

学号	1023008462
----	------------

# 武汉理工大学

## 《程序设计综合实验》报告

学 院 资源与环境工程学院

专 业 地理信息科学

班 级 地信 2302

姓 名 汤舜尧

指导教师 许毅

日期 2024-04-14

# 目录

<b>实验一 .....</b>	<b>3</b>
1 实验目的 .....	3
2 系统功能与描述 .....	3
3 典型算法分析(完整源码请见光盘) .....	4
4 开发难点与体会 .....	4
5 实验总结 .....	5
<b>实验二 .....</b>	<b>6</b>
1 实验目的 .....	6
2 系统功能与描述 .....	6
3 典型算法分析(完整源码请见光盘) .....	7
4 开发难点与体会 .....	8
5 实验总结 .....	8
<b>实验三 .....</b>	<b>9</b>
1 实验目的 .....	9
2 系统功能与描述 .....	9
3 典型算法分析(完整源码请见光盘) .....	9
4 开发难点与体会 .....	10
5 实验总结 .....	10
<b>实验四 .....</b>	<b>11</b>
1 实验目的 .....	11
2 系统功能与描述 .....	11
3 典型算法分析(完整源码请见光盘) .....	14
4 开发难点与体会 .....	14
5 实验总结 .....	15
<b>实验五 .....</b>	<b>16</b>
1 实验目的 .....	16
2 系统功能与描述 .....	16
3 典型算法分析(完整源码请见光盘) .....	17
4 开发难点与体会 .....	17
5 实验总结 .....	17
<b>实验六 .....</b>	<b>18</b>
1 实验目的 .....	18
2 系统功能与描述 .....	18
3 典型算法分析(完整源码请见光盘) .....	20
4 开发难点与体会 .....	20

5 实验总结 .....	21
--------------	----

## **实验七 .....22**

1 实验目的 .....	22
2 系统功能与描述 .....	22
3 典型算法分析(完整源码请见光盘).....	24
4 开发难点与体会 .....	24
5 实验总结 .....	24

## **《程序设计综合实验》成绩评定表 ..... 26**

# 实验一

## 1 实验目的

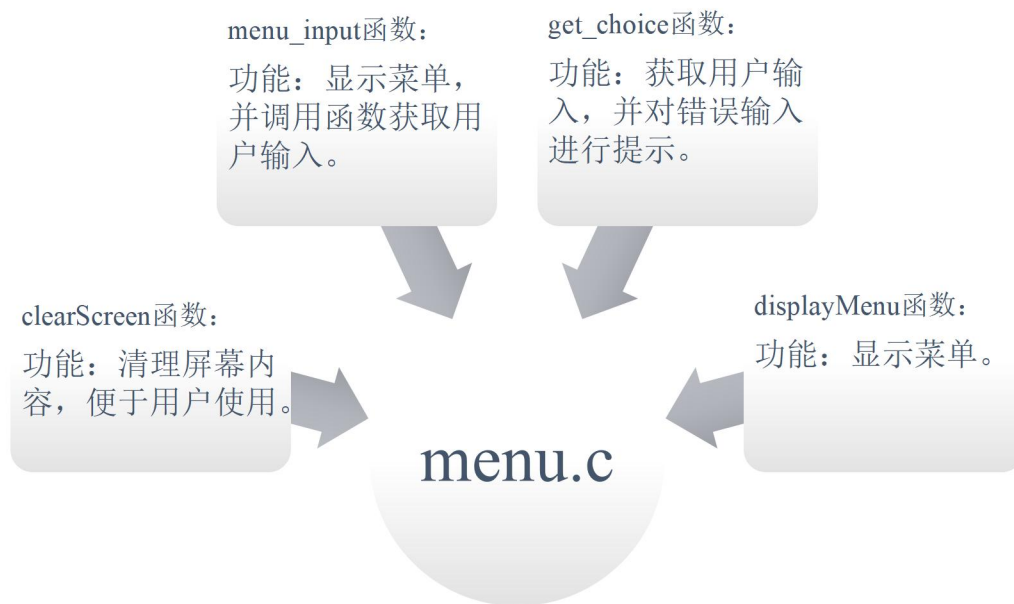
通过表示层函数，构建整体程序框架，明确计费管理系统的需求分析。将命令行模式菜单封装为函数，设置合理接口并使用。

## 2 系统功能与描述

1. 程序由三部分文件构成：menu.h，menu.c 与 main.c 文件。
2. 程序主要功能：接收用户输入的数字（0~8），并进行下一步功能实现；对于用户的各种不同的错误输入进行判断并提示重新输入。

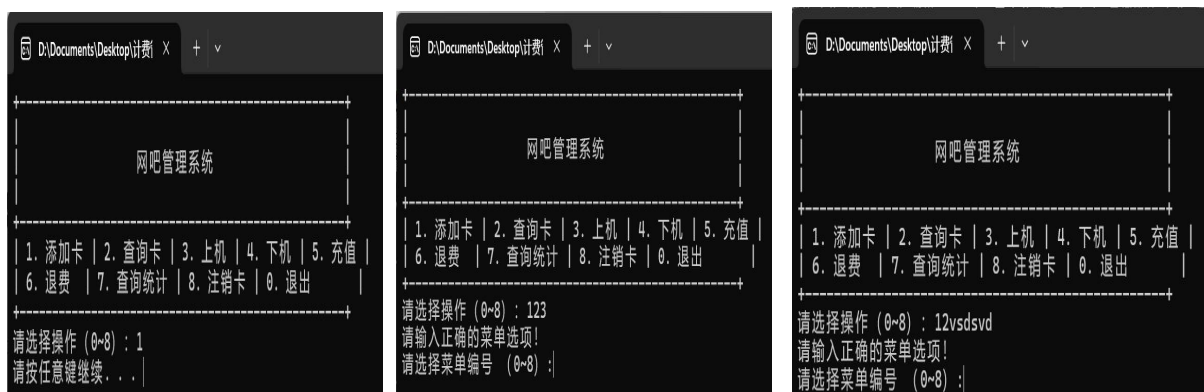
3. 文件主要功能：

(1) :



(2) : main 函数通过调用 menu.c 文件中的函数，构建出程序的整体框架。

4. 程序运行截图：



### 3 典型算法分析(完整源码请见光盘)

1. main 函数的对于用户输入的处理: 将 menu\_input 函数的返回值作为 while 的循环条件, 若输入值是 0 时便退出程序, 否则将有其他内容输出。
2. 对数字和字符混合输入时的处理: 将用户输入的第一位数字存储进 ch 变量中, 如果用户输入了其他的数字或者字符, 用 getchar() 来接收第二位输入, 如果不是回车('\n'), 就用一个 while 与 getchar() 的循环将后续输入全部清除。
3. 使用 system 系列函数: 使用 "cls" 函数在用户操作完一次后进行清屏, 保证界面简洁明了; 使用 "pause" 函数防止 "cls" 函数清除过快导致用户无法看到程序的反馈信息。

### 4 开发难点与体会

1. 最棘手的一点无疑是对数字和字符的混合输入, 为了保证代码的健壮性, 对于用户可能输入的各种情况都应该做出相应的判断。

程序要求用户输入 0~8 的数字, 最优先考虑的一定是对仅有数字情况的处理; 这也相对好判断一些, 我们使用 scanf("%d",&ch) 接收用户所输入的数字, 并储存在 int 型变量 ch 中, 再对 ch 判断是否是  $ch \geq 0 \& \& ch \leq 8$  就可以。

再复杂一些, 用户的输入可能仅有字符, 对此我们将 scanf 的返回值作为循环的判断条件。(scanf 的返回值为: 如果用户的输入类型与 scanf 函数中的格式转换符所期待的输入类型不一样时, 便会返回 0; 输入类型相同时, scanf 的返回值则为变量个数。)输入项仅有字符时, scanf 便会返回 0, 这样用户输入的第一项就检查完毕, 进入错误提示的循环; 如果后续仍有输入, 则通过 while(getchar() != '\n') 来获取后一项的内容, 如果不是换行符('\n'), 则继续循环(后续我们将用 clear 函数表示这种功能, clear 函数原型如下图)。

```
void clear(void)
{
    while (getchar() != '\n')
        continue;
}
```

clear 函数原型

最复杂的情况便是数字和字符混合输入（数字优先于字符，例如：123abc形式，字符优先与数字的情况与第二种相同）和输入浮点数的情况了。由于浮点数与数字和字符混合输入情况相同，在此仅讨论数字和字符混合输入的情况。由于 scanf 函数仅读取输入的第一位数字，而剩下的输入流会保留下来，于是我们可以用 getchar()来接收输入的第二项并进行判断，若 getchar()接收到的字符不为换行符('\n')时，则进入错误提示的循环，在循环中将后续的输入用 clear()函数处理，给予相应的错误提示。

2. 使用 system 系列的函数，使得交互界面简洁明了而不影响用户的使用体验。”cls”函数将上次操作显示的部分清除，为了防止清屏过快，利用”pause”函数作出适当停顿。

自此，第一部分的难点攻克，既保证了系统的健壮性，又提供的灵活的人机交互，给予用户的良好体验。

## 5 实验总结

1. 在本次实验中，我深刻理解了构建一个健壮的系统是一件需要考虑周全的事情，对于无法预估的用户可能带来的各种输入，我们需要通过不同的条件限制来确保系统不会因此崩溃，因此锻炼了我对全局的掌控能力。

2. 在人机交互的设计中，我学会了通过 system 等各种函数进行界面美观设计，给用户带来良好的使用体验。

# 实验二

## 1 实验目的

通过迭代式开发，使用结构体数组存储用户输入的卡信息，实现添加卡和查询卡操作。

## 2 系统功能与描述

1. 文件构成：表示层：main.c，menu.c；

业务逻辑层：card.c；

2. 功能描述：

(1)：添加卡：主界面键入 1 进行添加卡操作。用户输入的卡号仅由数字和英文大小写字符构成，不符合要求的将会提示重新输入。若卡号与已有的卡号相同，则提示重新输入。卡结构体数组容量为 100，若卡信息多于 100，则提示系统已满，立即退出程序。输入密码与充值金额后显示出当前卡信息。添加卡时可多次添加卡，若想退出添加卡系统，输入空行（即回车键）即可。

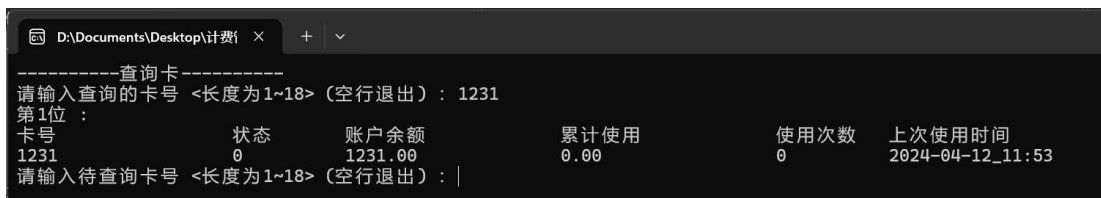
```
D:\Documents\Desktop\计费 X + v
-----添加卡-----
请输入卡号 <长度为1~18> (空行退出): 123
请输入密码 <长度为1~8> :123
请输入开卡金额 <RMB> : 123
----添加卡信息如下! ----
卡号      密码      状态      开卡金额
123      123      0      123.00
-----添加卡-----
请输入卡号 <长度为1~18> (空行退出): |
```

添加卡：正常卡添加

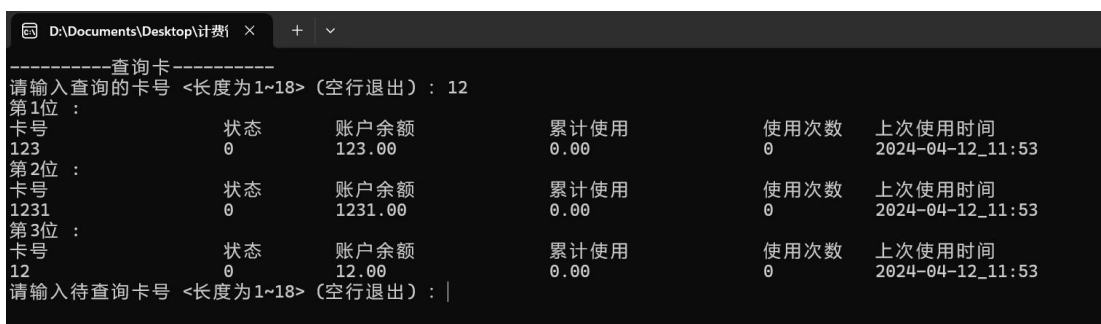
```
D:\Documents\Desktop\计费 X + v
-----添加卡-----
请输入卡号 <长度为1~18> (空行退出): 123
请输入密码 <长度为1~8> :123
请输入开卡金额 <RMB> : 123
----添加卡信息如下! ----
卡号      密码      状态      开卡金额
123      123      0      123.00
-----添加卡-----
请输入卡号 <长度为1~18> (空行退出): 123
输入卡号重复, 请重新输入卡号
请输入卡号 <长度为1~18> (空行退出): |
```

添加卡：重复卡添加

(2)：查询卡：主界面键入 2 实现查询卡。查询卡同时支持精准查询和模糊查询（即不输入完整的卡号也能显示出卡信息的查询方式）。用户输入完整卡号即可实现精准查询；输入不完整卡号将显示出所有符合要求的卡号。



查询卡：精准查询截图



查询卡：模糊查询截图

### 3 典型算法分析(完整源码请见光盘)

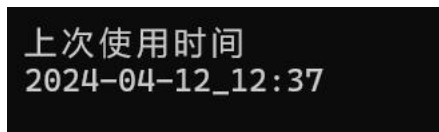
1. **s\_gets 函数**：使用 s\_gets 函数来接收用户输入的一行内容。s\_gets 函数接收的第一个参数是字符串指针，用来存储用户输入的内容；第二个参数是规定的长度（无规定长度则默认为最大长度 MAX=100）。使用 fgets 函数接收用户输入的内容，fgets 函数的接收长度为 MAX。定义 char 类型指针 ret\_val 接收 fgets 的返回值，若成功接收到用户的输入，则返回用户输入字符串的存储地址；若未成功则返回 NULL。又由于 fgets 函数会将输入的换行符（‘\n’）一同接收，于是可以用 strchr 函数定位到换行符的位置，定义 char 类型的指针 find 接收 strchr 的返回值，将\*find 的值由‘\n’改为‘\0’，就完成了一行字符串的读取。最后将 ret\_val 作为 s\_gets 函数的返回值，表明是否接收到数据。

2. **fFind 函数**：使用 fFind 函数实现模糊查询与精准查询。fFind 函数接收的参数为输入的字符串内容与结构体数组。使用 strncmp 函数来进行内容的比较。strncmp 函数的第一个、第二个参数是用户输入的卡号与结构体数组中保存的卡



号，第三个参数为用户输入卡号的长度 len。由于 strcmp 函数比较部分为前 len 个字符，每次出现符合条件的信息就输出一次卡的信息。因此可完成模糊查询与精准查询的功能。

**3. 使用时间函数 localtime 与 strftime 对时间进行表示：**首先使用 time() 获得 1970 至今的秒数，再通过 localtime 转换为当地的时间，最后使用 strftime 函数将时间格式转换为“年-月-日\_时\_分”的格式。



上次使用时间  
2024-04-12\_12:37

对时间的处理

## 4 开发难点与体会

本次实验的难点在于结构体数组的信息的添加和查询。初始化数组时可以将循环次数设置为数组的最大值，再将除了卡号、密码和卡内金额以外的值初始化，当用户进行添加卡时只需更改卡号、密码与金额即可。查询时将循环次数设置为数组最大值，一一进行比对直到找到符合条件的信息。

## 5 实验总结

本次实验中我学会了如何使用结构体数组存储用户信息，同时深化了我对用户输入正确处理的认知，提高了我的综合能力。

# 实验三

## 1 实验目的

通过迭代式开发，将结构体数组中的信息保存入文件中，实现数据的长期存储。

## 2 系统功能与描述

1. 文件存储层：card\_file.c, carrd\_file.h;
2. 将文件信息读入结构体数组和将结构体数组信息存入文件的函数：InitialiseItem 函数，FileAdd 函数和 ChangeFile 函数；
3. 卡文件函数将链表中的信息以“卡号、密码、卡内余额、总消费金额、卡状态、使用次数、删除状态、开始时间、结束时间、开始时间戳、结束时间戳、上下机间隔时间”这 12 个内容存入文件中，保证了下次打开程序时卡信息不会消失。

```
123##123##123.000000##0.000000##0##0##0##2024-04-12_16:28##2024-04-12_16:28##0##0##0
```

卡文件：文件中保存的卡信息

## 3 典型算法分析(完整源码请见光盘)

1. 将文件信息读取进结构体数组中：在 InitialiseItem 函数初始化结构体数组时，首先把结构体数组中卡号的第一位设置为'\0'，即初始化结构体数组，再以文本文件格式打开文件，再使用 fscanf 函数依次读取文件中卡信息的 12 个内容，并把 fscanf 的返回值与 12 作比较。若小于 12，则说明文件为空或者文件读取完毕或者读取某个值时出现错误，方便以后的检查，此时跳过读取文件，得到空结构体数组；把 fscanf 读取的卡信息存储进 Item 类型的变量 temp 中，然后与添加卡的步骤相同，创建一个新节点，再将节点作为结构体数组一项的值，依次读取文件中的内容，这样就完成了结构体的初始化。

将文件信息在初始化中就存入结构体数组，有效避免了操作时从外存读取数据过慢的问题。

**2. 更改文件信息：**在实现注销卡功能时，将会对文件中的卡信息进行更改。首先将结构体数组中的卡信息更改，再以”w”格式打开文件，重新将更改信息后的结构体数组存入文件中，避免了繁琐的查找对应信息过程。

**存入文件信息：**在添加卡结束后，得到一个装载有卡信息的结构体数组，此时将所有卡信息以文本文件格式写入文件即可。

## 4 开发难点与体会

**对特定格式卡文件的读取：**由于实验要求卡文件要以“卡号##密码”格式存储，因此不能简单使用 `fscanf` 并在格式转换符的中间添加“##”来读取（例如 `fscanf("%s##%s##.....")`），对于简单的整数和浮点数确实可以这样读取，但是字符串就会将文件中的”##”读取入变量中。于是我通过查询资料，了解到格式说明符：“%[^#]”，其中“%[^#]”是一个扫描集合的引导序列，他告诉 `fscanf` 读入并存储一系列字符，直到遇到扫描集合外的字符。在这种情况下，`[^#]`指定了扫描集合，即读取除#之外的字符。故此即可使用 `fscanf` 与 `fprintf` 向文件输出指定格式的内容并成功读取。

```
while (fscanf(fp, "%[^#]##%[^#]##%lf##%lf##%[^#]##%lu##%lu##%[^#]##%[^#]##%lu##%lu##%lu\n",
    pitem[i].ID, pitem[i].Password, &pitem[i].Amount, &pitem[i].AllConsumption,
    pitem[i].status, &pitem[i].count, &pitem[i].nDel, pitem[i].begin, pitem[i].end, &pitem[i].start,
    &pitem[i].finish, &pitem[i].duration) == 12) {
```

卡文件：fscanf 的具体实现

## 5 实验总结

通过本次实验，我掌握了文本文件读取的基本形式，了解了内存与外存的读取差异，进一步掌握了对数据结构的理解。

# 实验四

## 1 实验目的

通过迭代式开发，利用链表结构来代替结构体数组进行信息存储，并且在链表的基础上进行添加卡，查询卡，注销卡操作。

## 2 系统功能与描述

1. 文件构成: card.c , card.h , card\_file.c ,card\_file.h , menu.c ,menu.h 与 main.c。

其中 card.h 中储存了卡信息结构体和链表结构体。

```
typedef struct CARD {  
    char ID[MAX];           //卡号  
    char Password[MAX];     //密码  
    double Amount;          //卡内余额  
    char status[MAX];       //卡状态  
    char begin[MAX];        //上机时间  
    char end[MAX];          //下机时间  
    time_t start;           //上机时间戳  
    time_t finish;         //下机时间戳  
    time_t duration;        //上下机经过时间  
    unsigned int count;     //使用次数  
    double AllConsumption;  //总消费金额  
    double Consumption;    //单次消费金额  
    unsigned int nDel;      //注销状态  
} Item;
```

card.h: 卡信息结构体

```
typedef struct NODE {  
    Item item;  
    struct NODE* next;  
} Node;  
typedef Node* List;
```

card.h: 链表结构体

2. 程序主要功能: 添加卡，查询卡，注销卡。

(1) : 添加卡:

首先读取用户输入的卡号，由于卡号要求为 1~18 位，于是超出 18 位的卡号将会出现错误提示；在此程序的卡号，我设置为只能由数字与英文字符两种类型的字符组成，其余字符不能出现；如果该卡号被别人注册过且未被注销，将返回错误提示并要求重新输入，如果该卡号是注销过的卡，则会出现提示是否重新激活此卡，若同意，则该卡重新变成空卡，并清除之前的使用信息，否则重新输入卡号；添加卡时可多次添加卡，若想退出添加卡系统，输入空行（即回车键）即可。

```
D:\Documents\Desktop\计费1  X  +  v
-----添加卡-----
请输入卡号 <长度为1~18> (空行退出): 123/*-
请输入数字或字母组合!
请输入卡号 <长度为1~18> (空行退出): |
```

添加卡：卡号仅允许数字和英文字母运行截图

```
D:\Documents\Desktop\网吧1  X  +  v
-----添加卡-----
请输入卡号 <长度为1~18> (空行退出): 123
该卡为注销卡，请问是否需要重新启用？
输入 1 启用,输入 0 退出:1
请输入密码 <长度为1~8> :123
请输入开卡金额 <RMB> : 123
----添加卡信息如下! ----
卡号      密码      状态      开卡金额
123      123      0      123.00
```

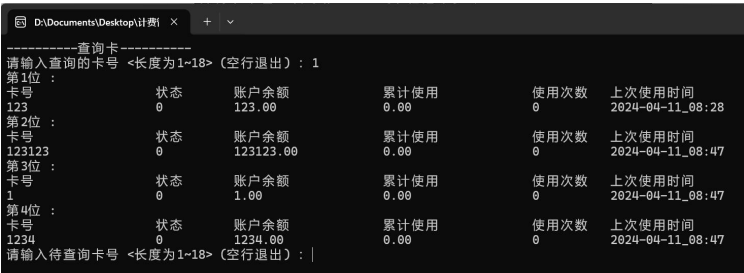
添加卡：注册已被注销卡运行截图

```
D:\Documents\Desktop\计费1  X  +  v
-----添加卡-----
请输入卡号 <长度为1~18> (空行退出): 123
请输入密码 <长度为1~8> :123
请输入开卡金额 <RMB> : 123
----添加卡信息如下! ----
卡号      密码      状态      开卡金额
123      123      0      123.00
-----添加卡-----
请输入卡号 <长度为1~18> (空行退出): 456
请输入密码 <长度为1~8> :456
请输入开卡金额 <RMB> : 456
----添加卡信息如下! ----
卡号      密码      状态      开卡金额
456      456      0      456.00
-----添加卡-----
请输入卡号 <长度为1~18> (空行退出): 789
请输入密码 <长度为1~8> :789
请输入开卡金额 <RMB> : 789
----添加卡信息如下! ----
卡号      密码      状态      开卡金额
789      789      0      789.00
-----添加卡-----
请输入卡号 <长度为1~18> (空行退出): |
```

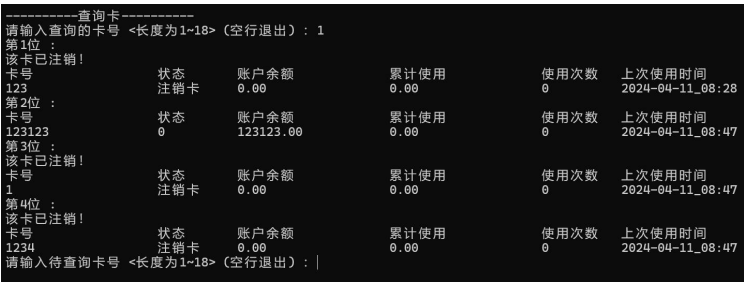
添加卡：多次输入的运行截图

(2) ： 查询卡：

查询卡方式为模糊查询，即不输入完整的卡号也能显示出卡信息的查询方式。用户输入完整卡号即可实现精准查询，显示出该卡的信息，若输入不完整的卡号，将会输出多个满足条件的卡信息。若该卡为注销卡，则该卡状态为“注销卡”。



查询卡：模糊查询运行截图



查询卡：注销卡查询截图

(3) ： 注销卡：

用户输入卡号与密码即可实现注销卡，并将卡内余额退款。注销后的卡，卡状态变为“注销卡”，在查询中依然能查到，直到该卡号被重新注册后，先前的卡信息被覆盖。



注销卡：注销卡运行截图

### 3 典型算法分析(完整源码请见光盘)

**1. 链表数据结构的构建：**构建链表我分为一下几步：初始化链表，创建节点，将节点添加在链表中，这样就构建出一个链表。我没有使用全局变量来创建链表，为了防止全局变量占用内存过多，而是将链表地址作为参数传递给函数。

**(1)：初始化链表：**将链表的头节点设置为 NULL；

**(2)：创建节点：**首先创建 Item 类型的变量 temp，并通过 EnterInformation 函数将一张卡信息保存在 temp 中，再创建 Node \*类型的指针变量 pnew,为 pnew 分配内存后，使用 CopyToNode 函数将 pnew->item 赋值为 temp，最后将 pnew->next 设置为 NULL，就构建好了一个的节点。

**(3)：将节点添加到链表中：**将创建好的节点 pnew 赋给链表尾节点的 next 指针，重复上述操作，就能得到装载信息的链表。

**(4)：构建函数 ListIsEmpty 和 ListIsFull 来判断链表是否为空或为满，**保证代码的健壮性。

**2. 模糊查询：**使用 strcmp 函数，将用户的输入 temp.ID 与链表中各个卡的卡号做对比，比较的长度设置为输入的长度（即 strlen(temp.ID)）。考虑到满足要求的信息可能有多个，于是将输出卡信息的函数 showcase 置于循环中，即每查询到一个符合要求的卡信息就输出一次，这样极大程度上简化了查询操作。

**3. 可重复输入卡号/查询/删除：**以添加卡为例，在添加时可多次添加卡，便于用户操作，如果想要退出，则输入空行（即键入换行符）即可。利用 s\_gets 函数读取用户输入并存入 temp.ID 中，若 temp.ID[0]为'\0'时，便结束输入。

### 4 开发难点与体会

**1.** 本次实验一大难点在于如何构建链表。之前我也想过每次在链表末端分配空间再填充信息，但是这实现起来十分麻烦，还会时不时 Segmentation fault。于是我想到了将卡信息保存进节点，再将节点附着在链表中，就很好解决了这个问题。

**2.** 对于删除链表中的卡信息也困扰了我一段时间，因为删除链表中的项需要

找到目标节点的上一个节点。如果设置前置节点指针 `prev` 的话，操作不当就会出现 `prev` 指向空指针的情况，十分麻烦且出错概率很大。因此我想到先用 `Find` 函数找到目标节点，再通过循环将辅助指针 `current` 指向目标节点的前置节点，就可以避免双指针的复杂操作。

## 5 实验总结

通过本次实验，我成功构建了链表，并完成了向链表中添加项、查询链表包含的信息以及删除链表中的项等操作，强化了我对数据结构的认知和运用。



# 实验五

## 1 实验目的

通过迭代式开发，读取卡文件中的信息并存入链表中，同时将新添加的卡信息存入文件中，实现数据的动态存储。

## 2 系统功能与描述

- 1. 存储卡信息文件 Card\_file.txt;
- 1. 将文件信息读入链表和将链表信息存入文件的函数：InitialiseList 函数，FileAdd 函数和 ChangeFile 函数；
- 2. 卡文件函数将链表中的信息以“卡号、密码、卡内余额、总消费金额、卡状态、使用次数、删除状态、开始时间、结束时间、开始时间戳、结束时间戳、上下机间隔时间”这 12 个内容存入文件中，保证了下次打开程序时卡信息不会消失。

```
123##123##123.000000##0.000000##0##0##0##2024-04-11_11:18##2024-04-11_11:18##0##0##0
456##456##456.000000##0.000000##0##0##0##2024-04-11_11:35##2024-04-11_11:35##0##0##0
789##789##78.000000##0.000000##0##0##0##2024-04-11_11:35##2024-04-11_11:35##0##0##0
```

卡文件：数据存储方式

```
C:\Users\TangSY\source\repo  X + v
-----查询卡-----
请输入查询的卡号 <长度为1~18> (空行退出) : 1
第1位 :
卡号      状态      账户余额      累计使用      使用次数      上次使用时间
123      0      123.00      0.00      0      2024-04-11_11:18
请输入待查询卡号 <长度为1~18> (空行退出) : 4
第1位 :
卡号      状态      账户余额      累计使用      使用次数      上次使用时间
456      0      456.00      0.00      0      2024-04-11_11:35
请输入待查询卡号 <长度为1~18> (空行退出) : 7
第1位 :
卡号      状态      账户余额      累计使用      使用次数      上次使用时间
789      0      78.00      0.00      0      2024-04-11_11:35
请输入待查询卡号 <长度为1~18> (空行退出) : |
```

卡文件：第二次打开程序的查询结果

### 3 典型算法分析(完整源码请见光盘)

**1. 将文件信息读取进链表中：**在 `InitialiseList` 函数初始化链表时，首先把链表设置为空，以文本文件格式打开文件，再使用 `fscanf` 函数依次读取文件中卡信息的 12 个内容，并把 `fscanf` 的返回值与 12 作比较。若小于 12，则说明文件为空或者文件读取完毕或者读取某个值时出现错误，方便以后的检查，此时跳过；把 `fscanf` 读取的卡信息存储进 `Item` 类型的变量 `temp` 中，然后与添加卡的步骤相同，创建一个新节点，再将节点附着在链表上，依次读取文件中的内容并构建起链表。这样就完成了链表的初始化。

将文件信息在初始化中就存入链表，有效避免了操作时从外存读取数据过慢的问题。

**2. 更改文件信息：**在实现注销卡功能时，将会对文件中的卡信息进行更改。首先将链表中的卡信息更改，再以“w”格式打开文件，重新将更改信息后的链表存入文件中，避免了繁琐的查找对应信息过程。

**存入文件信息：**在添加卡结束后，得到一条装载有卡信息的链表，此时将所有卡信息以文本文件格式写入文件即可。

### 4 开发难点与体会

`fscanf` 与 `fprintf` 读取和写入特定格式的文件。对于两组数据间的##，整数和浮点数可以直接在 `fscanf` 中添加##，但是对于字符串函数，需要使用`%[^#]`来代替`%s`。

### 5 实验总结

经过本次实验，我理解掌握了如何对特定格式的卡文件进行读取；了解到外存与内存读取速度的差异，对数据结构的理解进一步加深。

# 实验六

## 1 实验目的

通过迭代式开发，为计费管理系统添加上机和下机的功能，并在文件中实时更新卡信息。

## 2 系统功能与描述

1. 文件构成：表示层：main.c, menu.c;

业务逻辑层：card.c, billing.c;

数据存储层：card\_file.c, billing\_file.c;

2. 系统功能：

(1) ：上机：主界面键入 3 实现上机操作。在用户输入卡号与密码后，将会显示出用户卡号与上机时间，并增加一次使用次数；注销卡无法完成上机操作；用户在线则无法上机。



```
D:\Documents\Desktop\计费管理 > 3
-----上机-----
请输入上机卡号 <长度为1~18> (空行退出) : 123
请输入密码 <长度为1~8> : 123
-----上机成功-----
卡号      上机时间
123      2024-04-11_18:52
请输入上机卡号 <长度为1~18> (空行退出) : |
```

上机：正常卡上机截图



```
D:\Documents\Desktop\计费管理 > 3
-----上机-----
请输入上机卡号 <长度为1~18> (空行退出) : 456
请输入密码 <长度为1~8> : 456
该用户已注销!
请输入上机卡号 <长度为1~18> (空行退出) : |
```

上机：注销卡上机截图

```
D:\Documents\Desktop\计费1 x + v
-----上机-----
请输入上机卡号 <长度为1~18> (空行退出) : 123
请输入密码 <长度为1~8> :123
该用户上机中!
请输入上机卡号 <长度为1~18> (空行退出) : |
```

上机：在线卡无法上机

(2) ：下机：主界面键入 4 实现下机操作。用户输入卡号密码后，系统将会根据上下机间隔的时间确定使用时长并结算费用，从卡内扣除相应的金额，根据使用时长的不同将给予相应的优惠策略：使用时长小于两小时，每小时定价 8 元；大于两小时小于五小时时，每小时定价 6 元；大于 5 小时时，每小时定价 4 元，不足一小时的按一小时算。若卡内余额不足以支付，则提示充值费用。注销卡无法进行下机操作。用户离线则无法下机。每一次上下机都会产生一份消费文件存储在 Billing\_file.txt 文件中。

```
D:\Documents\Desktop\计费1 x + v
-----下机-----
请输入下机卡号 <长度为1~18> (空行退出) : 123
请输入密码 <长度为1~8> :123
-----下机成功-----
卡号      状态      账户余额      累计使用      使用次数      上次使用时间
123       0          115.00        8.00          1             2024-04-11_19:56
请输入下机卡号 <长度为1~18> (空行退出) : |
```

下机：正常卡下机截图

```
D:\Documents\Desktop\计费1 x + v
-----下机-----
请输入下机卡号 <长度为1~18> (空行退出) : 123
请输入密码 <长度为1~8> :123
该用户已注销!
请输入下机卡号 <长度为1~18> (空行退出) : |
```

下机：注销卡下机截图

```
D:\Documents\Desktop\计费1 x + v
-----下机-----
请输入下机卡号 <长度为1~18> (空行退出) : 123
请输入密码 <长度为1~8> :123
该用户不在线!
请输入下机卡号 <长度为1~18> (空行退出) : |
```

下机：离线卡无法下机

```
-----下机-----  
请输入下机卡号 <长度为1~18> (空行退出) : 1  
请输入密码 <长度为1~8> :1  
余额不足, 请立即充值!
```

下机: 卡余额不足

```
123##2024-04-11_20:04##2024-04-11_20:04##8.00##115.00
```

下机: Billing\_file.txt 文件截图

### 3 典型算法分析(完整源码请见光盘)

1. **卡消费信息文件读取方式:** 将文件以文本文件格式 “r+”打开, 由于我们采用 `fscanf` 一次性读取一条信息, 直到##停止读取, 存储进相应的变量中再读取下一个值。
2. **卡消费信息文件保存方式:** 下机添加消费信息时, 将文件以”a+”格式打开, 这样就可以在文件末尾追加卡信息。再使用 `fprintf` 函数将文件以特定的格式保存进文件中; 重新注册已注销卡时, 将会更改卡的消费信息, 此时将卡文件以”w”格式打开, 将更改后的卡消费信息链表重新写入文件。
3. **使用 Find\_ID\_Password 函数:** 将用户输入的卡号与密码与卡链表中的信息作比较, 返回目标节点, 上机下机操作直接在此节点上操作, 提高了运行效率。

### 4 开发难点与体会

本次实验中困扰我很久时间的是对上下机时间间隔的处理。由于存储上机时间的时间戳 `start` 和下机时间戳 `finfish` 均为 `time_t` 类型, 所对应的格式转换符为 `%lu`, (`unsigned int` 类型), 因此二者相减所得到的值 `duration` 也应是 `time_t` 类型, 但是由于之前对于时间类型数据认识不够, 使用 `%lf` 接收 `duration` 时经常出现极大的数据, 与实际情况完全不符, 经过数次调试后, 我终于明白应该使用 `%lu` 来接收 `duration`。但是带来了新的问题, 由于 `time()`

函数返回值是自 1970 年以来的秒数,而 duration 直接除以 3600 得到小时数 (duration/3600) 也出现了与实际不符合的数据。于是我想到了强制转换,将(finish-start)的值强制转换为 int 类型后再除以 3600,不足一小时按一小时算,再存入 duration 中,这样 duration 就成为了计算消费金额时使用的小时数。

```
if ((pt->item.finish - pt->item.start) % 3600 == 0)
    pt->item.duration = ((int)(pt->item.finish - pt->item.start)) / 3600;
else
    pt->item.duration = 1 + ((int)(pt->item.finish - pt->item.start)) / 3600;
```

上下机时间间隔处理: 不足一小时按一小时算

## 5 实验总结

通过本次实验,我对时间数据类型以及时间函数的理解进一步加深,并掌握了时间函数 time, localtime 和 strftime 的基础用法;同时进一步熟练运用使用链表来存储信息的方法和文件的读取操作,完善了整个计费系统,提高了我的综合能力。

# 实验七

## 1 实验目的

通过迭代式开发，实现计费系统的充值，退费以及查询卡消费信息的内容，基本完成计费管理系统的全部功能。

## 2 系统功能与描述

1. 文件构成：表示层：main.c，menu.c；

业务逻辑层：card.c，billing.c；

数据存储层：card\_file.c，billing\_file.c；

2. 系统功能：

(1)：充值：主界面键入 5 进入充值界面。用户输入卡号与密码后，将会提示输入充值金额，充值金额应为大于等于 0 的整数或浮点数；系统将会处理错误输入并提示重新输入。充值完毕后将会显示当前卡的余额。注销卡无法完成充值操作。

充值：正常卡充值截图

```
请输入充值卡号 <长度为1~18> (空行退出) : 123
请输入密码 <长度为1~8> :123
请输入充值金额:sadad
请输入正确的充值金额!
请输入充值金额:^A
```

充值：输入错误金额运行截图

```
D:\Documents\Desktop\网吧j x + v
-----充值-----
请输入充值卡号 <长度为1~18> (空行退出) : 123
请输入密码 <长度为1~8> :123
该用户已注销!
请输入充值卡号 <长度为1~18> (空行退出) : |
```

充值：注销卡无法充值

(2)：退费：主界面键入6进入退费界面。用户输入账号密码后，将会提示输入退费金额，退费金额为大于等于0的整数或浮点数，且退费金额不能大于卡内金额，系统将会处理错误输入并提示重新输入。退费后将显示卡内余额。注销卡无法完成退费。

```
D:\Documents\Desktop\网吧j x + v
-----退费-----
请输入退费卡号 <长度为1~18> (空行退出) : 123
请输入密码 <长度为1~8> :123
请输入退费金额:123
-----退费完成!-----
卡号      余额
123      0.00
请输入退费卡号 <长度为1~18> (空行退出) : 123
请输入密码 <长度为1~8> :123
请输入退费金额:123
余额不足，退费失败!
请输入退费卡号 <长度为1~18> (空行退出) : |
```

退费：正常卡退费以及退费失败的运行截图

(3)：查询统计：主界面键入7进入查询统计界面。查询统计功能为精准查询，用户输入完整的卡号和密码后，将会显示出该用户卡消费的记录（每一次上机下机记录一次），若无卡信息，则提示未查询到此人信息。注销卡也可进行卡消费记录查询。



您的消费记录如下：				
卡号	上机时间	下机时间	消费	账户余额
123	2024-04-12_13:42	2024-04-12_13:42	8.00	115.00
卡号	上机时间	下机时间	消费	账户余额
123	2024-04-12_13:43	2024-04-12_13:43	8.00	107.00
请输入待查询卡号 <长度为1~18> (空行退出)：				

查询统计：卡消费记录查询

### 3 典型算法分析(完整源码请见光盘)

1. **Recharge 函数**：首先输入充值余额，检查输入是否有效（非负数且是数字）；若输入有效，金额将添加进账户余额；系统将显示充值后的余额，以确认充值成功。

2. **Refund 函数**：用户输入希望退还的金额，系统将检查输入是否有效（非负数且不大于卡内余额）；若输入有效，该金额将从卡内余额扣除；系统将显示退费后的余额，以确认退费成功。

3. **文件的读取 Fread 与 Fwrite 函数**：本次实验文件由文本文件读取改为二进制文件读取，在需要处理大量数据时效率更高，并且有效避免了由于更新卡信息后信息变长导致信息读取出错的问题。使用 fread 函数与 fwrite 函数，可以一次读取一整条结构体信息，大大提高了编程效率。

### 4 开发难点与体会

1. **输入验证和错误处理**：在充值和退费功能中，必须确保用户输入的金额是有效和合理的（如非负数）。如果输入不当，系统需要能够恰当响应而不产生不可预期的行为，因此要实现严格的输入验证逻辑，确保所有输入都在预期范围内。对于不合规的输入，提供清晰的错误消息，并引导用户重新输入。

2. **确保充值操作的原子性**：即要么完全成功，要么完全不影响账户余额，防止在充值过程中出现部分更新的问题。在本系统中采用事务管理机制，即在充值操作中，如果更新过程中出现错误，可以回滚到操作前的状态，确保数据的一致性。

## 5 实验总结

通过本次实验，我完成了计费管理系统的基本内容，掌握了如何使用二进制文件来缩短文件读取时间，以及对于金额管理的操作，大大提高了我对程序开发的认知。

# 《程序设计综合实验》成绩评定表

班级：

姓名：

学号：

序号	评分项目	满分	实得分
1	学习态度认真、遵守纪律	10	
2	迭代开发进度合理，提交结果正确	40	
3	代码规范、注释清晰、可读性好	10	
4	软件功能完善、运行正确	20	
5	验收情况良好	10	
6	报告规范、描述清晰准确	10	
		总得分	
评语：			

指导教师签名：