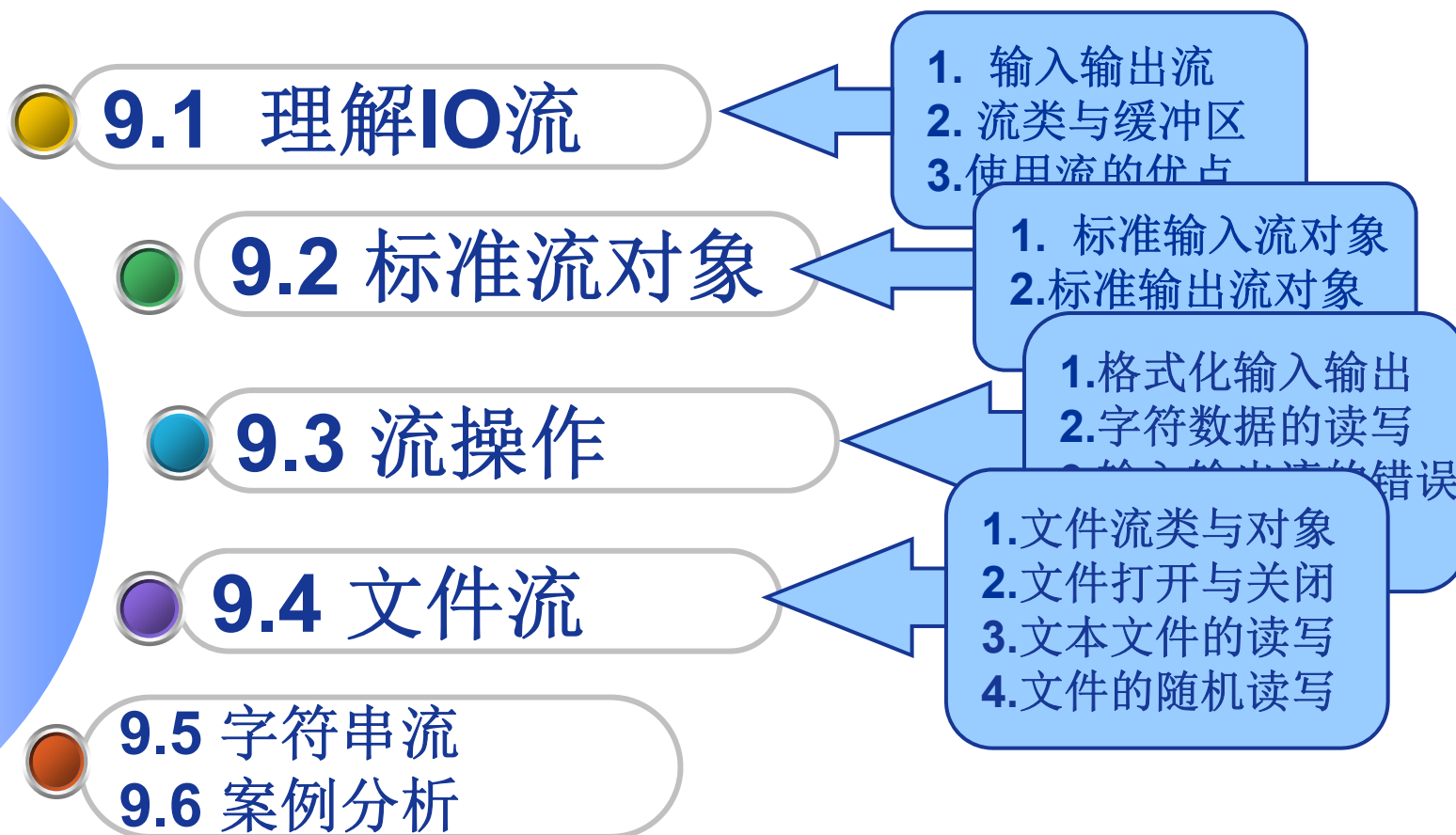




第9章 输入输出流

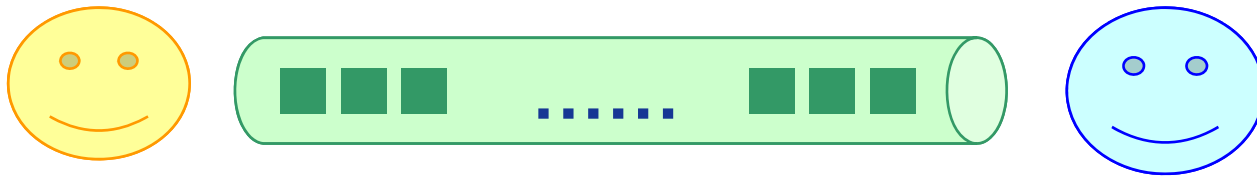
C++语言程序设计教程—大连理工大学软件学院

第9章 输入输出流



9.1 理解流

- 流(stream)表示信息从源到目的端的流动，负责建立数据生产者和消费者之间的联系，数据按顺序从一个对象传送到另一对象。



C++中把数据之间的传输操作称作“流”

I/O系统的任务就是在内存和外部设备之间稳定可靠地传输数据和解释数据。
程序中，对数据的输入/输出是以**字节流**实现的应用程序对字节序列作出各种数据解释。

9.1 理解流

9.1.1 I/O流

为了实现数据的有效流动，C++提供了庞大的I/O类库，包括：

(1) 标准I/O流

对系统指定的标准设备的I/O操作。

(2) 文件I/O流

以外存中的文件为对象进行输入和输出。

(3) 字符串流

对内存中指定空间进行输入和输出。

9.1 理解流

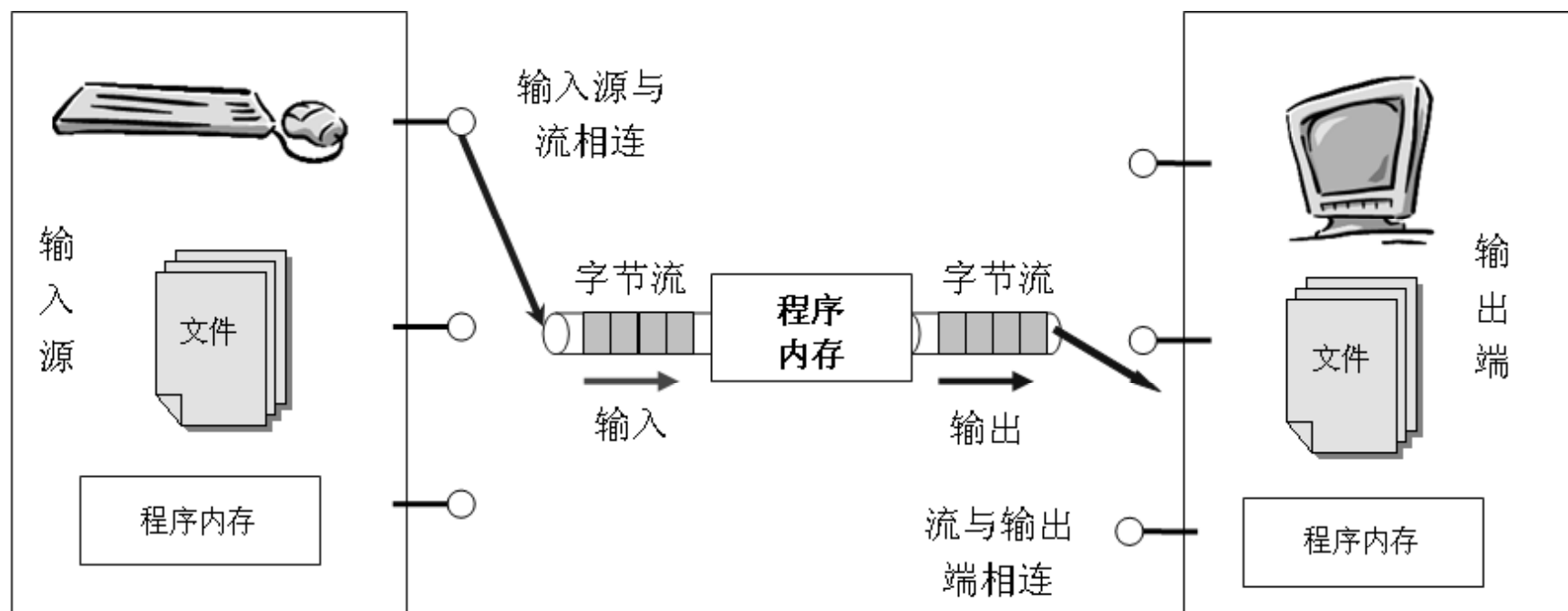
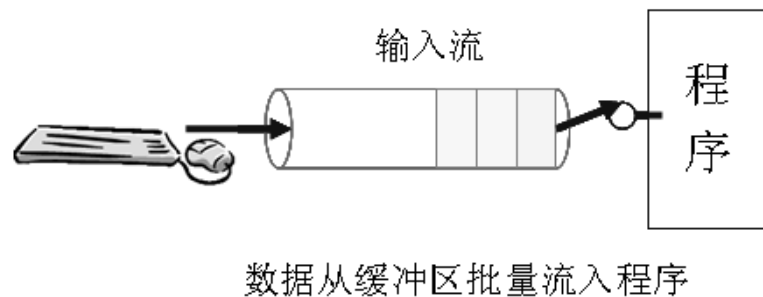
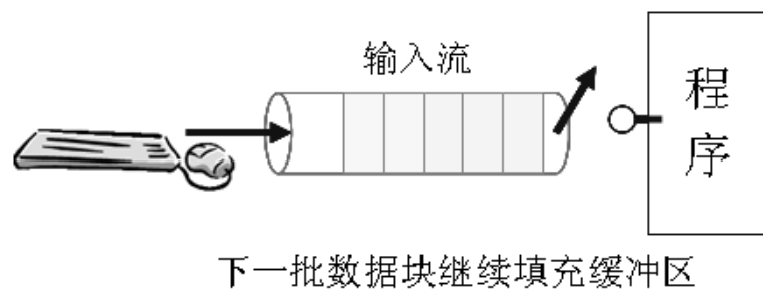
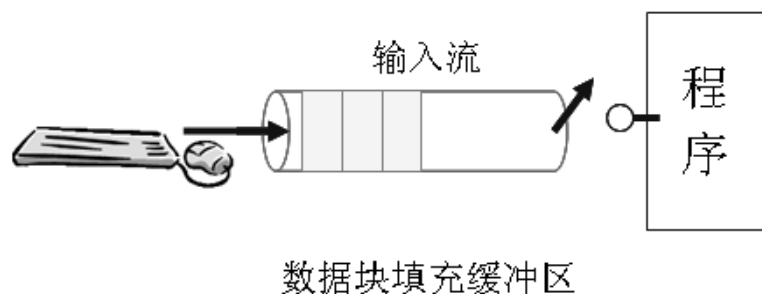


图9.1

数据在输入源、程序内存与输出端之间流动

9.1 理解流



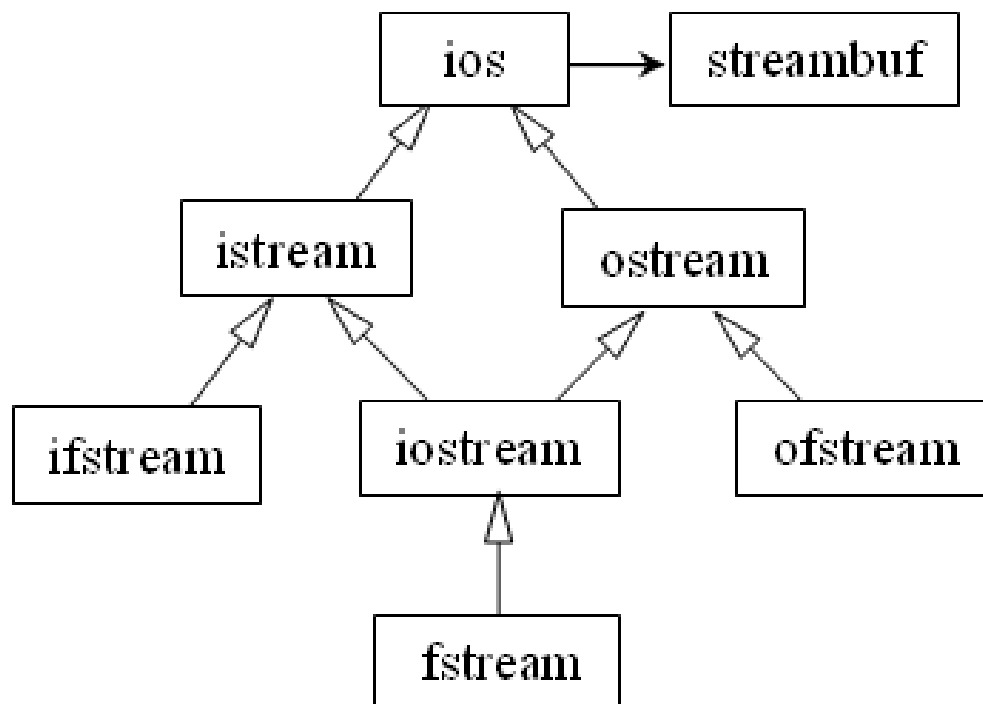
进行输入时标准流对象cin
负责建立输入通道
为数据开辟缓冲区

缓冲区(buffer)
一个临时存储区，
匹配不同设备数据传输率的差异

9.1 理解流

9.1.2 流类与缓冲区

C++类库包括庞大的**IO**类族，这些流类提供了丰富成员函数和运算符实现输入输出操作



9.1 理解流

9.1.2 流类与缓冲区

iostream类库中不同类的声明放在不同的头文件中

- **iostream.h**

包含操作所有输入/输出流所需的基本信息

istream.h , **ostream.h**

- **iomanip.h**

包含格式化I/O操纵算子，用于指定数据输入输出的格式

- **fstream.h**

处理文件信息，包括建立文件，读/写文件的各种操作接口

9.1 理解流

9.1.3 使用流的优点

C++程序可使用流对象**cin**与**cout**简洁安全地进行I/O操作。

➤ 自动识别类型

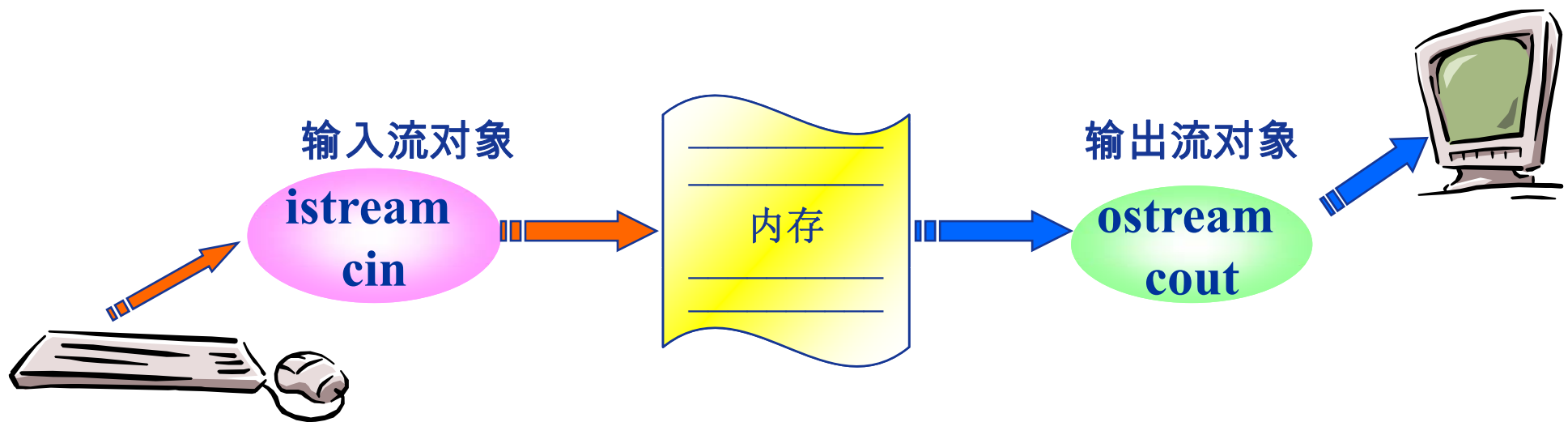
输出流类**ostream**中重载运算符“<<”，输入流类**istream**中重载运算符“>>”，使**cin**与**cout**能够识别对各种基本类型数据，自动按照指定的格式完成输入或者输出操作。

➤ 可扩展性好

对运算符“<<”和“>>”的重载，可以对各种用户自定义类型数据进行简洁安全的输入输出操作。

9.2 标准流对象

标准流对象是在**std**命名空间中定义的流对象，提供程序内存与常用外部设备进行数据交互功能。



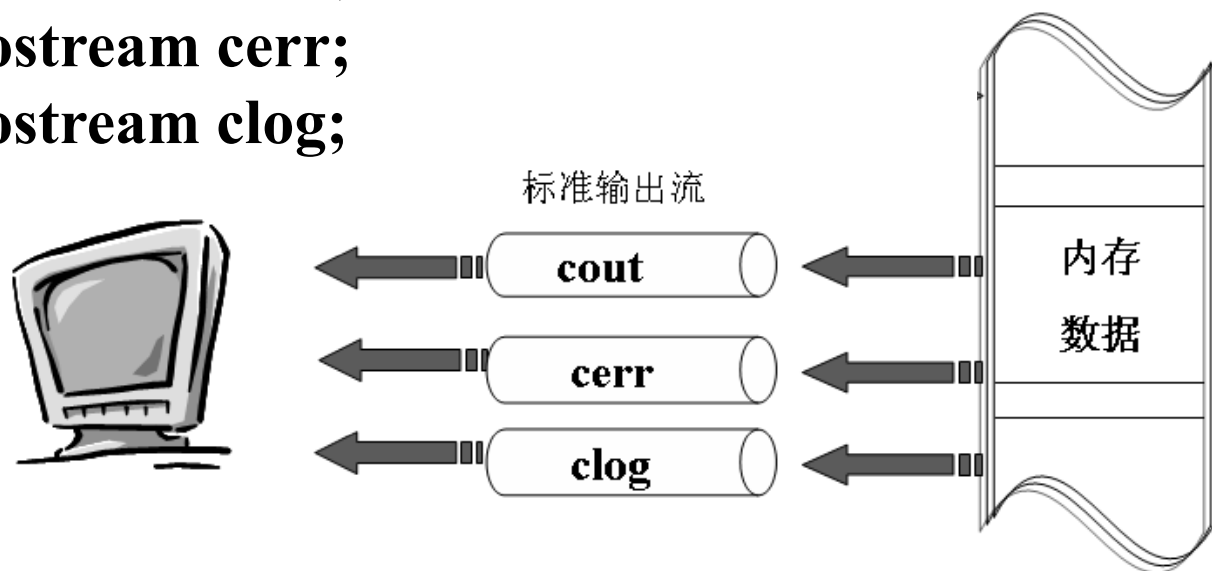
- I/O流类库预定义了一批流对象，连接常用的外部设备。
- 程序员可以定义所需的I/O流对象，使用流库提供的工作方式实现数据传输。

9.2 标准流对象

9.2.1 标准输出流对象

- 标准流输出对象是在std命名空间中定义的流对象，包括：

```
extern ostream cout;  
extern ostream cerr;  
extern ostream clog;
```



9.2 标准流对象

9.2.1 标准输出流对象

❖ `cout`

✓ `ostream` 类的对象（**console output**）

该对象将程序中的数据传送到标准输出设备上。

通常将数据在屏幕上显示出来。

使用 `<<` 顺序地将数据组装成字节序列，插入到流中输出。

`ostream& ostream::operator << (基本类型标识符);`

输出数据
的类型？



9.2 标准流对象

9.2.1 标准输出流对象

❖ cerr与clog

输出流对象clog(console log) cerr(console error)

作用相同：都能在标准输出设备上显示出错信息，
两个对象一般关联显示器进行输出。

区别：

cerr不经过缓冲区，直接向显示器上输出有关信息
clog中的信息存放在缓冲区中，缓冲区满后或接收
endl时输出。

9.2 标准流对象

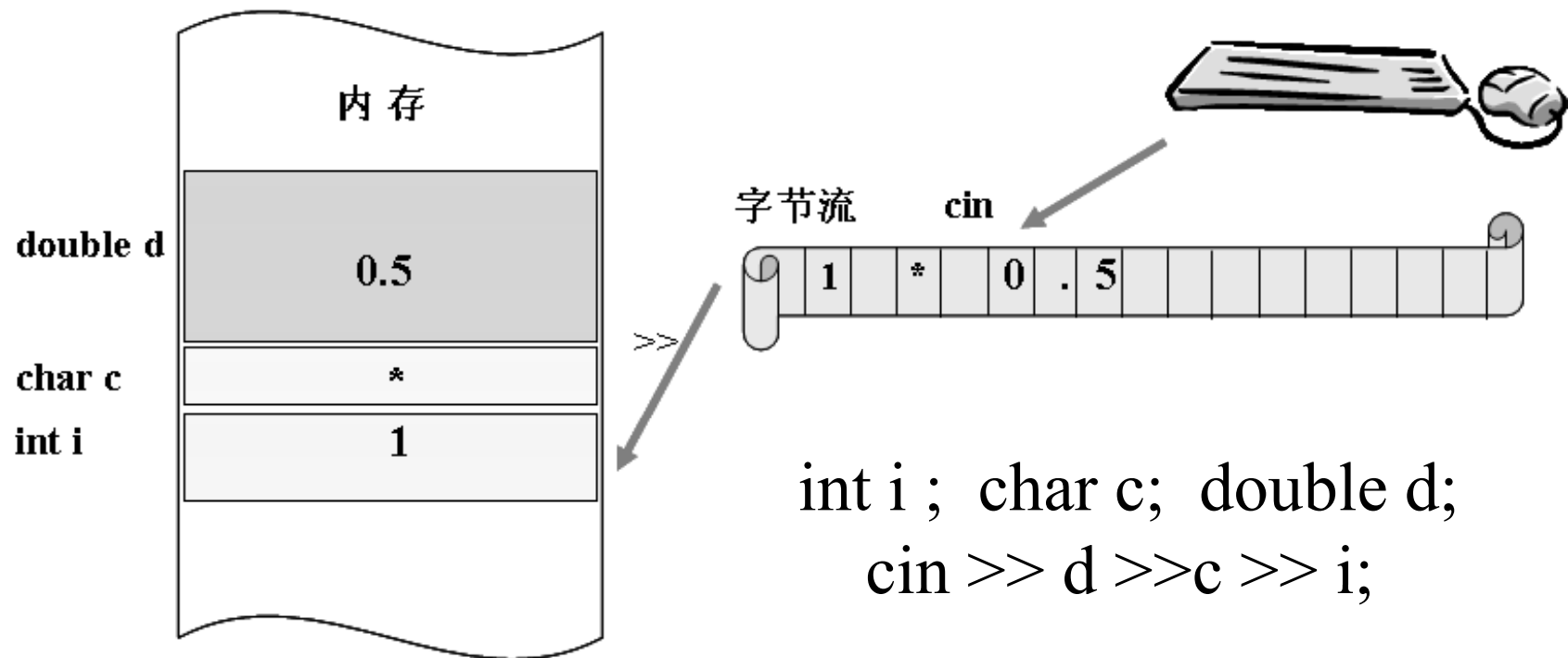
9.2.2 标准输入流对象

❖ cin

- ✓ istream类的对象，从标准输入设备(键盘)获取数据，
- ✓ cin使用流提取符“>>”将字节流中的数据进行解析后传输给内存。从字节流中抽取若干字节（遇到分隔符为止），按接收变量类型不同组装数据，即将字符型数据转换成二进制形式存储到变量中，使用这种流提取运算符输入称为格式化输入

9.2 标准流对象

9.2.2 标准输入流对象



9.2 标准流对象

9.2.2 标准输入流对象

❖ cin

在istream流类重载中>>的一组公用成员函数

istream& operator >> (基本类型标识符&) ;

- ❏ 流提取符从流中提取数据时通常跳过输入流中的空格、**tab**键、换行符等空白字符。
- ❏ 只有在输入完数据再按回车键后，该行数据才被送入键盘缓冲区，形成输入流，提取运算符才能从中提取数据。

9.3 流操作



9.3 流操作

9.3.1 格式化输入输出

控制符	作用
dec	转换整数的基数为十进制
oct	转换整数的基数为八进制
hex	转换整数的基数为十六进制
showbase	在输出中显示基数指示符
uppercase	十六进制输出时一律用大写字母
scientific	科学记数法显示浮点数
fixed	定点小数形式显示浮点数
showpoint	把带有小数点浮点数值输入到流中
showpos	正整数前加“+”号
unitbuf	输出操作后立即刷新流
left	输出数据在本域宽范围内左对齐
right	输出数据在本域宽范围内右对齐

9.3 流操作

9.3.1 格式化输入输出

在基类ios中定义了相关的控制符和成员函数
在头文件iomanip中还声明了一些普通函数

控制符	成员函数	作用
setfill(char c)	flag(char c)	设置填充字符为字符常量或字符变量c
setprecision(int n)	precision(int n)	设置显示小数的精度为n位
setw(int n)	width(int n)	设置域宽为n个字符
setiosflags()	setf()	设置输出格式的状态
resetiosflags()	unsetf()	清除已设置输出格式

9.3 流操作

9.3.2 字符数据的读写

ostream类的公有成员函数

函数	功能
put	无格式, 插入一个字节
ostream& ostream::put (char ch); 该函数的功能是把一个字符写入流中，在屏幕上显示该字符。该函数的参数可以是字符常量、字符变量或字符的 ASCII 代码，也可以是一个整型表达式。	

9.3 流操作

istream类的公有成员函数

函数	功能
read	无格式输入指定字节数
get	从流中提取字符，包括空格
getline	从流中提取一行字符

```
int get();
```

```
istream& get( char& rch );
```

```
istream& get( char* pch, int nCount, char delim = '\n' );
```

```
istream& getline( char* pch, int nCount, char delim = '\n' );
```

9.3 流操作

字符输入函数

❖ **int istream::get();**

功能: 从流中读取任意1个字符，返回值为读入的字符。
可用于从键盘读取字符，也可从文件中读取字符。
如果输入结束或达到文件末尾，该函数就返回字符EOF(End of File)

```
while((c=cin.get( ))!=EOF)  
    cout.put(c);    //输出字符
```

9.3 流操作

字符输入函数

❖ **`istream& istream::get(char &ch);`**

功能:带参数的get函数也能从流中读取一个字符, 所读取的字符存储在参数ch中。

如果读取字符不成功, 设置错误标志并在ch中存储EOF

```
while(cin.get(c)) cout.put(c);
```

```
while(cin >> c) cout<<c;
```



9.3 流操作

字符串输入函数

❖ **istream& istream::get (char* &pArray, streamsize n, char delim = '\n') ;**

功能:从输入流中读取n-1个字符(通常为long型), 赋给指定的字符数组pArray (或字符指针指向的数组), 如果在读取n-1个字符之前遇到指定的终止字符delim, 则提前结束读取。

```
cin.get (cArray, 5);  
cin.get (cArray, 5, '\n' );  
cin.get (cArray,10, '.' );
```


9.3 流操作

字符串输入函数

❖ **istream& istream::get (char* &pArray,
streamsize n, char delim = '\n') ;**

作用：读取一段文本，并将输入的字符串以及一个空字符一起存储，将指定的终止符留在流中。
如读取失败(遇EOF) 则流istream对象设置错误标志，不能继续进行输入操作。

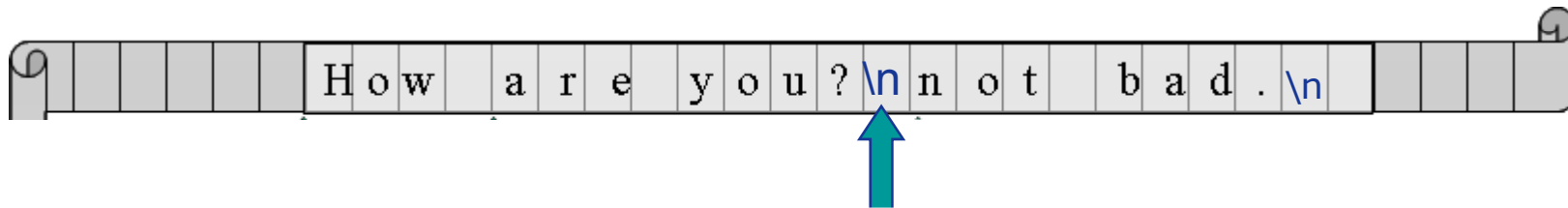
```
char str[20];  
cin >> str;  
cin.get (str, 20);
```



9.3 流操作

运行并分析输出结果

```
char ch1[20],ch2[20],ch3[20],ch4[20];  
cin>>ch1;                cout<<ch1<<endl;  
cin.get(ch2, 10);         cout<<ch2<<endl;  
cin.get(ch3, 20);         cout<<ch3<<endl;  
cin.get(ch4,20,'.');
```



9.3 流操作

字符串输入函数

❖ **`istream& istream::getline (char* &pArray, streamsize n, char delim = '\n') ;`**

作用：从输入流中读取n-1个字符，赋给指定的字符数组，如果在读取n-1个字符之前遇到指定的结束符(如换行符)则函数返回0值(假)。如果读取成功则函数返回非0值(真)。

`get()`

vs.

`getline()`

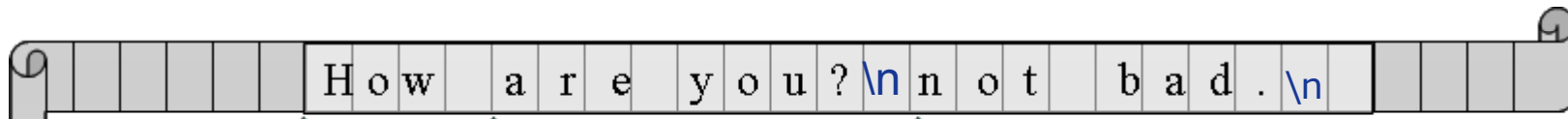


从流中提取字符串后流中删除分隔符，继续读取的字符串从分隔符后面的字符开始。

9.3 流操作

运行**ch9_5.cpp**并分析输出结果

```
char ch1[20],ch2[20],ch3[20],ch4[20];  
cin>>ch1;                cout<<ch1<<endl;  
cin.getline(ch2, 10);     cout<<ch2<<endl;  
cin.getline(ch3, 20);     cout<<ch3<<endl;  
cin.getline(ch4,20,'.');  cout<<ch4<<endl;
```



The diagram shows a character array `ch1` of size 20. The first 15 cells contain the string "How are you? \n not bad. \n". The 16th cell contains a null terminator '\0'. The remaining 4 cells are empty. The string is displayed in a scrollable window.

9.3 流操作

istream类的公有成员函数

函数	功能
read	无格式输入指定字节数
get	从流中提取字符，包括空格
getline	从流中提取一行字符
ignore	提取并丢弃流中指定字符
istream& ignore(int <i>nCount</i> = 1, int <i>delim</i> = EOF); 函数ignore()可以跳过输入流中的若干字符，或在遇到指定终止符时提前结束(跳出包括终止符在内的n个字符)。	
seekg	移动输入流指针
tellg	返回输入流中指定位置的指针值
operator>>	提取运算符

9.3 流操作

9.3.3 输入输出流的错误

为了保证程序能够安全进行输入输出操作，**C++**提供了检测**I/O**流状态的标志和成员函数。在流类中包含一个描述流状态的数据成员**state**，这个状态字中特定位记录流处于正常状态或不同的错误状态。

标识常量	值	意义																
ios:: goodbit	0x00	<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
ios:: eofbit	0x01	<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			
ios:: failbit	0x02	<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0			
ios:: badbit	0x04	<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0			

9.3 流操作

9.3.3 输入输出流的错误

流类提供一些成员函数用于测试或设置流对象的状态，常用函数

函数	功能
<code>int eof() const;</code>	返回eofbit状态值。文件结束符时返回1，否则返回0
<code>int fail() const;</code>	返回failbit状态值
<code>int bad() const;</code>	返回badbit状态
<code>int good() const;</code>	eofbit、failbit和badbit都没有被设置，则返回1
<code>int rdbuf() const;</code>	返回状态字
<code>void clear(int n= 0);</code>	恢复或设置状态字

9.3 流操作

9.3.4 重载流插入和提取运算符

➤ **C++**中定义的位左移运算符“<<”和位右移的运算符“>>”

进行位运算，在流类中对它们进行了重载，使之作为流插入运算符和流提取运算符，用来对各种基本类型的数据进行输出和输入操作。

9.3 流操作

9.3.4 重载流插入和提取运算符

➤ 重载流插入运算符<<

ostream流类中为基本类型数据重载“<<”，有如下形式：

```
ostream & ostream ::operator << (int);
```

```
ostream & ostream ::operator << (float);
```

```
ostream & ostream ::operator << (char);
```

```
ostream& ostream ::operator<< (const void *);
```

```
ostream& ostream ::operator<< (const char *);
```

可以在用户自定义的类A中重载该运算符

A类中定义如下友元函数：

```
friend ostream& operator << ( ostream& out, const A&);
```

9.3 流操作

9.3.4 重载流插入和提取运算符

➤ 重载流提取运算符>>

istream类中重载了一组成员函数**operator >>()**，对于各种基本类型**T**的数据，定义成员函数重载该运算符：

```
istream& istream ::operator>>( T& data);
```

```
istream& istream ::operator>>(char *);
```

可以在用户自定义的类**A**中重载>>运算符

A类中定义如下友元函数：

```
friend istream & operator >> ( istream &, Type&);
```

9.4 文件流

➤ 数据的层次结构

Bit – Byte – Field – Record – File - DBMS

➤ 文件的概念

存储在外部介质中的数据集合称为文件。

C++把文件看成有序的字节流，提供低级和高级的I/O功能

- 一批数据通常是以文件的形式存放在外部介质上的。
- 操作系统是以文件为单位对数据进行管理。

9.4 文件流

知识要点

- 文件以及文件流的概念

数据流动方向， 文本文件与二进制文件
文件流类与文件流对象

- 文件的打开与关闭

打开方式

- 文本文件的顺序读写操作

<<, >>, put, get(), getline(), read(), write()

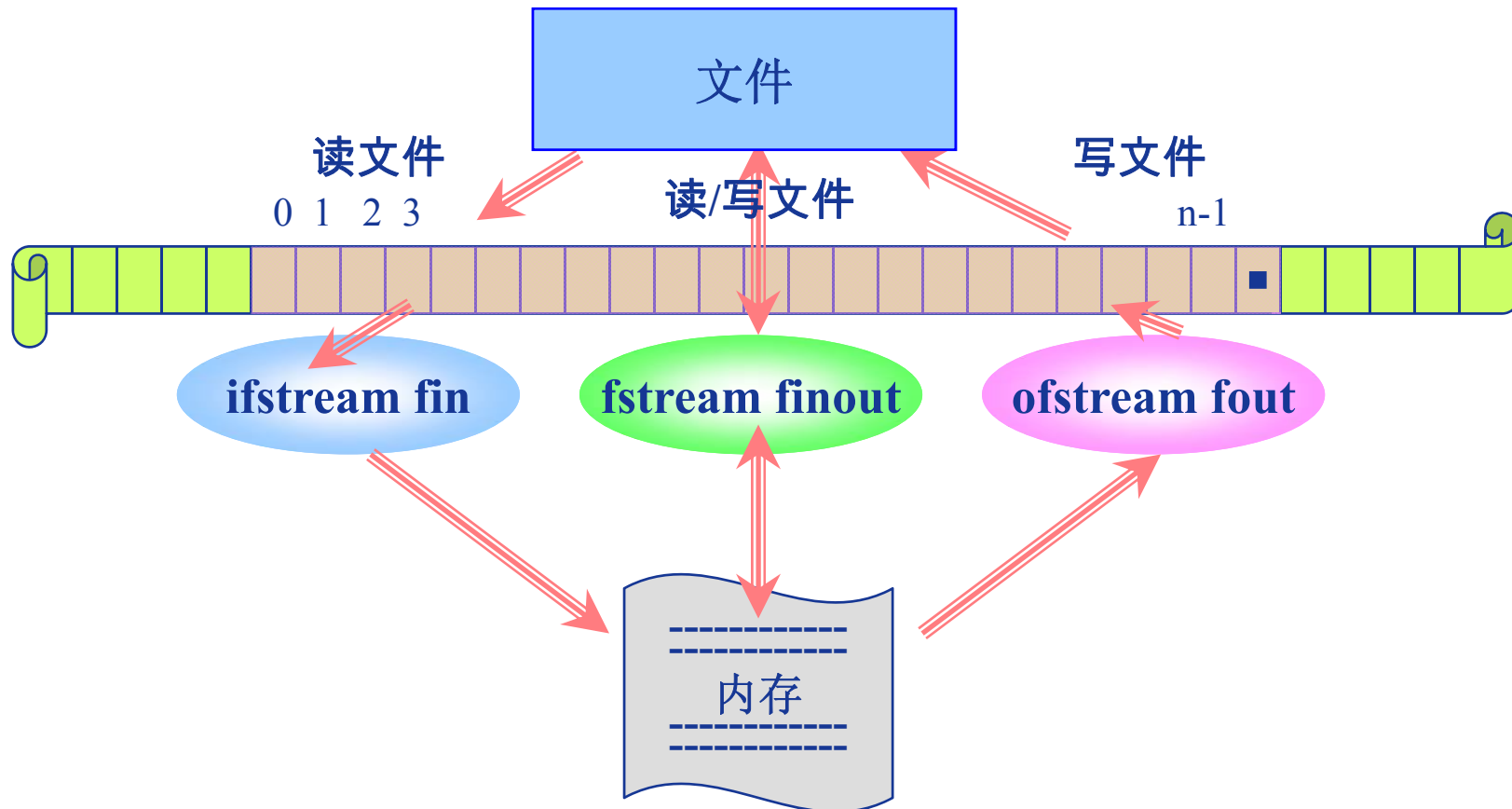
- 二进制文件的随机读写操作

9.4 文件流

9.4.1 文件流类与对象

➤ 文件流的概念

文件流是以外存文件为输入输出对象的数据流



9.4 文件流

9.4.1 文件流类与对象

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字节序列，
输入文件流是从外存文件流向内存的字节序列。
每一个文件流都有一个内存缓冲区与之对应。

I/O类库中定义了以下三种文件流进行文件读写操作。

1. **ifstream**类

从输入流**istream**类派生的，支持从磁盘文件的输入。

2. **ofstream**类

从输出流**ostream**类派生的，支持向磁盘文件的输出。

3. **fstream**类

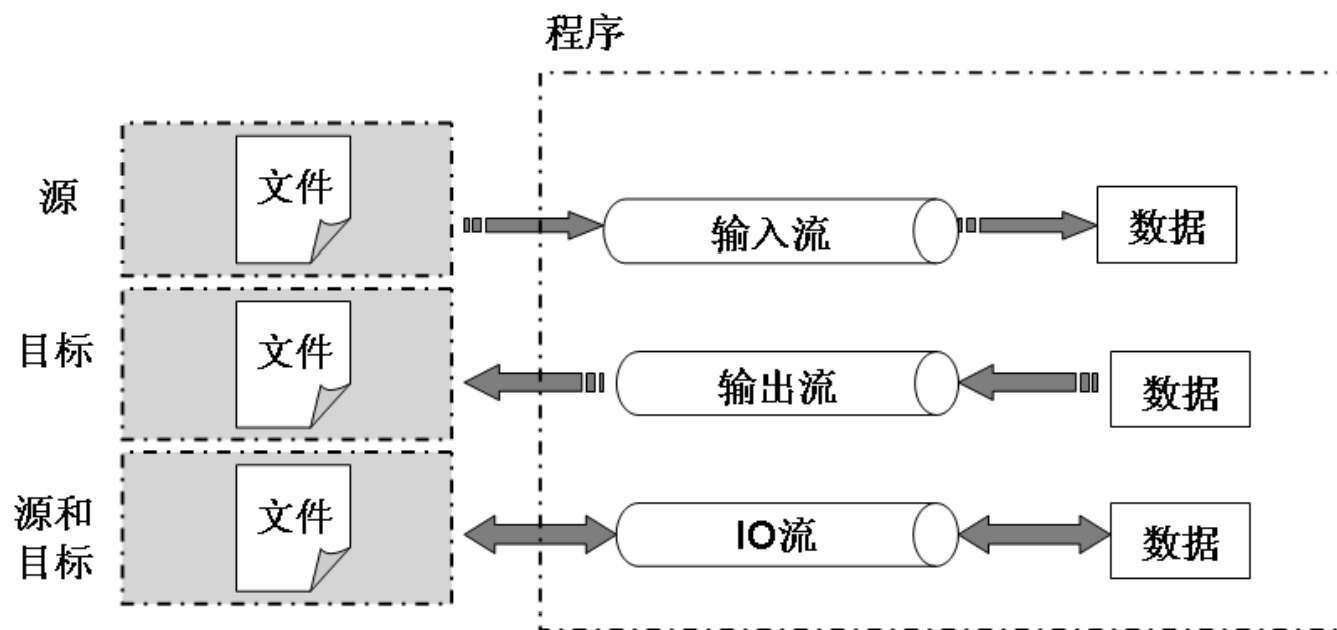
从输入输出流**iostream**类派生的，支持对磁盘文件的输入输出。

9.4 文件流

9.4.1 文件流类与对象

➤ **ifstream**、**ofstream** 和 **fstream** 类对象

用于内存与文件之间的数据传输



9.4 文件流

9.4.1 文件流类与对象

C++把文件看成有序的字节流，提供低级和高级的IO功能

- 使用方式： 程序文件 / 数据文件
- 存取方式： 顺序读写文件 / 随机读写文件
- 编码方式： 文本文件 / 二进制文件

➤ 文本模式

- 任何数据被解释为一系列的字符，
内存中存储的二进制数据需要转换为字符的形式。

➤ 二进制模式

- 无需进行格式的转换，流中直接传送原始字节数据。



9.4 文件流

➤ 文件操作的基本步骤：

- 打开文件

```
streamclass fileObj ( fileName , openMode ) ;  
fileObj.open( fileName , openMode) ;
```

- 读 / 写文件

```
<< , put(),          write()  
>> , get(),getline(), read()
```

- 关闭文件

```
fileObj.close( ) ;
```

9.4 文件流

9.4.2 文件的打开和关闭

➤ 打开文件

建立文件流对象；

流对象与磁盘文件关联；

指定文件的工作方式以及开辟数据缓冲区。



构造函数
`open()`

➤ 关闭文件

解除流对象与磁盘文件关联

释放相关的系统资源



析构函数
`close()`

9.4 文件流

➤ 打开文件-方法1

先建立流对象，调用 `fstream::open()` 连接外部文件

流类 ~~对象名~~ ；

`ifstream`、`ofstream` 或 `fstream`

对象名 . `open` (文件名 , 方式) ；

文件名是识别文件的标识符，包含路径、文件名称以及文件后缀

```
ofstream outFile ; //流对象打开文件后准备写入 数据  
outFile.open ( “C:\\Ch9\\filename.txt” ) ;  
outFile.open ( “filename.txt” ) ;
```

9.4 文件流

➤ 打开文件-方法2

调用流类带参数的构造函数，
建立流对象时连接外部文件

流类 对象名 (文件名 , 方式);

eg

```
ifstream infile ( "datafile.dat" , ios::in );  
ofstream outfile ( "d:\\newfile.dat" , ios::out );
```

9.4 文件流

基类**ios**中定义的相关标志位，用于设置文件的操作方式

标识常量	意义	默认为文本方式
<code>ios::in</code>	输入方式打开文件，进行读文件操作	
<code>ios::out</code>	输出方式打开文件，进行写文件操作	
<code>ios::ate</code>	打开文件时，指针指向文件尾	
<code>ios::app</code>	输出方式打开文件，追加方式写入	
<code>ios::trunc</code>	删除文件现有内容，把已有文件长度变为0	
<code>ios::nocreate</code>	如果文件不存在，则打开操作失败	
<code>ios::noreplace</code>	如果文件存在，则打开操作失败	
<code>ios::binary</code>	二进制读写方式，数据以字符的形式直接在内存与文件之间传送	

文件流的输入输出方式

9.4 文件流

打开一个已有文件准备读入数据的操作

```
ifstream infile;
```

```
infile.open( "datafile.dat" , ios::in );
```

准备向文件写数据的操作

```
ofstream outfile( "D:\\newfile.dat" , ios::out );
```

读写方式打开文件

```
fstream rfile ( "myfile.dat" , ios::in | ios::out );
```

读写二进制文件

```
fstream ioFile;
```

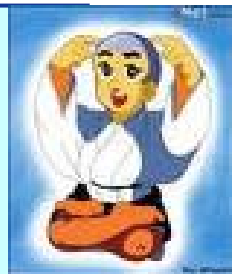
```
ioFile.open( file2, ios::in|ios::out|ios::binary);
```

```
ofstream ofile ( "a:\\binary" , ios :: binary | ios
```

若不存在该文件则在
D盘下自动创建文
若文件已存在则清除
原有内容后输出

用位运算符“|”对
输入输出方式进行组

文本文件
VS
二进制文件



9.4 文件流

2. 关闭文件

- 关闭文件操作包括把缓冲区数据完整地写入文件，
添加文件结束标志，
切断流对象和外部文件的连接
- 当一个流对象的生存期结束，系统也会自动关闭文件；
- 若流对象的生存期没有结束，用close()关闭文件后，
该流对象可以重用。

9.4 文件流

2. 关闭文件

使用同一对象可以读写多个文件，但要注意：
在某一时刻一个文件最多和一个对象关联，
也不能用多个对象读写同一个文件。
如果文件对象与不同的文件关联，
必须先关闭前一个文件，再打开新的文件

```
ofstream ofile ;
```

// 创建对象

等价于使用构造函数:

```
ofstream ofile ( "myfile1" );
```

```
ofile . open ( "myfile1" );
```

// ofile流与文件“myfile1”相关联

```
.....
```

// 访问文件“myfile1”

```
ofile . close ( ) ;
```

// 关闭文件“myfile1”

```
ofile . open ( "myfile2" );
```

// 重新打开

close () 函数关闭文件
但流对象仍然存在

9.4 文件流

3. 文件的错误检测

测试流状态检测文件是否成功打开

调用**ios**流类成员函数**good()**、**bad()**与**fail()**

在条件表达式中使用取反操作符（！）测试流状态

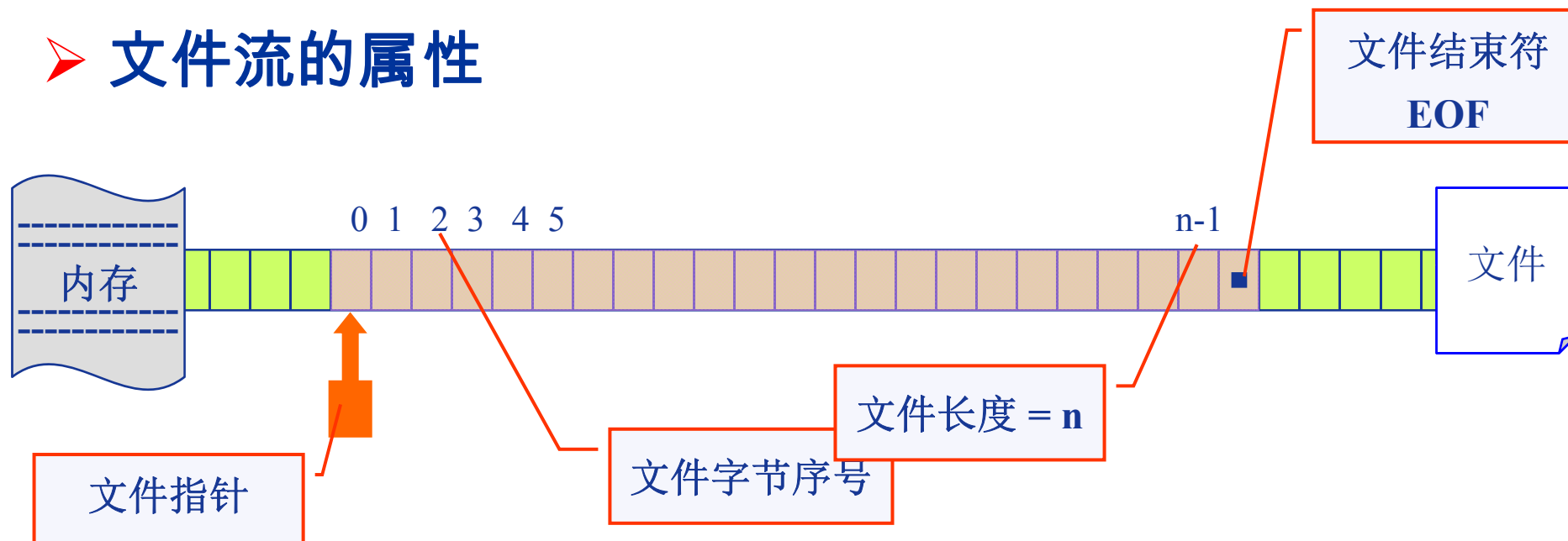
```
if( !outfile) // outfile.open("myfile2.dat")
{   cerr<<" error: unable to open file2! "; }
```

测试成功关闭文件，可以调用**fail()**函数

```
myfile2.close();
if(myfile2.fail())
    cerr<<"Error to close myfile2! ;"
```

9.4 文件流

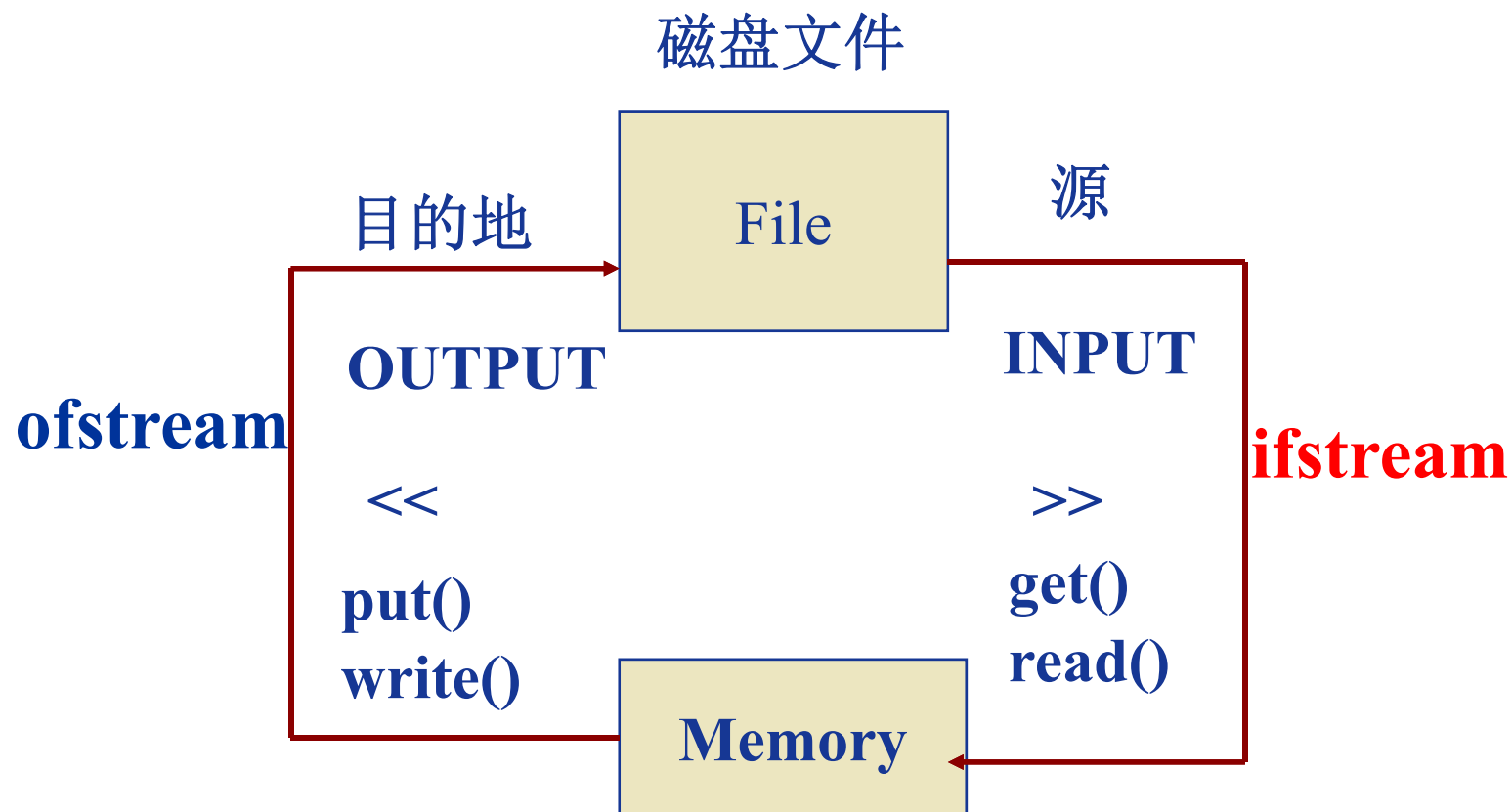
文件流的属性



每次读写都是从文件指针的当前位置开始，文件指针的初始位置由打开方式指定，打开文件时文件指针默认指向开始位置。随着数据的读写指针就向后移动。

9.4 文件流

使用流的磁盘文件I/O



9.4 文件流

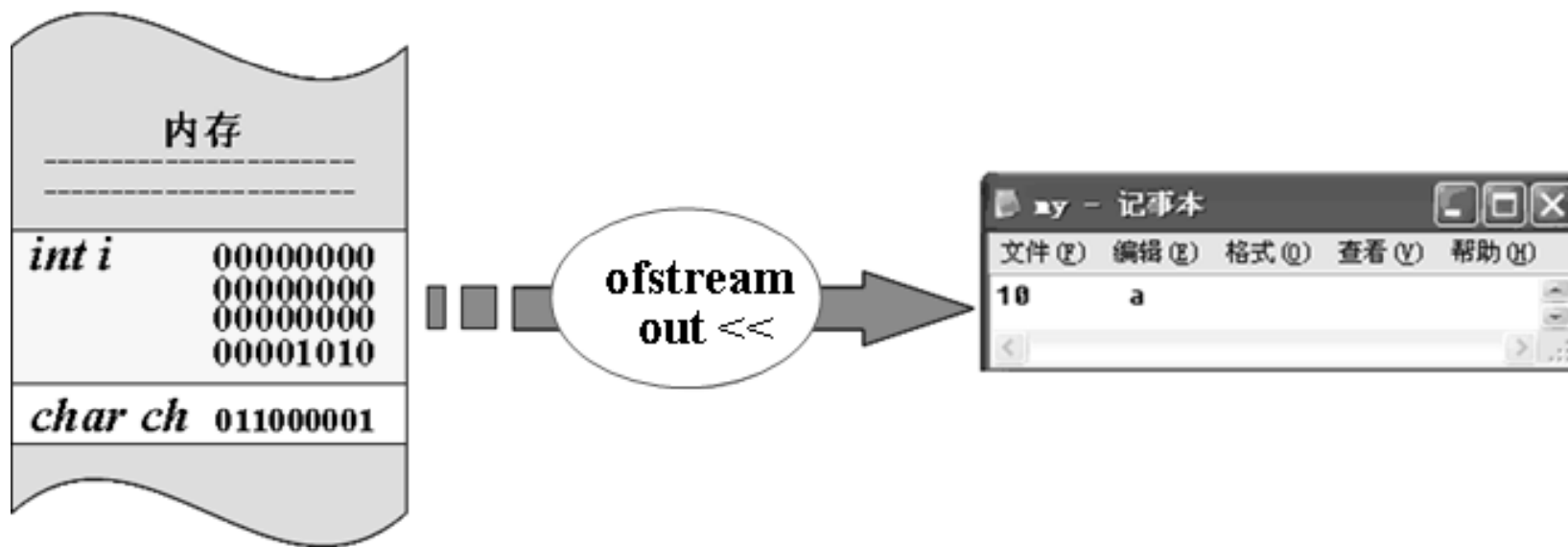
9.4.3 文本文件的读写

- 文件的每个字节都是用ASCII代码存放数据，
即每个字节存放一个字符，这种文件又称字符文件
- 文本文件是顺序存取文件
- 文件的默认打开方式为文本文件，格式化的文件读写操作。
使用流插入运算符“<<”向文件写入若干个字符，
使用流提取运算符“>>”从文件中读取若干个字节的数据，
调用文件流成员函数进行输入输出。

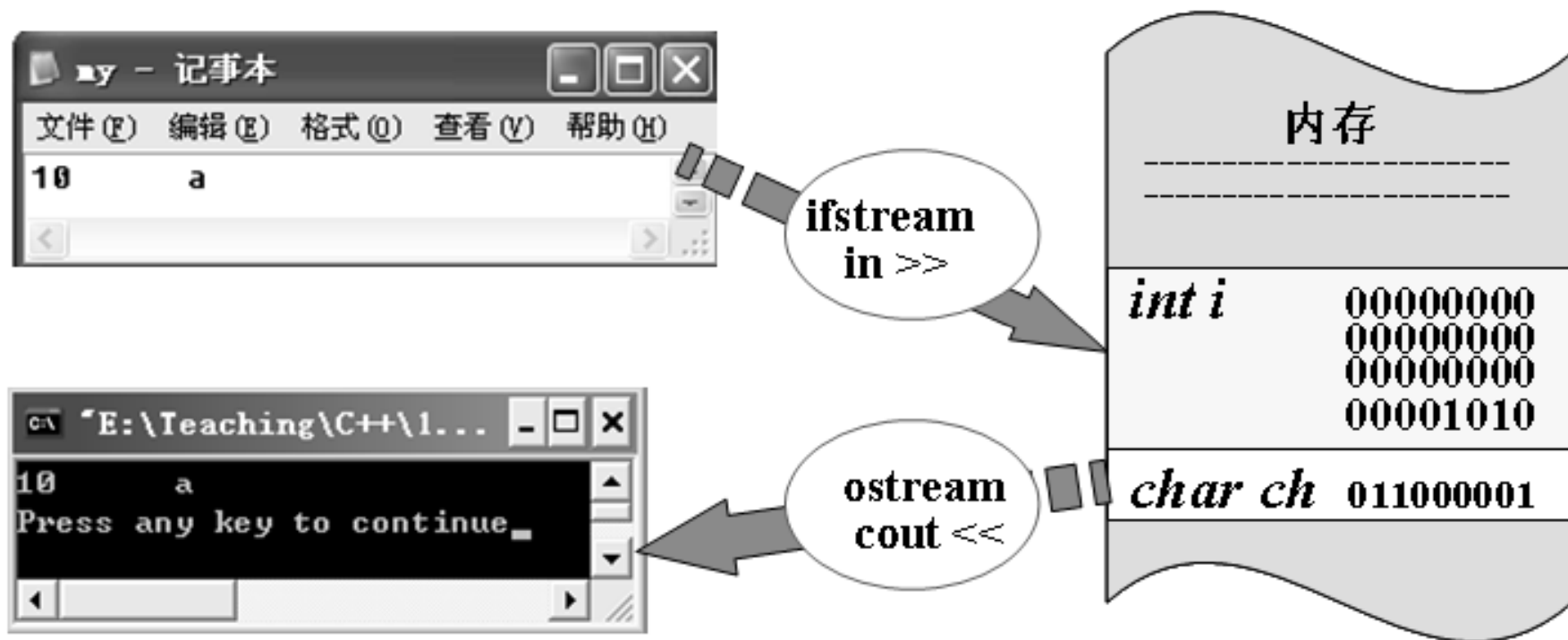
9.4 文件流

9.4.3 文本文件的读写

```
int i = 10;  
char ch = 'a';  
ofstream out ( "c:\\my.txt ");  
out << i << '\t' << ch << '\n';  
out . close ( );
```



9.4 文件流



```
ifstream in ( "c:\\my.dat" );  
in >> i >> ch;  
cout << i << '\t' << ch << endl;
```

9.4 文件流

例9.8 用筛选法产生25个素数，将其写入C盘文件primes.txt中。

例9.9 读取所有文件primes.txt 中的所有素数

➤ 使用<<, >>完成文件读写

向文本文件中写入数据，并从文件中读出数据

```
//格式化将素数写入文件
for(int i = 0; i < max; i++) {
    if(i % 5 == 0) outFile << endl;
    outFile << setw(10) << *(primes + i); }
```

由于文本文件本身没有数据的逻辑结构，为了保证顺利从文件中读取数据，通常数据项之间用空白符、换行符、制表符等分隔

9.4 文件流

例9.10 使用getline从文件中读取字符串

```
char filename[10] = "D://test",str[20];  
ofstream outfile(filename,ios::out|ios::app);  
outfile << filename <<endl;  
cout<<"input a string:";  
cin.getline(str,20);           // 键盘输入字符串  
outfile << str <<endl;         // 向文件写入字符串  
outfile.close();
```


9.4 文件流

例9.10 使用getline从文件中读取字符串

```
const int MAX = 50;  
char buffer[MAX];  
ifstream infile;  
infile.open(filename);  
while( !infile.eof() ) {  
    infile.getline(buffer, MAX);  
    cout << buffer << " ";  
}  
infile.close( )
```

运行结果
input a string:Tian linlin
D://test !
Tian linlin !
!



while(infile.getline(buffer, MAX);)
 cout << buffer << ' !'<< endl;

9.4 文件流

9.4.4 文件的随机读写

- 二进制文件以基本类型数据在内存中以二进制表示形式存放数据，不对写入或读出的数据做格式转换 ；
- 打开二进制文件用`ios::binary`方式 ；
可以即为输出又为输入文件。
- 二进制文件的读写方式由程序控制 ，
常用随机读写操作

9.4 文件流

9.4.4 文件的随机读写

二进制模式便于进行随机的文件读写操作，通过移动文件指针将数据写入文件的指定位置，或者任意读取文件中的某个数据。

二进制文件一般执行非格式化的读取操作：

`get()`、`getline()`与`put()`

`read()`与`write()`读写文件

可读取任何类型的数据，适合高效便利的传输数据块。

9.4 文件流

文件无格式读写函数

➤ istream 类中操作字节数据的成员函数

```
istream & istream :: read ( char * buf, int n );
```

从流中提取 *n* 个字节数据，更新对象 *buf*

➤ ostream 类中操作字节数据的成员函数

```
ostream & ostream :: write ( char * buf, int n );
```

向流插入 *buf* 对象的由第二个参数指定数目的字节数据

9.4 文件流

文件无格式读写函数

➤ istream 类中操作字节数据的成员函数

```
istream & istream :: read ( char * buf, int n );
```

从流中提取 *n* 个字节数据，更新对象 *buf*

```
input.read(reinterpret_cast< char*>(iAry), 3*sizeof(int));
```

```
ifstream input("file1.dat",ios::binary|ios::in);  
input.read((char*) (&iNum), sizeof(int)) ;  
input.read((char*) iAry, 3*sizeof(int)) ;  
iNum = input.gcount( )/sizeof(int);  
for(int i = 0; i< iNum;i++)  
    cout << iAry[i] <<ends;
```

9.4 文件流

文件无格式读写函数

➤ ostream 类中写入字节数据的成员函数

ostream & ostream :: write (char * *buf*, int *n*);

向流插入 *buf* 对象的由第二个参数指定数目的字节数据

```
ofstream outf( "test.dat" );
```

```
char str[10] = "Hello!";
```

```
outf.write( str , 10 );
```

```
int a[10] = {1,2,3,4,5};
```

```
for(int i = 0; i < 10 ; i ++ )
```

```
    outf.write( (char *) (&a[i]), sizeof(int) );
```

注意类型转换

```
outf.write((reinterpret_cast< char*> (a) , 10*sizeof(int));
```

9.4 文件流

有关学生数据的结构**Stu**，读入一条学生记录：

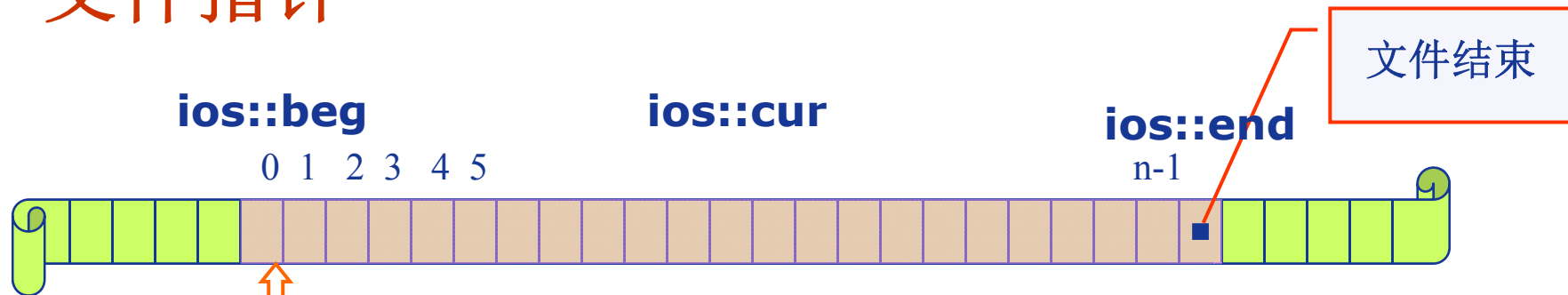
```
Bool readStudent(Stu& student, ifstream& fStudent)
{
    fStudent.read((char*)&student, sizeof(Stu));
    bool ioFlag = fStudent.good( ); //是否成功读取
    return ioFlag;
}
```

➤ 向文件写入一条结构记录**Stu aStudent**:

```
Ofstream out("stu.dat",ios::binary | ios::out | ios::app) ;
out.write((char*)&aStudent, sizeof(Stu));
//将一条记录写入文件末尾
```

9.4 文件流

文件指针



```
enum ios::seek_dir { beg = 0 ; cur = 1 , end = 2 } ;
```

`ios::seek_dir` 值:

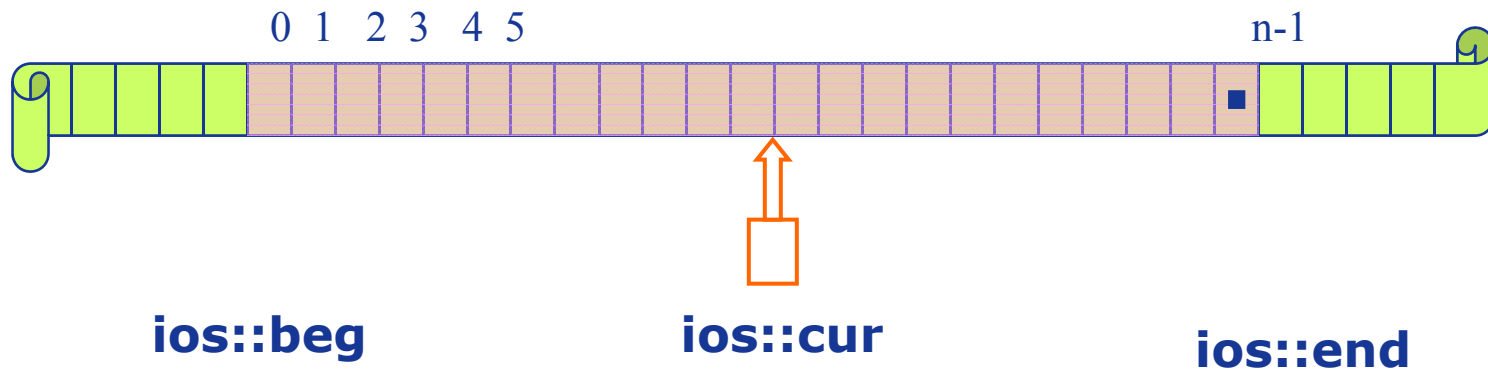
`cur` 相对于当前读指针所指定的当前位置

`beg` 相对于流的开始位置

`end` 相对于流的结尾处

9.4 文件流

文件指针



流类提供一些有控制流指针位置的成员函数，
可以获取和改变当前文件指针的位置，
以便在文件指定位置进行读写操作。

9.4 文件流

9.4.4 文件的随机读写

istream 类操作流读指针的成员函数

➤ `istream & istream :: seekg (long pos) ;`

读指针从流的起始位置向后移动由`pos`指定字节

➤ `istream & istream :: seekg (long off, ios::seek_dir) ;`

读指针从流的`seek_dir`位置移动 `off` 指定字节

➤ `istream & istream :: tellg () ;`

返回读指针当前所指位置值

9.4 文件流

9.4.4 文件的随机读写

istream 类操作流读指针的成员函数

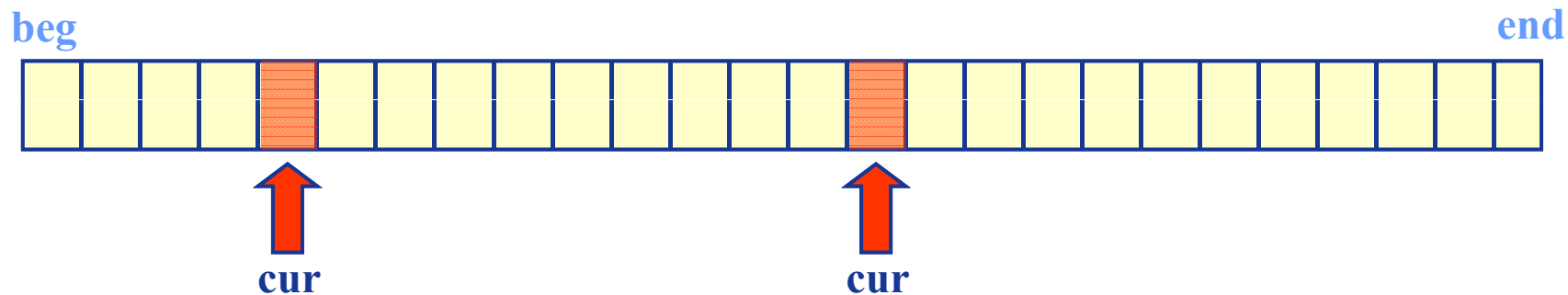
Eg:

```
istream input ;
```

.....

```
input . seekg ( - 10 , ios :: cur ) ;
```

// 读指针以当前位置为基准，向前移动 10 个字节



9.4 文件流

9.4.4 文件的随机读写

istream 类操作流读指针的成员函数

Eg:

```
istream input ;
```

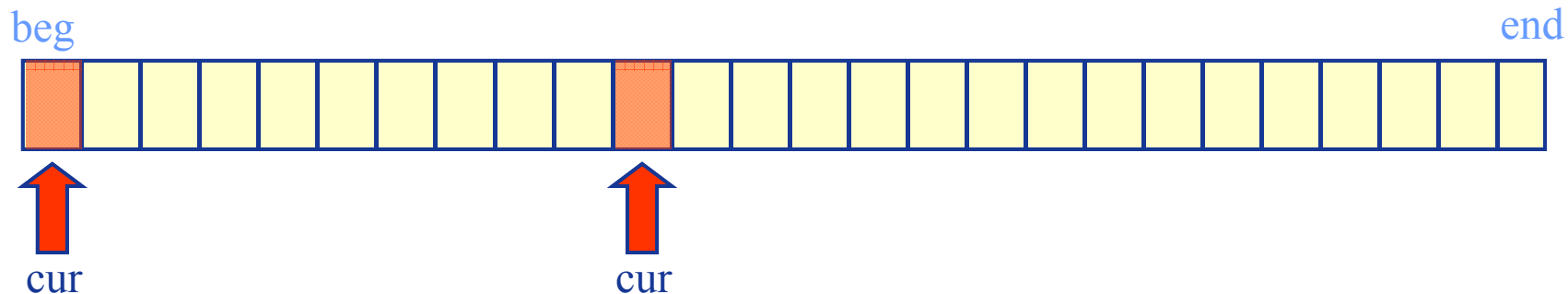
```
.....
```

```
input . seekg ( 10 , ios :: beg ) ;
```

函数 `seekg (n) ;`

等价 `seekg (n , ios::beg) ;`

// 读指针从流的开始位置, 向后移动 10 个字节



9.4 文件流

9.4.4 文件的随机读写

istream 类操作流读指针的成员函数

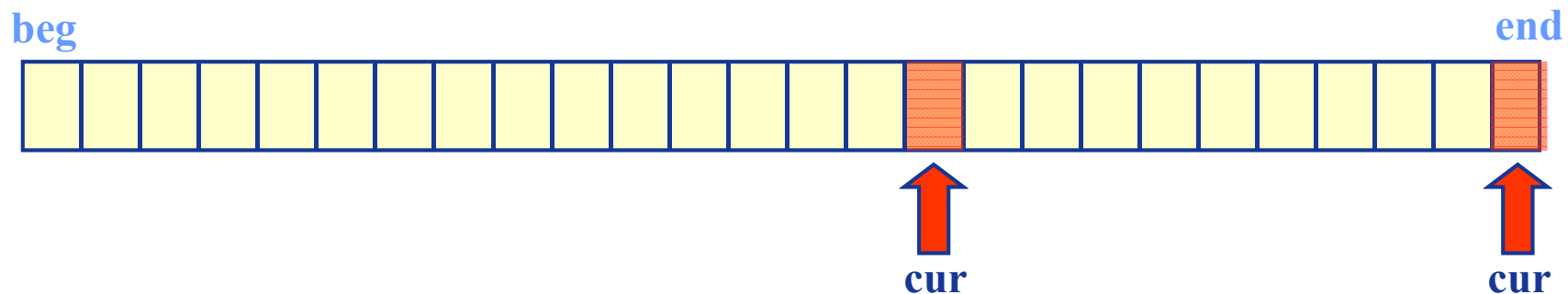
Eg:

```
istream input ;
```

.....

```
input . seekg ( -10 , ios :: end ) ;
```

// 读指针从流的结尾，向前移动 10 个字节



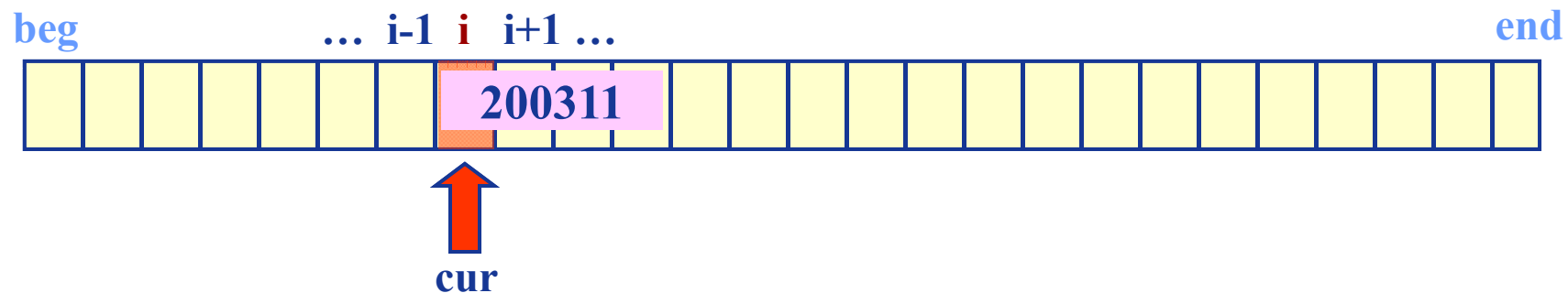
9.4 文件流

9.4.4 文件的随机读写

istream 类操作流读指针的成员函数

Eg:

```
istream input ;  
long pos = input . tellg ( ) ;    // 获取当前位置指针  
input >> number ;    // 读入一个整数，指针后移 4 字节  
.....  
input . seekg ( pos ) ;    // 指针返回原来位置  
input >> number ;    // 重读该整数
```



9.4 文件流

9.4.4 文件的随机读写

ostream 类操作流写指针的成员函数

➤ `ostream & ostream::seekp (long pos);`

写指针从流的起始位置向后移动由参数指定字节

➤ `ostream & ostream::seekp (long off, ios::seek_dir);`

写指针从流的`seek_dir`位置移动由 `off` 指定字节

➤ `ostream & ostream::tellp ();`

返回写指针当前所指位置值

9.4 文件流

例9.11 随机文件读写

```
fstream f( "DATA.dat" , ios::in | ios::out | ios::binary );  
int i;  
for( i = 0; i < 20; i ++ )    //先写入 20 个整数  
    f.write((char *)&i, sizeof(int) );  
long pos = f.tellp( );      //记录当前写指针位置值  
for( i = 20; i<40; i ++ )    //再写入 20 个整数  
    f.write( (char*)&i, sizeof(int) );
```


9.4 文件流

例9.11 随机文件读写

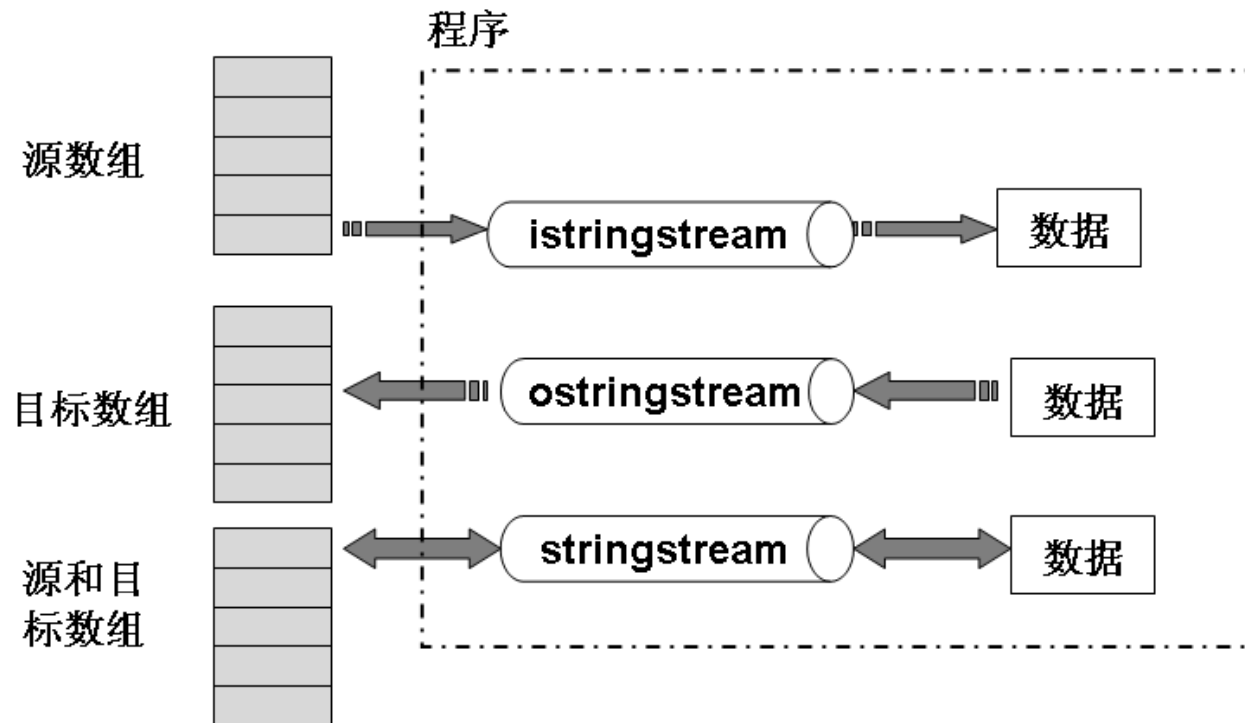
```
f.seekg(pos);      //将指针移到 pos 所表示的位置
f.read((char*)&i, sizeof(int) );  //读出一个数据
cout << "The data stored is " << i << endl
    << "file pointor pass: " << f.tellp()-pos<<endl;
//读文件后指针移动字节数
f.seekp( 0, ios::beg ); //指针移到文件开始
for( i = 0; i<40; i ++ )    //显示全部数据
{ f.read( (char*)&i, sizeof(int) );
  cout << i << ends;
}
```

去掉如何?

9.4 文件流

9.5 字符串流

串流类是 ios 的派生类:istringstream、ostringstream和stringstream。
串流对象一般关联string对象或字符数组，
可以把字符串中的数据通过流传送到在一组变量中，
也可以把一组数据转换为字符串写入string对象中。



9.4 文件流

9.5 字符串流

这些类的操作与文件流相同，是针对string对象I/O操作。

字符串流常与插入和提取运算符一起使用，主要应用是内存中格式化数据或分析输出。

串流I/O具有格式化功能：
串流提取数据时对字符串按变量类型解释；
插入数据时把类型数据转换成字符串。

例9.12 从字符串流中提取数据

例9.13 向字符串流中插入数据 。

小结

- 流对象是内存与文件（或字符串）之间数据传输的信道。
- 数据流本身没有逻辑格式。

数据的解释方式由应用程序的操作决定。

流类库提供了格式化和非格式化的I/O功能。

- 文本流I/O提供内存基本类型数据与文本之间的格式转换。
- 处理用户定义的文件I/O要用文件流对象。

小结

- 根据代码方式分为文本文件和二进制文件，
根据数据存取方式分为顺序存取文件和随机存取文件。
- 文件操作的三个主要步骤是：
打开文件；读/写文件；关闭文件流。
- 文件的性质由打开文件的方式决定。
移动流指针，可以对文件的任意位置进行读/写操作。