第八章 文件

- 引述
- 输入输出流和文件概述
- 文件操作
- 流的格式化输出
- 程序举例
- 本章小结

8.1 引述

C语言提供了文件这一数据类型，文件用于对磁盘文件进行读写操作；将文件看成有序的字符或字节序列，由一个一个字符或字节数据顺序组成，称为**字符流或字节流**

输入对象cin和输出对象cout实际上也是流

磁盘文件的读写操作与cin、cout的使用类似

任务8.1 从键盘输入40位学生的信息，将其存储到磁盘文件

任务8.2 从任务8.1建立的磁盘文件中读取学生信息，将其存放到结构数组并输出。

算法分析：

要解决这两个问题，主要是要解决学生信息的存储和读取问题，即如何使用C语言，将一些信息存入到一个磁盘文件中，然后从磁盘文件中读取信息并存放到数组中。为此需要用到C语言的文件这一数据类型。

8.2 输入输出流和文件概述

- 输入输出流概述
- 文件概述

8.2.1 输入输出流概述

通过C语言的I/O系统提供的统一的接口输入输出的信息，即流，有两种类型：

文本流(text stream)又称文字流或字符流

- 文本流是一个个的字符，用换行符表示一行的结束。
- 文本流文件的信息直接可见
- 利用文本流只能读写文本文件

二进制流(binary stream) 又称字节流

- 二进制流则由一系列字节组成
- 字节流文件信息一般不能直接可见
- 用于对声音、图像等非文本文件进行读写

cin和cout实际上都是文本流。cin控制字符序列流入内存；cout控制字符序列从内存流向输出设备。

8.2.2 文件概述

文件通常指磁盘文件，从输入输出的观点看，键盘、显示器、打印机等各种外部设备，也可以认为是文件。

C文件的读写均需先建立与此文件相关联的流，常被称为**流式文件**；

C语言提供了很多函数，这些**流式函数**将文件或数据项作为单个字符（或字节）构成的数据流来处理。

流式函数在读写流式文件的数据时采用了**缓冲存储区域**，可以一次传输大量数据，提高了输入输出的效率。

※ 缓冲存储技术用于解决内存、外存读写速度差异很大的问题。新型计算机一般都有很大的**高速缓存**，以加快读写速度。

8.3 文件操作

- 文件的打开和关闭
- 字符流文件的输入与输出
- 字节流文件的输入与输出
- 文件的随机访问

8.3.1 文件的打开和关闭

文件处理头文件fstream定义了文件处理相关的流。

cin和cout在定义时已经与相应的标准输入输出设备建立好了关联，可直接使用。

fstream定义的流需要进行显式说明，建立打开和关闭的关联关系。

C语言使用文件，必须先打开再读写，读写完成后必须再关闭。

fstream定义了三个流类型：

ifstream：输入流，用于读文件

ofstream：输出流，用于写文件

fstream：输入输出流，用于读写文件

```
ifstream fin(filePath);                    //打开字符文件用于读
ifstream fin(filePath,ios::binary); //打开字节文件用于读
ofstream fout(filePath);                  //打开字符文件用于写
ofstream fout(filePath,ios::binary);//打开字节文件用于写
```

也可以用open函数打开文件，如打开filePath用于以字节流方式进行读：

```
ifstream file;
file.open(filePath, ios::binary);   //打开字节文件用于读
```

※open是流类型的成员函数。

文件打开模式

模式参数	模式含义
ios::in	读
ios::out	写
ios::app	从文件末尾开始写
ios::binary	二进制模式
ios::nocreate	打开文件时，若文件不存在，不创建
ios::noreplace	打开文件时，若文件不存在，则创建
ios::trunc	打开一个文件，然后清空内容
ios::ate	打开一个文件时，将位置移动到文件尾

参数用|（按位或）操作符进行连接

```
fstream file;
file.open(filePath, iso::in|ios::out);
// 打开字符文件用于读写
```

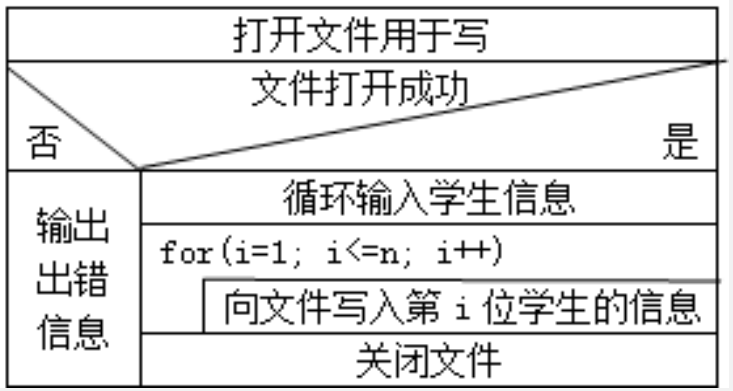
测试文件是否正确打开：打开文件不一定会成功，如果正确打开，流变量将不为空。

```
if(!file)
{
    // 文件打开失败
}
```

关闭文件：取消流变量与外部文件之间的关联。

```
file.close();
// 取消流变量file与外部文件之间的关联
```

8.3.2 字符流文件的输入和输出



1. 字符流文件的输出

与cout类似，使用插入运算符<<

任务8.1 算法

核心程序——打开文件，读入学生信息

```
ofstream fout("student.txt");// 打开字符文件用于写
if(!fout)
{
    cout<<"文件打开失败！"<<endl;
    return -1;                          // 返回错误代码-1
}
for(i = 1; i <= n; i++)                 // 循环输入学生信息
{
    cout<<"请输入第"<<i<<" 位学生信息："<<endl;
    stuList[i] = readStudent();
}
```

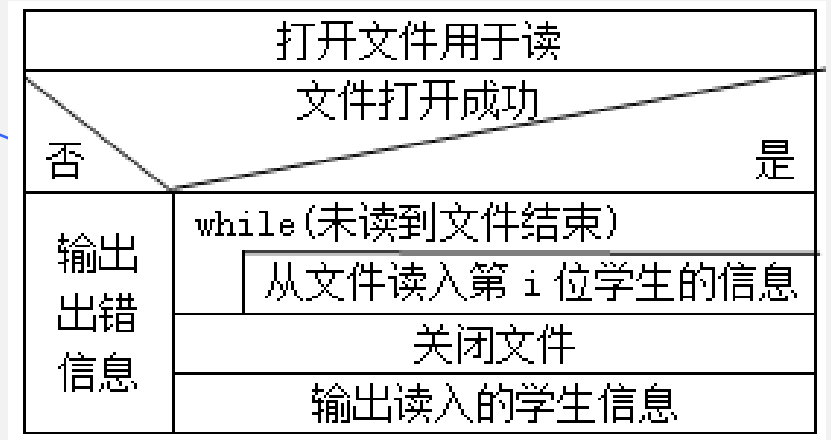
任务8.1 核心程序——学生信息写入文件

```
    for(i = 1; i <= n; i++) // 循环将学生信息写入文件
    {
        fout<<"姓名 "<<stuList[i].name<<endl;
        fout<<"学号 "<<stuList[i].no<<endl;
        fout<<"性别 "
            <<(stuList[i].sexy==1?"男":"女")<<endl;
        fout<<"生日 "<<stuList[i].birthday.year<<"-"
            <<stuList[i].birthday.month<<"-"
            <<stuList[i].birthday.day<<endl;
        fout<<"身高 "<<stuList[i].height<<endl;
        fout<<"体重 "<<stuList[i].weight<<endl;
        fout<<"电话 "<<stuList[i].telephone<<endl;
        fout<<"E_mail "<<stuList[i].e_mail<<endl;
        fout<<"QQ号 "<<stuList[i].qq<<endl;
    }
    fout.close();                    // 关闭文件
```

字符流文件的输出
生成的文本文件示例

```
姓名 wang xiao er
学号 201042001
性别 男
生日 1993-1-1
身高 1.71
体重 71
电话 13612345678
E_mail wangxe@scu.edu.cn
QQ号 12345678
姓名 zhang shan pi
学号 201042002
性别 男
生日 1993-2-2
身高 1.82
体重 82
电话 13623456789
E_mail zhangsp@scu.edu.cn
QQ号 23456789
```

2. 字符流文件的输入



任务8.2 算法分析

与cin类似，使用提取运算符>>

读入一行，使用成员函数getline（fin不会读入空格）

```
fin.getline(字符数组名, 最大字符长度);
fin.getline(stuList[i].name, 20);
```

//从文件读取当前行的最多20个字符到stuList[i]的name成员中

成员函数eof()判断整个文件的读取是否已经完成，如果读取完成，这个函数将返回真

任务8.2 核心程序——从文本文件中读取数据

```
while(!fin.eof()) // 循环读取文件，至读到文件结束为止
{
    fin>>propertyName;              // 读入姓名行的属性名称
    if(strcmp(propertyName, "姓名") != 0)
    {                   // 未读到姓名，转向读入下一行内容
       continue;
    }
    n++;                // 开始读取第n位同学的信息
    fin.getline(stuList[n].name, 20);
    fin>>propertyName;// 读入学号行的属性名称
    fin.getline(stuList[n].no, 20);
    fin>>propertyName;// 读入QQ行的属性名称
    fin.getline(stuList[n].qq, 20);
```

```
姓名 wang xiao er
学号 201042001
性别 男
生日 1993-1-1
身高 1.71
体重 71
电话 13612345678
E_mail wangxe@scu.edu.cn
QQ号 12345678
姓名 zhang shan pi
学号 201042002
性别 男
生日 1993-2-2
身高 1.82
体重 82
电话 13623456789
E_mail zhangsp@scu.edu.cn
QQ号 23456789
```

任务8.2 核心程序——从文本文件中读取数据

```
    fin>>propertyName;                  // 读入性别行的属性名称
    fin.getline(propertyValue, 20);
    if(strstr(propertyValue, “男”))
    {
        stuList[n].sexy = 1;
    }
    else
    {
        stuList[n].sexy = 0;
    }
    fin>>propertyName;                  // 读入生日行的属性名称
    fin.getline(propertyValue, 20);
    stuList[n].birthday = str2date(propertyValue);
}
```

任务8.2 核心程序——输出数据

```
    fin.close();                              // 关闭输入文件
    cout<<"从文件读入的学生信息："<<endl;
    writeStudentInfoTitle();                  // 输出学生信息标题行
    for(i = 1; i <= n; i++)                   // 逐行输出学生信息
    {
        writeInStudent(stuList[i]);
    }
    writeStudentInfoTail();                   // 输出学生信息尾部行
    cout<<"\t\t共有"<<n<<" 位学生"<<endl;
```

8.3.3 字节流文件的输入和输出

1. 字节流文件的输出

任务8.1 算法分析：使用字符流方式比较复杂。

使用字节流打开文件时，需要使用ios::binary模式申明。

write输出流文件成员函数：用于向文件写入变量的内容，使用方法：

fout.write((char *)(&变量名), sizeof(变量名));

前一参数为要写入文件的变量的地址，需转换为char *类型

后一参数为变量所占用的字节数

任务8.1 算法

以字节流方式打开一个文件，命名为student.dat，用于写

然后依次输入学生信息，

再使用write成员函数将其写入文件

1. 字节流文件的输出：任务8.1 核心程序——写入数据至文件

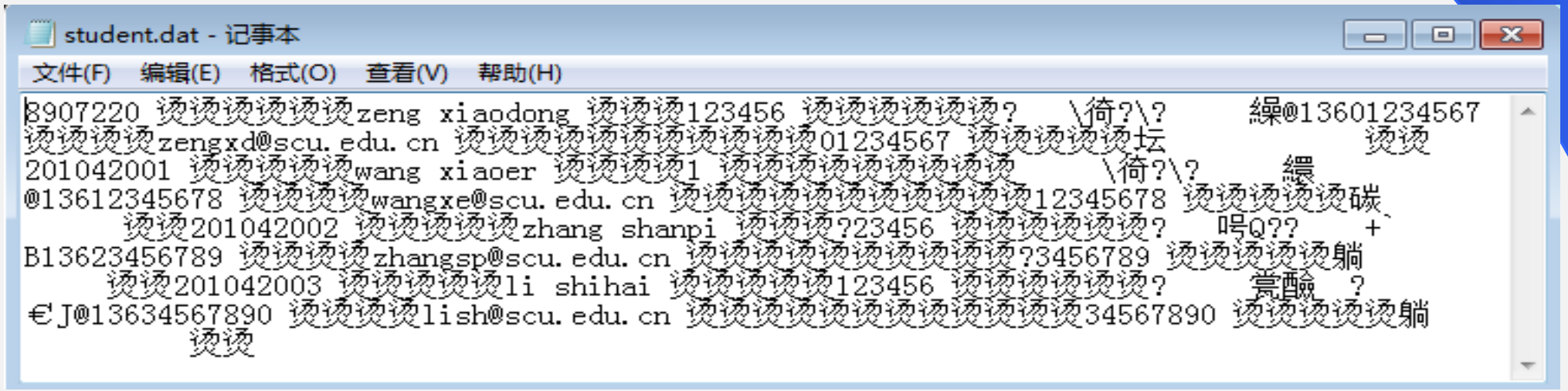
```
ofstream fout("student.dat", ios::binary);  // 以字节流方式打开文件student.data用于写
if(!fout)
{
    cout<<"文件打开失败！"<<endl;
    return -1;  // 返回错误代码-1
}
for(i = 1; i <= n; i++)                    // 循环输入学生信息
{
    cout<<"请输入第"<<i<<" 位学生信息："<<endl;
    stuList[i] = readStudent();
}
for(i = 1; i <= n; i++)
   {       // 循环将学生信息写入文件
      fout.write((char *)(stuList+i), sizeof(StudentInfo));
   }
fout.close();// 关闭文件
```

任务8.1 算法分析

fout.write((char *)(stuList+i), sizeof(StudentInfo)); **一条语句就完成了信息的写入操作，更加简单；**

而且就算学生信息结构体发生了变化，比如增加或减少了属性，这条语句也可以不作修改，程序的可重用度更高

文件是不能直接读的，用记事本打开所示内容如下。



2. 字节流文件的输入

任务8.2 算法分析

打开文件时，需要使用ios::binary模式申明

read成员函数，用于文件中读取数据

```
fin.read((char *)(&变量名), sizeof(变量名));
```

任务8.1 算法

可以将指定文件中读入学生信息至数组的程序封装为函数，放入头文件student.h中。此函数有三个参数

存储学生信息的结构数组，

读入到的学生人数，设为传引用

文件名

任务8.2 核心程序

```
void freadStudents(StudentInfo stuList[], int &nStudent, char fileName[])
（p298页函数定义）
{  // 首先打开文件
    ifstream fin(fileName, ios::binary);
    // 以字节流方式打开文件fileName用于读
    if(!fin)
    {
        cout<<"文件打开失败！"<<endl;
        return -1;  // 返回错误代码-1
    }
  //然后从文件中读取数据
```

```
while(!fin.eof())// 循环读取文件，至读到文件结束为止
   {
     nStudent++;     // 当前学生序号增1
     fin.read((char *)(stuList+nStudent),
              sizeof(StudentInfo));
     if(strlen(stuList[nStudent].name) < 1)
     {                // 如果读入的学生姓名为空，则忽略之
       nStudent--;
     }
   }
   fin.close();      // 关闭输入文件
}
```

3. 字符流文件和字节流文件的比较

使用字节流方式对记录信息进行读写时，操作比字符流方式更简单一些，文件安全性更高，所以，对文件进行操作时，一般使用字节流方式进行读写。

当要求文件可以直接查看时，使用字符流方式读写。

8.3.4 文件的随机访问

例8.1 从任务8.2写入的学生信息文件中，读入所有奇数序号的学生信息，即依次读入第1、3、5、...位学生的信息。

算法分析：

每个流式文件都维持了一个**位置指针**，指示着下一次读写操作的位置。这个**位置是从0开始，按字节进行计数**的。如果将位置移动到指定位置，即可实现随机读写。

成员函数seekg用于设置输入流文件的位置指针，用法：

fin.seekg(移偏量);或fin.seekg(移偏量, 移动的起始位置);

偏移量为正表示向后移动位置指针，偏移量为负表示向前移动。

移动的起始位置有三种模式：

ios::beg表示文件的起始位置；

ios::cur表示文件的当前位置；

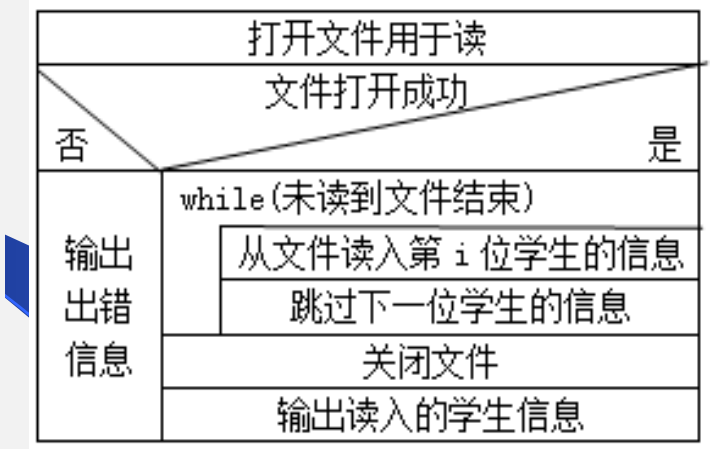
ios::end表示文件流的结束位置。

如果忽略移动的起始位置参数，使用ios::beg（从起始位置开始）

算法分析：本例希望间隔着读入学生信息，只需每次读入学生信息后，从当前位置向后移动位置指针，偏移量为一个学生记录所占用的字节数。

例8.1 核心程序

```
        while(!fin.eof())
    {                   // 循环读取文件，至读到文件结束为止
        n++;            // 当前学生序号增1
        fin.read((char *)(stuList+n),
                    sizeof(StudentInfo));
        if(!fin.eof()) // 如果文件尚未读完，
        {               // 将文件位置指针后移一个学生记录
            fin.seekg(sizeof(StudentInfo), ios::cur);
        }
        if(strlen(stuList[n].name) < 1)
        {               // 如果读入的学生姓名为空，则忽略之
            n--;
        }
    }
```



seekp成员函数，用于实现对输出文件的位置指针的改变

使用方式与输入文件流的seekg函数类似。

seekp和seekg，一般用于字节流文件。

因为文本文件（即字符流文件）在表示同一类型的信息时，长度通常是不固定的，所以计算位置时往往会发生混乱而达不到预期的目的。

C语言文件操作详解（查阅网上资料）

文件的打开操作 fopen 打开一个文件
文件的关闭操作 fclose 关闭一个文件
文件的读写操作 fgetc 从文件中读取一个字符
fputc 写一个字符到文件中去
fgets 从文件中读取一个字符串
fputs 写一个字符串到文件中去
fprintf 往文件中写格式化数据
fscanf 格式化读取文件中数据
fread 以二进制形式读取文件中的数据
fwrite 以二进制形式写数据到文件中去
getw 以二进制形式读取一个整数
putw 以二进制形式存贮一个整数
文件状态检查函数 feof 文件结束
ferror 文件读/写出错
clearerr 清除文件错误标志
ftell 了解文件指针的当前位置
文件定位函数 rewind 反绕
fseek 随机定位

文件的打开

1. 函数原型

```
FILE *fopen(char *pname,char *mode)
```

2. 功能说明

按照**mode** 规定的方式，打开由**pname**指定的文件。

```
//定义一个名叫fp文件指针
   FILE *fp;
//判断按读方式打开一个名叫test的文件是否失败
   if((fp=fopen（"test"，"r"）) == NULL)//打开操作不成功
{
   printf("The file can not be opened.\n");
   exit(1);//结束程序的执行
}
```

*文件的关闭

函数原型：`int fclose(FILE *fp);`

*文件的读写操作

如：从文件中读取一个字符

函数原型

```
int fgetc(FILE *fp);
```

如：写一个字符到文件中去

函数原型

```
int fputc(int ch,FILE *fp)
```

功能说明：把**ch**中的字符写入由**fp**指出的文件中去。

8.4 流的格式化输出

- 格式控制函数
- 行内格式的控制

任务8.3 编写函数将多位学生的信息以表格方式输出

```
从文件读入的学生信息:
    ┆-----学号-----┆-------姓名-------┆-----生日------┆-性别-┆-身高-┆-体重-┆
    ┆ 8907220      ┆ zeng xiaodong    ┆ 1971年12月13日┆  男  ┆ 1.71 ┆ 75.0 ┆
    ┆ 201042001    ┆ wang xiaoer      ┆ 1980年 1月 1日┆  男  ┆ 1.71 ┆ 71.0 ┆
    ┆ 201042002    ┆ zhang shanpi     ┆ 1993年 3月 2日┆  男  ┆ 1.82 ┆ 74.2 ┆
    ┆ 201042003    ┆ li shihai        ┆ 1993年 3月 3日┆  男  ┆ 1.63 ┆ 53.0 ┆
    ┆--------------┆------------------┆---------------┆------┆------┆------┆
                共有 4 位学生
```

输出表格标题行
for(i=1; i<=n; i++)
在一行中输出第 i 位学生的信息
输出表格尾部行
输出学生人数

算法分析：

设多位学生的信息存储在一个学生信息结构数组中，用一个整型变量存储学生人数，两者一并传入函数。

在函数中，首先输出表头信息，然后依次以上图所示格式输出学生信息，最后输出表格的结尾行及学生人数信息即可。

关键：如何控制输出格式，使不同长度的信息能以规范的格式进行输出。

一般使用cout的成员函数setf、width、precision等设置

8.4.1 格式控制函数

cout.setf(控制标志)；// 用以设置输出流的格式控制标志

控制标志	标志含义
ios::dec	将整数以10进制输出
ios::hex	将整数以16进制输出
ios::oct	将整数以8进制输出
ios::showbase	显示进制标志（0为8进制，0x为16进制）
ios::left	左对齐
ios::right	右对齐
ios::internal	符号左对齐，数值右对齐，中间用空格分隔
ios::fixed	浮点数以小数方式输出
ios::scientific	浮点数以科学计数式输出
ios::showpoint	浮点数输出小数点（如1将会输出1.0）
ios::showpos	正数前输出+

```
cout.unsetf();                  // 用于取消控制格式
```
一般同类的输出格式设置后，需要先执行复位，再设置别的格式
```
cout.unsetf(ios::dec);          // 取消以10进制方式输出整数
cout.setf(ios::oct);            // 设置以8进制方式输出整数
cout.setf(ios::right);              //上述进制要先取消再设置否则不起作用
cout.width(5);
cout.fill('0');
cout<<123<<“ ”<<123456;
// 输出00123 123456
cout.precision(4);//控制输出流显示浮点数的数字个数。
cout<<1.23<<" "<<1.23456<<1234567.8<<endl;
// 输出1.230 1.235 1.235e+006
cout.precision(4);
cout.setf(ios::fixed);
cout<<1.23<<" "<<1.23456<<1234567.8<<endl;
// 输出1.2300 1.2346 1234567.8000
```

8.4.2 行内格式控制

cout<<单行控制标志<<输出对象；

控制标志	标志含义
dec/hex/oct	将整数以10/16/8进制输出
showbase/unshowbase	显示/不显示进制标志（0为8进制，0x为16进制）
left/right	左/右对齐
internal	符号左对齐，数值右对齐，中间用空格分隔
fixed/ scientific	浮点数以小数方式/科学计数式输出
showpoint	强制浮点数输出小数点（如1将会输出1.0）
noshowpoint	不强制浮点数输出小数点（如1.0将会输出1）
showpos/noshowpos	正数前输出/不输出+
setfill(字符)	设置填充字符
setw(整数)	设置字符串宽度
setprecision(整数)	控制输出流显示浮点数的数字个数，如果和fixed合用，则用于控制小数点右边的位数。

```
cout<<showbase<<oct<<123<<" "<<noshowbase
    <<hex<<123<<endl;
cout<<showpos<<dec<<123<<" "<<noshowpos<<123<<endl;
cout<<showpoint<<123.0<<" "<<noshowpoint<<123.0<<endl;
cout<<setprecision(4)<<fixed<<1.23<<" "<<1.23456
    <<1234567.8<<endl;
```

输出结果：

```
0173 7b
+123 123
123. 123
1.2300 1.2346 1234567.8000
```

行内格式控制比成员函数的使用更简单，使用更普遍

任务8.3 核心程序

```
cout.setf(ios::left);    // 设置内容左对齐
cout.fill(' ');          // 位数不足时，以空格填充
cout<<showpoint;         // 输出浮点数时显示小数点
cout<<setprecision(3);   // 浮点数精度为3位
cout<<"    | ";
cout<<setw(13)<<stuList[i].no<<"| ";
cout<<setw(17)<<stuList[i].name<<"| ";
cout.unsetf(ios:left); //复位
cout.setf(ios::right);  // 设置内容右对齐
cout<<setw(4)<<stuList[i].birthday.year<<"年"
    <<setw(2)<<stuList[i].birthday.month<<"月"
    <<setw(2)<<stuList[i].birthday.day<<"日|";
cout<<"  "<<(stuList[i].sexy==1?"男":"女")<<"  "<<"| ";
cout<<setw(4)<<stuList[i].height<<" | ";
cout<<setw(4)<<stuList[i].weight<<" |";
cout.unsetf(ios::right);// 取消设置内容右对齐
cout<<endl;
```

8.5 程序举例

- 存储课程信息
- 读取课程信息

例8.2 编写函数将一个课程信息数组的所有课程记录存入到文件中。课程数组、文件名均作为函数的形参传入。假设在course.h中已经定义了名为CourseInfo的结构描述了课程信息，其定义如习题4.6所示。

算法分析

本例的函数需要传入课程信息数组名、课程的门数 、文件名

在函数中，打开相应文件，并将数组元素依次写入文件即可。

使用字节流文件存储课程信息。

算法与任务8.1的算法类似

例8.2 存储课程信息 核心程序

```
ofstream fout(fileName, ios::binary);
// 以字节流方式打开文件fileName用于写
if(!fout)
{
    return -1;                    // 文件打开出错
}
for(i = 1; i <= nCourse; i++)
{ // 循环将课程信息写入文件
    fout.write((char *)(courseArray+i),
                    sizeof(CourseInfo));
}
fout.close();                     // 关闭文件
```

例8.3 编写函数从例8.2存储的课程文件中读取课程信息至课程信息数组。课程数组和文件名均作为函数的形参传入。假设在course.h中，已经定义名为CourseInfo的结构描述了课程信息，其定义如习题4.6所示。

算法分析

按题意，与例8.2类似，本例的函数需要传入数组名、课程门数和文件名三个参数。与例8.2的区别在于，例8.2中，课程门数的值只需传入函数即可，因此使用的是普通的传值调用；而本例需将课程门数回传到调用函数中，因此，可使用传引用的方式调用。

本例算法与任务8.2算法类似，只是将核心的读文件语句用函数进行了封装而已。

例8.3 读取课程信息 核心程序

```
ifstream fin(fileName, ios::binary);
// 以字节流方式打开文件fileName用于读
nCourse = 0;        // 初始化课程门数为0
while(!fin.eof()) // 循环从文件读入课程信息
{
    nCourse++;     // 课程门数增1
    fin.read((char *)(courseArray + nCourse),
             sizeof(CourseInfo));
    if(strlen(courseArray[nCourse].name) < 1)
    {              // 如果课程名为空，则忽略当前课程
        nCourse--;
    }
}
fin.close();       // 关闭文件
```

8.6 本章小结

1. 流是一个字符或字节序列，无论是输入操作还是输出操作，都是控制这个字符或字节序列。流可以使用相同的操作方式对不同的数据类型进行不同的操作，如cin均使用>>连接不同的输入对象，C语言自行决定对不同类型的数据进行不同的输入操作，因此十分方便。

2. C的流式文件是有序的字符流或字节流，文件必须先打开，再使用。使用完成后，必须关闭。C的字符流文件的输入输出方式与cin和cout的使用方式基本相同；字节流文件可以将一个结构变量整体存储或读入，当存储的数据较复杂时，使用它比使用字符流文件更加方便。

3. 可以使用流的格式控制成员函数或控制标识符，对数据的输出格式如数据的占位宽度、对齐方式、浮点数的精度、整数的进制等进行设置。这些成员函数和控制标识是定义在头文件iomanip中的，使用前需先使用预处理命令#include。