第三部分 综合实训

实训案例 1　学生成绩管理系统

1.1　功能介绍

学生成绩管理系统用于实现对学生基本信息的管理，主要包括以下功能。

(1) 输入并存储学生信息：输入学生的学号、姓名和分数，把数据保存在创建的 students.txt 文件中。

(2) 打印学生信息：通过打印函数把学生的所有信息打印在屏幕上。

(3) 修改学生信息：通过查询功能查询某学生是否存在，如果存在就对该学生的信息进行修改，如果不存在则返回到主界面。

(4) 删除学生信息：对相应的学生进行删除操作，如果某学生存在就查找并进行删除。

(5) 按学生成绩进行排序：按照学生的总分从高到低进行排序。

(6) 查找学生信息：输入学生的学号，查找该学生的相关信息，如果查找到就输出该学生的信息，如果没有查找到则提示输入的学号不存在。

1.2　程序设计的思路

将学生信息设计成一个 Student 类，这里假设学生有语文、数学和英语三门课的成绩。

```
class Student: #定义一个学生类
  def __init__(self):
    self.name = ''
    self.ID ='
    self.score1 = 0       #语文成绩
    self.score2 = 0       #数学成绩
    self.score3 = 0       #英语成绩
    self.sum = 0          #总分
```

学生成绩管理系统在开始使用之前先进行初始化，判断 students.txt 文件中是否保存了学生的信息，如果保存了就把文件的内容读取出来，供接下来的操作使用；如果没有保存就初始化一个空的列表，用来保存用户的输入，程序中接下来的所有数据都会保存在该列表中。

在对学生基本信息进行操作(包括查找、修改、删除、排序)时，首先打开 students.txt 文

件，对文件中的内容进行读取操作，由于在文件中保存的内容是以空格进行分隔的，并且每个学生的信息占用一行，所以读出所有的内容，先通过换行进行分隔，得到每个学生的信息，然后对每个学生的信息以空格进行分隔，得到每个学生的详细信息，包括学生的姓名、学号、成绩，形成学生类对象并存入 stulist 列表中。对学生基本信息的所有操作都是针对 stulist 列表进行的，如果是添加学生，则追加写入文件中；如果是删除和修改学生，则在操作完成后将 stulist 列表覆盖写入文件中。

1.3 程序设计的步骤

1. 设计 Student 类。将学生信息设计成一个 Student 类，存储学生的语文、数学和英语成绩。在该类中定义计算总分的方法。

```
class Student:                          #定义一个学生类
  def __init__(self):
    self.name ="
    self.ID ="
    self.score1 = 0                     #语文成绩
    self.score2 = 0                     #数学成绩
    self.score3 = 0                     #英语成绩
    self.sum = 0                        #总分
  def sumscore(self):                   #计算总分
     self.sum=self.score1 + self.score2 + self.score3
  def input(self):                      #输入学生的信息
     self.name = input("请输入学生的姓名: ")
     self.ID = input("请输入学生的 ID: ")
     self.score1 = int(input("请输入学生的语文成绩: "))
     self.score2 = int(input("请输入学生的数学成绩: "))
     self.score3 = int(input("请输入学生的英语成绩: "))
     self.sumscore()
  def output(self,file_object):         #输出到文件中
     print(self.name,self.ID, self.score1, self.score2, self.score3, self.sum)
     file_object.write(self.ID)
     file_object.write(" ")
     file_object.write(self.name)
     file_object.write(" ")
     file_object.write(str(self.score1))
     file_object.write(" ")
     file_object.write(str(self.score2))
     file_object.write(" ")
     file_object.write(str(self.score3))
     file_object.write(" ")
     file_object.write(str(self.sum))
     file_object.write("\n")
```

2. 设计功能函数。

（1）添加学生信息。在添加一个学生信息时，首先判断学号是否已经存在，如果已经存在则取消添加操作；否则由用户选择是否保存，如果保存则以追加方式写入文件。

```
def Add(stulist,stu):                                    #添加一个学生信息
    if searchByID(stulist, stu.ID) == True:              #判断学号是否存在
        print("学号已经存在!")
        return False
    print("是否要保存学生信息?")
    nChoose = input("Choose Y/N")
    if nChoose == 'Y' or nChoose == 'y':
        stulist.append(stu)                              #加入列表
        print(stu.name,stu.ID, stu.score1, stu.score2, stu.score3, stu.sum)
        file_object = open("students.txt", "a")          #"a"为追加方式
        stu.output(file_object)                          #输出到文件中保存
        file_object.close()
        print("保存成功!")
```

(2) 删除学生信息。在删除一个学生信息时,首先遍历 stulist 列表中学生的 ID 是否为要删除的学号,如果是则从 stulist 列表中删除,然后采用覆盖写入方式将 stulist 列表中的剩余学生重新写入文件。

```
def Del(stulist, ID):                                    #删除一个学生信息
count = 0
flag=False
for item in stulist:
     if item.ID == ID:
          stulist.remove(item)                           #从列表中删除
          flag=True                                      #删除成功
          break
     count +=1
if flag==False:                                          #或者 count == len(stulist)
    print("没有该学生学号!")
    return
  file_object = open("students.txt", "w")                #覆盖写入
for stu in stulist:
    stu.output(file_object)
  print("删除保存成功!")
file_object.close()
```

(3) 修改学生信息。在修改一个学生信息时,首先遍历 stulist 列表中学生的 ID 是否为要修改的学号,如果是则输入这个被修改学生的新信息,添加此学生到文件中。

```
def Change(stulist, ID):                #修改学生信息
  count = 0
  flag=False
  for item in stulist:
    if item.ID == ID:
       flag=True
       stulist.remove(item)
       file_object = open("students.txt", "w")
       for stu in stulist:
```

```
            stu.output(file_object)
        file_object.close()
  if flag==False:
     print("没有该学生学号!")
     return
  #输入这个被修改学生的新信息
  stu = Student()
  stu.input()
  Add(stulist,stu) #添加 stu 学生信息到文件中
```

(4) 显示所有学生信息。这里指将 stulist 列表中的学生信息打印到屏幕上。

```
def display(stulist):              #显示所有学生信息
    print("学号\t 姓名 语文 数学 英语 总分")
    for item in stulist:
        #print(item.ID, '\t' ,item.name,'\t', item.score1,'\t',item.score2, '\t',
item.score3, '\t',item.sum)
        #格式化输出
         print("%5s%5s%3d %3d %3d %4d"%(item.ID,item.name,item.score1,item.
score2, item.score3,item.sum))
```

(5) 成绩排序。成绩排序指按学生成绩由高至低进行排序,在实现的时候使用比较排序算法,按照总分对 stulist 中保存的学生信息进行排序。

```
def Sort(stulist):                                  #按学生成绩排序
     insertSort(stulist)                            #比较排序
     display(stulist)

def insertSort(stulist):                            #比较排序
for i in range(len(stulist)-1):
    for j in range(i+1,len(stulist)):
        if stulist[i].sum<stulist[j].sum:           #交换
            temp = stulist[i]
            stulist[i] = stulist[j]
            stulist[j] = temp
```

(6) 查询学生信息。在查询学生信息时,按学号(ID)进行搜索,如果该学生的学号在 stulist 列表中,则将该学生的信息打印出来。

```
def Search(stulist, ID):           #查询学生信息
  print("学号\t 姓名\t 语文\t 数学\t 英语\t 总分")
count = 0
for item in stulist:
     if item.ID == ID:
         print(item.ID, '\t' ,item.name,'\t', item.score1,'\t',item.score2, '\t',
item.score3, '\t',item.sum)
         break
     count = count + 1
if count == len(stulist):
     print("没有该学生学号!")
```

（7）初始化函数。在从文件中读取学生信息后，以空格进行分隔形成列表 s，将列表 s 形成学生对象实例，依次加入保存所有学生信息的 stulist 列表中。stulist 列表中存放的是学生对象实例。

```
def Init(stulist):                                  #初始化函数
    print("初始化......")
    if os.path.exists('students.txt'):              #判断 students.txt 文件是否存在
        file_object = open('students.txt', 'r')
        for line in file_object:
            stu = Student()
            line = line.strip("\n")
            s = line.split(" ")                     #按空格分隔形成列表
            stu.ID = s[0]
            stu.name = s[1]
            stu.score1 = int(s[2])
            stu.score2 = int(s[3])
            stu.score3 = int(s[4])
            stu.sum = s[5]
            stulist.append(stu)
        file_object.close()
        print("初始化成功!")
    main()
```

3. 设计主函数。主函数是一个无限循环，用于实现用户选择的功能。

```
def main():                 #程序的入口函数
  while True:
    print("**********************")
    print("---------菜单----------")
    print("添加学生信息---------1")
    print("查找学生信息---------2")
    print("删除学生信息---------3")
    print("修改学生信息---------4")
    print("显示所有学生信息---------5")
    print("按照分数排序---------6")
    print("退出程序-------------0")
    print("* * * * * * * * * * * * * * * * * * * * * *")
    nChoose = input("请输入你的选择: ")
    if nChoose == "1":
        stu = Student()
        stu.input()
        Add(stulist,stu)
    if nChoose == '2':
        ID = input("请输入学生的 ID: ")
        Search(stulist, ID)
    if nChoose == '3':
        ID = input("请输入学生的 ID: ")
        Del(stulist, ID)
    if nChoose == '4':
```

```
            ID = input("请输入学生的 ID: ")
            Change(stulist, ID)
        if nChoose == '5':
            display(stulist)
        if nChoose == '6':
            Sort(stulist)
        if nChoose == '0':
            break
#主程序
if __name__ == '__main__':
    stulist =[]
    Init(stulist)                    #调用初始化函数
```

程序的运行结果如下：

```
初始化......
**********************
--------菜单---------
添加学生信息--------1
查找学生信息--------2
删除学生信息--------3
修改学生信息--------4
显示所有学生信息--------5
按照分数排序--------6
退出程序------------0
**********************
请输入你的选择: 1
请输入学生的姓名: 张海
请输入学生的 ID: 98001
请输入学生的语文成绩: 78
请输入学生的数学成绩: 88
请输入学生的英语成绩: 79
张海 98001 78 88 79 245
是否要保存学生信息? Choose Y/N
y
保存成功!
请输入你的选择: 5
学号      姓名    语文   数学    英语    总分
98001     张海    78     88      79      245
98002     赵大强  88     90      98      276
98003     李东方  77     66      77      220
请输入你的选择: 6
学号      姓名    语文   数学    英语    总分
98002     赵大强  88     90      98      276
98001     张海    78     88      79      245
98003     李东方  77     66      77      220
```

思考：在掌握以上方法后，请读者思考商品库存管理系统的实现，具体要求如下。

(1) 商品的信息用类来表示，包含商品编号 id、商品名称 name、商品类别 category、商品库存量 kcl、商品销售量 xsl 等信息。

（2）能够输入商品信息，能够显示所有商品信息。

（3）能够按商品名称、商品类别等查询，可以查询商品库存量小于 5 的商品，查询方式自定。

（4）能够按商品销售量进行排序，并在屏幕上打印排序结果。

（5）能够添加、删除、修改（增加商品库存量、商品销售量）商品的信息。

（6）商品的信息保存在文件中。

实训案例 2　基于文件存储的公交查询系统

2.1　功能介绍

随着公交系统越来越庞大，人们很难得到准确的公交信息，这样就给人们的出行带来了不便，因此急需一个方便、快捷的公交信息查询系统。本系统提供了换乘查询功能、路线查询功能，乘客可以方便地进行查询，以防止乘错车次。本系统主要有 4 个模块，即线路查询模块、站点查询模块、换乘查询模块和后台管理模块。

（1）线路查询模块：可以获得要查询公交所通过的各个站点。

（2）站点查询模块：通过输入的指定站点查询经过该站点的公交。

（3）换乘查询模块：分为公交直达、公交一次换乘，主要体现不可直达需要转车的路线的所有换乘方法。

（4）后台管理模块：用于管理员登录，实现添加、修改、删除公交线路等功能。

2.2　程序设计的思路

公交查询系统程序需要存储各公交线路的站点信息，这里用文件存储线路信息，形式如下：

```
    1s%通利公交公司%长途客运西站%建设路国棉六厂%建设路桐柏路站%建设路文化宫路站%建设路工人路站%碧沙岗公园西门%绿城广场%嵩山路伊河路站%解放军测绘学院%市骨科医院%陇海路站%陇海路京广客运站%陇海路铁英街站%郑州铁路局%锦荣商贸城%福寿街大同路站%火车站%6:00-21:00 票价 1 元 ABCD 卡有效
    1x%火车站%一马路陇海路站%郑州铁路局%陇海路铁英街站%陇海路京广客运站%陇海路站%市骨科医院%解放军测绘学院%嵩山路伊河路站%绿城广场%碧沙岗公园西门%碧沙岗%建设路工人路站%建设路文化宫路站%建设路桐柏路站%建设路国棉六厂%华山路建设路站%通利公交公司%6:00-21:00 票价 1 元 ABCD 卡有效
……
```

其中，1s 为上行，1x 为下行，站点信息之间以%分隔。

公交查询系统程序从文件中读取线路信息，其中线路信息存入 route 字典，线路名存入 routename 列表。route 字典的存储形式为{线路名：线路经过站点的列表}。

例如，以上线路信息存入 route 字典时键名为线路名 1s，键对应的内容为 1s 线路经过站点的列表：

```
print(route['1s'])
print(route['1x'])
```

结果如下：

```
    ['通利公交公司', '长途客运西站', '建设路国棉六厂', '建设路桐柏路站', '建设路文化宫路站', '建设路工人路站', '碧沙岗公园西门', '绿城广场', '嵩山路伊河路站', '解放军测绘学院', '市骨科医院', '陇海路站', '陇海路京广客运站', '陇海路铁英街站', '郑州铁路局', '锦荣商贸城', '福寿街大同路站', '火车站']
    ['火车站', '一马路陇海路站', '郑州铁路局', '陇海路铁英街站', '陇海路京广客运站', '陇海路站', '市骨科医院', '解放军测绘学院', '嵩山路伊河路站', '绿城广场', '碧沙岗公园西门', '碧沙岗', '建设路工人路站', '建设路文化宫路站', '建设路桐柏路站', '建设路国棉六厂', '华山路建设路站', '通利公交公司']
```

当线路信息存入 route 字典后，线路查询、站点查询、换乘功能的实现就可以转换为对 route 字典的相应操作。

2.3 程序设计的步骤

```
import math
route = {}                  #线路信息的字典
routename=[]                #线路名的列表
```

readRote()是读取线路信息的函数。readRote()从 gongjiao.txt 文件中读取线路信息(一行一个线路)，以%分隔后，第一个数据项是线路名(如 1s,201x)，存入 routename，后面的数据项存入 route 字典相应的键-值对中。其中，route 字典中键为线路名，对应值为线路经过站点的列表。

```
def readRote():
   fp=open("gongjiao.txt",'r', encoding='gbk') #注意文本文件编码,也可能为 utf-8
   #UnicodeDecodeError: 'gbk' codec can't decode byte 0xbf: illegal multibyte
    sequence
   while True:
      line=fp.readline()
      if line=="":                          #或者 if not line
        break
      list1=line.split("%")                 #以%分隔
      routename.append(list1[0])            #线路名(如 1s,201x),存入 routename
      route[list1[0]]=list1[1:-1]           #增加键-值对
   fp.close()                               #关闭文件
#print(route.values())                      #返回字典中所有的值(所有线路信息)
   #print(route['1x'])
```

findRote()是实现线路查询功能的函数，判断线路信息字典 route 中是否存在此线路名的键，如果存在就打印出此线路名的键对应的值(即此线路信息)。

```
def findRote():                             #线路查询功能
   findRoteName = input("请输入查询线路名: ")
   if findRoteName in route:
      print(route[findRoteName])            #打印出对应的值(即此线路信息)
   else:
      print("输入有错误! 没有你要查询的线路")
```

findStation()是实现站点查询功能的函数，仅需要遍历线路信息字典 route，判断键对应的值 value(线路经过的站点)是否存在查询站点名，如果存在则打印出键 key(线路名)。

```
def findStation():                    #站点查询功能
   stationName = input("请输入查询站点名: ")
   print('经过此站点的线路有: ',end=' ')
   #遍历字典
   for key,value in route.items():
      if(stationName in value):
         print(key,end='; ')
   print()
```

huanRote()是实现换乘查询功能的函数。此函数的功能比较复杂，首先需要判断起始站点到终点是否有直达线路，如果无直达线路则需要换乘。

对直达的判断比较简单，仅需要判断起始站点和终点是否在同一个线路上。

换乘功能算法是找出经过起点的线路存入 S 列表，找出经过终点的线路存入 D 列表，然后判断 S 列表中的 key1 线路和 D 列表中的 key2 线路是否有相同站点 sameStationName，如果有相同站点 sameStationName，则乘客从起始站点经过 key1 线路到 sameStationName 相同站点，换乘 key2 线路可以到达终点。

```
def huanRote():                                                    #换乘查询功能
   startStationName =input("请输入起始站点名: ")                    #"火车站"
   endStationName =input("请输入终点名: ")                          #"绿城广场"
   canGo=False
   #对直达的判断
   for key,value in route.items():
      if(startStationName in value) and (endStationName in value):   #在同一个线路上
         canGo=True
         print("公交可直达线路",key)
   if canGo==False:
      print("公交不可直达")
      #换乘
      S=[]    #经过起点的线路
      D=[]    #经过终点的线路
      for key,value in route.items():
         if(startStationName in value):
            S.append(key)
      for key,value in route.items():
         if(endStationName in value):
            D.append(key)
      #判断线路之间是否有相同站点
      for key1 in S:
        for key2 in D:
          if hasSameStation(key1,key2):
            sameStationName=hasSameStation(key1,key2)
            n1=stationNum(startStationName,sameStationName,key1)
            n2=stationNum(endStationName,sameStationName,key2)
```

```
                print("经过 key1 线路"+key1+n1+"站到"+hasSameStation(key1,key2)+
                "换乘 key2 线路"+key2+n2+"站到达"+endStationName)

#判断线路之间是否有相同站点
def hasSameStation(key1,key2):
    for stationName in route[key1]:
        if stationName in route[key2]:
            return stationName
    return False
#两个站点之间的站数
def stationNum(Station1,Station2,routeName):
    for i in range(len(route[routeName])):
        stationName=route[routeName][i]
        if Station1==stationName:
            i1=i
        if Station2==stationName:
            i2=i
    return str(int(math.fabs(i1-i2)))
```

main()是主函数,是公交查询系统程序的入口函数,主要通过循环实现用户选择的功能。

```
def main():         #主函数
    readRote()
    while True:
        print("**********************")
        print(u"---------菜单----------")
        print(u"线路查询------------1")
        print(u"站点查询------------2")
        print(u"换乘查询------------3")
        print(u"添加线路信息--------4")
        print(u"退出程序------------0")
        print("**********************")
        nChoose = input("请输入你的选择: ")
        if nChoose == "1":
           findRote()
        elif nChoose == "2":
           findStation()
        elif nChoose == "3":
           huanRote()
        elif nChoose == "4":
            #添加线路信息
            fp=open("gongjiao.txt",'a', encoding='gbk')
            addRoteName = input("请输入添加线路名: ")
            addRoteContent = input("请输入添加线路经过的站点及运行时间,以%分隔: \n")
            fp.write(addRoteName+"%"+addRoteContent+"\n")
            fp.close()
            routename.append(addRoteName)                    #添加线路名的列表
            route[addRoteName]=addRoteContent                #添加线路信息的字典
        elif nChoose == "0":
            break
```