燕山大學
YANSHAN UNIVERSITY

C++面向对象程序设计

实验指导书

（9）流类库与输入/输出

燕山大学软件工程系

目　录

实验 09 流类库与输入/输出

1.1 时间安排 2 学时

本实验安排 2 个实验课时。

1.2 实验目的和要求

1．熟悉流类库中给类之间的关系，以及成员函数的用法。

2．理解 C++I/O 输入输出的三大种类，即标准输入输入流、文件输入输出流及字符串输入输出流，掌握对各类流常规的操纵方法。

3．理解输出格式控制的两种方法，即操纵符方法和流成员函数方法；并能够简单应用。

1.3 实验内容 I (调试、理解、体会、掌握)

（1）I/O 流输出格式控制有两种方法，一是使用控制符(或称之为操纵符，包含在<iomanip>中)控制输出格式，二是使用流对象的成员函数控制输出格式。

注意：各控制命令的作用范围不同，有的命令仅对当前行起作用，如 setw 或 width；有的命令则一直起作用，直至该标志被清除，如 setprecision 等。

A）使用控制符控制格式输出

```
#include <iostream>
#include <iomanip> //不要忘记包含此头文件
using namespace std;
int main()
{
    int a;
    cout<<"input a:";
    cin>>a;
    cout<<"dec:"<<dec<<a<<endl; //以上进制形式输出整数
    cout<<"hex:"<<hex<<a<<endl; //以十六进制形式输出整数a
    cout<<"oct:"<<setbase(8)<<a<<endl;//以八进制形式输出整数a
    char *pt="China";
    //pt指向字符串”China”
    cout<<setw(10)<<pt<<endl; //指定域宽为，输出字符串
    cout<<setfill('*')<<setw(10)<<pt<<endl;//指定域宽，输出字符串，空白处以“*”填充
    double pi=22.0/7.0; //计算pi值
    cout<<setiosflags(ios::scientific)<<setprecision(8);//按指数形式输出，位小数
    cout<<"pi="<<pi<<endl; //输出pi值
    cout<<"pi="<<setprecision(4)<<pi<<endl;//改为位小数
    cout<<"pi="<<setiosflags(ios::fixed)<<pi<<endl;//改为小数形式输出
    cout<<resetiosflags(ios::fixed);          //取消按点输出显示
    cout<<resetiosflags(ios::scientific);         //取消按点输出显示
```

```
    cout<<"pi="<<pi<<endl;//改为小数形式输出
    return 0;
}
```

B）使用流对象的成员函数控制输出格式

```
#include <iostream>
using namespace std;
int main()
{   int a;
    cin>>a;
    cout.setf(ios::showbase); //设置输出时的基数符号
    cout<<"dec:"<<a<<endl; //默认以十进制形式输出a
    cout.unsetf(ios::dec); //终止十进制的格式设置
    cout.setf(ios::hex); //设置以十六进制输出的状态
    cout<<"hex:"<<a<<endl; //以十六进制形式输出a
    cout.unsetf(ios::hex); //终止十六进制的格式设置
    cout.setf(ios::oct); //设置以八进制输出的状态
    cout<<"oct:"<<a<<endl; //以八进制形式输出a
    cout.unsetf(ios::oct); //终止以八进制的输出格式设置
    char *pt="China"; //pt指向字符串"china"
    cout.width(10); //指定域宽为
    cout<<pt<<endl; //输出字符串
    cout.width(10); //指定域宽为
    cout.fill('*'); //指定空白处以'*'填充
    cout<<pt<<endl; //输出字符串
    double pi=22.0/7.0; //计算pi值
    cout.setf(ios::scientific);//指定用科学记数法输出
    cout<<"pi="; //输出"pi="
    cout.width(14); //指定域宽为
    cout<<pi<<endl; //输出"pi值
    cout.unsetf(ios::scientific); //终止科学记数法状态
    cout.setf(ios::fixed); //指定用定点形式输出
    cout.width(12); //指定域宽为
    cout.setf(ios::showpos); //在输出正数时显示"+"号
    cout.setf(ios::internal); //数符出现在左侧
    cout.precision(6); //保留位小数
    cout<<pi<<endl; //输出pi，注意数符"+"的位置
    return 0;
}
```

（2）文本或二进制文件复制方法

方法一：逐个字符复制

源码 1：

```
#include < fstream >
#include < string >
```

```
#include <iostream>
using namespace std;
void main(){
	ifstream input("in.jpg",ios::binary);
	ofstream output("out.jpg",ios::binary);
	char ch;
	while (input.get(ch)) output << ch;
}
```

源码 2：

```
#include <fstream>
#include <fstream>
#include <iostream>
using namespace std;
void main()
{
	char ch;
	ifstream file("E:/test.txt");//读取c盘的文本文件
	ofstream file1("E:/test1.txt");//创建文本文件
	//ofstream file1("C:/test1.txt",ios::app);//用此语句替换上一条语句，连续运行两次程序，看会有何不同
	file.get(ch);
	while(!file.eof())//读取文本中的内容
	{
		cout << ch;//输出文本内容到控制台
		file1<<ch;//写入内容到文件
		file.get(ch);
	}
	file.close();	//关闭文件流
	file1.close();
	cout<<endl;
}
```

注意：如果使用input>>ch读取字符，则必须先调用input.unsetf(ios::skipws)取消输入流默认的跳过空白符的输入格式，因为换行符是空白符的一种。

方法二：逐行复制

源码1：

```
#include < fstream >
#include < string >
#include <iostream>
using namespace std;
void main(){
	ifstream input("in.jpg",ios::binary);
	ofstream output("out.jpg",ios::binary);
	string line;
	while (getline(input,line))
```

```
        output << line << "\n";
}
```

源码2：

```
#include < fstream >
#include < string >
#include <iostream>
using namespace std;
void main(){
    ifstream input("E:/test.txt");
    ofstream output("E:/tes1.txt");
    string line;
    while (getline(input,line))
        output << line << "\n";
}
```

注意：这里的代码有一个小小的缺陷，如果文件不是纯文本格式的文件，或者文本文件的最后没有换行符，那么会导致复制后的文件末尾添加了一个多余的换行符。

方法三：迭代器复制　以二进制文件为例

```
#include < fstream >
#include < iterator >
#include < algorithm >
#include <iostream>
using namespace std;
void main()
{
  ifstream input("in.jpg",ios::binary);   //在当前文件夹下建立一个任意的二进制文件，如*.jpg,*.bmp,*.mp3
  ofstream output("out.jpg",ios::binary);
  input.unsetf(ios::skipws);
  typedef istream_iterator<char> in;
  typedef ostream_iterator<char> out;
  copy(in(input),in(),out(output));
}
```

注意：同样这里也有一个小技巧，输入流的格式默认为跳过空白字符，因此调用 **unsetf** 取消这个格式，才可保证正确的复制。

方法四：缓冲区复制　以二进制文件为例

```
#include < fstream >
#include < string >
#include <iostream>
using namespace std;
void main(){
    ifstream input("in.jpg",ios::binary);
```

```
    ofstream output("out.jpg",ios::binary);
    output << input.rdbuf();
}
```

注意：这里直接使用了输入流的缓冲区，因此没有引入额外的临时对象。

分析：很显然，上述四种方法中，最后一种方法最简洁，由于直接操作输入流的缓冲区，从运行效率上来说，也比其他方法有着略微的优势(当然，由于操作系统可能提供了额外的基于设备的文件缓冲机制，也许你无法证实这一点)。因此，除非要对输入内容进行处理，直接复制文件推荐最后一种方法，既不容易出错，又能获得良好的性能。

另外，对文件进行更改、删除、插入等操作，可以直接用以上方法也可以先把文件读入 **vector<string>**，处理后再输出，不过当文件很大的时候，**vector** 占用内存空间较大，而且输出时会破坏原文件格式，尽量不使用。

（3）调试《C++语言程序设计》教材第 498 页的综合实例（例 11-13），并仔细体会输入输出运算符的重载、文件操作等。

（4）设计一个管理图书的简单程序，描述一本书的信息包括：书号，书名，出版社和作者等。提供的基本功能包括：可连续将新书存入文件“book.dat”中，新书信息加入到文件的尾部；也可以根据输入的书名进行查找；把文件“book.dat”中同书名的所有书显示出来。

分析：可以把描述一本书的信息定义为一个 **Book** 类，它包含必要的成员函数。把加入的新书总是加入到文件尾部，所以，以增补方式打开输出文件。从文件中查找书时，总是从文件开始位置查找，以读方式打开文件。用一个循环语句实现可连续地将新书加入文件或从文件中查找指定的书名。由于是以一个 **Book** 类的实例进行文件输入输出的，所以，这文件的类型应该是二进制文件。

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
class   Book
{
    long int    num;                //书号
    char bookname[40];              //书名
    char publicname[40];            //出版社
    char name[20];                  //作者
public:
    Book(){ num=0; bookname[0] =0;publicname[0] =0; name[0] =0;}
    char * Getbookname(void)    { return bookname ;}
    long    Getnum(void ) { return num;}
    void Setdata(long , char *,char *,char *);
    void Show(void );
    Book(long , char *,char *,char *);
};
```

```
void    Book::Setdata(long nu , char *bn,char *p,char *n)
{
      num = nu; strcpy(bookname,bn);
      strcpy(publicname,p); strcpy(name,n);
}
void    Book::Show(void )
{
      cout<<"书号:"<<num<<'\t'<<"书名:"<<bookname<<'\t';
      cout<<"出版社:"<<publicname<<'\t'<<"作者:"<<name<<'\n';
}
Book::Book(long nu, char * bp,char *p,char *n)
{
      Setdata(nu , bp, p, n);
}
void    main(void)
{
      Book    b1,b2;
      long    nu;
      char bn[40];                                    //书名
      char pn[40];                                    //出版社
      char na[20];                                    //作者
      ifstream    file1;
      ofstream    file3;
      char flag = 'y';
      while( flag=='y' ||flag=='Y') //由flag控制循环
      {
            cout<<"\t\t 1: 按书名查找一本书!\n";
            cout<<"\t\t 2: 加入一本新书!\n";
            cout<<"\t\t 3: 退出!\n输入选择： ";
            int f;
            cin>>f;
            switch(f)
            {
             case 1:
                  cout<<"输入要查找的书名： ";
                  cin>>bn;
                  file1.open("book.dat",ios::in | ios::binary);//按读方式打开文件
                  while(!file1.eof())
                  {
                        int n;
                        file1.read((char *)&b1,sizeof(Book));
                        n=file1.gcount();
                        if(n==sizeof(Book))
                        {
```

```
                    if(strcmp(b1.Getbookname(),bn)==0)    //若两字符数组相等，则显示书的信息
                        b1.Show();
                }
            }
            file1.close();
            break;
        case 2:
            cout<<"输入书号："; cin>>nu;
            cout<<"输入书名："; cin>>bn;
            cout<<"输入出版社："; cin>>pn;
            cout<<"输入作者："; cin>>na;
            b1.Setdata(nu,bn,pn,na);
            file3.open("book.dat",ios::app|ios::binary);//增补方式打开文件
            file3.write((char*)&b1,sizeof(b1));
            file3.close();
            break;
        default:   flag = 'n';
        }
    }
}
```

1.4 实验内容 II　(自主完成)

注：将题目的构思过程、源码、运行结果（截图）、心得体会等内容按要求填写，详见实验报告模板。

1、编写一个程序：

（1）新建一个文本文件，里面有若干行文本。

（2）打开指定的文本文件，在每一行前加行号后将其输出到另一个文本文件中。

（3）改变文本文件路径，实现上述功能，理解绝对路径和相对路径。

2、选作

完善本实验指导书中题目（4）（图书管理）的代码，使之具有如下功能：

按出版社、作者从文件中查找书名；

输出同一出版社出版的所有书名；

输出文件中的所有书名等功能。