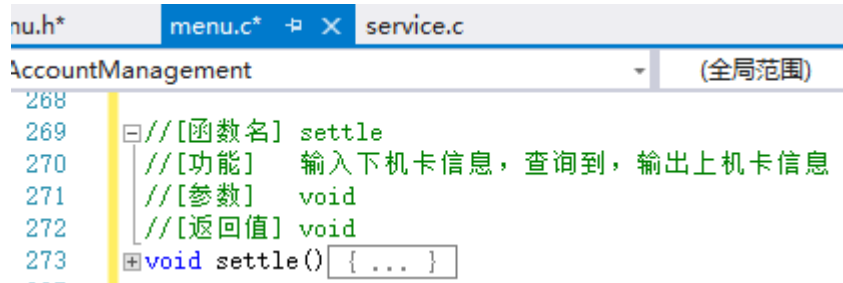


一. 计算消费金额

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.c 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）

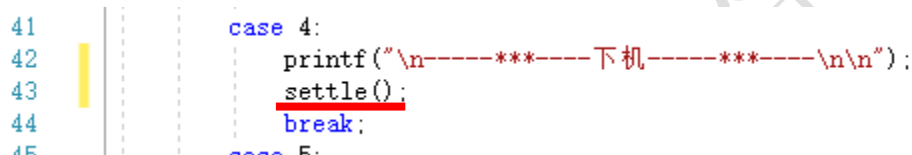


```

268
269 // [函数名] settle
270 // [功能] 输入下机卡信息，查询到，输出上机卡信息
271 // [参数] void
272 // [返回值] void
273 void settle() { ... }

```

在 main.c 文件的 main 函数中 case 4 下调用 settle 函数

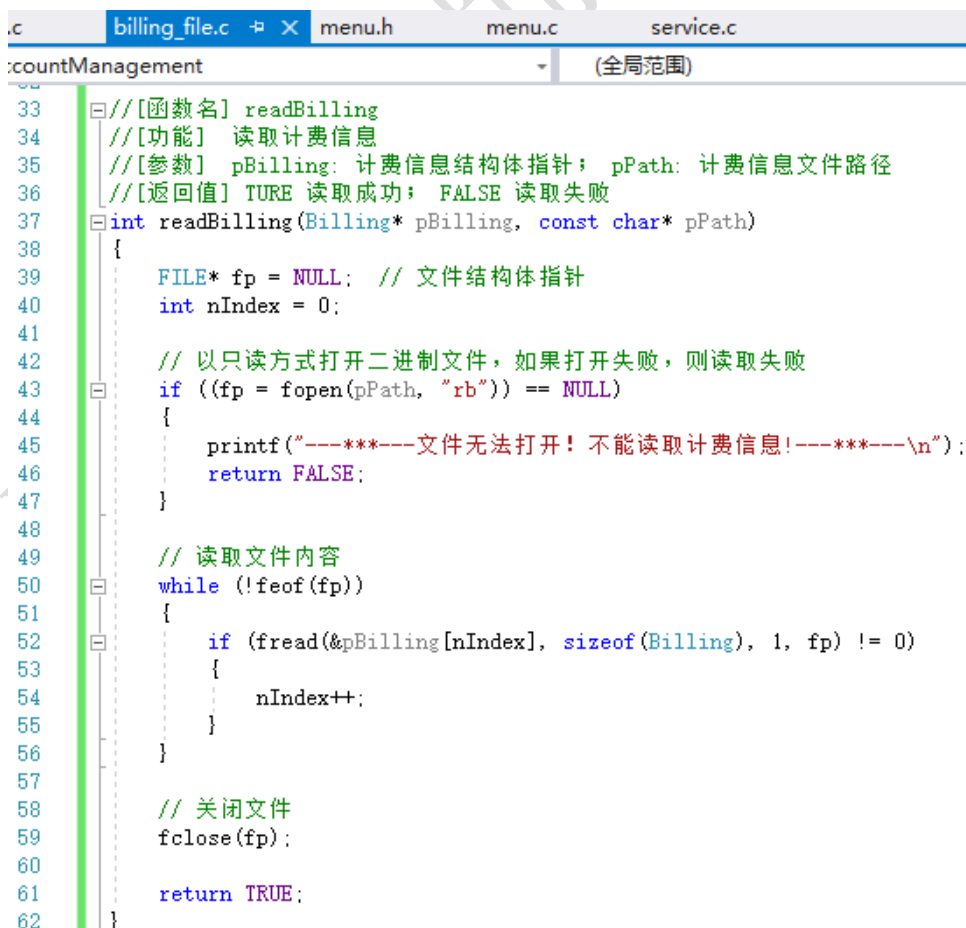


```

41 case 4:
42     printf("\n-----***-----下机-----***-----\n\n");
43     settle();
44     break;
45 case 5:

```

在 billing_file.c 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.c 文件中的 readCard 函数类似，区别是这里读取的是二进制文件，不再需要解析文件内容），代码如下：



```

33 // [函数名] readBilling
34 // [功能] 读取计费信息
35 // [参数] pBilling: 计费信息结构体指针; pPath: 计费信息文件路径
36 // [返回值] TURE 读取成功; FALSE 读取失败
37 int readBilling(Billing* pBilling, const char* pPath)
38 {
39     FILE* fp = NULL; // 文件结构体指针
40     int nIndex = 0;
41
42     // 以只读方式打开二进制文件，如果打开失败，则读取失败
43     if ((fp = fopen(pPath, "rb")) == NULL)
44     {
45         printf("-----***-----文件无法打开！不能读取计费信息！-----\n");
46         return FALSE;
47     }
48
49     // 读取文件内容
50     while (!feof(fp))
51     {
52         if (fread(&pBilling[nIndex], sizeof(Billing), 1, fp) != 0)
53         {
54             nIndex++;
55         }
56     }
57
58     // 关闭文件
59     fclose(fp);
60
61     return TRUE;
62 }

```

在 `billing_file.c` 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.c` 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：

```

g_file.h  billing_file.c  menu.h  menu.c  service.c
countManagement (全局范围)
64  //函数名 getBillingCount
65  //功能 获取文件中计费信息数量
66  //参数 计费信息文件路径
67  //返回值 计费信息数量
68  int getBillingCount(const char* pPath)
69  {
70      FILE* fp = NULL;
71      int nCount = 0;
72      Billing* pBilling = (Billing*)malloc(sizeof(Billing));
73
74      // 以只读方式打开二进制文件，如果打开失败，则读取失败
75      if ((fp = fopen(pPath, "rb")) == NULL)
76      {
77          printf("-----**--文件无法打开！不能统计计费信息数量!-----\n");
78          return 0;
79      }
80
81      // 读取文件，每读一条计数一次
82      while (!feof(fp))
83      {
84          if (fread(pBilling, sizeof(Billing), 1, fp) != 0)
85          {
86              nCount++;
87          }
88      }
89
90      // 关闭文件
91      fclose(fp);
92      free(pBilling);
93      return nCount;
94  }

```

在 `model.h` 文件中，`#endif` 之前定义计费信息链表结点

```

_g_file.h  model.h  billing_file.c
countManagement
45  //计费信息结点
46  typedef struct BillingNode
47  {
48      Billing data;
49      struct BillingNode *next;
50  }BillingNode, *lpBillingNode;
51
52  #endif

```

在 `billing_service.c` 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.c` 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：

```

_service.h  billing_service.c  billing_file.h  model.h
countManagement (全局范围)
1  #define _CRT_SECURE_NO_WARNINGS
2  #include<stdlib.h>    // 包含动态内存分配头文件
3  #include<string.h>
4  #include<time.h>
5  ...
6  #include "model.h"    // 包含数据类型定义头文件
7  #include "global.h"   // 包含全局定义头文件
8  #include "billing_file.h"
9
10  lpBillingNode billingList = NULL;    // 计费信息链表
11

```

```

_service.h  billing_service.c  billing_file.h  model.h
countManagement (全局范围)
34  //【函数名】:  initBillingList
35  //【函数功能】: 初始化计费信息链表
36  //【函数参数】: void
37  //【返回值】:  void
38  void initBillingList()
39  {
40      lpBillingNode head = NULL;
41      if (billingList == NULL)
42      {
43          head = (lpBillingNode)malloc(sizeof(BillingNode));
44          head->next = NULL;
45          billingList = head;
46      }
47

```

```

_service.h  billing_service.c  billing_file.h  model.h
countManagement (全局范围)
49  //【函数名】:  releaseBillingList
50  //【函数功能】: 释放计费信息链表
51  //【函数参数】: void
52  //【返回值】:  void
53  void releaseBillingList()
54  {
55      lpBillingNode cur = billingList;
56      lpBillingNode next = NULL;
57      //销毁链表
58      while (cur != NULL)
59      {
60          next = cur->next; //结点内存释放前，next保存其后继
61          free(cur); //释放结点内存
62          cur = next;
63      }
64      billingList = NULL;
65

```

在 billing_service.c 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.c 文件中的 getCard 函数类似)

```

g_service.h  billing_service.c  billing_file.h  model.h  billing_
countManagement  (全局范围)
67  // [函数名]: getBilling
68  // [函数功能]: 从计费信息文件中，获取计费信息保存到链表中
69  // [函数参数]: void
70  // [返回值]: TRUE 获取信息成功，FALSE 获取信息失败
71  int getBilling()
72  {
73      int nCount = 0;  // 计费信息数量
74      Billing* pBilling = NULL;  // 计费信息
75      lpBillingNode node = NULL;
76      int i = 0;  // 循环变量
77      lpBillingNode cur = NULL;
78
79      // 如果链表不为空，释放链表
80      if (billingList != NULL)
81      {
82          releaseBillingList();
83      }
84      // 初始化链表
85      initBillingList();
86
87      // 获取计费信息数量
88      nCount = getBillingCount(BILLINGPATH);
89      // 动态分配内存
90      pBilling = (Billing*)malloc(sizeof(Billing) * nCount);
91
92      if (pBilling != NULL)
93      {
94          // 获取计费信息
95          if (FALSE == readBilling(pBilling, BILLINGPATH))
96          {
97              free(pBilling);
98              return FALSE;
99          }
100         // 将计费信息保存到链表中
101         for (i = 0, node = billingList; i < nCount; i++)
102         {
103             // 为当前结点分配内存
104             cur = (lpBillingNode)malloc(sizeof(BillingNode));
105             // 如果分配失败，则在返回FALSE前，需要释放pBilling的内存
106             if (cur == NULL)
107             {
108                 free(pBilling);
109                 return FALSE;
110             }
111             // 初始化新的空间，全部赋值为0
112             memset(cur, 0, sizeof(BillingNode));
113             // 将数据添加到结点中
114             cur->data = pBilling[i];
115             cur->next = NULL;
116             // 将结点添加到链表中
117             node->next = cur;
118             node = cur;
119         }
120     }
121 }

```

```

119     free(pBilling);
120     return TRUE;
121 }
122 return FALSE;
123 }

```

在 `billing_service.c` 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.c` 文件中的 `checkCard` 函数类似）

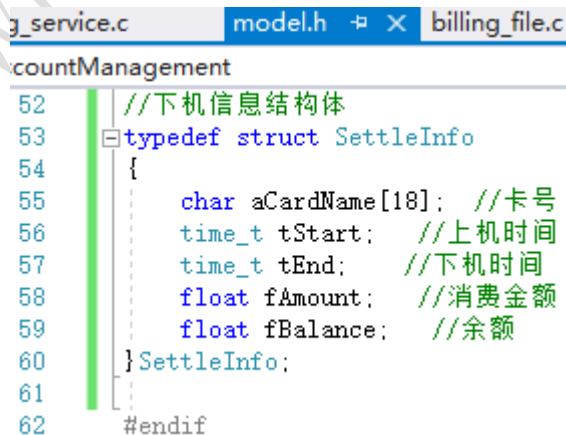


```

g_service.h  billing_service.c  billing_file.h  model.h  billing_file.c  menu
ccountManagement  (全局范围)
125  //函数名: queryBilling
126  //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
127  //函数参数: pName: 要查询的计费信息的卡号
128  //          pIndex: 查询到的计费信息在链表中的索引号
129  //返回值: 查询到的计费信息指针
130  Billing* queryBilling(const char* pName, int* pIndex)
131  {
132      lpBillingNode node = NULL;
133      int nIndex = 0;
134
135      if (FALSE == getBilling())
136      {
137          return NULL;
138      }
139      // 遍历链表
140      node = billingList->next;
141      while (node != NULL)
142      {
143          // 查询到卡号相同, 并且没有结算的计费信息
144          if (strcmp(node->data.aCardName, pName) == 0 && node->data.nStatus == 0)
145          {
146              return &node->data;
147          }
148          node = node->next;
149          nIndex++;
150          *pIndex = nIndex;
151      }
152      return NULL;
153  }

```

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



```

g_service.c  model.h  billing_file.c
ccountManagement
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.c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

```

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

```

```

el.h  service.c  x  billing_service.c
countManagement
1  #define _CRT_SECURE_NO_WARNINGS
2  #include<stdio.h>
3  #include<string.h>
4  #include "model.h"
5  #include "global.h"
6  #include "card_file.h"
7  #include "card_service.h"
8  #include "billing_file.h"
9  #include "billing_service.h"
10

```

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 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。

```

.h  global.h  x  service.c  billing_service.c  billing_file.c
countManagement  (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.txt" // 卡信息保存路径
12 #define BILLINGPATH "data\\billing.ams" // 计费信息保存路径
13

```

在 service.c 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 | service.c | billing_service.c | billing_file.c | menu.h | n |
|-------|-----------|-------------------|----------------|--------|---|

accountManagement (全局范围)

```

160 // [函数名] getAmount
161 // [功能] 根据上机时间, 计算消费金额
162 // [参数] 上机时间
163 // [返回值] 消费金额
164 double getAmount(time_t tStart)
165 {
166     double dbAmount = 0; // 消费金额
167     int nCount = 0; // 上机的时间单元数, 每个单元15分钟
168     int nSec = 0; // 消费时间(单位: 秒)
169     int nMinutes = 0; // 消费时间(单位: 分钟)
170     time_t tEnd = time(NULL); // 结算时间为当前时间
171
172     // 计算消费时长
173     nSec = (int)(tEnd - tStart);
174     nMinutes = nSec / 60;
175     // 计算消费的时间单元数
176     if (nMinutes % UNIT == 0)
177     {
178         nCount = nMinutes / UNIT; // 整除
179     }
180     else
181     {
182         nCount = nMinutes / UNIT + 1; // 不整除
183     }
184     // 计算消费金额
185     dbAmount = nCount * CHARGE;
186     return dbAmount;
187 }

```

3. 保存下机信息

修改 service.c 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 (getAmount 函数如果定义在 doSettle 函数后, 文件前面需要添加 #include "service.h")

| | | | | | |
|-------|-----------|-------------------|----------------|--------|---|
| bal.h | service.c | billing_service.c | billing_file.c | menu.h | n |
|-------|-----------|-------------------|----------------|--------|---|

accountManagement (全局范围)

```

125 int doSettle(const char* pName, const char* pPwd, SettleInfo* pInfo)
126 {
127     Card* pCard = NULL;
128     Billing* pBilling = NULL;
129     int nIndex = 0; // 卡信息在链表中的索引号
130     int nPosition = 0; // 计费信息在链表中的索引号
131     double dbAmount = 0.0; // 消费金额
132     float fBalance = 0.0; // 余额
133
134     // 查询上机卡
135     pCard = checkCard(pName, pPwd, &nIndex);
136
137     // ...

```



```

154 // 如果查询计费信息为空，表示下机失败
155 if (pBilling == NULL)
156 {
157     printf("----***---无该卡信息，请重新输入！----***---\n");
158     return FALSE;
159 }
160
161 // 计算消费金额
162 dbAmount = getAmount(pBilling->tStart);
163
164 // 如果余额小于消费金额，则不能进行下机
165 fBalance = pCard->fBalance - (float)dbAmount;
166 if (fBalance < 0)
167 {
168     return ENOUGHMONEY;
169 }
170
171 // 组装下机信息
172 strcpy(pInfo->aCardName, pName); // 卡号
173 pInfo->fAmount = (float)dbAmount; // 消费金额
174 pInfo->fBalance = fBalance; // 余额
175 pInfo->tStart = pBilling->tStart; // 上机时间
176 pInfo->tEnd = time(NULL); // 下机时间
177
178 return TRUE;
179 }

```

4. 显示下机信息

在 menu.c 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

```

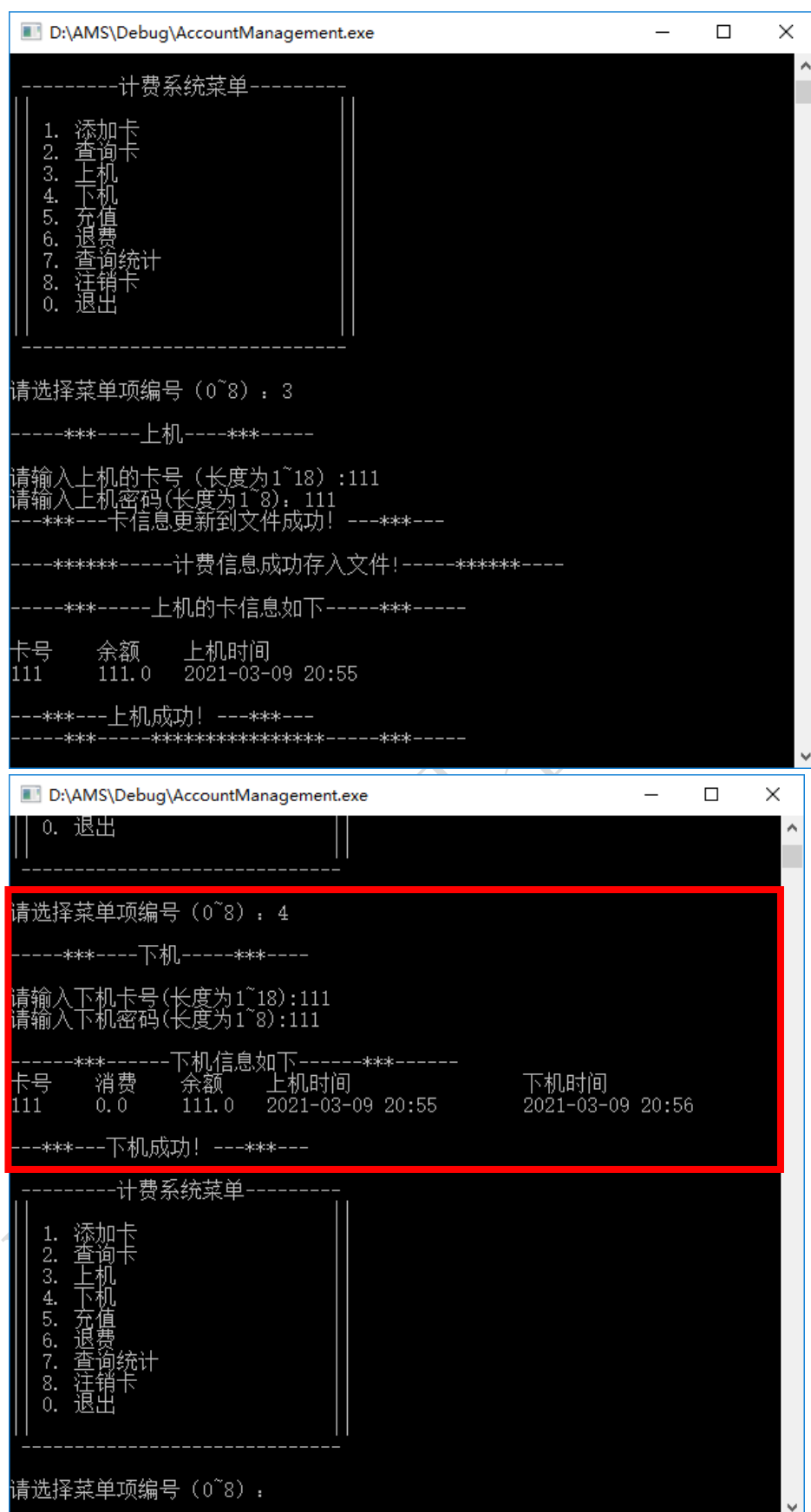
bal.h  menu.c  [X]
accountManagement (全局范围)
273 void settle()
274 {
275     char aName[18] = { 0 }; // 卡号
276     char aPwd[8] = { 0 }; // 密码
277     int nResult = -1; // 下机结果
278     SettleInfo* pInfo = NULL; // 下机信息
279     char aStartTime[30] = { 0 }; // 上机时间
280     char aEndTime[30] = { 0 }; // 下机时间
281
282     // 为下机信息动态分配内存
283     pInfo = (SettleInfo*)malloc(sizeof(SettleInfo));
284
285     printf("请输入下机卡号(长度为1~18):");
286     scanf("%s", aName);
287
288     printf("请输入下机密码(长度为1~8):");
289     scanf("%s", aPwd);
290
291     // 进行下机
292     nResult = doSettle(aName, aPwd, pInfo);
293 }

```

修改 service.c 文件中 releaseList 函数，调用 releaseBillingList 空间

```
al.h service.c menu.c
ccountManagement (全局范围)
116 void releaseList()
117 {
118     releaseCardList(); // 释放卡信息链表内存
119     releaseBillingList(); // 释放计费信息链表内存
120 }
```

编译连接并运行程序



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.c 文件中的 doSettle 函数，判断下机卡符合下机条件后，调用 card_service.c 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。

```

165 // 如果余额小于消费金额，则不能进行下机
166 fBalance = pCard->fBalance - (float)dbAmount;
167 if (fBalance < 0)
168 {
169     return ENOUGHMONEY;
170 }
171
172 // 更新卡信息
173 pCard->fBalance = fBalance; // 余额
174 pCard->nStatus = 0; // 状态
175 pCard->tLastTime = time(NULL); // 上次使用时间
176
177 // 更新文件中的卡信息
178 if (FALSE == updateCard(pCard, CARDPATH, nIndex))
179 {
180     return FALSE;
181 }
182
183 // 组装下机信息
184 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，为了当前的调试，可先手工修改 card.txt 卡信息文件为正确格式)

思考：如何避免上述情况？

2. 更新计费信息

在 billing_file.c 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.c 文件中的 updateCard 函数类似

```

ng_file.h  billing_file.c  global.h  service.c  menu.c
accountManagement  (全局范围)
96 // [函数名] updateBilling
97 // [功能] 更新计费信息文件中的一条计费信息
98 // [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
99 // [返回值] TRUE 更新成功; FALSE 更新失败
100 int updateBilling(const Billing* pBilling, const char* pPath, int nIndex)
101 {
102     FILE* fp = NULL; // 文件结构体指针
103     int nCount = 0; // 读取数量
104     long lPosition = 0; // 文件标记位置
105     Billing pbBuf;
106
107     // 以读写模式打开二进制文件
108     if ((fp = fopen(pPath, "rb+")) == NULL)
109     {
110         printf("-----文件无法打开! 不能更新计费信息!-----\n");
111         return FALSE;
112     }

```

```

113
114 // 遍历文件，获取消费信息在文件中位置
115 while ((!feof(fp)) && (nCount < nIndex))
116 {
117     if (fread(&pbBuf, sizeof(Billing), 1, fp) != 0)
118     {
119         // 获取文件标识位置，最后一次是找到的位置
120         lPosition = ftell(fp);
121     }
122     nCount++;
123 }
124
125 // 移动文件标记位置，到要更新的计费信息记录开头
126 fseek(fp, lPosition, 0);
127
128 // 更新计费信息到文件
129 fwrite(pBilling, sizeof(Billing), 1, fp);
130 printf("----***----计费信息更新到文件成功!----***----\n\n");
131 // 关闭文件
132 fclose(fp);
133
134 return TRUE;
135 }

```

修改 service.c 文件中的 doSettle 函数，调用 billing_service.c 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。

```

177 // 更新文件中的卡信息
178 if (FALSE == updateCard(pCard, CARDPATH, nIndex))
179 {
180     return FALSE;
181 }
182
183 // 更新计费信息
184 pBilling->fAmount = (float)dbAmount; // 消费信息
185 pBilling->nStatus = 1; // 状态，已结算
186 pBilling->tEnd = time(NULL); // 下机时间
187
188 // 更新文件中的计费信息
189 if (FALSE == updateBilling(pBilling, BILLINGPATH, nPosition))
190 {
191     return FALSE;
192 }
193
194 // 组装下机信息
195 strcpy(pInfo->aCardName, pName); // 卡号

```

(注意：多次调试，卡信息文件中信息的修改，可能跟计费信息文件中信息发生冲突，不一致，导致下机失败，可以删除计费信息文件，新建计费信息文件，重新调试)

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) : 3

-----***-----上机-----***-----

请输入上机的卡号 (长度为1~18) : 1212
请输入上机密码 (长度为1~8) : 1212
-----***-----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
1212    1212.0  2021-03-14 13:03

-----***-----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
```

```
D:\AMS\Debug\AccountManagement.exe
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) : 1212
请输入下机密码 (长度为1~8) : 1212
-----***-----卡信息更新到文件成功! -----***-----

-----***-----计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
1212    0.5     1211.5  2021-03-14 13:03  2021-03-14 13:04

-----***-----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. 本次任务的层次结构和主要调用关系

