

文件

Wang Houfeng

CCES, PKU

wanghf@pku.edu.cn

内容

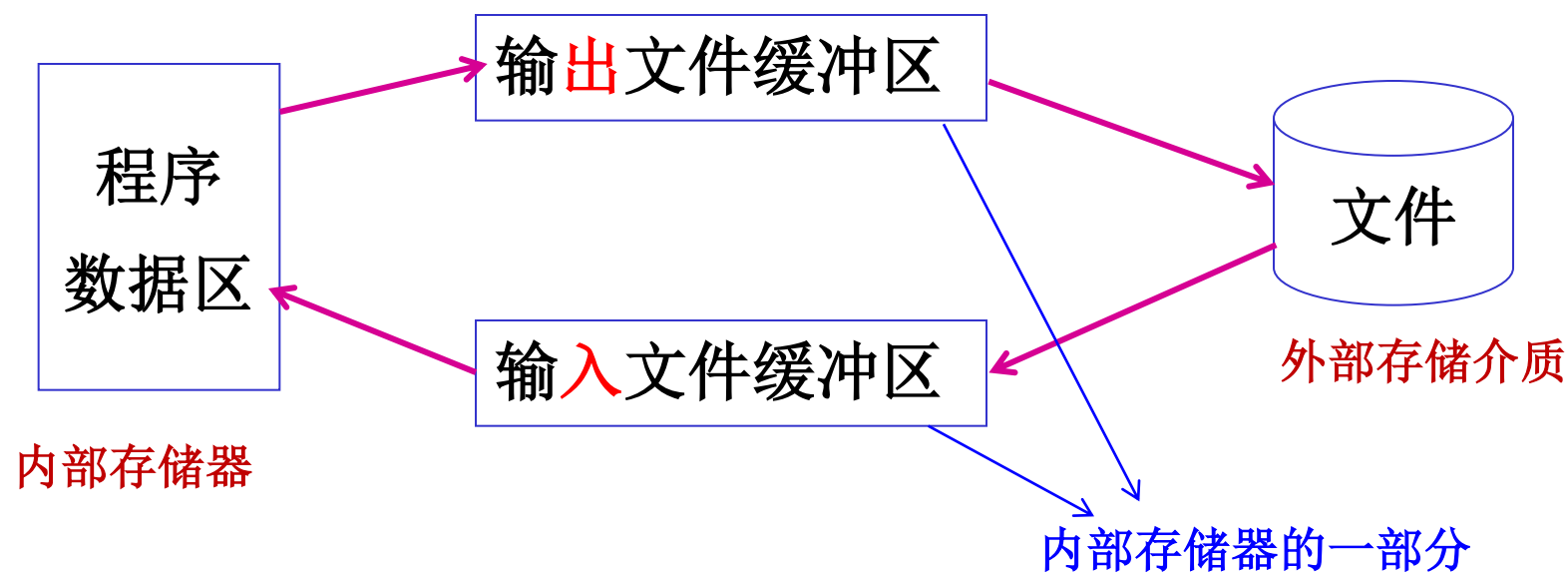
➤ 文件的基本概念

- C语言的文件操作
- C++语言的文件操作
- 一个应用实例

文件概述

文件： 文件指存储在外部介质（如磁盘磁带）上数据的集合。

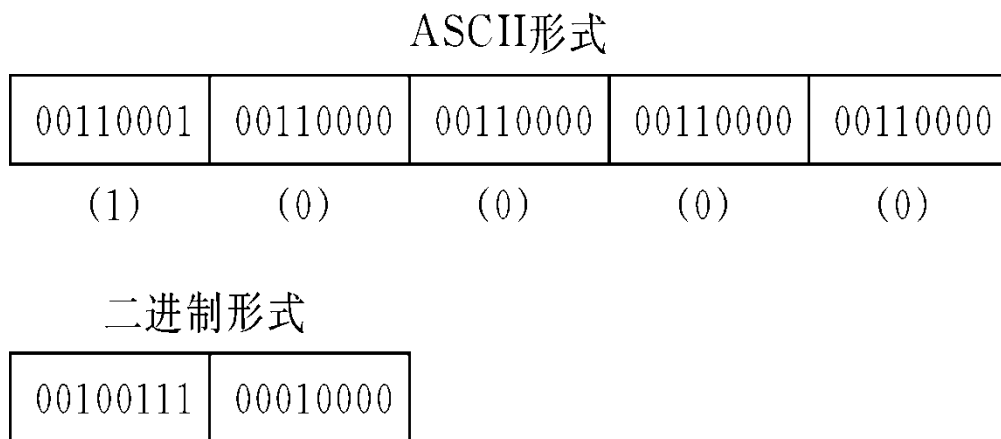
计算机以文件为单位对数据进行操作和管理



文件种类

- **ASCII文件** (**文本文件**): 以ASCII码形式存放在外存 (字符文件)
- **二进制文件**: 把数据以二进制形式存放 (数值文件)

例: 整数10000分别按ASCII码形式和二进制形式存储, 如下图所示:



文件操作

- 文件的相关操作

- ① 创建文件;
- ② 打开文件;
- ③ 对文件进行操作;
 - 读
 - 写
 - 查
- ④ 关闭文件;

■ 打开文件

在内存中建立缓冲区

- ◆ 如打开成功，**打开函数**返回一个内存地址值，由一个文件指针变量接收;
- ◆ 如内存不可建立缓冲区，则打开失败，**打开函数**返回NULL;

■ 关闭文件

将文件内容送到磁盘，并从内存中清除数据，及时释放内存空间，以保证文件安全;

计算机的设备也以文件形式管理，如标准输入设备（键盘）和标准输出设备（显示器）都以**特殊的文件**表示

内容

➤ 文件的基本概念

➤ C语言的文件操作

- C++语言的文件操作
- 一个应用实例

C语言文件类型指针

`FILE *fp;`

fp是一个指向FILE类型(系统定义的)的指针变量。可以使fp指向某一个文件，从而访问该文件。如果有 n 个文件，一般应设 n 个指针变量，使它们分别指向 n 个文件，以实现文件的访问（建立内存中的缓冲区）。

打开文件

- 调用函数：fopen

FILE *fp;

fp = fopen (文件名, 使用文件方式);

- ①需要打开的文件名，也就是准备访问的文件的名字(用字符串表示);
 - ②使用文件的方式 (“读” 还是 “写” 等);
 - ③让哪一个指针变量(如 fp)指向被打开的文件。
- fopen 的返回值:不成功时为0(NULL), 成功时为非 0;

注意：stdin 表示键盘(标准输入), stdout 表示显示器(标准输出). 不必再打开

使用文件的方式

"r"	(只读)为输入打开一个文本文件
"w"	(只写)为输出打开一个文本文件
"a"	(追加)向文本文件尾增加数据
"rb"	(只读)为输入打开一个二进制文件
"wb"	(只写)为输出打开一个二进制文件
"ab"	(追加)向二进制文件尾增加数据
"r+"	(读写)为读/写打开一个文本文件
"w+"	(读写)为读/写建立一个新的文本文件
"a+"	(读写)为读/写打开一个文本文件
"rb+"	(读写)为读/写打开一个二进制文件
"wb+"	(读写)为读/写建立一个新的二进制文件
"ab+"	(读写)为读/写打开一个二进制文件

一个例子

```
FILE *fp;
```

```
fp=fopen("c:\\cprogram\\test.txt", "r");
```

表示按 “读” 的方式打开目录:

C:\cprogram 下的文件 test.txt;

注: “\” 必须用转义符号 \

文件的关闭

- 调用函数 `fclose`

`fclose`（文件指针）；

函数功能：

使文件指针变量不指向该文件，也就是文件指针变量与文件“脱钩”，此后不能再通过该指针对原来与其相联系的文件进行读写操作，并将缓冲区的数写入文件。

返回值：

关闭成功返回值为 0；否则返回EOF(-1)

从文件中读1个字符: fgetc (fp)

```
#include <stdio.h>
void main( void )
{
    FILE *stream;
    char buffer[81];
    int i, ch;
    if( (stream = fopen( "fgetc.c", "r" )) == NULL )
        exit( 0 );
    ch = fgetc( stream );
    for( i=0; (i < 80 ) && ( !feof( stream ) ); i++ ) //检查结尾
    {
        buffer[i] = (char)ch;
        ch = fgetc( stream ); //再从文件fgetc.c中读一个符号
    }
    buffer[i] = '\0';
    printf( "%s\n", buffer ); //从显示器上输出
    fclose( stream );
}
```

向文件中写一个字符

- **fputc(ch, fp);** // 向 fp 中写入一个字符 ch, **返回 EOF 表示出现错误!**

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

```
void main()
```

```
{
```

```
    char strptr[] = "This is a test of fputc!!\n";
```

```
    char *p;
```

```
    p = strptr;
```

```
    while( (*p != '\0') && fputc( *(p++), stdout ) != EOF );
```

```
} // stdout 是标准输出文件(对应着显示器)
```

内容

- 文件的基本概念
- C语言的文件操作
- **C++语言的文件操作**
- 一个应用实例

C++的文件操作

- ifstream 类：支持从磁盘文件的**输入**
- ofstream 类：支持向磁盘文件的**输出**
- fstream 类：支持磁盘文件的**输入输出**
- 先建立流类对象（类似前面的FILE），再打开文件
 - ofstream **outfile**;
 //定义ofstream 类(输出文件流类)对象outfile
 - **outfile.open**("C:\\Temp\\myfile.dat", **ios::out**);
 //调用文件流的成员函数open,
 //使文件流与myfile.dat 文件按**输出形式**建立关联

cin和 cout 是两种特殊的文件，分别对应于键盘和显示器操作

C++中打开文件的方式

文件打开方式	功 能
<code>ios::in</code>	以输入方式打开文件
<code>ios::out</code>	以输出方式打开文件(默认方式)
<code>ios::app</code>	以追加方式打开文件，将所有输出写入文件末尾
<code>ios::ate</code>	打开一个已有文件，文件指针指向文件末尾
<code>ios::trunc</code>	如果文件存在，删除文件原有内容，否则创建新文件
<code>ios::binary</code>	以二进制方式打开文件，默认时为文本文件
<code>ios::nocreate</code>	打开已有文件，不存在，则打开失败
<code>ios::noreplace</code>	如文件不存在则建立新文件，文件已存在，则操作失败
<code>ios::in ios::out</code>	以输入/输出方式打开文件，文件可读可写
<code>ios::out ios::binary</code>	以二进制方式打开一个输出文件
<code>ios::in ios::binar</code>	以二进制方式打开一个输入文件

使用模式

- 如何判定一个文件是否被成功创建或打开
 - 如果打开操作失败，则流对象的值为0

ofstream outfile;

outfile.open("f1.dat");  默认为输出， ios::out

if (!outfile)  打开失败

{

 cerr<<"open error!"<<endl;

 return 1;

}

//操作文件

outfile.close();  关闭文件

C++文件流的错误状态

函 数	功 能
is_open	流对象是否与一个打开的文件相联系，若是则返回 true，否则返回 false
bad	如果进行了非法操作返回 true，否则返回 false
good	操作成功返回 true，否则返回 false
eof	若到文件尾则返回 true，否则返回 false
fail	操作失败返回非 0 值
clear	设置内部错误状态，如果用默认参量调用，则清除所有错误位
rdstate	返回当前错误状态

```
if(outfile.is_open()==0)
{
    cerr<<"open error!"<<endl;
    return 1;
}
```

ASCII文件的读/写方式之一

- 使用 “>>”运算符和 “<<”运算符读/写数据
 - ofstream outfile; outfile与cout用法类似
 - ifstream infile; infile与cin 用法类似

```
outfile << 1 << " " << 2 << endl;
```

```
// 输出 1, 2
```

```
infile >> a >> b;
```

```
//把infile对应文件中的2个值输入到a, b中
```

outfile 与 cout 类似

infile 和 cin 类似

cin 与 cout 的分别对应键盘和显示器

例1：将整数存入文件

- 从键盘输入10个整数，并保存到磁盘文件中，再从文件中读出数据，放入数组。

```
#include<iostream>
#include <fstream>
using namespace std;
int main( )
```

```
{  int a[10];
    ofstream outfile("f1.dat");
    if (!outfile)
    {
        cerr<<"open error!"<<endl;
        return 1;
    }
```



```
ofstream outfile;
outfile.open("f1.dat");
```

```

cout<<"enter 10 integer numbers: "<< endl;
for(int i = 0;i < 10;i++)
{
    cin>>a[i];                //从键盘读入
    outfile <<a[i]<<" ";    //写入文件，空格分开整数
}
outfile.close();           outfile 与 cout 的使用类似
ifstream infile("f1.dat");   infile 和 cin 的使用类似
if (!infile)
{
    cerr<<"open error!"<<endl; return 1; }

for(i = 0; i < 10; i++)
{
    infile>>a[i];             //从文件读入，跳过空格
    cout << a[i]<<" ";        //输出到显示器
}
cout<<endl; infile.close();
return 0;
}

```

字符读写类似

例2：将二维表存入文件

	Mon	Tue	Wen	Thu	Fri
John	10	3	11	9	7
Mike	11	15	4	3	7
Tom	9	10	12	2	9
Ben	12	5	7	14	12

- `char Name[10][5]; int Sum[10][5];`
- `int num;`

```
void savedata(int num, char Name[ ][5], int Sum[ ][5])
{
    ofstream outfile("data.txt");
    outfile << num << " ";
    for (int i = 0; i < num; i++)
    {
        outfile << Name[i] << " ";
    }
    for (int n = 0; n < num; n++)
        for(int j = 0; j < 5; j++)
            outfile << Sum[n][j] << " ";
}
```

写入文件data.txt的数据格式如下：

```
4 John Mike Tom Ben 10 3 11 9 7 11 15 4 3 7 .....
```

```
int loaddata(char Name[][5], int Sum[][5])
```

```
{
```

```
    int num;
```

```
    ifstream infile("data.txt");
```

```
    infile >> num;
```

```
    for (int i = 0; i < num; i++)
```

```
    {
```

```
        infile >> Name[i];
```

```
    }
```

```
    for (int n = 0; n < num; n++)
```

```
        for(int j = 0; j < 5; j++)
```

```
            infile >> Sum[n][j];
```

```
    return num;
```

```
}
```

从文件data.txt中读入数据，
格式如下：

	Mon	Tue	Wen	Thu	Fri
John	10	3	11	9	7
Mike	11	15	4	3	7
Tom	9	10	12	2	9
Ben	12	5	7	14	12

ASCII文件的读/写方式之二

- 使用文件对象调用put, get, getline 等成员函数, 进行数据读写;

函 数	功 能
输入函数	
read()	从文件读取特定数量的字节 主要用于二进制输入
get()	读取一个字符或一串字符
getline()	读取特定数量的字符, 或直到\n 为止
peek()	读取下一个字符, 但不删除当前字符
输出函数	
open()	把流与一个特定的磁盘文件关联起来。需要指定打开模式
write()	向文件写入一定数量的字节 主要用于二进制输出
put()	向文件写入一个字符
close()	关闭与一个输出文件流关联的磁盘文件

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
```

例1：将字符串存入文件

```
{
    char s[100];
    ofstream outf;
    outf.open("dialogue.dat");
    if(!outf)
    {
        cout<<"can't open dialogue.dat"<<endl;
        abort();
    }
    outf<<"How are you?\n"; //将若干字符串写到文件中
    outf<<"I'm fine,thank you, and you?\n";
    outf<<"I'm very well.\n";
    outf.close();
    ifstream inf("dialogue.dat");
    while(!inf.eof())           //判断是否到文件结尾
    {
        inf.getline(s,sizeof(s)); //从文件中读一行字符串到数组
        cout<<s<<endl;
    }
    system("pause");
    return 0;
}
```

例2：文本文件的拷贝

```
#include <fstream>
```

```
int main()
```

```
{  char ch, f1[10], f2[10];
```

```
    cout<<"Enter the source file name: ";
```

```
    cin.getline(f1, 10);
```

```
    ifstream infile(f1);
```

```
    if (!infile)
```

```
    {    cerr<<"open error!"<<endl;
```

```
        return 1;
```

```
    }
```

第一步：打开源文件；

从键盘上输入文件名

建立文件联系（打开文件）

```
cout<<"Enter the destination file name: ";
```

```
cin.getline(f2, 10);
```

```
ofstream outfile(f2);
```

```
if (!outfile)
```

```
{
```

```
    cerr<<"open error!"<<endl;
```

```
    return 1;
```

```
}
```

```
while(infile.get(ch))
```

```
    outfile.put(ch);
```

```
infile.close();
```

```
outfile.close();
```

```
return 0;
```

```
}
```

第二步：输入目标文件名，
并打开目标文件；

第三步：拷贝（读入并输出）

第四步：关闭文件（切断与缓冲
区的联系）

二进制文件的创建与打开

```
ofstream outfile("file.dat", ios::binary);  
if(!outfile)  
{  
    cerr<<"open error!"<<endl;  
    abort( );//退出程序  
}  
//文件操作  
outfile.close( );
```

二进制文件的读/写

- 对二进制文件的读写主要用 **istream** 类的成员函数 `read` 和 `write` 来实现:
 - `read(char *buffer,int len);`
 - 读取`len`是字节数，放在字符数组`buffer`中
 - `buffer`指向内存中的一段存储空间，`len`是字节数
 - 如果文件中没有足够的数量，则遇到文件结束为止
 - `write(const char * buffer,int len);`
 - 从数组`buffer`写`len`个字节到文件

二进制文件的操作举例

- 读写结构体数据并显示

```
void display(struct student *stud); // 显示
```

```
void writeToFile(struct student *stud); // 输出
```

```
void readFromFile(struct student *stud); // 输入
```

```
struct student
```

```
{ char name[20];
```

```
    int num;
```

```
    int age;
```

```
    char sex;
```

```
};
```

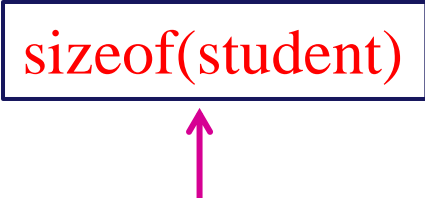
写-读-显示结构体数据

```
int main( )
{
    student stud1[3]={ "Li",1001,18,'f',
                        "Fang",1002,19,'m',
                        "Wang",1004,7, 'f' };

    student stud2[3];
    writeToFile(stud1);
    readFromFile(stud2);
    display(stud2);
    return 0;
}
```


写结构体数据到文件

```
void writeToFile(struct student *stud)
{
    ofstream outfile("student.dat", ios::binary);
    if (!outfile)
    {
        cerr<<"open outfile error!"<<endl;
        exit(1);
    }
    for (int i = 0; i < 3; i++)
        outfile.write((char *)(stud+i), sizeof(stud[i]));
    outfile.close();
}
```



A purple arrow points from the `sizeof(stud[i])` argument in the `outfile.write` call to a purple-bordered box containing the text `sizeof(student)`.

从文件中读结构体数据

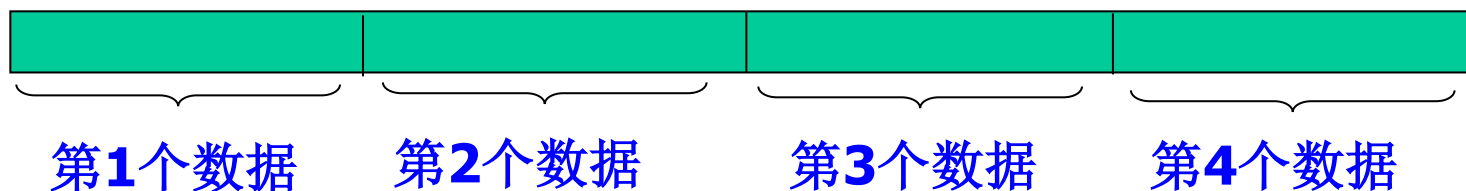
```
void readFromFile(struct student *stud)
{
    ifstream infile("student.dat", ios::binary);
    if (!infile)
    {
        cerr<<"open outfile error!"<<endl;
        exit(1);
    }
    for (int i = 0; i < 3; i++)
        infile.read((char *)(stud+i), sizeof(student));
    infile.close();
}
```

显示结构体数据

```
void display(struct student *stud)
{
    for (int i = 0; i < 3; i++)
    {
        cout << "No. " << i+1 << endl;
        cout << "Name: " << stud[i].name << endl;
        cout << "num: " << stud[i].num << endl;
        cout << "age: " << stud[i].age << endl;
        cout << "sex: " << stud[i].sex << endl;
    }
}
```

随机文件操作

- 顺序文件优点：操作简单；缺点：依赖位置，要把被访问数据的前面的数据全部读过去
- 随机访问要求数据具有相同的结构，例如，结构体



- 具有以下功能：
 - 能够从文件的任意地方读
 - 可以往文件中加记录

随机访问文件

- 打开一个新文件，如果原来有内容，则清空
 - `ofstream fout("a.dat", ios::out);`
- 打开一个已有文件，并往里面写数据,文件**可读可写**
 - `fstream file("a.dat", ios::in | ios::out);`
- 判断是否到了文件末尾
 - `if(!(file.eof()))` 到末尾返回NULL，实际是0
- `file.seekp((n - 1) * sizeof(student))`
 - 将指针定位到从文件头开始的第n个记录的地方

修改文件中某记录的内容

- 例如：
 - 一个文件保存若干数据，将其中第n个数据用变量data的值替换
- 两种方式
 - 顺序文件操作，将文件内容读到数组，修改数组的数据，再将数组全部写回文件。
 - 随机文件操作，将指针指到第n个位置，再直接写入data //替换掉第n个数据

顺序操作修改内容

```
int main()
{
    int data[100], i = 0, n ;
    ifstream fin("data.txt"); //先按输入文件读入
    while(!(fin.eof()))
        fin >> data[i++];
    fin.close();
    cout << "which data need to update" << endl;
    cin >> n;
    cout << "give the number to update" << endl;
    cin >> data[n-1]; //替换
    ofstream fout("data.txt", ios::out); //再按输出文件写入
    for(int j = 0; j < i - 1; j++)
        fout << data[j] << " ";
    fout.close();
    return 0;
}
```

随机操作修改内容

```
int main()
{    int n,data;
    cout << "which data need to update" << endl;
    cin >> n;
    cout << "give the number to update" << endl;
    cin >> data;
    fstream file;
    file.open("data.txt", ios::out|ios::in); //按输入输出打开
    file.seekp( (n-1) * sizeof(int)); //定位在哪修改
    file.write((char *)&data, sizeof(int));
    file.close();
    return 0;
}
```


文件操作小结（1）

- 文件打开要用文件类来定义对象
 - `ifstream`（输入），`ofstream`（输出），`fstream`（输入+输出）
- 操作前打开文件，有两种方法：
 - 1. `ifstream file1("myfile");` //直接打开
 - 2. `ofstream file2;` //先定义，
`file2.open("myfile");` //再调用打开函数
- 打开文件有不同属性
 - `file1.open("myfile")` //缺省指文本文件
 - `file1.open("myfile", ios::binary)` //以二进制形式打开
- 文件的指针可以在文件流中移动
 - 顺序移动，读/写一个数据，指针移到下一个数据的位置
 - 命令移动，`file1.seekp(字节数)`，正数向后移，负数往回移

文件操作小结（2）

- `file.seekp`可以有两个参数
 - `file.seekp (int , ios::beg, ios::end, ios::cur)` //任选一个，或不选
 - 第一个 (int) 是偏移量，第二个是相对位置，相对头部、结尾、当前三种移动类型。
- `file.seekp (n-1) :`
 - 缺省第二个参数，相对于开始的第n个记录
- `file.seekp(-1, ios::cur):`
 - 把指针从当前位置回移一个位置。
- `file.seekp(n, ios::cur):`
 - 把指针在当前位置向下移n位。
- `file.seekp(-1, ios::end):`
 - 把指针从文件末尾向回移一位。

文件操作小结（3）

- 文件主要是功能是读写。
 - 使用>> , << , `getline`等有格式的输入输出方法时就用文本方式
 - 使用`read`、`write`、`seekp`等进行数据块的批量输入和输出时就用二进制方式
- `file.get()`, `file.put()`, 每次读写一个字节, 两种文件格式都可以使用。
- `read`和`write`有两个参数, 一个是字符型的地址, 一个是读/写的字节数。
- 如果要写入的数据类型不是字符型, 需要强制转换:
`fin.read((char *)(&data), sizeof(data));`

文件操作小结（4）

- 如果用处理二进制数据的方式处理就用二进制打开
 - `ifstream fin("in.dat", ios_base::binary | ios_base::in);`
 - `fin.read((char *)(&data), sizeof(data));`
- 如果用处理文本信息的方式处理(类似于使用`cin` 和`cout`的方式)就用文本方式打开，比如：
 - `ifstream fin("in.txt");`
 - `fin >> data.x >> data.y >> data.z >> data.name;`
- 如果希望直接点击就看到文件的内容，用文本文件
- 如果是为了保存数据，两种格式都可以。

```
struct data {  
    int x, y, z;  
    string name;  
} data;
```

内容

- 文件的基本概念
- C语言的文件操作
- C++语言的文件操作
- 一个应用实例

综合应用（简单了解）

- 成绩管理
- 要求功能：
 - 1、输入数据(顺序)
 - 2、统计功能(按班号)
 - 3、列表功能(按学号)
 - 4、查询功能(按学号)
 - 5、删除功能(按学号)

简单的成绩管理系统

- 系统的框架：
 - 用一个文件存放学生记录数据，所有操作都基于该文件
 - 提供用户界面，让用户可以任意选择不同功能
 - 将各个功能模块写成函数，并由统一界面管理。

学号	姓名	班级	期中	作业	期末	总成绩	
1001	***	1	78	80	82		
1002	***	2	85	86	66		
1003	***	3	80	91	85		
...		
1180	***	4	90	??	77		

管理界面 & 主要数据结构

管理界面

欢迎使用成绩管理程序

1. 输入记录
2. 统计成绩
3. 全部显示
4. 查询成绩
5. 删除记录
6. 退 出

主要数据表示，放在 **statement.h** 中

```
struct student
{
    int    ID;
    char   name[10];
    int    age;
    int    class1;
    float  middle ; //期中
    float  hw; //作业
    float  end; // 期末
    float  score; //综合
    char   tag; //删除标记
}
```


主要功能与模块

```
void append()    //追加记录
{ ; }
void listall()   //显示所有学生情况
{ ; }
void statistic() //按班级统计平均成绩
{ ; }
void dele()      //删除记录（打标签）
{ ; }
void search()    //查找学生情况
{ ; }
```

```

#include <iostream>
#include <fstream>
#include <iomanip>
#include "statement.h"
using namespace std;
int main()
{ int n=0;
  while (n!=6)
  { system("cls"); //清屏
    cout <<" \n";
    cout <<" |  欢迎使用      | \n";
    cout <<" |  1. 追加记录      | \n";
    cout <<" |  2. 统计分数      | \n";
    cout <<" |  3. 全部显示      | \n";
    cout <<" |  4. 查询成绩      | \n";
    cout <<" |  5. 删除记录      | \n";
    cout <<" |  6. 退出          | \n";
    cout <<" \n";
    cout <<"请输入数据选择 1/2/3/4/5/6: " << endl;
    cin >> n;
    cin.get(); //跳过换行符
  }
}

```

```

switch(n)
{
    case 1: append();break;
    case 2: statistic();break;
    case 3: listall();break;
    case 4: search();break;
    case 5: dele();break;
    default: break;
}
return 0;
}

```

追加记录

```
void append()
{   student st;
    system("cls");
    ofstream FILE;
    FILE.open("data.txt",ios::app);
    if(!FILE)
    {
        cout << "Can't open the file" << endl;
        exit(1);
    }
```

```
FILE.open("data.txt",ios::app);
打开一个输出，在文件末尾添加新数据
```

```

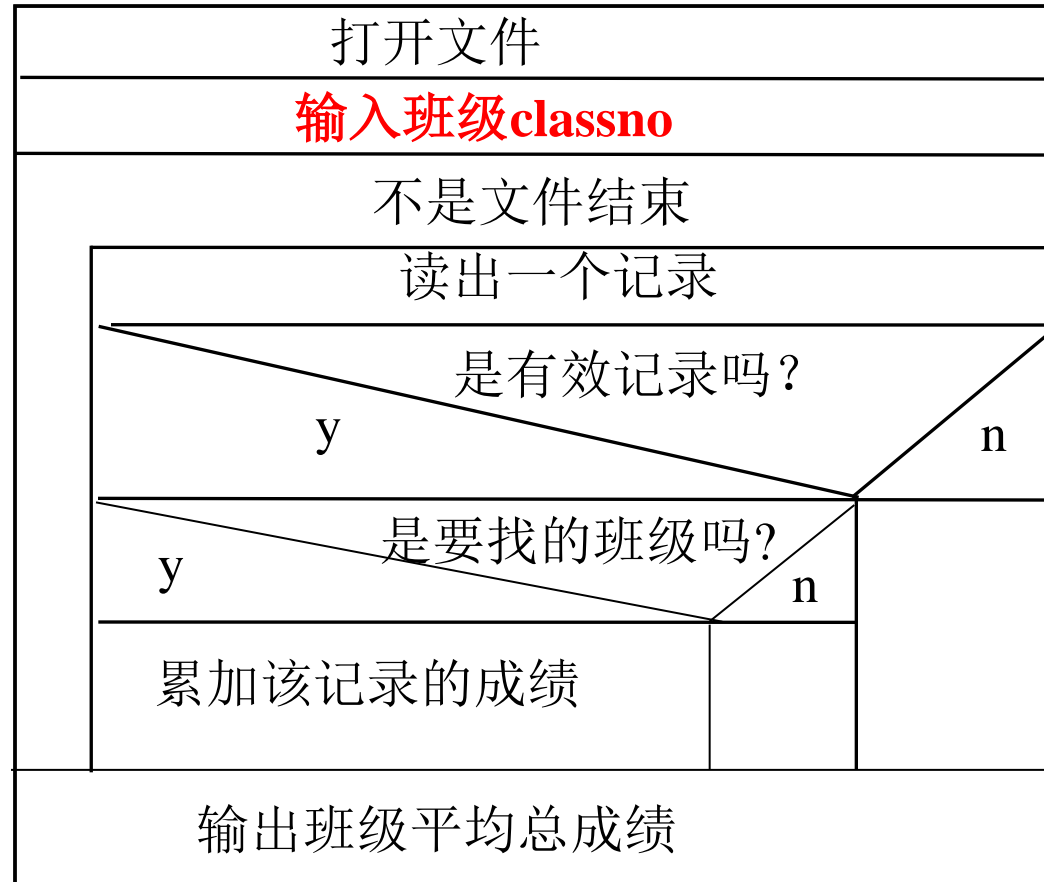
for (;;)
{
    cout << "Please input ID "; cin>>st.ID; cin.get();
    if (st.ID == -1) break; // 当输入学号为-1时结束
    cout << "Please input name ";
    cin.getline(st.name,10);
    cout << "Please input age ";
    cin >> st.age;
    cout << "please intput class ";
    cin >> st.class1;
    cout << "please input midterm, homework, final ";
    cin >> st.middle >> st.hw >> st.final;
    st.score = st.middle * 0.2 + st.hw * 0.1 + st.final * 0.7;
    st.tag='1';
    FILE.write((char *)&st, sizeof(student));
}
FILE.close();
}

```

显示所有学生成绩

```
void listall()
{ system("cls"); student st;
  ifstream FILE("data.txt");
  if(!FILE)
  { cout << "Can't open the file" << endl; exit(1); }
  cout << "学号 姓名 年龄 班级 期中 作业 期末 综合" << endl;
  cout << "===== " << endl;
  while(!FILE.eof())
  { if (FILE.read((char *)&st, sizeof(student)))
    { if(st.tag == '1')
      { cout << setw(5) << st.ID << setw(12) << st.name << setw(4)
        << st.age << setw(6) << st.class1 << setw(8) << fixed
        << setprecision(2) << st.middle << setw(6) << st.hw
        << setw(6) << st.final << setw(6) << st.score << endl;
        cout << "-----" << endl;
      } } }
  cin.get(); FILE.close();
}
```

统计成绩



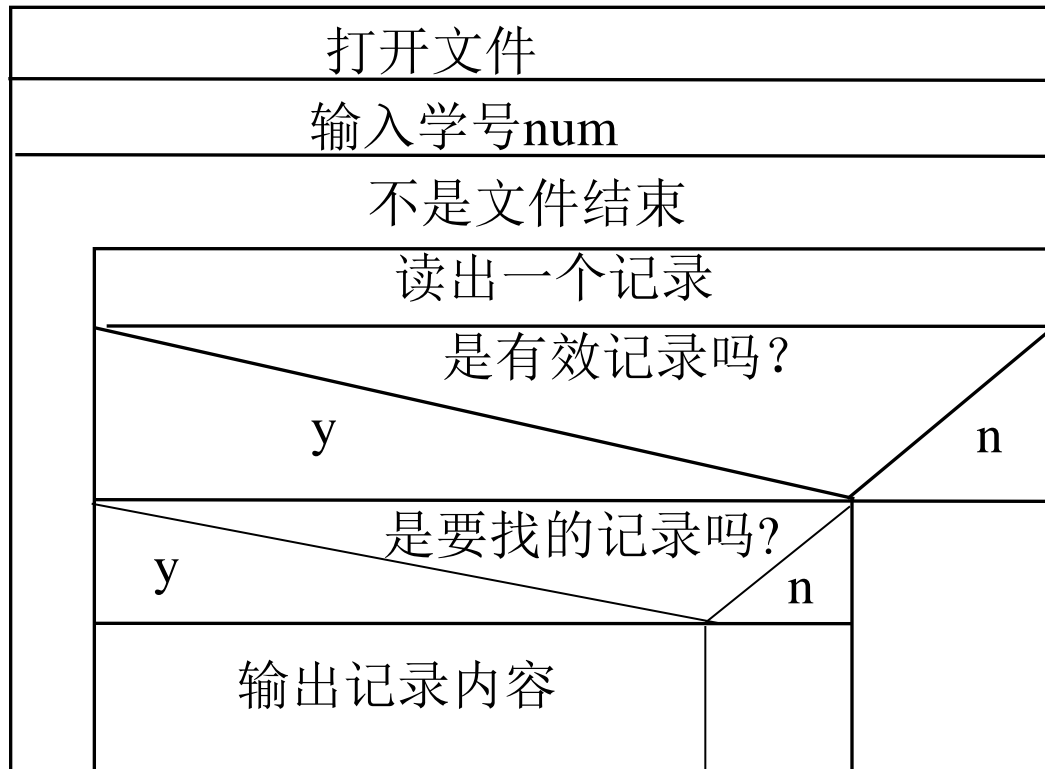
```
void statistic()
{
    student st;
    int classno,count=0;
    float sum = 0.0, aver = 0.0;
    ifstream FILE("data.txt");
    if(!FILE)
    {
        cout << "Can't open the file" << endl;
        exit(1);
    }
}
```

```

cout << "which class to be calculated:" ;
cin >> classno;
while(!FILE.eof())
{
    if (FILE.read((char *)&st,sizeof(student)))
    {
        if((st.tag=='1') && (st.class1==classno))
            {   count++;   sum += st.score;   }
    }
}
aver = sum / count;
cout << " The class " << classno
    << "'s average is " << fixed << setprecision(2)
    << aver << endl;
cin.get();
FILE.close();
}

```


查询记录



```
void search()
{
    student st;
    int choice,ID = -1 ,count=0, classno = -1,flag = 2;
    char name[10]=" ";

    ifstream FILE("data.txt");
    if(!FILE)
    {
        cout << "Can't open the file" << endl;
        exit(1);
    }
    cout << "by which key do you want to search " << endl;
    cout << "1--by ID" << endl;    //按学号
    cout << "2--by name" << endl;  //按名字
    cout << "3--by class" << endl; //按班级
    cin >> choice;
    cin.get();
}
```

```

switch(choice)
{
    case 1 :
        cout << "please input a ID " ;
        cin >> ID; break;
    case 2 :
        cout << " please input a name ";
        cin.getline(name,10); break;
    case 3 :
        cout << " please input class ";
        cin >> classno; break;
    default:
        cout << "error" << endl;
}
cout<<“学号 姓名 年龄 班级 期中 作业 期末 综合 ” <<endl;
cout<< "===== " <<endl;

```

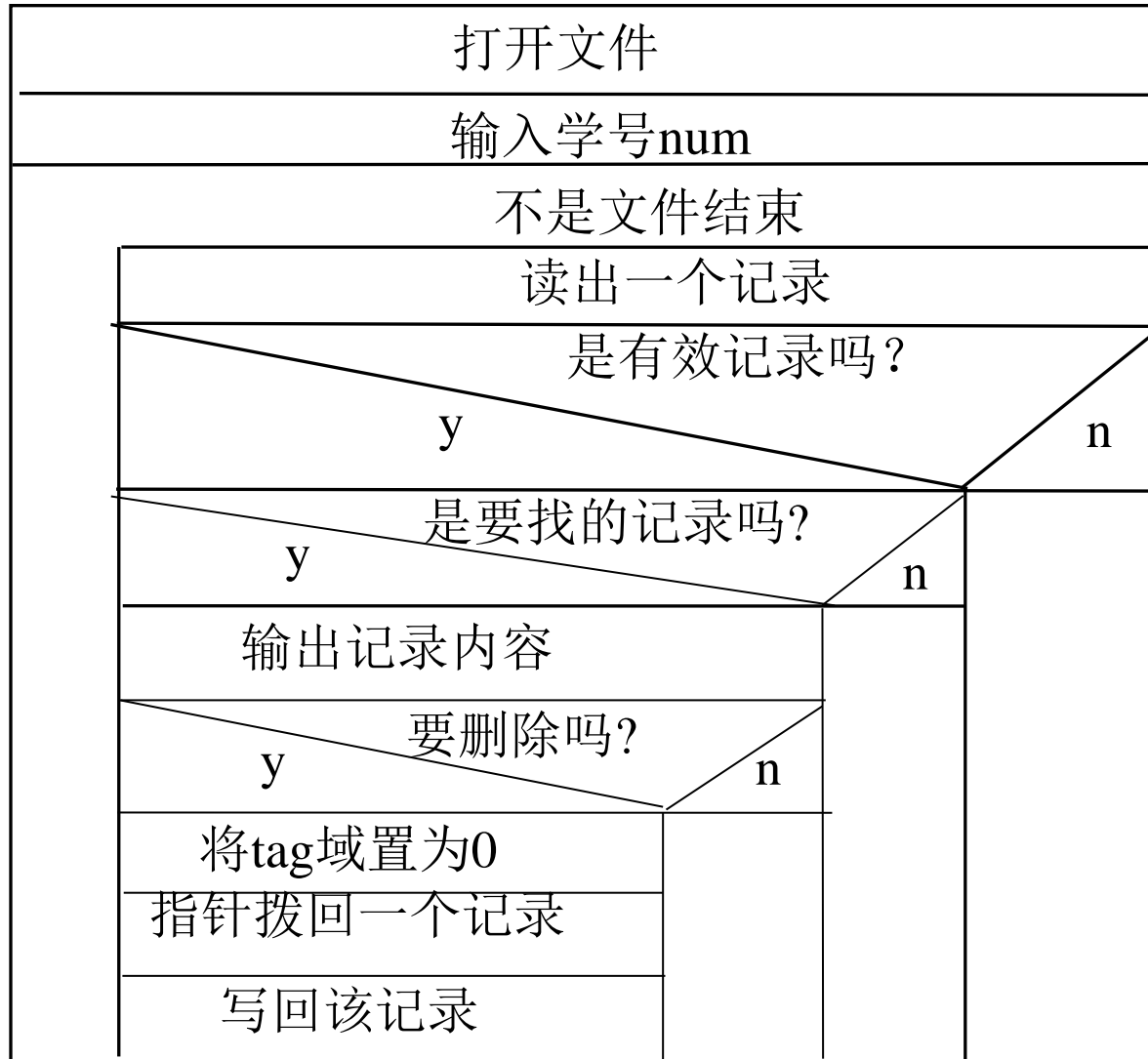
```

while(!FILE.eof())
{
    if (FILE.read((char *)&st,sizeof(student)))
    {
        if(st.tag=='1')
        {
            count ++;
            flag = strcmp(st.name,name);
            if (!flag||st.ID==ID||st.class1==classno)
            {
                显示记录;
                cout<< "-----" << endl;
            }
        }
    }
}

if (count ==0) cout << " not found" << endl;
    cin.get();    FILE.close();
}

```

删除记录



```
void dele()
{
    student st;          int ID, count = 0;
    char ch;
    fstream FILE("data.txt",ios::in|ios::out);
    if(!FILE)
    {
        cout << "Can't open the file" << endl;
        exit(1);
    }
    cout << "输入要删除的学号(-1 exit): ";
    cin >> ID; cin.get();
}
```

```

while(ID != -1)
{ FILE.seekp(0); //定位在文件最开始
  while (!FILE.eof())
  { if (FILE.read((char *)&st,sizeof(student))) //读入一个记录
    { count ++;
      if ((st.ID == ID) && (st.tag == '1'))
      { 显示记录;  }
      cout << "confirm (Y/N) ";cin >> ch;cin.get();
      if (ch == 'y' || ch == 'Y')
      { st.tag='0';
        FILE.seekp((count - 1) * sizeof(student));
        FILE.write((char *)&st,sizeof(student)); break;
      }
    }
  }
  if (count == 0 )
  { cout << "not found" << endl;}
  cout<<"输入要删除的学号: "; cin >> ID; cin.get();
}
FILE.close();
}

```