

一. 定义卡信息结构体类型

1. 添加 model.h 头文件

2. 在 model.h 文件中定义卡信息结构体类型，代码如下：



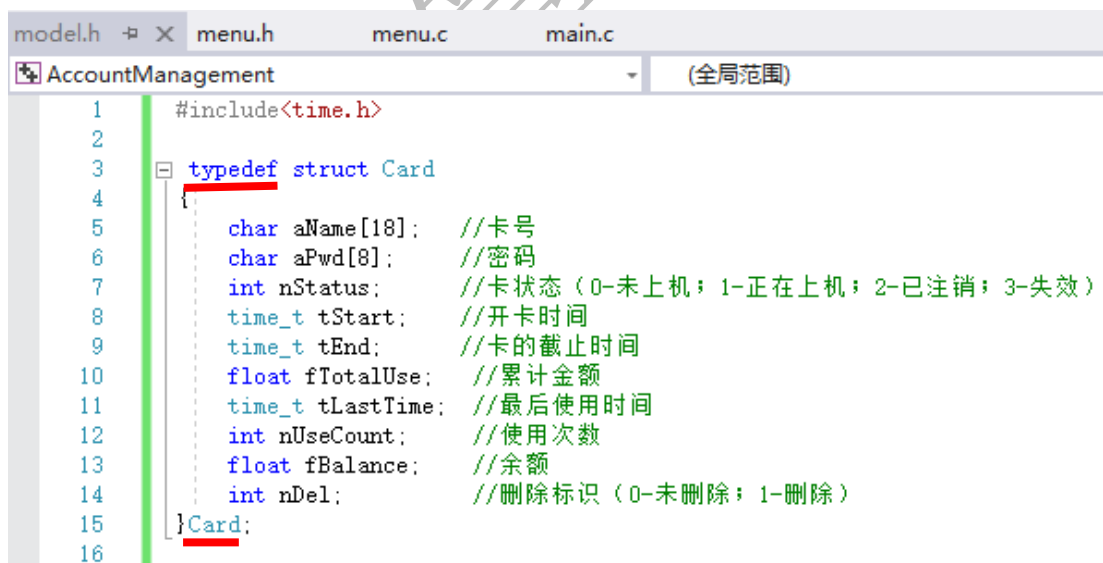
```

1  #include<time.h>
2
3  struct Card
4  {
5      char aName[18];    //卡号
6      char aPwd[8];      //密码
7      int nStatus;       //卡状态（0-未上机；1-正在上机；2-已注销；3-失效）
8      time_t tStart;     //开卡时间
9      time_t tEnd;       //卡的截止时间
10     float fTotalUse;    //累计金额
11     time_t tLastTime;   //最后使用时间
12     int nUseCount;      //使用次数
13     float fBalance;     //余额
14     int nDel;           //删除标识（0-未删除；1-删除）
15 }
16

```

其中用到 `time_t` 数据类型，它的定义在 `time.h` 头文件中，所以前面要包含这个头文件

3. 使用 typedef 指定简化的类型名



```

1  #include<time.h>
2
3  typedef struct Card
4  {
5      char aName[18];    //卡号
6      char aPwd[8];      //密码
7      int nStatus;       //卡状态（0-未上机；1-正在上机；2-已注销；3-失效）
8      time_t tStart;     //开卡时间
9      time_t tEnd;       //卡的截止时间
10     float fTotalUse;    //累计金额
11     time_t tLastTime;   //最后使用时间
12     int nUseCount;      //使用次数
13     float fBalance;     //余额
14     int nDel;           //删除标识（0-未删除；1-删除）
15 }Card;
16

```

思考：

- 1) 为什么要在新建的 `model.h` 头文件中定义卡信息结构体类型？
- 2) `time_t` 实际上是什么类型？（查看 `time.h` 头文件中的定义）
- 3) 使用 `typedef` 的作用是什么？
- 4) `typedef struct Card` 中的 `Card` 和 最后面的 `Card` 有什么不同？

二. `time.h` 头文件中与时间相关数据的处理1. `time_t` 数据类型

time.h 中对 time_t 的定义是 `typedef long time_t; /* 时间值 */`

2. time 函数，获取当前日历时间

函数原型为：`time_t time(time_t * timer);`

可以从参数 timer 返回现在的日历时间，也可以通过返回值返回现在的日历时间，即从 1970 年 1 月 1 日 0 时 0 分 0 秒到程序运行到此时的秒数数值

例：`time_t lt;`
`lt=time(NULL); // 或 time(<);`

lt 中是返回的秒数数值

3. 时间结构体类型 tm，分解时间

time.h 中对 tm 的定义如下：

```
struct tm {
    int  tm_sec; /* 秒 - 取值区间为[0,59] */
    int  tm_min; /* 分 取值区间为[0,59] */
    int  tm_hour; /* 时 取值区间为[0,23] */
    int  tm_mday; /* 一个月中的日期 取值区间为[1,31] */
    int  tm_mon; /* 月份（从一月开始，0 代表一月） 取值区间为[0,11] */
    int  tm_year; /* 年份，其值等于实际年份减去 1900 */
    int  tm_wday; /* 星期 - 取值区间为[0,6]，其中 0 代表星期天，1 代表星期一，类推 */
    int  tm_yday; /* 从每年的 1 月 1 日开始的天数 - 取值区间为[0,365]，其中 0 代表 1 月 1 日，1 代表 1 月 2 日，类推 */
    int  tm_isdst; /* 夏令时标识符，实行夏令时的时候，tm_isdst 为正。不实行夏令时的进候，tm_isdst 为 0；不了解情况时，tm_isdst() 为负。 */
};
```

4. 将日历时间转换为分解时间

函数 `gmtime` 和 `localtime` 的原型为：

`struct tm * gmtime(const time_t * timer);` //返回格林威治时间，比北京时间晚八小时
`struct tm * localtime(const time_t * timer);` //返回本地时间，即北京时间

例：`struct tm *local;`
`time_t t;`
`t=time(NULL);`
`local=localtime(&t);`

local 结构体变量中是返回的本地时间的分解形式

5. 将分解时间转换为日历时间

`mktime` 函数原型如下：

`time_t mktime(struct tm * timeptr);`

将以年、月、日、时、分、秒等分量保存的分解时间结构，转化为日历时间的秒数数值。

三. 添加卡信息到结构体变量中

1. 在 menu.c 文件中，定义添加卡函数 add（注意别忘了在 menu.h 中添加该函数声明）

2. 在 add 函数中定义一个 Card 卡信息结构体类型的变量，用来保存用户输入的卡信息（由

于用到 Card 结构体类型，前面要添加包含 model.h 头文件)

```

menu.h      menu.c*  main.c
AccountManagement (全局范围)
1  #include<stdio.h>
2
3  #include "model.h"
4
5  //[函数名]    outputMenu
6  //[功能]      输出系统菜单
7  //[参数]      void
8  //[返回值]    void
9  void outputMenu() { ... }
24
25  //[函数名]    add
26  //[功能]      添加用户卡信息到卡结构体变量，并屏幕显示
27  //[参数]      void
28  //[返回值]    void
29  void add()
30  {
31      Card card;
32
33
34

```

```

menu.h*  menu.c*  main.c
AccountManagement
1  //函数声明
2  void outputMenu();
3  void add();
4

```

3. 在 add 函数中提示用户输入卡号，密码，开卡金额，判断合格后保存到结构体变量中，并添加结构体变量中其他成员变量的值，最后以列表格式显示出来。代码如下：

```

u.h      menu.c  main.c
ccountManagement (全局范围)
1  #define _CRT_SECURE_NO_WARNINGS
2  #include<stdio.h>
3  #include<string.h>
4  #include<time.h>
5
6  #include "model.h"
7  #include "menu.h"
8
9  //[函数名]    outputMenu
10 //[功能]      输出系统菜单
11 //[参数]      void
12 //[返回值]    void
13 void outputMenu() { ... }
28

```

```

29 // [函数名]    add
30 // [功能]      添加用户卡信息到卡结构体变量，并屏幕显示
31 // [参数]      void
32 // [返回值]    void
33 void add()
34 {
35     Card card;
36     char name[30]; // 临时存放输入的用户名
37     char pwd[20];  // 临时存放输入的密码
38     struct tm* endTime; // 临时存放截止时间
39     struct tm* startTime; // 临时存放开卡时间
40
41     // 提示并接收输入的卡号
42     printf("请输入卡号 (长度为1~18): ");
43     scanf("%s", name);
44     // 判断输入的卡号长度是否符合要求
45     if (getSize(name) >= 18)
46     {
47         printf("输入的卡号长度超过最大值! \n");
48         return;
49     }
50     // 将输入的卡号保存到卡结构体中
51     strcpy(card.aName, name);
52
53     // 提示并接收输入密码
54     printf("请输入密码(长度为1~8): ");
55     scanf("%s", pwd);
56     // 判断输入的密码是否符合要求
57     if (getSize(pwd) >= 8)
58     {
59         printf("输入的密码长度超过最大值! \n");
60         return;
61     }
62     // 将输入的密码保存到卡结构体中
63     strcpy(card.aPwd, pwd);
64
65     printf("请输入开卡金额(RMB): ");
66     scanf("%f", &card.fBalance);
67
68     card.fTotalUse = card.fBalance; // 添加卡时，累计金额等于开卡金额
69     card.nDel = 0; // 删除标识
70     card.nStatus = 0; // 卡状态
71     card.nUseCount = 0; // 使用次数
72     // 开卡时间，截止时间，最后使用时间都默认为当前时间。
73     // time(NULL) 获取当前绝对时间(日历时间), 1970-01-01 00:00:00起到现在秒数
74     card.tStart = card.tEnd = card.tLastTime = time(NULL);
75
76     // 根据开卡时间，计算截止时间，每张卡的有效期为一年
77     startTime = localtime(&card.tStart); // 将开卡时间的日历时间转换为分解时间
78     endTime = localtime(&card.tEnd);
79     endTime->tm_year = startTime->tm_year + 1; // 截止时间的年数是开卡时间的年数加一
80     card.tEnd = mktime(endTime); // 将截止时间的分解时间转换为日历时间
81
82     printf("-----添加的卡信息如下-----");
83     printf("\n卡号\t密码\t状态\t开卡金额\n");
84     printf("%s\t%s\t%d\t%.1f\n\n", card.aName, card.aPwd, card.nStatus, card.fBalance);
85 }

```

其中用到 `strcpy` 函数进行字符串拷贝，需添加包含头文件 `#include <string.h>`

其中用到struct tm结构体，time函数，localtime函数，mktime函数，需添加包含头文件#include <time.h>

add函数中调用了一个新的自定义函数getSize，该函数用来判断字符串长度，函数定义在add函数后面，函数声明在menu.h中，具体操作如下。

4. 将判断字符串长度的功能封装到函数 getSize 中（在 menu.h 中添加该函数声明），在 add 函数中直接调用，文件前面添加对应包含头文件#include“menu.h”

```

u.h  menu.c  main.c
ccountManagement (全局范围)
86
87  // [函数名]  getSize
88  // [功能]    计算字符数组中字符长度
89  // [参数]    字符数组名
90  // [返回值]  字符个数
91  int getSize(const char* pString)
92  {
93      int nSize = 0;
94
95      // 计算字符串的字符个数
96      while (*(pString + nSize) != '\0')
97      {
98          nSize++;
99      }
100
101      // 返回字符个数
102      return nSize;
103  }
  
```

```

menu.h  menu.c  main.c
AccountManagement
1  // 函数声明
2  void outputMenu();
3  void add();
4  int getSize(const char* pString);
  
```

- 思考：
- 1) 字符数组 name[30]最多可存放多少个字符？能直接给字符数组变量赋值吗？
 - 2) 为什么 menu.c 前面要添加#include“menu.h”？如何修改可以不添加？
 - 3) getSize 函数的参数类型？其中的 const 的作用？add 函数调用时传递的参数类型？是否匹配？
 - 4) 当需要包含多个头文件时，头文件包含的先后顺序对程序有无影响？

四调用添加卡函数

在 main.c 文件中的 main 函数 case 1 的分支中直接调用

```

29  case 1:
30      printf("-----添加卡-----\n");
31      add();
32      break;
  
```

五. 重新编译并运行程序

```

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
9. 退出
请选择菜单项编号 (0~8) :
1
-----添加卡-----
请输入卡号 (长度为1~18) :test
请输入密码(长度为1~8): 123
请输入开卡金额(RMB): 123.123
-----添加的卡信息如下-----
卡号    密码    状态    开卡金额
test    123     0       123.1
-----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
9. 退出
请选择菜单项编号 (0~8) :

```

六. 保存卡信息到结构体数组

1. 添加 card_service.c 文件
2. 在 card_service.c 中定义全局的卡信息结构体数组 aCard[50], 用来保存多条卡信息(需包含头文件 model.h); 以及全局的整型变量 nCount, 用来记录数组中实际的卡信息条数。

```

card_service.c  menu.c  main.c
AccountManagement (全局范围)
1  #include "model.h"
2
3  Card aCard[50];    //卡结构体数组
4  int nCount = 0;    //卡结构体数组中的实际卡信息数
5

```

3. 在 card_service.c 中定义函数 addCard, 将卡信息结构体添加到数组中。代码如下:

```

card_service.c  menu.c  main.c
AccountManagement (全局范围)
1  #include<stdio.h>
2  #include <string.h>
3
4  #include "model.h"
5
6  Card aCard[50];    //卡结构体数组
7  int nCount = 0;    //卡结构体数组中的实际卡信息数
8
9  //[函数名]  addCard
10 //[功能]   添加卡信息到结构体数组
11 //[参数]   卡信息结构体
12 //[返回值]  整数1: 添加成功; 整数0: 不能添加
13 int addCard(Card crd)
14 {
15     if (nCount<50)
16     { //数组未滿, 添加一条卡信息
17         strcpy(aCard[nCount].aName, crd.aName);
18         strcpy(aCard[nCount].aPwd, crd.aPwd);
19         aCard[nCount].nStatus = crd.nStatus;
20         aCard[nCount].tStart = crd.tStart;
21         aCard[nCount].tEnd = crd.tEnd;
22         aCard[nCount].fTotalUse = crd.fTotalUse;
23         aCard[nCount].tLastTime = crd.tLastTime;
24         aCard[nCount].nUseCount = crd.nUseCount;
25         aCard[nCount].fBalance = crd.fBalance;
26         aCard[nCount].nDel = crd.nDel;
27         //计数增一
28         nCount++;
29         return 1;    //添加成功
30     }
31     else
32     {
33         printf("数组已滿, 不能添加! \n");
34         return 0;
35     }
36 }

```

其中用到 `strcpy` 函数进行字符串拷贝, 需添加包含头文件 `#include <string.h>`

思考: 1) 全局变量和局部变量的区别? 卡结构体数组 `aCard[50]` 和 实际卡信息数 `nCount` 为什么定义为全局变量? 有效范围是本文件吗? 如何扩展全局变量的使用范围?

2) `nCount=0` 为什么要初始化为零?

3) 虚线框中的语句可不可以只用一条语句 `aCard[nCount]=crd;` 代替? 所以可以直接给结构体变量赋值吗?

4. 添加 `card_service.h` 头文件, 其中添加 `addCard` 函数的声明

```

card_service.h  card_service.c  menu.c
AccountManagement
1  #include "model.h"
2
3  int addCard(Card crd);
4

```

5.在 menu.c 的 add 函数中调用 addCard 函数，文件前面要#include"card_service.h"

```

1  #define _CRT_SECURE_NO_WARNINGS
2  #include<stdio.h>
3  #include <string.h>
4  #include<time.h>
5  ...
6  #include "model.h"
7  #include "menu.h"
8  #include "card_service.h"
9
10 //函数名

```

在 add 函数最后添加代码调整如下：

```

84 printf("-----添加的卡信息如下-----");
85 printf("\n卡号\t密码\t状态\t开卡金额\n");
86 printf("%s\t%s\t%d\t%.1f\n\n", card.aName, card.aPwd, card.nStatus, card.fBalance);
87
88 //卡信息添加到结构体数组中
89 if ( FALSE == addCard(card) )
90 {
91     printf("-----*****--添加卡信息到数组失败!-----*****\n");
92 }
93 else
94 {
95     printf("-----*****--添加卡信息到数组成功!-----*****\n");
96 }
97

```

此时编译会提示两个错误：

```

输出
显示输出来源(S): 生成
1>----- 已启动全部重新生成: 项目: AccountManagement, 配置: Debug Win32 -----
1>card_service.c
1>main.c
1>menu.c
1>d:\ams\accountmanagement\model.h(4): error C2011: "Card": "struct" 类型重定义
1>d:\ams\accountmanagement\model.h(4): note: 参见 "Card" 的声明
1>d:\ams\accountmanagement\menu.c(88): error C2065: "FALSE": 未声明的标识符
1>正在生成代码...
1>已完成生成项目 "AccountManagement.vcxproj" 的操作 - 失败。
===== 全部重新生成: 成功 0 个, 失败 1 个, 跳过 0 个 =====

```

1) **Card 结构体类型重定义**，原因是包含了 2 次 model.h 中 Card 结构体的定义（一次直接在 menu.c，一次在 card_service.h 中间接包含）；

```

menu.c
1  #define _CRT_SECURE_NO_WARNINGS
2  #include<stdio.h>
3  #include <string.h>
4  #include<time.h>
5  ...
6  #include "global.h"
7  #include "model.h"
8  #include "menu.h"
9  #include "card_service.h"
10

card_service.h
1  #include "model.h"
2
3
4  int addCard(Card crd);

```


为避免头文件被重复包含，一般采用编译预处理命令处理头文件，目前的 3 个头文件修改处理如下（这种方式是依靠所定义的**宏名不重复**，来保证同一文件不被包含多次）：

```

1  #ifndef MODEL_H
2  #define MODEL_H
3
4  #include <time.h>
5
6  typedef struct Card
7  {
8      char aName[18];    //卡号
9      char aPwd[8];      //密码
10     int nStatus;        //卡状态（0-未上机；1-正在上机；2-已注销；3-失效）
11     time_t tStart;      //开卡时间
12     time_t tEnd;        //卡的截止时间
13     float fTotalUse;    //累计金额
14     time_t tLastTime;   //最后使用时间
15     int nUseCount;      //使用次数
16     float fBalance;     //余额
17     int nDel;           //删除标识（0-未删除；1-删除）
18 } Card;
19
20 #endif
21

```

宏名 **MODEL_H** 按照该头文件名 **model.h** 变化而来，各个头文件名不同所以宏名也不同

```

1  #ifndef MENU_H
2  #define MENU_H
3
4  //函数声明
5  void outputMenu();
6  void add();
7  int getSize(const char* pString);
8
9  #endif
10

```

```

1  #ifndef CARD_SERVICE_H
2  #define CARD_SERVICE_H
3
4  #include "model.h"
5
6  int addCard(Card crd);
7
8  #endif
9

```

提示：除了**#ifndef**方式，C++编译器还支持**#pragma once**方式，保证同一文件不被包含多次，只需在每个头文件开头加入该语句

这样同一个头文件可以采用上述两种方式中的任一种实现编译预处理，
方式一：

```
#ifndef MENU_H
```

```
#define MENU_H
```

```
// 一些声明语句
```

```
#endif
```

#ifndef 的方式是 C/C++语言中的宏定义，依赖于**宏名字不能冲突**。优点是**#ifndef 受语言天生的支持，不受编译器的任何限制**；缺点是如果不同头文件的**宏名不小心“撞车”**，可能会导致头文件明明存在，编译器硬说找不到声明的状况。

方式二：

```
#pragma once
```

```
// 一些声明语句
```

#pragma once 方式产生于#ifndef 之后，是编译器相关的，**由编译器提供保证**：同一个文件不会被包含多次。好处是不必再费劲想个宏名。缺点是如果某个头文件有多份拷贝，不能保证他们不被重复包含；不受一些较老版本的编译器支持，**兼容性不够好**。

这样，例如 menu.h 头文件在 VS 环境下可以表示为如下两种方式之一：

| | | |
|---|---|--|
| <pre>#ifndef MENU_H #define MENU_H void outputMenu(); #endif</pre> | 或 | <pre>#pragma once void outputMenu();</pre> |
|---|---|--|

2) **FALSE 未声明的标识符**，添加 global.h 头文件，其中添加宏定义，在 menu.c 中包含该头文件，代码如下：

```

global.h  card_service.c
AccountManagement
1 #pragma once
2
3 #define FALSE 0
4 #define TRUE 1
5

.h  card_service.c  menu.c*
AccountManagement
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3 #include <string.h>
4 #include <time.h>
5
6 #include "global.h"
7 #include "model.h"
8 #include "menu.h"
9 #include "card_service.h"
10

```

6. 修改addCard函数的返回值，1改成TRUE，0改成FALSE（前面要#include "global.h"）

```

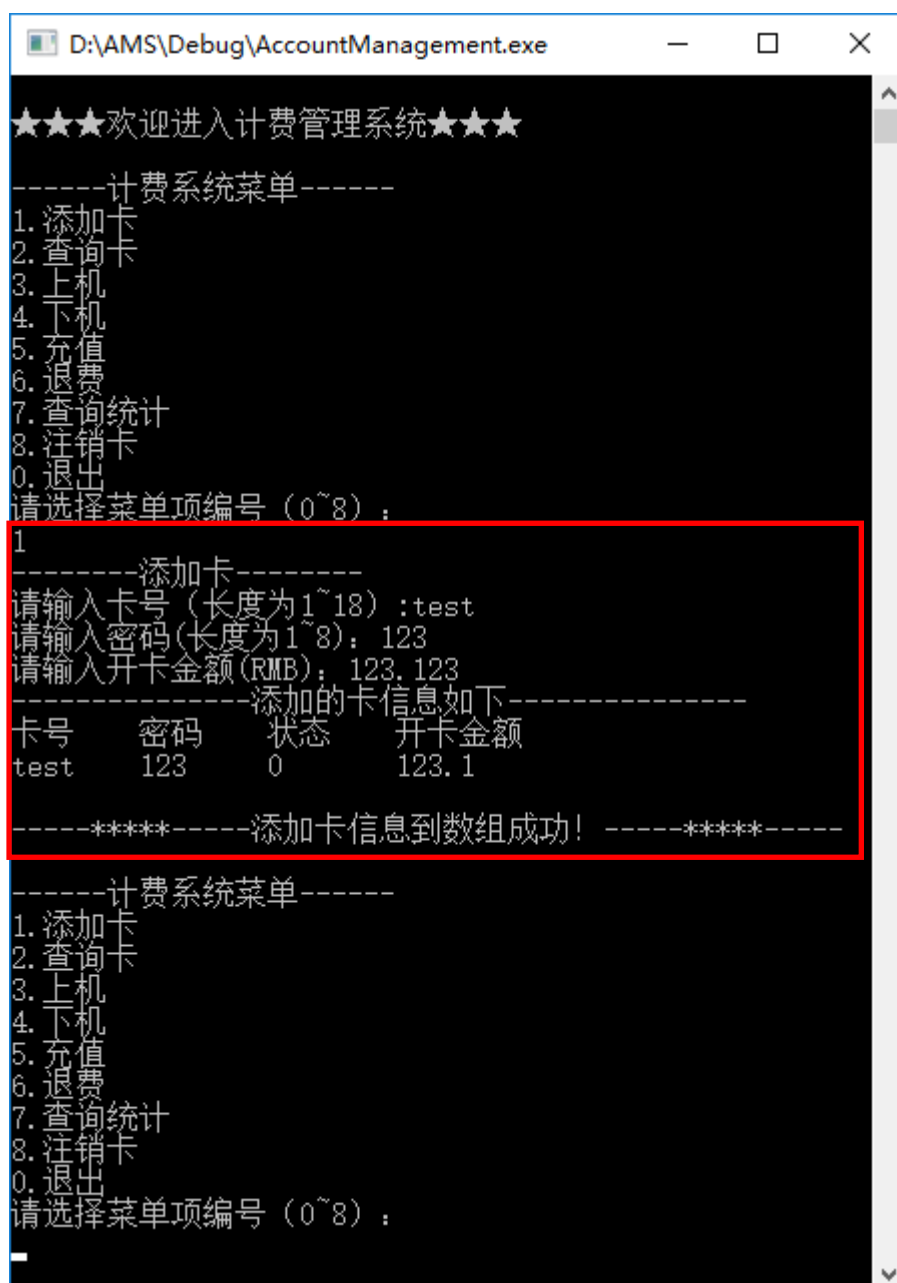
card_service.c  card_service.h  menu.h  model.h
AccountManagement (全局范围)
1 #include <stdio.h>
2 #include <string.h>
3
4 #include "global.h"
5 #include "model.h"
6
7 Card aCard[50]; //卡结构体数组
8 int nCount = 0; //卡结构体数组中的实际卡信息数
9
10 // [函数名] addCard
11 // [功能] 添加卡信息到结构体数组
12 // [参数] 卡信息结构体
13 // [返回值] 整数1: 添加成功; 整数0: 不能添加
14 int addCard(Card crd)
15 {
16     if (nCount < 50)
17     { //数组未满, 添加一条卡信息
18         aCard[nCount] = crd;
19         //计数增一
20         nCount++;
21         return TRUE; //添加成功
22     }
23     else
24     {
25         printf("数组已满, 不能添加! \n");
26         return FALSE;
27     }
28 }

```

思考: 1) 什么情况下头文件不使用编译预处理命令, 被包含多次也能编译通过? 什么情况下头文件必须使用编译预处理命令?

2) 程序中用宏 FALSE 代替数字 0, TRUE 代替数字 1 的好处?

七. 重新编译并运行程序



```
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) :
1
-----添加卡-----
请输入卡号 (长度为1~18) :test
请输入密码(长度为1~8): 123
请输入开卡金额(RMB): 123.123
-----添加的卡信息如下-----
卡号    密码    状态    开卡金额
test    123     0       123.1
-----*****-----添加卡信息到数组成功! -----*****-----

-----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) :
_
```

八. 定义查询卡函数并调用

1. 在 menu.c 中定义 query 函数, 其中提示用户输入要查询的卡号, 定义字符数组保存要查询的卡号 (同时在 menu.h 中添加该函数声明), 代码如下:

```

u.h*  menu.c*  main.c
ccountManagement (全局范围)
117  //【函数名】 query
118  //【功能】  根据输入的卡号，调用，查询是否有该卡，有的话，输出该卡信息
119  //【参数】  void
120  //【返回值】 void
121  void query()
122  {
123      char name[18] = { 0 };    //存放要查询的用户名
124
125      printf("请输入要查询的卡号(长度为1~18):");
126      scanf("%s", name);
127
128  }

```

2. 在 main.c 的 main 函数中选择菜单项 2 时，调用 query 函数，修改 main 函数如下：

```

u.h  menu.c  main.c
ccountManagement (全局范围)
33
34      case 2:
35          printf("-----查询卡-----\n");
36          query();
37          break;
38
39
40

```

九. 定义 queryCard 函数，在结构体数组中查询

1. 在 card_service.c 文件中定义 queryCard 函数，参数是用户输入的要查询的卡号地址，返回值是结构体数组中查询到的卡信息地址（同时在 card_service.h 文件中添加函数声明）；
2. queryCard 函数中，根据用户输入的卡号，在结构体数组中依次比较，第一个卡号完全相同的，即为要查找的卡信息，代码如下：

```

service.h  card_service.c  menu.h  menu.c  main.c
ccountManagement (全局范围)
31  //【函数名】 queryCard
32  //【功能】  在结构体数组中查找指定卡号的卡信息
33  //【参数】  用户输入的要查询的卡号地址
34  //【返回值】 结构体数组中查询到的卡信息地址,没有找到返回NULL
35  Card* queryCard(const char* pName)
36  {
37      Card* pCard = NULL;
38      int i;
39
40      for (i = 0; i < nCount; i++)
41      {
42          if (strcmp(aCard[i].aName, pName) == 0)
43          {
44              //在结构体数组中找到，返回卡信息地址
45              pCard = &aCard[i];
46              return pCard;
47          }
48      }
49      //没有找到，返回NULL
50      return pCard;

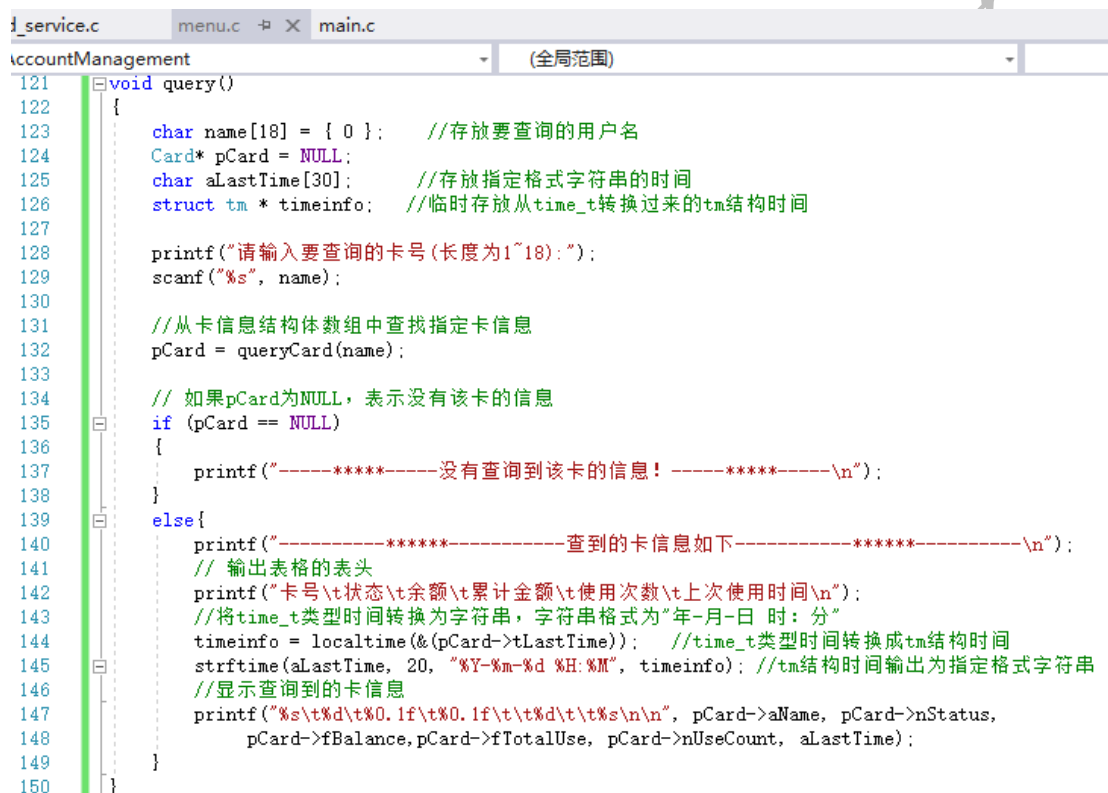
```

其中用到 `strcmp` 字符串比较函数，函数原型定义在 `string.h` 头文件，其参数是 2 个字符串，依次比较 2 个字符串中对应位置上字符的 ASCII 码，直到出现不同字符或某字符串结束，字符完全相同时返回零，不同时按第一个不同字符的 ASCII 码大小返回正值或负值

思考：函数调用时值传递和地址传递的不同？地址传递时形参和实参的结合方式？

十. 调用 `queryCard` 函数

在 `menu.c` 的 `query` 函数中，调用 `queryCard` 函数，得到查询到的卡信息，并将卡信息以列表形式显示出来（注意：结构体数组中保存的卡信息时间都是日历时间，需要转换成我们习惯的分解时间形式显示），`query` 函数代码如下：



```

121 void query()
122 {
123     char name[18] = { 0 }; //存放要查询的用户名
124     Card* pCard = NULL;
125     char aLastTime[30]; //存放指定格式字符串的时间
126     struct tm * timeinfo; //临时存放从time_t转换过来的tm结构时间
127
128     printf("请输入要查询的卡号(长度为1~18):");
129     scanf("%s", name);
130
131     //从卡信息结构体数组中查找指定卡信息
132     pCard = queryCard(name);
133
134     // 如果pCard为NULL，表示没有该卡的信息
135     if (pCard == NULL)
136     {
137         printf("-----*****--没有查询到该卡的信息!-----*****--\n");
138     }
139     else{
140         printf("-----*****-----查到的卡信息如下-----*****-----\n");
141         // 输出表格的表头
142         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");
143         //将time_t类型时间转换为字符串，字符串格式为"年-月-日 时:分"
144         timeinfo = localtime(&(pCard->tLastTime)); //time_t类型时间转换成tm结构时间
145         strftime(aLastTime, 20, "%Y-%m-%d %H:%M", timeinfo); //tm结构时间输出为指定格式字符串
146         //显示查询到的卡信息
147         printf("%s\t%d\t%.1f\t%.1f\t\t%d\t\t%s\n\n", pCard->aName, pCard->nStatus,
148             pCard->fBalance, pCard->fTotalUse, pCard->nUseCount, aLastTime);
149     }
150 }

```

其中用到 `strftime` 函数，将 `tm` 类型中的时间分量按照指定格式生成字符串，其定义在 `time.h` 头文件中，函数原型：

`strftime (char *strDest, size_t maxsize, count char *format, const struct tm *timeptr) ;`

参数的含义：strDest 转换后的字符串；

maxsize 字符串的最大长度；

format 字符串格式；

timeptr 要转换的 `tm` 结构体类型时间

这里字符串格式“`%Y-%m-%d %H:%M`”中除了分隔符号（英文状态下的短横杠 2 个，空格 1 个，冒号 1 个）外，各部分的格式含义为：

`%Y`-----带世纪部分的十制年份

`%m`-----十进制表示的月份

`%d`-----十进制表示的每月的第几天

`%H`-----24 小时制的小时

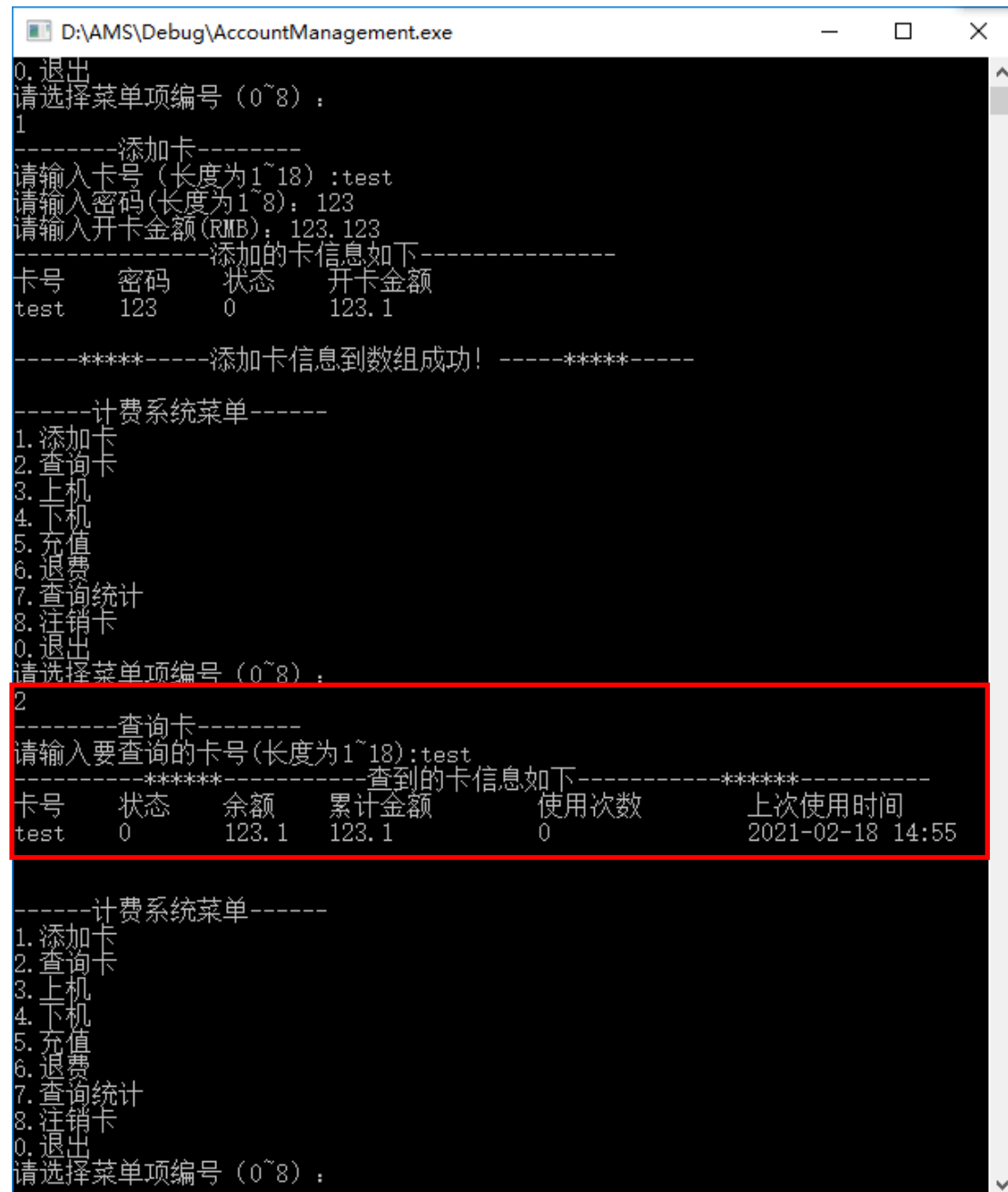
`%M`-----十时制表示的分钟数

例如：假如程序中 `timeinfo` 结构体变量中的实际时间为 2024 年 3 月 21 日 15 点 30 分，调用 `strftime` 函数后，分隔符号按原样输出，对应格式的内容按要求的字符串格式输出，这

样 aLastTime 中字符串即为 2024-03-21 15:30

十一. 重新编译并运行程序

先执行“添加卡”，再执行“查询卡”



```

D:\AMS\Debug\AccountManagement.exe
0. 退出
请选择菜单项编号 (0~8):
1
-----添加卡-----
请输入卡号 (长度为1~18): test
请输入密码 (长度为1~8): 123
请输入开卡金额 (RMB): 123.123
-----添加的卡信息如下-----
卡号    密码    状态    开卡金额
test    123     0       123.1
-----*****-----添加卡信息到数组成功! -----*****-----

-----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):
2
-----查询卡-----
请输入要查询的卡号 (长度为1~18): test
-----*****-----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
test    0       123.1    123.1       0           2021-02-18 14:55

-----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):

```

十二. 程序优化

1. 时间的存储和显示优化

时间在卡信息结构体中是 `time_t` 类型的长整型数值，输出时以分解结构的指定字符串“年-月-日 时：分”格式显示。新增 `tool.c` 文件，实现二者的装换。

1) 添加 `tool.c` 文件及 `tool.h` 文件（头文件中放函数声明）

```

tool.h*  x tool.c card_service.c menu.c
AccountManagement (全)
1  #pragma once
2  #include<time.h>
3
4  //函数声明
5  void timeToString(time_t t, char* pBuf);
6  time_t stringToTime(char* pTime);
7

```

- 2) 定义 timeToString 函数，将 time_t 格式时间，装换为“年-月-日 时：分”格式字符串，代码如下：

```

tool.c*  x tool.h card_service.c menu.c main.c
AccountManagement (全局范围)
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <time.h> // 包含时间类型头文件
3
4
5  //[[函数名] timeToString
6  //[[功能] 将time_t类型转换为字符串，字符串格式为“年-月-日 时：分”
7  //[[参数] time_t t: 需要转换的时间，char* pBuf:转换之后的字符串
8  //[[返回值] void
9  void timeToString(time_t t, char* pBuf)
10 {
11     struct tm * timeinfo;
12
13     timeinfo = localtime(&t);
14     strftime(pBuf, 20, "%Y-%m-%d %H:%M", timeinfo);
15 }

```

- 3) 修改 menu.c 的 query 函数，去掉删除线覆盖的三行，添加调用 timeToString 函数产生时间字符串，文件前面要#include “tool.h”

```

service.c menu.c*  x main.c
countManagement
1  #define _CRT_SECURE_NO_WARNINGS
2  #include<stdio.h>
3  #include <string.h>
4  #include<time.h>
5  ...
6  #include"global.h"
7  #include "model.h"
8  #include"tool.h"
9  #include "menu.h"
10 #include"card_service.h"
11

```



```

service.c  menu.c  main.c
accountManagement (全局范围)
122 void query()
123 {
124     char name[18] = { 0 }; //存放要查询的用户名
125     Card* pCard = NULL;
126     char aLastTime[30]; //存放指定格式字符串的时间
127     //struct tm * timeinfo; //临时存放从time_t转换过来的tm结构时间
128
129     printf("请输入要查询的卡号(长度为1~18):");
130     scanf("%s", name);
131
132     //从卡信息结构体数组中查找指定卡信息
133     pCard = queryCard(name);
134
135     // 如果pCard为NULL, 表示没有该卡的信息
136     if (pCard == NULL)
137     {
138         printf("-----*****-----没有查询到该卡的信息! -----*****-----\n");
139     }
140     else{
141         printf("-----*****-----查到的卡信息如下-----*****-----\n");
142         // 输出表格的表头
143         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");
144         //将time_t类型时间转换为字符串, 字符串格式为"年-月-日 时:分"
145         //timeinfo = localtime(&(pCard->tLastTime)); //time_t类型时间转换成tm结构时间
146         //strftime(aLastTime, 20, "%Y-%m-%d %H:%M", timeinfo); //tm结构时间输出为指定格式字符串
147         timeToString(pCard->tLastTime, aLastTime);
148         //显示查询到的卡信息
149         printf("%s\t%d\t%.1f\t%.1f\t\t%d\t\t%s\n\n", pCard->aName, pCard->nStatus,
150             pCard->fBalance, pCard->fTotalUse, pCard->nUseCount, aLastTime);
151     }
152 }

```

- 4) 定义stringToTime函数, 将“年-月-日 时:分”格式字符串转换为time_t格式时间, 文件前面要#include<stdio.h>, 代码如下:

```

tool.c  x card_service.c  menu.c  main.c
AccountManagement (全局范围)
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <time.h>  // 包含时间类型头文件
3  #include <stdio.h>  // 包含sscanf()函数头文件
4
5  // ...
9  void timeToString(time_t t, char* pBuf) { ... }
16
17  // [函数名] stringToTime
18  // [功能] 将格式为“年-月-日 时:分”的字符串转换为time_t类型时间
19  // [参数] char* pTime: “年-月-日 时:分”格式的字符串
20  // [返回值] time_t: 转换后的时间类型, 从1970年到该时间的秒数
21  time_t stringToTime(char* pTime)
22  {
23      struct tm tm1;
24      time_t time1;
25
26      sscanf(pTime, "%d-%d-%d %d:%d", &tm1.tm_year, &tm1.tm_mon,
27             &tm1.tm_mday, &tm1.tm_hour, &tm1.tm_min);
28
29      tm1.tm_year -= 1900;  // 年份为从1900年开始
30      tm1.tm_mon -= 1;      // 月份为0~11
31      tm1.tm_sec = 0;
32      tm1.tm_isdst = -1;
33
34      time1 = mktime(&tm1);
35
36      return time1;
37  }

```

其中使用sscanf函数（前面需#include <stdio.h>），从一个字符串中读进与指定格式相符的数据。函数原型: int sscanf(string str, string fmt, mixed var1, mixed var2 ...);

这里将字符串 pTime中对应于格式“%d-%d-%d %d:%d”的5个分量分别读入tm1结构体的tm_year, tm_mon, tm_mday, tm_hour, tm_min这5个成员变量中去;

例如: 假如pTime变量中时间为“2024-03-21 15:30”这个字符串, 调用sscanf函数后tm1结构体中各个分量为:

```

tm1.tm_year=2024
tm1.tm_mon=3
tm1.tm_mday=21
tm1.tm_hour=15
tm1.tm_min=30

```

然后还需要按照struct tm类型的时间结构体对各个分量的取值要求, 修改和设置tm_year, tm_mon, tm_sec, tm_isdst等时间分量;

最后使用mktime函数将struct tm类型的分解时间转换为time_t类型的日历时间

思考: 语句 tm1.tm_year-=1900; 是做了什么操作?

语句 tm1.tm_sec=0; 可以写成 tm1.tm_sec-=0; 吗?

2. 输出界面美化, 字符界面的字符都堆砌在一起, 不利于阅读, 输出时可适当增加一些符号, 空格及分隔线, 例如主菜单:



3. 为避免添加的卡号重复，在添加卡时，输入卡号后，先到结构体数组中查重，没有重复卡号才允许添加，否则重新输入卡号，修改 menu.c 中的 add 函数如下：

```

service.c  menu.c  main.c
countManagement  (全局范围)
39 void add()
40 {
41     Card card;
42     char name[30]; //临时存放输入的用户名
43     char pwd[20]; //临时存放输入的密码
44     struct tm* endTime; //临时存放截止时间
45     struct tm* startTime; //临时存放开卡时间
46
47     // 提示并接收输入的卡号
48     printf("请输入卡号 (长度为1~18): ");
49     scanf("%s", name);
50     // 判断输入的卡号长度是否符合要求
51     if (getSzie(name) >= 18)
52     {
53         printf("输入的卡号长度超过最大值! \n");
54         return;
55     }
56     // 判断输入的卡号是否已存在
57     if (queryCard(name) != NULL)
58     {
59         printf("输入的卡号已存在! 请重新输入! \n");
60         return;
61     }
62     // 将输入的卡号保存到卡结构体中
63     strcpy(card.aName, name);

```

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

请选择菜单项编号 (0~8) : 1
-----添加卡-----
请输入卡号 (长度为1~18) :123
请输入密码(长度为1~8): 123
请输入开卡金额(RMB): 123
-----添加的卡信息如下-----
卡号    密码    状态    开卡金额
123     123     0       123.0

-----*****-----添加卡信息到数组成功! -----*****-----

-----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) : 1
-----添加卡-----
请输入卡号 (长度为1~18) :123
输入的卡号已存在! 请重新输入!

-----计费系统菜单-----
| 1. 添加卡 |
| 2. 查询卡 |
| 3. 上机   |
| 4. 下机   |
| 5. 充值   |
```

十三.总结

本次任务的层次结构和主要调用关系

