

## 一. 卡信息链表结点定义

前面实验中卡信息在内存中存放在结构体数组, 数组在内存中要占用连续的存储空间, 并且大小固定, 要么会造成内存资源浪费, 要么会出现存储资源不足。本次实验采用链表这种动态的数据结构, 在需要存储时动态分配内存资源。

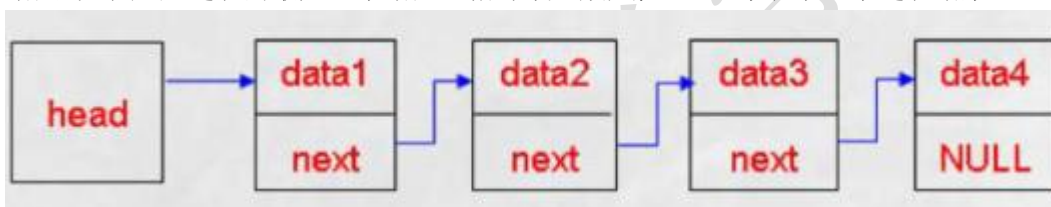
在 model.h 文件中 `endif` 之前定义链表(单链表)的结点结构体, 结点的数据域是卡信息结构体, 指针域是指向下一个结点的指针(这是把卡信息联系在一起的关键), 并用 `typedef` 声明了结点类型和指向结点的指针类型, 便于后面使用。

```

20 typedef struct CardNode
21 {
22     Card data;    //结点数据区
23     struct CardNode* next; //指向下一个结点的指针
24 }CardNode, *lpCardNode;
25
26 #endif
27

```

链表包括头指针, 结点和表尾。头指针只存放一个地址(即链表第一个结点所在的地址); 中间的结点数据域存放卡信息, 指针域存放下一个结点的地址(即指向下一个结点); 表尾是链表的最后一个结点, 指针域一般赋值 `NULL` 来表示整个链表结束



链表的各个结点在内存中一般不是连续存放的, 所以要找某个结点的信息, 需要从头指针开始, 按照指针的指向依次访问每个结点

(提示: 如果指针未初始化, 编译器会把栈内存上的未初始化指针全部填成 `0xffffffff`, 当成字符串看就成了“烫烫烫烫。。。”; 编译器会把堆内存上的未初始化指针全部填成 `0xcdcdcdcd`, 当成字符串看就成了“屯屯屯屯。。。”, 编译器把野指针自动初始化后, 便于调试程序时发现问题)

## 二. 初始化链表

### 1. 定义卡信息链表的指针变量

在 `card_service.cpp` 中定义一个全局变量, 指向卡信息链表结点的指针变量(待链表功能实现后, 删除原来的卡结构体数组定义变量)

```

9
10 Card aCard[50]; //卡结构体数组
11 int nCount=0; //卡结构体数组中的实际卡信息数
12 lpCardNode cardList=NULL; //卡信息链表类型变量
13

```

### 2. 定义函数

在 `card_service.cpp` 中定义 `initCardList` 函数(对应头文件中加该函数声明), 代码如下:

其中用到 `malloc` 函数和 `sizeof` 操作符(使用前可能需要 `#include <stdlib.h>`)

`sizeof` 操作符返回一个对象或者类型所占的内存字节数, 这里 `sizeof(CardNode)` 返回卡信息链表结点结构体类型所占用的内存大小;

`malloc` 函数为运行中的程序在堆区动态开辟一定大小的内存空间, 函数原型:

```
void * malloc (unsigned int size)
```

函数的返回值是一个指向所分配区域的无类型指针（即区域第一个字节的地址），这里需要**强制转换**为指向链表结点的指针类型 lpCardNode

```
card_service.h  card_service.cpp x  model.h
(未知范围)
105 //[[函数名] initCardList
106 //[[功能] 初始化卡信息链表
107 //[[参数] void
108 //[[返回值] int型, TRUE表示成功, FALSE表示失败
109 int initCardList()
110 {
111     lpCardNode head = NULL;
112     if(cardList == NULL)
113     {
114         // 为链表头结点分配内存
115         head = (lpCardNode)malloc(sizeof(CardNode));
116
117         // 如果分配成功, 则将链表指向
118         if(head != NULL)
119         {
120             head->next = NULL;
121             cardList = head;
122             return TRUE;
123         }
124     }
125     return FALSE;
126 }
127
```

思考:

1. 程序在内存中的分区有哪些? 各有什么特点?
2. malloc 函数在堆区分配存储区域, 必须用 free 函数释放? 否则?

### 三. 释放卡信息链表

1. 在 card\_service.cpp 中定义 releaseCardList 函数（对应头文件中加该函数声明），遍历链表的每个结点，每个结点通过 free 函数释放，最后链表指针为 NULL，代码如下：

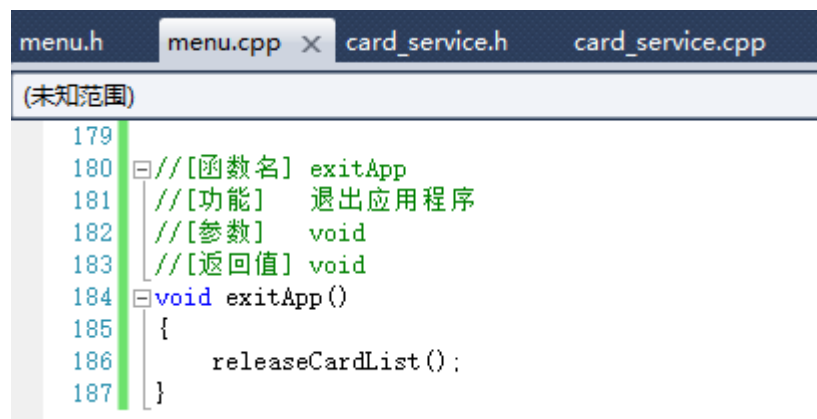
```
card_service.h  card_service.cpp x  model.h
(未知范围)
131 //[[函数名] releaseCardList
132 //[[功能] 释放卡信息链表
133 //[[参数] void
134 //[[返回值] void
135 void releaseCardList()
136 {
137     lpCardNode cur = cardList;
138     lpCardNode next = NULL;
139
140     while(cur != NULL)
141     {
142         next = cur->next; // 释放cur结点前, 用next保存它的后继结点
143         free(cur);       // 释放cur结点
144         cur = next;
145     }
146     cardList = NULL;
147 }
```

其中使用 free 函数，其原型：void free (void\* p)

释放由 p 指向的动态空间，p 是最近一次调用 malloc 函数得到的函数返回值

## 2.调用 releaseCardList 函数

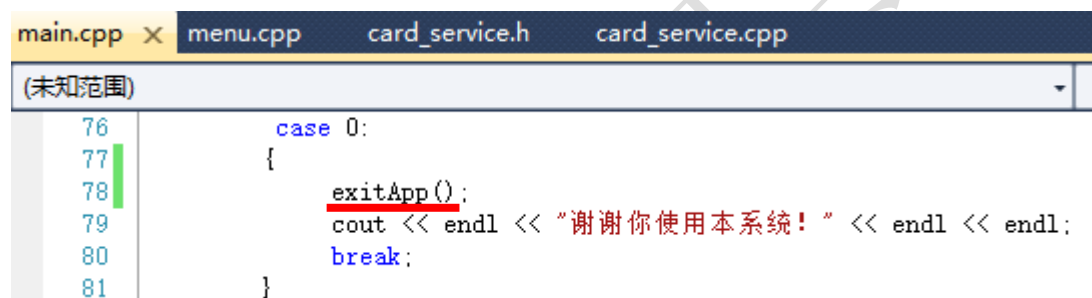
在 menu.cpp 中定义 exitApp 函数(对应头文件中加该函数声明)，调用 releaseCardList 函数：



```

menu.h  menu.cpp x  card_service.h  card_service.cpp
(未知范围)
179
180 // [函数名] exitApp
181 // [功能] 退出应用程序
182 // [参数] void
183 // [返回值] void
184 void exitApp()
185 {
186     releaseCardList();
187 }
  
```

在 main.cpp 中退出选项 (case 0) 后调用 exitApp 函数

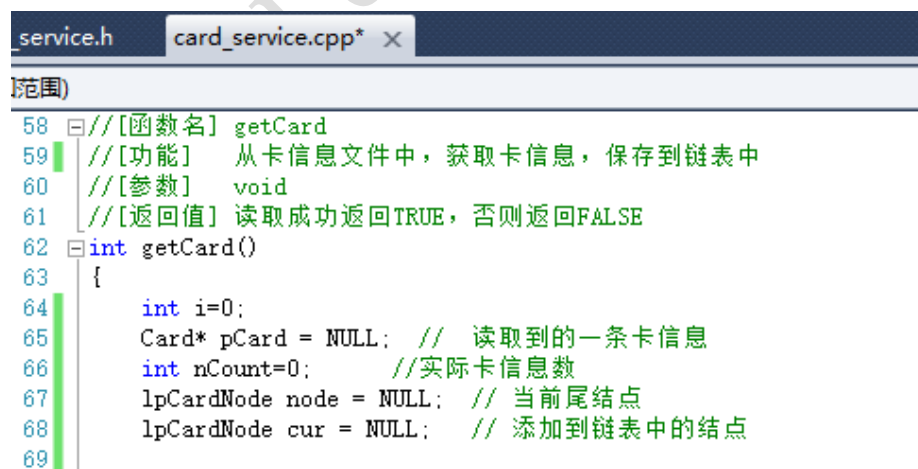


```

main.cpp x  menu.cpp  card_service.h  card_service.cpp
(未知范围)
76         case 0:
77         {
78             exitApp();
79             cout << endl << "谢谢你使用本系统!" << endl << endl;
80             break;
81         }
  
```

## 四. 从文件读取卡信息到链表

不管链表中有无数据，都先清空链表中的数据，保证链表和文件中数据同步一致。修改 card\_service.cpp 中 getCard 函数如下：



```

_service.h  card_service.cpp* x
[范围]
58 // [函数名] getCard
59 // [功能] 从卡信息文件中，获取卡信息，保存到链表中
60 // [参数] void
61 // [返回值] 读取成功返回TRUE，否则返回FALSE
62 int getCard()
63 {
64     int i=0;
65     Card* pCard = NULL; // 读取到的一条卡信息
66     int nCount=0; // 实际卡信息数
67     lpCardNode node = NULL; // 当前尾结点
68     lpCardNode cur = NULL; // 添加到链表中的结点
69 }
  
```

```

service.h  card_service.cpp* x
范围)
70 // 清除链表中已经存在的数据
71 if(cardList != NULL)
72 {
73     releaseCardList();
74 }
75 // 初始化链表
76 initCardList();
77
78 // 获取文件中卡信息个数
79 nCount = getCardCount(CARDPATH);
80 if(nCount == 0)
81 {
82     return FALSE;
83 }
84 else if(nCount == -1)
85 {
86     cout<<"文件无法打开!"<<endl;
87     return FALSE;
88 }
89
90 // 动态分配内存用来保存所有卡信息，相当于结构体数组，pCard相当于数组名
91 pCard = (Card*)malloc(sizeof(Card)*nCount);
92 if(pCard != NULL)
93 {
94     // 如果返回FALSE，表示读取卡信息失败
95     if(0 == readCard(pCard, CARDPATH))
96     {
97         free(pCard);
98         return FALSE;
99     }
100
101     // 将读取的卡信息，保存到链表中
102     for(i = 0, node = cardList; i < nCount; i++)
103     {
104         // 为结点分配内存
105         cur = (lpCardNode)malloc(sizeof(CardNode));
106         // 如果分配内存失败，则返回
107         if(cur == NULL)
108         {
109             free(pCard);
110             return FALSE;
111         }
112         // 初始化新的空间，全部赋值为0
113         memset(cur, 0, sizeof(CardNode));
114
115         // 将卡信息保存到结点中
116         cur->data = pCard[i]; //结构体指针当数组名使用
117         cur->next = NULL;
118
119         // 将结点添加到链表尾部
120         node->next = cur;
121         node = cur;
122     }
123
124     // 释放内存
125     free(pCard);
126     return TRUE;
127 }
128 return FALSE;
129 }

```

思考: 1. 用 malloc 函数分配的堆内存空间要 free 释放, 这里 cur 指针变量的怎么没有 free? 在哪儿 free?

2. 结构体指针可以用作数组名? 反过来, 数组名是否可以作为指针使用?

3. node 和 cur 变量起什么作用?

五. 从链表中精确查询卡信息

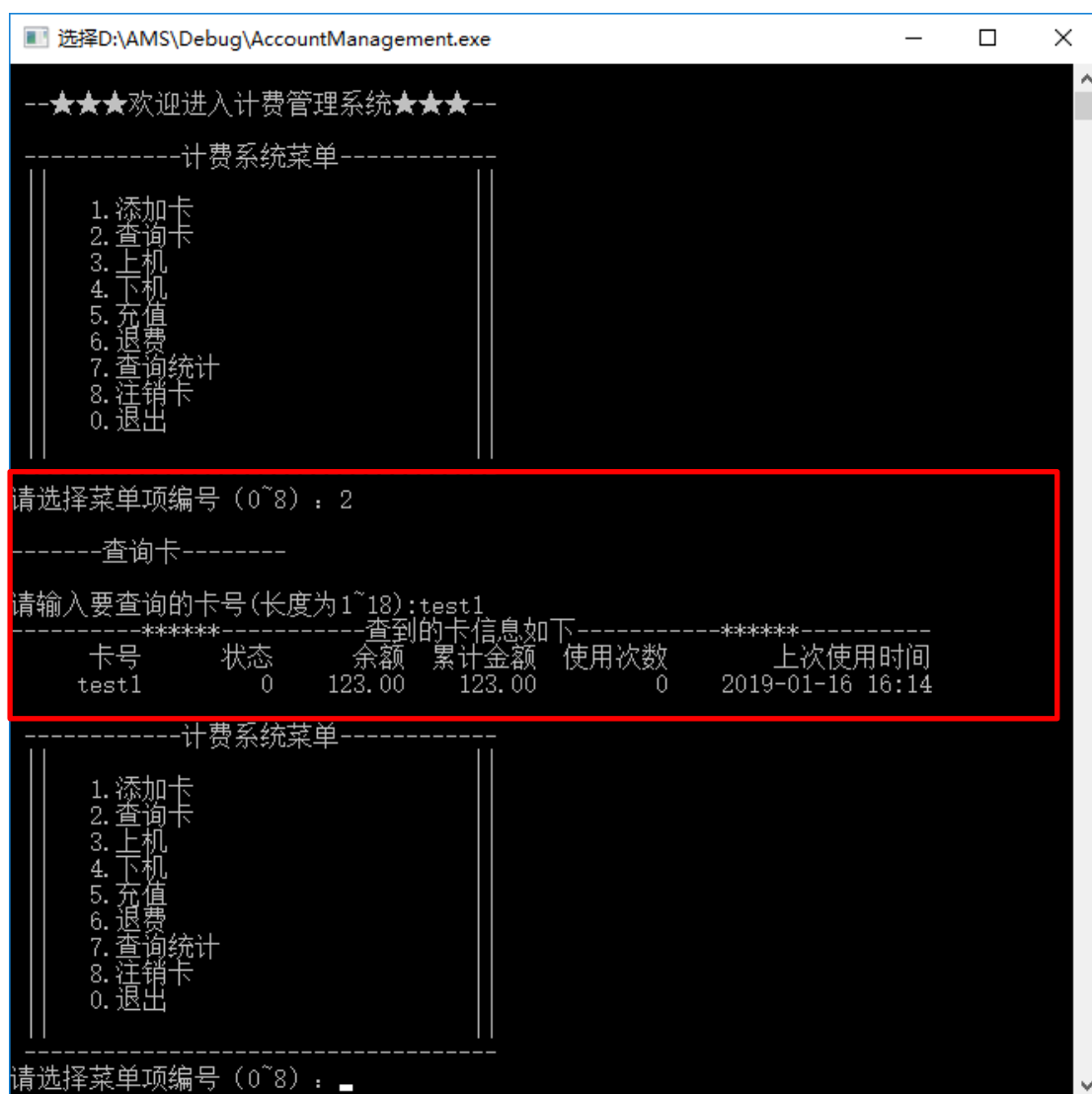
这个功能不仅在查询卡信息是调用, 还在添加卡时查重时调用, 修改 queryCard 函数如下:

```
d_service.h  card_service.cpp x
和范围)
30  //[[函数名] queryCard
31  //[[功能]   在链表中查找指定卡号的卡信息
32  //[[参数]   用户输入的要查询的卡号地址
33  //[[返回值] 链表中查询到的卡信息地址, 没有找到返回NULL
34  Card* queryCard(const char* pName)
35  {
36      Card* pCard = NULL;
37      lpCardNode p; //用于查找中迭代, 依次指向链表中的每个结点
38
39      //从卡信息文件中读取卡信息到链表, 失败返回NULL
40      if (FALSE==getCard())
41      {
42          return NULL;
43      }
44  }
```

```
service.h  card_service.cpp x
范围)
45      //指向链表头结点, 用于迭代
46      p=cardList;
47      //在链表中查找指定卡号
48      while(p!=NULL)
49      {
50          if(strcmp(p->data.aName,pName)==0)
51          {
52              pCard=&(p->data);
53              return pCard;
54          }
55          else
56          {
57              p=p->next;
58          }
59      }
60      //没有找到, 返回NULL
61      return pCard;
62  }
```

思考: 1. 查询时可不可以直接使用 cardList 迭代 (cardList=cardList->next)? 为什么要多使用一个 p 链表结点指针变量?

六. 重新编译连接并运行程序



#### 七. 从链表中模糊查询卡信息

模糊查询是根据用户输入的关键字，从链表中查找所有包含相同关键字的卡信息。在 `card_service.cpp` 中定义 `queryCards` 函数（对应头文件中加该函数声明），遍历链表，用 `strstr` 函数判断是否包含用户输入的卡号关键字，如果找到，就增大卡信息结构体(相当于数组)的存储空间，把找到的卡信息保存于其中，最后返回卡信息结构体，代码如下：

其中用到 **strstr** 函数，函数原型：`char *(strstr)(const char *s1, const char *s2);`  
 函数参数是字符串 `s1`, `s2`；函数功能是找到 `s2` 字符串在 `s1` 字符串中第一次出现的位置；返回该位置的指针，如找不到，返回空指针。

其中用到 **realloc** 函数，函数原型：`void *realloc(void *p, unsigned int size);`  
 如果已经通过 `malloc` 函数或 `calloc` 函数获得了动态空间，想改变其大小，可以用 `realloc` 函数重新分配，用 `realloc` 函数将 `p` 所指向的动态空间的大小改变为 `size`，`p` 的值不变。如果重分配不成功，返回 `NULL`。

如果新分配的内存减少，`realloc` 仅仅是改变索引的信息，返回原指针，原来内存中数据有可能丢失；如果 `size = 0`，则等价于释放内存

如果是将分配的内存扩大，则有以下情况：

1) 如果当前内存段后面有需要的内存空间，则直接扩展这段内存空间，`realloc()` 将返回原指

针。

2) 如果当前内存段后面的空闲字节不够, 那么就使用堆中的第一个能够满足这一要求的内存块, 将目前的数据复制到新的位置, 并将原来的数据块释放掉, 返回新的内存块位置。

3) 如果申请失败, 将返回NULL, 此时, 原来的指针仍然有效。

这里代码中将原来的空间增大一个Card结构体的大小, 以便保存下一个找到的卡信息。

```

d_service.h  card_service.cpp x
[范围]
178 //[[函数名] queryCards
179 //[[功能] 在卡信息链表中, 模糊查询包含的所有卡信息
180 //[[参数] pName: 指向用户输入的要查询的卡号; pIndex: 指向查到的卡信息数变量
181 //[[返回值] 指向卡信息结构体的指针
182 Card* queryCards(const char* pName, int* pIndex)
183 {
184     lpCardNode node = NULL;
185     Card* pCard = NULL; //保存查询到的符合条件的卡信息
186
187     //从卡信息文件中读取卡信息到链表, 失败返回NULL
188     if(FALSE==getCard())
189     {
190         return NULL;
191     }
192
193     // 首先分配一个Card大小内存空间
194     pCard = (Card*)malloc(sizeof(Card));
195     if(pCard == NULL)
196     {
197         return NULL;
198     }
199

```

```

_service.h  card_service.cpp x
[范围]
200 // 从链表的头结点指向的下一个结点开始遍历
201 node = cardList->next;
202
203 // 遍历链表, 结点为空表示到达链表尾部
204 while(node != NULL)
205 {
206     // 判断在遍历到的结点的卡号中, 查找是否包含pName字符串
207     if(strstr(node->data.aName, pName) != NULL)
208     {
209         // 如果有, 则保存结点中的数据
210         pCard[*pIndex] = node->data;
211         (*pIndex)++;
212
213         // 重新为指针分配内存, 包含已有的内容
214         pCard = (Card*)realloc(pCard, ((*pIndex)+1)*sizeof(Card));
215     }
216
217     // 移到链表的下一个结点
218     node = node->next;
219 }
220
221 return pCard;
222 }

```

思考：数组与数组变量可以直接赋值吗？结构体与结构体变量可以直接赋值吗？

（提示：不使用strstr函数，也可使用C++中string类的find方法）

八. 调用queryCards函数实现模糊查询

在menu.c中修改query函数，用户可以选择精确查询还是模糊查询，代码如下：

```

menu.cpp x card_service.cpp
(未知范围)
140 void query()
141 {
142     Card* pCard=NULL;
143     char name[18]; //存放要查询的用户名
144     char aLastTime[30] ; //存放指定格式字符串的时间
145     int icha = 0;
146     int nIndex = 0; // 卡查询到的信息数量
147     int i;
148
149     cout<<"请输入要查询的卡号(长度为1~18):";
150     cin>>name;
151     cin.clear();
152     cin.sync();
153
154     cout<<"1. 精确查询, 2. 模糊查询 (输入1或2) :";
155     cin>>icha;
156     cin.clear();
157     cin.sync();
158
159     if(icha==1) //选择精确查询
160     {
161         pCard = queryCard(name);
162     }
163     else //默认其他选择模糊查询
164     {
165         pCard = queryCards(name, &nIndex);
166     }
167
168     // 如果pCard为NULL, 表示没有该卡的信息
169     if(pCard == NULL)
170     {
171         cout<<"-----没有该卡的信息! -----"<<endl;
172     }
173     else
174     {
175         cout<<"-----查到的卡信息如下-----"<<endl;
176         // 输出表格的表头
177         cout<<setw(10)<<"卡号"<<setw(10)<<"状态"<<setw(10)<<"余额";
178         cout<<setw(10)<<"累计金额"<<setw(10)<<"使用次数"<<setw(20)<<"上次使用时间"<<endl;
179
180         if(icha==1) //精确查询结果输出
181         {
182             //将time_t类型时间转换为字符串, 字符串格式为"年-月-日 时: 分"
183             timeToString(pCard->tLastTime, aLastTime);
184
185             //输出查到的卡信息
186             cout<<setw(10)<<pCard->aName<<setw(10)<<pCard->nStatus; //一行输出书写语句过长, 分行
187             cout<<setw(10)<<fixed <<setprecision(2)<<pCard->fBalance;
188             cout<<setw(10)<<fixed <<setprecision(2)<<pCard->fTotalUse;
189             cout<<setw(10)<<pCard->nUseCount<<setw(20)<<aLastTime<<endl;
190         }
    
```



```

menu.cpp x card_service.cpp
(未知范围)
191         else    //模糊查询结果输出
192         {
193             for(i = 0; i < nIndex; i++)
194             {
195                 //将time_t类型时间转换为字符串，字符串格式为“年-月-日 时：分”
196                 timeToString(pCard[i].tLastTime, aLastTime);    //结构体指针当数组名使用
197
198                 //输出查到的卡信息
199                 cout<<setw(10)<<pCard[i].aName<<setw(10)<<pCard[i].nStatus; //一行输出书写语句过长，分行
200                 cout<<setw(10)<<fixed <<setprecision(2)<<pCard[i].fBalance;
201                 cout<<setw(10)<<fixed <<setprecision(2)<<pCard[i].fTotalUse;
202                 cout<<setw(10)<<pCard[i].nUseCount<<setw(20)<<aLastTime<<endl;
203             }
204             // 释放动态分配的内存
205             free(pCard);
206         }
207         pCard = NULL;
208     }
209 }

```

### 九. 重新编译连接并运行程序

D:\AMS\Debug\AccountManagement.exe

```

--★★★欢迎进入计费管理系统★★★--

-----计费系统菜单-----

1. 添加卡
2. 查询卡
3. 上机
4. 下机
5. 充值
6. 退费
7. 查询统计
8. 注销卡
0. 退出

-----

请选择菜单项编号 (0~8) : 2

-----查询卡-----

请输入要查询的卡号(长度为1~18): test
1. 精确查询, 2. 模糊查询 (输入1或2) : 2

*****查到的卡信息如下*****
卡号      状态      余额      累计金额      使用次数      上次使用时间
test1      0          123.00      123.00          0          2019-01-16 16:14
test2      0          111.00      111.00          0          2019-01-16 16:14

-----计费系统菜单-----

1. 添加卡
2. 查询卡
3. 上机
4. 下机
5. 充值
6. 退费
7. 查询统计
8. 注销卡
0. 退出

-----

请选择菜单项编号 (0~8) : 

```

✿ 十. 有兴趣的同学可以完成**附加功能**: 修改程序, 增加查询选项, 实现查询所有卡号信息的功能

#### 十一. 总结

本次任务的层次结构和主要调用关系

