

第 12 章 文件

12.1 教学要点

本章主要介绍文件的基本概念，文本文件和二进制文件类型，C 语言文件存储的基本原理：文件缓冲系统的使用，文件结构及文件操作指针的定义与使用，自定义类型，文件操作处理基本步骤和过程，常用的文件操作函数等知识。其中，要重点介绍的是文件的概念、数据存储文件缓冲系统原理、文件操作过程和常用文件操作函数，要求理解与掌握文件操作的含义，尤其是文件缓冲系统的工作原理和文件指针的使用，也是本章的难点内容。

12.1 节通过给出一个示例程序“素数文件”，引出文件的基本概念和主要知识点。教师在讲授时，可以先介绍和分析该示例程序的总体框架结构，结合程序把相关知识点引出，不需要对源代码进行详细分析，可以让学生课后自己细读，或尝试调试运行。同时，应重点介绍文件的概念，文本文件和二进制文件常用文件类型，讲清楚缓冲文件系统及工作原理，最后介绍文件结构与文件类型指针的定义，顺便引出自定义文件类型的定义与使用方法。

12.2 节通过示例程序“用户信息的加密和校验”的介绍，讲授文件操作的实现有关知识点，包括文件操作的基本过程步骤、实现方法及文件操作处理函数。教师讲授时，可以对背景含义做简要介绍，对示例程序的文件操作实现过程要做详细介绍，让学生能够理解程序本身的功能含义，重点强调操作实现过程，逐一对程序涉及的知识点展开讲解。学生需要理解所有 C 语言文件操作程序的实现都要具备的四个基本步骤：1)定义文件指针;2)打开文件;3)读写文件;4)关闭文件。本节重点要介绍的知识点首先是文件的打开与关闭，然后是文件的操作的实现，都是通过系统提供的文件操作函数来实现的，主要的文件操作函数包括：字符读写函数：`fgetc()`和`fputc()`；字符串读写函数：`fputs()`和`fgets()`；格式化读写函数：`fscanf()`和`fprint()`；二进制读写函数：`fread()`和`fwrite()`。通过这些函数实现对文本文件和二进制文件的操作处理。此外，还要介绍一下有关读写文件的位置与状态控制有关的函数。

12.3 节介绍了关于文件应用的综合示例程序，用身边实际例子说明了文件综合应用的强大的数据管理功能。要求学生能够学会使用文件功能可以实现比较大的管理工具或系统。

讲授学时：2 学时，实验学时同讲授学时。

本章的知识能力结构图见图 12.1。

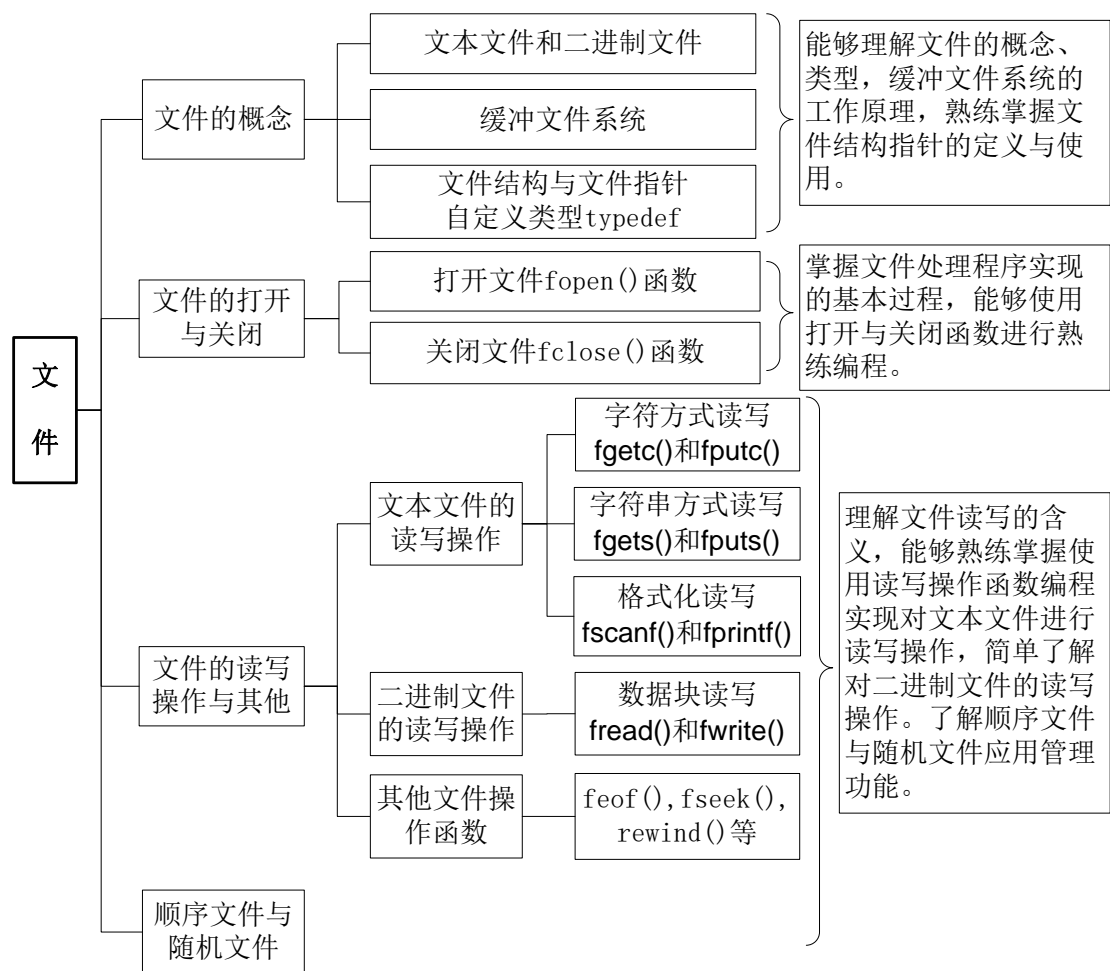

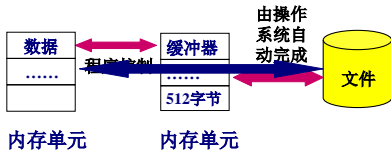
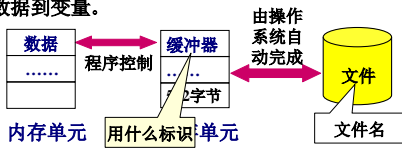



图 12.1 知识能力结构图

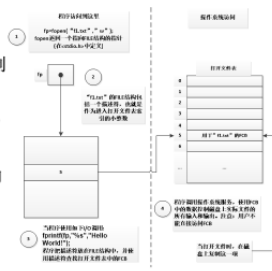
12.2 讲稿

1.	<p>Chap 12 文件</p> <p>12.1 素数文件</p> <p>12.2 用户信息加密和校验</p> <p>12.3 文件综合应用：资金账户管理</p>	<p>作为本章开始，先开篇做个总体介绍：</p> <p>1、常见“文件”一词，学生皆知，教师可先从最熟悉的操作系统中的文件管理，对“文件”做个简单介绍，然后引出本章将介绍关于通过编写 C 语言程序实现“文件”操作的一些方法。</p> <p>2、可以试探性提问学生：</p> <p>1) 文件是怎么产生的？</p> <p>2) 数据怎样写到文件中？</p> <p>3) 文件中的数据又是如何读出来？</p>
----	--	--

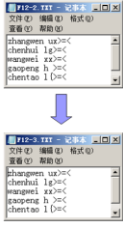
2.	<p>本章要点</p> <ul style="list-style-type: none"> ■ 什么是文件？C文件是如何存储的？ ■ 什么是文件缓冲系统？工作原理如何？ ■ 什么是文本文件和二进制文件？ ■ 怎样打开、关闭文件？ ■ 怎样编写文件读写程序？ ■ 怎样编写程序，实现简单的数据处理？ 	提出本章的学习要点。
3.	<p>12.1 素数文件</p> <p>【例12-1】从2开始依次找出500个素数，将这些素数存入文本文件prime.txt中。</p> 	讲授时，建议先直接运行一下示例程序。引出问题：数据如何“跑”到文件中？
4.	<p>例12-1 源程序</p> <pre> #include <stdio.h> #include <stdlib.h> #include <math.h> int prime(int n); /* 函数声明，定义省略 */ int main(void) { int n = 2, count = 0; FILE *fp; if((fp = fopen("prime.txt", "w")) == NULL){ /* (1) 定义文件指针 */ printf("File open error!\n"); /* (2) 打开文件 */ exit(0); } while(count < 500){ if(prime(n) != 0){ count++; fprintf(fp, "%d ", n); /* (3) 文件处理-写入 */ } n++; } if(fclose(fp)){ /* (4) 关闭文件 */ printf("Can not close the file!\n"); exit(0); } return 0; } </pre>	分析文件操作程序的总体结构，见程序中的注释部分，只是大概介绍，不做详细展开。
5.	<p>12.1.2 文件的概念</p> <ul style="list-style-type: none"> ■ 文件：操作系统中的文件是指驻留在外部介质（如磁盘等）中的一个有序数据集。 ■ 各种类型的文件 <ul style="list-style-type: none"> □ 程序文件：源文件、目标程序、可执行程序 □ 数据文件（输入/输出）：文本文件、图像文件、声音文件、可执行文件等 ■ 文件的特点： <ul style="list-style-type: none"> □ 数据永久保存；数据长度不定；数据按顺序存取 	用操作系统的观点对文件概念可做进一步解释。





6.	<div>12.1.3 文本文件和二进制文件</div> <div><div>字节字节字节字节字节字节.....</div><p>C语言中的文件是数据流(由一个个的字节数据组成)文件的两种数据形式:</p><ul style="list-style-type: none">ASCII码 (文本文件 text stream) 字符流二进制码 (二进制文件 binary stream) 二进制流<p>二进制文件是直接把内存数据以二进制形式保存。</p><p>例如, 整数1234</p><ul style="list-style-type: none">文本文件保存: 49 50 51 52 (4个字符)二进制文件保存: 04D2 (1234的二进制数)</div>	<p>讲解 C 语言文件的真正含义,是个数据集合并,是个按顺序保存在磁盘中的数据流。两种文件类型或数据形式。</p> <p>许多学生可能不是很理解二进制文件和文本文件的区别,建议举例子说明。</p> <p>比如用记事本打开磁盘中的任意文件,查看其内容。</p>
7.	<div>12.1.4 缓冲文件系统</div> <div><p>由于磁盘速度慢 直接把数据写到磁盘效率很低</p><p>内存单元 内存单元 文件</p><p>程序控制 由操作系统自动完成</p></div>	<p>任何程序在内存中实现,在内存中运行,所有数据以变量(内存)形式被存取处理,文件被存储在磁盘的数据,在两者之间引入了缓冲器,作为数据交换的临时空间。这里要讲清楚文件缓冲器的作用。顺便介绍一下非缓冲文件系统,两者的区别在哪里。</p>
8.	<div>12.1.4 缓冲文件系统</div> <div><ul style="list-style-type: none">向磁盘输出数据: 数据 → 缓冲器, 装满缓冲器后 → 磁盘文件。从磁盘读入数据: 先一次性从磁盘文件将一批数据输入到缓冲器,然后再从缓冲器逐个读入数据到变量。<p>内存单元 缓冲器 文件</p><p>程序控制 由操作系统自动完成</p><p>用什么标识单元 文件名</p></div>	<p>重点介绍缓冲文件系统实现数据从内存到磁盘文件和从文件中读入数据到内存的基本过程。</p> <p>引出问题: 缓冲器如何标识?</p>
9.	<div>缓冲文件与文件类型指针</div> <div><p>用文件指针指示文件缓冲区中具体读写的位置</p><p>FILE *fp;</p><p>内存单元 缓冲器 文件</p><p>程序控制 由操作系统自动完成</p><p>同时使用多个文件时,每个文件都有缓冲区,用不同的文件指针分别指示。</p></div>	<p>文件缓冲器是一段内存,可以用指针变量存储它的首地址,这样类似指针操作数组,可以对文件进行操作了。</p>

10	<h2>12.1.5 文件结构与文件类型指针</h2> <h3>1. 文件结构与自定义类型typedef</h3> <p>FILE: 结构类型, 用 typedef 定义(见stdio.h)</p> <pre>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;</pre>	<p>深入认识一下文件结构, 引出自定义数据类型的定义方法。</p> <p>需要说明这里的 FILE 是一个自定义类型, 并不是一种新的数据类型标识符。</p>
11	<p>自定义类型 (typedef):</p> <ul style="list-style-type: none"> □ 将C语言中的已有类型 (包括已定义过的自定义类型) 重新命名 □ 新的名称可以代替已有数据类型 □ 常用于简化对复杂数据类型定义的描述 <pre>typedef <已有类型名> <新类型名>; typedef int INTEGER; int i, j; <====> INTEGER i, j; typedef int* POINT; int* p1; <====> POINT p1;</pre>	<p>介绍自定义数据类型的定义语法。</p>
12	<h2>自定义类型 (typedef) 的使用方法</h2> <ul style="list-style-type: none"> □ 定义变量 □ 变量名 → 新类型名 □ 加上 typedef □ 用新类型名定义变量 <pre>int i int → INTEGER typedef int INTEGER INTEGER i; int num[10] int NUM[10] typedef int NUM[10] NUM a <====> int a[10]</pre>	<p>介绍自定义数据类型的定义步骤, 介绍了 typedef 的详细用法。</p>
13	<h2>2. 文件类型指针</h2> <p>FILE * fp 如何使 fp 与具体文件挂钩?</p> <p>指向文件缓冲区, 通过移动指针实现对文件的操作</p> <p>同时使用多个文件时, 每个文件都有缓冲区, 用不同的文件指针分别指示。</p>	<p>介绍文件结构指针的定义方法, 文件指针的作用, 与文件缓冲区之间的关系。</p>

14	<h3>12.1.6 文件控制块FCB</h3> <ul style="list-style-type: none"> 文件控制块FCB (File Control Block) OS中对文件的操作控制通过FCB, 处理的是FCB列表 一个文件对应一个FCB 文件缓冲区由程序中fopen语句动态创建 打开文件时, FCB的内容信息被复制到文件缓冲区保存 用文件指针指向文件缓冲区实现对文件数据的访问 	介绍文件控制块的含义, 与文件指针、文件缓冲区之间的关系
15	<h3>12.1.7 文件处理步骤</h3> <ul style="list-style-type: none"> 四个步骤: <ul style="list-style-type: none"> ① 定义文件指针 ② 打开文件: 文件指针指向磁盘文件缓冲区 ③ 文件处理: 文件读写操作 ④ 关闭文件 	介绍文件处理的一般步骤, 严格遵循四步。
16	<h3>12.2 用户信息加密和校验</h3> <p>【例12-2】为了保障系统安全, 通常采取用户帐号和密码登录系统。系统用户信息存放在一个文件中, 系统帐号名和密码由若干字母与数字字符构成, 因安全需要文件中的密码不能是明文, 必须要经过加密处理。请编程实现: 输入5个用户信息 (包含帐号名和密码) 并写入文件f12-2.dat。要求文件中每个用户信息占一行, 帐号名和加密过的密码之间用一个空格分隔。密码加密算法: 对每个字符ASCII码的低四位求反, 高四位保持不变 (即将其与15进行异或)。</p>	在介绍了文件概念后, 给出一个示例程序“用户信息的加密和校验”, 实例实现了用户帐号文件的管理, 密码的加密。逐步引出知识点: 文件打开与关闭、文件读写基本函数与其他函数。
17	<h3>12.2.1 程序解析</h3> <pre> #include <stdio.h> #include <string.h> struct sysuser { /* 用户帐号信息结构 */ char username[20]; char password[6]; }; int main(void) { FILE *fp; /* 1. 定义文件指针 */ int i; void encrypt(char *pwd); struct sysuser su; /* 2. 打开文件, 进行写入操作 */ if((fp=fopen("f12-2.txt", "w")) == NULL){ printf("File open error!\n"); exit(0); } for(i=1; i<=5; i++){ /* 3. 将5位用户帐号信息写入文件 */ printf("Enter %i th sysuser(name password):", i); scanf("%s%s", su.username, su.password); /* 输入用户名和密码 */ encrypt(su.password); /* 进行加密处理 */ fprintf(fp, "%s %s\n", su.username, su.password); /* 写入文件 */ } if(fclose(fp)) { /* 4. 关闭文件 */ printf("Can not close the file!\n"); exit(0); } return 0; } </pre> <pre> /* 加密算法 */ void encrypt(char *pwd) { int i; /* 与15 (二进制码是00001111) 异或, 实现低四位取反, 高四位保持不变 */ for(i=0; i<strlen(pwd); i++){ pwd[i] = pwd[i] ^ 15; } } </pre>	将源代码复制到记事本中运行演示, 结合源程序, 做深入分析, 指出写文件时“打开文件操作”的重要作用, 强调打开文件函数 fopen()与关闭文件函数 fclose()。

18	<div>12.2.2 打开文件和关闭文件</div> <pre>if((fp=fopen("f12-2.txt","w")) == NULL){ printf("File open error!\n"); exit(0); } fopen("文件名", "文件打开方式") □ 使文件指针与相应文件实体对应起来 □ 程序对文件指针进行操作, 即fp代表磁盘文件 ■ 函数fopen() 的返回值 □ 执行成功, 则返回包含文件缓冲区等信息的FILE型地址, 赋给文件指针fp □ 不成功, 则返回一个NULL (空值) exit(0): 关闭所有打开的文件, 并终止程序的执行 参数0表示程序正常结束; 非0参数通常表示不正常的程序结束</pre>	介绍打开文件函数 fopen()的使用, 文件打开方式的重要性。要求学生掌握。																																
19	<div>文件打开方式</div> <pre>fp=fopen("f12-2.txt","w") ■ 文件打开方式参数表</pre> <table><tr><th colspan="2">文 本 文 件 (ASCII)</th><th colspan="2">二 进 制 文 件(Binary)</th></tr><tr><th>使用方式</th><th>含 义</th><th>使用方式</th><th>含 义</th></tr><tr><td>" r "</td><td>打开只读文件</td><td>" rb "</td><td>打开只读文件</td></tr><tr><td>" w "</td><td>建立只写新文件</td><td>" wb "</td><td>建立只写新文件</td></tr><tr><td>" a "</td><td>打开添加写文件</td><td>" ab "</td><td>打开添加写文件</td></tr><tr><td>" r+ "</td><td>打开读/写文件</td><td>" rb+ "</td><td>打开读/写文件</td></tr><tr><td>" w+ "</td><td>建立读/写新文件</td><td>" wb+ "</td><td>建立读/写新文件</td></tr><tr><td>" a+ "</td><td>打开读/写文件</td><td>" ab+ "</td><td>打开读/写文件</td></tr></table>	文 本 文 件 (ASCII)		二 进 制 文 件(Binary)		使用方式	含 义	使用方式	含 义	" r "	打开只读文件	" rb "	打开只读文件	" w "	建立只写新文件	" wb "	建立只写新文件	" a "	打开添加写文件	" ab "	打开添加写文件	" r+ "	打开读/写文件	" rb+ "	打开读/写文件	" w+ "	建立读/写新文件	" wb+ "	建立读/写新文件	" a+ "	打开读/写文件	" ab+ "	打开读/写文件	介绍常用的文件打开方式。学生要重点掌握的几个“使用方式”：“r”，“w”，“a”；理解其含义。
文 本 文 件 (ASCII)		二 进 制 文 件(Binary)																																
使用方式	含 义	使用方式	含 义																															
" r "	打开只读文件	" rb "	打开只读文件																															
" w "	建立只写新文件	" wb "	建立只写新文件																															
" a "	打开添加写文件	" ab "	打开添加写文件																															
" r+ "	打开读/写文件	" rb+ "	打开读/写文件																															
" w+ "	建立读/写新文件	" wb+ "	建立读/写新文件																															
" a+ "	打开读/写文件	" ab+ "	打开读/写文件																															
20	<div>文件读写与打开方式</div> <pre>if 读文件 指定的文件必须存在, 否则出错; if 写文件(指定的文件可以存在, 也可以不存在) if 以 "w" 方式写 if 该文件已经存在 原文件将被删去重新建立; else 按指定的名字新建一个文件; else if 以 "a" 方式写 if 该文件已经存在 写入的数据将被添加到指定文件原有数据的后面, 不会删去原来的内容; else 按指定的名字新建一个文件 (与"w"相同); if 文件同时读和写 使用 "r+", "w+" 或 "a+" 打开文件</pre>	这是文件读写操作时文件打开方式的用法。																																
21	<div>关闭文件</div> <pre>if(fclose(fp)){ printf("Can not close the file!\n"); exit(0); } fclose(文件指针) □ 把缓冲区中的数据写入磁盘扇区, 确保写文件的正常完成 □ 释放文件缓冲区单元和FILE结构体, 使文件指针与具体文件脱钩。 函数fclose() 的返回值 □ 返回0: 正常关闭文件 □ 返回非0: 无法正常关闭文件</pre>	介绍关闭文件的实现方法。要学生注意其用法, 掌握关闭文件的作用。																																

22	<h3>12.2.3 文件读写</h3> <p>【例12-3】复制用户文件。将例12-2的用户信息文件f12-2.txt文件备份一份，取名为文件f12-3.txt。说明：运行程序前请将文件f12-2.txt与源程序放在同一目录下。</p>	<p>本小节要进一步深入展开文件操作知识讲解。</p> <p>“复制用户文件”示例程序，包含两个过程，数据从一个文件中读出，向另一个文件写入，完成文件复制功能。</p>
23	<h3>例12-3 源程序</h3> <pre>#include <stdio.h> int main(void) { FILE *fp1,*fp2; char ch; if((fp1 = fopen("f12-2.txt", "r")) == NULL){ printf("File open error!\n"); exit(0); } if((fp2 = fopen("f12-3.txt", "w")) == NULL){ printf("File open error!\n"); exit(0); } while(!feof(fp1)){ ch = fgetc(fp1); if(ch!=EOF) fputc(ch, fp2); } /*关闭文件f12-2.txt*/ if(fclose(fp1)){ printf("Can not close the file!\n"); exit(0);} /*关闭文件f12-3.txt*/ if(fclose(fp2)){ printf("Can not close the file!\n"); exit(0);} return 0; }</pre> 	<p>分析源程序，重点介绍读写文件打开方式的运用，从一个文件读，向另一文件写。指出程序中有两个重要函数：</p> <ol style="list-style-type: none"> 1) 从文件读出数据的方法：fgetc() 2) 写入数据到文件的方法：fputc() <p>语法暂时不做详细解释。</p>
24	<h3>打开多个文件</h3> <pre>if((fp1 = fopen("f12-2.txt", "r")) == NULL){ printf("File open error!\n"); exit(0); } if((fp2=fopen("f12-3.txt", "w")) == NULL){ printf("File open error!\n"); exit(0); }</pre> <p>C语言允许同时打开多个文件</p> <ul style="list-style-type: none"> □ 不同的文件对应不同的文件指针 □ 不允许同一个文件在关闭前再次打开 	<p>介绍在程序中实现同时打开多个文件的方法。利用多个不同文件指针可以指向不同文件打开的文件缓冲器。</p>
25	<h3>文件读写函数</h3> <ul style="list-style-type: none"> ■ 字符读写函数：fgetc() / fputc() ■ 字符串读写函数：fputs() / fgets() ■ 格式化读写函数：fscanf() / fprintf() ■ 二进制读写函数：fread ()/ fwrite() ■ 其他相关函数： <ul style="list-style-type: none"> □ 检测文件结尾函数feof() □ 检测文件读写出错函数ferror() □ 清除末尾标志和出错标志函数clearerr() □ 文件定位的函数fseek()、rewind()、ftell() 	<p>这里给出所有文件读写函数。</p>

26	<div>  <h2>1. 字符读写函数fgetc和fputc</h2> <pre>while(!feof(fp1)){ ch = fgetc(fp1); if(ch!=EOF) fputc(c, fp2); }</pre> <div> <div> ■ 函数fgetc() <code>ch = fgetc(fp);</code> 从fp所指示的磁盘文件上读入一个字符到ch □ 区分键盘字符输入函数 <code>getchar()</code> </div> <div> ■ 函数fputc() <code>fputc(ch, fp);</code> 把一个字符ch写到fp所指示的磁盘文件上 □ 返回值 ■ -1 (EOF): 写文件失败 ■ ch: 写文件成功 </div> </div> </div>	<p>介绍示例程序中实现数据读写的程序段中的两个函数 fgetc()和 fputc()。</p> <p>1) 先介绍从文件读出一个字符的函数 fgetc()。注意：函数执行后会返回一个字符，是从文件中读出的一个字符，建议最后保存到一个字符变量。</p> <p>结合一下 getchar()函数，实现从标准键盘输入的字符中读入一个字符。</p> <p>实际上 C 把标准键盘作为标准输入设备，也当作一个文件来处理。</p> <p>2) 介绍 fputc()函数的用法。</p> <p>注意：是把<u>一个</u>字符写入到文件。</p>
27	<div>  <h2>2. 字符串方式读写函数fgets和fputs</h2> <div> ■ 函数fputs() <code>fputs(s, fp);</code> 用来向指定的文本文件写入一个字符串 □ s: 要写入的字符串，结束符' \0'不写入文件。 □ 函数返回值 ■ 执行成功，函数返回所写的最后一个字符 ■ 否则，函数返回EOF </div> </div>	<p>实现以字符串方式的文件读写函数。介绍了函数 fputs()的用法，功能是：向文本文件写入一个字符串。</p> <p>注意：成功和失败的返回值。EOF 的值是-1。</p>
28	<div>  <h2>字符串方式读写函数fgets和fputs</h2> <div> ■ 函数fgets() <code>fgets(s, n, fp);</code> 从文本文件中读取字符串 □ s: 可以是字符数组名或字符指针；n: 指定读入的字符个数；fp: 文件指针 □ 函数被调用时，最多读取n-1个字符，并将读入的字符串存入s所指向内存地址开始的n-1个连续的内存单元中。 当函数读取的字符达到指定的个数，或接收到换行符，或接收到文件结束标志EOF时，将在读取的字符后面自动添加一个' \0'字符；若有换行符，则将换行符保留（换行符在' \0'字符之前）；若有EOF，则不保留 □ 函数返回值 ■ 执行成功，返回读取的字符串； ■ 如果失败，则返回空指针，这时，s的内容不确定 </div> </div>	<p>介绍函数 fgets()的语法，功能是从文件中读入一个给定个数的字符串。注意函数调用形式、函数的参数。</p>
29	<div>  <h2>例12-4</h2> <p>例12-2的f12-2.txt文件保存着系统用户信息，编写一个函数checkUserValid()用于登录系统时校验用户的合法性。检查方法是：</p> <div> <div>□ 在程序运行时输入用户名和密码，然后在用户文件中查找该用户信息，如果用户名和密码在文件中找到，则表示用户合法，返回1，否则返回0。</div> <div>□ 程序运行时，输入一个用户名和密码，调用checkUserValid()函数，如果返回1，则提示“Valid user!”，否则输出“Invalid user!”。</div> </div> <div> 提示：合法性检查的规则。由于文件中的用户名和密码按行存取，把一行看作整体得字符串s1，将输入的用户名和密码加密后生成另一个字符串s2，然后通过比较s1和s2，来确定文件中是否存在用户。 </div> </div>	<p>例 12-4 在给定的文件中给定的查找一个字符串，以校验用户存在的合法性。</p>

30	<p>例12-4源程序</p> <pre> /*校验用户信息的合法性，成功返回1，否则返回0*/ int checkUserValid(struct sysuser *psu) { FILE *fp; char usr[30],usr1[30],pwd[10]; int check=0; /*检查结果变量，初始化为0*/ /*连接生成待校验字符串*/ strcpy(usr,psu->username); /*复制psu->username到usr1*/ strcpy(pwd,psu->password); /*复制psu->password到pwd*/ encrypt(pwd); /*调用例12-2的encrypt对密码进行加密*/ /*连接usr、空格、pwd和\n构成新字符串usr，用于在文件中逐行检查*/ strcat(usr," "); strcat(usr,pwd); strcat(usr,"\n"); /*打开文件"f12-2.txt"读入*/ if(!fopen("f12-2.txt","r")){ printf("File open error!\n"); exit(0); } /*从文件读入用户信息数据，遍历判断是否存在*/ while(!feof(fp)){ fgets(usr1,30,fp); /*读入一行用户信息作为一个字符串到usr1*/ if(strcmp(usr,usr1)==0){ /*比较判断usr与usr1是否相同*/ check=1; break; } } if(!fclose(fp)){ printf("Can not close the file!\n"); exit(0); } /*关闭文件*/ return check; } </pre>	<p>本例实现的思路是：代码中，将输入的用户名和密码，密码被加密处理，链接后组成一个字符串，然后与文件中的各行整体构成的字符串进行比较，确定是否存在于用户文件中。</p>
31	<p>3. 格式化文件读写fscanf和fprintf</p> <ul style="list-style-type: none"> ■ fscanf（文件指针，格式字符串，输入表）； ■ fprintf（文件指针，格式字符串，输出表）； <p>指定格式的输入输出函数</p> <pre> FILE *fp; int n; float x; fp = fopen("a.txt", "r"); fscanf(fp, "%d%f", &n, &x); </pre> <p>表示从文件a.txt分别读入整数到变量n、浮点数到变量x</p> <pre> fp = fopen("b.txt", "w"); fprintf(fp, "%d%f", n, x); </pre> <p>表示把变量n和x的数值写入文件b.txt</p>	<p>介绍了以格式化方式进行文件读写的两个重要函数：fscanf()、fprintf()。对比标准格式化输入输出函数 scanf()、printf()。注意使用语法格式。</p>
32	<p>4. 数据块读写fread()和fwrite()</p> <ul style="list-style-type: none"> ■ fread(buffer, size, count, fp); 从二进制文件中读入一个数据块到变量 ■ fwrite(buffer, size, count, fp); 向二进制文件中写入一个数据块 <ul style="list-style-type: none"> □ buffer: 指针，表示存放数据的首地址； □ size: 数据块的字节数 □ count: 要读写的数据块块数 □ fp: 文件指针 	<p>介绍以数据块方式进行的文件读写操作函数。指出主要针对二进制文件读写。</p>
33	<p>12.2.4 其他相关函数</p> <div> <div> <p>■ 函数feof()</p> <p>feof(fp);</p> <p>判断fp指针是否已经到文件末尾，函数返回值</p> <ul style="list-style-type: none"> ■ 1: 到文件结束位置 ■ 0: 文件未结束 </div> <div> <p>■ 函数rewind()</p> <p>rewind(FILE *fp);</p> <p>定位文件指针，使文件指针指向读写文件的首地址，即打开文件时文件指针所指向的位置。</p> </div> </div>	<p>函数 feof(): 在读文件时，每读一次，文件指针会向后移动一步，这时需要判断是否已经读到文件尾，这样才能确定是否已经读完所有数据。注意函数返回为 1 时，表明已经到了文件结束位置。</p> <p>函数 rewind()可以定位文件指针，使文件指针重新指向读写文件的首地址。</p>

34	<p>■ 函数fseek()：控制指针移动</p> <p>fseek(fp, offset, from);</p> <ul style="list-style-type: none"> □ offset: 移动偏移量, long型 □ from: 起始位置, 文件首部、当前位置和文件尾部分别对应0,1,2, 或常量SEEK_SET、SEEK_CUR、SEEK_END。 <p>例如:</p> <p>fseek(fp, 20L, 0): 将文件位置指针移动到离文件首20字节处</p> <p>fseek(fp, -20L, SEEK_END): 将文件位置指针移动到离文件尾部前20字节处</p> <p>■ 函数ftell()</p> <p>ftell(文件指针);</p> <p>获取当前文件指针的位置, 即相对于文件开头的位移量 (字节数)</p> <ul style="list-style-type: none"> □ 函数出错时, 返回-1L 	<p>函数 fseek()用来控制文件指针移动到给定位置, 注意函数参数, 只要给出偏移量的值。</p> <p>函数 ftell()获得文件指针指示的位置。</p>
35	<p>其他相关函数</p> <p>■ ferror() 函数: 函数用来检查文件在用各种输入输出函数进行读写是否出错, 若返回值为0, 表示未出错, 则表示有错</p> <p>调用形式为: ferror(文件指针);</p> <ul style="list-style-type: none"> □ 文件指针必须是已经定义过的 <p>■ 函数clearerr()</p> <p>clearerr(文件指针);</p> <p>用来清除出错标志和文件结束标志, 使它们为0</p>	<p>简要介绍即可。</p>
36	<p>12.3 文件综合应用：个人资金账户管理</p> <p>■ 12.3.1顺序文件和随机文件</p> <p>按照C程序对文件访问的特点来分, 文件可分为顺序访问文件和随机访问文件, 简称为顺序文件和随机文件。前面介绍的所有例子都进行的是顺序访问, 通过使用fprintf()或fputs()函数创建的数据记录长度并不是完全一致的, 这种记录长度不确定的文件访问称为顺序访问。而随机访问文件要求文件中单个记录的长度固定, 可直接访问, 这样速度快, 并且无需通过其他记录查找特定记录。因此随机文件适合银行系统、航空售票系统、销售点系统和其他需要快速访问特定数据的事务处理系统。</p>	<p>讲解文件的访问特性, 把文件分为顺序文件和随机文件。</p>
37	<p>12.3.2 个人资金帐户的管理</p> <p>■ 要求</p> <ul style="list-style-type: none"> □ 个人资金账户的信息统一放在随机文件中, 该随机文件包括的数据项有记录ID、发生日期、发生事件、发生金额 (正+的表示收入, 负-表示支出) 和余额。每记录一次收支, 文件要增加一条记录, 并计算一次余额。 □ 程序实现3个功能, 包括: 1) 可以创建该文件并添加新收入或支出信息; 2) 可以显示所有记录列表, 得知资金账户的收支流水帐; 3) 查询最后一条记录, 获知账户最后的余额。 	<p>介绍文件的综合应用, 利用随机文件的特性, 可实现信息记录型数据的管理。本例由于程序较长, 课堂只需做一个简单介绍, 留给学生自己阅读。</p>

38	<h3>cashbox.dat文件的部分内容</h3> <table><thead><tr><th>LogID</th><th>CreateDate</th><th>Note</th><th>Charge</th><th>Balance</th></tr></thead><tbody><tr><td>1</td><td>2006-06-01</td><td>alimony</td><td>500.00</td><td>500.00</td></tr><tr><td>2</td><td>2006-06-08</td><td>shopping</td><td>-300.00</td><td>200.00</td></tr><tr><td>3</td><td>2006-06-15</td><td>shopping</td><td>-60.00</td><td>140.00</td></tr><tr><td>4</td><td>2006-06-20</td><td>workingpay</td><td>200.00</td><td>340.00</td></tr><tr><td>5</td><td>2006-08-01</td><td>scholarship</td><td>1000.00</td><td>1340.00</td></tr><tr><td colspan="5">.....</td></tr></tbody></table>	LogID	CreateDate	Note	Charge	Balance	1	2006-06-01	alimony	500.00	500.00	2	2006-06-08	shopping	-300.00	200.00	3	2006-06-15	shopping	-60.00	140.00	4	2006-06-20	workingpay	200.00	340.00	5	2006-08-01	scholarship	1000.00	1340.00					说明账户文件中记录的格式。
LogID	CreateDate	Note	Charge	Balance																																	
1	2006-06-01	alimony	500.00	500.00																																	
2	2006-06-08	shopping	-300.00	200.00																																	
3	2006-06-15	shopping	-60.00	140.00																																	
4	2006-06-20	workingpay	200.00	340.00																																	
5	2006-08-01	scholarship	1000.00	1340.00																																	
.....																																					
39	<div><h3>本章总结</h3><ul style="list-style-type: none">■ 文件的概念<ul style="list-style-type: none">□ 文本文件和二进制文件□ 文件缓冲系统□ 文件结构，文件指针，自定义类型■ 文件的打开与关闭<ul style="list-style-type: none">□ 文件处理实现过程■ 文件读写操作与常用文件操作函数■ 文件综合应用</div> <div><ol style="list-style-type: none">1) 掌握文件的概念，文件缓冲系统的基本原理，文件读写操作实现的基本过程；2) 能熟练使用文件进行编程。3) 掌握常用的文本文件读写操作函数。4) 了解顺序文件和随机文件的应用。</div>	回顾和总结本章的教学要点，对学生提出能力要求。																																			
40.																																					

12.3 练习与习题参考答案

12.3.1 练习参考答案

12-1 什么是文件缓冲区？在 C 程序中，文件类型指针主要的功能是什么？

解答：为了提高数据存取访问的效率，在程序与文件之间有一个内存缓冲区，程序与文件的数据交换通过该缓冲区来进行。文件类型指针是特殊指针，指向的是文件类型结构，每一个文件都有自己的 FILE 结构和文件缓冲区，通过文件指针可以访问和操作文件缓冲区中的数据。

12-2 改写例 12-2，加密规则改为例 7-12 中的凯撒密码。

解答：

将例 12-2 中的加密函数 encrypt()修改如下：

```
void encrypt ( char *s, int offset)
{
    int i;
    for(i = 0; s[i] != '\0'; i++){
        if(s[i] >= 'A' && s[i] <= 'Z'){
            s[i] = s[i] + offset;
            if(s[i] > 'Z'){
                s[i] = s[i] - 26;
            }
        }
        else if(s[i] >= 'a' && s[i] <= 'z'){
            s[i] = s[i] + offset;
            if(s[i] > 'z'){
```

```

        s[i] = s[i] - 26;
    }
}
}
}

```

将主函数中的调用语句 `encrypt(su.password);` 改为:

```
encrypt(su.password, 2); /* 第 2 个实参是偏移量, 可以自由设定 */
```

12-3 例 12-3 中,为什么在执行 `fputc(ch, fp2)` 前要判断 `ch` 的值是否等于 `EOF`? 改写例 12-3 的程序, 在复制用户信息文件后, 再统计被复制文件中字符的数量。

解答:

1) 为什么在执行 `fputc(ch, fp2)` 前要判断 `ch` 的值是否等于 `EOF` 呢?

我们来分析整个 `while` 循环语句所在的程序段:

```

while(!feof(fp1)){
    ch=fgetc(fp1);
    if(ch!=EOF) fputc(ch,fp2);
}

```

功能是只要 `!feof(fp1)` 的结果非 0, 就不断地执行循环体, 调用函数 `fgetc()` 从 `fp1` 所指向的文件中读取一个字符, 如果 `ch` 不是 `EOF`, 就用函数 `fputc()` 将字符写入另一个 `fp2` 指向的文件。函数 `feof()` 的功能是判断文件指针是否指到文件末尾, 一般要跟在读语句如 `fgetc()` 之后, 读语句 `fgetc()` 读到 `EOF`, `!feof()` 就返回 0。不难发现, 由于程序段先判断 `!feof()`, 再执行读语句 `ch=fgetc(fp1)`, 因此在 `!feof()` 结果还是非 0 即循环条件还满足时, 此时 `fgetc()` 还是会把 `EOF` 读过来的, 如果不加 `ch!=EOF` 的判断, 那么 `EOF` 会被 `fputc()` 写入到新文件, 这样新文件的字符数会比原来的文件多一个值为 `EOF` 的非法字符。

2) 统计字符的程序源代码:

```

#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *fp;
    char ch;
    int count=0; /*统计变量*/
    /*打开文件, 准备读数据*/
    if((fp=fopen("f12-2.txt", "r"))==NULL){
        printf("File open error!\n");
        exit(0);
    }
    /*从文件每读取 1 次, 统计变量增 1 */
    while(!feof(fp)){
        ch=fgetc(fp); /*从 fp 所指示的文件中读取一个字符*/
        if(ch!=EOF) count++; /* 过滤 EOF 不算有效 ASCII 字符, 否则多统计 1 次*/
    }
    printf("Char total: %d\n", count); /*输出字符个数*/
    /*关闭文件 f12-2.txt */
}

```

```

        if(fclose(fp)){
            printf("Can not close the file!\n");
            exit(0);
        }
        return 0;
    }
}

```

12-4 字母转换并统计行数：读取一个指定的文本文件，显示在屏幕上，如果有大写字母，则改成小写字母再输出，并统计行数。根据回车符统计文件的行数。试编写相应程序。

解答：

```

#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *fp;
    char ch, str[100];
    int line=0;
    /*键盘上输入文件名*/
    printf("Input filename:");
    gets(str);
    /*打开文件，读出数据*/
    if((fp=fopen(str,"r"))==NULL){
        printf("File open error!\n");
        exit(0);
    }
    /*从文件中逐一读取字符*/
    while(!feof(fp)){
        ch=fgetc(fp); /*从 fp1 所指示的文件中读取一个字符*/
        if(ch>='A' && ch<='Z') ch=ch+32; /*大写字母转换为小写的*/
        if(ch=='\n') line++; /*读到回车，则行数增 1 */
        putchar(ch);
    }

    printf("Line count: %d\n",line);/*输出行数*/

    /*关闭文件*/
    if(fclose(fp)){
        printf("Can not close the file!\n");
        exit(0);
    }
}

```

12-5 写字符并验证：从键盘输入一行字符，写入到文件f3.txt中，并重新读出，最终在屏幕上显示验证。程序输入以读到回车符'\n'为结束，读文件时要用EOF来控制循环。试编写相

应程序。

解答：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *fp;
    char ch, str[100];
    /*键盘上输入一行字符*/
    printf("Input a string:"); gets(str);
    /*以写方式打开文件*/
    if((fp=fopen("f3.txt","w"))==NULL){
        printf("File open error!\n");
        exit(0);
    }
    fputs(str,fp);    /*向文件写入字符串*/
    /*关闭文件*/
    if(fclose(fp)){
        printf("Can not close the file!\n");
        exit(0);
    }
    /*打开文件，读数据方式*/
    if((fp=fopen("f3.txt","r"))==NULL){
        printf("File open error!\n");
        exit(0);
    }
    /*从文件中逐一读取字符*/
    do {
        ch=fgetc(fp); /*从 fp 所指示的文件中读取一个字符*/
        putchar(ch); /*输出字符到屏幕*/
    } while(ch!=EOF)
    /*关闭文件*/
    if(fclose(fp)){
        printf("Can not close the file!\n");
        exit(0);
    }
}
```

12-6 实数取整写入文件：文件f1. txt中有若干个实数，请分别读出，将每个实数按四舍五入取整后存入文件f2. txt中。试编写相应程序。

解答：

```
#include <stdio.h>
#include "stdlib.h"
int main(void)
```

```

{
    FILE *fp1,*fp2;
    double x;
    /*打开文件，读出数据*/
    if((fp1=fopen("f1.txt","r"))==NULL){
        printf("File open error!\n");
        exit(0);
    }
    if((fp2=fopen("f2.txt","w"))==NULL){
        printf("File open error!\n");
        exit(0);
    }
    /*从文件中逐一读取实数*/
    while(!feof(fp1)){
        fscanf(fp1, "%lf",&x);/*从 fp1 指向的文件中读取一个实数*/
        x++;                /*加 1*/
        fprintf(fp2, "%f ",x); /*写入一个实数到 fp2 指向的文件*/
    }
    /*关闭文件*/
    if(fclose(fp1)){
        printf("Can not close the file!\n");
        exit(0);
    }
    if(fclose(fp2)){
        printf("Can not close the file!\n");
        exit(0);
    }
}

```

12-7 修改例 12-6，增加修改资金账户的功能。输入一个记录 ID，如果文件中已存在该记录，则输入新的记录信息并更新资金账户文件中相应记录的信息。要求定义和调用函数 UpdateLog()，其功能是修改资金账户记录。

解答：在主程序例 12-6 基础上，要修改的代码如下：

(1) 修改函数 inputchoice() 的 printf 语句：增加提示选择参数 4，表示更新记录功能。

/*选择操作参数*/

```

int inputchoice(){
    .....
    printf("3 - Query last cash log.\n4 - Updatelog.\n0 - End program.\n");
    .....
}

```

(2) 修改函数 main() 增加函数声明，对 while 语句中的 switch 语句增加一种 case 选择，调用函数 UpdateLog()。

```

int main(){
    .....
}

```



```

void Updatelog(FILE *cfptr); /*增加函数声明*/
.....
while((choice=inputchoice())!=0){
    switch(choice){
        .....
        case 4:
            .....
        case 5:fp=fopen("rb+");Updatelog(fp);break;/*新增语句更新资金账户记录*/
        default:
            printf("Input Error.");break;
    }
}
.....
}

```

(3) 函数 UpdateLog()

基本思路：程序运行时先输入一个记录 ID，然后在文件中查找。如果不存在该记录，则提示输入错误即记录号无效。如果存在记录，则先显示该记录，然后提示并输入新的记录信息。如果记录收支额有变化，则需重新计算该记录的余额值，还要注意更新某记录的收支额，会影响本记录的余额，同时也会影响到后面所有记录的余额信息，因此所有当前记录开始之后的记录的余额都要重新计算，最后将修改后的记录信息更新回文件。

/*函数功能：查询记录 ID 并更新账户记录*/

```

void Updatelog(FILE *cfptr){
    struct LogData log[1000],*plog=log, newlog;/*假定文件不超过 1000 记录*/
    long logcount,logid,i,index=-1;
    printf("Input LogID:"); scanf("%ld",&logid);/*输入要修改的记录 ID*/
    /*查找帐户号是否存在，若存在则更新*/
    logcount=getLogcount(cfptr);/*获取记录数*/
    rewind(cfptr);
    fread(plog,size,logcount,cfptr);
    for(i=0;i<logcount;i++){
        if(logid==log[i].logid) /*已经找到*/
        { /*显示当前记录*/
            printf("logid  logdate  lognote          charge    balance\n");
            /*输出当前记录*/
            printf("%6ld %-11s %-15s %10.2lf %10.2lf\n",
                log[i].logid,log[i].logdate,log[i].lognote,log[i].charge,log[i].balance);
            index=i;break;
        }
    }
    rewind(cfptr);
    if(index>=0){
        printf("Input logdate(format:2006-01-01):");scanf("%s",newlog.logdate);
        printf("Input lognote:");scanf("%s",newlog.lognote);
        printf("Input Charge:Income+ and expend-:");scanf("%lf",&newlog.charge);
    }
}

```

```

        if(strcmp(log[index].lognote,newlog.lognote)!=0)
            strcpy(log[index].lognote,newlog.lognote);
        if(strcmp(log[index].logdate,newlog.logdate)!=0)
            strcpy(log[index].logdate,newlog.logdate);
        /*如果输入的收支额度改变, 重新计算余额*/
        if(newlog.charge!=log[index].charge){
            /*计算新余额*/
            newlog.balance=log[index].balance-log[index].charge+newlog.charge;
            log[index].charge=newlog.charge;/*更新收支*/
            log[index].balance=newlog.balance;        /*更新余额*/
            /*当前记录之后的每条记录余额信息更新*/
            for(i=index+1;i<logcount;i++)
                log[i].balance=log[i-1].balance+log[i].charge;
        }
    }
    else printf("Error logid and try another!");
    rewind(cfptr);
    fwrite(plog,size,logcount,cfptr);/*写回去更新*/
}

```

12.3.2 习题参考答案

一. 选择题

1	2	3	4	5
B	C	D	A	D

二. 填空题

- | | | | |
|---|---|---|-------------------------|
| 1 | 数据长久保存 | 2 | 文本 (或 ASCII 码文件)
二进制 |
| 3 | feof() | 4 | "r"
fgetc(fp) |
| 5 | 将文件名为键盘输入的 infile 表示的
文件内容复制到文件名为键盘输入的
outfile 变量表示的文件中 | | |

三、程序设计题

1. 统计文本文件中各类字符个数: 分别统计一个文本文件中字母、数字及其它字符的个数。试编写相应程序。

解答:

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    char ch;

```

```

int letter=0,digit=0,other=0;

FILE *fp; /*1、定义文件指针*/
/*2、打开文件 假定文本文件名为 f1.txt */
if((fp=fopen("f1.txt","r")) == NULL){
    printf("File open error!\n");
    exit(0);
}
/*3、文件读写操作*/
while(ch!=EOF)
{
    ch=fgetc(fp);/*读入一个字符*/
    if(ch>='a' && ch <='z' || ch >='A' && ch <='Z')
        letter++;
    else if(ch>='0' && ch <='9')
        digit++;
    else
        other++;
}
/*输出结果*/
printf("letter=%d,digit=%d,other=%d\n",letter,digit,other);
/*4.关闭文件*/
if(fclose(fp)){
    printf("Can not close the file!\n");
    exit(0);
}
}

```

2. 将实数写入文件：从键盘输入若干实数（以特殊数值-1 结束），分别写到一个文本文件中。试编写相应程序。

解答：

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    float x;
    FILE *fp; /*1、定义文件指针*/
    /*2、打开文件 假定文本文件名为 f.txt */
    if((fp=fopen("f.txt","w")) == NULL){
        printf("File open error!\n");
        exit(0);
    }
    /*3、文件读写操作*/
    scanf("%f",&x);

```

```

while(x!=-1){
    fprintf(fp,"%f ",x);/* %f 之后有一个空格，保证文件数据间有一个空格*/
    scanf("%f",&x);/*说明：读入数据分隔符是空格或回车 */
}
/*关闭文件*/
if(fclose(fp)){
    printf("Can not close the file!\n");
    exit(0);
}
}

```

3. 比较两个文本文件是否相等：比较两个文本文件的内容是否相同，并输出两个文件中第一次出现不同字符内容的行号及列值。试编写相应程序。

解答：

分析：程序要求比较两个文件的内容，输出首次不同的行和字符在该行中的位置。基本实现方法是以读方式打开两个文件，分别逐个字符取出文件内容，依次比较，并进行行数统计，记录行号和本行中的位置，如果遇到不一样的，则直接输出行号与字符位置。

```

/*假定两个文本文件名分别为 f1.txt 和 f2.txt.*/
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char cha, chb;
    int linea=1, lineb=1, letterpos=1;
    FILE *fpa, *fpb;          /*1、定义文件指针*/

    if((fpa=fopen("f1.txt", "r")) == NULL){ /*2、打开文件 f1.txt  */
        printf("File open error!\n");
        exit(0);
    }

    if((fpb=fopen("f2.txt", "r")) == NULL){ /*2、打开文件 t2.txt  */
        printf("File open error!\n");
        exit(0);
    }
    /*3、文件读写操作*/
    while(!feof(fpa) && !feof(fpb))
    {
        cha=fgetc(fpa);/*从文件 1 读取字符*/
        chb=fgetc(fpb);/*从文件 2 读取字符*/
        if(cha!=chb) break;/*不相等，找到目标*/

        if(cha=="\n"){

```

```

        linea++;/*行号累加*/
        letterpos=1;/*重新初始化位置*/
    }
    if(chb=='\n') lineb++;/*行号累加*/
    letterpos++;/*字符位置累加*/
}
/*输出结果*/
printf("line number=%d,leter position=%d\n",linea,letterpos);
/*关闭文件*/
if(fcclose(fpa)){
    printf("Can not close the file!\n");
    exit(0);
}
if(fcclose(fpb)){
    printf("Can not close the file!\n");
    exit(0);
}
}

```

4. 将文件中的数据求和并写入文本文件尾：文件 Int_Data.dat 中存放了若干整数，将文件中所有数据相加，并把累加和写入该文件的最后。试编写相应程序。

解答：

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    long x,sum=0;
    FILE *fp;/*定义文件指针*/

    /*打开文件 ini.txt 读入数据 */
    if((fp=fopen("Int_Data.dat","r")) == NULL){
        printf("File open error!\n");
        exit(0);
    }
    /*读取数据并求和*/
    while(!feof(fp)){
        fscanf(fp,"%ld",&x); /*读取数据*/
        sum+=x;                /*累加*/
    }
    /*关闭文件*/
    if(fcclose(fp)){
        printf("Can not close the file!\n");
        exit(0);
    }
}

```

```

/*打开文件追加写入 */
if((fp=fopen("ini.txt","a")) == NULL){
    printf("File open error!\n");
    exit(0);
}
/*写入数据*/
fprintf(fp,"%ld\n",sum);
/*关闭文件*/
if(fclose(fp)){
    printf("Can not close the file!\n");
    exit(0);
}
}

```

5. 输出含 for 的行：将文本文件 test.txt 中所有包含字符串“for”的行输出。试编写相应程序。

解答：

分析：程序要求输出文本文件中包含给定字符串“for”的行。基本实现方法：按读方式打开文件，用函数 **fgets()** 逐行取出字符到一个字符串，分析其中是否包含字符串“for”，如包含，则输出该字符串。定义一个查找函数来判断一个字符串中是否包含给定的字符串。假定每行不超过 80 字符。

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char str[80];
    int pos=0,k=0;
    int search(char a[],char b[]);
    FILE *fp;          /*1、定义文件指针*/
    /*2、打开文件 test.txt */
    if((fp=fopen("test.txt","r")) == NULL){
        printf("File open error!\n");
        exit(0);
    }
    /*3、文件读写操作*/
    while(!feof(fp)){
        /*读取一行到字符串 str*/
        fgets(str,80,fp);
        /*判断 str 中是否含有"for"*/
        if(search(str,"for")) printf("%s",str);
    }
    /*关闭文件*/
    if(fclose(fp)){

```

```

        printf("Can not close the file!\n");
        exit(0);
    }
}

/*在 a 中查找 b，没有找到，则返回 0，否则返回 1*/
int search(char a[],char b[])
{
    int i,j,la,lb,flag;
    la=strlen(a);
    lb=strlen(b);
    if(la<lb) return 0;

    for(i=0;i<la-lb;i++) {
        flag=1;
        for(j=0;j<lb;j++)
            if(a[i+j]!=b[j]) flag=0;
        if(flag) return 1;
    }
    return 0;
}

```

6. 删除文件中的注释：将 C 语言源程序（hello.c）文件中的所有注释去掉后存入另一个文件（new_hello.c）。试编写相应程序。

解答：

分析：程序要求去掉一个 C 源程序文件中的注释，注释部分是包含/*和*/中间的部分字符串信息，/*和*/是成对出现的。只要是/*和*/之外的保留，其他的则略去即可。

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    char ch1,ch2;
    int flag=0;

    FILE *fpa,*fpb;          /*1、定义文件指针*/
    if((fpa=fopen("hello.c","r")) == NULL){/*2、打开文件 ff1.c  */
        printf("File open error!\n");
        exit(0);
    }
    if((fpb=fopen("new_hello.c","w")) == NULL){/*打开文件 ff2.c  */
        printf("File open error!\n");
        exit(0);
    }
    /*3、文件读写操作*/
}

```

```

if(!feof(fpa)) ch1=fgetc(fpa);/*读 1 个字符*/
if(ch1!='\n') fputc(ch1,fpb);;
while(!feof(fpa)){
    ch2=fgetc(fpa);
    /*注释部分的左边找到了,设置标记状态为 1,*/
    /*表示后面读出的字符是不用输出的*/
    if(ch1=='/' && ch2=='*') flag=1;
    /*注释部分的右边找到了,表示而后开始的字符是要输出的*/
    if(ch1=='*' && ch2=='/') flag=0;
    /*输出字符*/
    if(flag==0 && ch2 != '/') fputc(ch2,fpb);
    ch1=ch2;
}
/*关闭文件*/
if(fclose(fpa)){
    printf("Can not close the file!\n");
    exit(0);
}
if(fclose(fpb)){
    printf("Can not close the file!\n");
    exit(0);
}
}

```

7.（选做）账户余额管理：创建一个随机文件，用来存储银行账户和余额信息，程序要求能够查询某个账户的余额，当客户发生交易额时（正表示存入，负表示取出），能够更新余额。账户信息包括账号、账号名和余额三个数据项。试编写相应程序。

文件部分内容如下：

AcctNo	AcctName	Balance
1	zhangsan	1000.00
2	lisi	1300.00
3	wangwu	-100.00
.....		

解答：

分析：程序要求创建一个随机文件，文件名假定为 cash.dat。程序应至少实现三个功能：创建帐户信息、显示所有帐户信息、个别帐户余额信息查询和个别帐户交易余额更新功能。实现时，程序启动应提示用户选择要进行的操作，比如：1-创建帐户信息，2-表示显示所有帐户信息、3-个别帐户余额信息查询、4-个别帐户交易后余额更新（存入或取出），0-退出终止程序。

创建帐户信息操作方法：用户选择 1，提示输入帐户号、帐户名，余额初始为 0，先查询帐户号是否存在，若不存在则向随机文件追加生成一条新记录，否则提示“该帐户号已存在”。

显示所有帐户信息：用户选择 2，程序以列表形式列出所有帐户信息，包括帐户号，帐户名和余额。

查询个别帐户信息操作方法：用户选择 3，输入帐户号，查询到则显示本帐户完整信息，否则提示“不存在该帐户号！”。

个别帐户交易后余额更新方法：用户选择 4，输入帐户号与交易额，则先查找该帐户，取出余额，跟当前交易额做加或减操作，重新计算出余额，并更新本帐户余额信息。

/*cash.dat 是随机文件，记录帐户记录信息*/

/*程序的功能：创建随机文件，创建新帐户，帐户列表显示，更新帐户，查询帐户*/

```
#include "stdio.h"
```

```
#include "stdlib.h"
```

```
long size;/*记录宽度*/
```

```
struct AccountData{
```

```
    long acctno;/*帐户 ID*/
```

```
    char acctname[15];/*帐户名*/
```

```
    double balance;/*余额*/
```

```
};
```

```
int inputchoice()/*选择操作参数*/
```

```
{
```

```
    int mychoice;
```

```
    printf("\nEnter your choice:\n");
```

```
    printf("1 - Addnew a account info.\n");
```

```
    printf("2 - List all account info.\n");
```

```
    printf("3 - Query account info.\n");
```

```
    printf("4 - Update account info.\n");
```

```
    printf("0 - Exit program.\n");
```

```
    scanf("%d",&mychoice);
```

```
    return mychoice;
```

```
}
```

```
/*获取文件记录总数*/
```

```
long getAcctcount(FILE *cfptr)
```

```
{
```

```
    long begin,end,count;
```

```
    fseek(cfptr,0L,SEEK_SET);
```

```
    begin=ftell(cfptr);
```

```
    fseek(cfptr,size,SEEK_END);
```

```
    end=ftell(cfptr);
```

```
    count=(end-begin)/size-1;
```

```
    return count;
```

```
}
```

```
/*添加新帐户记录,余额初始为 0*/
```

```
void AddNewAcctInfo(FILE *cfptr)
```

```
{
```

```
    struct AccountData acct,ac[100],*pac=ac;
```

```
    long count,i;int flag=0;
```

```
    printf("Input acctno:");    /*输入帐户号*/
```

```

scanf("%ld",&acct.acctno);
printf("Input AcctName:"); /*输入帐户名*/
scanf("%s",acct.acctname);
acct.balance=0; /*余额初始化为 0*/
/*根据帐户号判断该帐户是否已经有了,帐户号唯一*/
count=getAcctcount(cfptr); /*获取记录数*/
rewind(cfptr);
fread(pac,size,count,cfptr);
/*查找帐户*/
flag=0;
for(i=0;i<count;i++)
{
    if(acct.acctno==ac[i].acctno) /*已经找到*/
    {
        flag=1; /*设置为 1 表示已找到*/
        break;
    }
}
/*没有找到, 则追加新记录*/
if(!flag)
{
    ac[count]=acct;
    rewind(cfptr);
    fwrite(pac,size,count+1,cfptr); /*写回去更新*/
}
}
/*更新个别帐户信息*/
void UpdateBalance(FILE *cfptr)
{
    struct AccountData ac[100],*pac=ac;
    long count,i,acctno;
    double transVal;

    printf("Input AcctNo:");
    scanf("%ld",&acctno);
    printf("Input transaction save+ or consume-:");
    scanf("%lf",&transVal);
    /*查找帐户号是否存在, 若存在则更新*/
    count=getAcctcount(cfptr); /*获取记录数*/
    rewind(cfptr);
    fread(pac,size,count,cfptr);
    for(i=0;i<count;i++)
    {
        if(acctno==ac[i].acctno) /*已经找到*/

```

```

        {
            ac[i].balance += transVal; /*取出余额值计算新的余额*/
            break;
        }
    }
    rewind(cfptr)
    fwrite(pac,size,count,cfptr); /*写回去更新*/
}
/*查询个别帐户信息      参数 acctno:表示要查询的帐户*/
void QueryAccount(FILE *cfptr)
{
    struct AccountData acct;
    long count,i,acctno;
    printf("Input AcctNo:");
    scanf("%ld",&acctno);
    count=getAcctcount(cfptr); /*获取记录数*/
    for(i=0;i<count;i++)
    {
        fseek(cfptr,size*i,SEEK_SET); /*定位*/
        fread(&acct,size,1,cfptr); /*读取当前记录*/
        if(acctno==acct.acctno) /*已经找到输出当前帐户信息*/
        {
            printf("AcctNo:%-6ld\nAcctName:%-20s\nbalance:%-10.2lf\n",
                acct.acctno,acct.acctname,acct.balance);
            break;
        }
    }
}
/*显示所有帐户信息*/
void ListAllAccount(FILE *cfptr)
{
    struct AccountData acct;
    long count,i;
    count=getAcctcount(cfptr); /*获取记录数*/
    printf("AcctNo   AcctName           balance\n");
    for(i=0;i<count;i++)
    {
        fseek(cfptr,size*i,SEEK_SET); /*定位*/
        fread(&acct,size,1,cfptr); /*读取当前记录*/
        printf("%-8ld%-20s%-10.2lf\n",
            acct.acctno,acct.acctname,acct.balance);
    }
}
int main()

```

```

{
    FILE *fp;int choice;
    if((fp=fopen("cash.dat", "rb+")) == NULL){
        printf("can not open file cash.dat!\n");
        exit(0);
    }
    size=sizeof(struct AccountData);
    while((choice=inputchoice())!=0){
        switch(choice)
        {
            case 1:/*创建帐户信息*/
                AddNewAcctInfo(fp);break;
            case 2:/*显示所有帐户信息*/
                ListAllAccount(fp); break;
            case 3:/*查询个别帐户信息*/
                QueryAccount(fp);break;
            case 4:/*更新个别帐户交易后余额信息*/
                UpdateBalance(fp);break;
            default:
                printf("Input Error.");break;
        }
    }
    /*关闭文件*/
    if(fclose(fp)){
        printf("Can not close the file!\n");
        exit(0);
    }
}

```

12.4 实验指导教材参考答案

一、调试示例

将字符写入文件：从键盘输入一行字符，写到文件 myfile.txt 中。

解答：略

二、基础编程题

(1) 统计文本文件中各类字符个数：分别统计一个文本文件中字母、数字及其它字符的个数。试编写相应程序。

解答：参见习题程序设计第 1 题。

(2) 将实数写入文件：从键盘输入若干实数（以特殊数值-1 结束），分别写到一个文本文件中。试编写相应程序。

解答：参见习题程序设计第 2 题。

(3) 统计成绩：从键盘输入以下 10 个学生的学号、姓名，以及数学、语文和英语成绩，写到文本文件 f3.txt 中，再从文件中取出数据，计算每个学生的总成绩和平均分，并将结果显示在屏幕上。试编写相应程序。

解答：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *fp;
    long stuNum; char stuName[10];
    int chinese,math,english,total,avg;
    int i;
    /*以写入方式打开文件*/
    if((fp = fopen("f3.txt", "w")) == NULL){
        printf("Can't Open File!");
        exit(0);
    }
    /*键盘上输入并写入文件*/
    for(i=0;i<10;i++){
        printf("input stunum stuname math chinese english:");
        scanf("%ld%s%d%d%d",&stuNum,stuName,&math,&chinese,&english);
        total=math+chinese+english;
        avg=total/3;
        fprintf(fp,"%ld%s%d%d%d%d",stuNum,stuName,math,chinese,english,total,avg);
    }
    /*关闭文件*/
    if(!fclose(fp)){
        printf("Close file error!");
        exit(0);
    }
    /*以只读方式再次打开文件，读取数据输出显示到屏幕*/
    if((fp = fopen("f3.txt", "r")) == NULL){
        printf("Can't Open File!");
        exit(0);
    }
    while(!feof(fp)){
        fscanf(fp,"%ld%s%d%d%d%d",&stuNum,stuName,&math,&chinese,&english,&total,&avg);
        printf("%ld%s%d%d%d%d",stuNum,stuName,math,chinese,english,total,avg);
    }
    if(!fclose(fp)){
        printf("Close file error!");
    }
}
```

```
    return 0;
}
```

(4) 比较两个文本文件是否相等：比较两个文本文件的内容是否相同，并输出两个文件中第一次出现不同字符内容的行号及列值。试编写相应程序。

解答：参见习题程序设计第 3 题。

(5) 字母转换并统计行数：读取一个指定的文本文件，显示在屏幕上，如果有大写字母，则改成小写字母再输出，并根据回车符统计行数。试编写相应程序。

解答：参见练习 12-4。

三、改错题

将文件中的数据求和并写入文件末尾：文件 Int_Data.txt 中存放了若干整数，将文件中所有数据相加，并把累加和写入该文件的最后。

解答：

错误行号： 6	正确语句： FILE *fp;
错误行号： 7	正确语句： sum=0;
错误行号： 8	正确语句： if((fp = fopen("Int_Data.txt", "r+")) == NULL) {
错误行号： 12	正确语句： while(fscanf(fp, "%d", &n) != EOF) {

四、拓展编程题

(1) 输出含 for 的行：将文本文件 test.txt 中所有包含字符串 "for" 的行输出。试编写相应程序。

解答：参见习题程序设计第 5 题。

(2) 删除文件中的注释：将 C 语言源程序 (hello.c) 文件中的所有注释去掉后存入另一个文件 (new_hello.c)。试编写相应程序。

解答：参见习题程序设计第 6 题。

(3) 账户余额管理：创建一个随机文件，用来存储银行账户和余额信息，程序要求能够查询某个账户的余额，当客户发生交易额时（正表示存入，负表示取出），并能更新余额。账户信息包括：账号、账号名和余额 3 个数据项。试编写相应程序。

解答：参见习题程序设计第 7 题。