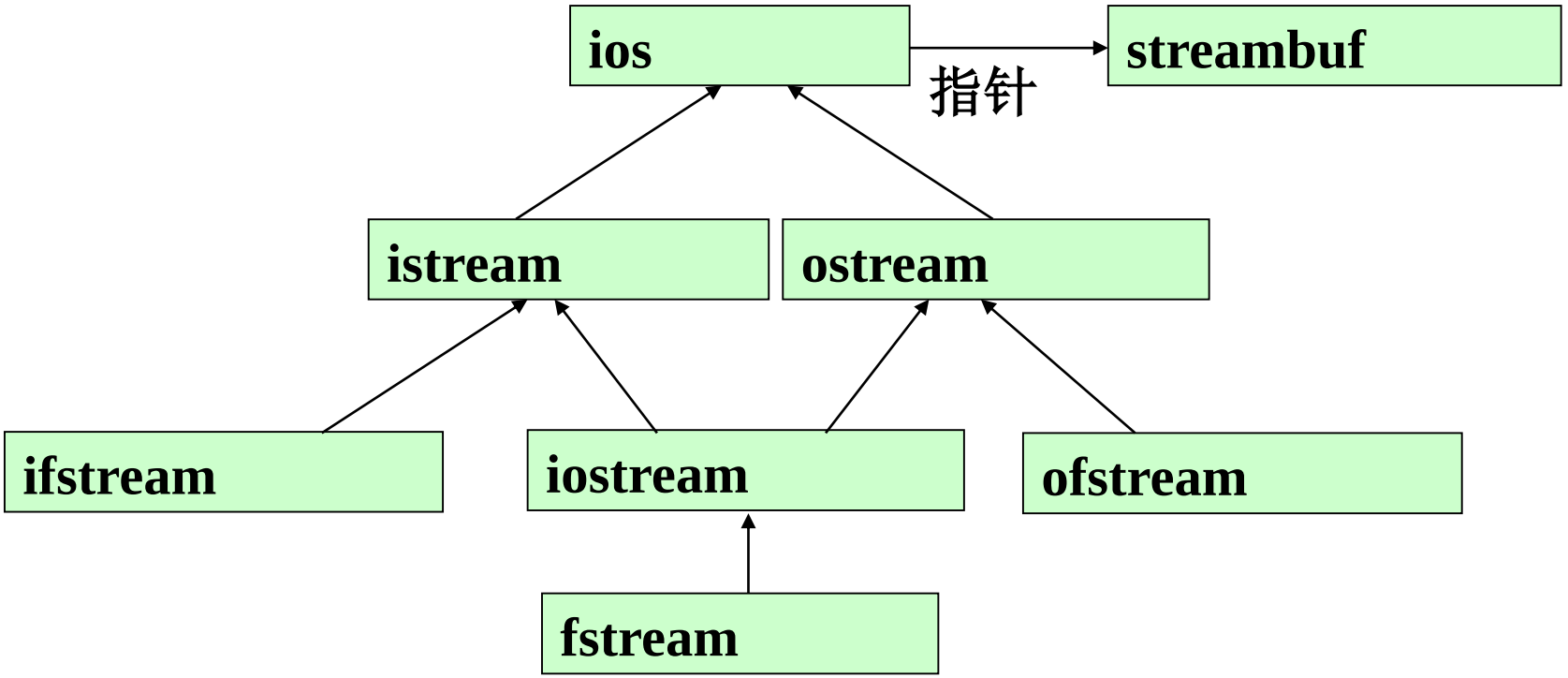
第1001章 文件操作

文件输入/出流

- 数据从一个对象到另一个对象的传送被抽象为“流”。数据的输入/输出就是通过输入/输出流来实现的。
- 流是一种抽象的概念，负责在数据的产生者和数据的使用者之间建立联系，并管理数据的流动。

C++的基本流类体系

- **流类体系：** 以抽象类模板ios为基类，流类模板派生体系见图。整个流类模板体系的标准I/O在头文件<iostream>中说明，它包含头文件<ios>、<streambuf>、<istream>和<ostream>。而输入输出文件流部分在头文件<fstream>中说明。



文件的输入输出

- 当打开一个文件时，该文件就和某个流关联起来了。对文件进行读写实际上受到一个文件定位指针（file position pointer）的控制。

- 输入流的指针也称为读指针，每一次提取操作将从读指针当前所指位置开始，每次提取操作自动将读指针向文件尾移动。

- 输出流指针也称写指针，每一次插入操作将从写指针当前位置开始，每次插入操作自动将写指针向文件尾移动。

文件的打开与关闭

1. 说明一个文件流对象，这又被称为内部文件：
   - **ifstream ifile；//只输入用**
   - **ofstream ofile；//只输出用**
   - **fstream iofile；//既输入又输出用**
2. 使用文件流对象的成员函数打开一个磁盘文件。文件流中说明了三个打开文件的成员函数。
   - **void ifstream::open(const char*,int =ios::in,**
     int=filebuf::openprot);
   - **void ofstream::open(const char *,int=ios::out,**
     int=filebuf::opernprot);
   - **void fstream::open(const char*,int, int=filebuf::openprot);**
   - 第一个参数为要打开的磁盘文件名。第二个参数为打开方式，有输入（**in**），输出（**out**）等，打开方式在**ios**基类中定义为枚举类型。第三个参数为指定打开文件的保护方式，一般取默认。所以第二步可如下进行：
   - **iofile.open("myfile.txt",ios::in|ios::out);**

文件的打开与关闭

- **打开方式解释：**
  - ios::in　//打开文件进行读操作
  - ios::out　//打开文件进行写操作
  - ios::ate　//打开时文件指针定位到文件尾
  - ios::app　//添加模式，所有增加都在文件尾部进行
  - ios::trunc　//如果文件已经存在则清空源文件
  - ios::nocreate　//如果文件不存在则打开失败
  - ios::noreplace　//如果文件存在则打开失败
  - ios::binary　//二进制文件（非文本文件）
- Mode的符号常量可以用位或运算|组合在一起，如：ios::in|ios::binary　只读方式打开二进制文件
- 对于ifstream流，mode默认方式是ios::in；对于ofstream流，mode默认方式是ios;;out；

文件的打开与关闭

- 有效的组合方式
  - out　　打开文件做写操作，删除文件中已有数据
  - out|app　　打开文件做写操作，在文件尾写入
  - out|trunc　　与out模式相同
  - in　　打开文件做读操作
  - in|out　　打开文件做读写操作，并定位于文件开头处
  - in|out|trunc 打开文件做读写操作，删除文件中已有的数据

文件的打开与关闭

3. 三个文件流类都重载了一个带默认参数的构造函数，功能与open函数一样：

- ifstream::ifstream(const char*, int=ios::in,
  int=filebuf::openprot);

- ofstream::ofstream(const char*,int=ios::out,
  int=filebuf::openprot);

- fstream::fstream(const char*,int,
  int=filebuf::operprot);

所以1，2两步可合成：

- fstream iofile(”myfile.txt”,ios::in|ios::out);
- 建议使用这种方式打开文件

文件的打开与关闭

- 判断文件打开是否成功:**必须有这步**

```
fstream iofile(”myfile.txt”, ios::in|ios::out);
if(!iofile){
      cout<<”不能打开文件:”<<”myfile.txt”<<endl;
      return -1;
 } //失败退回
```

```
ifstream  ifile(”myfile.txt”, ios::in);
if(!ifile){
      cout<<”不能打开文件:”<<”myfile.txt”<<endl;
      return -1;
 } //失败退回
```

- 文件关闭
  - iofile.close(); //可以省略，因为析构函数里会自动关闭文件

文件的读写

- 使用提取和插入运算符对文件进行读写操作，或使用成员函数进行读写。
  - 输出流函数：<<; write
  - 输入流函数：>>; read

<<：将1个数据写入到文本文件中，基本方法和cout相同，而且也有相同的格式控制功能，特别指出：仅用于写文本文件

>>：从文本文件中读入一个数据，读入过程中将忽略前导空格（回车、TAB），并将遇到第一个空格（回车、TAB）时结束输入

write(const char *buf,int num)；把内存中的一块内容写到一个输出文件流中，长度参数指出写的字节数。该函数遇到空字符时并不停止，因而能够写入完整的类结构，该函数带2个参数，一个char型指针（指向内存数据的起始地址）和一个所写的字节数。注意在该结构的对象地址之前要char做强制类型转换。

read(char *buf,int num)；从一个文件读字节到一个指定的存储器区域，由长度参数确定要读的字节数。虽然给出长度参数，但当遇到文件结束或者在文本模式文件中遇到文件结束标记时读结束。

文本文件读写举例

- 文件的随机读写(更多用于二进制文件)
  - 一个文件流保存一个内部指针以指出下一次读写数据的位置。输出/输入流随机访问函数有seekp（下一次写数据的位置）和tellp返回seekp()函数指针值。
  - seekp(偏移量，参照位置)
  - beg=0, //文件开头
  - cur=1, //文件指针的当前位置
  - end=2 //文件结尾
  - 偏移量：被定义为long型，以字节数为单位。
- 使用seekg可以实现面向记录的数据管理系统，用固定长度的记录大小乘以记录号便得到相对于文件头的字节位置，然后使用get读这个记录。

文件流的几点说明

- 1、将文件流与新文件绑定
  - ifstream in("in.txt");
  - in.close();
  - in.open("in2.txt");
- 2、清除文件流状态
  - ifstream in("in.txt");
  - in.close();
  - in.clear();　　//很重要！！！
  - in.open("in2.txt");
  - 如果没有对clear函数的调用，则上次的文件操作状态会保留到对in2.txt的操作中。

文件流的几点说明

- 3、反复使用一个文件流操作文件

```
ifstream& open_file(ifstream& in, const string& file)
{
    in.close();
    in.clear();
    in.open(file.c_str());
    return in;
}
```

- 说明：
  - 保证了in文件流对象每次重新打开新文件时状态时初始状态，从而上次操作的状态不会影响下次文件操作。

文件流的几点说明

- 4、文件流状态判断

成员函数	功能
eof()	
fail()	
bad()	
good()	如果流处于有效状态，在 返回true
clear()	将流的所有状态值重设为有效状态
clear(flag)	将流的某个指定条件状态设置为有效，flag类型strm::iostate
setstate(flag)	给流添加指定条件。flat类型为strm::iostate

字符串流<sstream>库

- **C++支持内存的 输入输出**
  - 将流与程序内存中的string对象绑定，就可以使用iostream输入输出操作符读取该字符串。
- **声明头文件：**sstream
- 三种流类：每个类都有一个对应的宽字符集版本
  - istringstream：读string的功能
  - ostringstream：写string的功能
  - stringstream，读写string的功能
- 举例：stringstream的使用

stringstream的使用

- 特定操作

stringstream strm ：创建自由的stringstream对象
stringstream strm(s)：创建存储s的副本的stringstream对象，其中s为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

- 提取一行中的单词

```
string line, word;
while(getline(cin, line)){
    istringstream sstrm(line);//创建一个stringstream对象
    while(sstrm>>word)  {
        //对单词的处理 ，如：cout<<word<<endl;
    }
}
```

stringstream的使用

- 提供类型转换/格式化
  - 可将数据先写入到stringstream流对象中，从而自动转换为string，使用时再从stringstream流对象中读取，还原为原来的类型。
  - 例如：

分隔源字符串与数值很重要，否则就无法读出来

```
int main(){
    int num1, num2;
    ostringstream ostrm;
    cin>>num1>>num2;
    ostrm<<"num1: "<<num1<<"\nnum2: "<<num2<<endl;
    cout<<ostrm.str()<<endl;
    istringstream istrm(ostrm.str());
    cout<<istrm.str()<<endl;
    int value1, value2;
    string temp;
    istrm>>temp>>value1;
    cout<<temp<<value1<<endl;
    istrm>>temp>>value2;
    cout<<temp<<value2<<endl;

    return 0;
}
```

文件举例(CH1001_01)

- 现有数据文件CH1001_01_data.txt，每行存放一个英文句子，请读取每个句子，并对句子中的单词按字母顺序进行升序排序，然后输出到屏幕上，要求每个句子占一行，句子中单词之间用空格分隔。