第7章

文件和流

本章主要内容

- 7.1 C++的流
- 7.2 文件流
- 7.3 文件的打开和关闭
- 7.4 文件的读写操作
- 7.5 综合实例
- 7.6 字符串流

7.1 C++的流

- 输入和输出是数据传送的过程，数据像流水一样从一处流向另一处，C++中形象地将此过程称之为流（stream）
- 从流中提取数据称为输入操作，在输入操作中，字节流从输入设备(例如键盘、磁盘、网络连接等)流向内存；
- 向流中添加数据称为输出操作，在输出操作中，字节流从内存流向输出设备(例如显示器、打印机、网络连接等)。
- C++带有一个I/O流类库，包含许多用于输入输出的类，称为流类。用流类定义的对象称为流对象。
- C++中包含几个预定义的流(流对象），它们是：标准输入流（流对象）cin、标准输出流(流对象)cout、非缓冲型的标准出错流（流对象）cerr、缓冲型的标准出错流（流对象）clog

7.2 文件流

1、文件的概念与分类

文件是指存储在存储介质上的数据的集合，每个文件有一个唯一的文件名称（含所在路径）。

- 按文件中数据的存放形式可将文件分为：
  - **ASCII文件（文本文件）**
    - 它的每一个字节存放一个**ASCII**代码，代表一个字符；
    - 其优点是可直接按字符形式输出文件的内容，也可用一般的字处理软件直接打开并查看文件的内容；
  - **二进制文件**
    - 将数据用二进制形式存放在文件中，并保持了数据在内存中存放的原有格式；
    - 其优点是存储效率高，无须进行存储形式的转换，但不能直接按字符形式输出。

'1'	'2'	'9'	'7'	<EOF>

以ASCII码表示的1297

49	50	57	55	<EOF>

以二进制表示的短整型数字1297

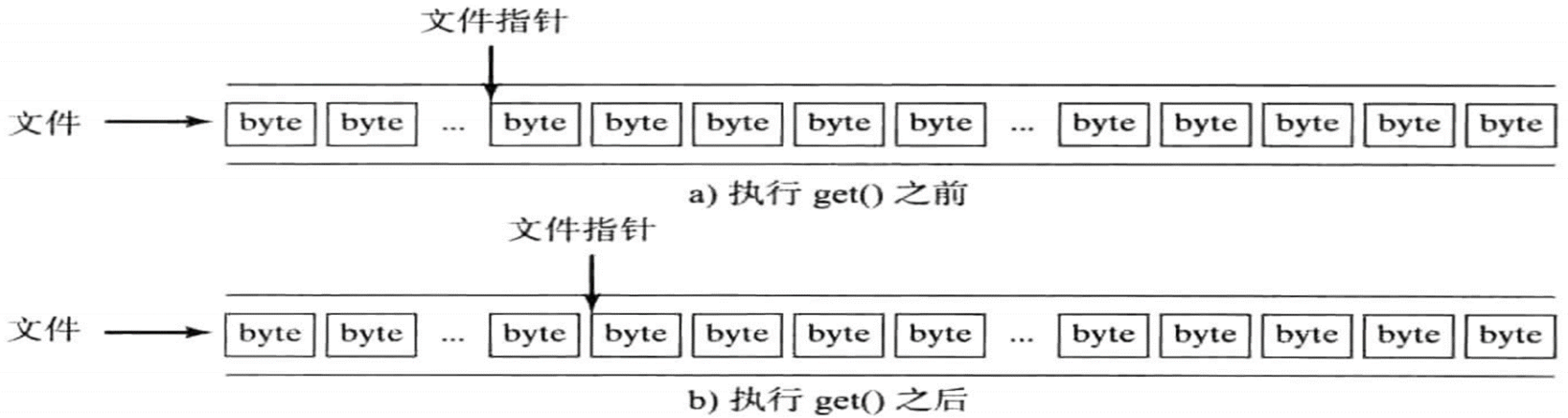
00000101	00010001

以十六进制表示的短整型数字1297

05	11

2、文件指针的概念

- 无论是文本文件还是二进制文件都需要用“文件指针”来操纵，“文件指针”是指表示读写文件的位置指示器。
- 当文件每一次打开时，文件指针默认指向文件的开始；
- 随着对文件进行操作，文件指针位置不断地在文件中移动，并一直指向最新处理的字符（字节）位置。



3、对文件的读写操作有两种方式

- 顺序文件操作
  - 从文件的第一个字符（字节）开始顺序地处理到文件的最后一个字符（字节）。
  - 只能从文件的开始处依次顺序读写文件内容，而不能任意读写文件内容。
- 随机文件操作
  - 在文件中通过相关的函数移动文件指针，并指向所要处理的字符（字节）。
  - 可以在文件中来回移动文件指针和非顺序地读写文件内容。
  - 能快速地检索、修改和删除文件中的信息。

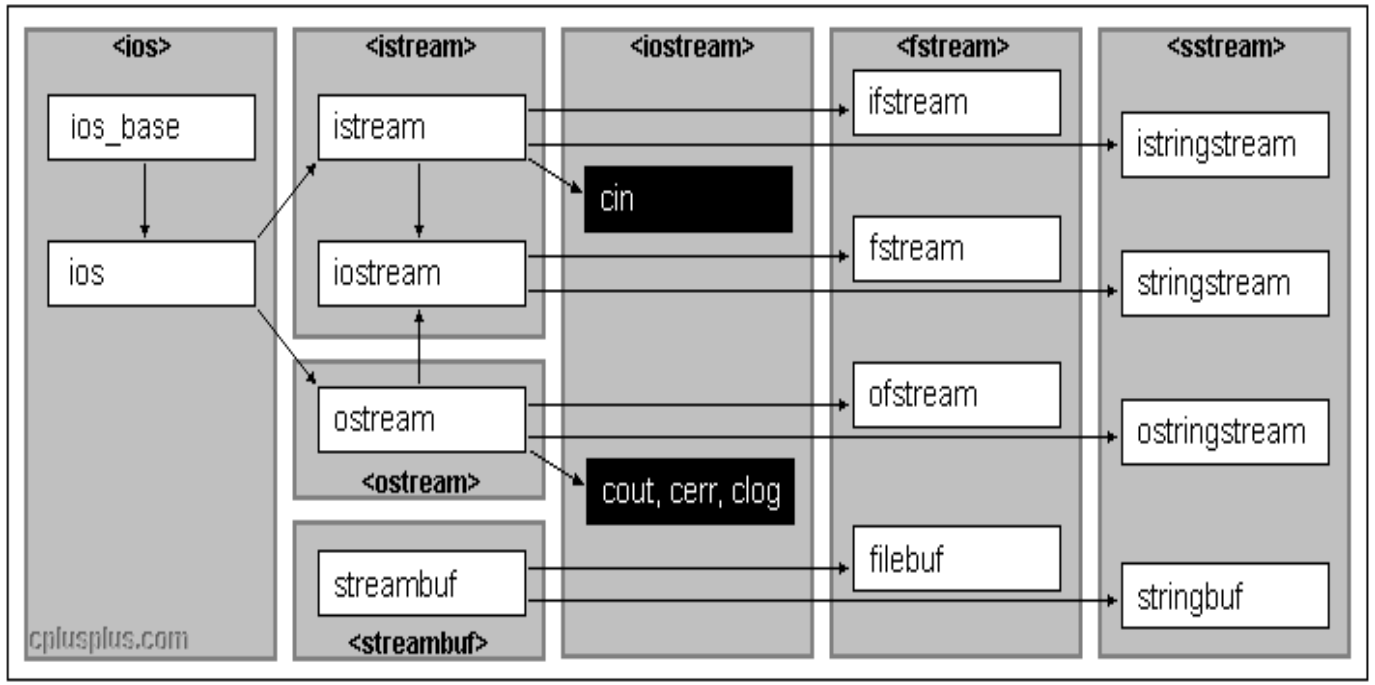
4、对文件读写的步骤

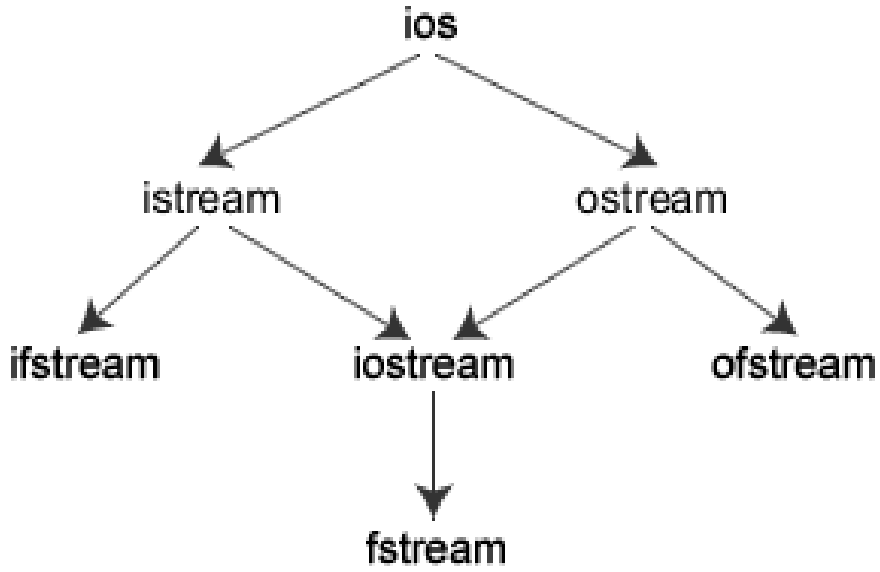
- 在C++中，可以将文件定义为文件流类的一个对象，对文件的输入/输出（即读/写），包括以下三步：
  - **先创建一个文件流对象，并与指定的文件关联，即打开文件；**
  - **然后才能进行读写操作；**
  - **完成后再关闭这个文件。**

5、文件流与对应的类

- C++对文件内容的操作（读写操作）是通过文件流来完成的。
- 文件流（Text Stream）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（写文件）。
- 输入文件流是从外存文件流向内存的数据（读文件）。
- C++有三种文件流类：
  - （1）ifstream流类，是从istream类派生的，用于文件的输入操作（读文件）；
  - （2）ofstream 流类，是从ostream类派生的，用于文件的输出操作（写文件）；
  - （3）fstream 流类，是从iostream类派生的，用于文件的输入和输出操作。
    - 在对文件进行输入输出操作时，首先应该在开始包含#include<fstream>

5、文件流与对应的类


<ios>
<istream>
<iostream>
<fstream>
<sstream>
ios_base
ios
istream
iostream
ostream
<ostream>
streambuf
<streambuf>
cin
cout, cerr, clog
ifstream
fstream
ofstream
filebuf
istringstream
stringstream
ostringstream
stringbuf
cplusplus.com


ios
istream
ostream
ifstream
iostream
ofstream
fstream

7.3 文件的打开和关闭

- 1、打开文件的方法
  - （1）使用文件流类的open()方法
  - （2）使用文件流类的构造函数

（1）使用文件流类的open方法

- open()方法原型为：

  void open(const unsigned char *filename, int mode, [int access=filebuf::openprot]);

  ①filename是一个字符型指针，它指定了要打开的文件名（含文件路径）；

  ②mode指定了文件的打开方式，其取值和对应功能如下表所示；

  ③access指定了文件的系统属性，其取值为：0一般文件；1只读文件；2隐藏文件；3系统文件

文件打开方式	含　　义
ios::in	以输入（读）方式打开文件
ios::out	以输出（写）方式打开文件
ios::app	打开一个文件使新的内容始终添加在文件的末尾
ios::ate	打开一个文件使新的内容添加在文件尾，但下一次添加时，写在当前位置处
ios::trunc	若文件存在，则清除文件所有内容；若文件不存在，则创建新文件
ios::binary	以二进制方式打开文件，缺省时以文本方式打开文件
ios::nocreate	打开一个已有文件，若该文件不存在，则打开失败
ios::noreplace	若打开的文件已经存在，则打开失败

说明：

①ios::in方式：只能从文件输入数据（即读文件），而且文件必须已经存在。

如果用类ifstream来创建一个文件流对象，则隐含为输入流，不必再指定打开方式。例如：

ifstream fin;　　fin.open("abc.txt");

②ios::out方式：只能向文件输出数据（即写文件）。

如果用类ofstream来创建一个文件流对象，则隐含为输出流，默认为“ios::out|ios::trunc”，不必显式地声明打开方式。如果文件不存在，则创建一个新文件；如果文件存在，则打开文件并清空文件，输出将进入一个空文件中。例如：

ofstream fout;　　fout.open("abc.txt");

③如果使用fstream类创建对象时，必须显式地提供模式，原因是fstream类不提供默认的模式值。

④ios::app方式：不删除文件原来数据，向文件末尾添加新数据。使用“ios::app”方式，文件必须存在，而且只能用于输出。

⑤ios::ate方式：打开一个已存在的文件，文件指针自动位于原有文件的尾部。

⑥在实际使用过程中，可根据需要将以上打开文件的方式用“|”组合起来。如：

ios::in|ios::out　　表示以读/写方式打开文件

ios::in|ios::binary　表示以二进制读方式打开文件

ios::out|ios::binary　表示以二进制写方式打开文件

ios::in|ios::out|ios::binary 表示以二进制读/写方式

⑦如果未指明以二进制方式打开文件，则默认是以文本方式打开文件。

⑧为了避免程序异常，在打开文件的代码之后最好要设置打开是否成功的代码。如果打开失败，流对象的值为0。

（2）使用构造函数

- **格式：**
  - **ifstream 对象名("文件名","打开方式");**
  - **ofstream 对象名("文件名","打开方式");**
  - **fstream 对象名("文件名","打开方式");**
- **说明：**
  - 使用ifstream和ofstream类的构造函数打开文件，可以省略第二个参数“打开模式”。在默认情况下，ifstream的打开模式为“ios::in”，ofstream的打开模式为“ios::out|ios::trunc”。

2 关闭文件

- 文件操作结束时调用文件流对象的函数close()来关闭文件。

  如：要关闭的文件流对象**myfile**，则可使用如下语句关闭文件：

  ```
  myfile.close();
  ```

7.4 文件的读写操作

- 1. 顺序文件操作
- 2. 随机文件操作

1．顺序文件操作

- 从一个文件中读出数据，可以使用
  - **iostream类的get()、getline()、read()成员函数以及提取运算符“>>”；**
- 向一个文件写入数据，可以使用
  - **iostream类的put()、write()函数以及插入运算符“<<”。**

函数原型	说　　明
get(char &ch)	从文件中读取一个字符
getline(char *pch, int count, char delim='\n')	从文件中读取多个字符，读取个数由参数 count 决定，参数 delim 是读取字符时指定的结束符
read(char *pch, int count)	从文件中读取多个字符，读取个数由参数 count 决定
put(char ch)	向文件写入一个字符
write (const char *pch, int count)	向文件写入多个字符，字符个数由 count 决定

一般用于二进制文件的读写操作

文件读写<<和>>

用插入运算符(<<)向文件**输出（向文件中写内容）**；

用提取运算符(>>)从文件**输入（从文件读取内容）**。

假设file1是以输入方式打开，file2以输出打开。示例如下：

```
file2<<”I Love You”;          //向文件写入字符串
int i;
file1>>i;                     //从文件输入一个整数值。
```

格式化类 iomanip

这种方式还有一种简单的格式化能力，具体的格式有:

操纵符	功能	
dec	格式化为十进制数值数据	输入和输出
endl	输出一个换行符并刷新此流	输出
ends	输出一个空字符（end string，终止字符串’\0’）	输出
hex	格式化为十六进制数值数据	输入和输出
oct	格式化为八进制数值数据	输入和输出
setprecision(int p)	设置浮点数的精度位数	输出

比如要把123当作十六进制输出：file1<<hex<<123;

要把3.1415926以5位精度输出：file1<<setprecision(5)<<3.1415926。

- 【例7-1】向文本文件中分别写入一个整数和一个字符串，然后再以读方式打开该文件，从中读取相应的信息并将其显示到屏幕上。
- 分析：
  - 以写方式打开文件，判断打开是否成功；若打开成功，用<<写入一个整数和一个字符串；关闭文件
  - 以读方式打开文件，判断打开是否成功；若打开成功，用getline函数读出，注意用eof函数判断文件是否结束；如果结束，则关闭文件

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
        fstream myfile;
        myfile.open("test.txt",ios::out);
        if(!myfile)
        {
                cout<<"Cannot open the test.txt file.\n";
                exit(0);
        }
        myfile<<100<<"\n"<<"file read and write
testing\n";
        myfile.close();
        myfile.open("test.txt",ios::in);
        if(!myfile)
        {
                cout<<"Cannot open the test.txt file.\n";
                exit(0);
        }
        char temp[50];
        while(!myfile.eof())//文件是否结束
        {

                myfile.getline(temp,sizeof(temp));
                cout<<temp<<"\n";
         }
         myfile.close();
         return 0;
}
```

练习

- （1）写文件：打开一个文本文件file.txt，在其中写入一行“hello C++!”，并等待将键盘输入的数据写入文件，遇到换行符结束
- （2）读文件：将file.txt的内容读出并显示

练习

- （3）写一个对象数组：打开一个二进制文件file.dat，写入一个Sprite类对象数组（对象数组中含有3个对象）：

```
Sprite sprites[3]={
        Sprite("法师","魔杖"),
        Sprite("战士","屠龙宝刀"),
        Sprite("道士","倚天剑")};
```

怎么写？ put write <<

- （4）读出file.dat文件，将内容显示到屏幕。

怎么读？ get getline read >>

```
class Sprite
{
private:
        string profession;//职业
        string weapon;//武器
        static int count;//个数
 public:
        Sprite() {}
        Sprite(string profession,string weapon)
        {
                this->profession = profession;
                this->weapon = weapon;
        }
        void showSprite();//显示信息
 };
int Sprite::count=0;
void Sprite::showSprite()
{
    ++count;
    cout<<"精灵"<<count<<" 职业:"<<profession<<" 武器:"<<weapon<<endl;
}
```

2. 随机文件操作

- （1）istream类中提供了3个操作文件读指针的成员函数：

  istream&istream::seekg (long pos);

  istream&istream::seekg(long off,dir);

  streampos istream::tellg();

- 其中
  - pos为文件指针的绝对位置（字节）；
  - off为文件指针的相对偏移量（字节）；
  - dir为文件指针的参照位置，其值可能为：
    - **ios::cur** 文件指针的当前位置
    - **ios::beg** 文件开头
    - **ios::end** 文件尾

tellg()函数没有参数，它返回一个long型值，用来表示从文件开始处到当前指针位置之间的字节数。

- （2）ostream类中也提供了3个操作文件写指针的成员函数：

  ostream&ostream::seekp (long pos);

  ostream&ostream::seekp(long off,dir);

  streampos ostream::tellp();

- 【例7-2】随机文件的读写操作。定义学生结构体，将3个学生的信息写入一个文件，然后读取文件，显示器输出。输出时按照3132的顺序输出学生信息。

```
#include <iostream>
#include <fstream>
using namespace std;
struct Student
{
        char name[20];
        char num[10];
        double score;
};
int main()
{
        Student stu[3] = {{"Mary","001",98},{"Susan","002",97},{"Peter","003",99}};
        fstream myfile;
        myfile.open("test.txt",ios::out|ios::in | ios::binary|ios::trunc);  // ios::in | ios::out 需要先创建文件
        if(!myfile){
            cout<<"cannot open test.txt\n";
            exit(0);
        }
```

```
        for(int i=0;i<3;i++)
            myfile.write((char *)&stu[i],sizeof(Student));
        myfile.seekp(sizeof(Student)*2);                    //指向第3个同学,在fstream中,seekp和seekg没有区别
        Student temp;
        myfile.read((char *)&temp,sizeof(Student));
        cout<<temp.name<<"\t"<<temp.num<<"\t"<<temp.score<<"\n";
        myfile.seekp(0);                                    //指向第1位同学
        myfile.read((char *)&temp,sizeof(Student));
        cout<<temp.name<<"\t"<<temp.num<<"\t"<<temp.score<<"\n";
        myfile.seekp(sizeof(Student)*1, ios::cur);          //指向第3个同学
        myfile.read((char *)&temp,sizeof(Student));
        cout<<temp.name<<"\t"<<temp.num<<"\t"<<temp.score<<"\n";
        myfile.seekp(sizeof(Student)*1, ios::beg);          //指向第2个同学
        myfile.read((char *)&temp,sizeof(Student));
        cout<<temp.name<<"\t"<<temp.num<<"\t"<<temp.score<<"\n";
        myfile.close();
        return 0;
}
```

练习

- 读出file.dat文件，将内容显示到屏幕。然后更改读取顺序，并显示到屏幕上，顺序如下：道士、战士、法师

7.5 综合实例

- 【例7-3】利用文件流对文件进行操作。

  （1）输入若干个学生的数据（包括学号、姓名、成绩）。

  （2）将数据存放在**Data.dat**文件中。

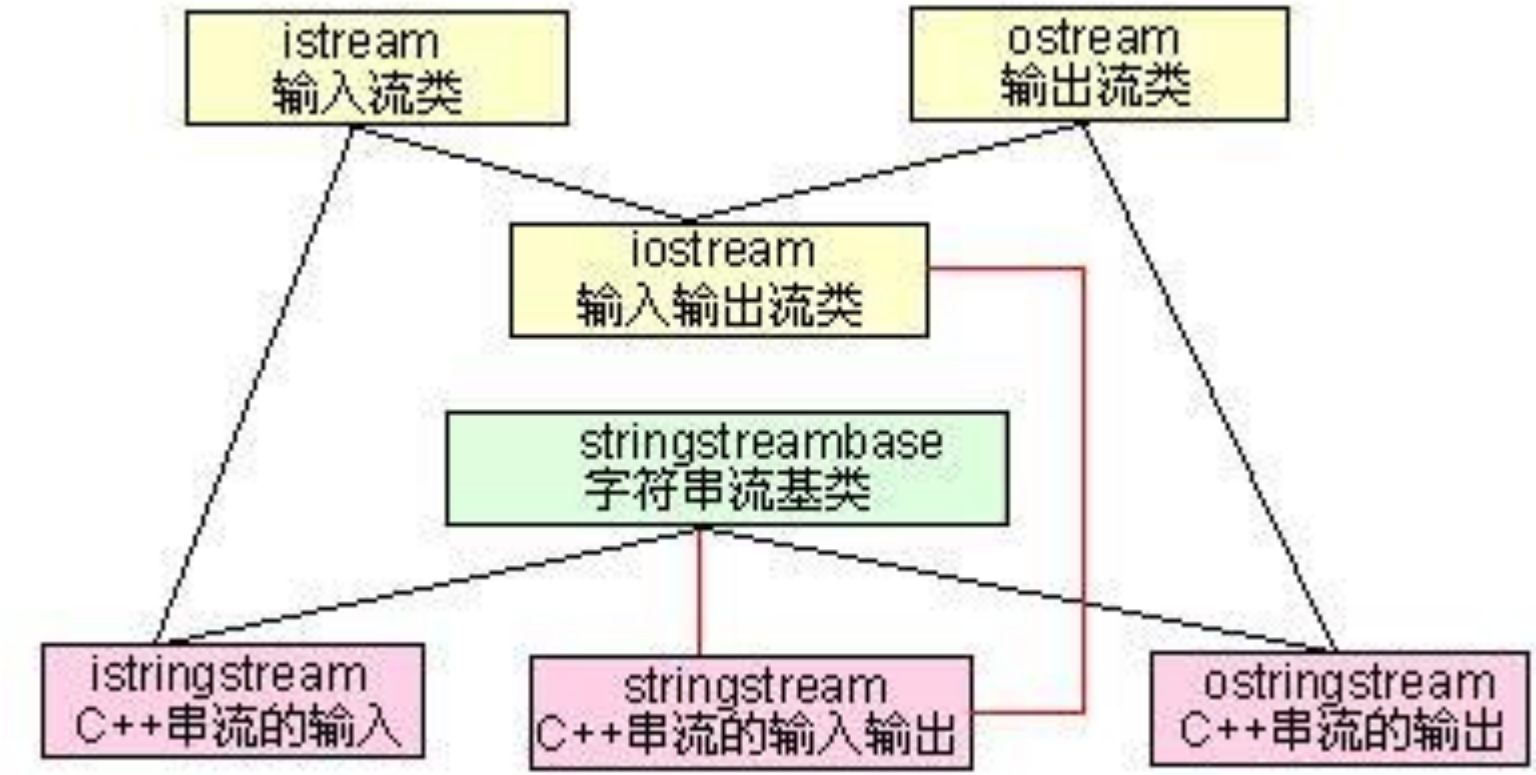
  （3）从**Data.dat**文件中读出所有数据并显示出来。

```
#include <iostream>
#include <fstream>
using namespace std;
struct Student{
        char name[20];
        char num[10];
        double score;
};
int main()
{
        Student temp[100];
        int i, n;
        cout<<"输入学生的人数： ";
        cin>>n;
        for(i=0;i<n;i++)
        {
            fflush(stdin);  //清空缓冲区
            cout<<"输入第"<<i+1<<"个学生的姓名、学号和成绩： ";
            cin>>temp[i].name;
            cin>>temp[i].num>>temp[i].score;
        }
```

```
    fstream fin, fout;
    fout.open("data.dat",ios::out);
    if(!fout){
        cout<<"cannot open data.dat.\n";
        exit(0);
    }
    for(i=0;i<n;i++)
        fout.write((char *)&temp[i],sizeof(Student));
    fout.close();
    fin.open("data.dat",ios::in);
    if(!fin){
        cout<<"cannot open data.dat.\n";
        exit(0);
    }
    cout<<"姓名\t学号\t成绩\n";
    for(i=0;i<n;i++){
        fout.read((char *)&temp[i],sizeof(Student));
        cout<<temp[i].name<<"\t"<<temp[i].num<<"\t"<<temp[i].score<<"\n";
    }
    fin.close();
    return 0;
}
```

7.6 字符串流

- iostream 标准库支持内存中的输入／输出，只要将流与存储在程序内存中的string 对象捆绑起来，就可使用 iostream 输入和输出操作符读写这个 string 对象。标准库定义了三种类型的字符串流：
  - **istringstream**，由 **istream** 派生而来，提供读 **string** 的功能。
  - **ostringstream**，由 **ostream** 派生而来，提供写 **string** 的功能。
  - **stringstream**，由 **iostream** 派生而来，提供读写 **string** 的功能。
- 要使用上述类，必须包含 sstream 头文件：#include<sstream>


istream
输入流类
ostream
输出流类
iostream
输入输出流类
stringstreambase
字符串流基类
istringstream
C++串流的输入
stringstream
C++串流的输入输出
ostringstream
C++串流的输出

- stringstream strm; //创建一个stringstream对象strm
- stringstream strm(s); //创建存储 s 的副本的 stringstream 对象，其中 s
- //是string 类型的对象
- strm.str(); //返回 strm 中存储的 string 类型对象
- strm.str(s); //将 string 类型的 s 复制给 strm，返回 void
- strm.str(“”); //清空stringstream的内存
- strm.clear(); //重置流的标志状态，不释放内存
- strm<<“1234”; //将字符串“1234”写到strm中
- strm>>first; //将strm读到first中，如果first不是string类型的对
- //象，则进行数据类型转换

【例7-4】数据类型转换

```
#include <sstream>
#include <iostream>
using namespace std;
int main()
{
    stringstream sstream;
    int first, second;
    sstream << "456";// 插入字符串
    sstream >> first;// 转换为int类型
    cout << first << endl;
    sstream.clear();// 在进行多次类型转换前，必须先运行clear()
    sstream << true;// 插入bool值
    sstream >> second;// 转换为int类型
    cout << second << endl;
    return 0;
}
```

【例7-5】字符串拼接

```
#include <string>
#include <sstream>
#include <iostream>
using namespace std;
int main()
{
        stringstream sstream;
        sstream << "first" << " " << "string,";// 将多个字符串放入 sstream 中
        sstream << " second string";
        cout <<  sstream.str() << endl;

        sstream.str("");// 清空 sstream
        sstream << "third string";
        cout <<sstream.str() << endl;
        return 0;
}
```

【例7-6】stringstream对象的使用

```
#include<sstream>
#include<iostream>
using namespace std;
int main()
{
    string line, word;
    while (getline(cin, line))//从键盘上输入多行带有空格的字符串，以换行符结束
   {
        stringstream stream(line);//将每行字符串复制到stream中
        cout << stream.str() << endl;//输出每行字符串
        while (stream >> word) //从stringstream流中的数据输入字符串到变量word里，字符串流通过空格
                                //判断一个字符串的结束
       {            cout << word << endl;                }
    }
    return 0;
}
```

【例7-7】统计输入的一行字符串（含有空格）的单词数量

```
#include<sstream>
#include<string>
#include<iostream>
using namespace std;
int main()
{
        string str,word;
        getline(cin, str);
        stringstream ss(str);
        int count = 0;
        while (ss >> word)
        {               count++;        }
        cout << count << endl;
        return 0;
}
```

【例7-8】统计输入的一行字符串（含有空格和数字）的单词数量，不统计纯数字的单字。

```
#include<sstream>
#include<string>
#include<iostream>
using namespace std;
int main()
{
        string str, word;
        int num;
        getline(cin, str);
        stringstream ss(str);
        int count = 0;

while (ss >> word)
        {
                //判断是否是数字
                stringstream temp(word);
                if (temp >> num)
                        continue;

                count++;
        }
        cout << count << endl;
        return 0;
}
```

【例7-9】统计输入的字符串中“hello”出现的次数（不区分大小写）

```
#include<sstream>
#include<string>
#include<iostream>
#include<algorithm>
using namespace std;
int main()
{
    string str,word;
    getline(cin, str);
    stringstream ss(str);
    int count = 0;
    while (ss >> word)
    {
        //去除标点
        word.erase(remove(word.begin(), word.end(), '.'), word.end());
        // 转小写
        transform(word.begin(), word.end(), word.begin(), ::tolower);
        if(word=="hello")
            count++;
     }
     cout << count << endl;
     return 0;
}
```

transform函数中,第一个参数是容器的首迭代器，第二个参数为容器的末迭代器，第三个参数为存放结果的容器，第四个参数为要进行操作的一元函数对象或**sturct**、**class**

【例7-10】e盘有一个in.txt文件，其中第一行有一个数字 N 代表接下來有 N 行数字，每一行数字里有不固定个数的整数，将每一行的总和写到e盘的out.txt文件。

```
#include<iostream>
#include<string>
#include<sstream>
#include<fstream>
using namespace std;
int main()
{
        fstream fp("e:\\in.txt",ios::in);
        fstream fp1("e:\\out.txt", ios::out);
        if (!fp||!fp1)
        {
                cout << "cannot open files\n";
                exit(0);
        }

        string s;
        stringstream ss;
        int n, i, sum, a;
        fp >> n;
        getline(fp, s); // 换行读取
        for (i = 0; i < n; i++)
        {
                getline(fp, s);
                ss.clear();
                ss.str(s);
                sum = 0;
                while (ss>>a)
                        sum += a;
                fp1 << sum << endl;
        }
        fp.close(); fp1.close();
}
```