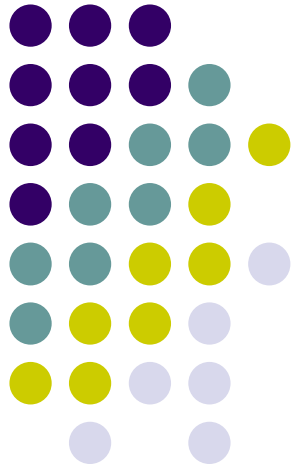
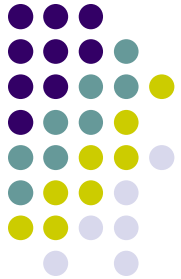
程序设计综合实验

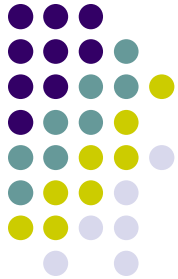


武汉理工大学计算机科学与技术学院



实验三：计费管理系统的文件存储管理

本次实验课授课内容



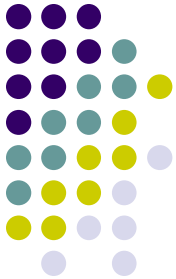
1. 基本概念：文件、文件结构体、文件类型指针
2. 文件的打开、关闭
3. 文件的常见读写方法
4. 计费管理系统的文件存储管理
5. 实验三的内容与要求

文件

- C文件概述
  - **文件**：
    - 是存储在外部介质上数据的集合

  - 文件分类
    - 按存储介质：
      - **普通文件**：存储介质文件（磁盘、磁带等）
      - 设备文件：非存储介质（键盘、显示器、打印机等）
    - 按数据的组织形式：
      - **文本文件**： ASCII文件，每个字节存放一个字符的ASCII码
      - **二进制文件**：数据按其在内存中的存储形式原样存放

文件结构体FILE

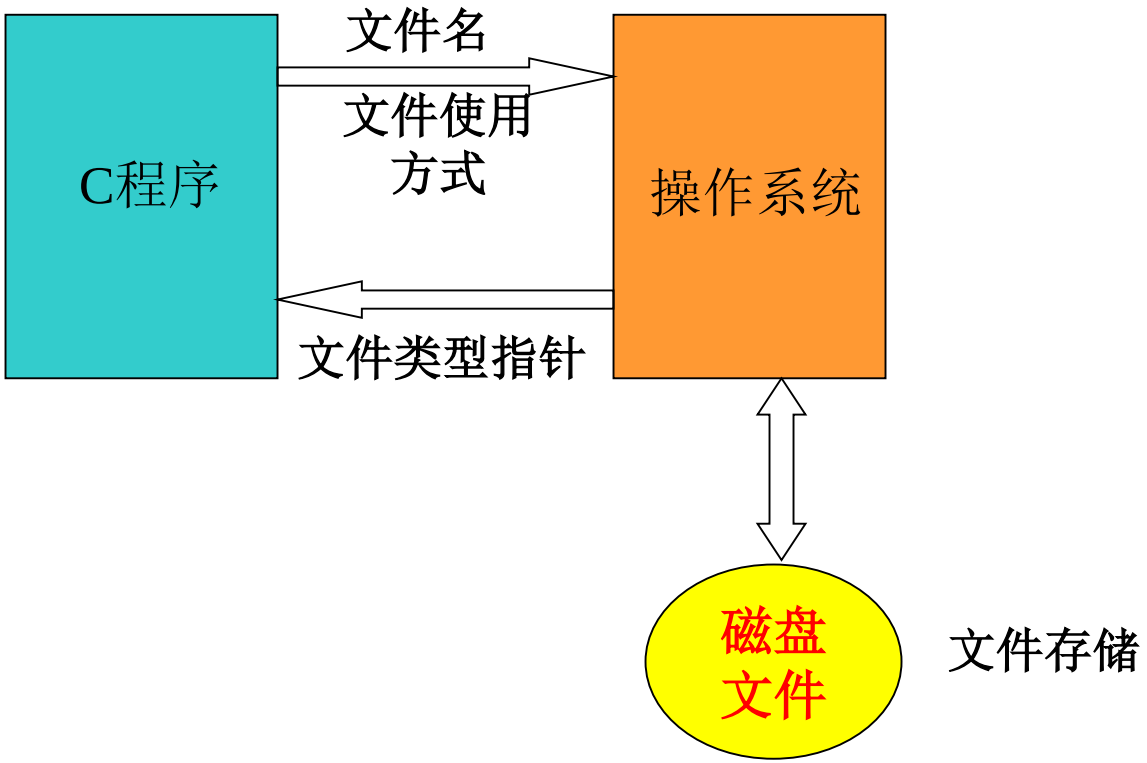


- 缓冲文件系统为每个正使用的文件在内存开辟文件信息区
- 文件信息用系统定义的名为FILE的结构体描述
- FILE定义在stdio.h中

```
typedef struct{
    short           level;      /* 缓冲区使用量  */
    unsigned        flags;      /* 文件状态标志 */
    char            fd;         /* 文件描述符 */
    short           bsize;      /* 缓冲区大小 */
    unsigned char   *buffer;    /* 文件缓冲区的首地址 */
    unsigned char   *curp;      /* 指向文件缓冲区的工作指针 */
    unsigned char   hold;       /* 其他信息 */
    unsigned        istemp;
    short           token;
} FILE;
```

文件类型指针

- 指针变量说明： FILE *fp;
- 用法：
  - 文件打开时，系统自动建立文件结构体，并把指向它的指针返回来，程序通过这个指针获得文件信息，访问文件。
  - 文件关闭后，它的文件结构体被释放。



- 文件的打开与关闭
  - C文件操作用库函数实现，包含在**stdio.h**
  - 系统自动打开和关闭三个标准文件：
    - 标准输入------键盘　　stdin
    - 标准输出------显示器　　stdout
    - 标准出错输出-----显示器　　stderr
  - 文件使用方式:

    打开文件 --> 文件读/写 --> 关闭文件

文件打开

打开文件fopen

- 函数原型： FILE *fopen(char *name, char *mode)

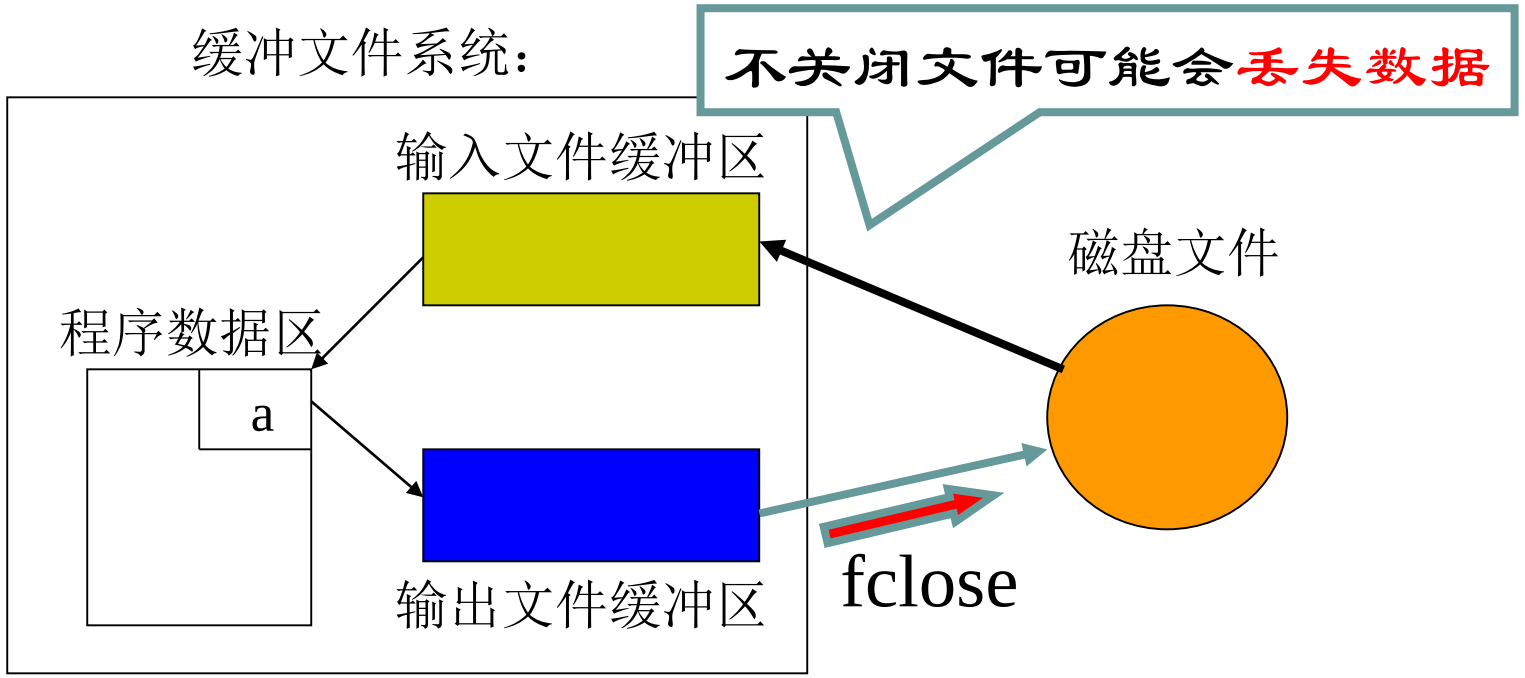
使用文件方式

要打开的文件名

- 功能：按指定方式打开文件
- 返值：
  - 正常打开，为指向文件结构体的指针
  - 打开失败，为NULL

文件使用方式	含义
“r/rb” (只读)	为输入打开一个文本/二进制文件
“w/wb” (只写)	为输出打开或建立一个文本/二进制文件
“a/ab” (追加)	向文本/二进制文件尾追加数据
“r+/rb+” (读写)	为读/写打开一个文本/二进制文件
“w+/wb+” (读写)	为读/写建立一个文本/二进制文件
“a+/ab+” (读写)	为读/写打开或建立一个文本/二进制文件

- 文件关闭**fclose**
  - 使文件指针变量与文件“脱钩”
  - 释放文件结构体和文件指针
- 函数原型：**int fclose(FILE *fp)**
  - 功能：关闭fp指向的文件
  - 返值：正常关闭为**0**；出错时为**非0**



文件的读写

– 字符串I/O:

- 函数原型:

```
char  *fgets(char  *s, int n, FILE  *fp)
int   fputs(char  *s, FILE  *fp)
```

- 功能：从fp指向的文件读/写一个字符串
  - **fgets**从**fp**所指文件读**n-1**个字符送入**s**指向的内存区,并在最后加一个'\0'
    (若读入n-1个字符前遇**换行符**'**\n**'或文件尾（**EOF**）即结束)
  - **fputs**把**s**指向的字符串写入**fp**指向的文件

- 返值:
  - **fgets**正常时返回读取字符串的首地址；出错或文件尾，返回**NULL**
  - **fputs**正常时返回写入的最后一个字符；出错为**EOF**

例　从键盘读入字符串存入文件，再从文件读回显示

```
#include<stdio.h>
main()
{   FILE  *fp;
    char  string[81];

    if((fp=fopen("file.txt", "w")) = =NULL)
    {   printf("cann't open file");exit(0); }
    while(strlen(gets(string))>0)
    {   fputs (string, fp);
        fputs ("\n", fp);
    }
    fclose(fp);
    if((fp=fopen("file.txt", "r")) = =NULL)
    {   printf("cann't open file");exit(0); }
    while (fgets(string, 81, fp) !=NULL)
      fputs(string, stdout);
    fclose(fp);
}
```

从stdin流读字符串函数gets()
直至接受到换行符或EOF时停止

判断字符串长度函数strlen ()

- 数据块I/O：fread与fwrite
  - 函数原型：

```
size_t fread(void *buffer, size_t size, size_t count, FILE *fp)
size_t fwrite(void *buffer, size_t size, size_t count, FILE *fp)
```

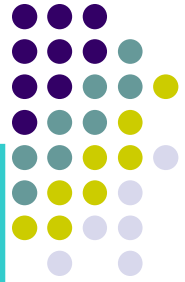
- 功能：读/写数据块
- 返值：**成功**返回读/写的**块数**；**出错**或文件尾返回**0**
- 说明：
  - typedef unsigned size_t;
  - **buffer:** 指向要输入/输出数据块的**首地址指针**
  - **size**: 每个要读/写的数据块的**大小**（字节数）
  - **count**: 要读/写的数据块的**个数**
  - **fp**: 要读/写的**文件指针**

从文件fp中读/写count个大小为size的元素到buffer指向的数据块

- fread与fwrite 一般用于**二进制文件**的输入/输出

例 从键盘输入4个学生数据，把他们转存到磁盘文件中去

```
#include <stdio.h>
#define SIZE 4
struct student_type
{    char name[10];
     int num;
     int age;
     char addr[15];
}stud[SIZE];

main()
{
     int i;
     for(i=0;i<SIZE;i++)
               scanf("%s%d%d%s", stud[i].name, &stud[i].num,
                                   &stud[i].age, stud[i].addr);
     save();
     display();
}
```



例 从键盘输入4个学生数据，把他们转存到磁盘文件中去

```
void save()
{   FILE *fp;
    int  i;

    if((fp=fopen("d:\\fengyi\\exe\\stu_dat","wb")) = =NULL)
    {    printf("cannot open file\n");
             return;
    }

    for(i=0;i<SIZE;i++)
       if (fwrite (&stud[i], sizeof(struct student_type),1,fp)!=1)
                printf("file write error\n");
    fclose(fp);
}
```

例 从键盘输入4个学生数据，把他们转存到磁盘文件中去

```
void display()
{   FILE *fp;
    int  i;

    if((fp=fopen("d:\\fengyi\\exe\\stu_dat","rb")) = =NULL)
    {    printf("cannot open file\n");
             return;
    }

    for(i=0;i<SIZE;i++)
    {  fread(&stud[i],sizeof(struct student_type),1,fp);
       printf("%-10s %4d %4d %-15s\n", stud[i].name,
              stud[i].num, stud[i].age, stud[i].addr);
    }
    fclose(fp);
}
```

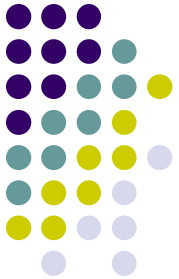
- 格式化I/O:
  - 函数原型：

int fscanf (FILE *fp, const char *format [,address,…])

int fprintf (FILE *fp, const char *format [,argument,…])

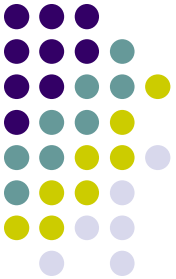
- 功能：按格式对文件进行I/O操作
- 返值：**成功**返回I/O的**个数**；**出错**或文件尾返回**EOF**

例 从键盘按格式输入数据存到磁盘文件中去



```
#include <stdio.h>
main()
{ char s[80],c[80];
  int a,b;
  FILE *fp;

  if((fp=fopen("test","w"))==NULL)
  {   puts("can't open file");   exit() ;   }

  fscanf(stdin,"%s %d",s,&a);  /*read from keyboard*/
  fprintf(fp,"%s %d",s,a);        /*write to file*/
  fclose(fp);

  if((fp=fopen("test","r"))==NULL)
  {   puts("can't open file"); exit();   }

  fscanf(fp,"%s %d",c,&b);        /*read from file*/
  fprintf(stdout,"%s %d",c,b);    /*print to screen*/
  fclose(fp);
}
```

实验三：计费管理系统的文件存储管理



实验目的

(1) 了解文件、文件类型指针 、 FILE 类型结构体

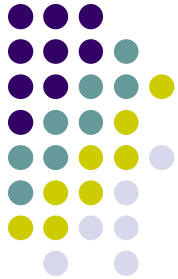
(2) 学习如何对文件**打开**、**读取**、**关闭**

(3) 学习字符串的**操作**、**分隔**、**类型转换**

(4) 学习如何将卡信息保存到文本文件

(5) 实现卡管理：**添加卡**、**查询卡**（文件）

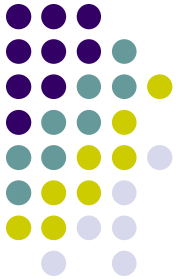
实验三：计费管理系统的文件存储管理



实验任务

1. 将卡信息保存到文件中
2. 从文件中读取卡信息
3. 获取卡信息文件中卡信息数量
4. 更新卡信息文件中的一条卡信息
5. 根据卡号判断卡信息文件中是否存在该项卡信息

添加卡



1、添加卡（文件）

在搭建程序框架的迭代中，根据三层结构，对程序结构进行划分，分为了表示层、业务逻辑层、数据访问层。

本次迭代，应用数据访问层，实现将添加的卡信息，保存到指定路径下的文本文件中，以便退出程序后，释放内存中的信息。但是再启动程序时，可以从文本文件中读取信息。

添加卡信息时，将添加的卡信息保存到工程目录下的card.txt文件的末尾。

保存失败，则提示添加卡信息失败；保存成功，则提示添加卡信息成功。

添加卡


卡管理(文件)
添加卡(文件)
将输入的卡信息保存到文本文件中
文件操作
文件
文件名
C语言写文件步骤
打开文件fopen()
文件类型指针
FILE类型结构体
打开方式
将数据写进文件
fprintf()
关闭文件fclose()
查询卡(文件)

处理过程

(1) 获取保存卡信息的文件路径，即工程目录下的card.txt文件。

(2) 获取添加的卡信息。

文件存储路径设计

将保存卡信息的文本文件命名为card，将card.txt文件保存到工程目录下的data文件夹下。

(3) 将每个卡信息组装成一条字符串，一张卡的每个信息间用“##”分隔。

(4) 将保存的卡信息的字符串写到工程目录下的card.txt文件末尾。

学生信息在文件中的保存格式如下：

卡号##密码##状态##开卡时间##截止时间##累积金额##最后使用时间##使用次数##当前余额##删除标识

图13-6 学生信息在文件中的保存格式

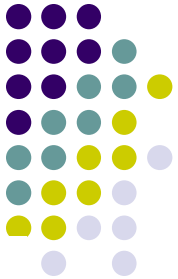
fprintf

输出格式： 按设计思路中的存储格式输出，即

“%s##%s##%d##%s##%s##%.1f##%s##%d##%.1f##%d\n”

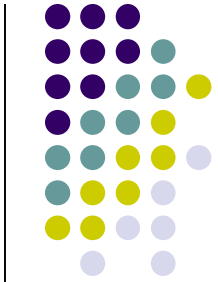
sscanf(buffer, "%[^#]##%[^#]##%d##%[^#]##%[^#]##%lf##%[^#]##%d##%lf##%d", c->aName, c->aPwd, &c->nStatus, tStart, tEnd, &c->fTotalUse, tLast, &c->nUseCount, &c->fBalance, &c->nDel)

添加卡



```
----------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)：50
------添加的卡信息如下------
卡号        密码        状态        开卡金额
test        123         0           50.0
----------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）：12345678912345678
输入的菜单编号错误!
```

查询卡



2、查询卡（文件）

上次迭代中，实现了将卡信息保存到工程目录下的card.txt文件中。本次迭代，将从该文件中读取并解析卡信息，然后显示到界面中。

从工程目录下的card.txt文件中，读取并解析卡信息，将卡信息显示到界面中。

读取失败，则提示“没有该卡的信息！”。

读取成功，则在界面输出卡号、状态、余额、累计使用、使用次数、上次使用时间，一共六个信息，按列输出。

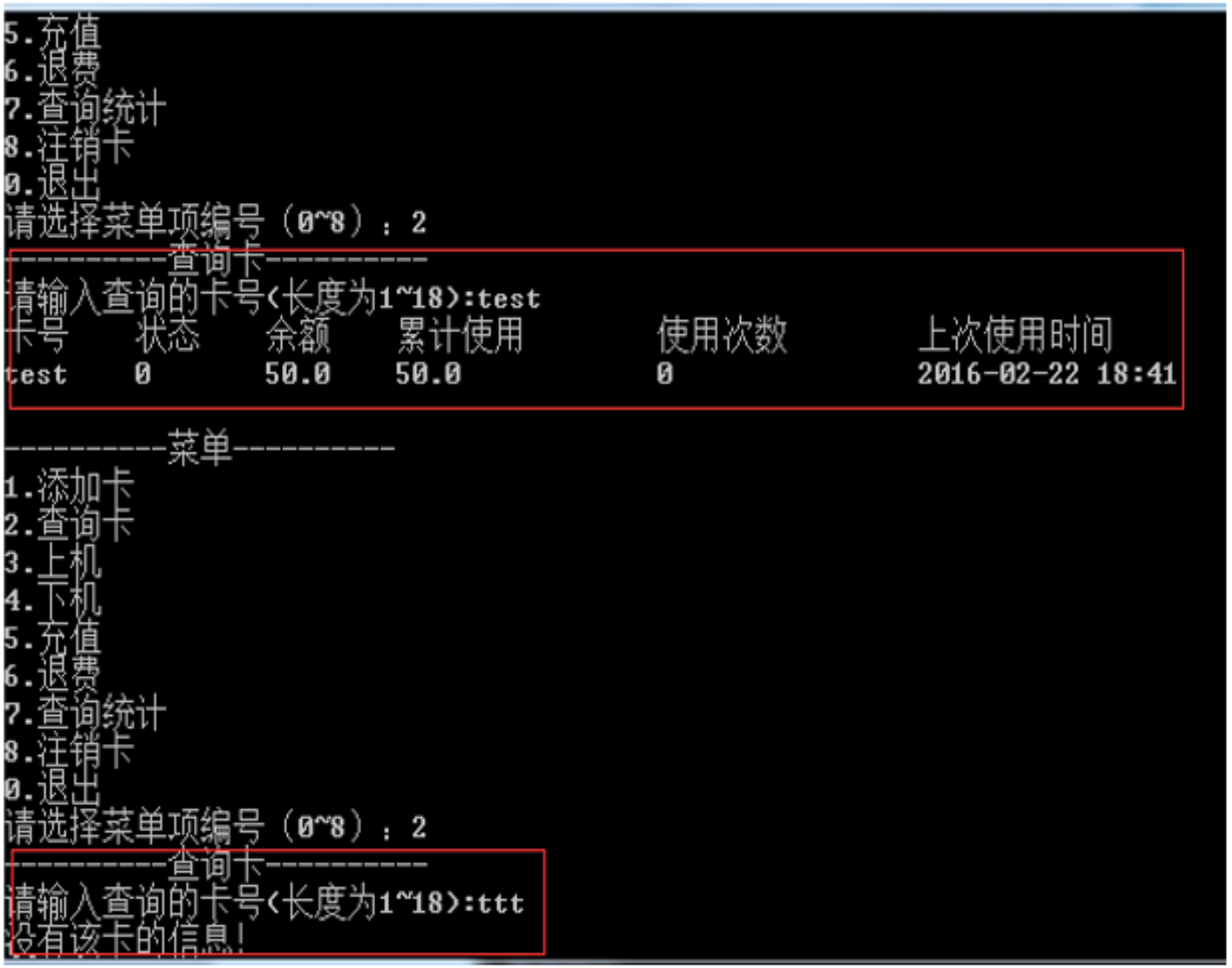
处理过程

(1) 获取工程目录下的card.txt文件的路径。

(2) 逐行读取该文件中的卡信息并解析。

(3) 将解析结果保存在内存中，在界面上显示读取出来的卡信息。

查询卡


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----------
请输入查询的卡号(长度为1~18):test
卡号 状态 余额 累计使用 使用次数 上次使用时间
test 0 50.0 50.0 0 2016-02-22 18:41
----------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----------
请输入查询的卡号(长度为1~18):ttt
没有该卡的信息!

实验三要点：实现卡管理：**添加卡、查询卡**（文件）

（在实验2（结构体数组）的基础上完成）

- 添加卡时，先将新的上机卡信息添加到结构体变量中，再将上机卡信息写入数据文件card.txt /card.dat中，实现卡信息的永久保存。

- 读取卡信息时，先从数据文件card.txt / card.dat中读入所有上机卡信息，并将上机卡信息存储在一个结构体变量/结构体数组中，然后搜索相关卡信息；

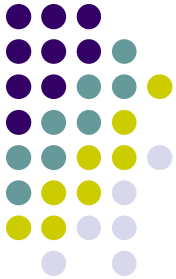
- **文件的两种形式：**
  - 文本文件：保存的是可读的字符（ASCII码）　　card.txt
  - 二进制文件：保存的是二进制数据　　card.dat
- **用户在存取文件时应该始终以相同的方式打开**
- **文件读写时，怎样写进去，就应该以相同的方式读出来**

实验三：计费管理系统的文件存储管理

项目名称	实验内容	交付物
计费管理系统的文件存储管理	（1）将用户输入的卡信息，按照指定的格式保存到卡信息文件中。 (2) 查询卡信息时，将卡信息文件中对应的卡信息显示在控制台中。	运行截图 + 文件截图

测试用例：分别输入菜单编号1、2；进而添加卡（密码正常/超长/重复卡号）；查询卡（存在/不存在）

实验三：计费管理系统的文件存储管理



测试用例：

一、输入菜单编号1，然后

1. 输入正常卡号、密码、金额
2. 输入超长卡号，正常密码、金额
3. 输入超长密码，正常卡号、金额
4. 输入非法金额，正常卡号、密码
5. 输入系统中已存在的卡号，正常密码、金额

二、输入菜单编号2，然后

1. 输入系统中存在的正常卡号
2. 输入超长卡号
3. 输入系统中不存在的正常卡号

验收要求：本次实验的正常运行需包含上述测试用例以及卡信息文本文件截图。