C++程序设计精要教程

华中科技大学

第14章 流及类库

◆14.1 流类概述

- 流的类库是C++在C文件流基础上的直接扩展，流是从源(source)到矢(sink)的数据流抽象
  - 从源输入：提取、得到和取来
  - 输出到矢：插入、存放和存储
- C++的流主要分为两大类，一类用于输入／输出，另一类用作输入／输出缓冲
  - C++提供没有缓冲的流操作

第14章 流及类库

◆14.1 流类概述

- 操作系统将键盘、显示器、打印机等映射为文件。输入／输出类可作为源或矢，或兼作源和矢
  - iostream.h定义带缓冲的输入／输出
  - fstream.h定义文件输入／输出
  - constream.h定义控制台输入／输出
  - strstream.h定义串输入／输出
- 类iostream有两个平行的类系列，即streambuf派生的类和ios派生的类
  - streambuf类提供了流与物理设备的接口
  - ios类包含一个指向streambuf的指针，它使用streambuf进行I/O格式化及错误检查

第14章 流及类库

◆14.1 流类概述

- C++预定义了4个流类对象，即cin、cout、cerr和clog标准流类对象，当C++程序开始执行时，这4个流类对象已被构造好，且不能被应用程序析构
  - extern istream_withassign cin; //相应于stdin
  - extern ostream_withassign cout; //相应于stdout
  - extern ostream_withassign cerr; //相应于stderr
  - extern ostream_withassign clog; //相应于有缓冲的cerr
- cerr与clog之间的区别是cerr没有缓冲，发送给cerr的内容立即输出

第14章 流及类库

◆14.2 输出流述

- **输出流通过重载左移运算符<<实现输出，其左操作数为ostream_withassign类型的对象cout，右操作数为所有简单类型的右值表达式**

  例如：cout<<"Hello!\n";

- **该语句隐含地调用cout.operator<<(const char *str)，该函数输出参数str所指定的字符串，并返回ostream_withassign类型的引用cout**
- **上述函数调用的结果可进一步作为<<的左操作数**

第14章 流及类库

◆14.2 输出流

- 运算符<<重载后仍然保持自左至右的结合方式，因此，可以一次自左至右地输出多个右值表达式

  例如：cout<<"i="<<i<<", d="<<d<<"\n";

- 由于重载不改变运算符的优先级，故运算符的优先级较<<高的运算可以不用括号

  例如： cout<<"sum="<<3+5<<"\n";

- 运算符优先级较<<低的运算则必须用括号

  例如： cout<<"X&y="<<(x&y)<<"\n";

第14章 流及类库

◆14.2 输出流

- **输出流为运算符<<预定义的右操作数的数据类型有：**
  char、short、int、long等有符号或无符号的整数类型
  char *、float、double、long double和void *等类型
- **所有的输出按printf规定的转换规则进行转换**
  例如，下面的两个输出语句产生完全一样的输出结果：

```
int   m;
long  n;
cout<<m<<'\t'<<n;
printf("%d\t%ld", m, n);
```

第14章 流及类库

◆14.2 输出流

●输出格式：由cout各种状态标志确定

●状态标志：由ios中public类型的枚举量定义

```
enum {
   skipws,          //跳过输入中的空白：空格、回车、换行及制表符等
   left ,           //左对齐输出
   right ,          //右对齐输出
   internal,        //在符号或基指示后填补
   dec,             //按十进制转换
```

第14章流及类库

```
    oct,            //按八进制转换
    hex,            //按十六进制转换
    showbase,               //在输出中使用基指示
    showpoint,              //在浮点输出中显示小数点
    uppercase,              //大写十六进制输出
    showpos,                //对正整数显示 +
    scientific,             //用科学计数法表示
    fixed,          //用小数点表示浮点数
    unitbuf,                //所有流在输出后刷新
    stdio                   //在输出到stdout, stderr后刷新
};
```

第14章 流及类库

◆14.2 输出流

- 可以使用以下函数成员来读取、设置和清除标志：

```
long  flags( );              //读取字符格式标志
long  flags(long);           //设置字符格式标志
long  setf(long, long);      //清除和设置字符格式标志
long  setf(long);            //设置字符格式标志
long  unsetf(long);          //清除字符格式标志
```

- 改变输出格式：可以使用操纵符改变输出宽度、填充字符等与输出格式有关的变量
  - 操纵符可以同输入／输出的变量或数据一起使用
  - 所有的操纵符都定义在iomanip.h中，引用前须包含#include

第14章流及类库

【例14.1】使用操纵符改变输出格式

```
#include <iostream>
#include <iomanip>
using namespace std;
void main(void)
{
    int i=3456, j=9012, k=78;
    cout<<setw(6)<<i<<j<<k<<"\n";
    cout<<setw(6)<<i<<setw(6)<<j<<setw(6)<<k;
}
```

上述程序产生的输出为：

```
3456901278
3456  9012    78
```

注意：setw(int)对输出流的影响只是暂时的

第14章 流及类库

◆14.2 输出流

- 带参数操纵符函数有setfill、setprecision、setiosflags、resetiosflags、setbase等
- 程序可以定义自已的操纵符函数,但不能带参数 .C++预定义的操纵符函数有：

```
dec( );                 //设置十进制转换
hex( );                 //设置十六进制转换
oct( );                 //设置八进制转换
ws( );                  //提取空白字符
endl( );                //插入回车并刷新输出流
ends( );                //插入空字符以终止串
setbase(int);           //设置进制标志为0,8,10,16。0表示缺省为十进制
resetiosflags(long);    //清除格式位
setiosflags(long);      //设置格式位
setfill(int);           //设置填充字符
setprecision(int);      //设置浮点精度位数
setw(int);              //设置域宽
```

- 注意，对于不带参数的dec及hex操纵符函数，调用时不写括号，它们对输出流的影响是长久的。

第14章流及类库

【例12.2】定义输出流的格式

```
#include <iostream>
using namespace std;
void main(void)
{
    int i=12;
    cout<<hex<<i<<i;
    cout<<i<<i<<endl;
    cout<<dec<<i<<i;
    cout<<i<<i<<endl;
}
```

上述程序的输出为：

```
cccc
12121212
```

第14章流及类库

【例14.3】重载输出流的运算符“<<”。

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string.h>
using namespace std;
struct CLERK{
    char *name;
    int  age;
public:
    CLERK(const char *, int);
    ~CLERK( ) noexcept;

};
```

第14章流及类库

```
CLERK::CLERK(const char *n, int a)
{
    name=new char [strlen(n)+1];
    strcpy(name, n);
    age=a;
}
CLERK::~CLERK( ) noexcept { delete name; }
ostream & operator<<(ostream &s, CLERK &c)              //重载为非成员函数
{
    return s<<c.name<<' '<<c.age;
}
void main(void)
{
    CLERK c("Zhang", 23);
    cout<<c;          //调用operator<<(cout, c)
}
```

第14章 流及类库

◆14.2 输出流

- C++为流定义了一些输出函数成员，这些函数是以字符或块为单位操作的
- 当输出的数据为字符类型时，输出函数按无符号和有符号字符进行重载。原型如下：

```
ostream &flush( );                          //刷新输出流
ostream& put(char);                         //输出一个字符
ostream& seekp(long);                       //确定输出位置
ostream& seekp(long, seek_dir);   //确定输出位置
long tellp( );                              //读取输出位置
ostream& write(const char*, int n);//输出一个字符块
```

第14章 流及类库

◆14.3 输入流

- 从流中输入（或称提取），输入流通过重载运算符>>实现输入
- 重载后运算符函数>>的左操作数为istream类型的对象，右操作数为预定义类型的引用
- 在缺省情况下，用运算符>>输入时将先跳过空白符，然后输入对应于输入对象的字符
- 是否跳过空白符由ios定义的skipws确定，若清除该标志将不跳过空白符
- 可通过操作符ws设置skipws标志，skipws被缺省设置为跳过空白符

第14章流及类库

【例12.4】输入流的用法

```
#include <iomanip>
#include <iostream>
using namespace std;
void main(void)
{
    char c, d, s[80];
    long f;
    f=cin.flags( );                 //返回格式化标志，缺省为跳过空白
    f=cin.flags(0L);                //设置格式化标志，返回原格式化标志
    cin>>c>>d;                      //不跳过空白字符输入
    cin>>ws>>c>>d;                  //跳过空白字符输入
    cin.flags(f);                   //恢复原格式化标志为跳过空白
    cin.width(sizeof(s)-1);         //避免溢出
    cin>>e;                         //跳过空白输入字符串
}
```

第14章 流及类库

◆14.4 文件流

- 文件输入流为ifstream，文件输出流为ofstream，这两个文件流都定义在包含文件fstream.h中

  ifstream继承了istream和fstreambase

  ofstream继承了ostream和fstreambase

- 前面介绍的格式化函数以及输入／输出函数都可以用于文件流
- 文件流对象必须在文件打开后才能输入／输出，在文件关闭后才能再次打开文件。定义文件流对象和打开文件可以同时进行，例如：

  ```
  ifstream f1("input");    ofstream f2("output");
  ```

第14章 流及类库

◆14.4 文件流

- 在缺省情况下，文件用正文模式打开，类ios定义了多种文件打开模式，这些模式包括：

ios::app	在文件尾追加数据
ios::ate	在已打开文件上找到文件尾
ios::in	打开的文件供读
ios::out	打开的文件供写，缺省为trunc方式
ios::binary	以正文方式打开文件
ios::trunc	若文件存在，则消除原文件内容
ios::nocreate	若要打开的文件不存在，则打开失败
ios::noreplace	除非同时设置ate或app，否则文件存在时打开失败

第14章流及类库

【例14.5】使用文件流编写文件拷贝程序。

```
#include <fstream>
#include <iostream>
using namespace std;
int main(int argc, char* argv[])
{
    ifstream f1;
    ofstream f2;
    char ch;
     if (argc != 3) {
        cerr << "Parameters error!\n";
        return 1;
    }
    f2.open(argv[2], ios::in + ios::ate);
    if (f2) {                //若文件存在
```

第14章流及类库

```
        cerr << "Object file already exist!\n";
        f2.close( );
        return 1;
    }
    f1.open(argv[1], ios::in+ios::binary);
    if ((!f1)) {            //若文件不存在
        cerr << "Source file open error!\n";          return 1;
    }
    f2.open(argv[2], ios::out+ios::binary);
    if ((!f2)) {
        cerr << "Object file open error!\n";
        f1.close( );              return 1;
    }
    while (f1.get(ch))              f2.put(ch);
    f1.close( );        f2.close( );
    return 0;
}
```

第14章 流及类库

◆14.5 串流处理

- 正如sscanf和sprintf一样，strstream.h定义的函数能按格式输入／输出字符串，且输入／输出格式更为丰富灵活。

  istrstream由类istream和strstreambase派生而来

  ostrstream是由类ostream和strstreambase派生而来

- 前面所介绍的有关流的格式化函数及输入／输出函数都能用在串流处理中。
- 例如，某种格式的正文文件按行存入商品标识、价格以及有关商品的描述信息。对于如下格式的正文文件：

  101 191 Big Book

  102 100.12 Small Book

第14章 流及类库

◆14.5 串流处理

- **如果要依次读入各行正文，加上相应的行号并打印出来，即产生如下形式的输出：**

  1: 101　191.00　Big Book

  2: 102　100.12　Small Book

- **必须使用串流输入/输出函数**

第14章流及类库

【例14.6】字符串流的输入 / 输出用法。

```
#define _CRT_SECURE_NO_WARNINGS
#include <fstream>
#include <strstream>
#include <iomanip>
#include <iostream>
#include <string.h>
using namespace std;
int main(int argc, char* argv[ ])
{
    int id;
    float amount;
    char description[41];
```

第14章流及类库

```
    ifstream inf(argv[1]);
    if (inf) {
        char inbuf[256];
        int lineno = 0;
        cout.setf(ios::fixed, ios::floatfield);
        cout.setf(ios::showpoint);
        while (inf.get(inbuf, 81)) {
            istrstream ins(inbuf, strlen(inbuf));
            ins >> id >> amount >> ws;
            ins.getline(description, 41);
            cout << ++lineno << ":" << id << '\t' << setprecision(2);
            cout << amount << '\t' << description << '\n';
        }
    }
    return 0;
}
```