | 999 | 请输入正确的身份证号 | 符合预期 |

测试完毕，该模块不符合预期需求，重新进行开发

重新开发后再次测试已符合预期需求

测试模块：查找模块

需求重述：

1．系统能够通过房间号，身份证号，和姓名三种方式查询

2．当输入以上三种信息输入有误，不符合规范时，需返回提示

3．当输入以上三种信息符合规范，但系统中未找到时，需返回提示

设计规范：房间号或身份证和姓名输入错误会返回提醒，身份证号需要五位数以上,

有效等价类：1=<选择序号<=3, 1001<=房间号 =1010 或 2001<=房间号<=2010 身份证号>=10000

无效等价类: 房间号>2010,或房间号<1001,或1010<房间号<2001 身份证号<10000

测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试模块 | 测试输入 | 预计结果 | 实际结果 |
| C001 | 查找模块-序号选择 | 1 | 按房间号查询 | 符合预计结果 |
| C002 | 查找模块-序号选择 | -1 | 请输入正确的数字 | 符合预计结果 |
| C003 | 查找模块-序号选择 | 4 | 请输入正确的数字 | 符合预计结果 |
| C004 | 查找模块-按房间号查询 | 1001 | 当房间有人入住时，返回该房间顾客信息，没人时返回提示空房间 | 符合预计结果 |
| C005 | 查找模块-按房间号查询 | 1000 | 您输入的房间号有误，请核对后再输入 | 符合预计结果 |
| C006 | 查找模块-按身份证号查询 | 180374118（该身份证号开过房） | 返回身份证号为输入身份证号的顾客信息 | 符合预计结果 |
| C007 | 查找模块-按身份证号查询 | ASDFG | 您输入的身份证号不规范，请核对 | 报错程序结束 |
| C008 | 查找模块-按身份证号查询 | 1111 | 请输入正确的身份证号 | 符合预计结果 |
| C009 | 查找模块-按身份证号查询 | 111111(符合规范，但是该身份证未开房) | 您所输入的身份证号没有开房信息，请重新核对！ | 符合预计结果 |
| C010 | 查找模块-按姓名查询 | 李晓明（该人已经登记开房） | 返回该顾客信息 | 符合预计结果 |
| C011 | 查找模块-按姓名查询 | Aaaaa(该人未登记入住) | 没有找到名为Aaaaa的顾客,请核对您的输入 | 符合预计结果 |

测试结束后，对不符合预计结果部分重新开发后测试通过

测试模块：退房模块

需求重述：

1. 系统仅能支持通过房间号退房

2. 退房成功与否都需要返回提示

3. 若所退房间为空房间需返回提示信息

4. 退房记录需保存留档，不能因退房而被系统删除

设计规范：房间号仅能为数字，不能输入其他数据

有效等价类：1=<选择序号<=3, 1001<=房间号 =1010 或 2001<=房间号<=2010

无效等价类: 房间号>2010,或房间号<1001,或1010<房间号<2001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试模块 | 测试输入 | 预计结果 | 实际结果 |
| D001 | 退房模块 | 1001（该房间有人入住） | {}号房间，顾客{}退房成功，查看日志文档有退房记录 | 符合预计结果 |
| D002 | 退房模块 | 1111 | 您输入的房间号有误请核对后再输入 | 符合预计结果 |
| D003 | 退房模块 | AASDAD | 您输入的房间号不规范，请核对 | 符合预计结果 |
| D004 | 退房模块 | 1002（该房间为空间） | 你所要退的房间为空房间,请再次核对您的输入 | 符合预计结果 |

测试模块：修改信息模块

需求重述：

1. 修改功能仅能对已经入住的房间中顾客信息修改，否则返回提示

2. 修改功能中对身份证号，入住天数等数据规范要求与开房功能中要求一致

3. 修改记录也需要保存留档，该记录永久保存，不能被修改，删除

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试模块 | 测试输入 | 预计结果 | 实际结果 |
| E001 | 修改模块-房间号 | 1001（该房间有顾客入住） | 可以正常进入下一步，输入姓名 | 符合预计结果 |
| E002 | 修改模块-房间号 | 1000 | 返回提示：您输入的房间号有误，请检查正确的房间号后重新录入! | 符合预计结果 |
| E003 | 修改模块-房间号 | 1002（空房间） | 返回提示：您抱歉，该房间还未入住，请您核对房间号 | 符合预计结果 |
| E004 | 修改模块-房间号 | asadasd | 您输入的房间号不规范，请核对 | 符合预计结果 |
| E005 | 修改模块-房间号 | 1001（重复录入同一房间号） | 返回提示：该房间已经有客人了，请您核对房间号 | 符合预期结果 |
| E006 | 修改模块-姓名 | Ming | 执行下一步 | 符合预计结果 |
| E007 | 修改模块-身份证号 | 1803744 | 执行下一步 | 符合预计结果 |
| E008 | 修改模块-身份证号 | Abcdef | 返回提示：您的输入有误，请重新输入 | 符合预期结果 |
| E009 | 修改模块-身份证号 | 999 | 请输入正确的身份证号 | 符合预期结果 |
| E010 | 修改模块-天数 | 1 | 修改完毕 | 符合预计结果 |
| E011 | 修改模块-天数 | 0 | 请输入规范的天数 | 符合预计结果 |

测试模块：排序模块

需求重述：

1. 支持通过对房间号，入住天数，以及房费三种数据进行排序

2. 排序展示要求能清晰看到排序后结果，最好可以对比排序前后结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试模块 | 测试输入 | 预计结果 | 实际结果 |
| F001 | 排序模块-按房间号排序 | 1 | 按房间号从小到大排序 | 符合预计结果 |
| F002 | 排序模块-按入住天数排序 | 2 | 按入住天数从小到大排序 | 符合预计结果 |
| F003 | 排序模块-按预付房费降序 | 3 | 按预付房费从大到小排序 | 符合预计结果 |
| F004 | 排序模块-按预付房费排序2 | 4 | 按预付房费从小到大排序 | 符合预计结果 |
| F005 | 排序模块-按预付房费排序3 | 5 | 按预付房费从小到大排序 | 符合预计结果 |
| F006 | 排序模块-序号 | 6 | 请输入正确的序号 | 符合预计结果 |

测试模块：统计模块

需求重述：

1.统计功能需统计空房间，已入住房间，以及豪华房，双人房，标间三种类型房间入住情况

2．统计功能需要展示出空房间的房间号

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 测试模块 | 测试输入 | 预计结果 | 实际结果 |
| G001 | 统计模块-统计空房间 | 1 | 显示空房间数和房间号 | 符合预计结果 |
| G002 | 统计模块-统计已入住房间 | 2 | 显示已经入住的房间数量和房间号 | 符合预计结果 |
| G003 | 统计模块-统计豪华房入住情况 | 3 | 显示豪华房空房数和空房号 | 符合预计结果 |
| G004 | 统计双人房入住情况 | 4 | 显示双人房空房数和空房号 | 符合预计结果 |
| G005 | 统计标间入住情况 | 5 | 显示标间空房数和房号 | 符合预计结果 |
| G006 | 测试序号 | 6 | 请输入正确的序号 | 符合预计结果 |

# 六、心得体会

对于算法，不仅仅是要学会基本的算法思想，更应该融会贯通，将算法融合进项目中。遗憾的是，这个项目可以发挥的算法比较少，所以仅写了排序算法和查找算法。但是算法虽然不多，但是关键是对算法的理解和应用，复杂度是一个算法的重要指标：时间复杂度、空间复杂度。我们希望“多快好省”实现某个功能，即希望执行时间快，又希望占用内存少，然而鱼与熊掌不可得兼，某些情况下就会“空间换时间”、”时间换空间“。复杂度分析是比较难的，通过公式进行推导，这要求有良好的数学基础。因为数据量比较小所以时间差别不大，在学习算法这门课前，我已经在LeetCode上刷了不少算法题了，但是刷算法题和在实际场景中应用算法还是有差别的，刷题只是理解算法，但是项目中运用是彻底将算法融会贯通了。

说一下测试心得，虽然代码早早的就写好了，但是在后面测试的时候还是发现了不少问题，例如在实际场景中，房间号ID，身份证号，若是输入的是字母，而导致程序直接抛出异常而终止退出怎么办，这时我们就需要主动写异常，主动抛出异常，我们不可能让每一次操作都符合程序的正常逻辑，所以我们的程序应该提前考虑到这些异常，输入的房间号没有怎么办，输入的身份证号不对怎么办，都需要提前设计好。

算法中心得，很多同学问我排序算法的时候，我观察到一个细节，很多人只关注到了排序，没有考虑到对那些数据进行排序，以本项目为例，假如对预付房费进行排序时，有的同学可能就只排序了房费，而其他列数据没有动，就导致了结果里面顾客姓名，入住天数，与最后房费不对应了，这个根本原因是在比较排序列大小后，进行交换时，只交换了要排序列的数据，而不是整体一行的数据。理解了这一点就基本上没有问题了。

实际上算法对于数学能力要求还是比较多的，虽然本项目中没有用到图，树等算法，但是在平时他们的应用场景还是不少的，在学习这门课的时候又重新学习了数据结构和线性代数，概率论这几门课程。很快就要实习了，我们平时关注实践较多，但是却忽视了理论部分，而这部分恰恰是我们缺失的一部分。这门课也提醒了我，是时候要重拾理论了，计算机网络和计算机组成原理，线性代数，概率论，数据结构都是时候重新复习一遍了。