C++程序设计

主讲：王老师

尚德机构

学习是一种信仰

考试题型

单选题　　1分×20题 = 20分

填空题　　1分×15题 = 15分

程序填空题　4分×5题 = 20分

程序分析题　6分×5题 = 30分

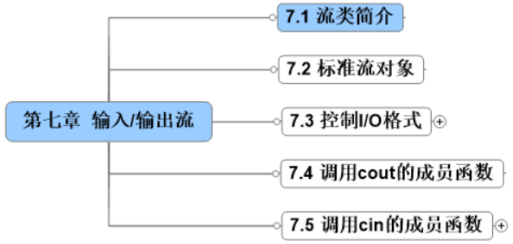
程序设计题　　2题 = 15分

（第一题5分，第二题10分）

第七章 输入/输出流

本章主要内容

- 流类简介
- 标准流对象
- 控制I/O格式
- 调用cout的成员函数
- 调用cin的成员函数



7.1 流类简介

- C++中凡是数据从一个地方传输到另一个地方的操作都是**流**的操作。因此，一般意义下的读操作在流数据抽象中被称为（从流中）“提取”，写操作被称为（向流中）“插入”。
- 在C++中，输入输出的完成是通过**流**。

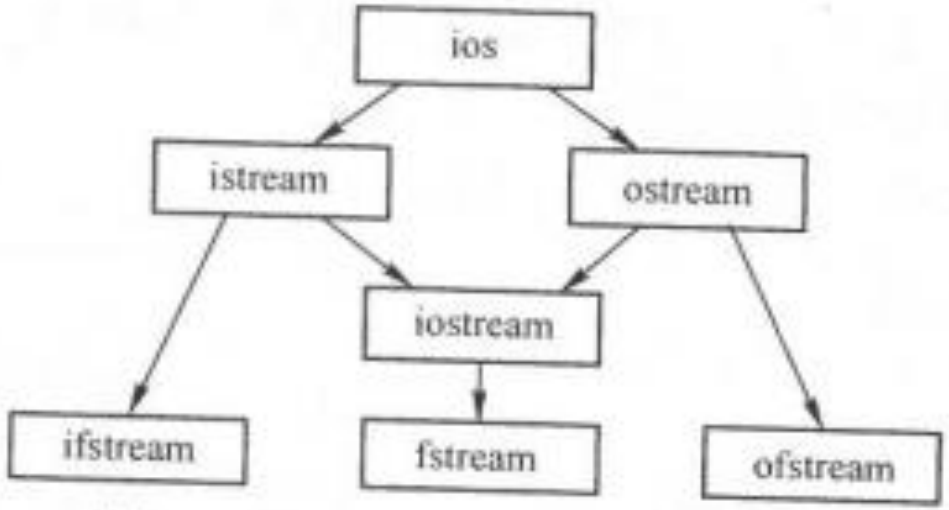


图 7-1 iostream 流类库的类关系图

7.1 流类简介

图7-1中的箭头代表派生关系。ios是抽象基类，提供输入/输出所需的公共操作，它派生出两个类istream和ostream。为了避免多重继承的二义性，从ios派生istream和ostream时，均使用了virtual关键字（虚继承）。

istream类提供了流的大部分输入操作，对系统预定义的所有输入流重载提取运算符“>>”。ostream类对系统预定义的所有输出流重载插入运算符“<<”。

由istream和ostream又共同派生了iostream类。

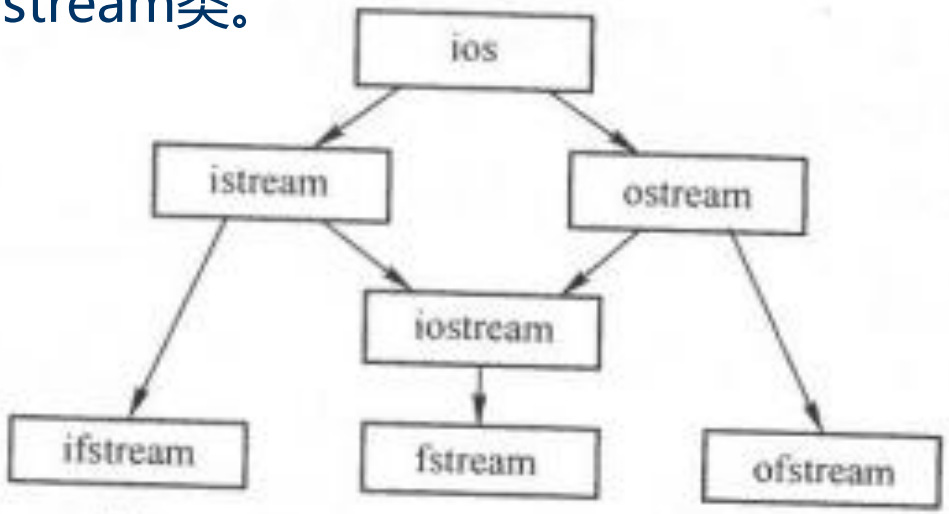


图 7-1 iostream 流类库的类关系图

7.1 流类简介

C++的iostream类库提供了数百种I/O功能，iostream类库的接口部分包含在几个头文件中。常见的头文件有以下3个：

(1)**iostream**

头文件**iostream**包含操作所有输入/输出流所需的基本信息，因此大多数C++程序都应包含这个头文件。该文件含有4个标准流对象，提供了无格式化和格式化的I/O功能。

(2)**iomanip**　　**例如：setw()，setprecision()，setfill()，setbase()等。**

头文件iomanip包含格式化I/O的带参数流操纵符，可用于指定数据输入/输出的格式。

(3)**fstream**

头文件**fstream**包含处理**文件**的有关信息，提供建立文件、读/写文件的各种操作接口。

课堂练习

下列流类中，可以用于输入/输出的是（　）。

A:ifstream

B:iostream

C:istream

D:ofstream

课堂练习

下列流类中，可以用于输入/输出的是（　）。

A:ifstream

B:iostream

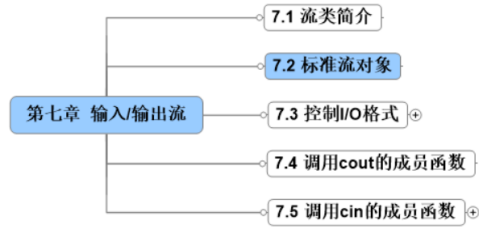
C:istream

D:ofstream

答案：B

类名	说明
ifstream	文件输入流类
iostream	通用输入/输出流基类和其他输出流基类
istream	通用输入流基类和其他输入流基类
ofstream	文件输出流类

7.2 标准流对象



C++在头文件iostream中为用户预定义了4个标准流对象，分别是：

- cin(标准输入流)
- cout(标准输出流)
- cerr(非缓冲错误输出流)
- clog(缓冲错误输出流)

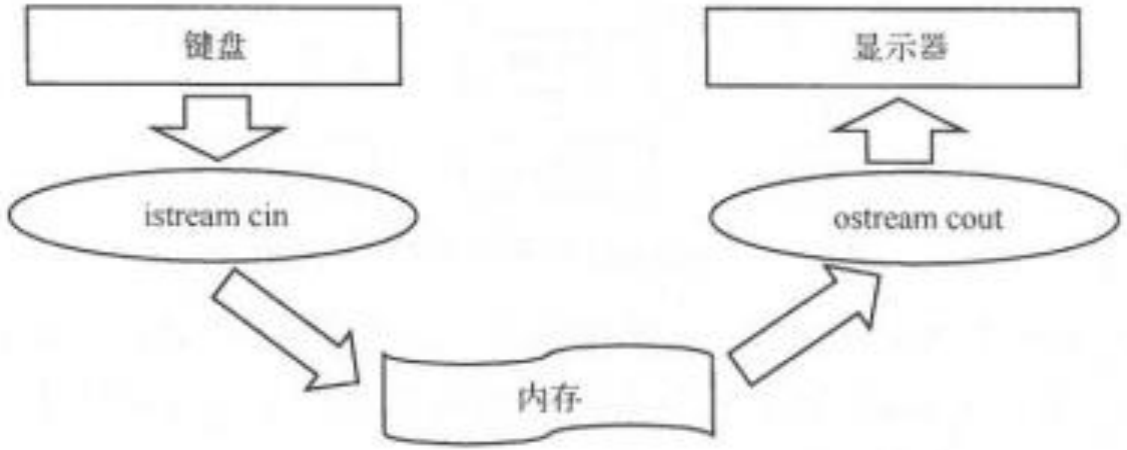


图 7-2　标准流与外设之间的关系

- cin与标准输入设备（键盘）相关联，用于读取数据，可以被重定向为从文件中读取数据
- cout与标准输出设备（显示器）相关联，用于输出数据，可以被重定向为向文件里写入数据
- cerr与标准错误信息输出设备（显示器）相关联（非缓冲），用于输出出错信息，不能被重定向。
- clog与标准错误信息输出设备相关联（缓冲），用于输出出错信息，不能被重定向。
- 在实际中，cin常用于从键盘输入数据，是流类istream的对象。cout常用于向屏幕输出数据，是流类ostream的对象。

【程序7-1】将标准输出cout重定向到文件

```
#include<iostream>
using namespace std;
int main( )
{
        int x,y;
        cin>>x>>y;
        freopen("test.txt","w",stdout);  //将标准输出重定向到文件test.txt
        if(y==0)                         //除数为0则输出错误信息
                cerr<<"error."<<endl;
        else
                cout<<x<<"/"<<y<<"="<<x/y<<endl;
        return 0;
}
```

函数freopen( )的功能是将stream按mode指定的模式重定向到路径path指向的文件。

课堂练习

下列选项中，不能作为输出流的对象是（ ）

A:文件

B:内存

C:键盘

D:显示器

课堂练习

下列选项中，不能作为输出流的对象是（ ）

A:文件

B:内存

C:键盘

D:显示器

答案：C

课堂练习

下列选项中，不是C++中的标准输入输出的是（　）

A:stdin

B:cout

C:clog

D:cerr

课堂练习

下列选项中，不是C++中的标准输入输出的是（　）

A:stdin

B:cout

C:clog

D:cerr

答案：A

课堂练习

在C++中，使用流进行输入输出，其中用于屏幕输入（ ）

A:cin

B:cerr

C:cout

D:clog

课堂练习

在C++中，使用流进行输入输出，其中用于屏幕输入（ ）

A:cin

B:cerr

C:cout

D:clog

答案：A

课堂练习

C++语言的跳转语句中，break和continue说法正确的是（　）

A:break语句只应用于循环体中

B:continue语句只应用于循环体中

C:break是无条件跳转语句，continue不是

D:break和continue的跳转范围不够明确，容易产生问题

课堂练习

C++语言的跳转语句中，break和continue说法正确的是（　）

A:break语句只应用于循环体中

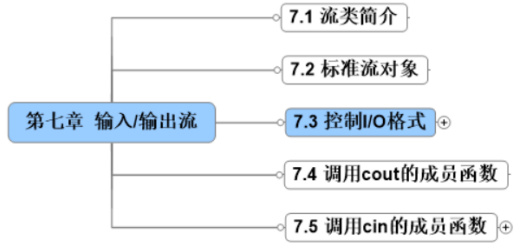
B:continue语句只应用于循环体中

C:break是无条件跳转语句，continue不是

D:break和continue的跳转范围不够明确，容易产生问题

答案：B

7.3 控制I/O格式



C++进行I/O格式控制的方式一般有使用流操纵符、设置标志字和调用成员函数。

7.3.1 流操纵符

流操纵符	作用	输入/输出
endl	换行符 输出一个新行符，并清空流	O
ends	输出字符串结束，并清空流	O
flush	清空流缓冲区	O
dec *(默认)	以十进制形式输入或输出整数	I/O
hex	以十六进制形式输入或输出整数	I/O
oct	以八进制形式输入或输出整数	I/O
ws	提取空白字符	O

7.3.1 流操纵符

流操纵符	作用
fixed	以普通小数形式输出浮点数
scientific	以科学计数法形式输出浮点数
left	左对齐，即在宽度不足时将填充字符添加到右边
right *	右对齐，即在宽度不足时将填充字符添加到左边
setbase(int b)	设置输出整数时的进制，b为8、10或16
setw(int w)	指定输出宽度为w个字符，或输入字符串时读入w个字符。一次有效
setfill(int c)	在指定输出宽度的情况下，输出的宽度不足时用ASCII码为c的字符填充（默认情况是用空格填充）
setprecision(int n)	设置输出浮点数的精度为n。在使用非fixed且非scientific方式输出的情况下，n即为有效数字最多的位数。如果有效数字位数超过n，则小数部分四舍五入，或自动变为科学计数法输出并保留一共n位有效数字；在使用fixed方式和scientific方式输出的情况下，n是小数点后面应保留的位数
setiosflags(fmtfalgs f)	通用操纵符。将格式标志f所对应的格式标志位置为1
resetiosflags(fmtfalgs f)	通用操纵符。将格式标志f所对应的格式标志位置为0(清除)
boolapha	把true和false输出为字符串
noboolalpha *	把true和false分别输出为1和0
showbase	输出表示数值进制的前缀
noshowbase *	不输出表示数值进制的前缀
showpoint	总是输出小数点
noshowpoint *	只有当小数部分存在时才显示小数点
showpos	在非负数值中显示+
noshowpos *	在非负数值中不显示+
skipws *	输入时跳过空白字符
noskipws	输入时不跳过空白字符
uppercase	十六进制数中使用’A’ ~’ E’ 。若输出前缀，则前缀输出“0x”，科学计数法中输出’ E’
no uppercase *	十六进制数中使用’a’ ~’ e’ 。若输出前缀，则前缀输出“0x”，科学计数法中输出’ e’
internal	数值的符号（正负号）在指定宽度内左对齐，数值右对齐，中间由填充字符填充

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    int n=65535,m=20;
    //1）分别输出一个整数的十进制、十六进制和八进制表示
    cout<<"1)"<<n<<"="<<hex<<n<<"="<<oct<<n<<endl;
    //2）使用setbase分别输出一个整数的十进制、十六进制和八进制表示
    cout<<"2)"<<setbase(10)<<m<<"="<<setbase(16)<<m<<"="<<setbase(8)<<m<<endl;
    //3）使用showbase和setbase分别输出一个整数的十进制、十六进制和八进制表示
    cout<<"3)"<<showbase;   //输出表示数值进制的前缀
    cout<<setbase(10)<<m<<"="<<setbase(16)<<m<<"="<<setbase(8)<<m<<endl;
    return 0;
}
```

```
1)65535=ffff=177777
2)20=14=24
3)20=0x14=024
```

课堂练习

下列选项中，用于清除基数格式位设置以十六进制数输出的语句是（ ）

A:cout<<setf(ios::dec,ios::basefield);

B:cout<<setf(ios::hex,ios::basefield);

C:cout<<setf(ios::oct,ios::basefield);

D:cin>>setf(ios::hex,ios::basefield);

课堂练习

下列选项中，用于清除基数格式位设置以十六进制数输出的语句是（ ）

A:cout<<setf(ios::dec,ios::basefield);

B:cout<<setf(ios::hex,ios::basefield);

C:cout<<setf(ios::oct,ios::basefield);

D:cin>>setf(ios::hex,ios::basefield);

答案：B

课堂练习

在C++中使用流进行输入输出，其中用于屏幕输出的对象是（ ）。

A:cerr

B:cin

C:cout

D:cfile

课堂练习

在C++中使用流进行输入输出，其中用于屏幕输出的对象是（ ）。

A:cerr

B:cin

C:cout

D:cfile

答案：C

课堂练习

下列格式控制符，既可以用于输入，又可以用于输出的是（　）

A:setbase

B:setfill

C:setprecision

D:setw

课堂练习

下列格式控制符，既可以用于输入，又可以用于输出的是（　）

A:setbase

B:setfill

C:setprecision

D:setw

答案：D

课堂练习

用于标识十六进制前缀或后缀是（　　）。

A:无

B:后缀L或e

C:前缀零

D:前缀0x

课堂练习

用于标识十六进制前缀或后缀是（　　）。

A:无

B:后缀L或e

C:前缀零

D:前缀0x

答案：D

解析:

十六进制常量——前缀0x

十进制常量——无前后缀

八进制常量——前缀0

长整型常量——后缀L或l

课堂练习

在C++语言中，080是（　）。

A:八进制数

B:十进制数

C:十六进制数

D:非法数

课堂练习

在C++语言中，080是（　）。

A:八进制数

B:十进制数

C:十六进制数

D:非法数

答案：D

在C++中，十进制数直接用0到9表示；

十六进制数如果开头的数字是字母，要在字母前面加0，并且还要在最后加H；

而八进制的数是以0开头，并用0到7表示。八进制中无8。是非法的。

课堂练习

下面的哪个保留字不能作为函数的返回类型（　）。

A:void

B:int

C:new

D:long

课堂练习

下面的哪个保留字不能作为函数的返回类型（　）。

A:void

B:int

C:new

D:long

答案：C

函数返回类型可以是：

预定义类型：如int 或double等。short、long等都与整型常数相同，分别表示短整型、长整型。

复合类型：如int&或double*

用户定义类型：如枚举类或void（指函数不返回值）

课堂练习

用于标识十进制常量的前缀或后缀是（　　）。

A:无前后缀

B:后缀L或l

C:前缀0

D:前缀0x

课堂练习

用于标识十进制常量的前缀或后缀是（　　）。

A:无前后缀

B:后缀L或l

C:前缀0

D:前缀0x

答案：A

解析：

十进制常量——无前后缀

长整型常量——后缀L或l

八进制常量——前缀零

十六进制常量——前缀0x

课堂练习

使用下列哪个格式控制符可设置转换十六进制为十进制（　）。

A:dec

B:oct

C:hex

D:endl

课堂练习

使用下列哪个格式控制符可设置转换十六进制为十进制（　）。

A:dec

B:oct

C:hex

D:endl

答案：A

dec 以十进制形式输入或输出整数

hex 以十六进制形式输入或输出整数

oct 以八进制形式输入或输出整数

endl 输出一个新行符，并清空流

7.3.2 标志字

标志常量名	值	含义	输入/输出
ios::skipws	0X0001	跳过输入中的空白	I
ios::left	0X0002	按输出域左对齐，用填充字符填充右边	O
ios::right *	0X0004	按输出域右对齐，用填充字符填充左边	O
ios::internal	0X0008	在符号位或基数指示符后填入字符	O
ios::dec *	0X0010	转换为十进制基数形式	I/O
ios::oct	0X0020	转换为八进制基数形式	I/O
ios::hex	0X0040	转换为十六进制基数形式	I/O
ios::showbase	0X0080	在输出中显示基数指示符	O
ios::showpoint	0X0100	在输出浮点数时必须带小数点和尾部的0	O
ios::uppercase	0X0200	以大写字母表示十六进制数，科学计数法使用大写字母E	O
ios::showpos	0X0400	正数前加“+”号	O
ios::scientific	0X0800	科学记数法显示浮点数	O
ios::fixed	0X1000	定点形式表示浮点数	O
ios::unitbuf	0X2000	插入操作后立即刷新流	O

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    double x=12.34;
    cout<<"1)"<<setiosflags(ios::scientific|ios::showpos)<<x<<endl;
    cout<<"2)"<<setiosflags(ios::fixed)<<x<<endl;
    cout<<"3)"<<resetiosflags(ios::fixed)
        <<setiosflags(ios::scientific|ios::showpos)<<x<<endl;
    cout<<"4)"<<resetiosflags(ios::showpos)<<x<<endl;   //清除要输出正号的标志
    return 0;
}
```

```
1)+1.234000e+001
2)+12.34
3)+1.234000e+001
4)1.234000e+001
```

7.4 调用cout的成员函数

成员函数	作用相同的流操纵符
precision(int np)	setprecision(np)
width(int nw)	setw(nw)
fill(char cFill)	setfill(cFill)
setf(long iFlags)	setiosflags(iFlags)
unsetf(long iFlags)	resetiosflags(iFIags)

```
#include<iostream>
using namespace std;
int main()
{
    double values[ ]={1.23,20.3456,300.4567,4000.45678,50000.1234567};
    cout.fill('*');                    //设置填充字符为星号*
    for(int i=0; i<sizeof(values) /sizeof(double); i++)
    {
        cout<<"values["<<i<< "]=(";
        cout.width(10);                //设置输出宽度
        cout<<values[i]<< ")" << endl;
    }
    cout.fill(' ');                    //设置填充字符为空格
    int j;
    for(j=0;j<sizeof(values)/sizeof(double);j++)
    {
        cout<<"values["<<j<<"]=(";
        cout.width(10);                //设置输出宽度
        cout.precision(j + 3);         //设置保留有效数字
        cout<<values[j]<<")"<<endl;
    }
    return 0;
}
```

```
values[0]=(******1.23)
values[1]=(***20.3456)
values[2]=(***300.457)
values[3]=(***4000.46)
values[4]=(***50000.1)
values[0]=(      1.23)
values[1]=(     20.35)
values[2]=(    300.46)
values[3]=(   4000.46)
values[4]=(  50000.12)
```

```
#include<iostream>
using namespace std;
int main()
{
    char c='a',str[80]="0123456789abcdefghijklmn";
    int x=65;
    cout<<"cout.put('a'):";    cout.put('a’);
    cout<<"\ncout.put(c+25):"; cout.put(c+25);
    cout<<"\ncout.put(x):";    cout.put(x);
    cout<<"\ncout.write(str,20):";
    cout.write(str,20);  //将str的前20个字节写入到输出流中
    return 0;
}
```

```
cout.put('a'):a
cout.put(c+25):z
cout.put(x):A
cout.write(str,20):0123456789abcdefghij
```

课堂练习

下列输出字符'd'的方法中，错误的是（　）

A:cout<<put('d')

B:cout<<'d'

C:cout.put('d')

D:char a='d';cout<<a;

课堂练习

下列输出字符'd'的方法中，错误的是（　）

A:cout<<put('d')

B:cout<<'d'

C:cout.put('d')

D:char a='d';cout<<a;

答案：A

课堂练习

能够把指定长度的字节序列插入到输出流中的函数名是（　）

A:put

B:write

C:cout

D:printf

课堂练习

能够把指定长度的字节序列插入到输出流中的函数名是（　）

A:put

B:write

C:cout

D:printf

答案：B

成员函数put()的功能是向输出流中插入一个字符c，即字符插入。

成员函数write()的功能是向输出流中插入pch指向的一个长度为nCount的字节序列，即数据块插入。

7.5 调用cin的成员函数

istream类提供了一些公有成员函数，它们可以以不同的方式提取输入流中的数据。

7.5.1 get( )函数


```
#include<iostream>
using namespace std;
int main( )
{
        int n=0;
        char ch;
        while((ch=cin.get( )) !=EOF)  //当文件没有结束时继续进行循环
        {
                cout.put(ch);
                n++;
        }
        cout<<"输入字符共计: "<<n<<endl;
        return 0;
}
```

在Windows环境下，当进行键盘输入时，在单独的一行按〈Ctrl+Z〉组合键后再按〈Enter〉键就代表文件输入结束。

7.5.2 getline( )函数

7.5 调用cin的成员函数
7.5.1 get()函数
7.5.2 getline()函数
7.5.3 eof()函数
7.5.4 ignore()函数
7.5.5 peek()函数

getline( )成员函数的原型如下： 从输入流中读取**一行字符**。

istream & getline(char * buf, int bufSize);

其功能是从输入流中的当前字符开始读取bufSize-1个字符到缓冲区buf，或读到’\n’为止（哪个条件先满足即按哪个执行）。函数会在buf中读入数据的结尾自动添加串结束标记'\0’。

istream & getline(char * buf, int bufSize, char delim);

其功能是从输入流中的当前字符开始读取bufSize-1个字符到缓冲区buf，或读到字符delim为止（哪个条件先满足即按哪个执行）。函数会在buf中读入数据的结尾自动添加'\0'。

两者的区别在于，前者是读到'\n'。为止，后者是读到指定字符delim为止。字符'\n'或delim都不会被存入buf中，但会从输入流中取走。

函数getline( )的返回值是函数所作用的对象的引用。如果输入流中'\n'或delim之前的字符个数达到或超过bufSize，则会导致读入操作出错，其结果是：虽然本次读入已经完成，但是之后的读入都会失败。

7.5.2 getline()函数

```
#include<iostream>
using namespace std;
int main()
{
    char buf[10];
    int i=0;
    while(cin.getline(buf,10))   //若输入流的一行超过9个字符，则会出错
        cout<<++i<<":"<<buf<<endl;
    cout<<"last:"<<buf<<endl;
    return 0;
}
```

```
Hello
1:Hello
this is 2
2:this is 2
123456789
3:123456789
abcdefghij
last:abcdefghi
```

7.5.3 eof()函数

7.5 调用cin的成员函数
7.5.1 get()函数
7.5.2 getline()函数
7.5.3 eof()函数
7.5.4 ignore()函数
7.5.5 peek()函数

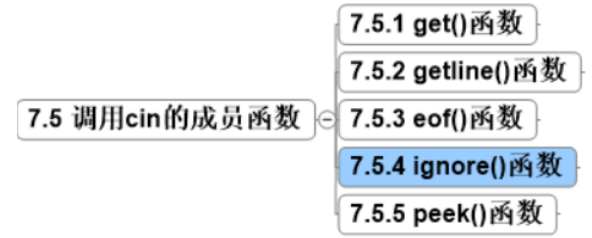
eof( )成员函数的原型如下：

bool **eof( );**

eof( )函数用于判断输入流是否已经结束。返回值为true表示输入结束。

在应用程序中可以用**eof( )函数**测试是否到达文件尾，当文件操作结束遇到文件尾时，函数返回1；否则返回0。

7.5.4 ignore()函数



ignore()成员函数的原型如下：

istream & ignore(int n=1, int delim=EOF);

此函数的作用是跳过输入流中的n个字符，或跳过delim及其之前的所有字符（哪个条件先满足就按哪个执行)。两个参数都有默认值。因此cin.ignore( ) 等效于cin.ignore(1,EOF)，即跳过一个字符。该函数常用于跳过输入中的无用部分，以便提取有用的部分。

```
#include<iostream>
using namespace std;
int main()
{
    char str[30];
    while(!cin.eof())
    {
        cin.ignore(10,':');
        if(!cin.eof())
        {
            cin>>str;
            cout<<str<<endl;
        }
    }
    return 0;
}
```

```
Home: 12345678
Tel: 12345678901
Office: 87654321
要求，将电话号码转换为如下形式：
12345678
12345678901
87654321
```

7.5.5 peek()函数

7.5 调用cin的成员函数
7.5.1 get()函数
7.5.2 getline()函数
7.5.3 eof()函数
7.5.4 ignore()函数
7.5.5 peek()函数

peek()成员函数的原型如下：

int peek( );

函数peek()返回输入流中的当前字符，但是并不将该字符从输入流中取走——相当于只是“看了一眼”将要读入的下一个字符，因此叫“窥视”。

cin.peek( )不会跳过输入流中的空格和回车符。在输入流已经结束的情况下，cin.peek( )返回EOF。

本章总结


第七章 输入/输出流
7.1 流类简介
7.2 标准流对象
7.3 控制I/O格式
7.3.1 流操纵符
7.3.2 标志字
7.4 调用cout的成员函数
7.5 调用cin的成员函数
7.5.1 get()函数
7.5.2 getline()函数
7.5.3 eof()函数
7.5.4 ignore()函数
7.5.5 peek()函数

祝大家顺利通过考试！