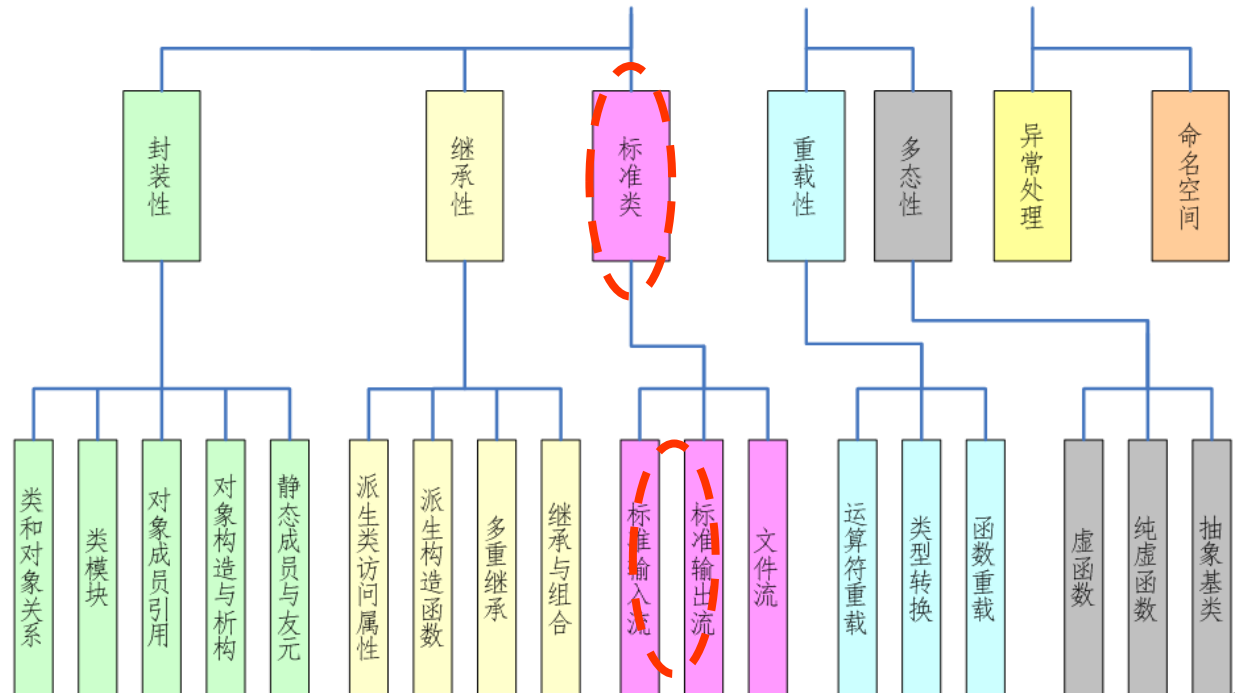


第10讲 输入输出流

10.1 文件操作与文件流

10.2 字符串流*

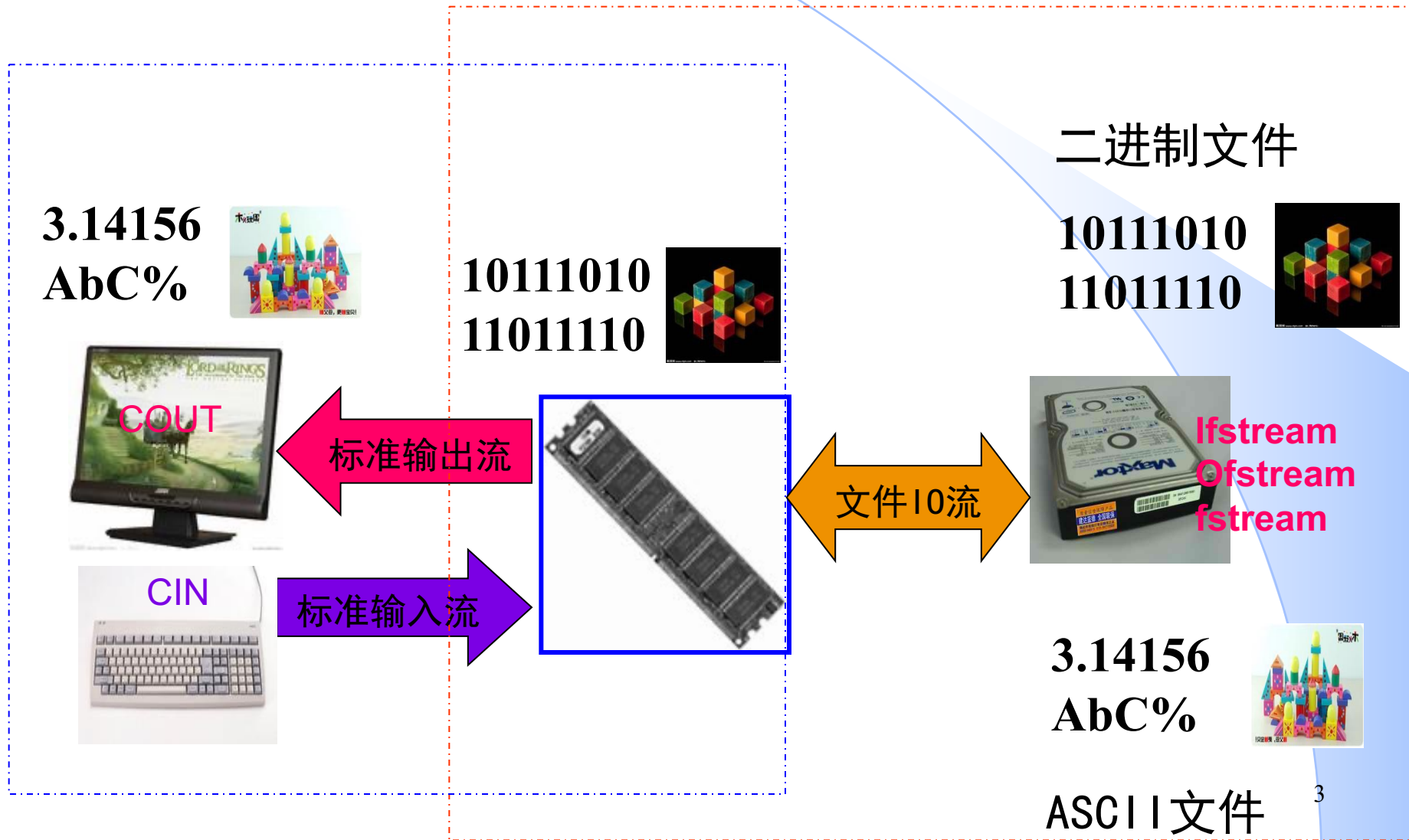
C++的OOP程序 = 对象 + 消息 + 工具



重 要 通 知

- 5月2号课程暂停，5月9号照常上课。请各位同学相互转达。

10.1 文件操作与文件流

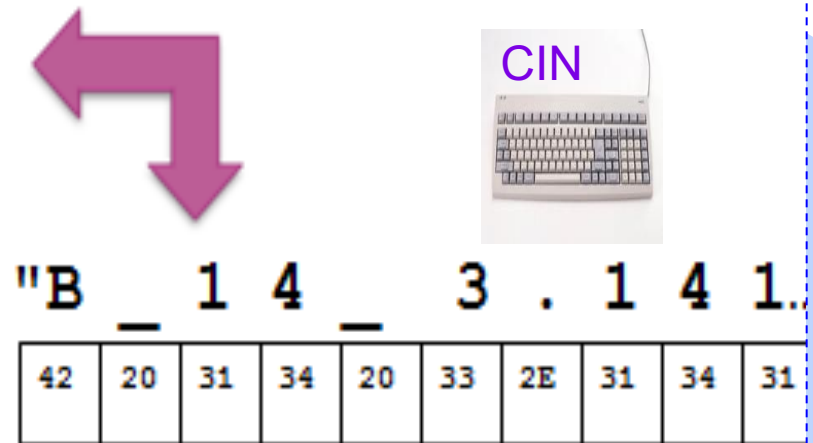


10.1 文件操作与文件流

- cin和cout能实现内存中的二进制数据和输入输出字符流之间的转换过程

```

0x42 char code;
'B'
0x0000000E int i;
14
0x000000E8D4A51000 long long gdp;
10000000000000
0x00006441 float payRate;
14.25
0xE0864454EB210940 double pi;
3.1415926536
    
```



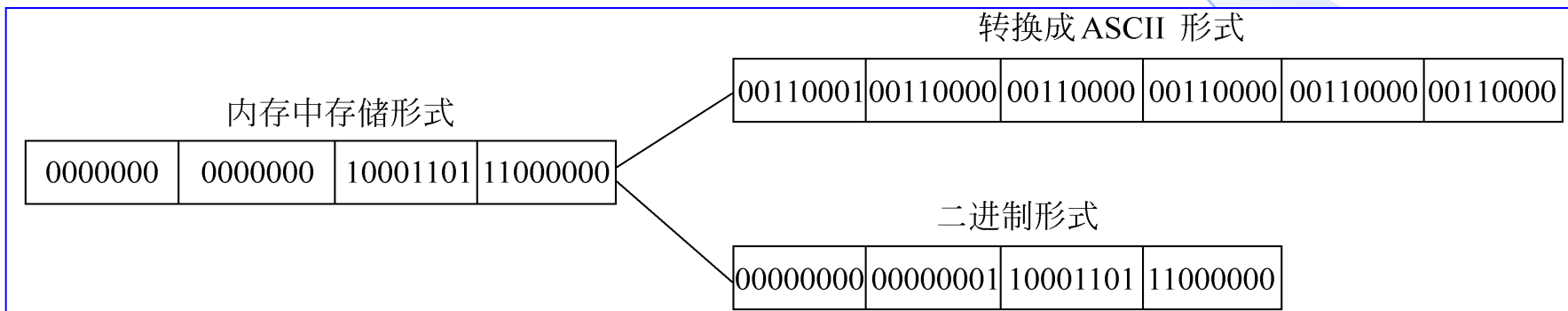
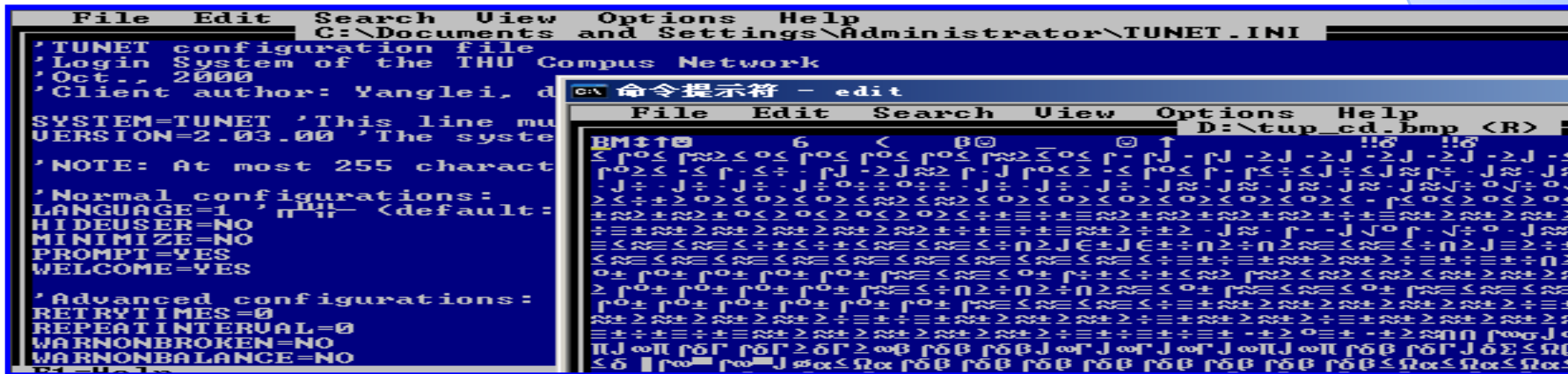
内存
(二进制形式)

输入、输出字符流
(字符串形式)

10.1 文件操作与文件流

- **文件的概念：**指存储在外部介质上数据集合。文件是外存的数据管理单位。
- **文件类型**
 - ✓ 程序文件、数据文件
 - ✓ ASCII文件、二进制文件

■ 提问：如何选择二进制或ASCII文件？

```
File Edit Search View Options Help
C:\Documents and Settings\Administrator\TUNET.INI
'TUNET configuration file
'Login System of the THU Compus Network
'Oct.. 2000
'Client author: Yanglei, d
SYSTEM=TUNET 'This line mu
VERSION=2.03.00 'The syste
'NOTE: At most 255 charact
'Normal configurations:
LANGUAGE=1 'n"ir- <default:
HIDEUSER=NO
MINIMIZE=NO
PROMPT=YES
WELCOME=YES
'Advanced configurations:
RETRYTIMES=0
REPEATINTERVAL=0
WARNONBROKEN=NO
WARNONBALANCE=NO
```

10.1 文件操作与文件流

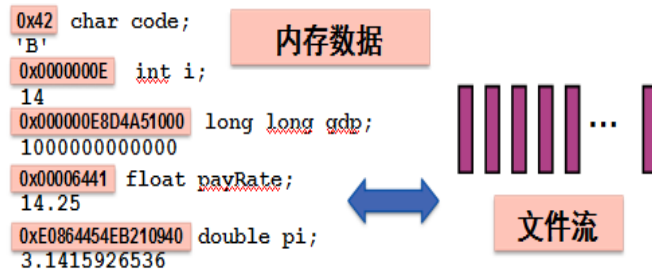
◆文件流类与文件流对象

文件流是以文件作为I/O对象的数据流, 每一个文件流都有一个内存缓冲区与之对应;

✓输出文件流: 程序→文件数据流;

✓输入文件流: 文件数据流→程序;

◆提问: 文件流和文件是同一概念吗?



◆C++定义3种文件类, 用于对文件的I/O操作

- (1) ifstream类, 支持文件输入; 例: ifstream infile;
- (2) ofstream类, 支持文件输出; 例: ofstream outfile;
- (3) fstream类, 支持文件输入输出。

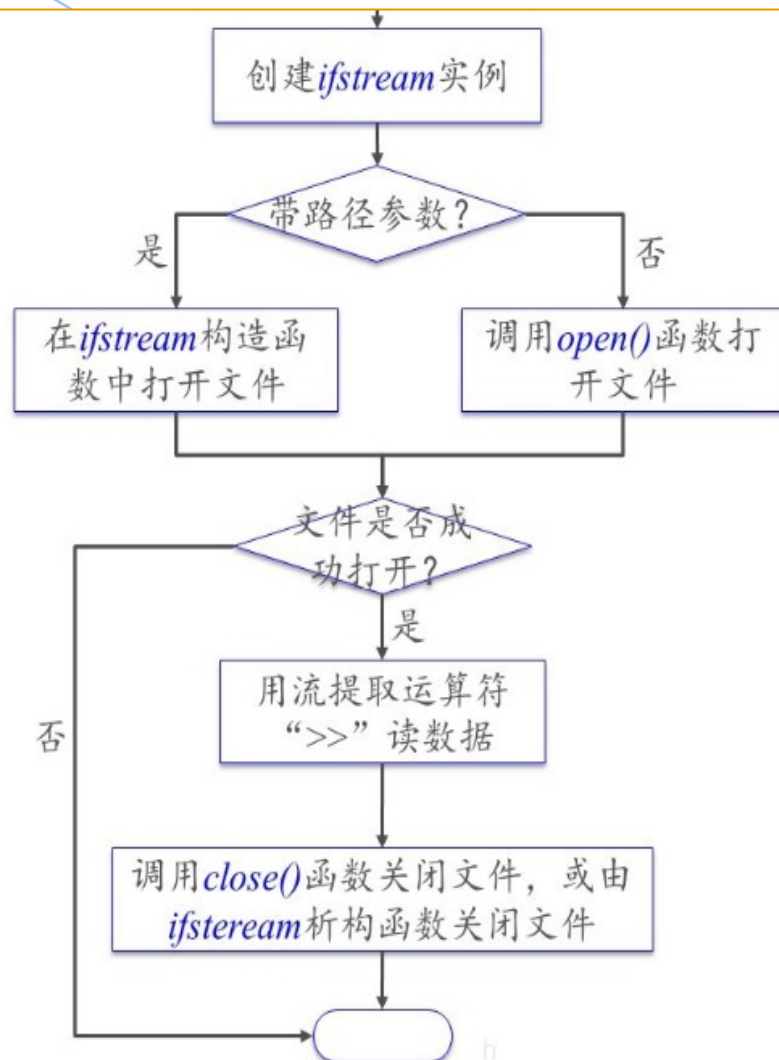
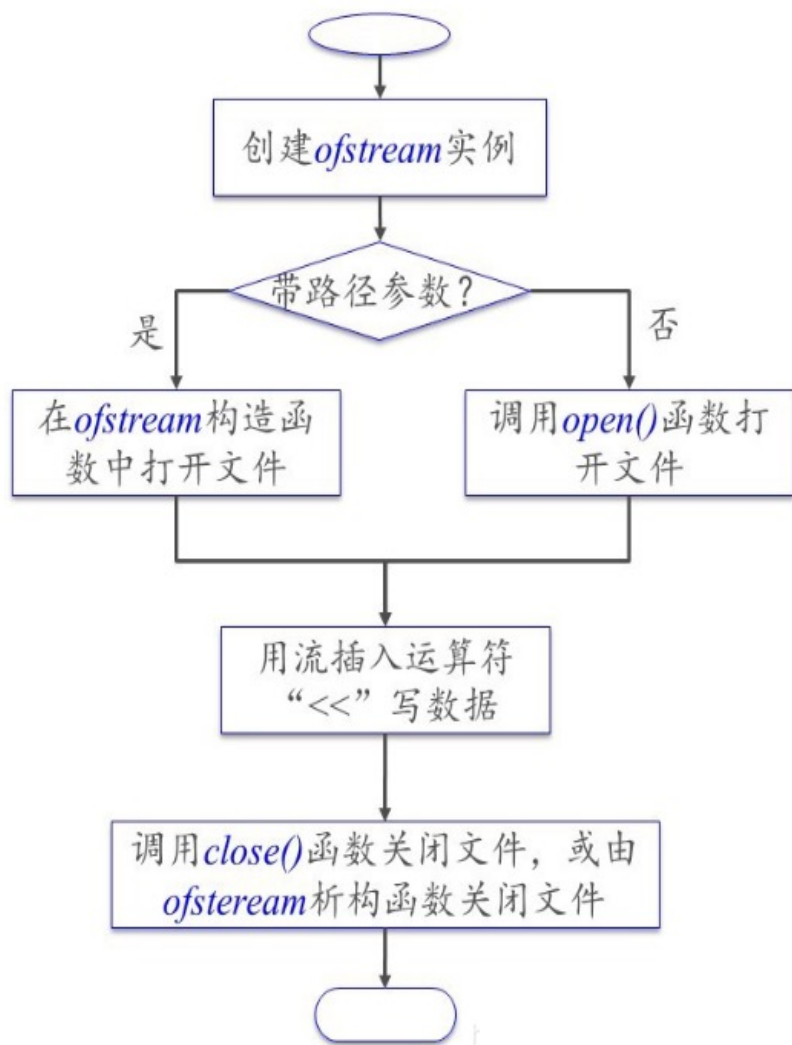
10.1 文件操作与文件流

- ◆文件操作：打开、关闭、读写和指针定位等；
- ◆打开文件：为文件流对象和指定的文件建立关联，以便使文件流流向指定的磁盘文件，并指定文件工作方式；
- ◆打开文件方式有2种
 - (1) 调用文件流的成员函数open。一般形式：
文件流对象.open(文件名，输入输出方式)；
例：

```
ofstream outfile;  
outfile.open( "f1.dat" , ios::out);
```
 - (2) 文件流类在声明时，定义了带参数的构造函数，其中包含了打开磁盘文件的功能。因此在定义文件流对象时调用该构造函数来实现“三个功能”
例：

```
ofstream outfile( "f1.dat" , ios::out);
```

10.1 文件操作与文件流



10.1 文件操作与文件流

表 13.6 文件输入输出方式设置值

方 式	作 用
<code>ios::in</code>	以输入方式打开文件
<code>ios::out</code>	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
<code>ios::app</code>	以输出方式打开文件,写入的数据添加在文件末尾
<code>ios::ate</code>	打开一个已有的文件,文件指针指向文件末尾
<code>ios::trunc</code>	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 <code>ios::out</code> 方式,而未指定 <code>ios::app</code> , <code>ios::ate</code> , <code>ios::in</code> ,则同时默认此方式
<code>ios::binary</code>	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
<code>ios::nocreate</code>	打开一个已有的文件,如文件不存在,则打开失败。 <code>nocreat</code> 的意思是不建立新文件
<code>ios::noreplace</code>	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, <code>noreplace</code> 的意思是不更新原有文件
<code>ios::in ios::out</code>	以输入和输出方式打开文件,文件可读可写
<code>ios::out ios::binary</code>	以二进制方式打开一个输出文件
<code>ios::in ios::binar</code>	以二进制方式打开一个输入文件

◆ 说明

- ① 每一个打开的文件都有一个文件指针；
- ② 可用“位或”运算符“|”对输入输出方式进行组合，例：
`outfile.open(“f2.dat”, ios::app|ios::nocreate);`
- ③ 如果打开操作失败，open函数的返回值为0（后来版本返回值为Void，需要使用is-open()来判断），如果是用调用构造函数的方式打开文件，则流对象的值为0
例：`if(!outfile.open(“f2.dat”, ios::app))
 {cout<<“open error”;exit(1);}`

◆ 关闭文件操作

- 关闭文件用成员函数close。如`outfile.close();`
- 关闭是解除该磁盘文件与文件流的关联，原来设置的工作方式也失效，这样，就不能再通过文件流对该文件进行I/O

10.1 文件操作与文件流

◆对ASCII文件

- ASCII文件：文件每个字节中均以ASCII代码形式存储，即一个字节存放一个字符；
- 程序可从ASCII文件中读出或写入若干个字符。

◆ASCII文件的读写方法

- 用流插入运算符“<<”和流提取运算符“>>”输入或输出标准类型的数据到文件；
- 用文件流的put(), get(), getline()等成员函数进行字符的读写。

□ 提问：为什么“<<”和“>>”可以用于文件流对象的I/O?

例. 有一个整型数组，含10个元素，从键盘输入10个整数给数组，将此数组送到磁盘文件中存放。

```

1 #include <fstream>
2 #include <iostream>
3 using namespace std;
4 int main()
5 {int a[10];
6  ofstream outfile("f1.dat");
7  if(!outfile)
8  {cerr<<"open error!"<<endl;
9   exit(1);
10 }
11 cout<<"enter 10 integer numbers:"<<endl;
12 for(int i=0;i<10;i++)
13 {cin>>a[i];
14  outfile<<a[i]<<" ";}
15 outfile.close();
16 return 0;
17 }

```

C:\WINDOWS\system32\cmd.exe

```

enter 10 integer numbers:
11 22 33 44 55 66 77 88 99 00
请按任意键继续. . .

```

C:\ 命令提示符 - edit

File Edit Search View Options Help

E:\vc-exercise\test2\test2\f1.dat

```
11 22 33 44 55 66 77 88 99 0
```

例. 从上例建立的数据文件f1. dat中读入10个整数放在数组中, 找出并输出10个数中的最大者和它在数组中的序号

```

1 #include <fstream>
2 #include <iostream>
3 using namespace std;
4 int main( )
5 {int a[10],max,i,order;
6  ifstream infile("f1.dat",ios::in);//定义输入文件流对象, 以输入方式打开磁盘文件f1.dat
7  if(!infile)
8      {cerr<<"open error!"<<endl;
9        exit(1); }
10 for(i=0;i<10;i++)
11     {infile>>a[i];//从磁盘文件读入10个整数, 顺序存放在a数组中
12      cout<<a[i]<<" ";} //在显示器上顺序显示10个数
13 cout<<endl;
14 max=a[0];
15 order=0;
16 for(i=1;i<10;i++)
17     if(a[i]>max)
18     {max=a[i]; //将当前最大值放在max中
19      order=i; } //将当前最大值的元素序号放在order中
20 cout<<"max="<<max<<endl<<"order="<<order<<endl;
21  infile.close();
22  return 0;}

```

```

C:\WINDOWS\system32\cmd.exe
3 4 5 3 4 5 3 56 7 6
max=56
order=7
请按任意键继续. . .

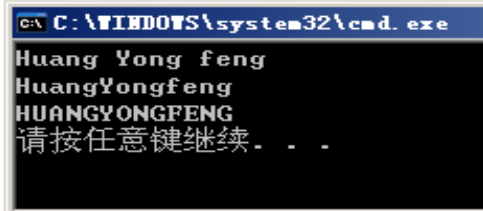
```

例. 从键盘读入一行字符, 把其中的字母字符依次存放在磁盘文件 f2.dat 中。再把它从磁盘文件读入程序, 将其中的小写字母改为大写字母, 再存入磁盘文件 f3.dat。

```

1 #include <fstream>
2 #include <iostream>
3 using namespace std;
4 void save_to_file()//从键盘读入一行字符, 并将其中的字母存入磁盘文件
5 {ofstream outfile("f2.dat");//定义输出文件流对象outfile, 以输出方式打开磁盘文件f2.dat
6 if(!outfile) {cerr<<"open f2.dat error!"<<endl;exit(1); }
7 char c[80]; cin.getline(c,80);//从键盘读入一行字符
8 for(int i=0;c[i]!='\0';i++) //对字符逐个处理, 直到遇'\0'为止
9 if(c[i]>=65 && c[i]<=90||c[i]>=97 && c[i]<=122)//如果是字母字符
10 {outfile.put(c[i]); //将字母字符存入磁盘文件f2.dat
11 cout<<c[i];} //同时送显示器显示
12 cout<<endl; outfile.close(); } //关闭f2.dat
13
14 void get_from_file()//从磁盘文件f2.dat读入字符, 将小写改为大写, 再存入f3.dat
15 {char ch;
16 ifstream infile("f2.dat",ios::in); //定义输入文件流infile, 以输入方式打开f2.dat
17 if(!infile) {cerr<<"open f2.dat error!"<<endl; exit(1); }
18 ofstream outfile("f3.dat");//定义输出文件流outfile, 以输出方式打开磁盘文件f3.dat
19 if(!outfile) {cerr<<"open f3.dat error!"<<endl; exit(1); }
20 while(infile.get(ch))//当读取字符成功时执行下面的复合语句
21 {if(ch>=97 && ch<=122) //判断ch是否为小写字母
22 ch=ch-32; //将小写字母变为大写字母
23 outfile.put(ch); //将该大写字母存入磁盘文件f3.dat
24 cout<<ch; } //同时在显示器输出
25 cout<<endl;
26 infile.close(); //关闭磁盘文件f2.dat
27 outfile.close(); //关闭磁盘文件f3.dat
28 int main()
29 {save_to_file(); //从键盘读入一行字符并将其中的字母存入磁盘文件f2.dat
30 get_from_file(); //从f2.dat读入字母字符, 改为大写字母, 再存入f3.dat
31 return 0;}

```



```

C:\WINDOWS\system32\cmd.exe
Huang Yong feng
HuangYongfeng
HUANGYONGFENG
请按任意键继续. . .

```



```

命令提示符 - edit
File Edit Search
huangyongfeng

命令提示符 - edit
File Edit Search
HUANGYONGFENG

```

在软件调试时，一个好技巧是将软件运行中的一些信息同时写到“日志文件”和屏幕上，以便程序员查看信息。

```
std::ofstream output("debug.log", ios::out);
output << __FILE__ << ":" << __LINE__ << "\t" << "Variable x = " << x;
cout << __FILE__ << ":" << __LINE__ << "\t" << "Variable x = " << x;
```

```
int main()
{
    std::ofstream output("debug.log");
    //1、创建文件/屏幕输出流对象tee
    teestream tee(std::cout, output);

    auto x = 1.1;
    tee << __FILE__ << ":" << __LINE__ << "\t" << "Variable x = " << x;

    return 0;
}
```

 debug.log - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

D:\C++\CppCode\fwirte_test\main.cpp:105

Variable x = 1.1

10.1 文件操作与文件流

◆ 二进制文件的读写操作

- 二进制文件：内存中数据存储形式不加转换地传送到磁盘文件，又称为内存数据的映像文件或为字节文件。
- 打开时用 `ios::binary` 指定为以二进制形式。二进制文件除了可以作为输入文件或输出文件外，还可是既能输入又能输出的文件。
- 成员函数 `read` 和 `write` 读写二进制文件

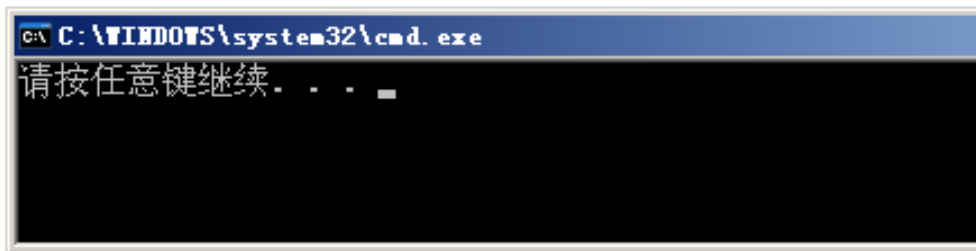
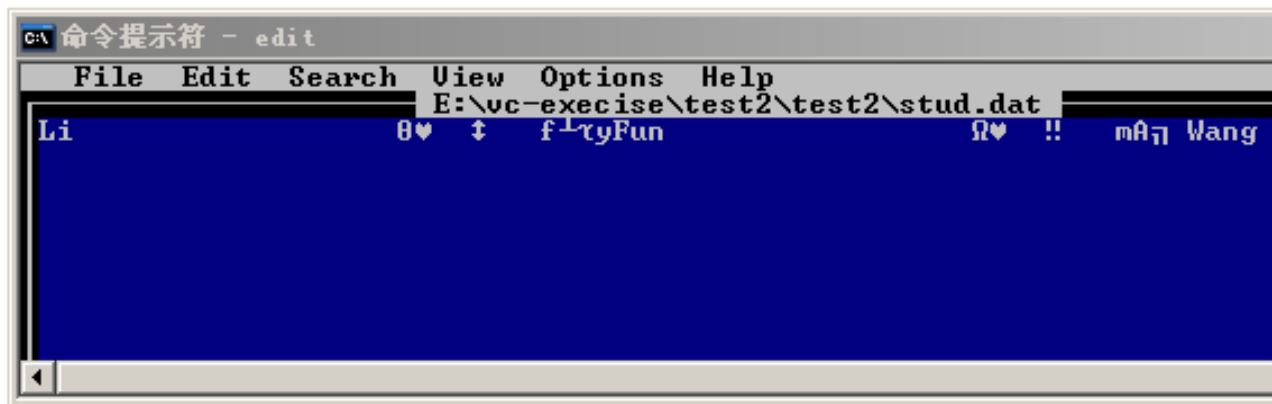
```
istream& read(char *buffer, int len);  
ostream& write(const char * buffer, int len);
```
- 字符指针 `buffer` 指向内存中一段存储空间。`len` 是读写的字节数。调用的方式为 `a.write(p1, 50); b.read(p2, 30);`。

例. 将一批数据以二进制形式存放在磁盘文件中

```

1 #include <iostream>
2 #include <fstream>
3 using namespace std;
4 struct student
5 { char name[20];
6   int num;
7   int age;
8   char sex;
9 };
10 int main()
11 { student stud[3]={"Li",1001,18,'f',"Fun",1002,19,'m',"Wang",1004,17,'f'};
12   ofstream outfile("stud.dat",ios::binary);
13   if(!outfile)
14   { cerr<<"open error!"<<endl;
15     abort();
16   }
17   for(int i=0;i<3;i++)
18     outfile.write((char *)&stud[i],sizeof(stud[i]));
19   outfile.close();
20   return 0;
21 }

```



outfile.write((char*)&stud[0],sizeof(stud));

执行一次write函数即输出了结构体数组的全部数据

例. 将上例中以二进制形式存放在磁盘文件中的数据读入内存并在显示器上显示

```

1 #include <fstream>
2 #include <iostream>
3 using namespace std;
4 struct student
5 { char name[10];
6   int num;
7   int age;
8   char sex; };
9 int main( )
10 { student stud[3];
11   int i;
12   ifstream infile("stud.dat",ios::binary);
13   if(!infile)
14   { cerr<<"open error!"<<endl;
15     abort( ); }
16   for(i=0;i<3;i++)
17   infile.read((char*)&stud[i],sizeof(stud[i]));
18   infile.close( );
19   for(i=0;i<3;i++)
20   { cout<<"NO."<<i+1<<endl;
21     cout<<"name:"<<stud[i].name<<endl;
22     cout<<"num:"<<stud[i].num<<endl;;
23     cout<<"age:"<<stud[i].age<<endl;
24     cout<<"sex:"<<stud[i].sex<<endl<<endl; }
25   return 0; }

```

C:\WINDOWS\system32\cmd.exe

```

NO.1
name:Li
num:0
age:0
sex:

NO.2
name:↑
num:0
age:0
sex:

NO.3
name:
num:12271981
age:1735287127
sex:

```

请按任意键继续. . .

文件指针操作

打开文件时分配一个文件指针，指明当前应进行读写位置

表 13.7 文件流与文件指针有关的成员函数

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

- `infile.seekg(100)` ; 指针向前移到100字节位置
- `infile.seekg(-50, ios::cur)` ; 从当前位置后移50字节
- `outfile.seekp(-5, ios::end)` ; 指针从文件尾后移5字节

◆ **随机访问数据文件实例：**利用成员函数移动指针，随机地访问文件中任一位置上的数据

假设：有5个学生的数据，要求：

- (1) 把它们存到磁盘文件中；
- (2) 将磁盘文件中第1, 3, 5个学生数据读入程序显示出来；
- (3) 将第3个学生的数据修改后存回磁盘文件中的原有位置；
- (4) 从磁盘文件读入修改后5个学生的数据并显示出来。

◆ **需要解决的问题**

- (1) 由于同一文件频繁进行输入和输出，需将文件工作方式指定为输入输出文件，`ios::in|ios::out|ios::binary`；
- (2) 正确计算好每次访问时指针的定位，使用`seekg`或`seekp`函数移动指针；
- (3) 正确进行文件中数据的重写(更新)。

◆ **问题：**在下例中是否可以`ifstream/ofstream`类定义输入输出的二进制文件流对象？

```

1 #include <iostream>
2 #include <fstream>
3 using namespace std;
4 struct student
5 {int num;
6  char name[20];
7  float score;};
8 int main()
9 {int i;
10  student stud[5]={1001,"Li",85,1002,"Fun",97.5,1004,"Wang",54,
11                  1006,"Tan",76.5,1010,"ling",96};
12  fstream iofile("stud.dat",ios::in|ios::out|ios::binary);
13  if(!iofile)
14  {cerr<<"open error!"<<endl;
15   abort(); }
16  for(i=0;i<5;i++)
17   iofile.write((char *)&stud[i],sizeof(stud[i]));
18  student stud1[5];
19  for(i=0;i<5;i=i+2)
20   {iofile.seekg(i*sizeof(stud[i]),ios::beg);
21    iofile.read((char *)&stud1[i/2],sizeof(stud1[i]));
22    cout<<stud1[i/2].num<<" "<<stud1[i/2].name<<" "<<stud1[i/2].score<<endl;
23   cout<<endl;
24   stud[2].num=1012;
25   strcpy(stud[2].name,"Wu");
26   stud[2].score=60;
27   iofile.seekp(2*sizeof(stud[0]),ios::beg);
28   iofile.write((char *)&stud[2],sizeof(stud[2]));
29   iofile.seekg(0,ios::beg);
30   for(i=0;i<5;i++)
31    {iofile.read((char *)&stud[i],sizeof(stud[i]));
32     cout<<stud[i].num<<" "<<stud[i].name<<" "<<stud[i].score<<endl; }
33  iofile.close();
34  return 0;}

```

```

1001 Li 85
1004 Wang 54
1010 ling 96

1001 Li 85
1002 Fun 97.5
1012 Wu 60
1006 Tan 76.5
1010 ling 96
请按任意键继续. . .

```

10.2 字符串流

◆字符串流:以内存中用户定义的字符数组(字符串)作为I/O对象,也称为**内存流**;

➤输入字符串流:内存中的**字符数组**→(缓冲区)→**程序**;

➤输出字符串流: **程序**→(缓冲区)→**内存**中的字符数组;

◆字符串流也有相应的**缓冲区**。如果向字符数组存入数据,当流缓冲区满(或遇换行符),一起存入字符数组。如果是从字符数组读数据,先将字符数组中的数据送到流缓冲区,然后从缓冲区中提取数据赋给有关变量。

◆字符串流类

➤Istream: 输入字符串类;

➤Ostream: 输出字符串类;

➤Strstream: 输出/输入字符串流类。



1. 建立输出字符串流对象

➤ ostream类提供的构造函数的原型

```
ostream::ostream(char *buf, int n, int mode =  
ios::out);
```

其中:buf是指向字符数组首元素的指针，n为指定的流缓冲区的大小，第3个参数是可选，默认为ios::out方式。

➤ 建立输出字符串流对象并与字符数组建立关联。

```
ostream strout(ch1, 20);
```

作用：建立输出字符串流对象strout，并使strout与字符数组ch1关联，流缓冲区大小为20。

3个字符串流类都是在头文件<sstream>中定义的，程序中用到istringstream、ostringstream和stringstream类时应包含头文件<sstream>

2. 建立输入字符串流对象

◆ `istream`类提供了两个带参的构造函数，原型为

➤ `istream::istream(char *buffer);`

➤ `istream::istream(char *buffer, int n);`

`buffer`指向字符数组首地址，使流对象与字符数组建立关联

◆ 建立输入字符串流对象：`istream strin(ch2);`

建立对象`strin`，使用字符数组`ch2`全部数据

◆ `istream strin(ch2, 20)`：只将字符数组`ch2`中前20个字符作为输入字符串流的内容。

3. 建立输入输出字符串流对象

◆ `stringstream`类提供的构造函数原型为

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

◆ 建立输入输出字符串流对象：

`stringstream strio(ch3, sizeof(ch3), ios::in|ios::out);`

以数组`ch3`为输入输出对象，流缓冲区大小与数组`ch3`相同。


```

1 #include <sstream>
2 #include <iostream>
3 using namespace std;
4 struct student
5 {
6     int num;
7     char name[20];
8     float score;
9 };
10 int main()
11 {
12     student stud[3]={1001,"Li",78,1002,"Wang",89.5,1004,"Fun",90};
13     char c[50];
14     ostringstream strout(c,30);
15     for(int i=0;i<3;i++)
16         strout<<stud[i].num<<stud[i].name<<stud[i].score;
17     strout<<ends;
18     cout<<"array c:"<<endl<<c<<endl;
19     return 0;
20 }

```

C:\WINDOWS\system32\cmd.exe

```

array c:
1001Li781002Wang89.51004Fun90
请按任意键继续. . .

```

◆ 字符数组c中的数据之间没空格，如果以后想将这些数据读回赋给程序中相应的变量，就会出现问题，因为无法分隔两个相邻的数据。

```

1 #include <sstream>
2 #include <iostream>
3 using namespace std;
4 int main()
5 { char c[50]="12 34 65 -23 -32 33 61 99 321 32";
6   int a[10],i,j,t;
7   cout<<"array c:"<<c<<endl;
8   istringstream strin(c, sizeof(c));
9   for(i=0;i<10;i++)
10    strin>>a[i];
11   cout<<"array a:";
12   for(i=0;i<10;i++)
13    cout<<a[i]<<" ";
14   cout<<endl;
15   for(i=0;i<9;i++)
16    for(j=0;j<9-i;j++)
17     if(a[j]>a[j+1])
18      {t=a[j];a[j]=a[j+1];a[j+1]=t;}
19   ostringstream strout(c, sizeof(c));
20   for(i=0;i<10;i++)
21    strout<<a[i]<<" ";
22   strout<<ends;
23   cout<<"array c:"<<c<<endl;
24   return 0;}

```

```

C:\WINDOWS\system32\cmd.exe
array c:12 34 65 -23 -32 33 61 99 321 32
array a:12 34 65 -23 -32 33 61 99 321 32
array c:-32 -23 12 32 33 34 61 65 99 321
请按任意键继续. . .

```

何种排序算法？

小 结

- ◆建立了两个字符串流`strin`和`strout`，与字符数组`C`关联。
`strin`从字符数组`C`中获取数据，`strout`将数据传送给字符数组。甚至可对字符数组交叉进行读写。
- ◆问 题 : 如 果 定 义 `stringstream`
`strio(c, sizeof(c), ios::in|ios::out);`则I/O需注意啥?
- ◆与字符串流关联的字符数组相当于内存中“临时仓库”，可用来存放各种类型数据(ASCII格式)，需要时再从中读回来；
- ◆优点：标准设备不能保存数据，字符数组中的内容可以随时用ASCII字符输出。比外存文件使用方便，不必建立文件(不需打开与关闭)，存取速度快；
- ◆缺点：生命周期与其所在的模块(如主函数)相同，该模块的生命周期结束后，字符数组也不存在了。

本讲重点分析

- 文件I/O类和文件流
- 文件操作函数
- 字符I/O类和字符I/O流
- 注意：>>、get()、getline、read()的使用差别
- 同理：<<、put()、write()的使用差别

流 I/O 操作



```

cin>>
cin.get()
cin.getline()
cin.putback()
  
```



```

cout<<
cout.put()
  
```



```

Outfile<<
Outfile.put()
Outfile.fwrite()
  
```

```

Infile>>
Infile.get()
Infile.getline()
Infile.read()
  
```



文件

第10次练习

本次作业2道必做题

1. 在第8次作业第1题的基础上，最少实现4个功能：新建文件、保持文件、修改文件、查询文件；
 - (1) 新建文件：根据用户输入的文件名，建立新数据文件；
 - (2) 保持文件：录入教师的数据等信息，保存在现有数据文件；
 - (3) 修改文件：根据工号读出教师信息，并进行修改；将修改后的数据写入在文件中的原来位置；
 - (4) 查询文件：输入教师姓名，从文件中读出教师信息呈现。
2. 修改本讲课件P23的程序实例，要求对写入到字符流strout中的数据读出，然后写入到一个二进制文件中。

选做题：阅读下列程序，写出执行结果

```
#include<iostream.h>
void main()
{double x=123.456;
cout.width(10);
cout.setf(ios::dec,ios::basefield);
cout<<x<<endl;
cout.setf(ios::left);
cout<<x<<endl;
cout.width(15);
cout.setf(ios::right,ios::left);
cout<<x<<endl;
cout.setf(ios::showpos);
cout<<x<<endl;
cout<<-x<<endl;
cout.setf(ios::scientific);
cout<<x<<endl;
}
```