

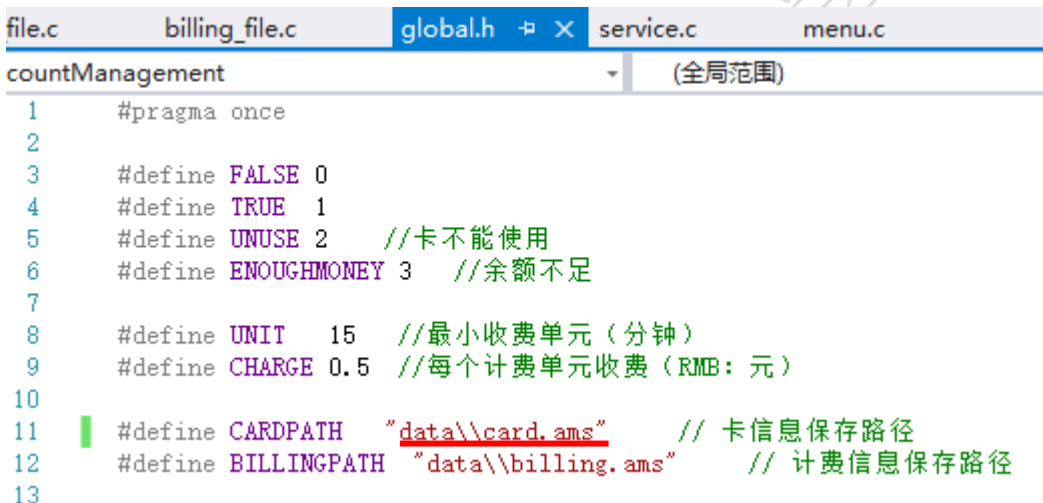
一. 用二进制文件保存卡信息

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字符长度也会相应的发生变化，即某条卡的信息字符长度会变长或变短。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中文件类型



```

1  #pragma once
2
3  #define FALSE 0
4  #define TRUE 1
5  #define UNUSE 2 //卡不能使用
6  #define ENOUGHMONEY 3 //余额不足
7
8  #define UNIT 15 //最小收费单元（分钟）
9  #define CHARGE 0.5 //每个计费单元收费（RMB：元）
10
11 #define CARDPATH "data\\card.ams" // 卡信息保存路径
12 #define BILLINGPATH "data\\billing.ams" // 计费信息保存路径
13

```

在 data 目录下删掉原来的 card.txt 文件，手工新建一个文件 card.ams（先新建一个 card.txt 文件，然后将文件后缀由 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，只不过原来的文本文件中写入的是二进制数据，不再是文本，不能使用“记事本”等文本编辑软件来查看内容了，因为“记事本”等中显示出来的是乱码）

2. 在 card_file.c 文件中，修改 saveCard 函数（可以参考 billing_file.c 文件中类似函数的代码）

g_file.h card_file.c billing_file.c global.h service.c menu.

ccountManagement (全局范围)

```

15  int saveCard(const Card* pCard, const char* pPath)
16  {
17      FILE* fp = NULL;    // 文件结构体指针
18
19      // 以追加方式打开一个二进制文件
20      if ((fp = fopen(pPath, "ab")) == NULL)
21      {
22          // 如果以追加方式失败，则以只写方式创建一个文件并打开
23          if ((fp = fopen(pPath, "wb")) == NULL)
24          {
25              printf("-----添加卡信息打开文件失败! -----\n");
26              return FALSE;
27          }
28      }
29      // 将卡信息保存到文件中
30      fwrite(pCard, sizeof(Card), 1, fp);
31      // 关闭文件
32      fclose(fp);
33      printf("\n-----卡信息成功存入文件!-----\n\n");
34      return TRUE;
35  }

```

3. 在 card_file.c 文件中，修改 readCard 函数

service.c billing_file.h card_file.c billing_file.c global.h

ccountManagement (全局范围)

```

44  int readCard(Card* pCard, const char* pPath)
45  {
46      FILE* fp = NULL;    // 文件结构体指针
47      int nIndex = 0;     // 卡信息数量
48
49      // 以只读的方式打开二进制文件
50      if ((fp = fopen(pPath, "rb")) == NULL)
51      {
52          printf("-----读取卡信息打开文件失败! -----");
53          return FALSE;
54      }
55
56      // 从文件中逐行读取卡信息
57      while (!feof(fp))
58      {
59          if (fread(&pCard[nIndex], sizeof(Card), 1, fp) != 0)
60          {
61              nIndex++;
62          }
63      }
64
65      // 关闭文件
66      fclose(fp);
67
68      return TRUE;
69  }

```

4. 在 card_file.c 文件中, 修改 getCardCount 函数

card.h card_file.c billing_file.c service.c menu.c

accountManagement (全局范围)

```

108 int getCardCount(const char* pPath)
109 {
110     FILE* fp = NULL; // 文件指针
111     int nIndex = 0; // 卡信息数量
112     Card* pCard = (Card*)malloc(sizeof(Card));
113
114     // 以只读模式打开文件
115     if ((fp = fopen(pPath, "rb")) == NULL)
116     {
117         printf("-----获取卡信息数量打开文件失败! -----");
118         return 0;
119     }
120
121     // 逐行读取文件内容, 获取的文件行数就是卡信息数
122     while (!feof(fp))
123     {
124         if (fread(pCard, sizeof(Card), 1, fp) != 0)
125         {
126             nIndex++;
127         }
128     }
129
130     // 关闭文件
131     fclose(fp);
132     free(pCard);
133     return nIndex;
134 }

```

5. 在 card_file.c 文件中, 修改 updateCard 函数

card.h card_file.c billing_file.c service.c menu.c

accountManagement (全局范围)

```

141 int updateCard(const Card* pCard, const char* pPath, int nIndex)
142 {
143     FILE* fp = NULL; // 文件指针
144     int nLine = 0; // 文件卡信息数
145     long lPosition = 0; // 文件位置标记
146     Card bBuf;
147     // 以读写模式打开文件, 如果失败, 返回FALSE
148     if ((fp = fopen(pPath, "rb+")) == NULL)
149     {
150         printf("-----更新卡信息打开文件失败! -----");
151         return FALSE;
152     }
153     // 遍历文件, 获取卡在文件中位置
154     // 注意nIndex是doLogon函数中获得的节点在链表(文件)中序号, 从0开始编号的
155     while (!feof(fp) && (nLine < nIndex))
156     {
157         // 逐行读取文件内容
158         if (fread(&bBuf, sizeof(Card), 1, fp) != 0)
159         {
160             // 获取文件标识位置, 循环的最后一次是找到的位置,
161             // 在要更新的卡信息开始位置
162             lPosition = ftell(fp);
163             nLine++;
164         }
165     }

```

```

166 // 移到文件标识位置
167 fseek(fp, lPosition, 0);
168 //将信息更新到文件
169 fwrite(pCard, sizeof(Card), 1, fp);
170 printf("----***---卡信息更新到文件成功! ----***---\n\n");
171 // 关闭文件
172 fclose(fp);
173 return TRUE;
174 }

```

6. 在 card_file.c 文件中, 删除 praseCard 函数 (二进制文件不需要解析文本内容, 直接读取得到的就是卡信息结构体的数据), 并删除 card_file.h 文件中该函数声明

7. 在 card_file.c 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

2. 查找充值卡并更新卡信息

在 menu.c 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.c 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.c 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.c 文件中的 updateCard 函数，更新文件中的卡信息。

在 menu.c 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：

```

u.h* menu.c
accountManagement (全局范围)
325 // [函数名] addMoney
326 // [功能] 充值
327 // [参数] void
328 // [返回值] void
329 void addMoney()
330 {
331     char aName[18] = { 0 }; // 卡号
332     char aPwd[8] = { 0 }; // 密码
333     float fAmount = 0; // 充值金额
334     MoneyInfo sMoneyInfo; // 充值信息
335
336     printf("请输入充值卡号(长度为1~18):");
337     scanf("%s", aName);
338
339     printf("请输入充值密码(长度为1~8):");
340     scanf("%s", aPwd);
341
342     printf("请输入充值金额(RMB): ");
343     scanf("%f", &fAmount);
344
345     // 保存充值金额
346     sMoneyInfo.fMoney = fAmount;
347
348     // 判断充值是否成功
349     if (TRUE == doAddMoney(aName, aPwd, &sMoneyInfo))
350     {
351         // 提示充值信息
352         printf("-----充值信息如下-----\n");
353         // 输出表头
354         printf("卡号\t充值金额\t余额\n");
355         printf("%s\t%.1f\t%.1f\n", sMoneyInfo.aCardName, sMoneyInfo.fMoney, sMoneyInfo.fBalance);
356     }
357     else
358     {
359         printf("-----充值失败! -----\n");
360     }
361 }

```

```

vice.h  service.c  menu.h  menu.c
AccountManagement (全局范围)
233  //【函数名】 doAddMoney
234  //【功能】 进行充值操作
235  //【参数】  pName:充值卡的卡号
236  //          pPwd : 充值卡的密码
237  //          pMoneyInfo : 充值信息
238  //【返回值】 充值的结果, TRUE 充值成功 FALSE 充值失败
239  int doAddMoney(const char* pName, const char* pPwd, MoneyInfo* pMoneyInfo)
240  {
241      Card* pCard = NULL;
242      int nIndex = 0; // 卡信息在链表中的索引号
243
244      // 查询充值卡
245      pCard = checkCard(pName, pPwd, &nIndex);
246
247      // 如果卡信息为空, 表示没有该卡信息, 充值失败
248      if (pCard == NULL)
249      {
250          printf("----*---无该卡信息, 不能充值! ----*---\n");
251          return FALSE;
252      }
253
254      // 判断该卡是否未使用或正在上机,
255      // 只有未使用和正在上机的卡才能进行充值操作
256      if (pCard->nStatus != 0 && pCard->nStatus != 1)
257      {
258          return FALSE;
259      }
260
261      // 如果可以充值, 更新卡信息
262      pCard->fBalance += pMoneyInfo->fMoney;
263      pCard->fTotalUse += pMoneyInfo->fMoney;
264
265      // 更新文件中的卡信息
266      if (FALSE == updateCard(pCard, CARDPATH, nIndex))
267      {
268          return FALSE;
269      }
270
271      return FALSE;
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  service.h  service.c  menu.h  menu.c
AccountManagement (全局范围)
1  #pragma once
2
3  #define FALSE 0
4  #define TRUE 1
5  #define UNUSE 2 //卡不能使用
6  #define ENOUGHMONEY 3 //余额不足
7
8  #define UNIT 15 //最小收费单元(分钟)
9  #define CHARGE 0.5 //每个计费单元收费(RMB: 元)
10
11 #define CARDPATH "data\\card.ams" // 卡信息保存路径
12 #define BILLINGPATH "data\\billing.ams" // 计费信息保存路径
13 #define MONEYPATH "data\\money.ams" // 充值退费信息保存路径
14

```

新建 money_file.c 文件和 money_file.h, 在 money_file.c 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

```

money_file.c  money_file.h  global.h  service.h  service.c
countManagement (全局范围)
1  #ifndef MONEY_FILE_H
2  #define MONEY_FILE_H
3
4  #include "model.h"
5  int saveMoney(const Money* pMoney, const char* pPath);
6
7  #endif

money_file.c  model.h  money_file.c  money_file.h  global.h  service
countManagement (全局范围)
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  #include "model.h"
5  #include "global.h"
6
7  //[[函数名] saveMoney
8  //[[功能] 将充值退费信息保存到文件中
9  //[[参数] pMoney: 充值退费结构体  pPath: 充值退费信息保存路径
10 //[[返回值] TRUE 保存成功, FALSE 保存失败
11 int saveMoney(const Money* pMoney, const char* pPath)
12 {
13     FILE* fp = NULL; // 文件结构体指针
14
15     // 以追加的模式打开文件, 如果打开失败, 则以只写的模式打开文件
16     if ((fp = fopen(pPath, "ab")) == NULL)
17     {
18         if ((fp = fopen(pPath, "wb")) == NULL)
19         {
20             printf("----***----写入充值退费信息打开文件失败! ----***----\n");
21             return FALSE;
22         }
23     }
24     //将充值退费信息保存到文件
25     fwrite(pMoney, sizeof(Money), 1, fp);
26
27     // 关闭文件
28     fclose(fp);
29     printf("----***---- 充值退费信息成功保存到文件! ----***----\n\n");
30     return TRUE;
31 }
  
```

在 service.c 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.c 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:

```

ice.h  service.c  menu.h  menu.c
ccountManagement  (全局范围)
239 int doAddMoney(const char* pName, const char* pPwd, MoneyInfo* pMoneyInfo)
240 {
241     Card* pCard = NULL;
242     int nIndex = 0; // 卡信息在链表中的索引号
243     Money sMoney;
244     // 查询充值卡
245     pCard = checkCard(pName, pPwd, &nIndex);
246
247     // 更新文件中的卡信息
248     if (FALSE == updateCard(pCard, CARDPATH, nIndex))
249     {
250         return FALSE;
251     }
252
253     // 组装充值信息
254     strcpy(sMoney.aCardName, pCard->aName);
255     sMoney.tTime = time(NULL);
256     sMoney.nStatus = 0;
257     sMoney.fMoney = pMoneyInfo->fMoney;
258     sMoney.nDel = 0;
259
260     // 将充值记录保存到文件中
261     if (TRUE == saveMoney(&sMoney, MONEYPATH))
262     {
263         // 组装界面显示的充值信息
264         strcpy(pMoneyInfo->aCardName, sMoney.aCardName);
265         pMoneyInfo->fBalance = pCard->fBalance;
266
267         return TRUE;
268     }
269     return FALSE;
270 }

```

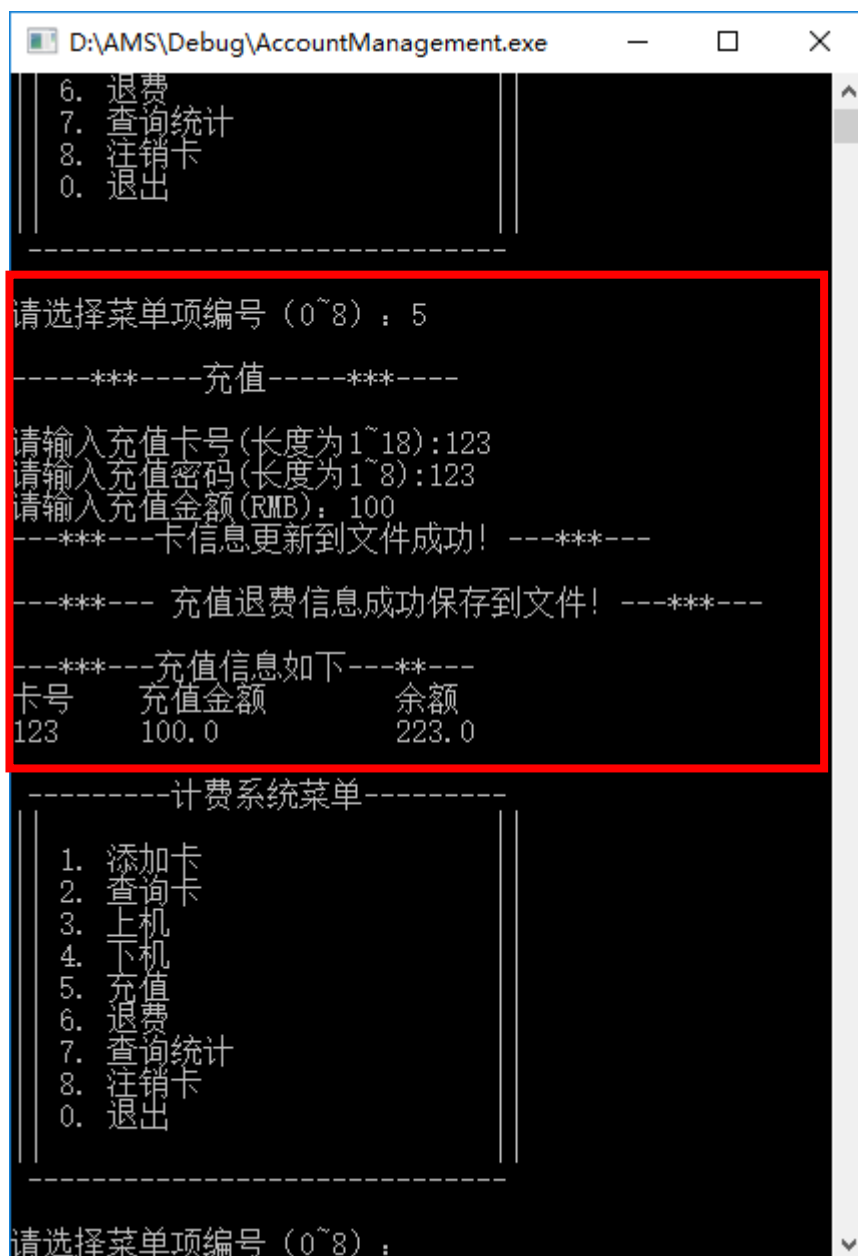
4. 在 main.c 的 main 函数中 case 5 下调用 menu.c 的 addMoney 函数

```

45 case 5:
46     printf("\n-----***-----充值-----***-----\n\n");
47     addMoney();
48     break;

```

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.c 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.c 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.c 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.c 文件中的 updateCard 函数，更新文件中的卡信息。

在 menu.c 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：

```

362
363 //[[函数名] refundMoney
364 //[[功能] 退费
365 //[[参数] void
366 //[[返回值] void
367 void refundMoney()
368 {
369     char aName[18] = { 0 }; // 卡号
370     char aPwd[8] = { 0 }; // 密码
371     int nResult = -1; // 退费结果
372     MoneyInfo sMoneyInfo; // 退费信息
373
374     printf("请输入退费卡号(长度为1~18):");
375     scanf("%s", aName);
376
377     printf("请输入退费密码(长度为1~8):");
378     scanf("%s", aPwd);
379
380     // 进行退费
381     nResult = doRefundMoney(aName, aPwd, &sMoneyInfo);
382
383 // 根据退费结果，提示不同信息
384 switch (nResult)
385 {
386     case 0: // 退费失败
387         printf("-----退费失败! -----\n");
388         break;
389     case 1: // 退费成功
390         printf("-----退费信息如下-----\n");
391         //输出表头
392         printf("卡号\t退费金额\t余额\n");
393         //输出退费信息
394         printf("%s\t%.1f\t%.1f\n", sMoneyInfo.aCardName, sMoneyInfo.fMoney, sMoneyInfo.fBalance);
395         break;
396     case 2: // 正在使用或者已注销
397         printf("-----该卡正在使用或者已注销! -----\n");
398         break;
399     case 3: // 卡余额不足
400         printf("-----卡余额不足! -----\n");
401         break;
402     default:
403         break;
404 }
405

```

```

ice.h  service.c  menu.h  menu.c
ccountManagement  (全局范围)

292  //【函数名】 doRefundMoney
293  //【功能】 进行退费操作
294  //【参数】  pName 退费卡号;  pPwd 退费卡密码;  pMoneyInfo 充值退费信息结构体
295  //【返回值】 TRUE 退费成功; FALSE 退费失败
296  int doRefundMoney(const char* pName, const char* pPwd, MoneyInfo* pMoneyInfo)
297  {
298      Card* pCard = NULL;
299      int nIndex = 0; // 卡信息在链表中的索引号
300      float fBalance = 0.0; // 卡的余额
301      Money sMoney;
302
303      // 查询退费卡
304      pCard = checkCard(pName, pPwd, &nIndex);
305
306      // 如果为空, 表示没有该卡信息, 返回FALSE
307      if (pCard == NULL)
308      {
309          printf("----***---无该卡信息, 不能退费! ----***---\n");
310          return FALSE;
311      }
312
313      // 判断该卡是未使用, 只有未使用的卡才能进行退费操作
314      if (pCard->nStatus != 0)
315      {
316          return UNUSE;
317      }
318
319      // 如果余额等于0, 则不能退费
320      fBalance = pCard->fBalance;
321      if (fBalance <= 0)
322      {
323          return ENOUGHMONEY;
324      }
325
326      // 更新卡信息
327      pCard->fBalance = 0; // 余额
328      pCard->fTotalUse -= fBalance; // 累计金额
329
330      // 更新文件中的卡信息
331      if (FALSE == updateCard(pCard, CARDPATH, nIndex))
332      {
333          return FALSE;
334      }
335      return TRUE;
336  }

```

2. 将退费信息保存到文件中

在service.c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.c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:

```

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

```

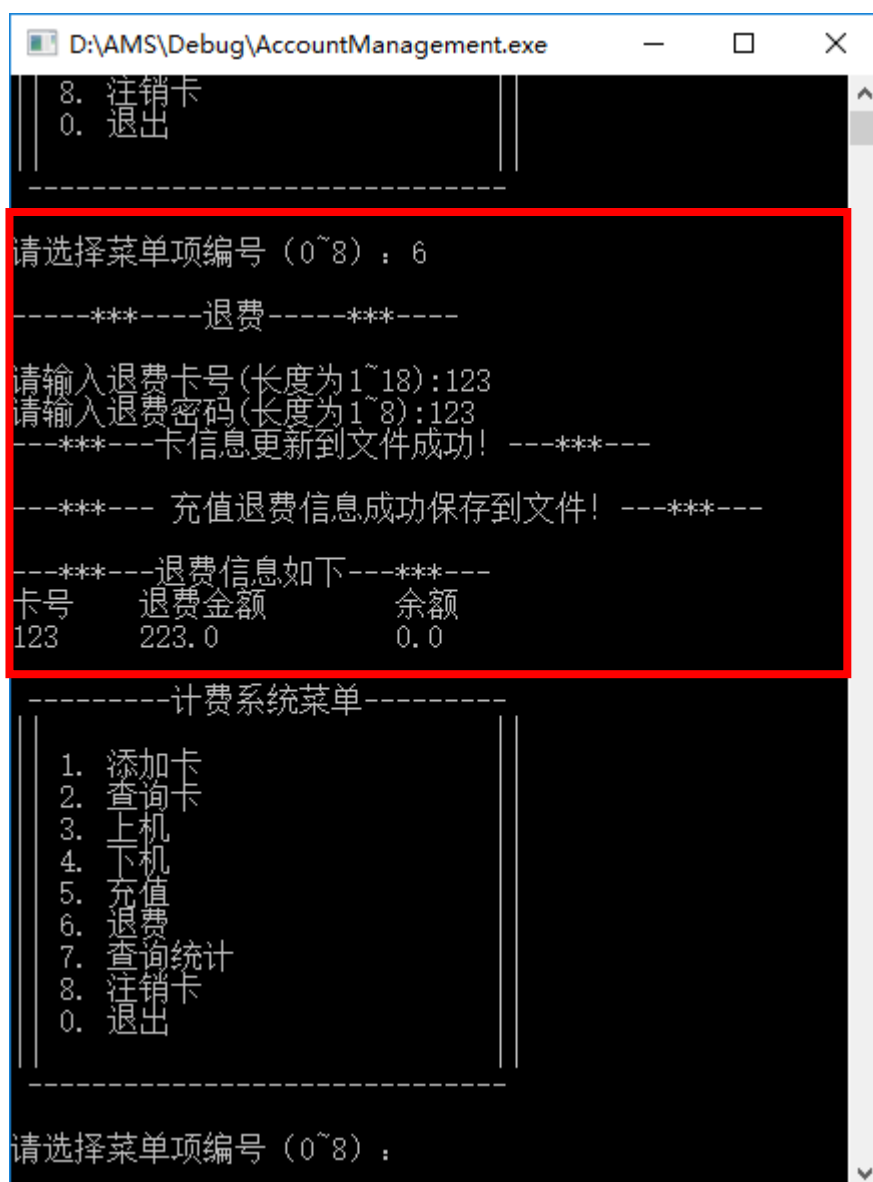
3. 在 main.c 的 main 函数的 case 6 中调用 menu.c 的 refundMoney 函数

```

49 case 6:
50     printf("\n-----***-----退费-----***-----\n\n");
51     refundMoney();
52     break;

```

4. 编译连接并运行程序



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

8. 注销卡
0. 退出

-----
请选择菜单项编号 (0~8) : 6

-----***-----退费-----***-----

请输入退费卡号(长度为1~18):123
请输入退费密码(长度为1~8):123
-----***-----卡信息更新到文件成功! -----***-----

-----***----- 充值退费信息成功保存到文件! -----***-----

-----***-----退费信息如下-----***-----
卡号      退费金额      余额
123       223.0         0.0

-----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) :
```

四. 注销卡

注销卡是将不再使用的卡进行注销处理（不是删除只是做标记，一般历史使用过的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

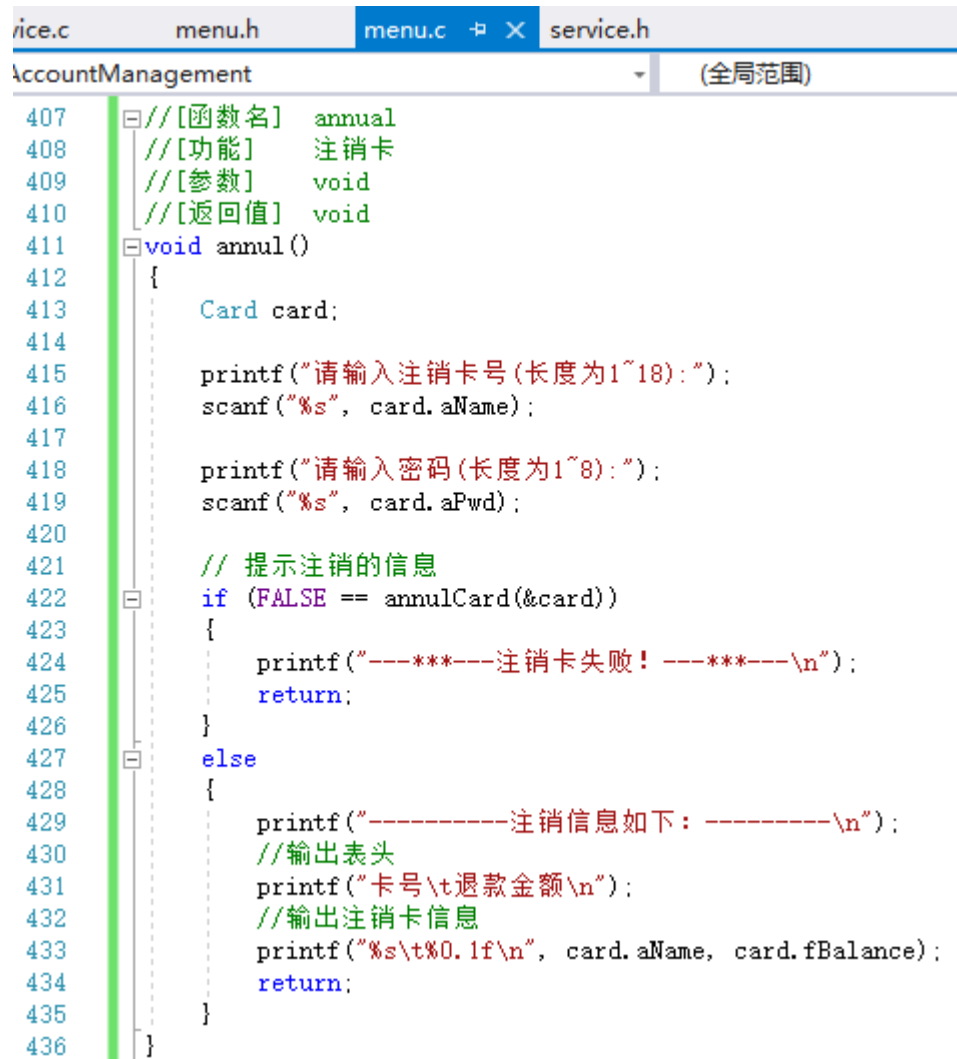
在 menu.c 文件中添加 annul 函数（对应头文件中加函数声明），提示用户输入要注销的卡号，密码，保存在结构体中。

在 service.c 文件中，添加 annulCard 函数，根据输入的卡号和密码，调用 card_service.c 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.c 文件中的 updateCard 函数，更新文件中的注销卡的

信息。

在 menu.c 文件的 annul 函数中调用 service.c 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：



```

407  // [函数名]  annual
408  // [功能]    注销卡
409  // [参数]    void
410  // [返回值]  void
411  void annul()
412  {
413      Card card;
414
415      printf("请输入注销卡号(长度为1~18):");
416      scanf("%s", card.aName);
417
418      printf("请输入密码(长度为1~8):");
419      scanf("%s", card.aPwd);
420
421      // 提示注销的信息
422      if (FALSE == annulCard(&card))
423      {
424          printf("-----注销卡失败! -----\n");
425          return;
426      }
427      else
428      {
429          printf("-----注销信息如下: -----\n");
430          // 输出表头
431          printf("卡号\t退款金额\n");
432          // 输出注销卡信息
433          printf("%s\t%.1f\n", card.aName, card.fBalance);
434          return;
435      }
436  }

```

```

u.h  service.c*  menu.c  service.h
ccountManagement  (全局范围)
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
371      // 根据卡号和密码, 查询卡信息
372      pCurCard = checkCard(pCard->aName, pCard->aPwd, &nIndex);
373
374      if (pCurCard == NULL)
375      {
376          return FALSE;
377      }
378
379      // 只有未上机的卡才能注销
380      if (pCurCard->nStatus != 0)
381      {
382          return FALSE;
383      }
384
385      // 保存注销卡的余额
386      pCard->fBalance = pCurCard->fBalance;
387
388      // 更新注销卡信息
389      pCurCard->nStatus = 2;  // 状态为已经注销
390      pCurCard->nDel = 1;    // 删除标识为已删除
391      pCurCard->fBalance = 0;  // 卡余额为0
392      pCurCard->tLastTime = time(NULL); // 最后使用时间为当前时间
393
394      // 更新卡在文件中的信息
395      if (FALSE == updateCard(pCurCard, CARDPATH, nIndex))
396      {
397          return FALSE;
398      }
399      return TRUE;
400  }

```

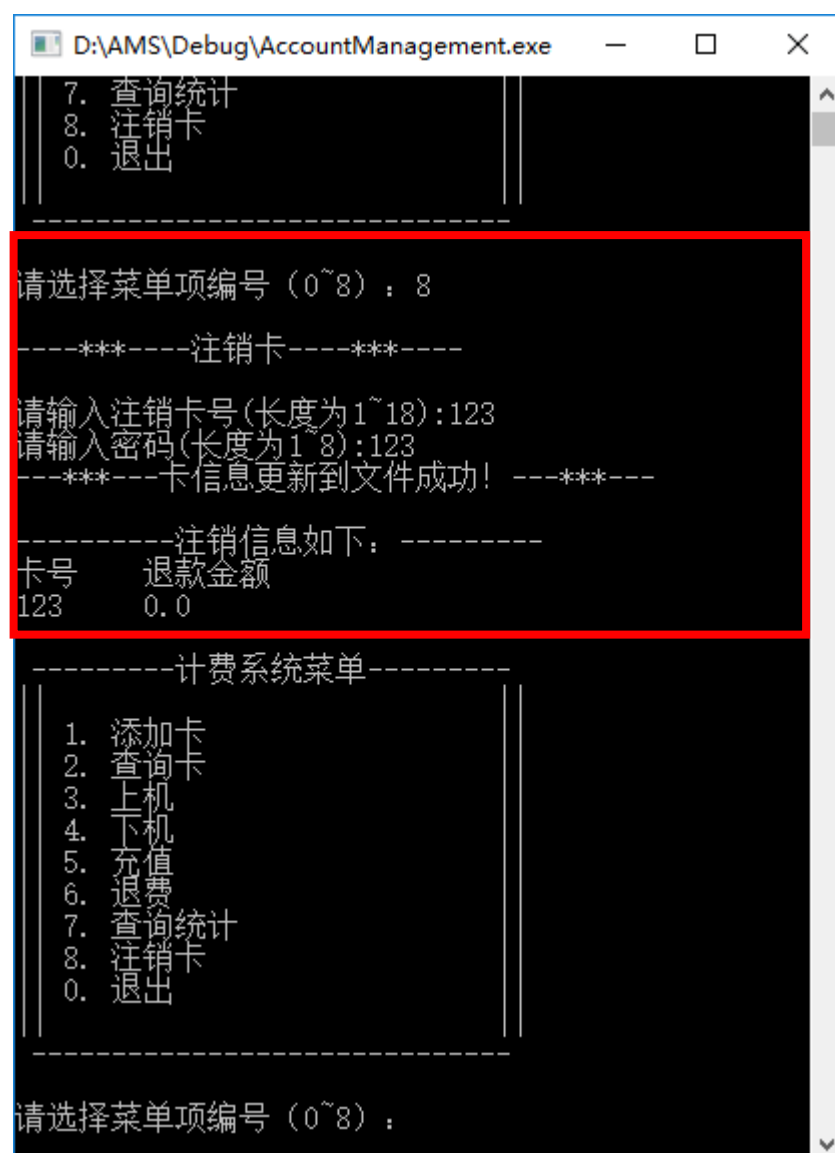
2. 在 main.c 文件的 main 函数 case 8 中, 调用 annul 函数

```

56      case 8:
57          printf("\n-----***-----注销卡-----***-----\n\n");
58          annul();
59          break;

```

3 编译并运行程序



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

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):123
请输入密码(长度为1~8):123
-----***-----卡信息更新到文件成功! -----***-----

-----注销信息如下: -----
卡号      退款金额
123       0.0

-----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

