清華大學 Tsinghua University

腾讯会议 ID： 908 210 197/ 2021

计算机程序设计基础

第10讲 文件

沈瑜（010-62782951）
shenyu@tsinghua.edu.cn
清华大学电机系

2021.11.16

主要内容

- 文件系统的概念
- 文本文件的打开与关闭
- 文本文件的读写
- 综合实例

参考教材10.1节-10.4节

10.1 文件系统的概念

10.1.1 C文件概述

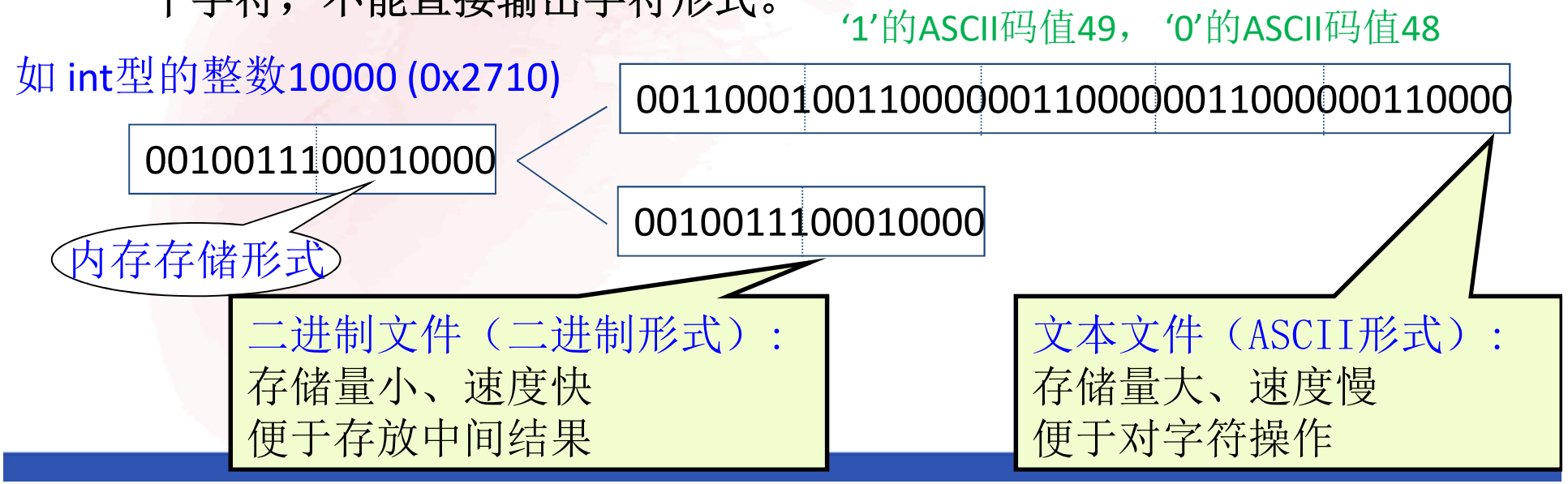
所谓“文件”一般指存储在外部介质上数据的集合。C语言把文件看作一个字节序列，即由一连串的字节组成，称为“流（stream）”

（1）文本文件（本章重点）

每一个字节存放一个ASCII码，代表一个字符。文本文件由文本行组成，每行中可以有0个或多个字符，并以换行符‘\n’结尾。

（2）二进制文件

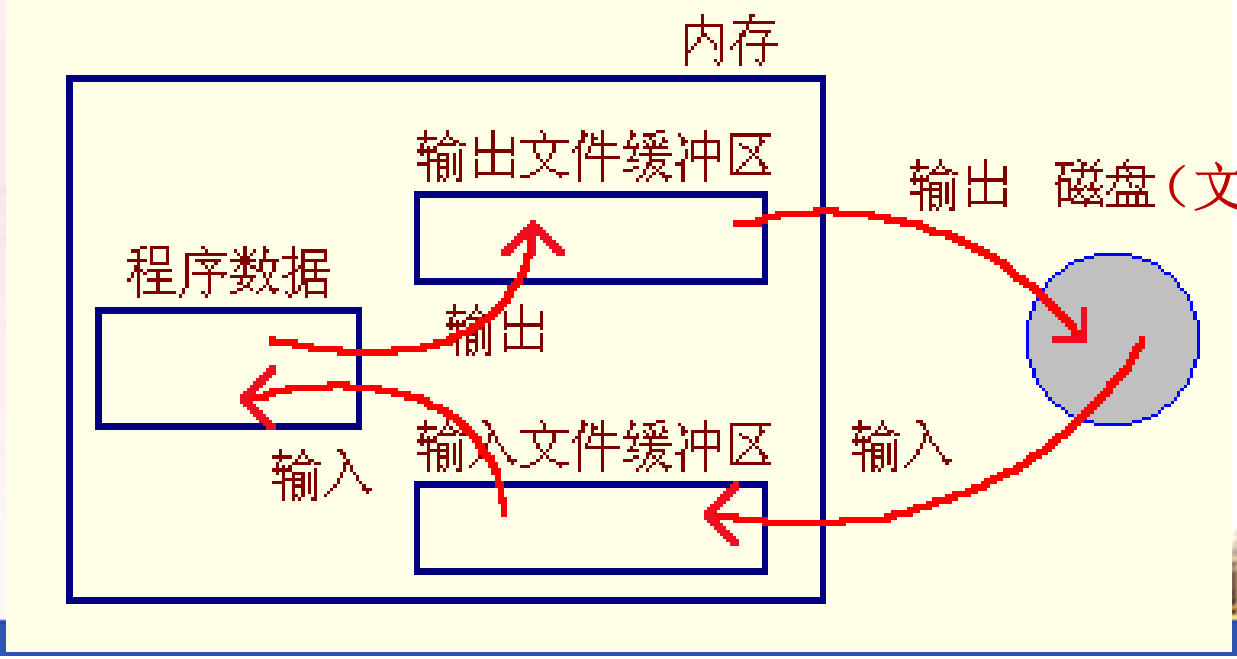
把数据按其在内存中的存储形式原样存放在磁盘上，一个字节并不对应一个字符，不能直接输出字符形式。



10.1.2 缓冲文件系统

- 其特点是在内存开辟一个“缓冲区”，供文件读写使用
- 读文件时，先从磁盘将文件数据读入内存“缓冲区”，装满后再从内存“缓冲区”依次读入程序数据
- 写文件时，先将程序数据写入内存“缓冲区”，待内存“缓冲区”装满后再写入文件。

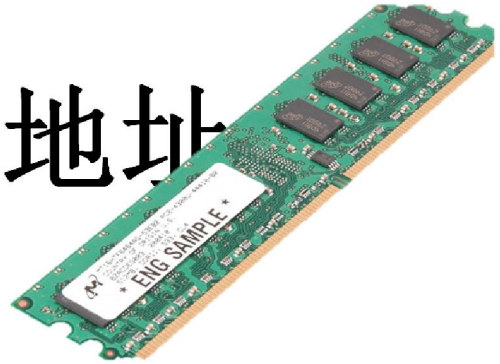
buffer



10.2 文件的打开与关闭

- 文件操作的过程：先打开，后读写，最后关闭
- 打开文件实际上是建立文件的各种有关信息，并使文件指针指向该文件，以便进行其它操作
- 关闭文件则是断开文件指针与文件之间的联系，也就避免了通过该文件指针对文件进行误操作

知识点1：内存与内存地址

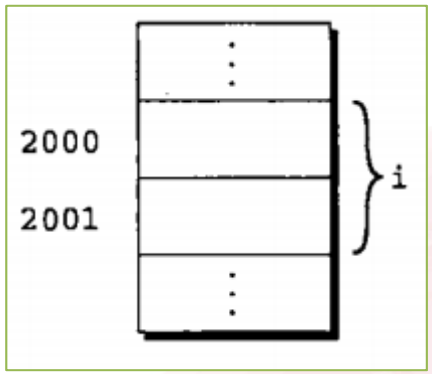


地址	内容
0	01010011
1	01110101
2	01110011
3	01100001
4	01101110
	⋮
n-1	01000011

内存

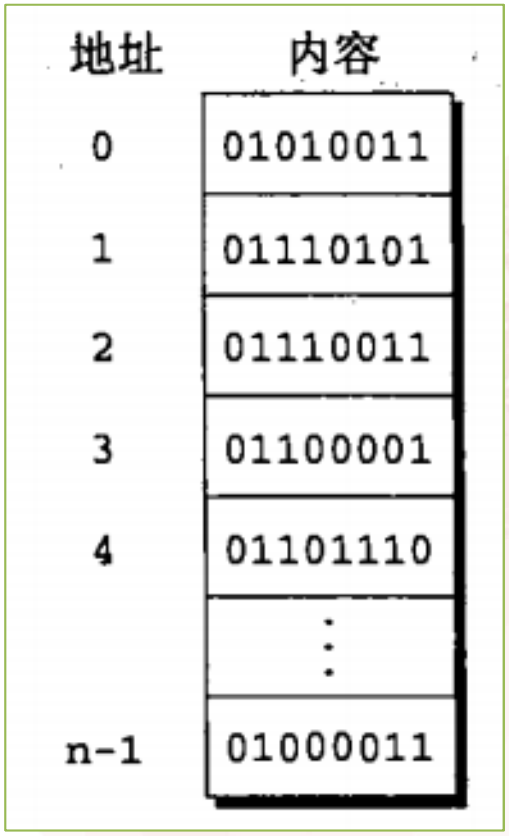
- 程序中用到的变量和函数都转换成二进制位，存储在内存芯片中
- 内存芯片中空间的基本单位是字节（8个bit位）
  - 4G内存芯片，相当与有4G个字节存储空间
  - 存储空间从0开始编号

知识点2：指针就是地址



- 程序中每个变量占一个到多个字节，其第一个字节的位置称为变量的地址
- 虽然用数表示地址，但其取值范围与整数不同，所以我们用新的类型——指针类型存储地址
- 用指针变量p存储整型变量i的地址时，我们称：
  - p指向i
  - p为整型指针

知识点3：空指针



- 值为0的指针称为空指针
  - 用NULL表示
  - 表示指针不指向内存中任何位置

10.2.1 文件类型指针

在C语言中用一个指针变量指向一个文件，其实是指向存放该文件信息的结构体类型变量，这个指针称为文件指针。定义说明文件指针的一般形式为：

FILE *指针变量名;

例如：**FILE *fp;**

表示**fp**是指向**FILE**结构的指针变量，通过**fp**即可找到存放某个文件信息的结构体变量，然后按结构变量提供的信息找到该文件，实施对文件的操作。

因文件指针类型及对文件进行的操作函数都是原型说明都是放到“**stdio.h**”头文件中，因此对文件操作的程序，在最前面都应写一行文件头包含命令： **#include < stdio.h >**。

10.2.2 文件的打开

打开文件使用系统提供的文件打开函数fopen()，其调用的一般形式为：

文件指针名=fopen(文件名，使用文件方式);

- “文件指针名”必须是被说明为FILE类型的指针变量
- “文件名”是被打开文件的文件名
- “使用文件方式”是指文件的类型和操作要求
- “文件名”是字符串常量或字符数组。

例如：

```
FILE *fp;
fp=fopen(“d:\\tc\\source.dat ”, "r");
```

10.2.2文件的打开

文本文件的打开方式列表

打开方式	含义及说明
"r"	以只读方式打开一个文本文件，只允许读数据
"w"	以只写方式打开或建立一个文本文件，只允许写数据
"a"	以追加方式打开一个文本文件，并在文件末尾写数据
"r+"	以读写方式打开一个文本文件，允许读和写
"w+"	以读写方式打开或建立一个文本文件，允许读写
"a+"	以读写方式打开一个文本文件，允许读，或在文件末追加数

如果文件打开成功，文件指针fp指向文本文件信息区

如果文件打开失败，fopen函数会返回一个空指针NULL

例如：以“只读”方式（"r"）打开一个并不存在的文件

10.2.2 文件的打开

为避免因上述原因的出错，常用以下的方法来打开一个文件：

```
if((fp=fopen("d:\\myfile.dat", "w+"))==NULL)
   printf("This file could not be opened !\n");
 else
{
      /* 此处编写打开文件后，对文件读\写的代码 */
 }
```

10.2.2 文件的关闭

文件使用完后，为确保文件中的数据不丢失，应使用文件的关闭函数fclose进行关闭，其调用形式：

fclose(文件指针变量);

函数功能：关闭一个由fopen()函数打开的文件

例如：

fclose(fp);

//用fclose函数使文件指针fp与文件脱离关联

//同时刷新文件输入 / 输出缓冲区

10.3 文件的读写

C语言中提供了多种文件读写的函数：

- 格式化读写函数：fscanf和fprinf
- 字符串读写函数：fgets和fputs
- 字符读写函数：fgetc和fputc

使用fopen函数打开文件成功后，会有属于该文件一个文件读写位置指针，表示文件内部即将要读写的位置。上面的文件读写函数均是指顺序读写，即读写了一条数据后，文件读写位置指针自动指向下一个读写单元。

注意：文件指针和文件内部的位置指针不是一回事

10.3.1文件的格式化输入和输出

1．格式化输入函数fscanf()

函数调用格式为：

fscanf(文件指针，格式字符串，输入表列)

- fscanf函数与前面使用的scanf函数的功能相似，都是格式化读写函数
- 两者的区别在于 fscanf 函数和fprintf函数的读写对象不是键盘和显示器，而是磁盘文件

例如：

```
fscanf(fp, "%d%f", &i, &x);
```

函数的返回值为EOF，表明读错误；否则读数据成功。

End Of File

10.3.1文件的格式化输入和输出

2. 格式化输出函数fprintf()

把格式化的数据写到文件中,调用示例:

```
fprintf(fp, "s=%f, i=%d\n", s, i);
```

- 其返回值为实际写入文件中的字符个数（字节数）；如果写错误，则返回一个负数。
- 格式化的规定与printf()函数相同，所不同的只是fprintf()函数是向文件中写入，而printf()是向屏幕输出

例10-1：读取学籍信息文件，并将其中的女同学信息输出至屏幕

```c
void main()
{
    struct student std[200];
    FILE *fp;
    if((fp=fopen("E:\\studentPhoneNumber.txt", "r"))==NULL) /* 打开文件失败 */
    {
        printf("Cannot open file the file exit!");
        exit(0);   }

    int i = 0;
    while(!feof(fp))
    {
        //学籍卡结构体赋值
        fscanf(fp, "%d %s %s %s", &std[i].stdNumber, std[i].name,
            std[i].sex, std[i].phoneNumber);

        //将所有女同学的信息输出到屏幕
        if(strcmp(std[i].sex, "女") == 0)
            printf("%d %-10s %s %s\n", std[i].stdNumber, std[i].name,
            std[i].sex, std[i].phoneNumber);
        i++;
    }
    fclose(fp);
}
```

```c
struct student
{
    int stdNumber;//学号
    char name[20];//姓名
    char sex[6];//性别
    char phoneNumber[20];//电话号码
};
```

例10-2 将学籍卡信息中所有同学的电话号码前加上地区代号(86)

```
14      struct student std[200];
15      char code[]="86"; //地区代码
16      char newPhoneNumber[20]=""; //临时电话号码
17      FILE *fp1,*fp2;
18      fp1=fopen("E:\\studentPhoneNumber.txt", "r");
19      fp2=fopen("E:\\studentPhoneNumber_new.txt", "w");
20
```

```
21      int i = 0;//学生人数计数变量
22      while(!feof(fp1))
23      {
24          //学籍卡结构体赋值
25          fscanf(fp1, "%d %s %s %s", &std[i].stdNumber, std[i].name,
26              std[i].sex, std[i].phoneNumber);
27
28          strcpy(newPhoneNumber,"\0");//清空临时电话号码
29          strcat(newPhoneNumber,code);//连接地区代码
30          strcat(newPhoneNumber,std[i].phoneNumber);//连接原电话号码
31          strcpy(std[i].phoneNumber,newPhoneNumber);//复制新电话号码
32          i++;
33      }
34      fclose(fp1);
35
36      for(int j=0; j<i;j++)
37      {
38          fprintf(fp2, "%-12d %-10s %-6s %s\n", std[j].stdNumber, std[j].name,
39              std[j].sex, std[j].phoneNumber);
40      }
41
42      fclose(fp2);
```

10.3.2字符串的文件输入和输出

1．字符串输出函数fputs()：

将一个字符串写入到文件中，调用的一般格式如下：

```
fputs(string, fp);
```

- fp是已定义的文件指针变量
- string是要输出的字符串变量
- 该函数的功能是，将字符串string输出到fp所指的文件中
- fputs函数返回0时，表明操作成功；返回非0时，表明写操作失败。

10.3.2字符串的文件输入和输出

2．字符串输入函数fgets()：

从指定的文件中读一个字符串到字符数组中，调用的形式为：

fgets(字符数组名，n，文件指针)；

- n是一个正整数，表示从文件中读出的字符串不超过 n-1个字符。在读入的最后一个字符后加上串结束标志'\0'。
- fgets函数从文件中读取字符直到遇见回车符或EOF为止，或直到读入了所限定的字符数（至多n-1个字符）为止

例如：

fgets(str，n，fp)；

- 从fp所指的文件中读出n-1个字符送入字符数组str中，并在最后加上‘\0’字符。
- 函数读成功返回str指针；失败返回一个空指针NULL

10.3.2字符串的文件输入和输出

例10-3**使用函数**fputs()：**从键盘输入1个字符串，再将其写入文件**

```
1  #include<stdlib.h>
2  #include<stdio.h>
3  void main()
4  {
5      FILE *fp;
6      char ch[80], *p=ch;
7      int n;
8      if((fp=fopen("E:\\myfile.txt", "w"))==NULL)
9      {
10         printf("Cannot open file the file exit!");
11         exit(0);      /* 退出程序 */
12     }
13     printf("input a string:\n");
14     for(n=1;n<=3;n++)
15     {
16         gets(p);              /* 输入一行字符 */
17         fputs(p, fp);         /* 写入该行字符 */
18         fputc('\n', fp);      /* 写入换行符 */
19     }
20     fclose(fp);
21 }
```

例10-4使用fgets()：从文件读入一行字符串

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  void main()
4  {
5      FILE *fp;
6      char ch[50];
7      int n;
8      if((fp=fopen("E:\\myfile.txt", "r"))==NULL)
9      {  printf("Cannot open file the file,exit!");
10     getchar();          /* 暂停 */
11     exit(0);          /* 退出程序 */
12     }
13     while(!feof(fp))
14     {
15         fgets(ch, 50, fp);/* 读取一行字符 */
16         printf("%s", ch);
17     }
18     fclose(fp);
19 }
```

10.3.3 字符的文件输入和输出

1. 字符输出函数fputc()

将一个字符写入到文件中，调用的一般格式如下：

```
fputc(ch, fp);
```

- fp是已定义的文件指针变量
- ch是要输出的字符，它可以是一个字符常量，也可是字符变量
- 该函数的功能是，将字符（ch的值）输出到fp所指向的文件中去
- fputc函数也有返回值，若写操作成功，则返回一个向文件所写字符的值；否则返回EOF（文件结束标志，其值为-1，在stdio.h中定义），表示写操作失败

10.3.3 字符的文件输入和输出

2. 字符输入函数fgetc()

fgetc函数的功能是从指定的文件中读一个字符，函数调用的一般形式为：

字符变量=fgetc(文件指针)；

例如：

ch=fgetc(fp)；

- 其意义是从打开的文件fp中读取一个字符并送入ch中。
- 在fgetc函数调用中，读取的文件必须是以读或读写方式打开的，读取成功返回文件当前位置的一个字符；读错误时返回EOF。

例10-5 字符的文件读写综合示例：复制文件

```c
#include <stdio.h>
#include <stdlib.h>
void main( )
{
    FILE *in,*out;
    char  ch;
    //打开文件
    if((in=fopen("E:\\data1.txt","r"))==NULL
        || (out=fopen("E:\\data2.txt","w"))==NULL)
    {
        printf("无法打开文件\n");
        exit(0);
    }
  //逐一读取并复制源文件中的字符至目标文件，直到遇到源文件结尾
    while(!feof(in))
    {
        ch=fgetc(in);
        fputc(ch,out);
    }
    //关闭文件
    fclose(in);
    fclose(out);
}
```

10.3.4 文件的随机读写

C语言提供了一组文件的随机读写函数，可以将文件读写位置指针定位在所要求读写的地方，从而实现随机读写。

对文件的随机读写是指在文件内部任意对文件内容进行访问，这也就需要对文件进行详细的定位，只有定位准确，才有可能对文件随机访问。下面是有关文件定位的函数一般形式：

```
int fseek (FILE *fp, long offset, int fromwhere);
long ftell(FILE * fp);
int rewind(FILE * fp);
```

10.3.4文件的随机读写

1. fseek()函数

其作用是将文件的读写位置指针设置到特定的位置,调用格式

```
fseek (FILE *fp, long offset, int fromwhere);
```

- fp是文件指针，offset是位移量，fromwhere是位移的起始点
  其的取值有如下三种情况：

  SEEK_SET：（即数值0）表示文件开头

  SEEK_CUR：（即数值1）表示文件指针的现行位置

  SEEK_END：（即数值2）表示文件末尾

例:

```
fseek(fp, 2L, SEEK_ CUR); /* 将位置指针从当前位置向后移2个字节 */
fseek(fp, -2L, SEEK_ END); /* 将位置指针从文件尾向前移2个字节 */
```

其中，数字后加L表示位移量是long型

10.3.4文件的随机读写

2. ftell()函数

返回文件读写位置指针的当前值，这个值是从文件头开始算起到文件指针位置的字节数，返回的数为长整型数；当返回-1时，表明出现错误

3. rewind()函数

用于把文件读写位置指示器移到文件的起点处，成功时返回0；否则，返回非0值

扩展知识点1：字符串输入输出函数异同比较

1. 字符串输出函数综合比较

```
char ch1[] = “ Tsinghua and Peking \0 universities";
```

输出方式	函数名	输出开始条件	输出结束条件	特殊处理
屏幕输出	printf	从首个字符开始	遇空字符（'\0'）	无
	puts	从首个字符开始	遇空字符（'\0'）	再输出一个换行符
文件输出	fprintf	从首个字符开始	遇空字符（'\0'）	无
	fputs	从首个字符开始	遇空字符（'\0'）	无

示例：K_1_printf_puts_fprintf_fputs.cpp

2. 字符串输入函数比较

```
char ch1[] = “ Tsinghua and Peking universities";
```

输入方式	函数名	输入开始条件	输入结束条件	特殊处理
键盘输入	scanf	从首个非空白符开始	遇空白字符	结尾补空字符（’\0’）
	gets	从首个字符开始	遇换行符	结尾补空字符（’\0’）
文件输入	fscanf	从首个非空白符开始	遇空白字符	结尾补空字符（’\0’）
	fgets	从首个字符开始）	遇换行符 或已读入n-1个字符	结尾补换行符（’\10’） 和空字符（’\0’）
备注：空白字符包括空格（ACII码序32）、制表符（\t,码序9），换行符（\n，码序10）等				

示例：K_2_scanf_gets_fscanf_fgets.cpp

扩展知识点2：文件文件的格式

格式可以自由设计

- 对文本文件，最严格的检查，最好每次读入一行字符串，并进行从字符串到整型、浮点数的转换
- 需要用到的函数

  fgets

  atoi 把字符串转换成整型数(ASCII to integer 的缩写)

  atof 把字符串转换成浮点数（ASCII to floating point numbers)
- 需要用到的头文件

```
#include <stdlib.h>
```

示例：**K_3_filefomat.cpp**

综合实例1：挑战者程序

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>

int readStudentInfo(FILE* f, char students[][100]); //输入函数（从文件读入学生信息）
int generateChallenger(char students[][100], int); //处理函数1（产生挑战者）
int MarkTheStudent(); //处理函数2，给学生回答评分
void recordStudentScore(char *, int); //输出函数，将答题结果输出到文件

int main()
{
    char studentInfo[200][100] = {{""}};
    FILE *filename = fopen("E:\\studentInfo.txt","r");
    char *challenger = NULL;
    int count = 0;
    int select = 0;
    int score = 0;

    count = readStudentInfo(filename, studentInfo);//输入函数（从文件读入学生信息）
    select = generateChallenger(studentInfo, count);//处理函数1（产生挑战者）
    challenger = studentInfo[select];//记录挑战者
    score = MarkTheStudent();//处理函数2，给学生回答评分
    recordStudentScore(challenger, score);//输出函数，将答题结果输出到文件

    return 0;
}
```

●第1步：输入，从文件读取学生信息（子函数1）

```
29 int readStudentInfo(FILE *f, char students[][100])
30 {
31     if (f== NULL)
32     {
33         printf("Can not find the file!\n");
34         return 0;
35     }
36
37     int count = 0;
38     while(!feof(f)) //如果没有读到文件结尾，继续读文件
39     {
40         if (fgets(students[count],100,f) != NULL) //读取一行信息（一名同学）
41             count++;                               //学生人数计数加1
42         if (count >= 200) break;                  //超过预设容量，不再读取
43     }
44     fclose(f);                                    //读取结束，关闭文件
45     printf("count=%d",count);                     //输出学生人数
46
47     return count;
48 }
```

●第2步：处理，产生挑战者（子函数2）

```
50 int generateChallenger(char students[][100], int count)
51 {
52     srand((unsigned)time(NULL)); //产生随机数
53     int k = rand() % count;      //归一化
54
55     printf("\n==============================\n\n");
56     printf("挑战者 %s\n",students[k]);   //输出挑战者信息
57     printf("==============================\n");
58     printf("\n");
59     printf("有请挑战者...\n\n");
60     return k;
61 }
```

●第3步：处理，答题评分（子函数3）

```
63 int MarkTheStudent()
64 {
65     char result;
66
67     while(1)
68     {
69         printf("\n挑战者回答是否正确？（T/F）\n");
70         scanf("%c", &result);        //从键盘输入答题成绩
71         if((result == 'T') || (result == 'F')) break;
72         else printf("输入有误！\n");
73     }
74     if(result == 'T')
75     {
76         printf("回答正确！得分100分\n");
77         return 100; //回答正确，100分
78     }
79     else
80     {
81         printf("回答错误！得分50分\n");
82         return 50;//回答错误，50分，出勤奖励
83     }
84 }
```

●第4步：输出，记录答题成绩（子函数4）

```
86  void recordStudentScore(char *challenger, int score)
87  {
88      //创建并打开记录成绩的文件
89      FILE *f = fopen("E:\\studentScore.txt","a");
90
91      if (f == NULL)
92      {
93          printf("Can not find the file!\n");
94          return;
95      }
96
97      //将学生信息及成绩写入文件
98      fprintf(f, "%s得分：%d\n", challenger, score);
99      fclose(f);//关闭文件
100 }
```

综合实例2：学籍卡排序（按姓名拼音）

- 变量和函数声明

```c
#include <stdio.h>
#include <string.h>

int readStudentCard(FILE *f, struct student std[]);
void sortStudentCard(int index[], struct student std[], int count);
void recordSortedCard(int index[], struct student std[], int count);

struct student
{
    char name[20];//姓名
    char sex[6];//性别
    char addr[100];//籍贯
    int age;//年龄
    int score;//成绩
};
```

●主函数

```
18 int main()
19 {
20     struct student studentCard[200];//预设最大容量为200个学生
21     int count = 0;  //实际学生数
22     int index[200]  = {0};//排序后的索引变量   //序号数组
23     //打开文件
24     FILE *fp = fopen("E:\\studentCard.txt","r");
25     //读取学籍信息
26     count = readStudentCard(fp, studentCard);
27     //学籍卡按姓名拼音排序
28     sortStudentCard(index, studentCard, count);
29     //将排序后的学籍卡信息输出到文件
30     recordSortedCard(index, studentCard, count);
31
32     return 0;
33 }
```

●第1步：输入，从文件读取学籍信息（子函数1）

```
35 int readStudentCard(FILE *f, struct student std[])
36 {
37     if (f == NULL)
38     {
39         printf("Can not find the file!\n");
40         return 0;
41     }
42     int i = 0;
43     while(!feof(f))
44     {
45         //学籍卡结构体赋值
46         fscanf(f, "%s %s %d %d %s", std[i].name, std[i].sex,
47             &std[i].age, &std[i].score, std[i].addr);
48         i++;
49     }
50     fclose(f);
51
52     return i;//返回学籍卡包含的学生数量
53 }
```

●第2步：处理，学籍卡排序（子函数2）

```
55 void sortStudentCard(int index[], struct student std[], int count)
56 {
57     int temp;
58
59     for(int i=0; i<count; i++) index[i] = i;      //默认排序 0, 1, 2, 3, ....
60
61     for(int j=0; j<count; j++)
62     {
63         for(int k=0; k<count-j-1; k++)
64         {
65             if(strcmp(std[index[k]].name, std[index[k+1]].name) > 0)
66             {
67                 //可以只记录学籍卡序号信息，不用复制学籍卡结构体
68                 temp = index[k];
69                 index[k] = index[k+1];          //冒泡法排序，调整序号数组
70                 index[k+1] = temp;
71             }
72         }
73     }
74 }
```

- 第3步：输出，排序后的学籍卡输出到文件（子函数3）

```
76  void recordSortedCard(int index[], struct student std[], int count)
77  {
78      //创建并打开学籍卡信息输出文件
79      FILE *f = fopen("E:\\studentCard_new.txt","w");
80      if (f == NULL)
81      {
82          printf("Can not find the file!\n");
83          return;
84      }
85
86      for(int i=0; i<count; i++)
87      {
88          //按照学籍卡的排序逐一输出学籍信息
89          fprintf(f, "%-20s\t %6s %d\t %d\t %-20s", std[index[i]].name, std[index[i]].sex,
90                  std[index[i]].age, std[index[i]].score, std[index[i]].addr);
91          if(i != count-1) fprintf(f, "\n");
92      }
93      fclose(f);
94  }
```