



# 程序设计实习

郭炜 微博 <http://weibo.com/guoweiofpku>

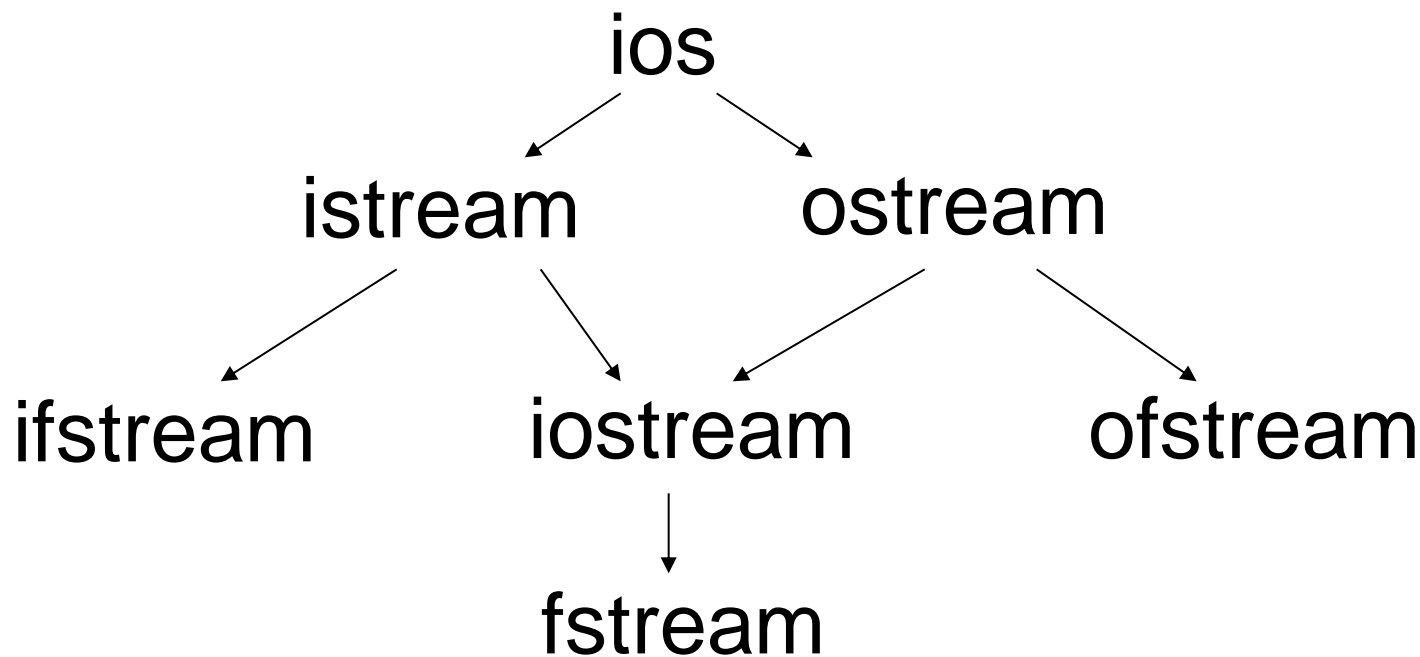
<http://blog.sina.com.cn/u/3266490431>

刘家瑛 微博 <http://weibo.com/pkuliujiaying>



# 输入和输出

# 与输入输出流操作相关的类



# 与输入输出流操作相关的类

`istream`是用于输入的流类，`cin`就是该类的对象。

`ostream`是用于输出的流类，`cout`就是该类的对象。

`ifstream`是用于从文件读取数据的类。

`ofstream`是用于向文件写入数据的类。

`iostream`是既能用于输入，又能用于输出的类。

`fstream` 是既能从文件读取数据，又能向文件写入数据的类。

# 标准流对象

- 输入流对象: `cin`    与标准输入设备相连
- 输出流对象: `cout`    与标准输出设备相连
- `cerr`    与标准错误输出设备相连
- `clog`    与标准错误输出设备相连

缺省情况下

```
cerr << "Hello, world" << endl;
```

```
clog << "Hello, world" << endl;
```

和

```
cout << "Hello, world" << endl;    一样
```

# 标准流对象

- `cin`对应于标准输入流，用于从键盘读取数据，也可以被重定向为从文件中读取数据。
- `cout`对应于标准输出流，用于向屏幕输出数据，也可以被重定向为向文件写入数据。
- `cerr`对应于标准错误输出流，用于向屏幕输出出错信息，
- `clog`对应于标准错误输出流，用于向屏幕输出出错信息，
- `cerr`和`clog`的区别在于`cerr`不使用缓冲区，直接向显示器输出信息；而输出到`clog`中的信息先会被存放在缓冲区，缓冲区满或者刷新时才输出到屏幕。

# 输出重定向

```
#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 test.txt 文件
    if( y == 0 ) //除数为0则在屏幕上输出错误信息
        cerr << "error." << endl;
    else
        cout << x /y ; //输出结果到 test.txt
    return 0;
}
```

# 输入重定向

```
#include <iostream >
using namespace std;
int main() {
    double f;      int n;
    freopen("t.txt","r",stdin); //cin被改为从 t.txt中读取数据
    cin >> f >> n;
    cout << f << "," <<n << endl;
    return 0;
}
```

t.txt:  
3.14 123

输出:  
3.14,123



# 判断输入流结束

可以用如下方法判断输入流结束：

```
int x;  
while(cin>>x) {  
    ...  
}  
return 0;
```

```
istream &operator >>(int a)  
{  
    .....  
    return *this;  
}
```

- 如果是从文件输入，比如前面有  
`freopen("some.txt", "r", stdin);`  
那么，读到文件尾部，输入流就算结束
- 如果从键盘输入，则在单独一行输入Ctrl+Z代表输入流结束

# istream类的成员函数

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

从输入流中读取bufSize-1个字符到缓冲区buf，或读到碰到‘\n’为止（哪个先到算哪个）。

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

从输入流中读取bufSize-1个字符到缓冲区buf，或读到碰到delim字符为止（哪个先到算哪个）。

两个函数都会自动在buf中读入数据的结尾添加‘\0’。，‘\n’或delim都不会被读入buf，但会被从输入流中取走。如果输入流中‘\n’或delim之前的字符个数达到或超过了bufSize个，就导致读入出错，其结果就是：虽然本次读入已经完成，但是之后的读入都会失败了。

可以用 `if(!cin.getline(...))` 判断输入是否结束

# istream类的成员函数

`bool eof();` 判断输入流是否结束

`int peek();` 返回下一个字符,但不从流中去掉.

`istream & putback(char c);` 将字符ch放回输入流

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

从流中删掉最多nCount个字符,遇到EOF时结束。

# istream类的成员函数

```
#include <iostream >
using namespace std;
int main() {
    int x;
    char buf[100];
    cin >> x;
    cin.getline(buf,90);
    cout << buf << endl;
    return 0;
}
```

输入:

12 abcd✓

输出:

abcd (空格+abcd)

输入

12✓

程序立即结束，输出:

12

因为getline读到留在流中的'\n'  
就会返回