第15讲 输入与输出（IO）

目录 CONTENTS

15.1 C++流类库

- 在C++中，数据的输入/输出(如从键盘读取数据、在显示器上显示数据、从文件读取数据、将数据写入文件等)可基于流类库完成。

15.1.1 C++的流

- “流”是指数据从一个地方到另一个地方的流动抽象。例如，将数据从键盘或文件读入内存时，称为“输入流”；将数据从内存输出到显示器或文件中时，称为“输出流”。
- 可以从流中获取数据，也可以向流中添加数据。从流中获取数据的操作称为“提取”操作，向流中添加数据的操作称为“插入”操作。

15.1.2 输入输出的头文件

C++用于输入输出的常用头文件有:

- iostream 包含了对输入输出流进行操作所需的基本信息。使用cin、cout等流对象进行针对标准设备的I/O操作时，须包含此头文件。
- fstream 用于用户管理文件的I/O操作。使用文件流对象进行针对磁盘文件的操作，须包含此头文件。

15.1.2 输入输出的头文件

- strstream 用于字符串流的I/O操作。使用字符串流对象进行针对内存字符串空间的I/O操作，须包含此头文件。

- iomanip 用于输入输出的格式控制。在使用setw、fixed等大多数操作符进行格式控制时，须包含此头文件。

15.1.3 输入输出流类

I/O流类库中包含了许多用于输入输出操作的类：

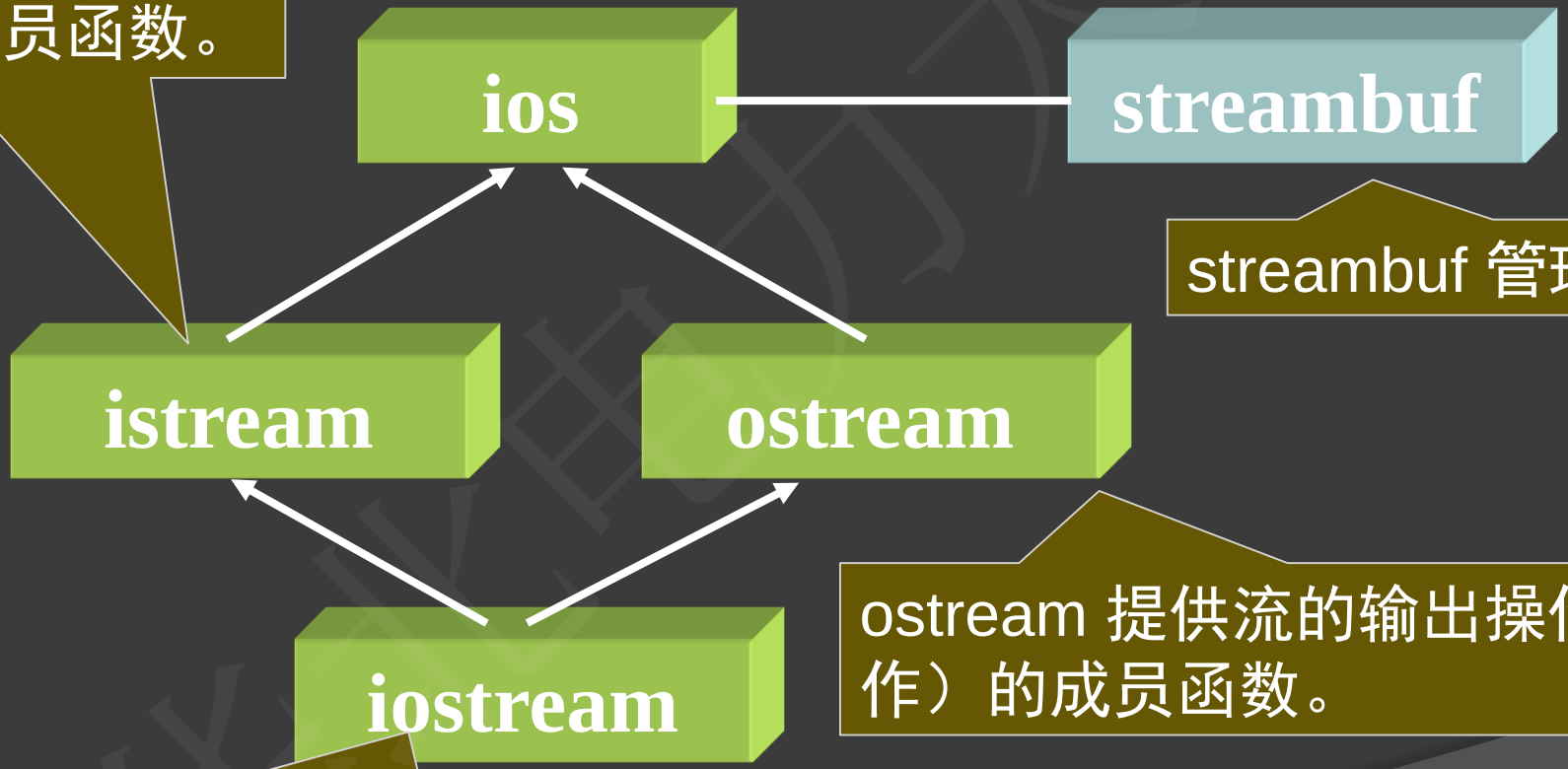
- ios是抽象基类，输入流类istream和输出流类ostream是通过单继承从基类ios派生而来的；

- 输入输出流类iostream是通过多继承从类istream和ostream派生而来的。

15.1.4 流类库的基本结构

ios是个虚基类。提供流的格式化输入/输出操作成员函数和错误处理成员函数。

istream 提供流的输入操作（提取操作）的成员函数。



streambuf 管理流的缓冲区。

ostream 提供流的输出操作（插入操作）的成员函数。

iostream 支持对同一个流同时进行输入/输出操作（提取操作和插入操作）。

15.1.5 支持文件的流类

单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

下面哪个流类不能用于文本文件的输入或输出

A ifstream

B ofstream

C fstream

D istream

提交

15.2 预定义的流对象

- 用流类定义的对象称为流对象。
  - 与输入设备(如键盘)相关联的流(流对象)称为输入流(流对象);
  - 与输出设备(如屏幕)相关联的流(流对象)称为输出流(流对象);
  - 与输入输出设备(如磁盘)相联系的流(流对象)称为输入流出流(流对象)。

15.2 预定义的流对象

- C++中包含几个预定义的流(流对象)，它们是：
  - 标准输入流 (流对象) cin
  - 标准输出流(流对象) cout
  - 非缓冲型的标准出错流 (流对象) cerr
  - 缓冲型的标准出错流 (流对象) clog

15.2 预定义的流对象

(1) cin----标准输入流对象, 与标准输入设备相联系（键盘）。

例如: cin>>变量名;

“>>”为提取运算符(输入运算符)，表示从键盘读取数据放入变量中。

(2) cout---标准输出流(流对象), 与标准输出设备相联系（显示器）。

例如: cout<<“数据”；

“<<”为插入运算符(输出运算符)，表示将“数据”写到显示器上。

15.2 预定义的流对象

(3) cerr---非缓冲型的标准出错流对象，与标准输出设备相联系（显示器）。

(4) clog---缓冲型的标准出错流，与标准输出设备相联系（显示器）。

- cerr与clog均用来输出出错信息。
- cerr和clog之间的区别是：
  - cerr是不经过缓冲区，直接向显示器上输出有关信息，因而发送给它的任何内容都立即输出;
  - clog中的信息存放在缓冲区中，缓冲区满后或遇上endl时向显示器输出。

只要在程序中包含头文件iostream,C++程序开始运行时这四个标准流对象的构造函数都被自动调用。

15.3 文件流类

- 可通过C++的文件流类ifstream、ofstream和fstream进行文本文件的读取和保存。

15.3.1 文件的打开与关闭

- 所谓“文件”,一般指存放在外部介质上的数据的集合。
- 一批数据(可以是一段程序、一批实验数据,或者是一篇文章、一幅图像、一段音乐等)是以文件的形式存放在外部介质(如磁盘、光盘、U盘)上的。
- 从操作系统的角度来说,每一个与主机相联的输入输出设备都可以看出是一个文件。例如: 键盘是输入文件，显示器是输出文件。

15.3.2 文件分类

根据文件中数据的组织形式:

文件 { 文本文件
二进制文件

- 文本文件又称ASCII文件,它的每个字节存放一个ASCII代码,代表一个字符。
- 二进制文件则是把内存中的数据,按其在内存中的存储形式原样写到磁盘上存放。

15.3.3 文本文件与二进制文件

- 用文本形式输出时，一个字节对应一个字符，因而便于对字符进行逐个处理，也便于输出字符，缺点是占存贮空间较多。
- 用二进制形式输出数据，可以节省存贮空间和转换时间，但一个字节不能对应一个字符，不能直接以字符形式输出。
- 对于需要暂时保存在外存上，以后又需要输入到内存的中间结果数据，通常用二进制形式保存。

15.3.4 操作文件的步骤

(1)为要进行操作的文件定义一个流对象。

(2)建立(或打开)文件。如果文件不存在，则建立该文件。如果磁盘上已存在该文件，则打开它。

(3)进行读写操作。在建立(或打开)的文件基础上执行所要求的输入或输出操作。

(4)关闭文件。当完成输入输出操作时，应把已打开的文件关闭。

15.3.5 建立流对象

在C++中，打开一个文件，就是将这个文件与一个流对象建立关联；关闭一个文件，就是取消这种关联。

C++提供了以下三种类型的文件流类：

ofstream	输出文件流类
ifstream	输入文件流类
fstream	输入输出文件流类

这些文件流类都定义在fstream文件中。

15.3.6 打开文件

使用函数open()打开文件，也就是使某一指定的磁盘文件与某一已定义的文件流对象建立关联。

调用成员函数open的一般形式为:

```
文件流对象.open(文件名, 打开方式);
```

可以包括路径(如“d:\c++\test.dat”),如缺省路径,则默认为当前目录下的文件

决定文件将如何被打开

文件的打开方式

方式	功能
ios::app	打开一个已存在的文件，以将数据添加到文件的尾部。这种方式打开的文件只能用于输出
ios::ate	打开一个已存文件，并将文件指针移到文件的尾部
ios::binary	以二进制的形式打开文件
ios::in	打开一个文件，进行文件输入操作
ios::out	打开一个文件，进行文件输出操作
ios::trunc	打开一个文件，如果该文件已存在，则清除文件的内容，文件的长度变为零

关于打开文件的说明

例如:

```
ofstream out;
out.open("test.dat",ios::out);
```

定义类 ofstream的对象out

打开一个输出文件test.dat

（1）文件使用方式有默认值，对于类ifstream，默认值为ios::in；对于类ofstream，默认值为ios::out。因此，上述语句通常可写成:

```
ofstream out;
out.open("test.dat");
```

关于打开文件的说明

(2) 当一个文件需要用两种或多种方式打开时，可以用“位或”操作符(即“|”)把几种方式组合在一起。

关于打开文件的说明

例如:

```
fstream myst;
myst.open("test.dat",ios::in | ios::out|ios::binary);
```

其他例子:

```
ios::in|ios:out            //以输入和输出方式打开文件,
ios::out|ios:binary        //以二进制方式打开一个输出文件
ios::in|ios::binary        //以二进制方式打开一个输入文件
ios::in|ios::nocreate      //打开一个输入文件，若文件不存在，则返回打开失败的信息
ios::app|ios::nocreate
      //打开一个输出文件，在文件尾接着写数据，若文件不存在，则返回打开失败的信息
```

关于打开文件的说明

模式标记	适用对象	作用
ios::in	ifstream fstream	打开文件用于读取数据。如果文件不存在，则打开出错。
ios::out	ofstream fstream	打开文件用于写入数据。如果文件不存在，则新建该文件；如果文件原来就存在，则打开时清除原来的内容。
ios::app	ofstream fstream	打开文件，用于在其尾部添加数据。如果文件不存在，则新建该文件。
ios::ate	ifstream	打开一个已有的文件，并将文件读指针指向文件末尾（读写指 的概念后面解释）。如果文件不存在，则打开出错。
ios:: trunc	ofstream	打开文件时会清空内部存储的所有数据，单独使用时与 ios::out 相同。
ios::binary	ifstream ofstream fstream	以二进制方式打开文件。若不指定此模式，则以文本模式打开。
ios::in | ios::out	fstream	打开已存在的文件，既可读取其内容，也可向其写入数据。文件刚打开时，原有内容保持不变。如果文件不存在，则打开出错。
ios::in | ios::out	ofstream	打开已存在的文件，可以向其写入数据。文件刚打开时，原有内容保持不变。如果文件不存在，则打开出错。
ios::in | ios::out | ios::trunc	fstream	打开文件，既可读取其内容，也可向其写入数据。如果文件本来就存在，则打开时清除原来的内容；如果文件不存在，则新建该文件。

关于打开文件的说明

(3)打开文件的另一种方法，　例如:

```
ofstream out("test.dat");
```

相当于:

```
ofstream out;
out.open("test.dat");
```

如果文件打开操作失败，则与文件相联系的流对象的值为0。

关于打开文件的说明

(4) 通常都要测试打开文件是否成功。可以使用类似下面的方法进行检测：

```
if (!out.fail())
  {
     cout<<"Cannot open";
     //错误处理代码
  }
```

```
if (!out)
  {
     cout<<"Cannot open";
     //错误处理代码
  }
```

```
if (!out.is_open())
  {
     cout<<"Cannot open";
     //错误处理代码
  }
```

15.3.7 关闭文件

- 输入输出操作完成后，应该将文件关闭。所谓关闭，实际上就是将所打开的磁盘文件与流对象“脱钩”。
- 关闭文件可使用close()函数完成，close()函数也是流类中的成员函数，它不带参数。

15.3.7 关闭文件

例如:

```
ofstream out;          //建立输出流对象out
out.open("test.dat"); //流对象out与test.dat建立了关联
. . .
out.close();     //将与流对象out所关联的磁盘文件test.dat关闭
```

15.3.8 文件读写

- 文本文件读写，一般使用<<和>>，当读取文件时，还可以使用getline()方法，但需要配合eof()方法。

- 二进制文件读写，<<和>>也符合语法，但使用时没有意义，一般使用write和read方法。
  - write (char * buffer, streamsize size);
  - read (char * buffer, streamsize size);

15.3.8 文本文件的读写

例1 把基本数据类型写入磁盘文件test1.dat中。

```
#include <iostream> #include <fstream> using namespace std;
 int main()
 {
     ofstream fout1("c:\\test1.dat",ios::out);
     if(!fout1.is_open())            //如果文件打开失败
    {  cout<<"Cannot open output file.\n";
       exit(1);
    }
    fout1<<123<<" "<<7.8;  //把int和double写到test1.dat中
    fout1.close();     //将与fout1所关联的输入文件test1.dat关闭
    return 0;
 }
```

15.3.8 文本文件的读写

例2 把磁盘文件test1.dat中的内容读出并显示在屏幕上。

```
#include <iostream>   #include <fstream>   using namespace std;
int main()
{  ifstream  fin1("test1.dat",ios::in);
   if(!fin1.is_open())                          //如果文件打开失败
    {  cout<<"Cannot open output file.\n";
      exit(1);
   }
   int i; double d;
   fin1>>i>>d;          //从test1.dat读入整型和double
   cout<<i<<“ ”<<d<<endl;              //屏幕上显示出str的值
   fin1.close();                    //将与fin1所关联的输入文件test1.dat关闭
   return 0;
}
```

15.3.8 文本文件的读写

例3 把字符串“I am a student.”写入磁盘文件test1.dat中。

```
#include <iostream> #include <fstream> using namespace std;
 int main()
 {
    ofstream fout1(“c:\\test1.dat",ios::out);
     if(!fout1.is_open())            //如果文件打开失败
    {  cout<<"Cannot open output file.\n";
       exit(1);
    }
    fout1<<"I am a student.";  //把一个字符串写到test1.dat中
    fout1.close();     //将与fout1所关联的输入文件test1.dat关闭
    return 0;
 }
```

15.3.8 文本文件的读写

例4 把磁盘文件test1.dat中的内容读出并显示在屏幕上。

```
#include <iostream>   #include <fstream>   using namespace std;
int main()
{  ifstream  fin1("test1.dat",ios::in);
   if(!fin1.is_open())                          //如果文件打开失败
   {  cout<<"Cannot open output file.\n";
      exit(1);
   }
   char str[80];
   fin1.getline(str,80);          //从test1.dat读入字符串赋给字符数组str
   cout<<str<<endl;               //屏幕上显示出str的值
   fin1.close();                   //将与fin1所关联的输入文件test1.dat关闭
   return 0;
}
```

getline()函数

fs.getline(str, 256); //从文件中读取信息

函数getline()是basic_istream的成员函数，其函数声明是：
basic_istream& getline(char_type *_Str, streamsize _Count);

其含义是：从文件流中读取_Count个字节到_Str所指的内存中，如果还没有读取_Count个字节就遇到回车符'\n'，则读取过程中止。

15.3.8 文本文件的读写

例4 把磁盘文件test1.dat中的内容读出并显示在屏幕上。

```
#include <iostream>   #include <fstream>   using namespace std;
int main()
{  ifstream  fin1("test1.dat",ios::in);
   if(!fin1.is_open())                          //如果文件打开失败
   {  cout<<"Cannot open output file.\n";
      exit(1);
   }
   string str;
   fin1>>str;                          //从test1.dat读入字符串赋给字符串str
   cout<<str<<endl;                    //屏幕上显示出str的值
   fin1.close();                        //将与fin1所关联的输入文件test1.dat关闭
   return 0;
}
```

15.3.9 二进制文件的读写

- 任何文件，都能以文本方式或二进制方式打开。在缺省情况下，文件用文本方式打开。

- 文本方式和二进制方式主要的区别是:
  - 在文本方式下输入时，回车'\r'和换行'\n'两个字符要转换为字符'\n'。
  - 在输出时，字符'\n'转换为回车'\r'和换行'\n'两个字符。
  - 这些转换在二进制方式下是不进行的。

15.3.9 二进制文件的读写

- 对二进制文件进行读写有两种方式：
  (1) 使用的是函数get和put
  (2) 使用的是函数read和write。
- 这四种函数也可以用于文本文件的读写。
- 除字符转换方面略有差别外,文本文件的处理过程与二进制文件的处理过程基本相同。

用get和put函数读写二进制文件

- get函数是输入流类istream中定义的成员函数，它可以从与流对象连接的文件中读出数据，每次读出一个字节(字符)。
- put函数是输出流类ostream中的成员函数，它可以向与流对象连接的文件中写入数据，每次写入一个字节(字符)。

例5 将26个英文字母写入文件, 并读取显示出来。

```
#include<iostream> #include<fstream>using namespace std;
int test_write()
{
   ofstream fout2(“bin.dat”,ios::binary);
   if (!fout2.is_open())        //如果文件打开失败
   {   cout<<“Cannot open output file\n,”;
       exit(1);    }
   char ch='a';
   for (int i=0;i<26;i++)
   {   fout2.put(ch);
       ch++;          }
  fout2.close();
  return 0;
}
```

定义输出文件流对象fout2，打开二进制输出文件bin.dat

```
int test_read()
{
   ifstream fin2(“bin.dat", ios::binary);
   if (!fin2.is_open())          //如果文件打开失败
   {     cout<<"Cannot open input file\n,";
         exit(1);     }
   char ch;
   while( fin2.get(ch) )
         cout<<ch;
   fin2.close();
   return 0;
}
int main()
{
    test_write();
    test_read();
    return 0;
}
```

检测流对象是否为零，为零表示文件结束

程序运行结果如下：

abcdefghijklmnopqrstuvwxyz

用read和write函数读写二进制文件

fin是输入文件流对象

参数len:要读入的数据的字节数

- C++提供了两个函数read和write，用来读写一个数据块，read函数最常用的调用格式如下:

```
fin.read(char *buf, int len)
```

- 功能：从与输入文件流对象fin相关联的磁盘文件中，读取len个字节(或遇EOF结束)，并把它们存放在字符指针buf所指的一段内存空间内。如果在len个字节(字符)被读出之前就达到了文件尾，则read函数停止执行。

参数buf：指向读入数据所存放的内存空间的起始地址;

用read和write函数读写二进制文件

参数buf:指向读入数据所存放的内存空间的起始地址

参数len:要写出数据的字节数

- write函数最常用的调用格式如下:

```
fout.write(const char *buf,int len)
```

- 功能：将字符指针buf所给出的地址开始的len个字节的内容不加转换地写到与输出文件流对象fout相关联的磁盘文件中。
- 注意：第1个参数的数据类型为 char*，如果是其他类型的数据，必须进行类型转换。例如:

```
int array[]={50,60,70};
write((char*) array, sizeof (array));
```

```
write(reinterpret_cast<char*>array, sizeof(array));
```

例6 将课程及成绩信息写入文件, 并读取显示出来。

```
#include<iostream> #include<fstream>using namespace std;
class info
{
public:
        info(char *c, int s)
        {
                strcpy(course, c);
                score = s;      }
        info()
        {       }
private:
        char course[15];
        int score;
};
```

```
int main()
{
   info student[2]={info("Computer",90),
                    info("Mathematics",78)};
   ofstream fout3("test.dat",ios::binary);
   if (!fout3.is_open())            //如果文件打开失败
   { cout<<"Cannot open output file.\n";
     exit(1);
   }          //退出程序,其作用与exit相同
   for (int i=0;i<2;i++)
      fout3.write(reinterpret_cast<char*>(&student[i]),sizeof(student[i]));
   fout3.close();
```

```
    info s[2];
    ifstream fin3("test.dat",ios::binary);
    if (!fin3.is_open())            //如果文件打开失败
    {    cout<<"Cannot open input file.\n";
       exit(1);   }
    for (int i=0;i<2;i++)
    {
       fin3.read(reinterpret_cast<char*>(&s[i]),sizeof(s[i]));
       cout<<s[i].course<<" "<<s[i].score<<endl;
    }
    fin3.close();

    return 0;
}
```