

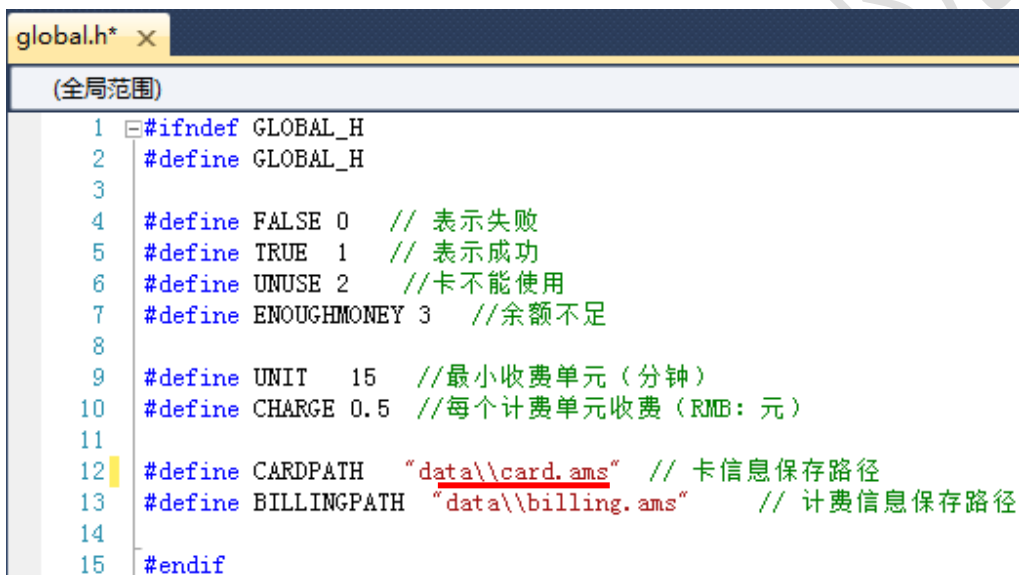
## 一. 用二进制文件保存卡信息

前面的迭代中，卡信息存储在文本文件中，卡信息的各个信息字段用“##”分隔，一条卡信息和下一条卡信息用回车换行符分隔。

由于卡信息中的“累计金额”，“使用次数”，和“余额”等字段的值会随着程序的使用运行不断变化，其在文本文件中保存的长度也会相应的发生变化，即某条卡的信息会变长或变短。当定位某条卡信息进行更新后，如果卡信息变长，长出来的部分会覆盖下一条卡信息的开头部分信息；如果卡信息变短，会在该条信息的后面留下残余的垃圾信息，这都会造成下一次文件无法正确被读取。

本次迭代，将采用二进制形式保存卡信息，二进制文件直接将信息在内存中的二进制数据保存起来，而信息在内存中的二进制数据的长度是由数据类型决定的，与数值大小无关。所以一条卡信息的长度始终是一个卡信息结构体在内存中占用的空间大小，与卡信息结构体中实际存储的数值大小，变化无关。

### 1. 在 global.h 文件中修改卡信息存储路径



```
global.h* x
(全局范围)
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 UNIT 15    // 最小收费单元 (分钟)
10 #define CHARGE 0.5 // 每个计费单元收费 (RMB: 元)
11
12 #define CARDPATH "data\\card.ams" // 卡信息保存路径
13 #define BILLINGPATH "data\\billing.ams" // 计费信息保存路径
14
15 #endif
```

将文件后缀由 txt 改为 ams，这里的后缀表示这是本计费管理系统 (AccountManagementSystem) 程序读写的文件，当然你也可以命名为其他后缀形式。(提示：实际上也可以不修改，只不过文件中写入的是二进制数据，不再是文本，不能使用“记事本”查看内容了，因为“记事本”中显示出来的是乱码)

在 data 目录下手工新建一个文件 card.ams。(提示：也可以不手工建立这个文件，在第一次添加卡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)

### 2. 在 card\_file.cpp 文件中，修改 saveCard 函数 (可以参考 billing\_file.cpp 文件中类似函数的代码)

```

el.h  billing_file.cpp  card_file.cpp x global.h
全局范围)
17 int saveCard(const Card* pCard, const char* pPath)
18 {
19     //以追加方式，二进制方式写入
20     ofstream ofile(pPath, ios_base::out|ios_base::app|ios_base::binary);
21     if(!ofile.is_open())
22     {
23         cout<<"文件无法正确打开！不能写入卡信息！"<<endl;
24         ofile.close();
25         return FALSE;
26     }
27
28     // 将卡信息保存到文件中
29     ofile.write((const char *)pCard, sizeof(Card));
30
31     // 关闭文件
32     ofile.close();
33
34     cout<<endl;
35     cout<<"-----*****-----卡信息成功存入文件!-----*****-----"<<endl<<endl;
36     return TRUE;
37 }

```

3. 在 card\_file.cpp 文件中，修改 readCard 函数

```

el.h  billing_file.cpp  card_file.cpp x global.h
全局范围)
44 int readCard(Card* pCard, const char* pPath)
45 {
46     int nIndex = 0; // 卡信息数量
47
48     //以二进制方式读取
49     ifstream ifile(pPath, ios_base::in|ios_base::binary);
50     if(!ifile.is_open())
51     {
52         cout<<"文件无法正确打开！不能读取卡信息！"<<endl;
53         ifile.close();
54         return FALSE;
55     }
56     else{
57         //遍历文件,读取文件内容
58         while(!ifile.eof())
59         {
60             if(ifile.read((char *)(&pCard[nIndex]), sizeof(Card))!=0)
61             {
62                 nIndex++;
63             }
64         }
65         // 关闭文件
66         ifile.close();
67         return TRUE;
68     }
69 }

```

4. 在 card\_file.cpp 文件中，修改 getCardCount 函数

```

d_service.cpp  model.h  billing_file.cpp  card_file.cpp x  global.h
(全局范围)
109 // [返回值] 卡信息数量
110 int getCardCount(const char* pPath)
111 {
112     int nCount = 0; // 卡信息数量
113     Card* pCard = (Card*)malloc(sizeof(Card));
114     // 以二进制方式读取
115     ifstream ifile(pPath, ios_base::in|ios_base::binary);
116     if(!ifile.is_open())
117     {
118         cout<<"文件无法正确打开！不能统计卡信息数量！"<<endl;
119         ifile.close();
120         return -1; // -1 表示文件没有正确打开
121     }
122     else{
123         // 遍历文件
124         while(!ifile.eof())
125         {
126             if(ifile.read((char *)pCard, sizeof(Card))!=0)
127             {
128                 nCount++;
129             }
130         }
131         // 关闭文件
132         ifile.close();
133         free(pCard);
134         return nCount;
135     }
136 }

```

##### 5. 在 card\_file.cpp 文件中，修改 updateCard 函数

```

d_service.cpp  billing_file.cpp  card_file.cpp x  global.h
(全局范围)
142 int updateCard(const Card* pCard, const char* pPath, int nIndex)
143 {
144     int nCount = 0; // 文件中当前卡序号
145     long lPosition = 0; // 文件标记位置
146     Card pcBuf;
147
148     // 以读写模式, 二进制模式打开文件, 如果失败, 返回FALSE
149     fstream iofile(pPath, ios_base::out|ios_base::in|ios_base::binary);
150     if(!iofile.is_open())
151     {
152         cout<<"文件无法正确打开！不能更新卡信息！"<<endl;
153         iofile.close();
154         return FALSE;
155     }
156
157     // 遍历文件, 获取卡信息在文件中位置
158     while(!iofile.eof() && (nCount < nIndex) )
159     {
160         if(iofile.read((char *)(&pcBuf), sizeof(Card))!=0)
161         {
162             // 获取文件标识位置, 最后一次是找到的位置
163             lPosition = iofile.tellg();
164         }
165         nCount++;
166     }
167     // 移到文件标识位置
168     // 注意指针是在该条消费信息之后还是之前!!!
169     // 上面得到的读写指针在下一条信息开头, 要移到本条信息开头处
170     iofile.seekp(lPosition-sizeof(Card), ios::beg);
171     // 更新消费信息到文件
172     iofile.write((const char *) (pCard), sizeof(Card));
173     cout<<"-----卡信息更新到文件成功! -----"<<endl<<endl;
174     // 关闭文件
175     iofile.close();
176     return TRUE;
177 }

```

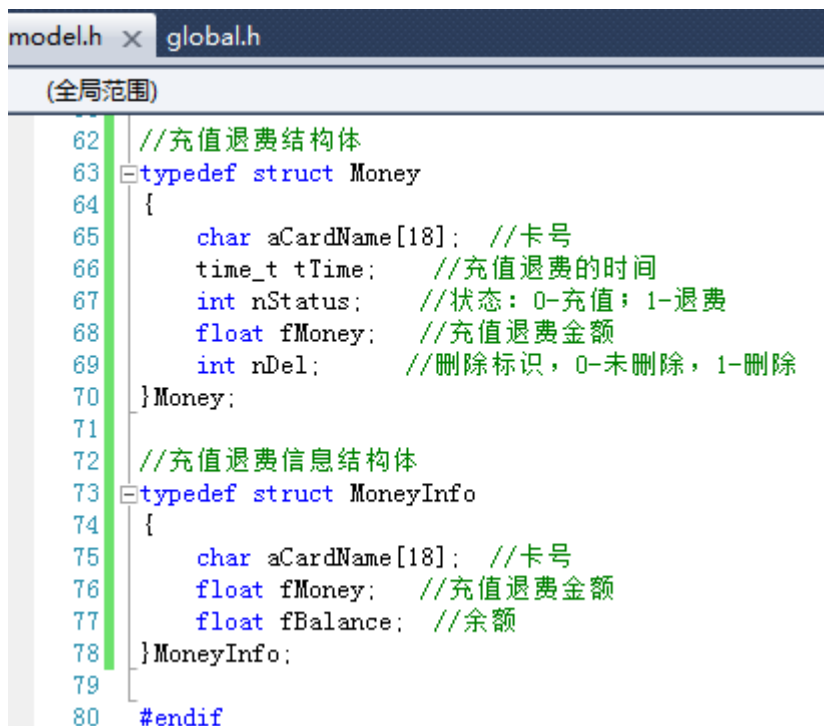
6. 在 card\_file.cpp 文件中,删除 praseCard 函数(二进制文件不需要解析文本内容)
7. 在 card\_file.cpp 文件中,删除宏定义 #define CARDCHARNUM 256

## 二. 充值

当用户选择菜单项“5.充值”时,根据用户输入的卡号,密码对一张未注销,未失效的卡进行充值操作。充值成功后,以列表方式显示卡的充值信息;如果失败,则提示用户充值失败。

### 1. 定义相关的数据结构

在 model.h 文件的#endif 前定义充值退费结构体和充值退费信息结构体



```

model.h x global.h
(全局范围)
62 //充值退费结构体
63 typedef struct Money
64 {
65     char aCardName[18]; //卡号
66     time_t tTime; //充值退费的时间
67     int nStatus; //状态: 0-充值; 1-退费
68     float fMoney; //充值退费金额
69     int nDel; //删除标识, 0-未删除, 1-删除
70 }Money;
71
72 //充值退费信息结构体
73 typedef struct MoneyInfo
74 {
75     char aCardName[18]; //卡号
76     float fMoney; //充值退费金额
77     float fBalance; //余额
78 }MoneyInfo;
79
80 #endif
  
```

### 2. 查找充值卡并更新卡信息

在 menu.cpp 文件中添加 addMoney 函数(对应头文件中加函数声明),提示用户输入要充值的卡号,密码,充值金额,并将金额保存到充值退费信息结构体 MoneyInfo 中。

在 service.cpp 文件中添加 doAddMoney 函数(对应头文件中加函数声明),根据输入的卡号和密码,调用 card\_service.cpp 文件中的 checkCard 函数,在链表中查询卡信息,获取查询到的卡信息在链表中的位置;然后判断查找到的充值卡的卡状态,只有未使用和正在使用的卡才能进行充值操作;符合充值条件后,将 MoneyInfo 中存储的充值金额,累加到相应卡信息结构体的余额和累计金额中;最后调用 card\_file.cpp 文件中的 updateCard 函数,更新文件中的卡信息。

在 menu.cpp 文件的 addMoney 函数中调用 doAddMoney 函数,判断充值是否成功,充值成功就输出充值后卡的相关信息,否则提示充值失败。

程序代码如下:

```

vice.cpp  menu.cpp X
知范围)
363 // [函数名] addMoney
364 // [功能] 充值
365 // [参数] void
366 // [返回值] void
367 void addMoney()
368 {
369     char aName[18] = {0}; // 卡号
370     char aPwd[8] = {0}; // 密码
371     float fAmount = 0; // 充值金额
372     MoneyInfo sMoneyInfo; // 充值信息
373
374     cout<<"请输入要充值的卡号(长度为1~18): ";
375     cin>>aName;
376     cin.clear();
377     cin.sync();
378
379     cout<<"请输入充值卡的密码(长度为1~8): ";
380     cin>>aPwd;
381     cin.clear();
382     cin.sync();
383
384     cout<<"请输入充值金额(RMB): ";
385     cin>>fAmount;
386     cin.clear();
387     cin.sync();
388
389     // 保存充值金额
390     sMoneyInfo.fMoney = fAmount;
391
392 // 判断充值是否成功
393 if(TRUE == doAddMoney(aName, aPwd, &sMoneyInfo))
394 {
395     // 提示充值信息
396     cout<<"-----***----充值信息如下-----***-----"<<endl;
397     // 输出表格的表头
398     cout<<setw(12)<<"卡号"<<setw(12)<<"充值金额"<<setw(12)<<"余额"<<endl;
399     // 输出充值卡信息
400     cout<<setw(12)<<sMoneyInfo.aCardName;
401     cout<<setw(12)<<fixed <<setprecision(2)<<sMoneyInfo.fMoney;
402     cout<<setw(12)<<fixed <<setprecision(2)<<sMoneyInfo.fBalance;
403     cout<<endl<<endl;
404 }
405 else
406 {
407     cout<<"-----充值失败! -----"<<endl;
408 }
409 }

```

```

service.cpp x menu.cpp
(未知范围)
237 // [函数名] doAddMoney
238 // [功能] 进行充值操作
239 // [参数] pName: 充值卡的卡号; pPwd: 充值卡的密码; pMoneyInfo: 充值信息
240 // [返回值] int: 充值的结果, TRUE 充值成功, FALSE 充值失败
241 int doAddMoney(const char* pName, const char* pPwd, MoneyInfo* pMoneyInfo)
242 {
243     Card* pCard = NULL;
244     int nIndex = 0; // 卡信息在链表中的索引号
245
246     // 查询充值卡
247     pCard = checkCard(pName, pPwd, &nIndex);
248
249     // 如果卡信息为空, 表示没有该卡信息, 充值失败
250     if(pCard == NULL)
251     {
252         cout<<"无该卡信息, 不能充值!"<<endl;
253         return FALSE;
254     }
255
256     // 判断该卡是否未使用或正在上机, 只有未使用和正在上机的卡才能进行充值操作
257     if(pCard->nStatus != 0 && pCard->nStatus != 1)
258     {
259         return FALSE;
260     }
261

```

```

service.cpp x menu.cpp
(未知范围)
262 // 如果可以充值, 更新卡信息
263 pCard->fBalance += pMoneyInfo->fMoney;
264 pCard->fTotalUse += pMoneyInfo->fMoney;
265
266 // 更新文件中的卡信息
267 if(FALSE == updateCard(pCard, CARDPATH, nIndex))
268 {
269     return FALSE;
270 }
271 return TRUE;
272 }

```

### 3. 将充值信息保存到文件中

在 global.h 文件中定义充值退费文件的存储路径

```

global.h x service.cpp menu.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 ENOUGH MONEY 3 // 余额不足
8
9 #define UNIT 15 // 最小收费单元 (分钟)
10 #define CHARGE 0.5 // 每个计费单元收费 (RMB: 元)
11
12 #define CARDPATH "data\\card.ams" // 卡信息保存路径
13 #define BILLINGPATH "data\\billing.ams" // 计费信息保存路径
14 #define MONEYPATH "data\\money.ams" // 充值退费信息保存路径
15
16 #endif

```

新建 money\_file.cpp 文件和 money\_file.h，在 cpp 文件中定义 saveMoney 函数（对应头文件中加函数声明），将充值退费信息保存到文件中

```

y_file.cpp  money_file.h x global.h  service.cpp  menu.cpp
范围)
1  #ifndef MONEY_FILE_H
2  #define MONEY_FILE_H
3
4  //函数声明
5  int saveMoney(const Money* pMoney, const char* pPath);
6
7  #endif

```

```

_file.cpp  money_file.cpp x global.h  service.cpp  menu.cpp
范围)
1  #include <iostream>
2  #include <fstream>
3  #include "model.h"
4  #include "global.h"
5
6  using namespace std;
7
8  //函数名 saveMoney
9  //功能 将充值退费信息保存到文件中
10 //参数 pMoney: 充值退费结构体 pPath: 充值退费信息保存路径
11 //返回值 TRUE 保存成功, FALSE 保存失败
12 int saveMoney(const Money* pMoney, 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! 不能写入充值退费信息!"<<endl;
19         ofile.close();
20         return FALSE;
21     }
22     // 将充值退费信息保存到文件中
23     ofile.write((const char *)pMoney, sizeof(Money));
24     // 关闭文件
25     ofile.close();
26     cout<<endl;
27     cout<<"-----*****-----充值退费信息成功存入文件!-----*****-----"<<endl<<endl;
28     return TRUE;
29 }

```

在 service.cpp 文件的 doAddMoney 函数中添加代码，将相关的充值信息，保存到充值退费结构体，组装完充值信息后，调用 money\_file.cpp 文件的 saveMoney 函数（文件前面添加 #include "money\_file.h"），将充值记录保存到文件中；保存成功后，就将相关信息保存到充值退费信息结构体，便于在界面显示相关信息。doAddMoney 函数修改程序如下：

```

file.cpp  service.cpp x  menu.cpp
[范围]
242 int doAddMoney(const char* pName, const char* pPwd, MoneyInfo* pMoneyInfo)
243 {
244     Card* pCard = NULL;
245     int nIndex = 0;          // 卡信息在链表中的索引号
246     Money sMoney;
247
248     // 查询充值卡
249     pCard = checkCard(pName, pPwd, &nIndex);

```

```

file.cpp  service.cpp x  menu.cpp
[范围]
268 // 更新文件中的卡信息
269 if(FALSE == updateCard(pCard, CARDPATH, nIndex))
270 {
271     return FALSE;
272 }
273
274 // 组装充值信息
275 strcpy(sMoney.aCardName, pCard->aName);
276 sMoney.tTime = time(NULL);
277 sMoney.nStatus = 0;
278 sMoney.fMoney = pMoneyInfo->fMoney;
279 sMoney.nDel = 0;
280
281 // 将充值记录保存到文件中
282 if(TRUE == saveMoney(&sMoney, MONEYPATH))
283 {
284     // 组装界面显示的充值信息
285     strcpy(pMoneyInfo->aCardName, sMoney.aCardName);
286     pMoneyInfo->fBalance = pCard->fBalance;
287
288     return TRUE;
289 }
290 return FALSE;
291 }

```

4. 在 main.cpp 的 main 函数中调用 menu.cpp 的 addMoney 函数

```

main.cpp x  card_file.cpp  service.cpp  menu.cpp
(未知范围)
57     case 5:
58     {
59         cout << endl << "-----充值-----" << endl << endl;
60         addMoney();
61         break;
62     }

```

5. 编译连接并运行程序





### 三. 退费

当用户选择菜单项“6.退费”时，根据用户输入的卡号，密码对不再消费的卡进行退费操作，将卡中余额退还给用户。退费成功后，以列表方式显示卡的退费信息；如果失败，则提示用户退费失败。

#### 1. 查找退费卡并更新卡信息

在 menu.cpp 文件中添加 refundMoney 函数（对应头文件中加函数声明），提示用户输入要退费的卡号，密码，并将金额保存到充值退费信息结构体 MoneyInfo 中。

在 service.cpp 文件中添加 doRefundMoney 函数（对应头文件中加函数声明），根据输入的卡号和密码，调用 card\_service.cpp 文件中的 checkCard 函数，在链表中查询卡信息，获取查询到的卡信息在链表中的位置；然后判断查找到的退费卡的卡状态，只有未使用的卡才能进行退费操作，否则返回 UNUSE；读取该卡的余额，只有余额大于 0 的卡才能进行退费操作，否则返回 ENOUGHMONEY；符合退费条件后，将相应卡信息结构体的余额修改为 0，并从累计金额中减去退费的金额；最后调用 card\_file.cpp 文件中的 updateCard 函数，更新文件中的卡信息。

在 menu.cpp 文件的 refundMoney 函数中调用 doRefundMoney 函数，判断退费是否成功，退费成功就输出退费后卡的相关信息，否则根据不同返回值输出相应提示信息。

程序代码如下：

```

n.cpp  menu.cpp x  service.cpp*
[范围]
411 // [函数名] refundMoney
412 // [功能] 退费
413 // [参数] void
414 // [返回值] void
415 void refundMoney()
416 {
417     char aName[18] = {0}; // 卡号
418     char aPwd[8] = {0}; // 密码
419     int nResult = -1; // 退费结果
420     MoneyInfo sMoneyInfo; // 退费信息
421
422     cout<<"请输入退费卡号(长度为1~18):";
423     cin>>aName;
424     cin.clear();
425     cin.sync();
426
427     cout<<"请输入退费密码(长度为1~8):";
428     cin>>aPwd;
429     cin.clear();
430     cin.sync();
431
432     // 进行退费
433     nResult = doRefundMoney(aName, aPwd, &sMoneyInfo);
434
435     // 根据退费结果，提示不同信息
436     switch(nResult)
437     {
438         case 0: // 退费失败
439         {
440             cout<<"-----退费失败! -----"<<endl;
441             break;
442         }
443         case 1: // 退费成功
444         {
445             // 提示退费信息
446             cout<<"-----***-----退费信息如下-----***-----"<<endl;
447             // 输出表格的表头
448             cout<<setw(12)<<"卡号"<<setw(12)<<"退费金额"<<setw(12)<<"余额"<<endl;
449             // 输出退费卡信息
450             cout<<setw(12)<<sMoneyInfo.aCardName;
451             cout<<setw(12)<<fixed <<setprecision(2)<<sMoneyInfo.fMoney;
452             cout<<setw(12)<<fixed <<setprecision(2)<<sMoneyInfo.fBalance;
453             cout<<endl<<endl;
454             break;
455         }
456         case 2: // 正在使用或者已注销
457         {
458             cout<<"-----该卡正在使用或者已注销! -----"<<endl;
459             break;
460         }
461     }

```

```

n.cpp  menu.cpp x  service.cpp
和范围)
461         case 3: // 卡余额不足
462         {
463             cout<<"-----卡余额不足!-----"<<endl;
464             break;
465         }
466         default:
467         {
468             break;
469         }
470     }
471 }

u.cpp  service.cpp* x
[范围)
293 //[[函数名] doRefundMoney
294 //[[功能] 进行退费操作
295 //[[参数] pName 退费卡号; pPwd 退费卡密码; pMoneyInfo 充值退费信息结构体
296 //[[返回值]
297 int doRefundMoney(const char* pName, const char* pPwd, MoneyInfo* pMoneyInfo)
298 {
299     Card* pCard = NULL;
300     int nIndex = 0; // 卡信息在链表中的索引号
301     float fBalance = 0.0; // 卡的余额
302     Money sMoney;
303
304     // 查询退费卡
305     pCard = checkCard(pName, pPwd, &nIndex);
306
307     // 如果为空, 表示没有该卡信息, 返回FALSE
308     if(pCard == NULL)
309     {
310         cout<<"无该卡信息, 不能退费!"<<endl;
311         return FALSE;
312     }
313
314     // 判断该卡是未使用, 只有未使用的卡才能进行退费操作
315     if(pCard->nStatus != 0)
316     {
317         return UNUSE;
318     }
319

```

```

service.cpp* X
320 // 如果余额等于0，则不能退费
321 fBalance = pCard->fBalance;
322 if(fBalance <= 0)
323 {
324     return ENOUGHMONEY;
325 }
326
327 // 更新卡信息
328 pCard->fBalance = 0; // 余额
329 pCard->fTotalUse -= fBalance; // 累计金额
330
331 // 更新文件中的卡信息
332 if(FALSE == updateCard(pCard, CARDPATH, nIndex))
333 {
334     return FALSE;
335 }
336 return TRUE;
337 }

```

## 2. 将退费信息保存到文件中

在service.cpp文件的doRefundMoney函数中添加代码，将相关的退费信息，保存到充值退费结构体，组装完退费信息后，调用money\_file.cpp文件的saveMoney函数，将退费记录保存到文件中；保存成功后，就将相关信息保存到充值退费信息结构体，便于在界面显示相关信息。doRefundMoney函数修改程序如下：

```

menu.cpp service.cpp* X
331 // 更新文件中的卡信息
332 if(FALSE == updateCard(pCard, CARDPATH, nIndex))
333 {
334     return FALSE;
335 }
336
337 // 组合退费信息
338 strcpy(sMoney.aCardName, pCard->aName);
339 sMoney.tTime = time(NULL);
340 sMoney.nStatus = 1;
341 sMoney.fMoney = fBalance;
342 sMoney.nDel = 0;
343
344 // 更新文件中的充值退费信息
345 if(TRUE == saveMoney(&sMoney, MONEYPATH))
346 {
347     // 组装退费信息
348     strcpy(pMoneyInfo->aCardName, sMoney.aCardName);
349     pMoneyInfo->fMoney = sMoney.fMoney;
350     pMoneyInfo->fBalance = pCard->fBalance;
351     return TRUE;
352 }
353 return FALSE;
354 }
355 }

```

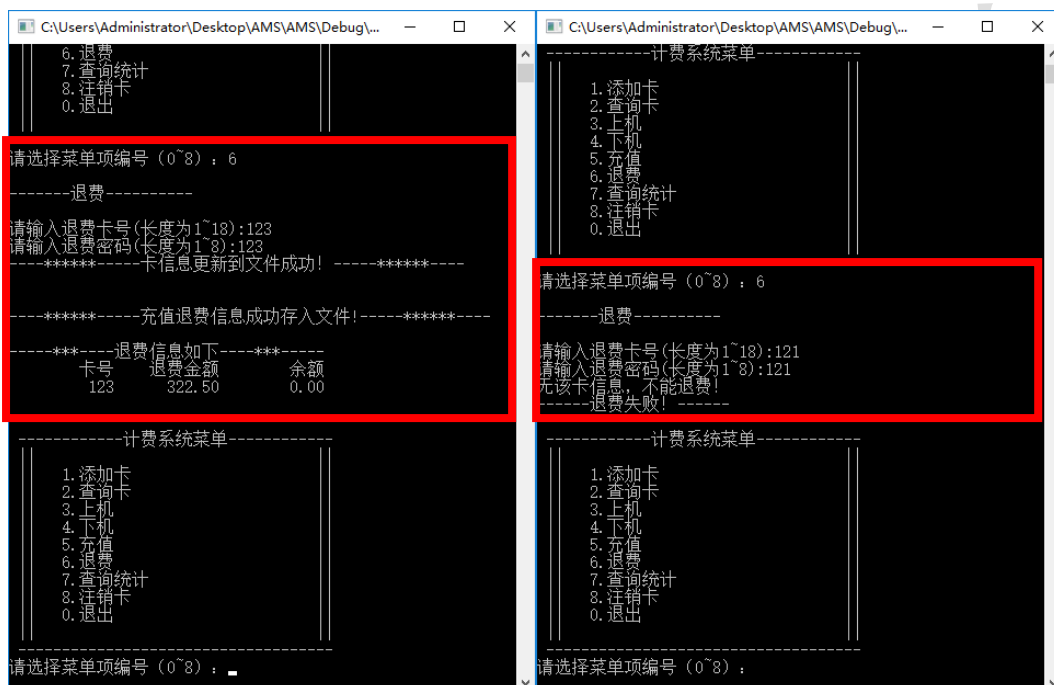
## 3. 在 main.cpp 的 main 函数中调用 menu.cpp 的 refundMoney 函数

```

.h    main.cpp* x  menu.cpp  service.cpp*
范围)
63      case 6:
64      {
65          cout << endl << "-----退费-----" << endl << endl;
66          refundMoney();
67          break;
68      }

```

## 4. 编译连接并运行程序



## 四. 注销卡

注销卡是将不再使用的卡进行注销处理（不是删除，一般历史使用信息应尽可能保留备查），如果卡中还有余额，则退回卡中余额。当用户选择菜单项“8 注销卡”，根据用户输入的注销卡号和密码，遍历卡信息链表，查询符合条件的卡，若找到就列表显示注销卡信息，没找到则提示用户。

## 1. 查找注销卡并更新卡信息

在 menu.cpp 文件中添加 annul 函数（对应头文件中加函数声明），提示用户输入要注销的卡号，密码，保存在结构体中。

在 service.cpp 文件中，添加 annulCard 函数，根据输入的卡号和密码，调用 card\_service.cpp 文件中的 checkCard 函数，找到需要注销的卡；判断该卡是否符合注销条件，如果符合条件，则退还卡中的余额，修改相关信息。更新链表中注销卡的信息：卡状态，卡余额，卡的最后使用时间，删除标记等；调用 card\_file.cpp 文件中的 updateCard 函数，更新文件中的注销卡的信息。

在 menu.cpp 文件的 annul 函数中调用 service.cpp 文件的 annulCard 函数，注销成功，在界面上列表显示注销卡信息，失败则提示用户。

代码如下：

```

n.cpp  menu.cpp x  service.cpp
和范围)
473  // [函数名]  annual
474  // [功能]    注销卡
475  // [参数]    void
476  // [返回值]  void
477  void annul()
478  {
479      Card card;
480
481      cout<<"请输入注销卡号(长度为1~18):";
482      cin>>card.aName;
483      cin.clear();
484      cin.sync();
485
486      cout<<"请输入密码(长度为1~8):";
487      cin>> card.aPwd;
488      cin.clear();
489      cin.sync();
490
491      if(FALSE == annulCard(&card))
492      {
493          cout<<"-----注销卡失败! -----"<<endl;
494          return;
495      }
496
497      else
498      {
499          // 提示注销的信息
500          cout<<"-----注销信息如下-----"<<endl;
501          // 输出表格的表头
502          cout<<setw(12)<<"卡号"<<setw(12)<<"退款金额"<<endl;
503          //输出注销卡信息
504          cout<<setw(12)<<card.aName;
505          cout<<setw(12)<<fixed <<setprecision(1)<<card.fBalance;
506
507          return;
508      }
509  }

```

```

u.cpp  service.cpp x
[范围)
357 // [函数名] annulCard
358 // [功能] 注销卡
359 // [参数] pCard 卡信息结构体
360 // [返回值] TRUE 注销成功; FALSE 注销失败
361 int annulCard(Card* pCard)
362 {
363     Card* pCurCard = NULL;
364     int nIndex = -1; // 卡信息在链表中的索引
365
366     if(pCard == NULL)
367     {
368         return FALSE;
369     }
370     // 根据卡号和密码, 查询卡信息
371     pCurCard = checkCard(pCard->aName, pCard->aPwd, &nIndex);
372
373     if(pCurCard == NULL)
374     {
375         return FALSE;
376     }
377     // 只有未上机的卡才能注销
378     if(pCurCard->nStatus != 0)
379     {
380         return FALSE;
381     }
382     // 保存注销卡的余额
383     pCard->fBalance = pCurCard->fBalance;
384
385     // 更新注销卡信息
386     pCurCard->nStatus = 2; // 状态为已经注销
387     pCurCard->nDel = 1; // 删除标识为已删除
388     pCurCard->fBalance = 0; // 卡余额为0
389     pCurCard->tLastTime = time(NULL); // 最后使用时间为当前时间
390
391     // 更新卡在文件中的信息
392     if(FALSE == updateCard(pCurCard, CARDPATH, nIndex))
393     {
394         return FALSE;
395     }
396     return TRUE;
397 }

```

2. 在 main.cpp 文件的 main 函数中, 调用 annul 函数

```

74         case 8:
75         {
76             cout << endl << "-----注销卡-----" << endl << endl;
77             annul();
78             break;
79         }

```

### 3 编译并运行程序

```

选择C:\Users\Administrator\Desktop\AMS\AMS\Debu...
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) : 8

-----注销卡-----

请输入注销卡号(长度为1~18):111
请输入密码(长度为1~8):111
-----*****-----卡信息更新到文件成功! -----*****-----

---***---注销信息如下---***---
      卡号      退款金额
      111      321.0

-----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) : _

```

### 五. 本次任务的层次结构和主要调用关系



