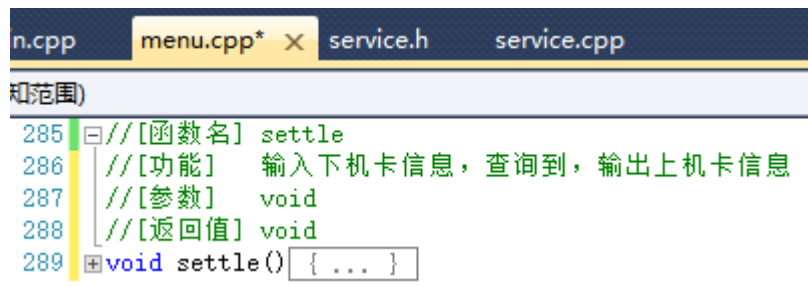


## 一. 计算消费金额

正在使用的上机卡，根据用户输入的卡号，密码，进行下机操作。选择“4.下机”时，根据上机时间，下机时间，计费标准，计算消费金额。

## 1. 查找下机卡的消费信息

在 menu.cpp 文件中定义 settle 函数（对应头文件中添加函数声明），提示用户输入下机卡的卡号和密码，从文件中获取该卡的消费信息。（函数的语句后面再添加）

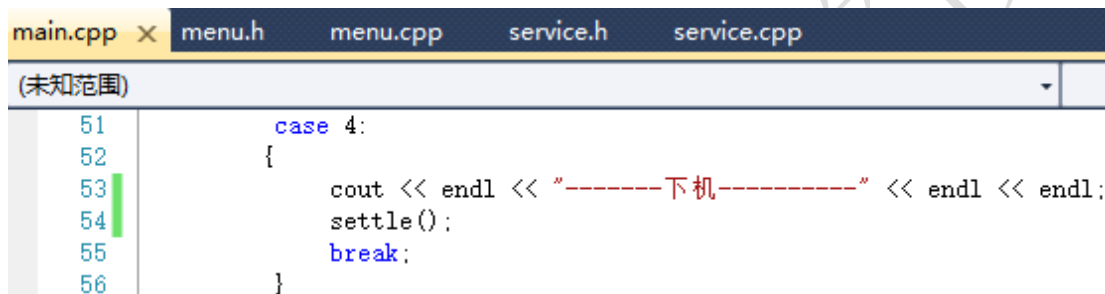


```

285 // [函数名] settle
286 // [功能] 输入下机卡信息，查询到，输出上机卡信息
287 // [参数] void
288 // [返回值] void
289 void settle() { ... }

```

在 main.cpp 文件的 main 函数中调用 settle 函数

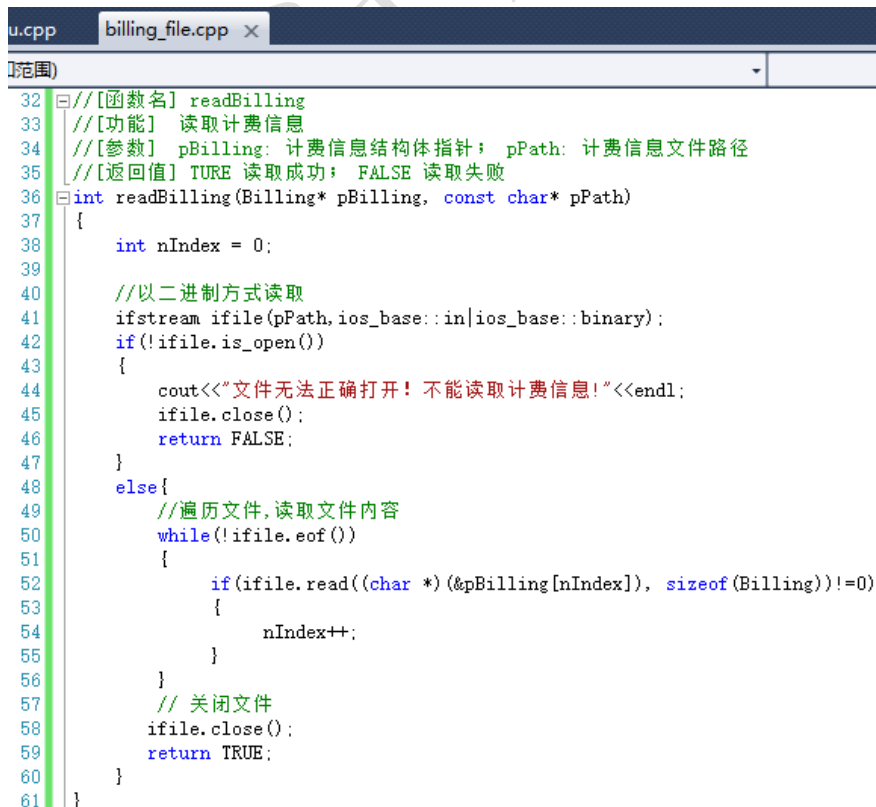


```

51 case 4:
52 {
53     cout << endl << "-----下机-----" << endl << endl;
54     settle();
55     break;
56 }

```

在 billing\_file.cpp 文件中定义 readBilling 函数（对应头文件添加函数声明），从计费文件读取计费信息。（实现方法和 card\_file.cpp 文件中的 readCard 函数类似，区别是这里读取的是二进制文件，不需要解析文件内容），代码如下：



```

32 // [函数名] readBilling
33 // [功能] 读取计费信息
34 // [参数] pBilling: 计费信息结构体指针; pPath: 计费信息文件路径
35 // [返回值] TURE 读取成功; FALSE 读取失败
36 int readBilling(Billing* pBilling, const char* pPath)
37 {
38     int nIndex = 0;
39
40     // 以二进制方式读取
41     ifstream ifile(pPath, ios_base::in | ios_base::binary);
42     if (!ifile.is_open())
43     {
44         cout << "文件无法正确打开！不能读取计费信息！" << endl;
45         ifile.close();
46         return FALSE;
47     }
48     else {
49         // 遍历文件, 读取文件内容
50         while (!ifile.eof())
51         {
52             if (ifile.read((char *)(&pBilling[nIndex]), sizeof(Billing)) != 0)
53             {
54                 nIndex++;
55             }
56         }
57         // 关闭文件
58         ifile.close();
59         return TRUE;
60     }
61 }

```

在 `billing_file.cpp` 文件中定义 `getBillingCount` 函数（对应头文件添加函数声明），获取计费文件中消费信息的总数量，（实现方法和 `card_file.cpp` 文件中的 `getCardCount` 函数类似，区别是这里读取的是二进制文件），代码如下：

```

file.cpp  menu.cpp  billing_file.cpp* x
[范围]
63  //函数名 getBillingCount
64  //功能  获取文件中计费信息数量
65  //参数  计费信息文件路径
66  //返回值  计费信息数量
67  int getBillingCount(const char* pPath)
68  {
69      int nCount = 0;
70      Billing* pBilling = (Billing*)malloc(sizeof(Billing));
71      //以二进制方式读取
72      ifstream ifile(pPath, ios_base::in|ios_base::binary);
73      if(!ifile.is_open())
74      {
75          cout<<"文件无法正确打开！不能统计计费信息数量！"<<endl;
76          ifile.close();
77          return FALSE;
78      }
79      else{
80          //遍历文件
81          while(!ifile.eof())
82          {
83              if(ifile.read((char *)pBilling, sizeof(Billing))!=0)
84              {
85                  nCount++;
86              }
87          }
88          // 关闭文件
89          ifile.close();
90          free(pBilling);
91          return nCount;
92      }
93  }

```

在 `model.h` 文件中，`#endif` 之前定义计费信息链表结点

```

model.h x  card_file.cpp  menu.cpp  b
(未知范围)
45  //计费信息结点
46  typedef struct BillingNode
47  {
48      Billing data;
49      struct BillingNode *next;
50  }BillingNode, *lpBillingNode;

```

在 `billing_service.cpp` 文件中，定义 `initBillingList` 函数和 `releaseBillingList` 函数（对应头文件添加函数声明），用来初始化和释放 `billingList` 链表，（实现方法和 `card_service.cpp` 文件中的 `initCardList` 函数和 `releaseCardList` 函数类似），先定义一个全局变量，再实现 2 个函数：



The image displays three sequential screenshots of a C++ code editor, showing the implementation of a linked list for billing information in the `billing_service.cpp` file. The editor tabs at the top include `billing_service.cpp`, `model.h`, `card_file.cpp`, `menu.cpp`, and `billing_file.c`.

**First Screenshot:** Shows the initial setup of the `lpBillingNode` pointer.

```

1 #include <stdlib.h> // 包含动态内存分配头文件
2 #include <string.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 lpBillingNode billingList = NULL; // 计费信息链表
9

```

**Second Screenshot:** Shows the `initBillingList` function implementation.

```

32 // [函数名]: initBillingList
33 // [函数功能]: 初始化计费信息链表
34 // [函数参数]: void
35 // [返回值]: void
36 void initBillingList()
37 {
38     lpBillingNode head = NULL;
39     if (billingList == NULL)
40     {
41         head = (lpBillingNode)malloc(sizeof(BillingNode));
42         head->next = NULL;
43         billingList = head;
44     }
45 }

```

**Third Screenshot:** Shows the `releaseBillingList` function implementation.

```

47 // [函数名]: releaseBillingList
48 // [函数功能]: 释放计费信息链表
49 // [函数参数]: void
50 // [返回值]: void
51 void releaseBillingList()
52 {
53     lpBillingNode cur = billingList;
54     lpBillingNode next = NULL;
55     // 销毁链表
56     while (cur != NULL)
57     {
58         next = cur->next; // 结点内存释放前, next 保存其后继
59         free(cur); // 释放结点内存
60         cur = next;
61     }
62     billingList = NULL;
63 }

```

在 `billing_service.cpp` 文件中, 定义 `getBilling` 函数 (对应头文件添加函数声明), 从消费信息文件中获取消费信息, 保存到链表中 (实现方法和 `card_service.cpp` 文件中的 `getCard` 函数类似)

```

billing_service.cpp x model.h card_file.cpp menu.cpp billing_file.cpp
(未知范围)
65 // [函数名]: getBilling
66 // [函数功能]: 从计费信息文件中, 获取计费信息保存到链表中
67 // [函数参数]: void
68 // [返回值]: TRUE 获取信息成功, FALSE 获取信息失败
69 int getBilling()
70 {
71     int nCount = 0; // 计费信息数量
72     Billing* pBilling = NULL; // 计费信息
73     lpBillingNode node = NULL;
74     int i = 0; // 循环变量
75     lpBillingNode cur = NULL;
76
77     // 如果链表不为空, 释放链表
78     if(billingList != NULL)
79     {
80         releaseBillingList();
81     }
82     // 初始化链表
83     initBillingList();
84
85     // 获取计费信息数量
86     nCount = getBillingCount(BILLINGPATH);
87     // 动态分配内存
88     pBilling = (Billing*)malloc(sizeof(Billing) * nCount);

```

```

billing_service.cpp x model.h card_file.cpp menu.cpp billing_file.cpp
(未知范围)
89 if(pBilling != NULL)
90 {
91     // 获取计费信息
92     if(FALSE == readBilling(pBilling, BILLINGPATH))
93     {
94         free(pBilling);
95         return FALSE;
96     }
97     // 将计费信息保存到链表中
98     for(i = 0, node = billingList; i < nCount; i++)
99     {
100         // 为当前结点分配内存
101         cur = (lpBillingNode)malloc(sizeof(BillingNode));
102         // 如果分配失败, 则在返回FALSE前, 需要释放pBilling的内存
103         if(cur == NULL)
104         {
105             free(pBilling);
106             return FALSE;
107         }
108         // 初始化新的空间, 全部赋值为0
109         memset(cur, 0, sizeof(BillingNode));
110         // 将数据添加到结点中
111         cur->data = pBilling[i];
112         cur->next = NULL;
113         // 将结点添加到链表中
114         node->next = cur;
115         node = cur;
116     }

```

```

billing_service.cpp x model.h card_file.cpp menu.cpp billing_file.c
(未知范围)
117     free(pBilling);
118     return TRUE;
119 }
120 return FALSE;
121 }

```

在 billing\_service.cpp 文件中，定义 queryBilling 函数（对应头文件添加函数声明），在消费信息链表中，根据输入的卡号，查找对应卡的消费信息，以及在链表中的索引号（实现方法和 card\_file.cpp 文件中的 checkCard 函数类似）

```

billing_service.cpp x card_file.cpp billing_file.cpp service.cpp menu.cpp
(未知范围)
123 //[[函数名]: queryBilling
124 //[[函数功能]: 在计费信息链表中，查询对应卡的计费信息，并获取其在链表中的索引号
125 //[[函数参数]: pName: 要查询的计费信息的卡号
126 //             pIndex: 查询到的计费信息在链表中的索引号
127 //[[返回值]: 查询到的计费信息指针
128 Billing* queryBilling(const char* pName, int* pIndex)
129 {
130     lpBillingNode node = NULL;
131     int nIndex = 0;
132
133     if(FALSE == getBilling())
134     {
135         return NULL;
136     }
137     // 遍历链表
138     node = billingList;
139     while(node != NULL)
140     {
141         // 查询到卡号相同，并且没有结算的计费信息
142         if(strcmp(node->data.aCardName, pName) == 0 && node->data.nStatus == 0)
143         {
144             return &node->data;
145         }
146         node = node->next;
147         nIndex++;
148         *pIndex = nIndex;
149     }
150     return NULL;
151 }

```

在 model.h 文件中定义下机信息结构体，保存下机卡的下机信息，方便下机信息的读取和修改

```

model.h x service.cpp* billing_service.h
(未知范围)
52 //下机信息结构体
53 typedef struct SettleInfo
54 {
55     char aCardName[18]; //卡号
56     time_t tStart; //上机时间
57     time_t tEnd; //下机时间
58     float fAmount; //消费金额
59     float fBalance; //余额
60 }SettleInfo;
61
62 #endif

```

在 service.cpp 文件中，定义 doSettle 函数（对应头文件添加函数声明），查找下机卡的卡

信息和消费信息，没有查到，则下机失败；下机卡的状态是“正在上机”才能进行下机操作才能下机。其中调用queryBilling函数（文件前面#include "billing\_service.h"），判断是否查询到下机卡的计费信息

```

nu.cpp  service.cpp x  billing_service.h
[范围]
124 // [函数名] doSettle
125 // [功能] 进行下机操作
126 // [参数] pName: 下机卡号; pPwd: 下机密码; pInfo: 指向下机信息结构体
127 // [返回值] 下机信息值, 不同情况输出不同信息
128 int doSettle(const char* pName, const char* pPwd, SettleInfo* pInfo)
129 {
130     Card* pCard = NULL;
131     Billing* pBilling = NULL;
132     int nIndex = 0; // 卡信息在链表中的索引号
133     int nPosition = 0; // 计费信息在链表中的索引号
134
135     // 查询上机卡
136     pCard = checkCard(pName, pPwd, &nIndex);
137
138     // 如果为空, 表示没有该卡信息, 返回FALSE
139     if(pCard == NULL)
140     {
141         cout<<"无该卡信息, 请重新输入!"<<endl;
142         return FALSE;
143     }
144
145     // 判断该卡是否正在上机, 只有正在上机卡才能进行下机操作
146     if(pCard->nStatus != 1)
147     {
148         cout<<"该卡未上机!"<<endl;
149         return UNUSE;
150     }
151
152     // 根据卡号, 查询计费信息
153     pBilling = queryBilling(pName, &nPosition);
154
155     // 如果查询计费信息为空, 表示下机失败
156     if(pBilling == NULL)
157     {
158         cout<<"无该卡信息, 请重新输入!"<<endl;
159         return FALSE;
160     }
161     return TRUE;
162 }

```

## 2. 计算本次消费金额

在 global.h 文件定义两个计费常量，最小收费单元，每 15 分钟计费一次，最后一次未满足 15 分钟也算一次；每个计费单元的收费是 0.5 元；计费时长是下机时间减去上机时间。

```

global.h x menu.cpp service.cpp billing_service.h
(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 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.txt" // 卡信息保存路径
13 #define BILLINGPATH "data\\billing.ams" // 计费信息保存路径
14
15 #endif

```

在 service.cpp 文件中，定义 getAmount 函数（对应头文件添加函数声明），计算消费金额

```

bal.h menu.cpp service.cpp x billing_service.h
(未知范围)
187 // [函数名] getAmount
188 // [功能] 根据上机时间，计算消费金额
189 // [参数] 上机时间
190 // [返回值] 消费金额
191 double getAmount(time_t tStart)
192 {
193     double dbAmount = 0; // 消费金额
194     int nCount = 0; // 上机的时间单元数，每个单元15分钟
195     int nSec = 0; // 消费时间(单位：秒)
196     int nMinutes = 0; // 消费时间(单位：分钟)
197     time_t tEnd = time(NULL); // 结算时间为当前时间
198
199     // 计算消费时长
200     nSec = (int)(tEnd - tStart);
201     nMinutes = nSec / 60;
202     // 计算消费的时间单元数
203     if(nMinutes % UNIT == 0)
204     {
205         nCount = nMinutes/UNIT; // 整除
206     }
207     else
208     {
209         nCount = nMinutes/UNIT + 1; // 不整除
210     }
211     // 计算消费金额
212     dbAmount = nCount * CHARGE;
213     return dbAmount;
214 }

```

### 3. 保存下机信息

修改 service.cpp 文件中 doSettle 函数，调用 getAmount 函数，保存消费金额，计算下机卡的余额，下机卡的消费后余额必须不小于零才能下机，组装下机信息，将下机信息添加到下机信息结构体



```

ice.h  global.h  menu.cpp  service.cpp x  billing_service.h
[范围)
129 int doSettle(const char* pName, const char* pPwd, SettleInfo* pInfo)
130 {
131     Card* pCard = NULL;
132     Billing* pBilling = NULL;
133     int nIndex = 0; // 卡信息在链表中的索引号
134     int nPosition = 0; // 计费信息在链表中的索引号
135     double dbAmount = 0.0; // 消费金额
136     float fBalance = 0.0; // 余额
137
138     // 查询上机卡
139     pCard = checkCard(pName, pPwd, &nIndex);
140
141     // 如果查询计费信息为空，表示下机失败
142     if(pBilling == NULL)
143     {
144         cout<<"无该卡信息，请重新输入！"<<endl;
145         return FALSE;
146     }
147
148     // 计算消费金额
149     dbAmount = getAmount(pBilling->tStart);
150
151     // 如果余额小于消费金额，则不能进行下机
152     fBalance = pCard->fBalance - (float)dbAmount;
153     if(fBalance < 0)
154     {
155         return ENOUGHMONEY;
156     }
157
158     // 组装下机信息
159     strcpy(pInfo->aCardName, pName); // 卡号
160     pInfo->fAmount = (float)dbAmount; // 消费金额
161     pInfo->fBalance = fBalance; // 余额
162     pInfo->tStart = pBilling->tStart; // 上机时间
163     pInfo->tEnd = time(NULL); // 下机时间
164
165     return TRUE;
166 }
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183

```

#### 4. 显示下机信息

在 menu.cpp 文件的 settle 函数中调用 doSettle 函数，根据不同的返回值，显示相应的下机信息



```

ice.h    global.h    menu.cpp* x    service.cpp    billing_service.h
[范围]
289 void settle()
290 {
291     char aName[18] = {0};    // 卡号
292     char aPwd[8] = {0};    // 密码
293     int nResult = -1;    // 下机结果
294     SettleInfo* pInfo = NULL;    // 下机信息
295     char aStartTime[30] = {0};    // 上机时间
296     char aEndTime[30] = {0};    // 下机时间
297
298     // 为下机信息动态分配内存
299     pInfo = (SettleInfo*)malloc(sizeof(SettleInfo));
300
301     cout<<"请输入下机卡号(长度为1~18):";
302     cin>>aName;
303     cin.clear();
304     cin.sync();
305
306     cout<<"请输入下机密码(长度为1~8):";
307     cin>>aPwd;
308     cin.clear();
309     cin.sync();
310
311     // 进行下机
312     nResult = doSettle(aName, aPwd, pInfo);
313
314     // 根据下机结果, 提示不同信息
315     cout<<endl;
316     cout<<"-----下机信息如下-----"<<endl;
317     switch(nResult)
318     {
319     case 0:    // 下机失败
320     {
321         cout<<"下机失败!"<<endl;
322         break;
323     }
324     case 1:    // 下机成功
325     {
326         // 输出表格的表头
327         cout<<setw(10)<<"卡号"<<setw(10)<<"消费"<<setw(10)<<"余额";
328         cout<<setw(20)<<"上机时间"<<setw(20)<<"下机时间"<<endl;
329         // 上机时间默认为卡的最后使用时间, 后面添加计费信息后, 使用计费信息时间
330         // 将time_t类型时间转换为字符串, 字符串格式为"年-月-日 时:分"
331         timeToString(pInfo->tStart, aStartTime);
332         timeToString(pInfo->tEnd, aEndTime);
333
334         // 输出下机卡信息
335         cout<<setw(10)<<pInfo->aCardName;    // 一行输出书写语句过长, 分行
336         cout<<setw(10)<<fixed <<setprecision(2)<<pInfo->fAmount;
337         cout<<setw(10)<<fixed <<setprecision(2)<<pInfo->fBalance;
338         cout<<setw(20)<<aStartTime;
339         cout<<setw(20)<<aEndTime<<endl<<endl;
340

```

```

ice.h    global.h    menu.cpp* x    service.cpp    billing_service.h
[范围]
341         cout<<"-----下机成功! -----"<<endl;
342         break;
343     }
344     case 2:    // 正在使用或者已注销
345     {
346         cout<<"该卡未在使用"<<endl;
347         break;
348     }
349     case 3:    // 卡余额不足
350     {
351         cout<<"卡余额不足"<<endl;
352         break;
353     }
354     default:
355     {
356         break;
357     }
358 }
359 // 释放动态分配的内存
360 free(pInfo);
361 }

```

修改 service.cpp 文件中 releaseList 函数，调用 releaseBillingList 函数，释放计费信息链表空间

```

u.cpp    service.cpp x    billing_service.h
[范围]
119 void releaseList()
120 {
121     releaseCardList();    // 释放卡信息链表内存
122     releaseBillingList();    // 释放计费信息链表内存
123 }
124 }

```

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): 3

-----上机-----

请输入上机的卡号 (长度为1~18): 111
请输入上机密码 (长度为1~8): 111
*****-----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
111      111.00      2019-02-16 21:53

-----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
||
```

```
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): 4

-----下机-----

请输入下机卡号 (长度为1~18): 111
请输入下机密码 (长度为1~8): 111

*****-----下机信息如下-----*****
卡号      消费      余额      上机时间      下机时间
111      1.50      109.50      2019-02-16 21:25      2019-02-16 22:03

-----下机成功!-----

-----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):
```

## 二. 更新下机信息

根据下机卡的下机信息，更新文件中的卡信息和消费信息。

### 1. 更新卡信息

修改 service.cpp 文件中的 doSettle 函数，判断下机卡符合下机条件后，调用 card\_service.cpp 文件中的 updateCard 函数，更新卡的相关信息，如果更新失败，则下机失败。

```

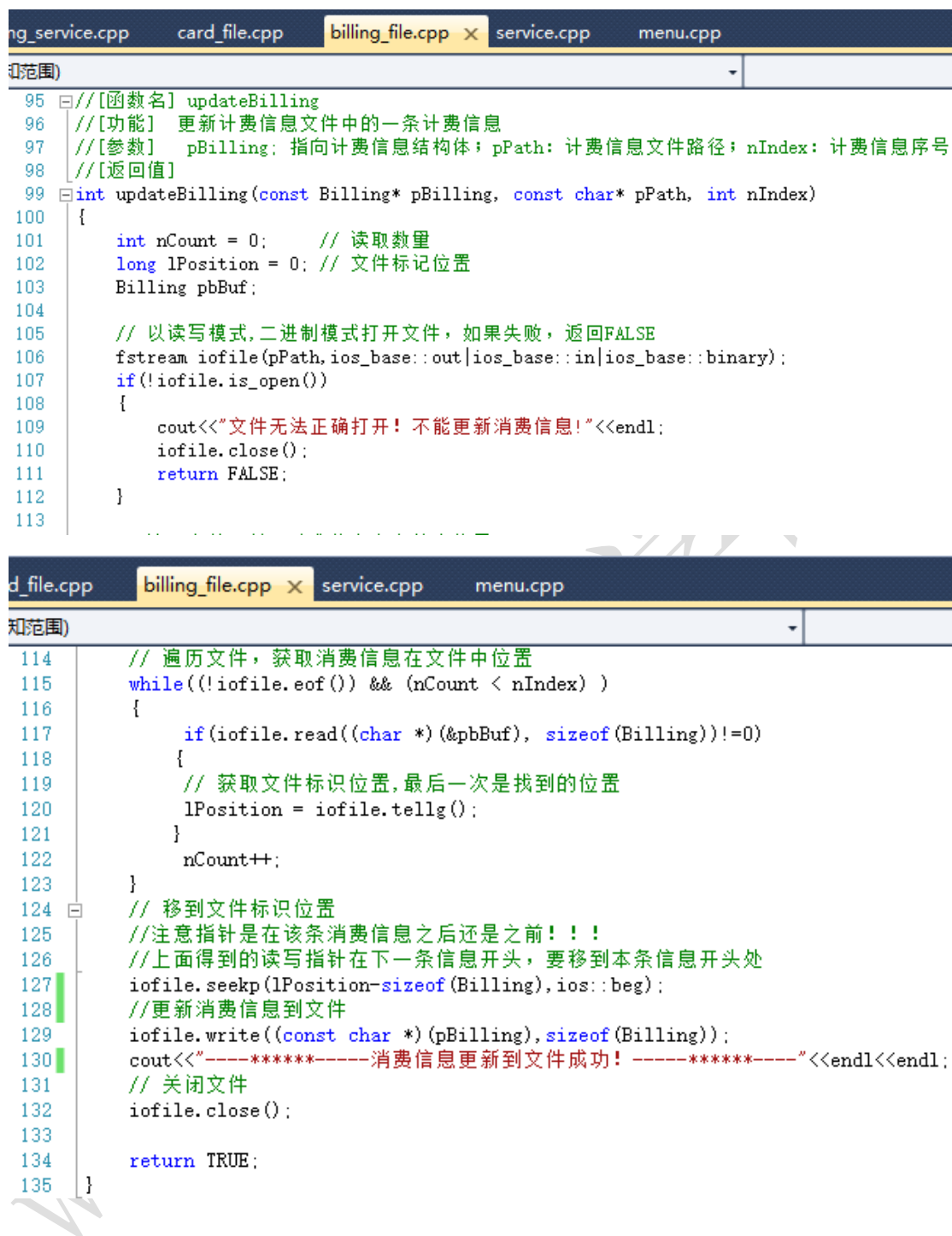
service.cpp x menu.cpp
(未知范围)
169 // 如果余额小于消费金额，则不能进行下机
170 fBalance = pCard->fBalance - (float)dbAmount;
171 if(fBalance < 0)
172 {
173     return ENOUGHMONEY;
174 }
175
176 // 更新卡信息
177 pCard->fBalance = fBalance; // 余额
178 pCard->nStatus = 0; // 状态
179 pCard->tLastTime = time(NULL); // 上次使用时间
180
181 // 更新文件中的卡信息
182 if(FALSE == updateCard(pCard, CARDPATH, nIndex))
183 {
184     return FALSE;
185 }
186
187 // 组装下机信息
188 strcpy(pInfo->aCardName, pName); // 卡号
  
```

（注意：编译连接并运行程序后，卡信息文件的某条记录被更新后，信息字符数变长或缩短，可能覆盖后面的卡信息或在尾部留下垃圾信息，影响下一次对文件的读写!!! 调试时注意在记事本中查看卡信息文件内容的变化，发生变化时，为了调试，可先手工修改卡信息文件）

思考：如何避免上述情况？

### 2. 更新计费信息

在 billing\_file.cpp 文件中定义 updateBilling 函数（对应头文件添加函数声明），实现方法和 card\_file.cpp 文件中的 updateCard 函数类似



```

ng_service.cpp  card_file.cpp  billing_file.cpp x  service.cpp  menu.cpp
[和范围)
95  // [函数名] updateBilling
96  // [功能] 更新计费信息文件中的一条计费信息
97  // [参数]  pBilling: 指向计费信息结构体; pPath: 计费信息文件路径; nIndex: 计费信息序号
98  // [返回值]
99  int updateBilling(const Billing* pBilling, const char* pPath, int nIndex)
100  {
101      int nCount = 0;    // 读取数量
102      long lPosition = 0; // 文件标记位置
103      Billing pbBuf;
104
105      // 以读写模式,二进制模式打开文件, 如果失败, 返回FALSE
106      fstream iofile(pPath, ios_base::out|ios_base::in|ios_base::binary);
107      if(!iofile.is_open())
108      {
109          cout<<"文件无法正确打开! 不能更新消费信息!"<<endl;
110          iofile.close();
111          return FALSE;
112      }
113      ...
114      // 遍历文件, 获取消费信息在文件中位置
115      while((!iofile.eof()) && (nCount < nIndex) )
116      {
117          if(iofile.read((char *)(&pbBuf), sizeof(Billing))!=0)
118          {
119              // 获取文件标识位置, 最后一次是找到的位置
120              lPosition = iofile.tellg();
121          }
122          nCount++;
123      }
124      // 移到文件标识位置
125      // 注意指针是在该条消费信息之后还是之前!!!
126      // 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
127      iofile.seekp(lPosition-sizeof(Billing), ios::beg);
128      // 更新消费信息到文件
129      iofile.write((const char *) (pBilling), sizeof(Billing));
130      cout<<"-----*****-----消费信息更新到文件成功! -----*****-----"<<endl<<endl;
131      // 关闭文件
132      iofile.close();
133
134      return TRUE;
135  }

```

修改 service.cpp 文件中的 doSettle 函数, 调用 billing\_service.cpp 文件中的 updateBilling 函数, 更新消费信息的相关信息, 如果更新失败, 则下机失败。

```

service.cpp* x menu.cpp
(未知范围)
181 // 更新文件中的卡信息
182 if(FALSE == updateCard(pCard, CARDPATH, nIndex))
183 {
184     return FALSE;
185 }
186
187 // 更新计费信息
188 pBilling->fAmount = (float)dbAmount; // 消费信息
189 pBilling->nStatus = 1; // 状态, 已结算
190 pBilling->tEnd = time(NULL); // 下机时间
191
192 // 更新文件中的计费信息
193 if(FALSE == updateBilling(pBilling, BILLINGPATH, nPosition))
194 {
195     return FALSE;
196 }
197
198 // 组装下机信息

```

编译连接并运行程序

```

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    2019-02-17 20:06
-----上机成功! -----
-----计费系统菜单-----
1. 添加卡
2. 查询卡
3. 上机
4. 下机

```

```
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) : 4
-----下机-----
请输入下机卡号(长度为1~18):123
请输入下机密码(长度为1~8):123
-----*****-----卡信息更新到文件成功! -----*****-----
-----*****-----消费信息更新到文件成功! -----*****-----

-----*****-----下机信息如下-----*****-----
      卡号      消费      余额      上机时间      下机时间
      123      0.50      122.50      2019-02-17 20:06      2019-02-17 20:07
-----下机成功! -----

-----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) : _
```

### 三. 本节任务的层次结构和主要调用关系

