



# 输入/输出流

C++流的概念

格式化I/O

文件流

字符串流

# 输入输出的途径

- C语言中**没有**提供专门的**输入输出语句**,
- 同样, C++语言中**也没有**专门的**输入/输出(I/O)**语句,
- C++中的**I/O**操作是通过一组**标准I/O函数和I/O流**来实现的。

# 优点

- C++的标准I/O函数是从C语言得来的，同时对C语言的标准I/O函数进行了扩充。
- C++的I/O流不仅拥有标准I/O函数的功能，而且比标准I/O函数功能更强、更方便、更可靠。

# C++流的概念

- 在C++语言中，数据的输入和输出（简写为I/O）包括：
  1. 对标准输入设备键盘和标准输出设备显示器；
  2. 对在外存磁盘上的文件；
  3. 和对内存中指定的字符串存储空间进行输入输出。

# 三种I/O

- 对标准输入设备和标准输出设备的输入输出简称为**标准I/O**,
- 对在外存磁盘上文件的输入输出简称为**文件I/O**,
- 对内存中指定的字符串存储空间的输入输出简称为**串I/O**。

# 流

- 流是一种抽象，它负责在数据的**生产者**和数据的**消费者**之间建立联系，并管理数据的流动。
- 程序建立一个**流对象**，并指定这个流对象与某个文件对象建立连接，程序操作流对象，流对象通过文件系统对所连接的文件对象产生作用。
- 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**。

- 流以内存为中心

- 冯.诺依曼

- 在**C++**中，流既可以表示数据从内存传送到某个载体或设备中，即**输出流**。

- 也可以表示数据从某个载体或设备传送到内存缓冲区变量中，即**输入流**。

# 预定义对象

- 为用户进行标准I/O操作定义了四个类对象，它们分别是
  - **cin**
  - **cout**
  - **cerr**
  - **clog**



# 流操作符

- C++的流通过重载运算符“<<”和“>>”执行输入和输出操作。
- 输出操作是向流中**插入**一个字符序列，因此，在流操作中，将运算符“<<”称为**插入运算符**。
- 输入操作是从流中**提取**一个字符序列，因此，将运算符“>>”称为**提取运算符**。

# 输入输出流

- `#include <iostream>`
  - `istream`类
  - `ostream`类
- 使用步骤
  - 1. 产生流对象
  - 2. 操作流对象
- 系统预先产生了四个对象
  - 无需用户产生对象

# 1 cout

- Ostream流的对象
- 主要操作
  - <<
  - 被重载
  - endl

## 2 cin

- istream对象
- 主要操作
  - >>
  - get
- 如何获取空白字符

***char ch;***

***cin>>ch;***

***//ch=cin.get();***

***cout<<ch<<endl;***

### 3 cerr

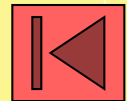
**cerr**类似标准错误文件。**cerr**与**cout**的差别在于：

- (1) **cerr**是**不能重定向**的；
- (2) **cerr****无缓冲**，它的输出总是直接传达到标准输出设备上。

## 4 clog

- 日志

**clog**是不能重定向的，但是可以被缓冲。



# 输出的格式化

- C语言中printf通过格式符的使用，可以精确控制输出的格式
- C++的输出流同样可以实现
  - 直接控制
  - 通过iomanip控制

- 程序1:

```
#include<iostream>
#include<iomanip>
using namespace std;
void main()
{
    int x=30, y=300, z=1024;
    cout<<x<<' '<<y<<' '<<z<<endl; //按十进制输出
    cout<<oct<<x<<' '<<y<<' '<<z<<endl; //按八进制输出
    cout<<hex<<x<<' '<<y<<' '<<z<<endl; //按十六进制输出
    cout<<setiosflags(ios::showbase | ios::uppercase);
        //设置基指示符和数值中的字母大写输出
    cout<<x<<' '<<y<<' '<<z<<endl; //仍按十六进制输出
    cout<<resetiosflags(ios::showbase | ios::uppercase);
        //取消基指示符和数值中的字母大写输出
    cout<<x<<' '<<y<<' '<<z<<endl; //仍按十六进制输出
    cout<<dec<<x<<' '<<y<<' '<<z<<endl; //按十进制输出
}
```





- 此程序的运行结果如下:

- 30 300 1024

- 36 454 2000

- 1e 12c 400

- 0X1E 0X12C 0X400

- 1e 12c 400

- 30 300 1024

- 程序2:

```
#include<iostream>
#include<iomanip>
using namespace std;
void main()
{
    int x=468;
    double y=-3.425648;
    cout<<"x="<<setw(10)<<x;
    cout<<"y="<<setw(10)<<y<<endl;
    cout<<setiosflags(ios::left); //设置按左对齐输出
    cout<<"x="<<setw(10)<<x;
    cout<<"y="<<setw(10)<<y<<endl;
    cout<<setfill('*'); //设置填充字符为'*'
    cout<<setprecision(3); //设置浮点数输出精度为
    cout<<setiosflags(ios::showpos); //设置正数的正号输出
    cout<<"x="<<setw(10)<<x;
    cout<<"y="<<setw(10)<<y<<endl;
    cout<<resetiosflags(ios::left | ios::showpos);
    cout<<setfill(' ');
}
```

此程序运行结果如下：

**x= 468y= -3.42565**

**x=468 y=-3.42565**

**x=+468\*\*\*\*\*y=-3.43\*\*\*\*\***

# 文件流

## 文件的概念

在磁盘上保存的信息是**按文件的形式组织**的，每个文件都对应一个文件名，并且属于某个物理盘或逻辑盘的目录层次结构中一个确定的目录之下。

一个文件名由**文件主名和扩展名**两部分组成，它们之间用圆点（即小数点）分开，扩展名可以省略，当省略时也要省略掉前面的圆点。

# 文件流

- `#include <fstream>`
- `ifstream`
  - 文件输入流
- `ofstream`
  - 文件输出流
- `fstream`
  - 既可以输入也可以输出

## • 文件的打开

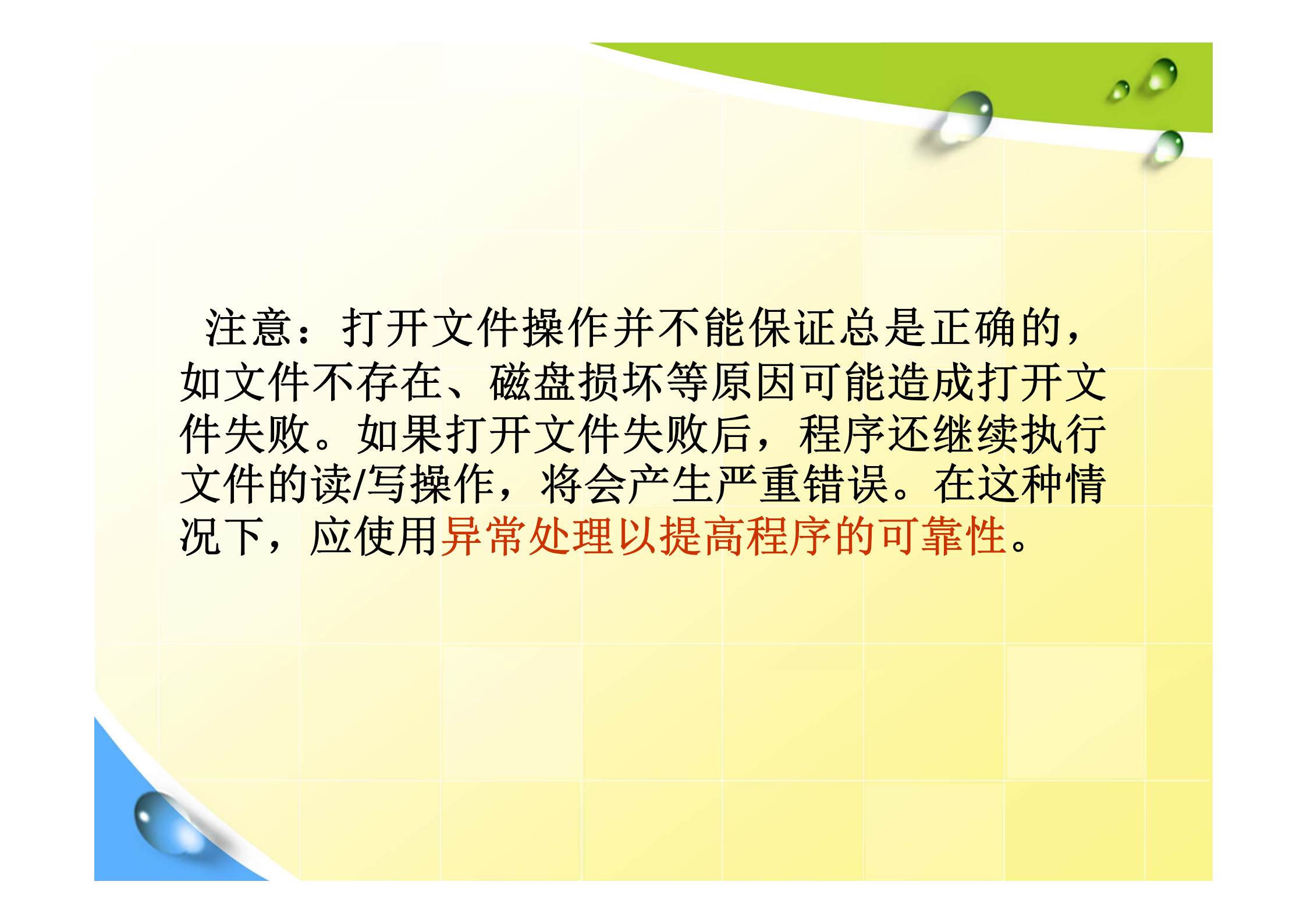
– **open** ( ) 的函数原型为:

```
void open(const char * filename,int mode,int prot=filebuf::openprot);
```

**mode**的取值必须是以下值之一或组合:

<b>ios::in</b>	打开文件进行读操作
<b>ios::out</b>	打开文件进行写操作
<b>ios::ate</b>	打开时文件指针定位到文件尾
<b>ios::app</b>	添加模式，所有增加都在文件尾部进行
<b>ios::trunc</b>	如果文件已存在则清空原文件
<b>ios::nocreate</b>	如果文件不存在则打开失败
<b>ios::noreplace</b>	如果文件存在则打开失败
<b>ios::binary</b>	二进制文件（非文本文件）

- 对于ifstream流，mode的默认值为ios::in;
- 对于ofstream流，mode的默认值为ios::out。
- **prot**决定文件的访问方式，取值为：
  - 0 普通文件
  - 1 只读文件
  - 2 隐含文件
  - 4 系统文件
  - 一般情况下，该访问方式使用默认值。
- **mode**的可用位或运算“|”组合
  - 如ios::in|ios::binary表示以只读方式打开二进制文件。



注意：打开文件操作并不能保证总是正确的，如文件不存在、磁盘损坏等原因可能造成打开文件失败。如果打开文件失败后，程序还继续执行文件的读/写操作，将会产生严重错误。在这种情况下，应使用异常处理以提高程序的可靠性。



# 文件的读写

## (1) 使用流运算符直接读写。

文件的读/写操作可以直接使用流的插入运算符“<<”和提取运算符“>>”，这些运算符将完成文件的字符转换工作。

## (2) 使用流成员函数

常用的输出流成员函数为：**put**函数、**write**函数

常用的输入流成员函数如下：**get**函数、**getline**函数、**read**函数

# 文件输入流成员函数

- **open**函数把该流与一个特定磁盘文件相关联。
- **get**函数的功能与提取运算符（>>）很相像，主要的不同点是**get**函数在读入数据时包括空白字符
- **getline**的功能是从输入流中读取多个字符，并且允许指定输入终止字符，读取完成后，从读取的内容中删除终止字符。
- **read**成员函数从一个文件读字节到一个指定的内存区域，由长度参数确定要读的字节数。  
如果给出长度参数，当遇到文件结束或者在文本模式文件中遇到文件结束标记字符时结束读取。

# 文件输出流成员函数

- **open**函数  
把流与一个特定的磁盘文件关联起来。
- **put**函数  
把一个字符写到输出流中。
- **write**函数  
把内存中的一块内容写到一个输出文件流中
- **seekp**和**tellp**函数  
操作文件流的内部指针
- **close**函数  
关闭与一个输出文件流关联的磁盘文件

# 字符串流

- 字符串流类包括输入字符串流类**istream**，输出字符串流类**ostream**和输入输出字符串流类**stringstream**三种。
- 它们都被定义在系统头文件**stringstream**中。
- 只要在程序中带有该头文件，就可以使用任一种字符串流类定义**字符串流对象**。每个字符串流对象简称为字符串流。

三种字符串流类的构造函数声明格式分别如下：

**istream(const char\* buffer);**

**ostream(char\* buffer, int n);**

**stringstream(char\* buffer, int n, int mode);**

对字符串流的操作方法通常与对字符文件流的操作方法相同。

# 字符串流的例子

```
#include<iostream>
#include<sstream>
using namespace std;
void main()
{
    char buff[128]={0};
    ostringstream os(buff,128);
    os<<"hello";

    cout<<buff<<endl;

    char buff1[128]="1 2.5 A";
    istringstream is(buff1);
    int num1;
    double num2;
    char num3;
    is>>num1>>num2>>num3;

    cout<<num1<<" "<<num2<<" "<<num3<<endl;
}
```