**输入、输出与文件**

**C++ 输入和输出概述：**

**流和缓冲区 ：**

1. 通常流充当了程序和流源或目标之间的桥梁 ；
2. 通过使用流，c++ 程序处理方式独立与其取向 ；

**用于输入输出的类对象 ：**

**cin** : 关联标准输入设备 （wcin与wchar\_t类型匹配）

**cout** : 关联标准输出流 （wcin与wchar\_t类型匹配）

**cerr** : 对应标准错误流 （这个流没有被缓冲， wcerr 对应 wchar\_t 类型）

**clog** : 对应标准错误流 （有缓冲 ， wclog 对应 wchar\_t 类型）

**使用 cout 进行输出 ：**

1. **重载 << 运算符 ：**

输出的所有的类型数据都自动转换为字符 ；

插入运算符的返回类型为 ostream & 即允许拼接输出 ；

1. **ostream 方法：**

**put :**

原型 : **ostream & put (char) ;** // cout.put(‘d’).put(‘j’) ;

注意 ： 可以将 数值型参数传递给 put( ) （将转化为 ASCLL） ；

**write :**

模板原型 ：**basic\_ostream <char T , traits> & write (const char\_type \*s , streamsize n ) ;**

说明 ： write 第一个参数提供了要显示字符串的地址 ， 第二个参数指出了显示的字节数目

**刷新缓冲区 :**

**endl :** 插入换行符 ， 刷新缓冲区 ；

**flush :** 刷新缓冲区 ;

1. **使用 cout 进行格式化 ：**
2. **修改显示时的计数系统 ：**

**dec :**  十进制 ；

**hex :** 十六进制 ；

**oct :**  八进制 ；

1. **调整字宽度 ：**

原型 ：

**int width ( ) ;** // 返回字宽度的当前设置

**int width ( int i ) ;** // 将字宽度设置为 i 并返回以前的字宽度 ；

1. **填充字符 ：**

**fill ( ) ;**

1. **设置浮点数的精度 :**

**precision ( ) ;**

**说明 ：**

浮点数精度的含义取决于输出模式 ， 在默认情况下它指的是显示的总位数 ， 在定点模式和科学模式下，精度指的是小数点后的位数 ；

1. **stef ( ) :**

**说明 ：**ios\_base 类有一个受保护的数据成员 ， 其中的各位分别控制这格式化的各方面 ， 打开一个标记称设置标记 ， 并意味这相应为设置为1 ， 位标记是编程开关 ；

**注意 ：**

这样的修改一直有效 ， 直到被覆盖 ；

**原型Ⅰ ：**

**fmflags setf ( fmflags ) ;**

**格式常量 ：**

**ios\_base::boolalpha :** 输入和输出 bool 值，可以为 true 或 flase .

**ios\_base::showbase :** 使用c++ 基数前缀 ；

**ios\_base::showpoint :** 显示末尾的小数点跟末尾的0 填充 ；

**ios\_base::showpos :** 在正数前面加上 + ;

**ios\_base::uppercase :** 对于16 进制输出，使用大写字母 E 表示法

**原型Ⅱ ：**

**fmflags setf ( fmflags , fmflags ) ;**

|  |  |  |
| --- | --- | --- |
| **第二参数** | **第一参数** | **含义** |
| **ios\_base::basefield** | **`ios\_base::dec** | **十进制** |
| **ios\_base::hex** | **十六进制** |
| **ios\_base::oct** | **八进制** |
| **ios\_base::floatfield** | **ios\_base::fixed** | **定点计数法** |
| **ios\_base::scientific** | **科学计数法** |
| **ios\_base::adjustfield** | **ios\_base::left** | **左对齐** |
| **ios\_base::right** | **右对齐** |
| **ios\_base::internal** | **符号左，值右** |

**注意 ：**

1. serf ( ) 返回值类型为 fmflags 数字 ； 用来指出所有标记以前的设置， 如果想要恢复原始设置 ， 则可保存这个值 ；
2. 浮点数默认显示模式 **cout .setf ( 0 , ios\_base::floatfield ) ;**

**unsetf ( )** // 消除设置的位 ；

1. **标准控制符 :**

**boolalpha // noboolalpha // showpoint // noshowpoint // showbase // noshowbase // showpos // noshowpos // uppercase // nouppercase**

**internal // left // right**

**dec // hex // oct**

**fixed // scientific**

1. **iomanip :**

**setprecision( ) : //** 设置精度

**setw ( ) //** 设置字段宽度 ；

**setfill ( ) //** 设置填充

**使用 cin 进行输入 ：**

**cin >> 如何检查输入 ：** 他们跳过空白，直到遇到非空白字符

流状态 :

|  |  |
| --- | --- |
| **成员** | **描述** |
| **eofbit** | **到达文件尾 ，则被设置** |
| **badbit** | **文件读取错误等则被设置** |
| **failbit** | **未能读到预期字符则被设置** |
| **goodbit** | **另一种表示0的方法** |
| **eof( )** | **eof被设置则返回 true** |
| **bad ( )** | **类上** |
| **fail ( )** | **类上** |
| **rdstate ( )** | **返回流状态** |
| **exceptions ( )** | **返回一个位掩码 ， 指出哪些标记导致异常被引发** |
| **exceptions ( isostate ex )** | **设置哪些状态将导致 clear ( ) 引发异常 ；** |
| **clear (iostate s)** | **将流状态设置为 s // 默认 goodbit** |
| **setstate (iostate s)** | **将流状态设置为 s // 其他位不变** |

**I/O 和异常 ：**使用 exceptions ( ) 可设置相关位字段异常 ；

**其他istream 类方法 :**

**单字符输入 :**

**get (char &) :**

可拼接 、可接受空格等字符

当读取到达文件尾时 ， 将设置 // failbit

**get (void) :**

返回类型为 int 不可拼接 ；

当读取到达文件尾时 ， 将设置 EOF

**字符串输入 ：**

**istream & get ( char \* , int , char ) ;**

**istream &get (char \* , int ) ;**

**istream &getline ( char \* , int , char ) ;**

**istream &get (char \* , int ) ;**

此处不再解释作用 ；

**ignore ( ) :** 接受并丢弃字符串

**说明 ：**

接受两个参数 ： 一个数字 ， 指定要读取的最大字符数 ； 另一个是字符， 用作输入分界符 。；

接受并丢弃字符串 ；

**read ( ) :**

**注意 ：** read ( )不会在输入后加上空值字符 ， 因此不能将输入转化成字符串 ， 它最常与 ostream write( ) 函数结合使用 ；

**peek( ) :**

返回输入中的下一个字符 ， 但不抽取输入流中的字符

**gcount :**

返回最后一个非格式化抽取方法读取的字符数 ， 这意味着是由 get ( ) , getline() ,ignore( ) , read ( ) ;

**putback ( ) :**

将一个字符插入到输入字符串中 ， 被插入的字符将是下一条输入语句读取的第一个字符 ， putback ( ) 接受一个参数 ， 就是要插入的字符 ；

**文件输入和输出 :**

**流状态检查和 is\_open ( ) :**

is\_open ( ) : 能过检测出其他方式不能检测的微妙问题 ;

**命令行处理技术 ：**

C++ 有一种让在命令行环境中运行的程序能够访问命令行参数的机制 ， 方法是使用下面的 main ( ) 函数 ：

**int main ( int argc , char \* argv [] )**

**文件模式 :**

|  |  |
| --- | --- |
| **模式** | **含义** |
| **ios\_base::in** | 打开文件 ，读取 |
| **ios\_base::out** | 打开文件 ， 写入 |
| **ios\_base::ate** | 打开文件 ， 移到文件尾 |
| **ios\_base::app** | 追加到文件尾 |
| **ios\_base::trunc** | 如果文件存在则截断文件 |
| **ios\_base::binary** | 二进制文件 |

**二进制文件 ：**

**使用二进制模式**

**： c++ mode || ios\_base::binary**

**使用 write( ) 跟read ( ) 进行读写 :**

**随机存取 :**

**类方法 :**

**seekg ( ) :将输入指针移动到指定文件位置 ；**

**seepg ( ) : 将输出指针移动到指定文件位置 ;**

**版本Ⅰ：**

只有一个参数 ， 接受一个整数 ， 指定将指针移动到离开始处的位置；

**版本Ⅱ：**

接受两个参数 ， 第一个参数如上 ， 第二个指出参考位置

**beg // cur // end**

**tellp ( ) : 返回输出指针位置**

**tellg ( ): 返回输入指针位置**

**使用临时文件 ：**

**char \* tmpnam (char \* pname ) ;**

**内核格式化 :**

**头文件 : sstream ;;**

**ostringstream ( ) : 将 string 对象与输出流相关联 // 可以将 cout 方法用于其中**

**istringostream( ): 将 string 对象与输入流相关联 // 可以将 cin 方法用于其中**