Chap06 输入输出流

程勇

北京化工大学

信息科学与技术学院计算机系

Sept. 2020

课程提纲

- Chap01 绪论
- Chap02 面向对象程序设计方法
- Chap03 重载与类型转换
- Chap04 继承与派生
- Chap05 多态性
- Chap06 输入输出流
- Chap07 容错与异常处理
- Chap08 模版

本章提纲

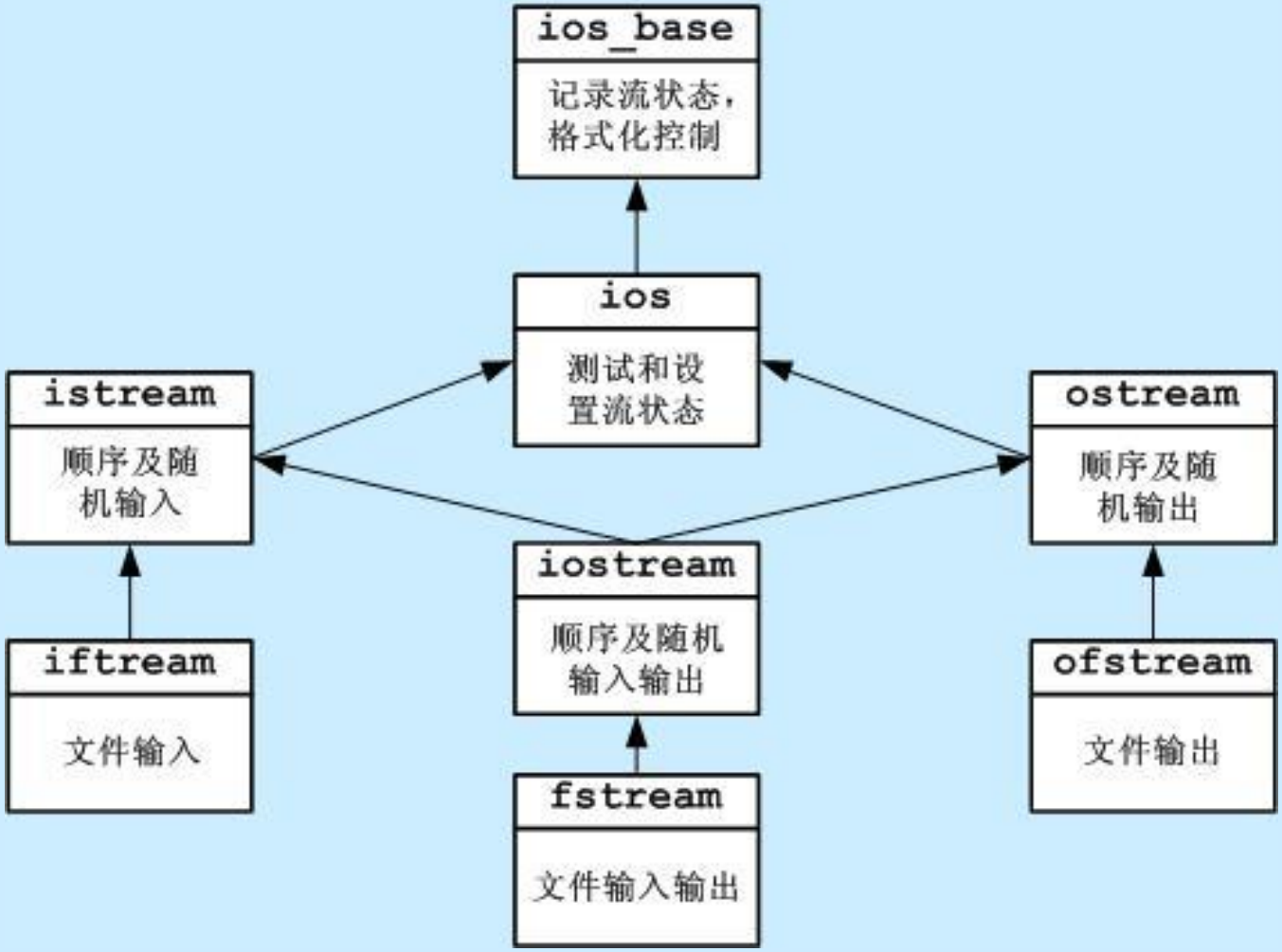
- 输入与输出流概念
- 标准输出流
- 标准输入流
- 文件流
  - 文件打开模式
  - 写入文件
  - 读取文件
  - 二进制模式流操作
- 对象流

基本概念

- C++的输入和输出
  - 大多数情况下，C++的输入和输出并不是C++标准的一部分，而是C++开发环境的一部分；
  - C++标准库不仅可以提供文件的输入输出功能，而且还可以使用基于字符串的I/O进行数据格式化；
- 流
  - 流是程序中输入和输出设备的抽象表示，输入和输出设备分别是数据的来源和目的地，因此流也可以看作外部设备和计算机内存之间流动的一系列字节；
  - 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；
  - 使用流主要优点为输入输出操作实现独立于具体的物理设备，这样不必担心每个设备的具体机制，而且还可以处理不同的输入和输出设备而必须修改操作代码；

基本概念(续)

- 流类层次结构



基本概念(续)

- C++流类定义
  - ios_base是一个普通类，其他都模板的实例，如下所示：

    typedef basic_ios<char> ios;

    typedef basic_istream<char> istream;

    typedef basic_ostream<char> ostream;

    typedef basic_iostream<char> iostream;

    typedef basic_ifstream<char> ifstream;

    typedef basic_ofstream<char> ofstream;

    typedef basic_fstream<char> fstream;
  - 标准输入流
    - extern istream cin;
  - 标准输出流
    - extern ostream cout;
    - extern ostream cerr;
    - extern ostream clog;

流的插入与提取操作

- 流的插入操作
  - operator<<()函数实现为ostream类的重载；

```
std::cout << i << ‘ ‘<< x;   // 该语句调用了三次operator<<()函数
operator<<(std::cout.operator <<(i), ‘ ‘).operator <<(x);
```

  - 给输出流写入单个字符的函数原型(非成员函数)

```
ostream& operator<<(ostream& out, char ch);
```

  - 输出非空字符串的函数原型(非成员函数)

```
ostream& operator<<(ostream& out, const char* pStr);
```

- 流的提取操作
  - operator>>()函数实现为istream类的重载；

```
std::cin>>i>>x;   // 该语句调用了二次operator>>()函数
(std::cin.operator>>(i)).operator>>(x);
```

  - 给输出流写入单个字符的函数原型(非成员函数)

```
istream& operator>>(istream& in, signed char* pStr);
istream& operator>>(istream& in, unsigned char* pStr);
```

文件流

- 文件流
  - 有三种流对象可以处理文件，分别是ifstream、ofstream、fstream，它们分别可以可读取的文件流、可写入的文件流和可读写的文件流；
- 写入文件
  - 先创建ofstream对象，然后就可以以默认的文本模式写入数据，写入完毕后要关闭文件流；

```
std::ofstream outfile("C:/demo/primes.txt");
outfile<<*(primes + i);
```

- 读取文件
  - 先创建ifstream对象，把它与物理文件关联起来，然后就可以以文本模式读取文件内容，完毕后关闭文件流；

```
const char* filename = "C:/demo/primes.txt";
std::ifstream infile(filename);
long aprime = 0;
infile>>aprime;
```

文件打开模式(1)

- 文件打开模式
  - ifstream和ofstream对象的文件打开模式确定了处理文件的方式；
  - 文件打开模式由ios_base类中定义的位掩码值和ios类中继承为openmode类型的掩码值组合来确定；
- 文件流位掩码值

值	含　义
ios::app	每次写入操作之前移动到文件的末尾(追加)
ios::ate	打开文件时移动到文件末尾，之后可以移动到其他位置
ios::binary	设置二进制模式
ios::in	打开文件，进行读取
ios::out	打开文件，进行写入
ios::trunc	把已有的文件长度变为0

文件打开模式(2)

- 详细解释
  - in标识打开文件用于输入操作(从文件读取)。打开方式只要含in，如文件不存在则返回失败。在打开为输入输出方式时（同时用out），编程应注意判断是否失败，失败时千万不可再写入文件。
  - out标识打开文件用于输出操作(写入文件)。如文件不存在，则建立新文件，如文件存在，未同时设app, in则文件清空。
  - trunc标识打开文件，并清空它（文件长度截为0）。文件不存在则建立新文件，与out默认操作相同。但与in配合，文件不存在则返回失败。
  - app标识打开文件用于输出，原文件内容保留，新数据接在尾部
  - ate意思是at end，标识打开文件，文件指针在文件尾，但文件指针可以移动，即新数据可写到任何位置。文件是否清空由其它标识决定。

    以上三个标识最好配合out、in等一起用，因为不同的C++平台，要求不同，一起用不会出错。不一起用，至少VC++不认这种格式。
  - binary标识以二进制方式打开文件。同时用out时，如文件不存在，则建立新文件，并且新文件能用，不必清状态字。

二进制文件

- 二进制文件优点
  - 可以控制字节长度，读写数据时不会出现二义性，可靠性高。同时不知格式是无法读取的，保密性好。文件结束后，系统不会再读(见eofbit的说明)，但程序不会自动停下来，所以要判断文件中是否已没有数据。如写完数据后没有关闭文件，直接开始读，则必须把文件定位指针移到文件头。如关闭文件后重新打开，文件定位指针就在文件头。
- 文件的随机访问
  - 在C++中可以由程序控制文件指针的移动，从而实现文件的随机访问，即可读写流中任意一段内容。一般文本文件很难准确定位，所以随机访问多用于二进制文件。
  - 公有枚举类型(ios类中说明)

```
enum seek_dir{
    beg=0, //文件开头
    cur=1, //文件指针的当前位置
    end=2 //文件结尾
};
```

二进制模式流操作

- 二进制模式复制文件
  - 设置文件模式

```
ifstream in(source.c_str(), ios::in, ios::binary);
```

  - 读一个字符

```
in.get(ch);
```

  - 写一个字符

```
out.put(ch);
```

- 二进制模式下写入数值数据
  - 标准库中没有以二进制模式读写数据值的流函数，但使用read()和write()函数就可以一组把任意数值类型写为二进制数据的函数；
  - 将double类型的数值写入文件

```
void write(std::ostream& out, double value){
    out.write(reinterpret_cast<char*>(&value), sizeof(double));
}
```

对象流

- 可以使用基本类型的提取和插入运算符来读写对象，需要重载这两个运算符
  - 重载插入运算符

```
std::ostream& operator<<(std::ostream& out, const Box& rbox){
    return out << rbox.length << " " << rbox.width << " " <<rbox.height;
}
```

  - 重载提取运算符

```
std::istream& operator>>(std::istream& in, Box& rbox){
    return in >> rbox.length >> rbox.width >> rbox.height;
}
```

  - 处理派生类

    派生类的情况比较复杂，因为operator>>()和operator<<()都不是类的成员，因此需要虚拟的输入输出机制，描述如下：

    - virtual std::ostream& put(std::ostream& out) const;
    - virtual std::istream& get(std::istream& in);

本章小结

- 输入与输出流概念
- 标准输出流
- 标准输入流
- 文件流
- 对象流

- 课程网站
  - http://www.jiaowu.buct.edu.cn