第八章 输入输出流标准库

I/O流概述

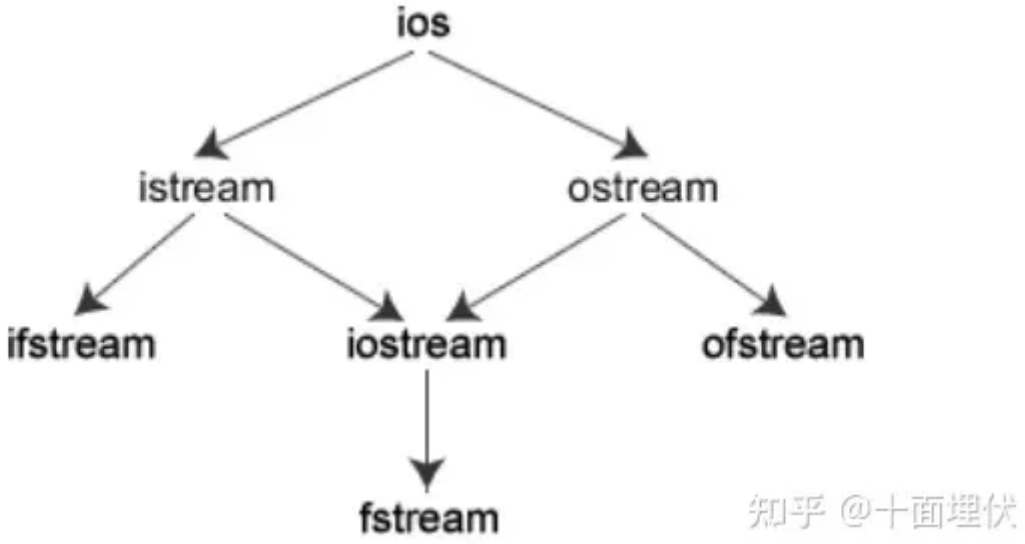
1.什么是流：流是一种抽象，它是对字节结构的数据从一个对象向另一个对象移动的抽象。
当程序**与外界进行数据交换时**，涉及的是程序中的对象和流对象。
在C++程序中，数据可以从键盘或外部文件流入到程序的数据结构中（输入），也可从程序中流向屏幕或外部文件（输出）。
通常把从流中获取数据的操作（读操作）称为提取操作，而向流中添加数据的操作（写操作）称为插入操作。
（输出流用于写入数据，输入流用于读取数据）

I/O流的结构

I/O类库中的常用流类

类名	作用	在哪个头文件中声明
ios	抽象基类	iostream
istream	通用输入流和其他输入流的基类	iostream
ostream	通用输出流和其他输出流的基类	iostream
iostream	通用输入输出流和其他输入输出流的基类	iostream
ifstream	输入文件流类	fstream
ofstream	输出文件流类	fstream
fstream	输入输出文件流类	fstream
istrstream	输入字符串流类	strstream
ostrstream	输出字符串流类	strstream
strstream	输入输出字符串流类	strstream

类的继承关系



预定义流的对象与通用的流运算符

预定义流的对象

- 1.cin 是 istream 类的**对象**，用于处理键盘的输入
- 2.cout 是 ostream 类的**对象**，用于处理屏幕的输出
- 3.cerr 是 ostream 类的对象，用于处理在屏幕上输出出错信息（无缓冲）
- 4.clog 是 ostream 类的对象，用于处理在屏幕上输出错误信息（带缓冲），当缓冲区满时输出

流插入和提取运算符

重载后的流插入运算符（<<）表示写出数据到流对象中，流提起运算符（>>）表示读取数据到计算机内存中。可以重载。

使用输入流提取运算符（>>）如果遇到输入的数据类型有错或者是终止符时，会停止输入（比如空格）

输出流

C++输出流的类主要是**ostream和ofstream**
ostream适合顺序文本的输出
ofstream支持磁盘文件的输出

定义文件输出流对象

1.头文件：iostream.h和fstream.h文件

2.方法：

- 方法一：先定义文件对象，然后调用open成员函数打开该文件

```
ofstream outfile;  //建立文件输出流对象
outfile.open("data.txt",ios::out);
```

期中，ios::out表示文件打开方式为输出（写）方式
用于文件输出流的打开方式还有：

- ios::in，以读模式打开文件；
- ios::app，表示追加方式，文件打开时文件指针位于文件尾；
- ios::ate，表示打开一现存文件并将文件指针指向文件尾；
- ios::nocreate，表示文件存在则打开，否则该操作失败；
- ios::noreplace，表示文件不存在则打开它，文件存在则操作失败；
- ios::trunc，表示打开文件后**清除原有内容**，ios::out隐含该方式；
- ios::binary，表示以二进制模式打开一个文件
  可以同时指定以上几种方式，如：

```
ofstream outfile;
outfile.open("datal.dat",ios::out|ios::binary|ios::app);
```

- 方法二：在定义文件输出流对象并且初始化时打开文件（定义时直接打开）

```
ofstream outfile("datal.dat",ios::out|ios::binary|ios::app);
```

输出流常用成员函数

除了open函数外，输出流常用成员函数还有put、write、seekp、tellp、close、eof等

put函数

用法：

```
输出流对象名.put(ch);
//该函数仅输出包含在ch中的一个字符，返回当前输出流对象
cout.put('A').put('t');
```

write函数

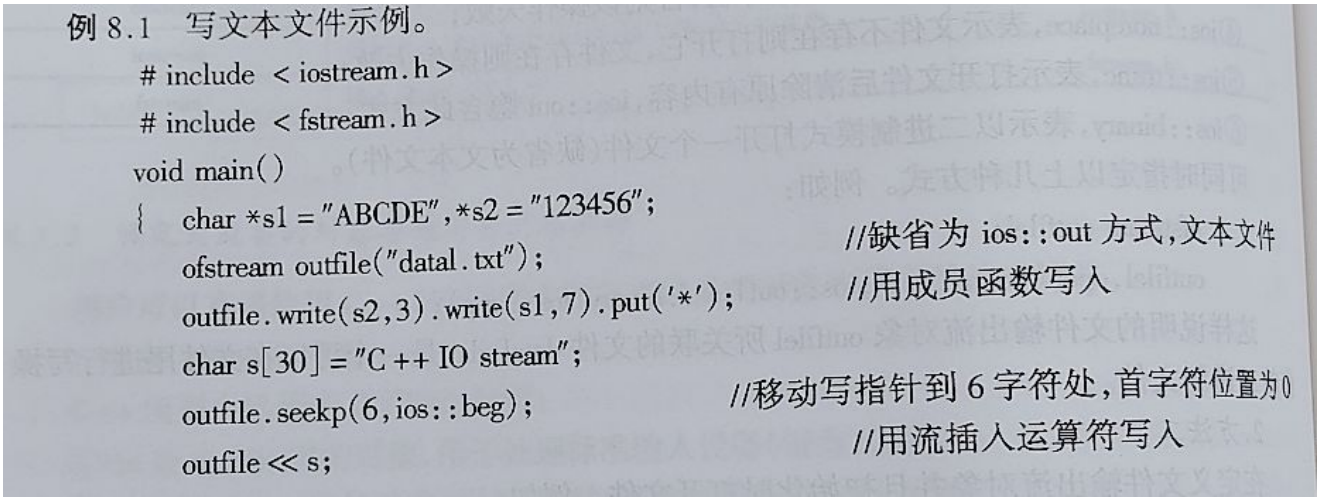
用法:

```
输出流对象名.write(s,n);
//该函数输出字符指针s与所指向的字符串中的n个字符。当s所指字符串不足n时补空格，返回当前输出流对象
cout.write("12345",3).write("ABCDE",7).put('t');
//输出结果为: 123ABCDE t
```

seekp函数

用法:

```
输出流对象名.seekp(流中位置)
输出流对象名.seekp(偏移量,参照位置)
//其中，流中位置，偏移量均为long型，以字节为单位，参照位置有三种
beg(0)   //相对于流的开始位置
cur(1)   //相对于当前写指针所指向的位置
end(2)   //相对于流的结尾处
//seekp函数是设置文件输出流写指针位置用的，指出下册写出数据的起始位置
```



例 8.1 写文本文件示例。

```
# include <iostream.h>
# include <fstream.h>
void main()
{   char *s1="ABCDE",*s2="123456";
    ofstream outfile("datal.txt");                    //缺省为 ios::out 方式,文本文件
    outfile.write(s2,3).write(s1,7).put('*');         //用成员函数写入
    char s[30]="C++ IO stream";
    outfile.seekp(6,ios::beg);                        //移动写指针到 6 字符处,首字符位置为0
    outfile<<s;                                       //用流插入运算符写入
```

该程序在datal.txt文件中开始写入11个字符"123ABCDE*"，移动写指针到6，指向字符D，接着从这个位置写入s数组中的字符串，最后文件中的字符为"123ABCC++ IO stream"

tellp函数

用法:

```
输出流对象名.tellp();
//返回当前输出流写指针的位置值
```

close函数

用法：

```
输出流对象名.close();
//关闭文件，该函数无参数无返回值
```

eof函数

用法：

```
输出流对象.eof();
//执行该函数时，若遇到文件结尾的条件，则返回非0值
```

输入流

C++中输入流类主要是 **istream和ifstream**
istream适合输入顺序文本
ifstream支持磁盘文件的输入

定义文件输入流对象

1.头文件：iostream.h和fstream.h

2.方法：

- 方法一：先定义文件对象，再用open函数打开文件

```
ifstream infile;
infile.open("datal.txt",ios::in);
```

期中，ios::in表示文件打开方式为输入（读）方式
用于文件输入流的打开方式还有：

- ios::nocreate，表示文件存在则打开，否则该操作失败；
- ios::binary，表示以二进制模式打开一个文件
  也可以同时使用（同上）
- 方法二：在定义文件输入流对象时初始化

```
ifstream infile("data1.txt",ios::in);
```

补充：如果既要读又要写，可以定义为fstream类的对象

```
fstream iofile("data1.txt",ios::in|ios::out|ios::binary);
```

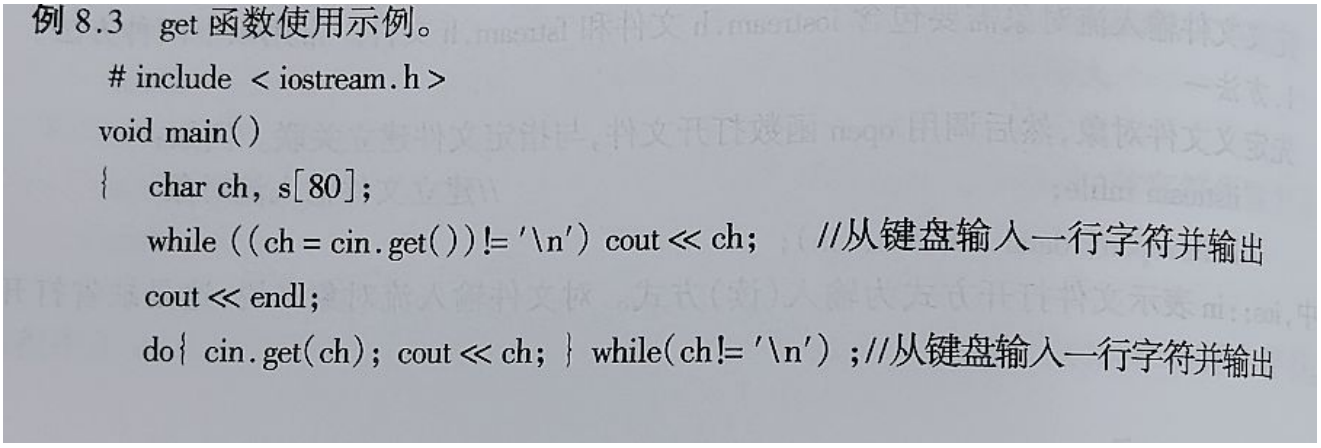
输入流常用成员函数

除了open函数外，常用输入流函数有get、getline、read、seekg、tellg、peek、gcout、close等

get函数

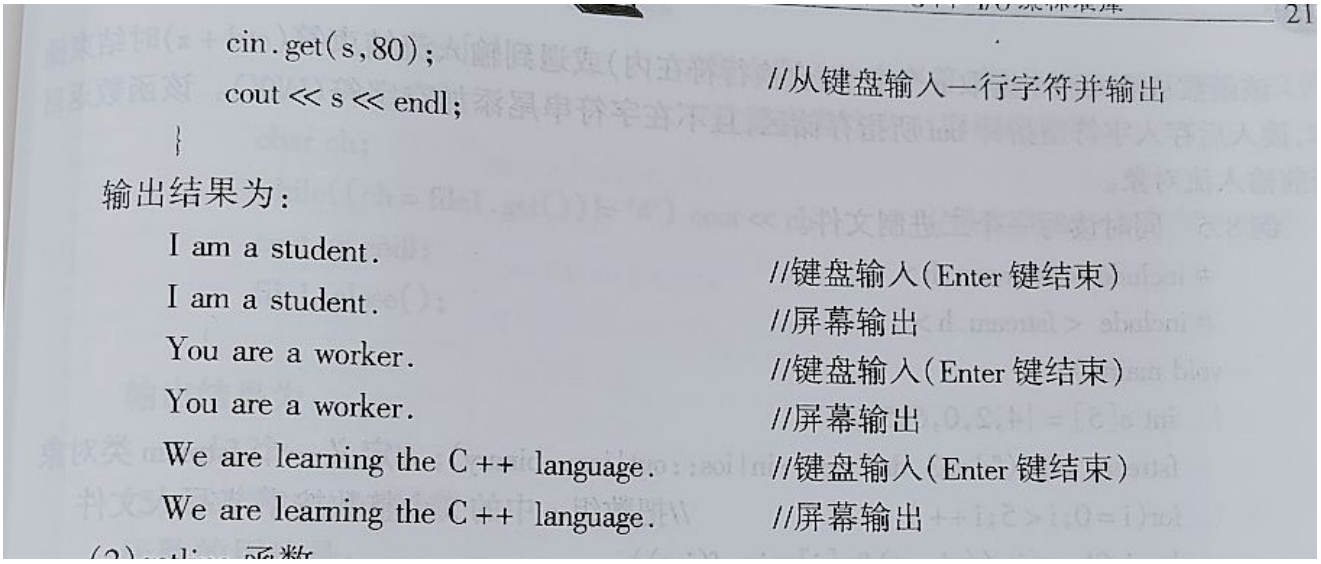
用法:

```
输入流对象名.get();
//从输入流中获取一字符（包括空白符）作为函数值（空白符指空格符、换行符以及转义字符）
// （）里面放变量名
```



例 8.3　get 函数使用示例。

```
#include <iostream.h>
void main()
{  char ch, s[80];
   while ((ch = cin.get()) != '\n') cout << ch;     //从键盘输入一行字符并输出
   cout << endl;
   do{ cin.get(ch); cout << ch; } while(ch != '\n') ;//从键盘输入一行字符并输出
```



```
        cin.get(s,80);
        cout << s << endl;                    //从键盘输入一行字符并输出
    }
```

输出结果为：

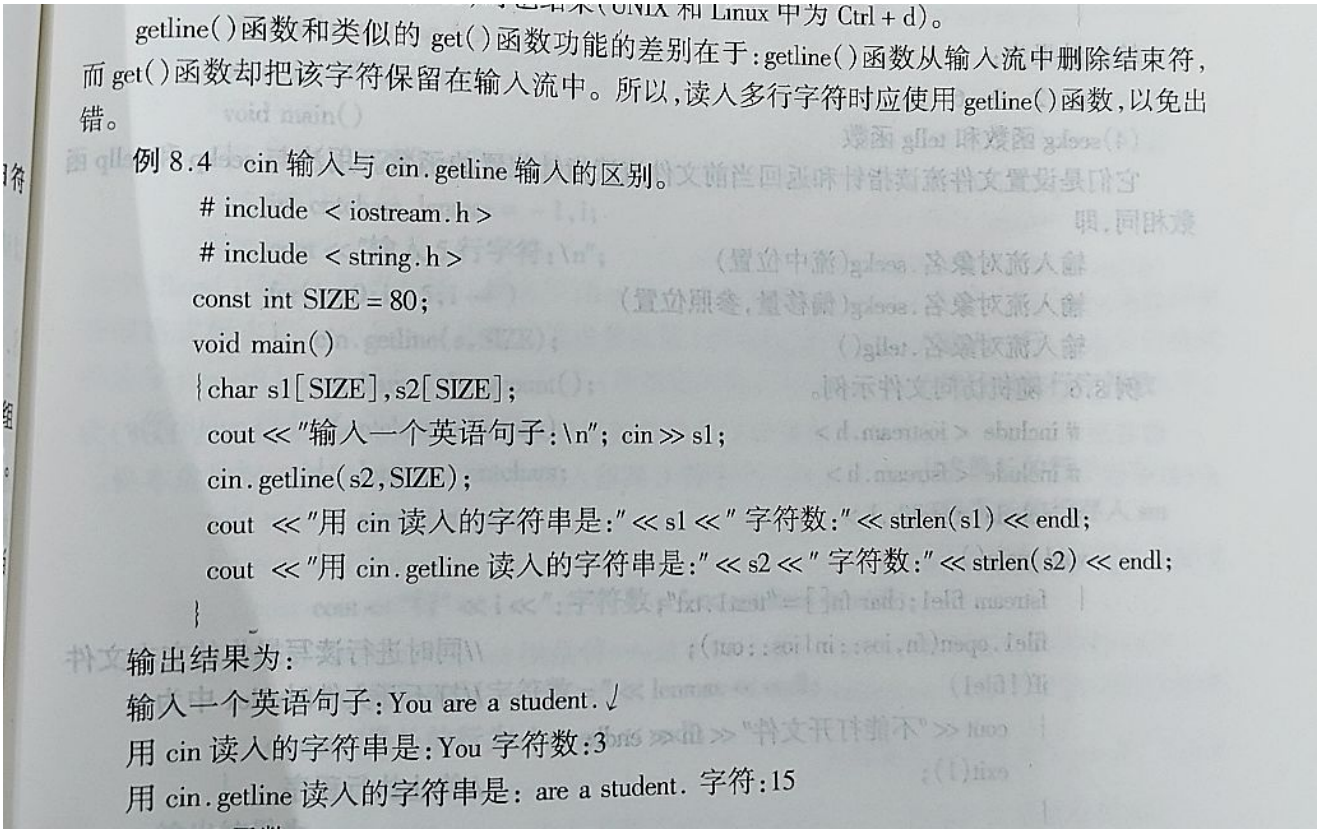
```
    I am a student.                           //键盘输入(Enter 键结束)
    I am a student.                           //屏幕输出
    You are a worker.                         //键盘输入(Enter 键结束)
    You are a worker.                         //屏幕输出
    We are learning the C++ language.         //键盘输入(Enter 键结束)
    We are learning the C++ language.         //屏幕输出
```

getline函数

用法:

```
输入流对象名.getline(buf,n,Delim);
//该函数最多获取n-1个字符并存入指针buf所指向的存储区中，缺省结束符Delim为换行符，插入字符串末尾
```

与别的函数的区别和联系：



getline()函数和类似的 get()函数功能的差别在于：getline()函数从输入流中删除结束符，而 get()函数却把该字符保留在输入流中。所以，读入多行字符时应使用 getline()函数，以免出错。

例 8.4　cin 输入与 cin.getline 输入的区别。

```
#include <iostream.h>
#include <string.h>
const int SIZE = 80;
void main()
{char s1[SIZE],s2[SIZE];
 cout << "输入一个英语句子:\n"; cin >> s1;
 cin.getline(s2,SIZE);
 cout << "用 cin 读入的字符串是:" << s1 << " 字符数:" << strlen(s1) << endl;
 cout << "用 cin.getline 读入的字符串是:" << s2 << " 字符数:" << strlen(s2) << endl;
}
```

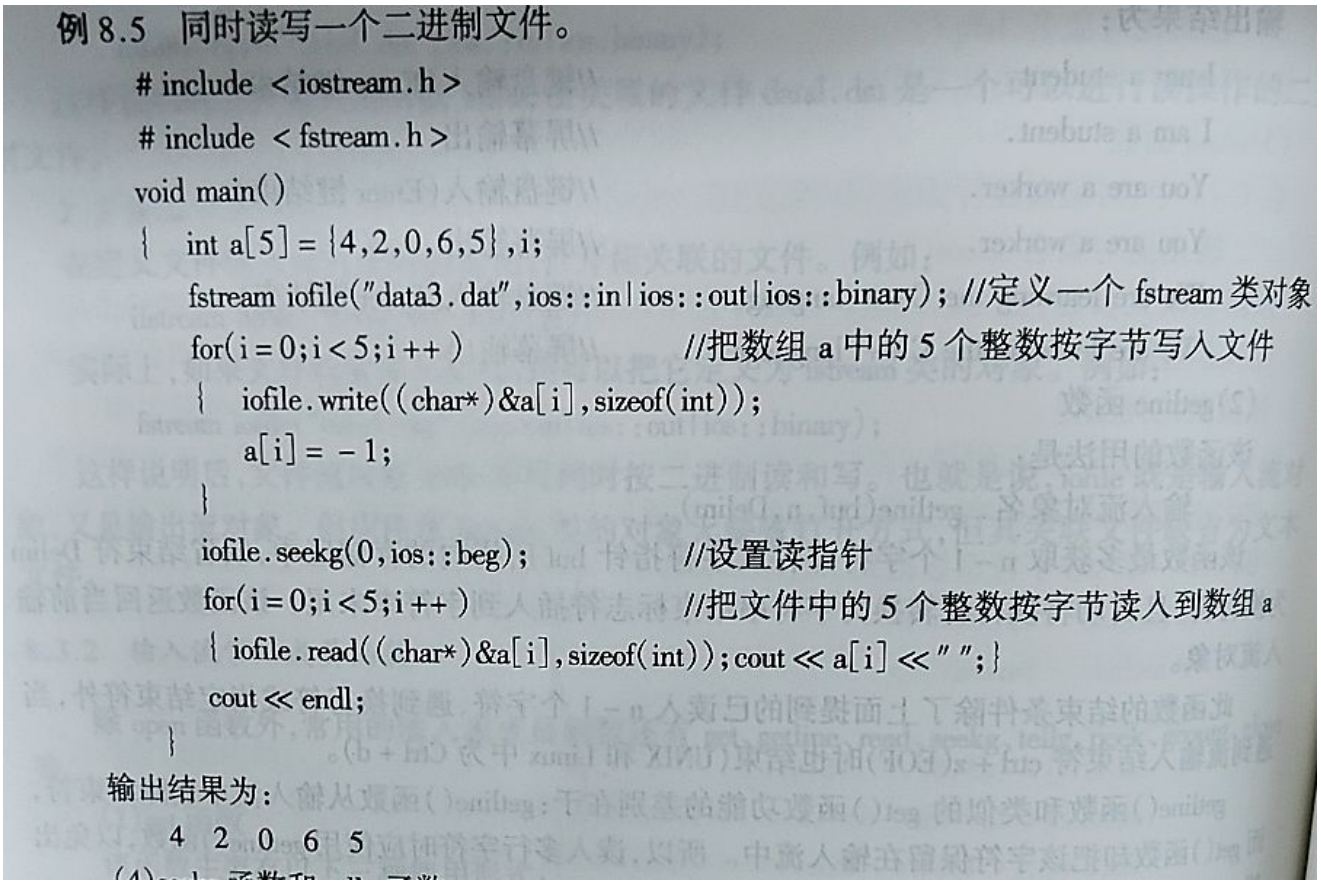
输出结果为：

```
输入一个英语句子:You are a student.↓
用 cin 读入的字符串是:You 字符数:3
用 cin.getline 读入的字符串是: are a student. 字符:15
```

read函数

用法：

```
输入流对象名.read(buf,n);
//从输入流中读取n个字节（包括换行符），或遇到输入流结束符（crtl+z）时结束操作，读入后存入字符型指针buf所指存储区，且不在字符串结尾添加空字符（\0）
```



例 8.5　同时读写一个二进制文件。

```
#include <iostream.h>
#include <fstream.h>
void main()
{   int a[5]={4,2,0,6,5},i;
    fstream iofile("data3.dat",ios::in|ios::out|ios::binary); //定义一个 fstream 类对象
    for(i=0;i<5;i++)                    //把数组 a 中的 5 个整数按字节写入文件
    {   iofile.write((char*)&a[i],sizeof(int));
        a[i]=-1;
    }
    iofile.seekg(0,ios::beg);           //设置读指针
    for(i=0;i<5;i++)                    //把文件中的 5 个整数按字节读入到数组 a
    { iofile.read((char*)&a[i],sizeof(int));cout<<a[i]<<" ";}
    cout<<endl;
}
```

输出结果为：

```
4  2  0  6  5
```

seekg函数与tellg函数

用法与seekp和tellp函数相同

peek和gcount函数

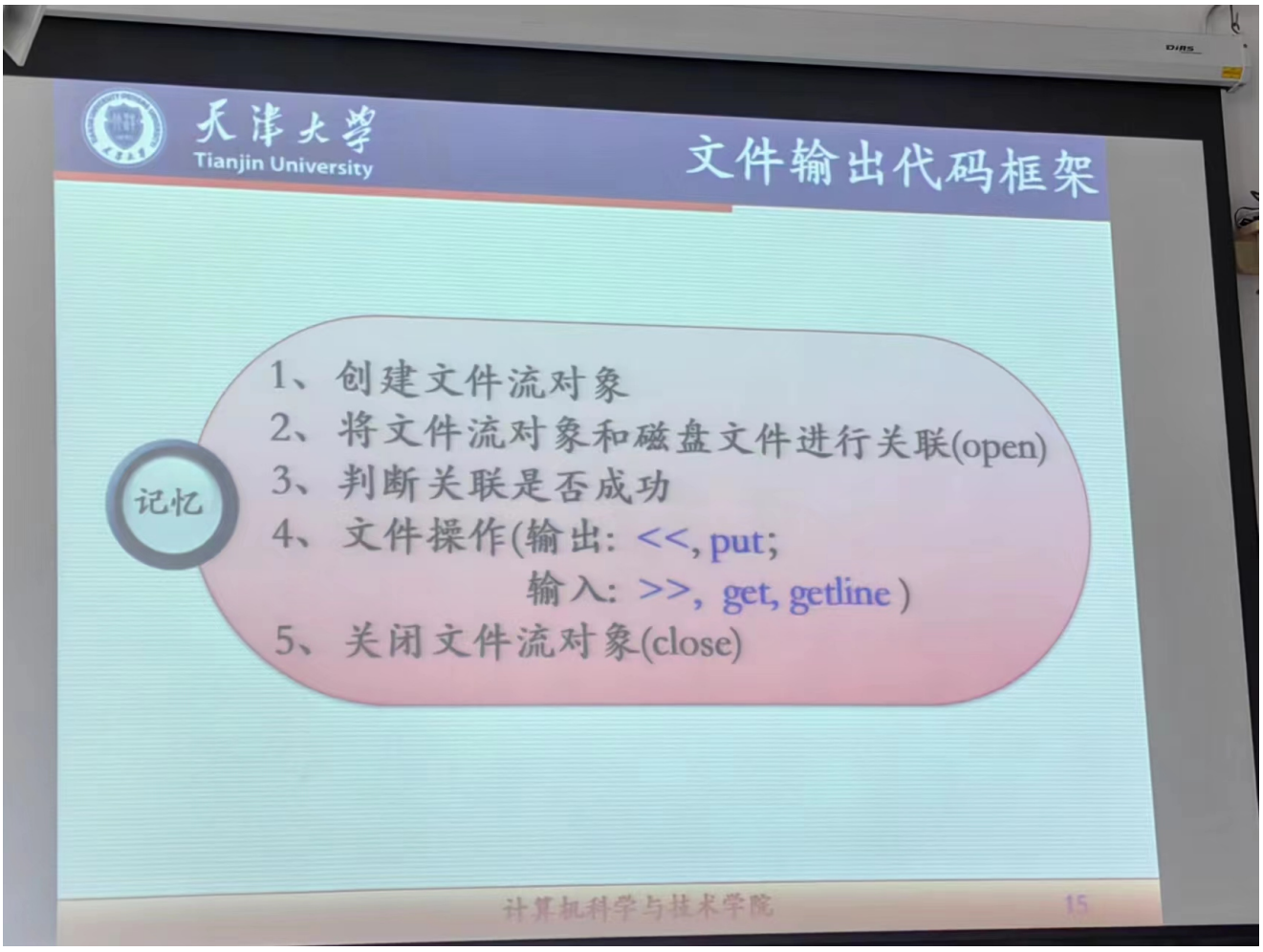
用法：

```
输入流对象名.peek();
输入流对象名.gcount();
//都返回一涵整数值，前者返回的是输入流中下一个即将被读取的字符，后者返回最近一次用getline或read函数时间从输入流中读取的字符个数
```

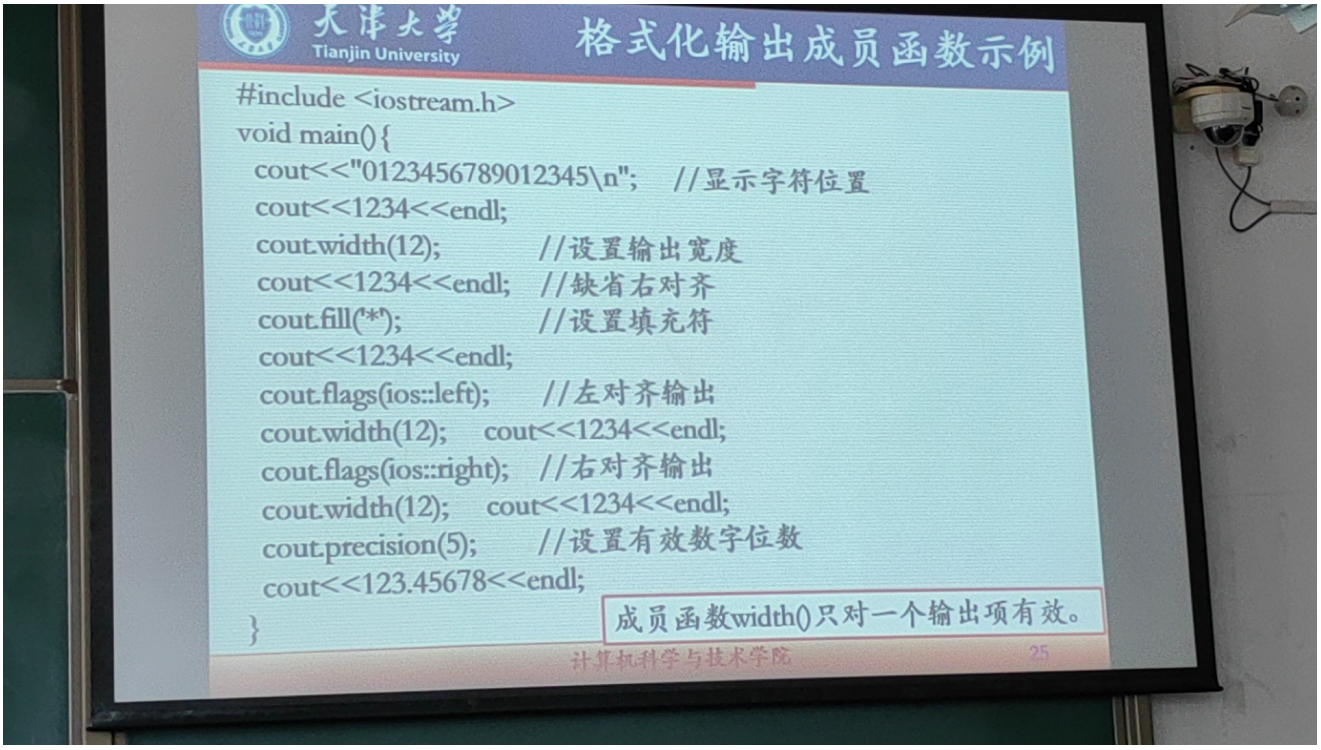
ignore和putback函数

略

总结


天津大学
Tianjin University
文件输出代码框架
记忆
1、创建文件流对象
2、将文件流对象和磁盘文件进行关联(open)
3、判断关联是否成功
4、文件操作(输出: <<, put;
输入: >>, get, getline)
5、关闭文件流对象(close)
计算机科学与技术学院
15

格式化输出


天津大学
Tianjin University
格式化输出成员函数示例
#include <iostream.h>
void main(){
cout<<"0123456789012345\n"; //显示字符位置
cout<<1234<<endl;
cout.width(12); //设置输出宽度
cout<<1234<<endl; //缺省右对齐
cout.fill('*'); //设置填充符
cout<<1234<<endl;
cout.flags(ios::left); //左对齐输出
cout.width(12); cout<<1234<<endl;
cout.flags(ios::right); //右对齐输出
cout.width(12); cout<<1234<<endl;
cout.precision(5); //设置有效数字位数
cout<<123.45678<<endl;
}
成员函数width()只对一个输出项有效。
计算机科学与技术学院
25