

## 一. 卡信息链表结点定义

前面实验中卡信息在内存中存放在结构体数组, 数组在内存中要占用连续的存储空间, 并且大小固定, 要么在数据量较少时会造成内存资源浪费, 要么在数据量较大时会出现存储资源不足。本次实验采用链表这种动态的数据结构, 在需要存储时才动态分配内存资源, 不需要时释放占用的内存资源。

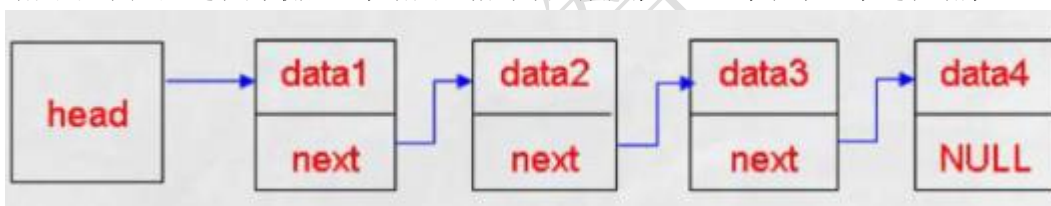
在 `model.h` 文件中 `endif` 之前定义链表 (单链表) 的结点结构体, 结点的数据域是卡信息结构体, 指针域是指向下一个结点的指针 (这是把多条分散存储的卡信息联系在一起的键), 并用 `typedef` 声明了结点类型和指向结点的指针类型, 便于后面使用。

```

20 typedef struct CardNode
21 {
22     Card data;    // 结点数据区
23     struct CardNode* next; // 指向下一个结点的指针
24 } CardNode, *lpCardNode;
25
26 #endif
27

```

链表包括头指针, 结点和表尾。头指针只存放一个地址 (即链表第一个结点所在的地址); 中间结点的数据域存放卡信息, 指针域存放下一个结点的地址 (即指向下一个结点); 表尾是链表的最后一个结点, 指针域一般赋值 `NULL` 来表示整个链表结束



链表的各个结点在内存中一般不是连续存放的, 所以要找某个结点的信息, 需要从头指针开始, 按照指针的指向依次访问每个结点

(提示: 如果指针未初始化, 编译器会把栈内存上的未初始化指针全部填成 `0xcccccccc`, 当成字符串看就成了“烫烫烫烫。。。”; 编译器会把堆内存上的未初始化指针全部填成 `0xcdcdcdcd`, 当成字符串看就成了“屯屯屯屯。。。”, 编译器把野指针自动初始化后, 便于调试程序时发现问题)

## 二. 初始化链表

### 1. 定义卡信息链表的指针变量

在 `card_service.c` 中定义一个 **全局变量**, 指向卡信息链表结点的指针变量, 即链表的头指针 (待链表功能实现后, 删除原来的卡结构体数组定义的相关变量: 卡结构体数组 `aCard[50]` 和卡数量 `nCount`)

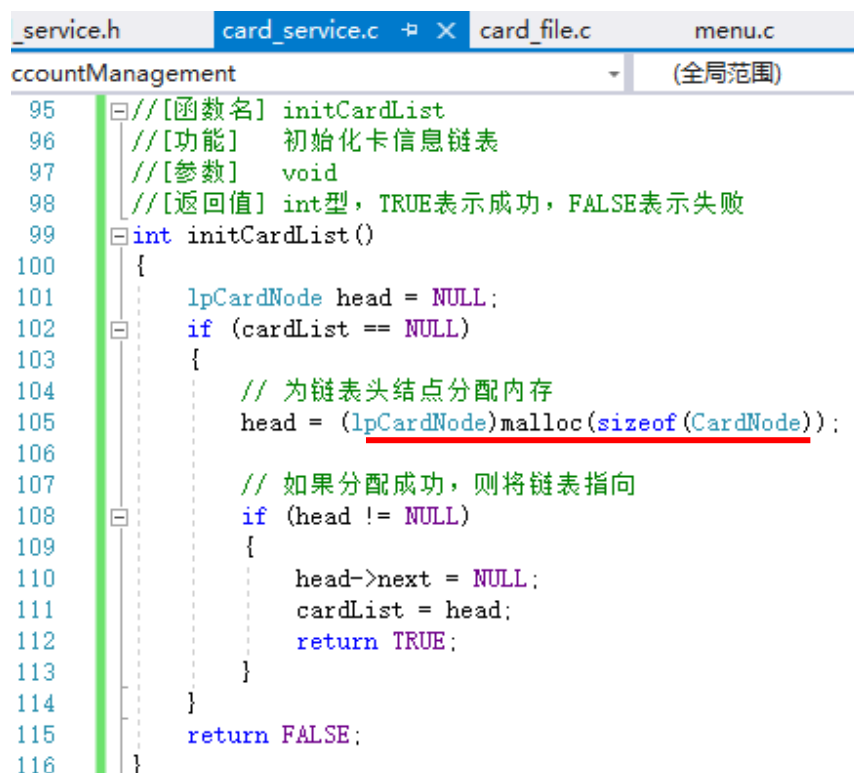
```

9
10 Card aCard[50]; // 卡结构体数组
11 int nCount=0; // 卡结构体数组中的实际卡信息数
12 lpCardNode cardList=NULL; // 卡信息链表类型变量
13

```

### 2. 定义函数

在 card\_service.c 中定义 initCardList 函数(对应头文件中加该函数声明),该函数初始化链表,代码如下:



```

_service.h  card_service.c  card_file.c  menu.c
ccountManagement (全局范围)
95  //[[函数名] initCardList
96  //[[功能] 初始化卡信息链表
97  //[[参数] void
98  //[[返回值] int型, TRUE表示成功, FALSE表示失败
99  int initCardList()
100 {
101     lpCardNode head = NULL;
102     if (cardList == NULL)
103     {
104         // 为链表头结点分配内存
105         head = (lpCardNode)malloc(sizeof(CardNode));
106
107         // 如果分配成功, 则将链表指向
108         if (head != NULL)
109         {
110             head->next = NULL;
111             cardList = head;
112             return TRUE;
113         }
114     }
115     return FALSE;
116 }

```

其中用到 malloc 函数和 sizeof 操作符(使用前可能需要#include <stdlib.h> )

sizeof 操作符返回一个对象或者类型所占的内存字节数,这里 sizeof(CardNode)返回卡信息链表结点结构体类型所占用的内存大小;

malloc 函数为运行中的程序在堆区动态开辟一定大小的内存空间,函数原型:

void \* malloc (unsigned int size)

函数的返回值是一个指向所分配区域的无类型指针(即区域第一个字节的地址),这里需要强制转换为指向链表结点的指针类型 lpCardNode

思考:

1. 程序在内存中的分区有哪些? 各有什么特点?
2. malloc 函数在堆区分配存储区域,必须用 free 函数释放吗? 如果在堆区分配的空间不释放会影响程序运行吗?

### 三. 释放卡信息链表

1. 在 card\_service.c 中定义 releaseCardList 函数(对应头文件中加该函数声明),该函数释放链表占用的内存资源,遍历链表的每个结点,每个结点通过 free 函数释放,最后链表头指针为 NULL,不指向任何结点,代码如下:



```

118 // [函数名] releaseCardList
119 // [功能] 释放卡信息链表
120 // [参数] void
121 // [返回值] void
122 void releaseCardList()
123 {
124     lpCardNode cur = cardList;
125     lpCardNode next = NULL;
126
127     while (cur != NULL)
128     {
129         next = cur->next; // 释放cur结点前，用next保存它的后继结点
130         free(cur);        // 释放cur结点
131         cur = next;
132     }
133     cardList = NULL;
134 }

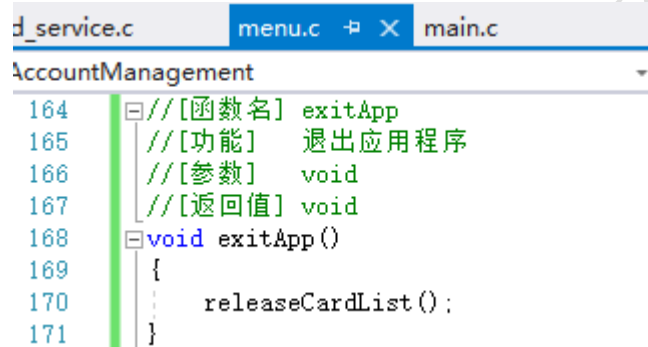
```

其中使用 free 函数，其原型：void free (void\* p)

释放由 p 指向的动态空间，p 是最近一次调用 malloc 函数得到的函数返回值

## 2. 调用 releaseCardList 函数

在 menu.c 中定义 exitApp 函数（对应头文件中加该函数声明），调用 releaseCardList 函数：



```

164 // [函数名] exitApp
165 // [功能] 退出应用程序
166 // [参数] void
167 // [返回值] void
168 void exitApp()
169 {
170     releaseCardList();
171 }

```

在 main.c 中退出选项（case 0）后调用 exitApp 函数



```

55 case 0:
56     exitApp();
57     printf("\n---***---谢谢你使用本系统! ---***---\n");
58     break;

```

## 四. 从文件读取卡信息到链表

不管链表中有无数据，都先清空链表中的数据，保证链表和文件中数据同步一致。修改 card\_service.c 中 getCard 函数如下：

```

.h      card_service.c  menu.c      main.c
countManagement  (全局范围)
70  // [函数名] getCard
71  // [功能] 从卡信息文件中，获取卡信息，保存到链表中
72  // [参数] void
73  // [返回值] 读取成功返回TRUE，否则返回FALSE
74  int getCard()
75  {
76      int i = 0;
77      Card* pCard = NULL; // 读取到的一条卡信息
78      int nCount = 0; // 实际卡信息数
79      lpCardNode node = NULL; // 当前尾结点指针
80      lpCardNode cur = NULL; // 要添加到链表中的结点指针
81
82      // 清除链表中已经存在的数据
83      if (cardList != NULL)
84      {
85          releaseCardList();
86      }
87      // 初始化链表
88      initCardList();
89
90      // 获取文件中卡信息个数
91      nCount = getCardCount(CARDPATH);
92      if (nCount == 0)
93      {
94          return FALSE;
95      }
96      // 动态分配内存用来保存所有卡信息，相当于结构体数组，pCard相当于数组名
97      pCard = (Card*)malloc(sizeof(Card)*nCount);
98      if (pCard != NULL)
99      {
100         // 如果返回FALSE，表示读取卡信息失败
101         if (0 == readCard(pCard, CARDPATH))
102         {
103             free(pCard);
104             return FALSE;
105         }
106         // 将读取的卡信息，保存到链表中
107         for (i = 0, node = cardList; i < nCount; i++)
108         {
109             // 为结点分配内存
110             cur = (lpCardNode)malloc(sizeof(CardNode));
111             // 如果分配内存失败，则返回
112             if (cur == NULL)
113             {
114                 free(pCard);
115                 return FALSE;
116             }

```

```

117 // 初始化新的空间，全部赋值为0
118 memset(cur, 0, sizeof(CardNode));
119 // 将卡信息保存到结点中
120 cur->data = pCard[i]; //结构体指针当数组名使用
121 cur->next = NULL;
122 // 将结点添加到链表尾部
123 node->next = cur;
124 //尾结点指针后移
125 node = cur;
126 }
127 // 释放内存
128 free(pCard);
129 return TRUE;
130 }
131 return FALSE;
132 }

```

注意：这里程序中先将文件中的卡信息一次性全部读取到 pCard 指向的堆内存块（这里的堆内存块是分配的连续的内存空间，空间大小是 nCount 个 Card 结构体的大小，即相当于声明了 Card pCard[nCount] 数组，pCard 指针名相当于数组名），然后再将这个堆内存块中的卡信息逐条添加到堆中新分配的链表节点中（链表中各个节点可能分散在不连续的内存空间），程序运行中实际上要在堆中保存两份所有卡信息的内容。当卡信息数量较少，堆内存空间足够大时不会出现问题，但是如果卡信息数量较多或者同时运行的其他程序较多时，运行时可能出现的问题，比如：堆中剩余的内存空间，不足以分配能存放所有卡信息的连续的内存空间（即 pCard 分配失败），堆中剩余的内存空间耗尽，已经放不下两份所有卡信息（即运行中 cur 分配失败）等等问题。。

✪ 选做内容：有兴趣的同学可以尝试修改或改进程序，解决以上问题。。

思考：

1. 用 malloc 函数分配的堆内存空间要 free 释放，这里 cur 指针变量的怎么没有 free？在哪儿 free？
2. 结构体指针可以用作数组名？反过来，数组名是否可以作为指针使用？
3. node 和 cur 变量起什么作用？

## 五. 从链表中精确查询卡信息

这个功能不仅在查询卡信息是调用，还在添加卡时查重时调用，修改 queryCard 函数如下：

```

u.h  card_service.c  menu.c  main.c
ccountManagement  (全局范围)
42  //[[函数名] queryCard
43  //[[功能] 在链表中查找指定卡号的卡信息
44  //[[参数] 用户输入的要查询的卡号地址
45  //[[返回值] 链表中查询到的卡信息地址,没有找到返回NULL
46  Card* queryCard(const char* pName)
47  {
48      Card* pCard = NULL;
49      lpCardNode p; //用于查找中迭代,依次指向链表中的每个结点
50
51      //从卡信息文件中读取卡信息到链表,失败返回NULL
52      if (FALSE == getCard())
53      {
54          return NULL;
55      }
56      //指向链表第一个结点
57      p = cardList->next;
58      //在链表中查找指定卡号
59      while (p != NULL)
60      {
61          if (strcmp(p->data.aName, pName) == 0)
62          {
63              pCard = &(p->data);
64              return pCard;
65          }
66          else
67          {
68              p = p->next;
69          }
70      }
71      //没有找到,返回NULL
72      return pCard;
73  }

```

思考:

1. 查询时可不可以直接使用 cardList 迭代 (cardList=cardList->next)? 为什么要使用一个 p 链表结点指针变量?
2. pCard 是函数内声明的局部变量(指针),为什么可以在函数结束时通过 return pCard 返回 pCard 指向的卡信息地址?

六. 重新编译连接并运行程序

执行“查询卡”



#### 七. 从链表中模糊查询卡信息

模糊查询是根据用户输入的关键字，从链表中查找所有包含相同关键字的卡信息。

在card\_service.c中定义queryCards函数(对应头文件中加该函数声明)，遍历链表，用strstr函数判断是否包含用户输入的卡号关键字，如果找到，就增大卡信息结构体(相当于数组)的存储空间，把找到的卡信息保存于其中，最后返回卡信息结构体，代码如下：

其中用到**strstr函数**，函数原型：char \*(strstr)(const char \*s1,const char \*s2)；

函数参数是字符串s1，s2；函数功能是找到s2字符串在s1字符串中第一次出现的位置；返回该位置的指针，如找不到，返回空指针。

其中用到**realloc函数**，函数原型：void \*realloc(void \*p,unsigned int size)；

如果已经通过malloc函数或calloc函数获得了动态空间，想改变其大小，可以用realloc函数**重新分配**，用realloc函数将p所指向的动态空间的大小改变为size，p的值不变。如果重分配不成功，返回NULL。

如果新分配的内存减少，realloc仅仅是改变索引的信息，返回原指针，原来内存中数据有可能丢失；如果size = 0，则等价于释放内存

如果是将分配的内存扩大，则有以下情况：

- 1) 如果当前内存段后面有需要的内存空间，则直接扩展这段内存空间，realloc()将返回原指针。

- 2) 如果当前内存段后面的空闲字节不够,那么就使用堆中的第一个能够满足这一要求的内存块,将目前的数据复制到新的位置,并将原来的数据块释放掉,返回新的内存块位置。
- 3) 如果申请失败,将返回NULL,此时,原来的指针仍然有效。

这里代码中将原来的空间增大一个Card结构体的大小,以便保存下一条找到的卡信息。

```

d_service.h  card_service.c  menu.c  main.c
AccountManagement  (全局范围)
180  //函数名 queryCards
181  //功能 在卡信息链表中,模糊查询包含的所有卡信息
182  //参数  pName: 指向用户输入的要查询的卡号;  pIndex: 指向查到的卡信息数变量
183  //返回值 指向卡信息结构体的指针
184  Card* queryCards(const char* pName, int* pIndex)
185  {
186      lpCardNode node = NULL;
187      Card* pCard = NULL; //保存查询到的符合条件的卡信息
188
189      //从卡信息文件中读取卡信息到链表,失败返回NULL
190      if (FALSE == getCard())
191      {
192          return NULL;
193      }
194
195      // 首先分配一个Card大小内存空间
196      pCard = (Card*)malloc(sizeof(Card));
197      if (pCard == NULL)
198      {
199          return NULL;
200      }
201
202      // 从链表的头结点指向的第一个结点开始遍历
203      node = cardList->next;
204
205      // 遍历链表,结点为空表示到达链表尾部
206      while (node != NULL)
207      {
208          // 判断在遍历到的结点的卡号中,是否包含pName字符串
209          if (strstr(node->data.aName, pName) != NULL)
210          {
211              // 如果有,则保存结点中的数据
212              pCard[*pIndex] = node->data;
213              (*pIndex)++;
214
215              // 重新为指针分配内存,比原来正好大一个Card的大小,包含已有的内容
216              pCard = (Card*)realloc(pCard, ((*pIndex) + 1) * sizeof(Card));
217          }
218          // 移到链表的下一个结点
219          node = node->next;
220      }
221      //如果没有找到符合条件的卡,释放分配的空间
222      if (*pIndex == 0) { free(pCard); pCard = NULL; }
223
224      return pCard;
225  }

```

注意:

1. 模糊查询queryCards函数的参数有2个,比queryCard函数增加了一个pIndex整型指针变量,它所指向的整型变量存放的是当前查询到的符合条件的卡数量,函数执行结束时,实际上就是最终查询到的符合条件的卡数量,是作为返回值使用的;

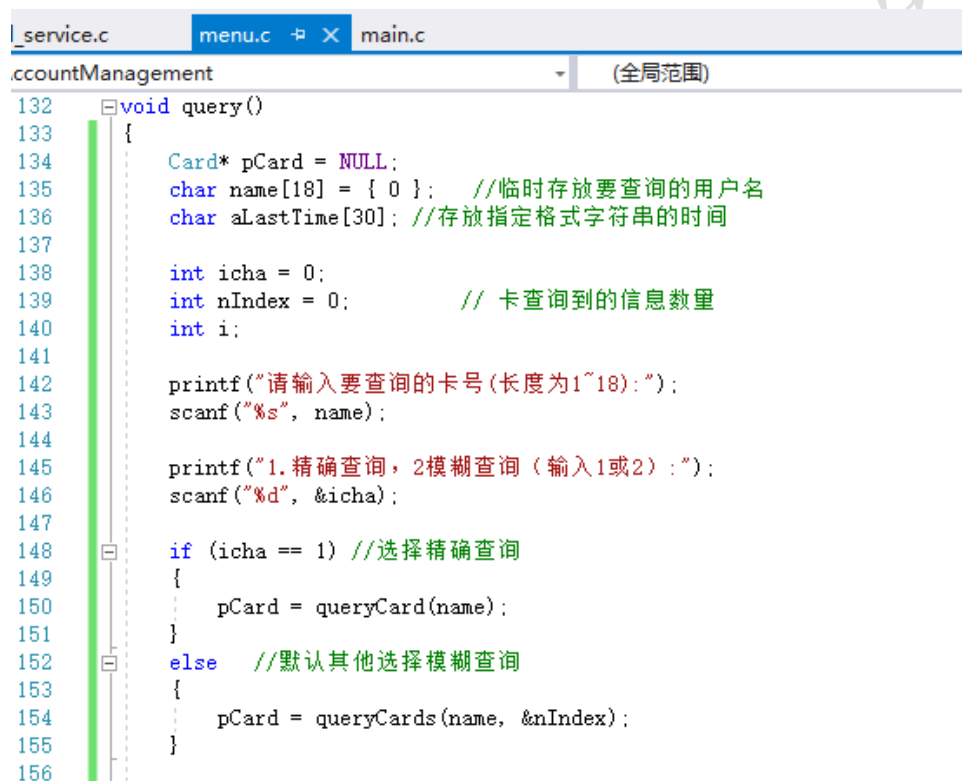


2. 在遍历链表的循环中，每找到一条符合条件的卡信息，pIndex指向的卡数量变量的值要加一，所以是(\*pIndex)++，而不是pIndex++（指针加一）；
3. pCard指向的空间相当一个可以不断增加数量大小的动态卡信息数组，在需要时增加一个卡信息大小的空间（相当于数组成员数量加一），要增大一个Card大小的存储空间sizeof(Card)时，整个存储空间中的卡数量是原来的数量上加一，即(\*pIndex)+1，所以新分配的存储空间大小是(((\*pIndex)+1)\* sizeof(Card))。

思考：数组与数组变量可以直接赋值吗？结构体与结构体变量可以直接赋值吗？

#### 八. 调用queryCards函数实现模糊查询

在menu.c中修改query函数，用户可以选择精确查询还是模糊查询（文件开头可能需要加#include<stdlib.h>），代码如下：



```

_service.c  menu.c  main.c
ccountManagement  (全局范围)
132 void query()
133 {
134     Card* pCard = NULL;
135     char name[18] = { 0 }; //临时存放要查询的用户名
136     char aLastTime[30]; //存放指定格式字符串的时间
137
138     int icha = 0;
139     int nIndex = 0; // 卡查询到的信息数量
140     int i;
141
142     printf("请输入要查询的卡号(长度为1~18):");
143     scanf("%s", name);
144
145     printf("1.精确查询, 2模糊查询(输入1或2):");
146     scanf("%d", &icha);
147
148     if (icha == 1) //选择精确查询
149     {
150         pCard = queryCard(name);
151     }
152     else //默认其他选择模糊查询
153     {
154         pCard = queryCards(name, &nIndex);
155     }
156

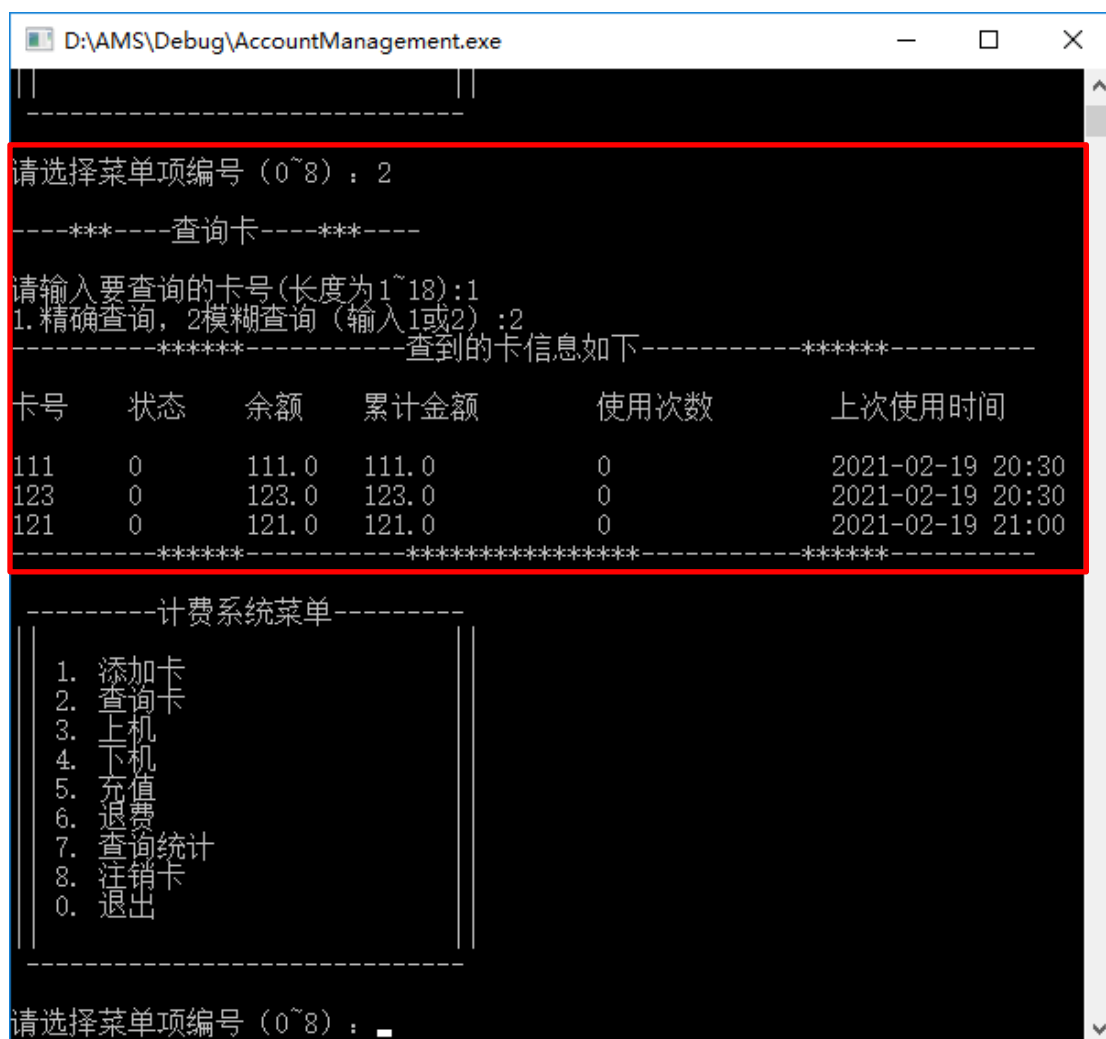
```

```

157 // 如果pCard为NULL，表示没有该卡的信息
158 if (pCard == NULL)
159 {
160     printf("\n-----*****---没有查询到该卡的信息!-----*****---\n");
161 }
162 else
163 {
164     printf("-----*****---查到的卡信息如下-----*****---\n\n");
165     // 输出表格的表头
166     printf("卡号\t状态\t余额\t累计金额\t使用次数\t上次使用时间\n\n");
167
168     if (icha == 1) //精确查询结果输出
169     {
170         //将time_t类型时间转换为字符串，字符串格式为“年-月-日 时：分”
171         timeToString(pCard->tLastTime, aLastTime);
172         //显示查询到的卡信息
173         printf("%s\t%d\t%.1f\t%.1f\t\t%d\t\t%s\n\n", pCard->aName, pCard->nStatus,
174             pCard->fBalance, pCard->fTotalUse, pCard->nUseCount, aLastTime);
175     }
176     else //模糊查询结果输出
177     {
178         for (i = 0; i < nIndex; i++)
179         {
180             //将time_t类型时间转换为字符串，字符串格式为“年-月-日 时：分”
181             timeToString(pCard[i].tLastTime, aLastTime); //结构体指针当数组名使用
182             //显示查询到的卡信息
183             printf("%s\t%d\t%.1f\t%.1f\t\t%d\t\t%s\n", pCard[i].aName, pCard[i].nStatus,
184                 pCard[i].fBalance, pCard[i].fTotalUse, pCard[i].nUseCount, aLastTime);
185         }
186         // 释放动态分配的内存
187         free(pCard);
188     }
189     printf("-----*****---*****\n");
190     pCard = NULL;
191 }
192 }

```

九. 重新编译连接并运行程序  
执行“查询卡”



```

D:\AMS\Debug\AccountManagement.exe

-----
请选择菜单项编号 (0~8) : 2

----***----查询卡----***----

请输入要查询的卡号(长度为1~18):1
1.精确查询, 2模糊查询 (输入1或2):2
-----*****-----查到的卡信息如下-----*****-----

卡号      状态      余额      累计金额      使用次数      上次使用时间
111        0         111.0     111.0         0             2021-02-19 20:30
123        0         123.0     123.0         0             2021-02-19 20:30
121        0         121.0     121.0         0             2021-02-19 21:00
-----*****-----*****

-----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) : 

```

✿ 十. 有兴趣的同学可以选做**附加功能**: 修改程序, 增加查询选项, 实现可查询所有卡信息, 并列表显示

## 十一. 总结

本次任务的层次结构和主要调用关系

