Ch-13

输入输出流

主要内容

- C++的输入输出流
- 标准输出流
- 标准输入流
- 对数据文件的操作与文件流
- 字符串流

13.1 C++的输入输出流

C++通过I/O类库来实现丰富的输入输出功能。 I/O类库中的类称为流类，用流类定义的对象称为流对象。

C++的输入输出流是指由若干个字节组成的字节序列，这些字节中的数据按顺序从一个对象传送到另一个对象。

C++的输入输出流会在内存中为每个数据流开辟一个缓冲区，所有输入输出的数据都要先存到缓冲区中，然后才会被处理。

- C++输出时，把要输出的数据送到程序的输出缓冲区中，形成cout流，当缓冲区满或遇到endl后，就将缓冲区中的全部数据送到显示器上；
- C++输入时，首先把键盘的输入数据存入键盘的缓冲区中，当按回车键后，键盘缓冲区中的数据送到程序的输入缓冲区中，形成cin流，然后用“>>”从输入缓冲区中提取数据给变量赋值。

□ C++的I/O流类库

C++提供了专门用于输入输出的类，这些类通过继承方法建立起来，并组合成流类库。这些类有两个基类：ios类和streambuf类，所有其他流类都是从他们直接或间接派生出来的。

类名	作用	声明的头文件
ios	抽象基类	iostream
istream	通用输入流和其他输入流的基类	iostream
ostream	通用输出流和其他输出流的基类	iostream
iostream	通用输入输出流和其他输入输出流的基类	iostream
ifstream	输入文件流类	fstream
ofstream	输出文件流类	fstream
fstream	输入输出文件流类	fstream
istrstream	输入字符串流类	strstream
ostrstream	输出字符串流类	strstream
strstream	输入输出字符串流类	strstream

□ iostream头文件

- iostream头文件包含了对输入输出流进行操作所需的基本信息。该文件不仅对ios，istream，ostream，iostream，istream_withassign，ostream_withassign，iostream_withassign等流类进行了声明，而且定义了4种流对象。

流对象的定义形式（以cout为例）

ostream_withassign cout(stdout)

如果在程序中包含iostream头文件，则在程序开始运行时，会自动调用相应的构造函数构造出这4个标准流对象，把流和标准设备进行关联。

对象	含义	对应设备	对应的类	对应的C标准文件
cin	标准输入流	键盘	istream_withassign	stdin
cout	标准输出流	屏幕	ostream_withassign	stdout
cerr	标准错误流	屏幕	ostream_withassign	stderr
clog	标准错误流	屏幕	ostream_withassign	stderr

- **在iostream头文件的istream和ostream类中分别有一组成员函数对位移运算符“<<”和“>>”进行重载，以便用于标准类型数据的输入和输出。例如：**

```
ostream operator <<(int);

ostream operator <<(char);

ostream operator <<(char *);
```

◆ **“<<”和“>>”不能直接用于用户自定义类型数据的输入输出。如果需要，则程序员自己应该定义新的运算符重载函数。**

13.2 标准输出流

标准输出流是流向标准输出设备（显示器）的数据

13.2.1 cout、cerr和clog流

1、cout（console output）流

- cout流对象是容纳数据的载体，其中的数据是由流插入运算符“<<”顺序加入的；
- 用“cout <<” 输出数据时，编译系统会自行判断数据的类型并调用与之匹配的运算符重载函数；
- cout流在内存开辟一个缓冲区存放流中的数据，当遇到endl时，立即输出流中的所有数据，然后插入一个换行符并刷新流（即清空缓冲区）。

2、cerr（console error）流

- **cerr流对象是标准错误流，作用是向标准错误设备（显示器）输出有关的出错信息。**
- **cerr流中的信息是用户根据需要制定的。**
- **cerr流中的信息只能在显示器输出，而不能被重定向输出到文件。**

例如：cerr << “a is equal to zero, error!” << endl;

3、clog（console log）流

clog流对象也是向显示器上显示输出信息，但是clog中的信息首先存放在缓冲区中，当缓冲区满或遇endl时向显示器输出；但cerr则不经过缓冲区而直接向显示器上输出有关信息。

13.2.2 标准类型数据的格式输出

C++在输入输出时：若不指定格式，则编译系统自动根据数据类型采取默认格式输入输出；若指定格式，则编译系统按用户指定格式输入输出。

□ 使用控制符控制输出格式

需要包含<iomanip>头文件。

控制符如左图所示。

输入输出流的控制符

控 制 符	作　　用
dec	设置整数的基数为 10
hex	设置整数的基数为 16
oct	设置整数的基数为 8
setbase(n)	设置整数的基数为 n(n 只能是 8,10,16 三者之一)
setfill(c)	设置填充字符 c,c 可以是字符常量或字符变量
setprecision(n)	设置实数的精度为 n 位。在以一般十进制小数形式输出时 n 代表有效数字。在以 fixed(固定小数位数)形式和 scientific(指数)形式输出时 n 为小数位数
setw(n)	设置字段宽度为 n 位
setiosflags(ios :: fixed)	设置浮点数以固定的小数位数显示
setiosflags(ios :: scientific)	设置浮点数以科学记数法(即指数形式)显示
setiosflags(ios :: left)	输出数据左对齐
setiosflags(ios :: right)	输出数据右对齐
setiosflags(ios :: skipws)	忽略前导的空格
setiosflags(ios :: uppercase)	在以科学记数法输出 E 和以十六进制输出字母 X 时以大写表示
setiosflags(ios :: showpos)	输出正数时给出“ + ”号
resetioflags( )	终止已设置的输出格式状态,在括号中应指定内容

□ 用流对象的成员函数控制输出格式

除了用控制符来控制输出格式，还可以通过调用流对象cout中用于控制输出格式的成员函数来控制输出格式。

用于控制输出格式的流成员函数

流成员函数	与之作用相同的控制符	作　　用
precision(n)	setprecision(n)	设置实数的精度为 n 位
width(n)	setw(n)	设置字段宽度为 n 位
fill(c)	setfill(c)	设置填充字符 c
setf( )	setiosflags( )	设置输出格式状态，括号中应给出格式状态，内容与控制符 setiosflags 括号中的内容相同，如表 7.5 所示
unsetf( )	resetioflags( )	终止已设置的输出格式状态，在括号中应指定内容

□ 格式标志

控制符setiosflags()和流成员函数setf()括号中的参数表示格式状态，它是通过格式标志来指定的。格式标志在类ios中被定义为枚举值，因此在引用这些格式标志时要在前面加上类名ios和域运算符“::”。

设置格式状态的格式标志

格式标志	作　　用
ios :: left	输出数据在本域宽范围内向左对齐
ios :: right	输出数据在本域宽范围内向右对齐
ios :: internal	数值的符号位在域宽内左对齐,数值右对齐,中间由填充字符填充
ios :: dec	设置整数的基数为 10
ios :: oct	设置整数的基数为 8
ios :: hex	设置整数的基数为 16
ios :: showbase	强制输出整数的基数(八进制数以 0 打头,十六进制数以 0x 打头)
ios :: showpoint	强制输出浮点数的小点和尾数 0
ios :: uppercase	在以科学记数法格式 E 和以十六进制输出字母时以大写表示
ios :: showpos	对正数显示“ + ”号
ios :: scientific	浮点数以科学记数法格式输出
ios :: fixed	浮点数以定点格式(小数形式)输出
ios :: unitbuf	每次输出之后刷新所有的流
ios :: stdio	每次输出之后清除 stdout,stderr

□ 说明

- 控制符setw(n)和成员函数width(n)只对其后的第1个输出项有效。
- 格式标志表分为5组，每组中只能同时选用一种格式标志。如果想修改此标志为同组中的另一种标志，则应使用控制符resetiosflags或成员函数unsetf()先终止原标志，然后再设置其他标志。
- 函数setf()设置的基数（dec、oct或hex）仅在本次输出流中生效，之后无论是否用函数unsetf()终止，下次输出流数据又以默认的十进制基数显示。但控制符resetiosflags()设置的基数只在resetiosflags()后才会被终止。
- 由于这些格式标志在ios类中被定义为枚举值，每个格式标志以一个二进制位代表，因此若需要同时设置多个格式标志，则可以用位或运算符“|”进行组合，例如：cout.setf(ios::internal | ios::showpos)。

13.2.3 用流成员函数put输出单个字符

口 格式

cout.put(字符常量或变量/整型表达式)[.put(…) …]

例如：cout.put('a'); 或 cout.put(97); 或 cout.put(71).put(79).put(79).put(68).put('\n');

口 说明

- 如是字符常量或变量，直接输出该字符；
- 如是整型表达式，（1）其值可以用八进制、十进制或十六进制表示；（2）如果表达式的值不在0~255的范围内，则对256取模，得到对应的ASCII码并输出字符（编译时可能会得到一条Warning）。

13.3 标准输入流

13.3.1 cin流

cin是istream类的对象，从标准输入设备读取数据。

流提取运算符>>在流中提取数据时通常跳过流中的空格、tab键、换行符等空白字符。只有输入回车键时输入的数据才进入键盘缓冲区，形成输入流，流提取运算符>>才能从其中提取数据。

当遇到无效字符（与变量数据类型不一致）或数据流（标准输入数据流或文件数据流）结束标志符EOF（EOF是在iostream头文件中定义的符号常量，值为-1）时，输入流cin就处于出错状态，此时对cin流的所有操作都被终止。当输入流出错时，cin的值是false，所以可以根据cin的值判断流对象是否处于正常状态。

13.3.2 用于字符输入的流成员函数

□ 用get函数读入一个字符

- 无参数的get函数

cin.get()

起作用是从指定的输入流中提取一个字符（包括空白字符），函数的返回值就是读入的字符。如果遇到输入流中的文件结束符，则返回EOF。

- 有一个参数的get函数

cin.get(ch)

其作用是从输入流中读取一个字符，赋给字符变量ch。如果读取成功，则返回非0值；如果失败（或遇EOF），则返回0。

- **有3个参数的get函数**

cin.get(字符数组或指针, 字符个数n, 终止字符)

其作用是从输入流中读取n-1个字符，赋给（字符指针指向的）字符数组。如果在读取n-1个字符之前遇到指定的终止字符，则提前结束读取；如果读取成功，则返回非0值；如果失败（或遇EOF），则返回0。

□ 用getline函数读入一行字符

cin.getline(字符数组或指针, 字符个数n, 终止字符)

例如：cin.getline(ch, 20, '/');，其作用是读入19个字符（或遇'/'结束），然后加一个'\0'共20个字符存放在字符数组ch中。

13.3.3 istream类的其他成员函数

□ eof函数

调用形式：

```
cin.eof()
```

eof是end of file的缩写，表示“文件结束”。从输入流读取数据，如果到达文件末尾（遇到EOF），则返回非0值，否则返回0。

□ peek函数

调用形式：

```
c = cin.peek();
```

peek函数的返回值是指针指向的当前字符，但指针仍停留在当前位置，并不后移。如果要访问的字符是EOF，则返回EOF。

- **putback函数**

调用形式：

```
cin.putback(ch);
```

其作用是将前面用get或getline函数从输入流中读取的字符ch插入到当前指针位置。

- **ignore函数**

调用形式：

```
cin.ignore(n, 终止字符)  或  cin.ignore()
```

其作用是跳过输入流中n个字符，若在跳跃过程中遇到指定的终止字符时则提前结束（此时跳过包括终止字符在内的若干字符）。如果ignore函数没有参数，则默认n值为1，终止字符为EOF。

13.4 对数据文件的操作与文件流

13.4.1 文件的概念

文件是指存储在外部存储介质上的数据集合，操作系统以文件为单位对数据进行管理。

对用户来说，常用到的文件有两大类：一类是程序文件，如源程序文件、目标文件和可执行文件等；另一类是数据文件，程序中输入和输出的对象就是数据文件。

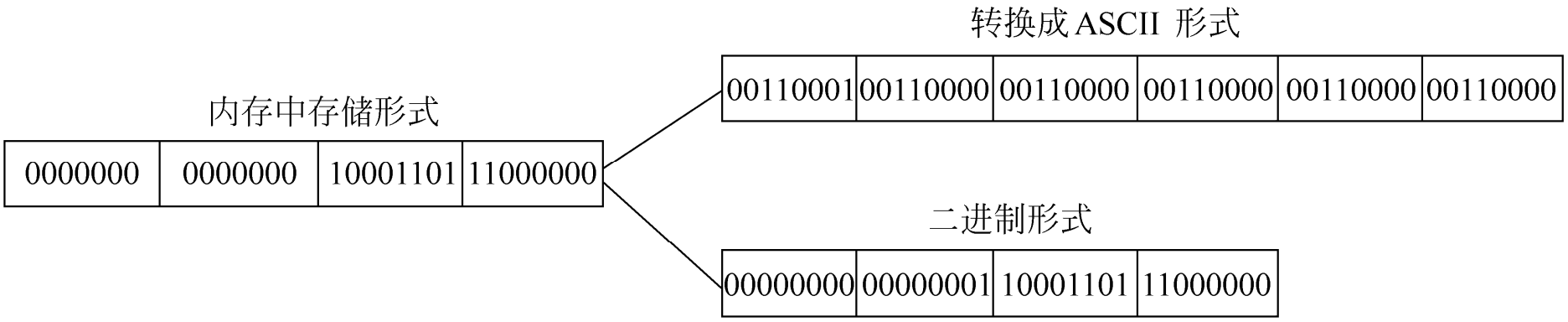
根据文件中数据的表示形式，文件分为ASCII文件和二进制文件。ASCII文件就是文本文件，每个字节表示一个字符。二进制文件又称为字节文件，是把内存中的数据、指令按其在内存中的存储形式原样输出存放在磁盘文件上。

字符信息在内存也是以ASCII码形式存放，所以字符在ASCII码文件和在二进制文件中形式是一样的。对于数值数据，两者是不一样的。例如，一个十进制长整数100000，用二进制表示时用四个字节；而用ASCII码表示时用六个字节。

□ C++提供了低级I/O功能和高级I/O功能。

低级I/O功能是以字节为单位，以二进制形式进行输入输出，期间不进行数据格式的转换。这种输入输入速度快、效率高，更适于大容量文件的I/O。

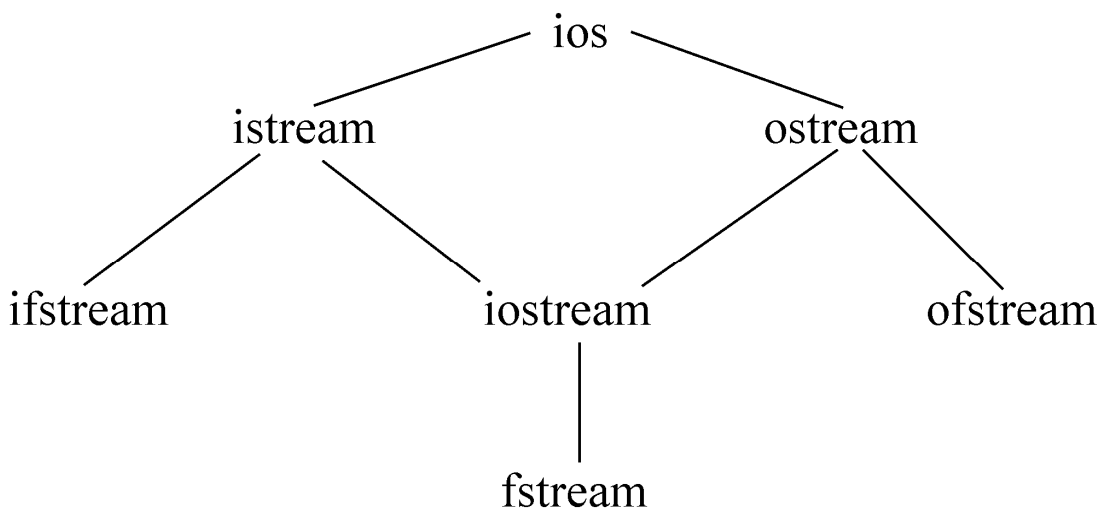
高级I/O功能是把若干个字节组合为一个有意义的单元（如标准类型数据或用户自定义类型数据等），然后以ASCII字符形式输入输出。例如将数据从内存输出到显示器：先将内存中的数据转换为ASCII字符，然后按照不同的数据类型的形式进行输出。



13.4.2 文件流类与文件流对象

文件流是以外存文件为输入输出对象的数据流。输出文件流是从内存流向外存文件的数据流，输入文件流是从外存文件流向内存的数据流。为了弥补访问内存和访问外存的速度差，每个文件流都有一个内存缓冲区。

在C++的I/O类库里定义了几种文件类，专门用于文件的输入和输出操作。



要对文件进行输入输出，必须定义一个文件流类的对象，用该对象去调用文件流类的成员函数对文件操作。例如定义输出文件流的对象：

```
ofstream  outfile;
```

13.4.3 文件的打开与关闭

13.4.3.1 打开磁盘文件

□ 打开文件是指在读写文件前做必要的准备工作，包括：

（1）在文件流对象和磁盘文件之间建立关联；

（2）指定文件的格式和操作方式。

□ 打开文件有两种方法：

1、建立文件流对象，用该对象调用类成员函数open。

- **调用成员函数的一般形式为**

文件流对象.open(文件名，输入输出方式);

例如：ofstream outfile; outfile.open(“f1.txt”, ios::out);

说明：（1）文件名可以包括路径，如省略路径，默认在当前目录；

（2）输入输出方式在ios类中定义，它们是枚举常量，有多种选择。

2、在定义文件流对象时指定参数

文件流类 对象(文件名，输入输出方式);

例如：ofstream outfile (“f1.txt”, ios::out);

文件输入输出方式设置值

方　　式	作　　用
ios :: in	以输入方式打开文件
ios :: out	以输出方式打开文件(这是默认方式),如果已有此名字的文件,则将其原有内容全部清除
ios :: app	以输出方式打开文件,写入的数据添加在文件末尾
ios :: ate	打开一个已有的文件,文件指针指向文件末尾
ios :: trunc	打开一个文件,如果文件已存在,则删除其中全部数据,如文件不存在,则建立新文件。如已指定了 ios :: out 方式,而未指定 ios :: app,ios :: ate,ios :: in,则同时默认此方式
ios :: binary	以二进制方式打开一个文件,如不指定此方式则默认为 ASCII 方式
ios :: nocreate	打开一个已有的文件,如文件不存在,则打开失败。nocreat 的意思是不建立新文件
ios :: noreplace	如果文件不存在则建立新文件,如果文件已存在则操作失败,noreplace 的意思是不更新原有文件
ios :: in | ios :: out	以输入和输出方式打开文件,文件可读可写
ios :: out | ios :: binary	以二进制方式打开一个输出文件
ios :: in | ios :: binar	以二进制方式打开一个输入文件

13.4.3.2 关闭磁盘文件

文件使用结束，必须关闭文件，用文件流对象调用关闭文件成员函数实现。

口 格式：

文件流对象.close();

功能：解除文件流对象与磁盘文件的关联。例如 outfile.close();

13.4.4 对ACSII文件的操作

ASCII文件也是文本文件，文件中一个字节存放一个字符。对ASCII码文件操作包括向文件写入字符和从文件读取字符。

读写ASCII文件有两种方法：

- 用文件流对象与流提取、插入运算符；
- 用文件流对象调用类的成员函数put, get, getline等。

例13.1 定义一个有十个元素的整型数组，从键盘输入十个整数，将它们放入数组，同时用插入运算符将它们写入当前目录下的f1.txt文件。

```
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
    int a[10];
    ofstream outfile("f1.txt"); //等价于ofstream outfile("f1.txt", ios::out);
    if(!outfile) {
        cerr<<"open error!"<<endl;
        exit(1);
    }
    cout<<"enter 10 integer numbers:"<<endl;
    for(int i=0; i<10; i++) {
        cin>>a[i];
        outfile<<a[i]<<" ";
    }
    outfile.close();
    return 0;
}
```

13.4.5 对二进制文件的操作

二进制文件是按内存中的数据存储形式写入磁盘文件，因此又称为内存数据的映象文件。

对二进制文件操作与对文本文件操作相似的是先定义文件流对象，然后打开文件，使用完要关闭文件。在打开时必须指定文件的存储形式是二进制形式，二进制文件即可以作为输入文件也可以作为输出文件，还可以作为既能输入又能输出的文件。这是与ASCII文件不同的地方。

13.4.5.1 用成员函数read和write读写文件

读二进制文件用istream类read成员函数；写二进制文件用ostream类write成员函数。

- 函数原型：

```
istream& read(char * buffer, int len);
ostream& write( const char * buffer, int len);
```

其中，字符指针buffer指向内存要读写的数据区域的起始位置。len是一次要读写的数据字节个数（数据长度）。

- 调用格式：

```
输入文件流对象.read(内存指针, 长度);
输出文件流对象.write(内存指针, 长度);
```

例13.2 把一批数据以二进制形式写入磁盘文件。

```cpp
#include <iostream>
#include <fstream>
using namespace std;
struct student {
    char name[20];
    int num;
    int age;
    char sex;
};
int main()
{
    student
    stud[3]= {  "Li",1001,18,'f',
                "Fun",1002,19,'m',
                "Wang",1004,17,'f'};
    ofstream outfile("stud.dat",ios::binary);
    if(!outfile) {
        cerr<<"open error!"<<endl;
        abort();
    }
    outfile.write( (char *)&stud,sizeof(stud));
    outfile.close();
    return 0;
}
```

例13.3 从例13.2产生的文件中读数据并显示到屏幕上。

分析：从文件读取数据必须先放入内存，所以必须设置一个与文件数据格式相同的数据结构既结构数组。然后再将结构数组元素逐个输出。

```
#include <iostream>
#include <fstream>
using namespace std;
struct student {
    char name[20];
    int num;
    int age;
    char sex;
};
```

```
int main()
{
    student stud[3];
    int i;
    ifstream infile("stud.dat",ios::binary);
    if(!infile) {
        cerr<<"open error!"<<endl;
        abort();
    }
    infile.read((char*)stud,sizeof(stud));
    infile.close();
    for(i=0; i<3; i++) {
        cout<<"NO."<<i+1<<endl;
        cout<<"姓名:"<<stud[i].name<<endl;
        cout<<"学号:"<<stud[i].num<<endl;
        cout<<"年龄:"<<stud[i].age<<endl;
        cout<<"性别:"<<stud[i].sex<<endl;
    }
    return 0;
}
```

13.4.5.2 与文件指针有关的流类成员函数

在磁盘文件中有一个文件读写位置标记（简称文件位置标记）来指明当前应进行读写的位置。从文件中每读入或每输出一个字节，文件位置标记向后移动一个字节。对于二进制文件，允许程序控制文件位置标记移动，实现随机访问文件。

文件流与文件指针有关的成员函数

成员函数	作　　用
gcount( )	返回最后一次输入所读入的字节数
tellg( )	返回输入文件指针的当前位置
seekg(文件中的位置)	将输入文件中指针移到指定的位置
seekg(位移量,参照位置)	以参照位置为基础移动若干字节(“参照位置”的用法见说明)
tellp( )	返回输出文件指针当前的位置
seekp(文件中的位置)	将输出文件中指针移到指定的位置
seekp(位移量,参照位置)	以参照位置为基础移动若干字节

说明：

- **这些函数名头或尾字母不是g就是p。带g的用于输入，带p的用于输出。对于输入输出文件不区分g和p。**
- **函数参数中的“文件中的位置”和“位移量”被指定为long型整数，以字节为单位。“参照位置” 表示位移的参照起点所在的位置，它们是在ios类中的定义的枚举常量：**

ios::beg 以文件开始为起点，这是默认值。

ios::cur 以文件位置标记的当前位置为起点。

ios::end 以文件结尾为起点。

例如： infile.seekg(100);

infile.seekg(-50, ios::cur);

outfile.seekp(-75, ios::end);

13.4.5.3 随机访问二进制数据文件

一般情况下读写是顺序进行的，即逐个字节进行读写。但对于二进制数据文件，可以利用流类的成员函数通过移动文件位置标记，实现随机访问文件中任何一个字节里的数据。

例：有五个学生的数据，要求：

① 把它们写入磁盘文件；

② 从磁盘文件读第1，3，5学生数据并显示；

③ 修改第3 个学生的数据并保存到原来位置；

④ 从磁盘文件读入修改过的5个学生数据并显示。

程序见13.4.cpp

13.5 字符串流

字符串流把用户定义的字符数组（字符串）作为数据的输出目标或者数据的输入源，即将字符串流中的数据输出（读取）到字符数组，或向字符串流输入（写入）字符数组中的数据。

字符串流也需要缓冲区，读取或写入时，流缓冲区中的数据不断增加，待缓冲区满或遇到换行符时，缓冲区中数据一起写入字符数组或赋予指定变量。

1、把字符串流中的数据输出（读取）到字符数组

- ostrstream类的构造函数原型是

```
ostrstream::ostrstream(char *buffer, int n, int mode =ios::out);
```

其中，buffer是指向字符数组首址的指针；n是指定流缓冲区的长度；第三个参数可省略，默认是ios::out。

例如：ostrstream strout(ch1, 20);

作用是建立字符流对象strout，并与字符数组ch1关联（把字符串流中的数据写入字符数组ch1），流缓冲区长度是20个字节。

2、向字符串流输入（写入）字符数组中的数据

istrstream::istrstream(char *buffer);

istrstream::istrstream(char *buffer, int n);

其中，buffer是指向字符数组首址的指针，n 是流缓冲区的长度。如果没有n，表示缓冲区的长度与字符串数组长度相同。

例如：istrstream strin(ch2);

istrstream strin(ch2, 20);

第一条语句是建立字符串流对象strin，将字符数组ch2所有数据写入字符串流；

第二条语句是建立字符串流对象strin，将字符数组ch2前20个字符写入字符串流。

3、用字符串流进行数据的输入输出

□ strstream类提供的构造函数的原型为

```
strstream::strstream(char *buffer, int n, int mode);
```

例如：strstream strio(ch3, sizeof(ch3), ios::in|ios::out);

例13.5 在一个字符数组c中存放10个整数，以空格为分隔符，要求将它们放到整型数组中排升序，然后再写入原来的字符数组中。

见程序13.5.cpp

□ 字符串流的两种头文件sstream和strstream

字符串流头文件	包含的类	说明
strstream	class strstreambuf	它们是基于C字符串char*编写的
	class istrstream	
	class ostrstream	
	class strstream	
sstream	class stringbuf	它们是基于C++字符串std::string编写的
	class istringstream	
	class ostringstream	
	class stringstream	
	其他流类	

两个头文件中实现的东西基本一样，一般情况下C++标准委员会推荐使用sstream头文件。但若为了兼容C字符串，则可以使用strstream。

◆ 但要记住一点，strstream虽仍然是C++语言标准的一部分，但已被C++标准宣称为“deprecated”，也就是不再提倡使用了。

CH13 End