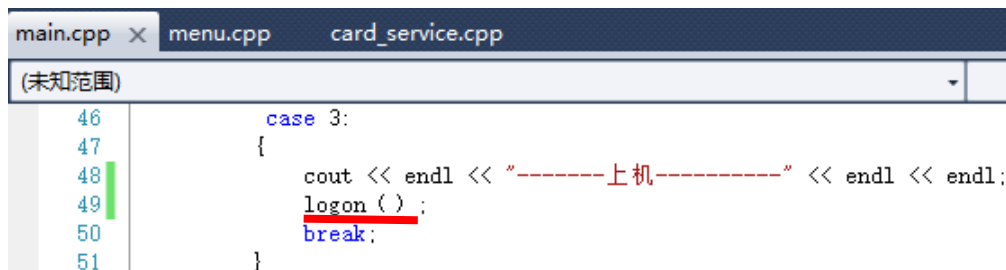


一. 更新上机卡

对已添加的卡,当用户通过输入相匹配的卡号和密码,开始上机,在界面显示上机信息,更新上机状态。

1. 查找上机卡

在 main.cpp 的 main 函数中,用户选择“3.上机”时,调用 logon 函数,实现上机操作。修改代码如下:

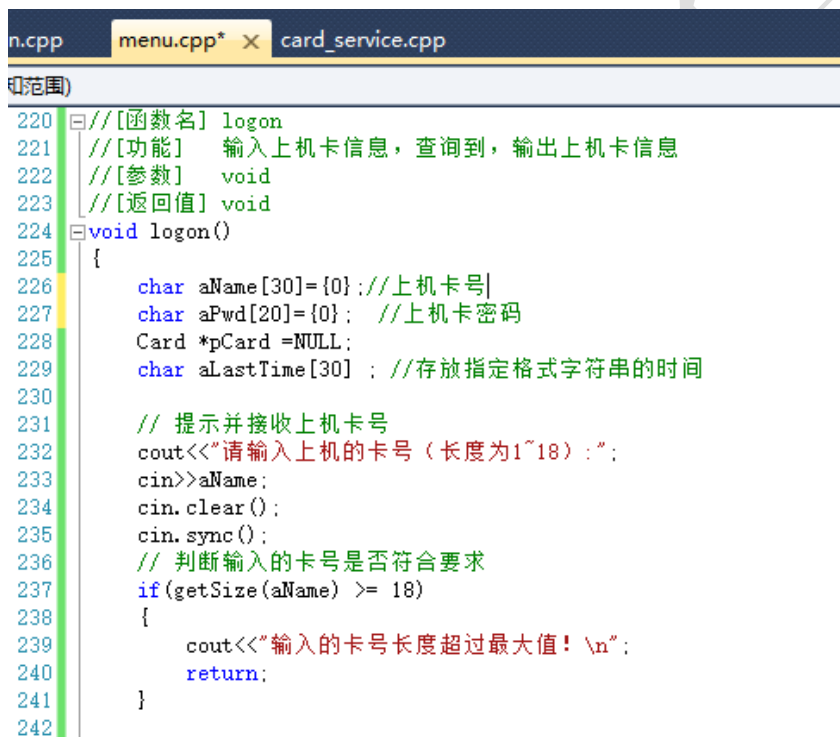


```

46         case 3:
47         {
48             cout << endl << "-----上机-----" << endl << endl;
49             logon();
50             break;
51         }

```

在 menu.cpp 中创建 logon 函数(同时在对头文件中添加函数声明),提示并接收用户输入的卡号和密码,获取上机的卡信息结果后,输出不同信息。如果未获得卡信息,表示上机失败,提示用户“上机失败”,否则,通过卡结构体指针,向界面列表输出卡号,余额和上机时间,上机时间默认为卡的最后使用时间,转换成字符串格式输出。代码如下:



```

220 // [函数名] logon
221 // [功能] 输入上机卡信息, 查询到, 输出上机卡信息
222 // [参数] void
223 // [返回值] void
224 void logon()
225 {
226     char aName[30]={0}; //上机卡号
227     char aPwd[20]={0}; //上机卡密码
228     Card *pCard =NULL;
229     char aLastTime[30]; //存放指定格式字符串的时间
230
231     // 提示并接收上机卡号
232     cout<<\"请输入上机的卡号（长度为1~18）:\";
233     cin>>aName;
234     cin.clear();
235     cin.sync();
236     // 判断输入的卡号是否符合要求
237     if(getSize(aName) >= 18)
238     {
239         cout<<\"输入的卡号长度超过最大值! \n\";
240         return;
241     }
242

```

```

n.cpp  menu.cpp*  card_service.cpp
和范围)
243 // 提示并接收上机密码
244 cout<<"请输入上机密码(长度为1~8): ";
245 cin>>aPwd;
246 cin.clear();
247 cin.sync();
248 // 判断输入的密码是否符合要求
249 if(getSize(aPwd) >= 8)
250 {
251     cout<<"输入的密码长度超过最大值! \n";
252     return;
253 }
254

n.cpp  menu.cpp  card_service.cpp
和范围)
255 //开始上机, 获取上机结果
256 pCard=doLogon(aName, aPwd);
257
258 //根据上机结果, 提示不同信息
259 if(pCard==NULL)
260 {
261     cout<<"上机失败! \n";
262 }
263 else
264 {
265     cout<<"-----**-----上机的卡信息如下-----**-----"<<endl;
266     // 输出表格的表头
267     cout<<setw(10)<<"卡号"<<setw(10)<<"余额"<<setw(20)<<"上机时间"<<endl;
268
269     //将time_t类型时间转换为字符串, 字符串格式为"年-月-日 时: 分"
270     timeToString(pCard->tLastTime, aLastTime); //结构体指针当数组名使用
271
272     //输出查到的卡信息
273     cout<<setw(10)<<pCard->aName; //一行输出书写语句过长, 分行
274     cout<<setw(10)<<fixed <<setprecision(2)<<pCard->fBalance;
275     cout<<setw(20)<<aLastTime<<endl;
276 }
277 }

```

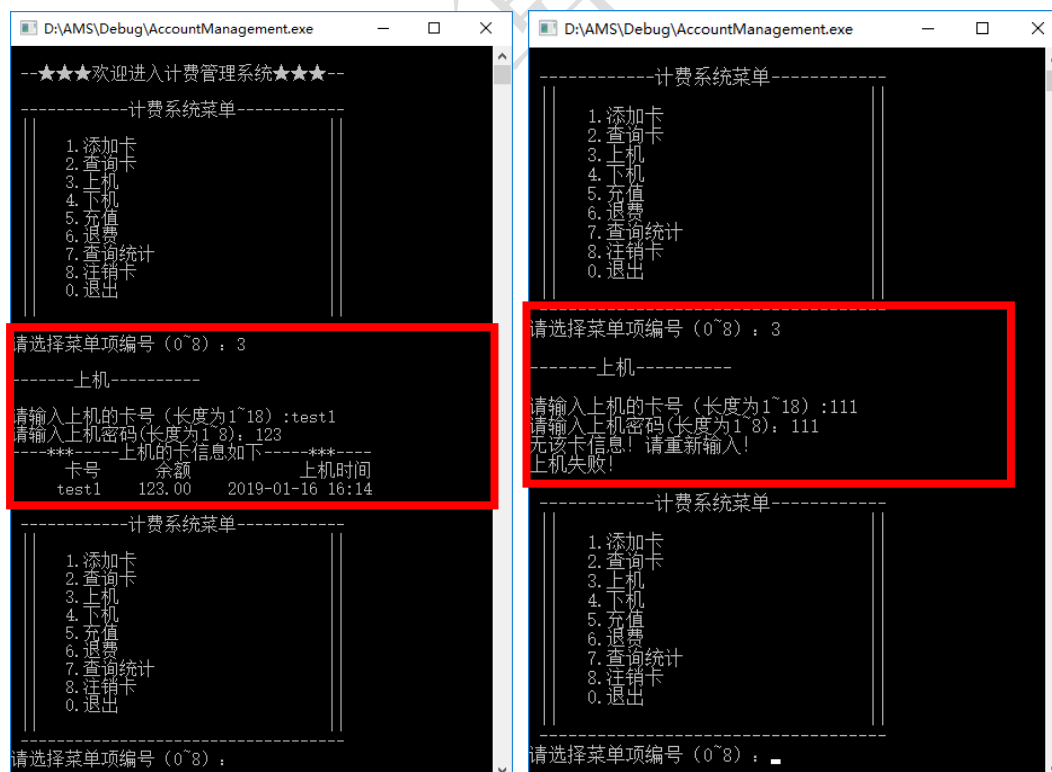
在 card_service.cpp 中定义 doLogon 函数（同时在对应头文件中添加函数声明），查找上机卡的卡信息。先将文件中卡信息读取到链表，然后遍历链表查找匹配的卡信息，代码如下：

```

in.cpp  menu.cpp  card_service.cpp* x
知范围)
222 //函数名 doLogon
223 //功能 从文件读取卡信息到链表，在链表中查询匹配的卡信息
224 //参数 pName: 上机卡号; pPwd: 上机密码
225 //返回值 上机卡信息结构体
226 Card* doLogon(const char* pName, const char* pPwd)
227 {
228     lpCardNode node = NULL; //当前结点
229
230     //从卡信息文件中读取卡信息到链表，失败返回NULL
231     if (FALSE == getCard())
232     {
233         return NULL;
234     }
235     //当前结点指向头结点
236     if (cardList != NULL) node = cardList;
237
238     while (node != NULL)
239     {
240         //在链表中查找是否有对应的卡信息
241         if ((strcmp(node->data.aName, pName) == 0) && (strcmp(node->data.aPwd, pPwd) == 0))
242         {
243             return &node->data;
244         }
245         node = node->next;
246     }
247     if (node == NULL) cout << "无该卡信息！请重新输入！" << endl;
248     return NULL;
249 }

```

编译连接并运行程序：



2. 更新卡信息

查询到上机卡信息后，卡的一些状态信息需要修改，要更新链表中的卡信息，同时更新文件中的卡信息（调用 `updateCard` 函数来实现），更新成功才表示上机成功，再返回该卡的卡信息地址。修改 `doLogon` 函数如下：

```

d_file.cpp  card_service.cpp x
知范围)
222 //函数名 doLogon
223 //功能 从文件读取卡信息到链表，在链表中查询匹配的卡信息，更新链表和文件中对应卡信息
224 //参数 pName: 上机卡号； pPwd: 上机密码
225 //返回值 上机卡信息结构体
226 Card* doLogon(const char* pName, const char* pPwd)
227 {
228     lpCardNode node = NULL; //当前结点
229     int nIndex=0; //匹配的卡信息结点在链表中序号，用于更新卡信息
230
231     //从卡信息文件中读取卡信息到链表，失败返回NULL
232     if (FALSE==getCard())
233     {
234         return NULL;
235     }
236     //当前结点指向头结点
237     if (cardList!=NULL) node=cardList;
238
239     while (node !=NULL)
240     {
241         //在链表中查找是否有对应的卡信息
242         if ((strcmp(node->data.aName, pName)==0) && (strcmp(node->data.aPwd, pPwd)==0))
243         {
244             //更新链表中的卡信息
245             node->data.nStatus=1; //状态为正在使用
246             node->data.nUseCount++; //使用次数加1
247             node->data.tLastTime=time(NULL); //最后使用时间为当前时间
248         }

```

```

d_file.cpp  card_service.cpp x
知范围)
249 //更新文件中的卡信息
250 if (TRUE==updateCard(&node->data, CARDPATH, nIndex))
251 {
252     return &node->data;
253 }
254 else
255 {
256     return NULL;
257 }
258 }
259 node=node->next;
260 nIndex++;
261 }
262 if (node==NULL) cout<<"无该卡信息！请重新输入！"<<endl;
263 return NULL;
264 }

```

在 `card_file.cpp` 中定义 `updateCard` 函数（同时在对应头文件中添加函数声明），代码如下：

```

card_file.cpp x card_service.cpp
(未知范围)
150 // [函数名] updateCard
151 // [功能] 更新卡信息文件中对应的一条卡信息
152 // [参数] pCard: 更新后的卡内容 pPath: 卡信息文件的路径
153 // nIndex: 需要更新的卡信息在文件中的卡序号
154 // [返回值] TRUE 更新成功, FALSE 更新失败
155 int updateCard(const Card* pCard, const char* pPath, int nIndex)
156 {
157     char aBuf[CARDCHARNUM] = {0};
158     char aStart[30]; // 存放转换后的时间字符串
159     char aEnd[30]; // 存放转换后的时间字符串
160     char aLast[30]; // 存放转换后的时间字符串
161     int nLine = 0; // 文件中当前卡序号(行)
162     long lPosition = 0; // 文件位置标记
163
164     // 将time_t类型时间转换为字符串, 字符串格式为"年-月-日 时:分"
165     timeToString(pCard->tStart, aStart);
166     timeToString(pCard->tEnd, aEnd);
167     timeToString(pCard->tLastTime, aLast);
168
169     // 以读写模式打开文件, 是默认方式, 如果失败, 返回FALSE
170     fstream iofile(pPath, ios_base::out | ios_base::in);
171     if(!iofile.is_open())
172     {
173         cout<<"文件无法正确打开! 不能更新卡信息!"<<endl;
174         iofile.close();
175         return FALSE;
176     }
177
178     // 遍历文件, 获取卡在文件中位置
179     while((!iofile.eof()) && (nLine < nIndex))
180     {
181         memset(aBuf, 0, CARDCHARNUM); // 清空字符数组
182         // 逐行读取文件内容
183         iofile.getline(aBuf, CARDCHARNUM);
184         // 获取文件标识位置, 最后一次是找到的位置
185         lPosition = iofile.tellg();
186         nLine++;
187     }
188     // 移到文件标识位置
189     // 注意指针是在该条卡信息之后还是之前!!!
190     // 上面得到的读指针在下一条卡信息开头, 要移到本条卡信息开头处
191     iofile.seekp(lPosition - strlen(aBuf) - 2, ios::beg);
192
193     // 向文件写入更新的卡信息
194     iofile<<pCard->aName<<"##"<<pCard->aPwd<<"##"<<pCard->nStatus<<"##";
195     iofile<<aStart<<"##"<<aEnd<<"##"<<pCard->fTotalUse<<"##"<<aLast<<"##";
196     iofile<<pCard->nUseCount<<"##"<<pCard->fBalance<<"##"<<pCard->nDel<<endl;
197
198     cout<<"-----*****-----卡信息更新到文件成功! -----*****-----"<<endl<<endl;
199
200     // 关闭文件
201     iofile.close();
202     return TRUE;
203 }

```

程序中使用了 `fstream` 类的方法, `fstream` 类对象既可以读, 也可以写。此处, 先读文件, 找到要更新的卡信息处, 再写文件, 将新的卡信息覆盖原来的卡信息。

输入输出文件有一个读写指针(实际上是两个指针, 但是二者联动, 可看作一个指针), 程序中使用了 `tellg` 和 `seekp` 成员函数, 分别用来得到当前读写指针位置和设置当前读写指针位置。

```
tellg();
```

```
tellp();
```

这两个成员函数不用传入参数, 返回一个 `pos_type` 类型的值(整数), 代表当前 `get` 流指针的位置 (用 `tellg`) 或 `put` 流指针的位置(用 `tellp`)。

```
seekg ( off_type offset, seekdir direction );
```

```
seekp ( off_type offset, seekdir direction );
```

这两个成员函数分别用来改变流指针 `get` 和 `put` 的位置。

思考: 1) 语句 `iofile.seekp(lPosition-strlen(aBuf)-2, ios::beg);` 中的 `lPosition-strlen(aBuf)-2` 是什么意思, 为什么?

2) 程序中 `while` 循环的起什么作用?

3. 程序优化

上机的条件, 除了卡号和密码匹配, 还有其他一些条件需要满足, 修改 `card_service.cpp` 中的 `doLogon` 函数如下:



```

239     while(node !=NULL)
240     {
241         //在链表中查找是否有对应的卡信息
242         if ((strcmp(node->data.aName, pName)==0) && (strcmp(node->data.aPwd, pPwd)==0))
243         {
244             //只有未上机的卡才能上机
245             if (node->data.nStatus!=0)
246             {
247                 cout<<\"该卡正在上机!\"<<endl;
248                 return NULL;
249             }
250             //余额大于0的才能上机
251             if (node->data.fBalance<=0)
252             {
253                 cout<<\"余额不足, 请充值后再上机!\"<<endl;
254                 return NULL;
255             }
256             //更新链表中的卡信息
257             node->data.nStatus=1; //状态为正在使用
258             node->data.nUseCount++; //使用次数加1
259             node->data.tLastTime=time(NULL); //最后使用时间为当前时间
260
261             //更新文件中的卡信息

```

编译连接并运行程序:

The screenshot shows a Windows application window titled "D:\AMS\Debug\AccountManagement.exe". The application displays a menu with the following options:

```
--★★★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): 3
-----上机-----

请输入上机的卡号(长度为1~18): cdy
请输入上机密码(长度为1~8): 123
*****卡信息更新到文件成功!*****

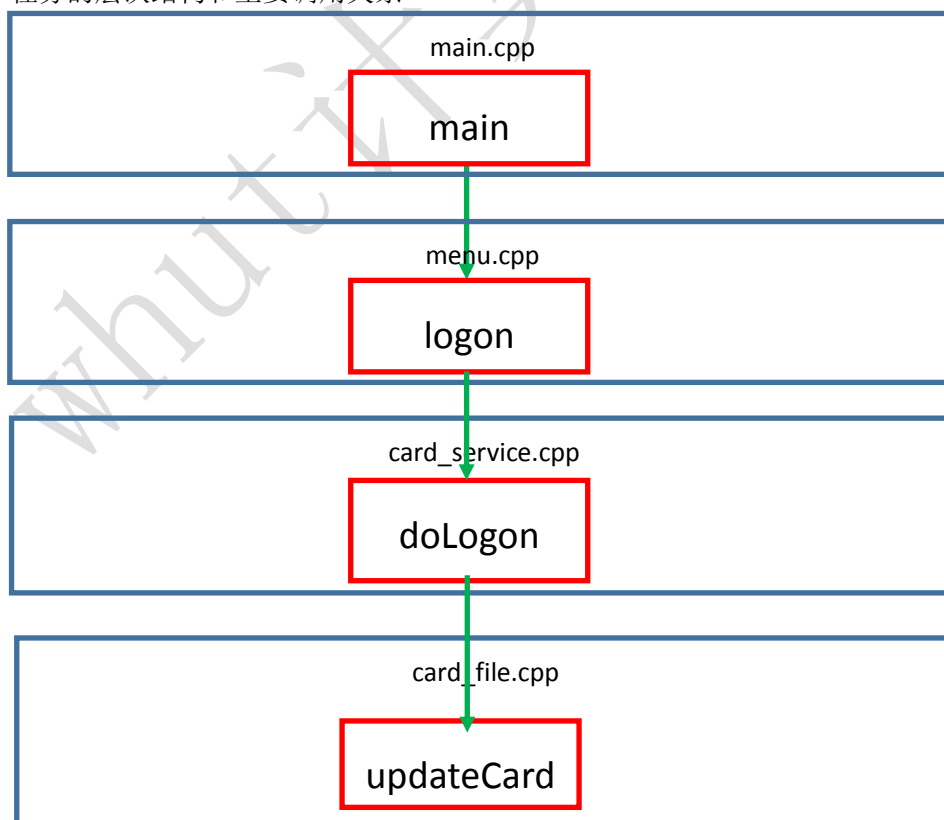
***上机的卡信息如下***
卡号      余额      上机时间
cdy      123.00    2019-02-11 21:43

-----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):
```

A red rectangle highlights the section where the user selects option 3, enters the card number 'cdy' and password '123', and sees the success message and card details.

本节任务的层次结构和主要调用关系



二. 添加消费记录

上机操作时, 还要保存上机卡的消费信息, 采用二进制文件存储。

1. 保存消费信息

在 model.h 文件中, 在 #endif 之前定义保存消费信息的结构体

```

model.h x menu.cpp card_file.cpp card_service.cpp
(未知范围)
25
26 //消费信息结构体
27 typedef struct Billing
28 {
29     char aCardName[18]; //卡号
30     time_t tStart; //上机时间
31     time_t tEnd; //下机时间
32     float fAmount; //消费金额
33     int nStatus; //消费状态, 0-未结算, 1-已经结算
34     int nDel; //删除标识, 0-未删除, 1-已删除
35 }Billing;
36
37 #endif

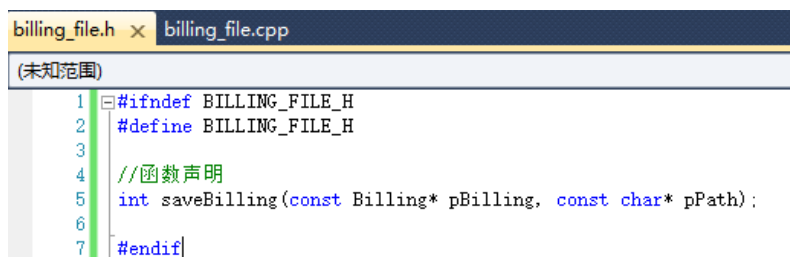
```

添加 billing_file.cpp 和 billing_file.h 文件, 在 cpp 文件中定义 saveBilling 函数 (同时在对头文件添加函数声明), 代码如下:

```

_file.cpp billing_file.cpp x
(范围)
1 #include <iostream>
2 #include <fstream>
3 #include "model.h" // 包含数据类型定义头文件
4 #include "global.h" // 包含全局定义头文件
5
6 using namespace std;
7
8 //[函数名] saveBilling
9 //[功能] 保存计费信息
10 //[参数] pBilling: 计费信息结构体指针; pPath: 保存计费信息文件路径
11 //[返回值] TURE 保存成功; FALSE 保存失败
12 int saveBilling(const Billing* pBilling, const char* pPath)
13 {
14     //以追加方式, 二进制方式写入
15     ofstream ofile(pPath, ios_base::out|ios_base::app|ios_base::binary);
16     if(!ofile.is_open())
17     {
18         cout<<"文件无法正确打开! 不能写入计费信息!"<<endl;
19         ofile.close();
20         return FALSE;
21     }
22
23     // 将计费信息保存到文件中
24     ofile.write((const char *)pBilling, sizeof(Billing));
25
26     // 关闭文件
27     ofile.close();
28     cout<<"-----计费信息成功存入文件!-----"<<endl<<endl;
29     return TRUE;
30 }

```

```

1 #ifndef BILLING_FILE_H
2 #define BILLING_FILE_H
3
4 //函数声明
5 int saveBilling(const Billing* pBilling, const char* pPath);
6
7 #endif

```

二进制文件的读写使用 `read` 和 `write` 成员函数。

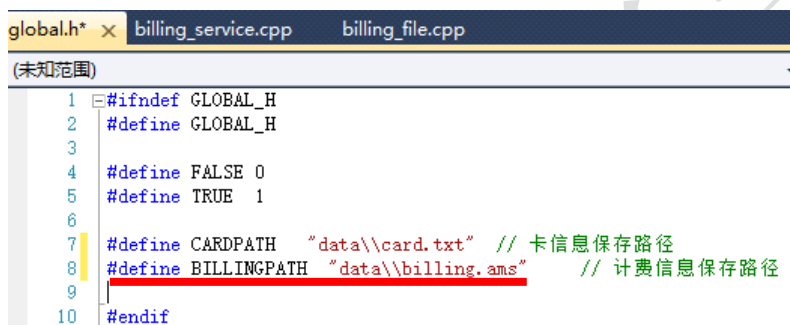
`read(unsigned char *buf,int num);` //从文件中读取 `num` 个字符到 `buf` 指向的缓存中

`write(const unsigned char *buf,int num);` //从 `buf` 指向的缓存写 `num` 个字符到文件中

注意：缓存的类型是 **unsigned char ***，有时可能**需要类型转换**

在 `global.h` 文件中，定义消费信息文件的保存路径

在 `data` 目录下手工新建一个文件 `billing.ams`。（提示：也可以不手工建立这个文件，在第一次添加消费信息时会自动建立这个文件，但是可能会出来一些“不能打开文件”之类的提示信息）



```

1 #ifndef GLOBAL_H
2 #define GLOBAL_H
3
4 #define FALSE 0
5 #define TRUE 1
6
7 #define CARDPATH "data\\card.txt" // 卡信息保存路径
8 #define BILLINGPATH "data\\billing.ams" // 计费信息保存路径
9
10 #endif

```

添加 `billing_service.cpp` 和 `billing_service.h` 文件，在 `cpp` 文件中定义 `addBilling` 函数（同时在对应头文件添加函数声明），先将相关信息写入消费信息结构体，再调用 `billing_file.cpp` 中的 `saveBilling` 函数，将消费信息结构体的内容写入计费信息文件保存下来，代码如下：

```

billing_service.cpp x billing_file.cpp
(未知范围)
1 #include<string.h>
2 #include<time.h>
3
4 #include "model.h" // 包含数据类型定义头文件
5 #include "global.h" // 包含全局定义头文件
6 #include "billing_file.h"
7
8 //[[函数名] addBilling
9 //[[功能] 上机时添加消费信息到文件
10 //[[参数] pName: 上机卡号; pBilling: 指向消费信息结构体
11 //[[返回值] TURE: 保存成功; FALSE: 保存失败
12 int addBilling(const char *pName,Billing *pBilling)
13 {
14     //将消费信息写入结构体
15     strcpy(pBilling->aCardName,pName); //上机卡号
16     pBilling->tStart =time(NULL) ; //上机时间
17     pBilling->tEnd =0; //下机时间
18     pBilling->fAmount=0; //消费金额
19     pBilling->nStatus=0; //消费状态, 0-未结算, 1-已经结算
20     pBilling->nDel=0; //删除标识, 0-未删除, 1-已删除
21
22     //消费信息结构体写入文件
23     if(FALSE == saveBilling(pBilling,BILLINGPATH))
24     {
25         return FALSE;
26     }
27     return TRUE;|
28 }

```

在 card_service.cpp 中修改 doLogon 函数，在更新完卡信息后，添加消费信息，添加成功才表示上机成功。

```

card_service.cpp x billing_service.cpp billing_file.cpp
(未知范围)
222 //[[函数名] doLogon
223 //[[功能] 从文件读取卡信息到链表，在链表中查询匹配的卡信息，
224 // 更新链表和文件中对应卡信息，保存消费信息
225 //[[参数] pName: 上机卡号; pPwd: 上机密码
226 //[[返回值] TURE: 上机成功; FALSE: 上机失败
227 int doLogon(const char* pName,const char* pPwd)
228 {
229     lpCardNode node = NULL; //当前结点
230     int nIndex=0; //匹配的卡信息结点在链表中序号，用于更新卡信息
231     Billing *pBilling = NULL; //计费信息
232     //从卡信息文件中读取卡信息到链表，失败返回NULL
233     if (FALSE==getCard())
234     {
235         return FALSE;
236     }
237     //当前结点指向头结点
238     if(cardList!=NULL) node=cardList;
239

```



```

card_service.cpp* x billing_service.cpp billing_file.cpp
(未知范围)
240 while(node !=NULL)
241 {
242     //在链表中查找是否有对应的卡信息，找到后更新信息
243     if((strcmp(node->data.aName,pName)==0) &&(strcmp(node->data.aPwd,pPwd)==0))
244     {
245         //只有未上机的卡才能上机
246         if(node->data.nStatus!=0)
247         {
248             cout<<"该卡正在上机！"<<endl;
249             return FALSE;
250         }
251         //余额大于0的才能上机
252         if(node->data.fBalance<=0)
253         {
254             cout<<"余额不足，请充值后再上机！"<<endl;
255             return FALSE;
256         }
257         //更新链表中的卡信息
258         node->data.nStatus=1;//状态为正在使用
259         node->data.nUseCount++;//使用次数加1
260         node->data.tLastTime=time(NULL);//最后使用时间为当前时间
261         //更新文件中的卡信息
262         if(FALSE==updateCard(&node->data,CARDPATH,nIndex))
263         {
264             //文件更新失败返回，更新成功才继续添加消费记录
265             return FALSE;
266         }
267         //添加消费记录到文件
268         pBilling=(Billing*)malloc(sizeof(Billing));
269         if(TRUE==addBilling(node->data.aName,pBilling))
270         {
271             free(pBilling);
272             //消费信息保存成功后，则表示上机成功
273             return TRUE;
274         }
275         else
276         {
277             return FALSE;
278         }
279     }
280     node=node->next;
281     nIndex++;
282 }
283 if(node==NULL) cout<<"无该卡信息！请重新输入！"<<endl;
284 return FALSE;
285 }

```

函数返回值类型改为 `int`（修改对应头文件中该函数的声明），修改函数中的返回值为 `TURE` 或 `FALSE`

要调用 `addBilling` 函数，文件前面添加 `#include "billing_service.h"`

2. 保存并显示上机信息

在 `model.h` 文件中，在 `#endif` 之前定义上机信息结构体

```

model.h x menu.cpp card_service.cpp billing_service.
(未知范围)
37 //上机信息结构体
38 typedef struct LogonInfo
39 {
40     char aCardName[18]; //上机卡号
41     time_t tLogon; //上机时间
42     float fBalance; //上机时卡余额
43 }LogonInfo;
44
45 #endif

```

在 card_service.cpp 中修改 doLogon 函数，新增一个参数 pInfo（修改对应头文件中该函数的声明），代码中在成功添加计费信息后，将上机信息保存到上机信息结构体 pInfo 中返回，其中卡号和密码从卡信息中获取，上机时间从消费信息中获取。

```

nu.cpp card_service.cpp x billing_service.cpp billing_file.cpp
(未知范围)
222 //函数名 doLogon
223 //功能 从文件读取卡信息到链表，在链表中查询匹配的卡信息，
224 // 更新链表和文件中对应卡信息，保存消费信息
225 //参数 pName:上机卡号; pPwd:上机密码; pInfo:指向上机信息结构体
226 //返回值 TRUE:上机成功; FALSE:上机失败
227 int doLogon(const char* pName, const char* pPwd, LogonInfo* pInfo)
228 {
    ...
    //添加消费记录到文件
    pBilling = (Billing*)malloc(sizeof(Billing));
    if (TRUE == addBilling(node->data.aName, pBilling))
    {
        //成功上机，上机信息保存到上机信息结构体
        strcpy(pInfo->aCardName, node->data.aName);
        pInfo->tLogon = pBilling->tStart;
        pInfo->fBalance = node->data.fBalance;

        free(pBilling);
        //消费信息保存成功后，则表示上机成功
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}
287

```

在 global.h 文件中定义几个常量

```

global.h x menu.cpp card_service.cpp billing_service.cpp billing_file.cpp
(未知范围)
1 #ifndef GLOBAL_H
2 #define GLOBAL_H
3
4 #define FALSE 0 // 表示失败
5 #define TRUE 1 // 表示成功
6 #define UNUSE 2 // 卡不能使用
7 #define ENOUGHMONEY 3 // 余额不足
8
9 #define CARDPATH "data\\card.txt" // 卡信息保存路径
10 #define BILLINGPATH "data\\billing.ams" // 计费信息保存路径
11
12 #endif

```

在 card_service.cpp 中修改 doLogon 函数中的返回值:

```

nu.cpp card_service.cpp* x billing_service.cpp billing_file.cpp
(未知范围)
243 //在链表中查找是否有对应的卡信息，找到后更新信息
244 if((strcmp(node->data.aName,pName)==0) &&(strcmp(node->data.aPwd,pPwd)==0))
245 {
246     //只有未上机的卡才能上机
247     if (node->data.nStatus!=0)
248     {
249         cout<<"该卡正在上机!"<<endl;
250         return UNUSE;
251     }
252     //余额大于0的才能上机
253     if (node->data.fBalance<=0)
254     {
255         cout<<"余额不足，请充值后再上机!"<<endl;
256         return ENOUGHMONEY;
257     }

```

在 menu.cpp 中修改 logon 函数，删除原来的显示上机卡信息的语句，调用 doLogon 函数后，根据返回值，提示显示上机的不同信息；若上机成功，从返回的上机信息结构体中获取上机信息，列表显示上机信息。代码如下：

```

menu.cpp x card_service.cpp billing_service.cpp billing_file.cpp
(未知范围)
224 void logon()
225 {
226     char aName[30]={0}; //上机卡号
227     char aPwd[20]={0}; //上机卡密码
228     //Card *pCard=NULL;
229     char aLastTime[30]; //存放指定格式字符串的时间
230     LogonInfo *pInfo=NULL;
231     int nResult=0;
232
233     // 提示并接收上机卡号
234     cout<<"请输入上机的卡号（长度为1~18）:";
235     cin>>aName;
236     cin.clear();
237     cin.sync();
238     // 判断输入的卡号是否符合要求
239     if(getSize(aName) >= 18)
240     {
241         cout<<"输入的卡号长度超过最大值！\n";
242         return;
243     }

```

```

menu.cpp* x card_service.cpp billing_service.cpp billing_file.cpp
(未知范围)
255 //开始上机，获取上机结果
256 //pCard=doLogon(aName,aPwd);
257 pInfo = (LogonInfo*)malloc(sizeof(LogonInfo));
258 //根据上机结果，提示输出不同信息
259 nResult = doLogon(aName,aPwd,pInfo);
260
261 switch(nResult)
262 {
263     case 0:
264         cout<<"-----上机失败!-----"<<endl; break;
265     case 1:
266         cout<<"-----***-----上机的卡信息如下-----***-----"<<endl;
267         // 输出表格的表头
268         cout<<setw(10)<<"卡号"<<setw(10)<<"余额"<<setw(20)<<"上机时间"<<endl;
269         //将time_t类型时间转换为字符串，字符串格式为"年-月-日 时:分"
270         timeToString(pInfo->tLogon,aLastTime); //结构体指针当数组名使用
271         //输出上机卡信息
272         cout<<setw(10)<<pInfo->aCardName; //一行输出书写语句过长，分行
273         cout<<setw(10)<<fixed <<setprecision(2)<<pInfo->fBalance;
274         cout<<setw(20)<<aLastTime<<endl;
275         cout<<"-----上机成功!-----"<<endl;
276         break;
277     case 2:
278         cout<<"-----该卡不能使用!-----"<<endl; break;
279     case 3:
280         cout<<"-----余额不足!-----"<<endl; break;
281 }
282 //释放上机信息
283 free(pInfo);
284 }

```

编译并运行程序

```

D:\AMS\Debug\AccountManagement.exe
请选择菜单项编号 (0~8) : 3
-----上机-----
请输入上机的卡号(长度为1~18): cdy
请输入上机密码(长度为1~8): 123
*****卡信息更新到文件成功!*****
*****计费信息成功存入文件!*****
-----***-----上机的卡信息如下-----***-----
卡号      余额      上机时间
cdy      123.00    2019-02-12 15:31
-----上机成功!-----

-----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) :

```

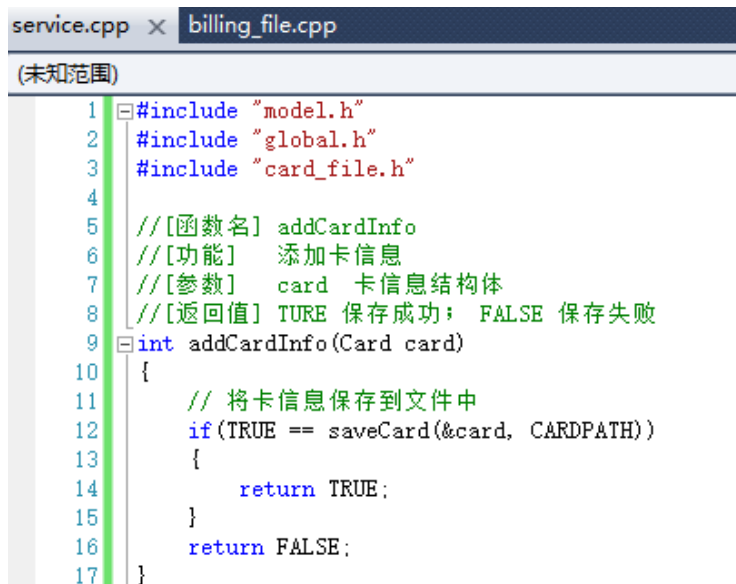
3. 调整程序结构

程序中涉及到同层次的两个业务逻辑处理，卡信息管理 card_service.cpp 和消费信息管

理 billing_service.cpp, 处理上机操作时, card_service.cpp 要调用 billing_service.cpp 中的函数, 混淆了程序的层次和功能。

新建 service.cpp 文件 (以及 service.h 头文件), 分别管理 card_service.cpp 和 billing_service.cpp 文件, 让 card_service.cpp 只处理与卡信息链表相关的业务逻辑, billing_service.cpp 只处理与消费信息链表相关的业务逻辑。

1) 添加 service.cpp 文件, 在其中定义 addCardInfo 函数 (对应头文件添加函数声明) 处理添加卡信息的功能。其中调用 card_file.cpp 文件的 saveCard 函数, 将卡信息保存到文件。



```

1  #include "model.h"
2  #include "global.h"
3  #include "card_file.h"
4
5  //[函数名] addCardInfo
6  //[功能] 添加卡信息
7  //[参数] card 卡信息结构体
8  //[返回值] TURE 保存成功; FALSE 保存失败
9  int addCardInfo(Card card)
10 {
11     // 将卡信息保存到文件中
12     if(TRUE == saveCard(&card, CARDPATH))
13     {
14         return TRUE;
15     }
16     return FALSE;
17 }
  
```

在 menu.cpp 文件中修改 add 函数, 在最后判断保存卡信息是否成功时, 调用 service.cpp 文件中的 addCardInfo 函数 (文件前面添加 #include "service.h")



```

102 cout<<"-----添加的卡信息如下-----"<<endl;
103 cout<<setw(12)<<"卡号"<<setw(12)<<"密码"<<setw(12)<<"状态"<<setw(12)<<"开卡金额"<<endl;
104 cout<<setw(12)<< card.aName<<setw(12)<<card.aPwd; //一行输出书写语句过长, 分行
105 cout<<setw(12)<<card.nStatus<<setw(12)<<fixed << setprecision(2)<<card.fBalance<<endl;
106
107 //卡信息添加到文件中
108 // if(FALSE == addCard(card))
109 if(FALSE == addCardInfo(card))
110 {
111     cout<<"-----*****--添加卡信息失败!-----*****"<<endl;
112 }
113 else
114 {
115     cout<<"-----*****--添加卡信息成功!-----*****"<<endl;
116 }
117 }
  
```

2) 在 service.cpp 文件中定义 queryCardInfo 和 queryCardsInfo 函数 (对应头文件添加函数声明) 处理查询卡信息的功能。分别调用 card_service.cpp 文件中 queryCard 和 queryCards 函数, 查询卡信息 (文件前面添加 #include "card_service.h")

```

nu.cpp  service.cpp X
(未知范围)
21 // [函数名] queryCardInfo
22 // [功能] 精确查询卡信息
23 // [参数] pName: 指向用户输入的要查询的卡号
24 // [返回值] 指向卡信息结构体的指针
25 Card* queryCardInfo(const char* pName)
26 {
27     Card* pCard = NULL;
28     pCard = queryCard(pName);
29     return pCard;
30 }
31
32 // [函数名] queryCardsInfo
33 // [功能] 模糊查询卡信息
34 // [参数] pName: 指向用户输入的要查询的卡号;
35 //        pIndex: 指向查到的卡信息数变量
36 // [返回值] 指向卡信息结构体的指针
37 Card* queryCardsInfo(const char* pName, int* pIndex)
38 {
39     Card* pCard = NULL;
40     pCard = queryCards(pName, pIndex);
41     return pCard;
42 }

```

在 menu.cpp 文件中修改 query 函数，在原来调用 queryCard 和 queryCards 函数处，分别改为调用 queryCardInfo 和 queryCardsInfo 函数。

```

menu.cpp X  service.cpp
(未知范围)
162 if(icha==1) //选择精确查询
163 {
164     pCard = queryCardInfo(name);
165 }
166 else //默认其他选择模糊查询
167 {
168     pCard = queryCardsInfo(name, &nIndex);
169 }
170

```

在 menu.cpp 文件中修改 add 函数，在原来调用 queryCard 函数处，改为调用 queryCardInfo 函数。

```

ice.cpp  menu.cpp X  card_service.cpp  billing_service.cpp
(未知范围)
61 // 判断输入的卡号是否存在
62 if( queryCardInfo(name) != NULL)
63 {
64     cout<<"输入的卡号已存在！请重新输入！"<<endl;
65     return;
66 }

```

3) 在 service.cpp 文件中定义 doLogon 函数（对应头文件添加函数声明）实现上机功能。修改 card_service.cpp 文件中原有的 doLogon 函数，函数名改为 checkCard（修改对应头文件中函数声明），函数实现的功能是从文件中将卡信息读取出来，添加到链表，从链表中查找到符合

条件的上机卡。其他的功能移到新的doLogon函数。doLogon函数中调用checkCard函数，找到上机卡后，更新卡信息，调用card_file.cpp中saveBilling函数，添加消费记录。（文件前添加#include "billing_file.h"）

```
card_service.cpp x service.cpp card_file.cpp main.cpp
(未知范围)
295 //[[函数名]checkCard
296 //[[功能] 从文件读取卡信息到链表，在链表中查询卡信息，并获取其在链表中的位置
297 //[[参数] pName:上机卡号；pPwd:上机密码；pIndex:返回卡的索引号
298 //[[返回值]上机卡结构体
299 Card* checkCard(const char* pName, const char* pPwd, int* pIndex)
300 {
301     lpCardNode cardNode = NULL;
302     int nIndex = 0; // 上机卡在卡信息链表中的索引号
303     // 如果从文件中获取卡信息失败，则上机失败
304     if (FALSE == getCard())
305     {
306         return FALSE;
307     }
308     // 指向链表
309     cardNode = cardList;
310     // 遍历链表
311     while (cardNode != NULL)
312     {
313         // 查找上机卡，判断卡号和密码是否正确
314         if ((strcmp(cardNode->data.aName, pName) == 0) && (strcmp(cardNode->data.aPwd, pPwd) == 0))
315         {
316             // 返回卡信息结点数据的地址
317             *pIndex = nIndex;
318             return &cardNode->data;
319         }
320         cardNode = cardNode->next;
321         nIndex++;
322     }
323     return NULL;
324 }
```

```
1.cpp service.cpp x
范围)
47 //[[函数名 doLogon
48 //[[功能] 进行上机操作
49 //[[参数] pName:上机卡号；pPwd:上机密码；pInfo:指向上机信息结构体
50 //[[返回值] TURE:上机成功；FALSE:上机失败
51 int doLogon(const char* pName, const char* pPwd, LogonInfo* pInfo)
52 {
53     Card* pCard = NULL;
54     int nIndex = 0; // 卡信息在链表中的索引，用于更新卡信息
55     Billing billing; // 计费信息
56
57     // 根据卡号和密码，从链表中获取卡信息和卡信息在链表中的索引
58     pCard = checkCard(pName, pPwd, &nIndex);
59 }
```

```

u.cpp  service.cpp x
范围)
60 // 如果卡信息为空，表示没有该卡信息，上机失败
61 if(pCard == NULL)
62 {
63     return FALSE;
64 }
65
66 // 如果卡状态不为0，表示该卡不能上机
67 if(pCard->nStatus != 0)
68 {
69     return UNUSE;
70 }
71
72 // 如果卡的余额为0，不能上机
73 if(pCard->fBalance <= 0)
74 {
75     return ENOUGHMONEY;
76 }
77
78 // 如果可以上机，更新卡信息
79 pCard->nStatus = 1; // 状态为正在使用
80 pCard->nUseCount++; // 使用次数加1
81 pCard->tLastTime = time(NULL); // 更新最后使用时间为当前时间
82
83 // 更新文件中的卡信息
84 if(FALSE == updateCard(pCard, CARDPATH, nIndex))
85 {
86     //文件更新失败返回，更新成功才继续添加消费记录
87     return FALSE;
88 }
89

```

```

u.cpp  service.cpp x
范围)
90 // 添加消费记录
91 strcpy(billing.aCardName, pName); // 上机卡号
92 billing.tStart = time(NULL); // 上机时间
93 billing.tEnd = 0; // 下机时间
94 billing.nStatus = 0; // 消费状态
95 billing.fAmount = 0; // 消费金额
96 billing.nDel = 0; // 删除标识
97
98 // 先将计费信息保存到文件中
99 if(TRUE == saveBilling(&billing, BILLINGPATH))
100 {
101     // 组装上机信息
102     strcpy(pInfo->aCardName, pName);
103     pInfo->fBalance = pCard->fBalance;
104     pInfo->tLogon = billing.tStart;
105     return TRUE;
106 }
107 return FALSE;
108 }

```

在 menu.cpp 文件 logon 函数中调用 service.cpp 文件中的 doLogon 函数(代码不用修改)

4) 在 service.cpp 文件中定义 releaseList 函数（对应头文件添加函数声明）处理退出应用程序时释放链表内存，releaseList 函数调用 card_service.cpp 文件中 releaseCardList 函数

```
service.cpp x
(未知范围)
110 // [函数名] releaseList
111 // [功能] 退出应用程序时，释放链表内存
112 // [参数] void
113 // [返回值] void
114 void releaseList()
115 {
116     releaseCardList(); // 释放卡信息链表内存
117 }
118 }
```

在 menu.cpp 文件中修改 exitApp 函数，调用 service.cpp 文件中 releaseList 函数

```
menu.cpp x service.h service.cpp
(未知范围)
212 // [函数名] exitApp
213 // [功能] 退出应用程序
214 // [参数] void
215 // [返回值] void
216 void exitApp()
217 {
218     releaseList();
219 }
```

4. 编译并运行程序

```
D:\AMS\Debug\AccountManagement.exe

-----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) : 3

-----上机-----
请输入上机的卡号 (长度为1~18) : 123
请输入上机密码 (长度为1~8) : 123
*****卡信息更新到文件成功! *****

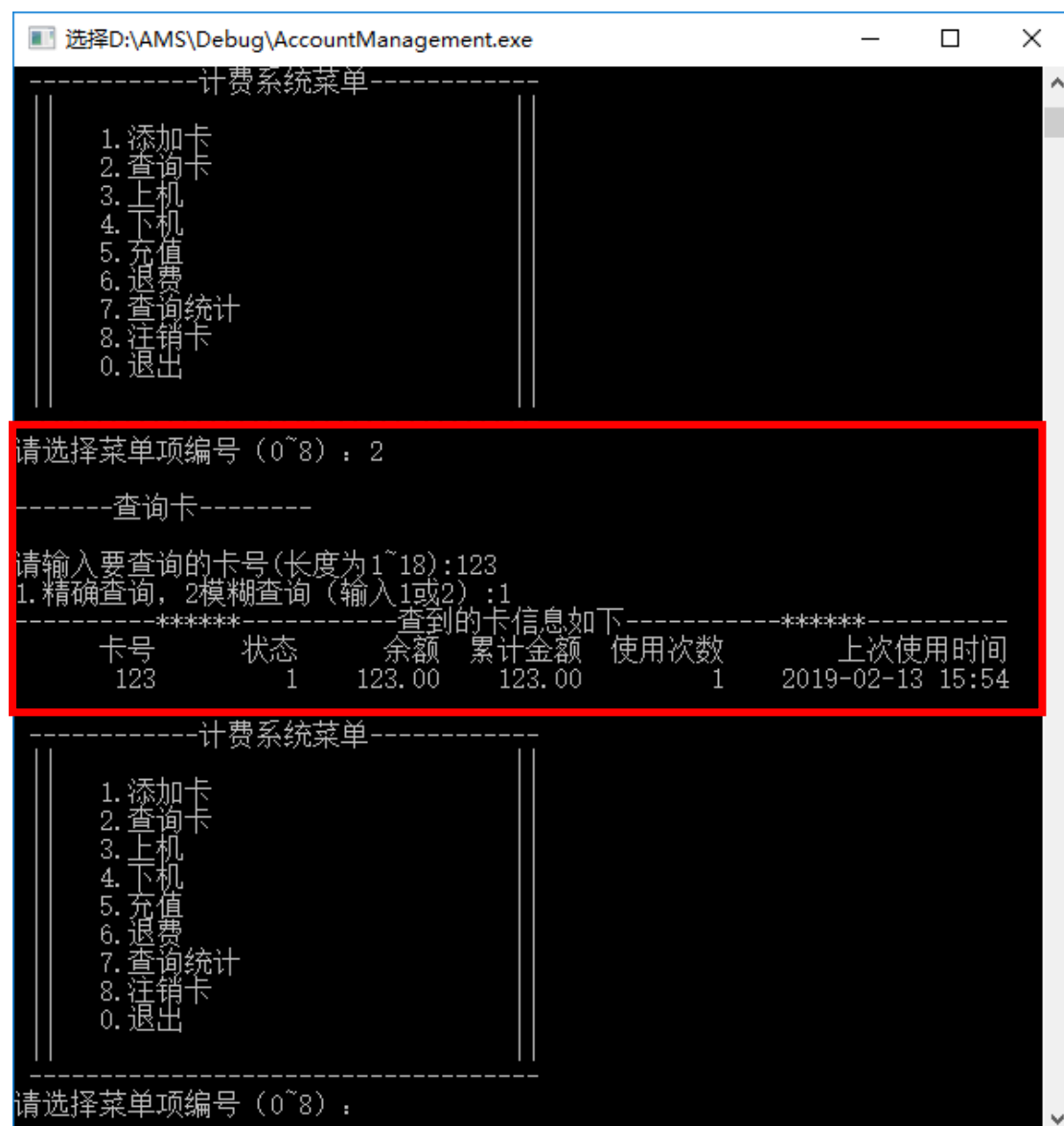
*****计费信息成功存入文件! *****

***-----上机的卡信息如下-----***
卡号      余额      上机时间
123      123.00    2019-02-13 15:54

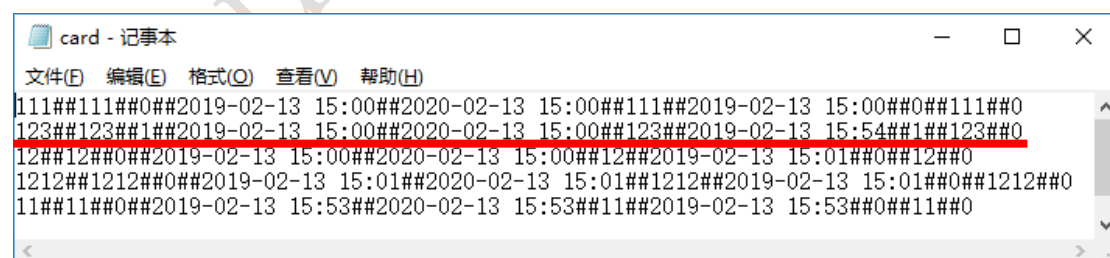
-----上机成功! -----

-----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) :
```



上机卡的状态已经改变



文件中的状态也已更改

(提示：调试程序时多看看卡信息文件中的内容是否更改，是否正确)

本节任务的层次结构和主要调用关系

