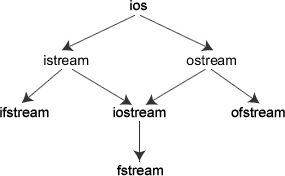
### **1、流的控制**



**iomanip**在使用格式化I/O时应包含此头文件。  
    **stdiostream**用于混合使用C和C + +的I/O机制时，例如想将C程序转变为C++程序

### **2、类继承关系**

****  
ios是抽象基类，由它派生出istream类和ostream类， iostream类支持输入输出操作，iostream类是从istream类和ostream类通过多重继承而派生的类

类ifstream继承了类istream，类ofstream继承了类ostream，类fstream继承了 类iostream  
**iostream头文件中4种流对象**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **对象** | **含义** | **对应设备** | **对应的类** | **c语言中相应的标准文件** |
| cin | 标准输入流 | 键盘 | istream\_withassign | stdin |
| cout | 标准输出流 | 屏幕 | ostream\_withassign | stdout |
| cerr | 标准错误流 | 屏幕 | ostream\_withassign | stderr |
| clog | 标准错误流 | 屏幕 | ostream\_withassign | stderr |

**cout补充**  
1、用“cout<<”输出基本类型的数据时，可以不必考虑数据是什么类型，系统会判断数据的类型

      并根据其类型选择调用与之匹配的运算符重载函数。这个过程都是自动的，用户不必干预。

        如果在C语言中用prinf函数输出不同类型的数据，必须分别指定相应的输出格式符，十分麻烦，而且容易出错

2、cout流在内存中对应开辟了一个缓冲区，用来存放流中的数据，当向cout流插 人一个endl时，

     不论缓冲区是否已满，都立即输出流中所有数据，然后插入一个换行符， 并刷新流（清空缓冲区）。

        注意如果插人一个换行符”\n“（如cout<<a<<"\n"），则只输出和换行，而不刷新cout 流(但并不是所有编译系统都体现出这一区别）。

3、在iostream中只对"<<"和">>"运算符用于标准类型数据的输入输出进行了重载，但未对用户声明的类型数据的输入输出 进行重载。

    如果用户声明了新的类型，并希望用"<<"和">>"运算符对其进行输入输出，按照重运算符重载来做。

cout 流通常是传送到显示器输出，但也可以被重定向 输出到磁盘文件，而cerr流中的信息只能在显示器输出

cerr是不经过缓冲区，直接向显示器上输出有关信息，而clog中的信息存放在缓冲区中，缓冲区满后或遇endl时向显示器输出

### **3、标准输入流 cin**

    重点掌握的函数  
    cin.get()                 //读入一个字符并返回它的值  
    cin.get(一个参数)  /**/**读入一个字符并把它存储在ch   
    cin.get(两个参数)  //可以读取字符串  
    cin.get(三个参数)  //可以读字符串          
    cin.getline()  
    cin.ignore()           //读取字符并忽略指定字符  
    cin.peek()            //检查下一个输入的字符，不会把字符从流中移除  
    cin.putback()       //返回一个字符给一个流

**重要**  
1、使用cin，从流中读出的字符，流中就没有这字符了，再次读取时只能读取剩下的  
2、缓冲去只有在遇到EOF、手动敲回车、流（缓存区）满时，才将流中的字符全部读出（即清空缓存区）

**练习**  
1、从流中取一个字符，然后在放进去；  
2、判断流中的第一个字符是不是放进去的那个字符；  
3、从流中读取10个字符；  
4、从流中忽略5个字符，再读取10个字符；  
5、最后读取剩下的字符，最后输出读到的所有字符

#include <iostream>using namespace std;

int main()

{

char ch1;

int look;

char str1[11] = {0};

char str2[11] = {0};

char str3[100] = {0};

//从流中取一个字符，然后在放进去

ch1 = cin.get();

cin.putback(ch1);

//判断流中的第一个字符是不是放进去的那个字符

look = cin.peek();

if(look == (int)ch1)

{

cout<<"cin.peek()放进去的字符在第一个位置"<<endl;

}

else

{

cout<<"cin.peek()放进去的字符不是在第一个位置"<<endl;

}

//从流中读取10个字符

cin.get(str1,11);

//从流中忽略5个字符，再读取10个字符

cin.ignore(5);

cin.get(str2,11,EOF);

//最后读取剩下的字符，最后输出读到的所有字符

cin.getline(str3,100);

//输出读到的数据

cout<<"第一个字符"<<ch1<<endl;

cout<<"第一组 字符串："<<str1<<endl;

cout<<"第二组 字符串："<<str2<<endl;

cout<<"剩下的字符串："<<str3<<endl;

system("pause");

return 0;

}

输出：

0123456789abcde9876543210zzzzzzzzzzxxxxxxxxxxxxxyyyyyyyyyyyycin.peek()放进去的字符在第一个位置

第一个字符0

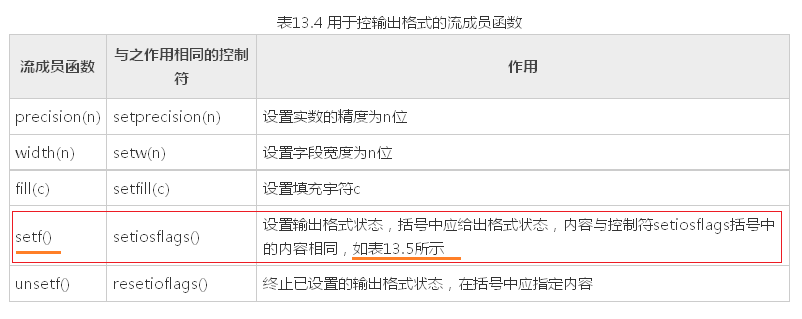
第一组 字符串：0123456789

第二组 字符串：9876543210

剩下的字符串：zzzzzzzzzzxxxxxxxxxxxxxyyyyyyyyyyyy

### **4、标准输出流 cout**

#### **流成员函数**



  cout.flush()      //清空缓冲区  
  cout.put()        //把字符写入流中  
  cout.write()      //将字符串写入当前输出流中



  eg : cout.setf( ios::dec );

         cout.setf(ios::hex,ios::basefield)；【建议使用这种】

  需注意：1、fmtflags setf( fmtflags flags ); 使用这种，一定要先取消当前基【cout.unself()】，之后才可以设置新的基

 2、fmtflags setf( fmtflags flags, fmtflags needed ); 使用这种，第二个参数设为当前的基，或者当不知道当前基时，设为ios\_base::basefield清除当前的所有可能的基

#### **控制符,头文件<iomanip>**

  
 eg:  cout<<setw(5);

### **5、文件I/O**

由于文件设备并不像显示器屏幕与键盘那样是标准默认设备，不能像cout那样**预先定义的全局对象**，所以我们必须**自己定义一个该类的对象。**

ifstream类，它是从istream类派生的，用来支持从磁盘文件的输入。

ofstream类，它是从ostream类派生的，用来支持向磁盘文件的输出。

fstream类，它是从iostream类派生的，用来支持对磁盘文件的输入输出。

#### **文件原理**

    文件打开都有一个**文件指针**，该指针的初始位置由I/O方式指定，每次读写都从文件指针的当前位置开始。每读入一个字节，指针就后移一个字节。当**文件指针移到最后**，就会遇到**文件结束EOF**（文件结束符也占一个字节，其值为-1)，此时流对象的**成员函数eof的值为非0值(一般设为1)，表示文件结束 了**。

    文件关闭，实际上是解除该磁盘文件与文件流的关联，原来设置的工作方式也失效，这样，就不能再通过文件流对该文件进行输入或输出

    文件类型：1、**ASCII文件**：文件的每一个字节中均以ASCII代码形式存放数据,即一个字节存放一个字符,这个文件就是ASCII文件(或称字符文件)。

     2、**二进制文件**：文件中的信息不是字符数据，而是字节中的二进制形式的信息，因此它又称为字节文件

#### **常用函数**

##### **打开文件：**



**方式1**：对文件输出流、文件流对象，通过ofstream类的构造函数打开文件

    格式： ofstream (磁盘文件名, **输入输出方式**);

    如果ofstream  为0(假)，表示打开操作失败

    如：   ofstream fout1(fname,ios::out);

    输入输出方式可以使用能够进行组合使用的，以“或”运算（“|”）的方式，如：fstream fout(fname,ios::out|ios::in)

**方式2**：对文件的输出和输入流对象、文件流对象，都可通过open函数打开文件

    格式： 文件流对象.open(磁盘文件名, **输入输出方式**);

    返回值：为0(假)，打开操作失败

    如：   fout.open(fname,ios::out)

##### **关闭文件：**

 在对已打开的磁盘文件的读写操作完成后，必须要关闭该文件  如：  outfile.close( );

##### **文件操作**

    1、可以用流插入运算符“<<”和流提取运算符“>>”输入输出标准类型的数据（>> 读出时在遇空格、换行都结束）。

    2、也可以用文件流的put、get、geiline等成员函数进行字符的输入输出。

#include <iostream>using namespace std;#include "fstream"

int main()

{

char fname[] = "d:/file1.txt";

char buff[1024]= {0};

/\*\*\*\*\*\*\*\*\*\*\* 写文件 \*\*\*\*\*\*\*\*\*\*\*\*\*/

//方式1 输出流ofstream对象调用fopen函数

ofstream fout;

fout.open(fname,ios::out);

if(!fout)

{

cout<<"打开文件失败"<<fname<<endl;

}

fout<< "hello world !"; //通过左移运算符写入字符串

fout.flush();

fout.close();

//方式2 调用输出流ofstream对象的构造函数

ofstream fout1(fname,ios::out);

if(!fout1)

{

cout<<"打开文件失败"<<fname<<endl;

}

fout1.put('h'); //通过put函数写入字符

fout1.put('e');

fout1.put('l');

fout1.put('l');

fout1.put('o');

fout1.put('\n');

fout1.flush();

fout1.close();

//文件流对象写文件

fstream file2(fname,ios::in|ios::out);

file2<<"abdfd\n";

file2<<"11111\n";

file2.flush();

file2.close();

/\*\*\*\*\*\*\*\*\*\*\* 读文件 \*\*\*\*\*\*\*\*\*\*\*\*\*/

//输入流ifstream对象读取文件内容

ifstream fin;

fin.open(fname,ios::in);

fin.getline(buff,1024); //通过getline函数读取字符串

cout<<buff<<endl;

fin.close();

//文件流对象读文件内容

fstream file1(fname,ios::in|ios::out);

file1>>buff; //通过右移运算符读出字符串

file1.close();

cout<<buff<<endl;

system("pause");

return 0;

}

#### **二进制文件操作**

    对二进制文件的读写主要用istream类的成员函数read和write来实现。这两个成员函数的原型为

     istream& read(char \*buffer,int len);

     ostream& write(const char \* buffer,int len);

#include <iostream>using namespace std;#include <fstream>

class Teacher

{public:

Teacher()

{

}

Teacher(int age,char name[20])

{

this->age = age;

strcpy(this->name,name);

}

void prinfInfo()

{

cout<<"Teacher name:"<<this->name<<" age:"<<this->age<<endl;

}private:

int age;

char name[20];

};

int main()

{

Teacher t1(31,"xiaoming");

Teacher t2(32,"xiaohong");

Teacher t3(33,"xiaohua");

Teacher t4(34,"xiaoxin");

char fname[] = "d:/file2";

fstream fs(fname,ios::binary|ios::out);

if(!fs)

{

cout<<"文件打开失败"<<endl;

}

fs.write((char \*)&t1,sizeof(Teacher));

fs.write((char \*)&t2,sizeof(Teacher));

fs.write((char \*)&t3,sizeof(Teacher));

fs.write((char \*)&t4,sizeof(Teacher));

fs.flush();

fs.close();

fstream fs2(fname,ios::binary|ios::in);

if(!fs)

{

cout<<"文件打开失败"<<endl;

}

Teacher tt;

fs2.read((char \*)&tt,sizeof(Teacher));

tt.prinfInfo();

fs2.read((char \*)&tt,sizeof(Teacher));

tt.prinfInfo();

fs2.read((char \*)&tt,sizeof(Teacher));

tt.prinfInfo();

fs2.read((char \*)&tt,sizeof(Teacher));

tt.prinfInfo();

fs2.close();

system("pause");

return 0;

}

输出：

Teacher name:xiaoming age:31Teacher name:xiaohong age:32Teacher name:xiaohua age:33Teacher name:xiaoxin age:34

http://my.oschina.net/u/1783725/blog/700901