第十章 简单输入输出

目录

学习目标

1. 了解常用 IO 类的继承关系和理解 IO 流基本工作流程；
2. 掌握常见的输入输出格式控制；
3. 掌握文件流和 string 流的使用方法。

概念　代码　说明　问题/答案　注意

10.1 基本知识

C++ 的 IO 操作

C++ 语言不能直接处理 IO 操作，依靠不同的 IO 类来实现从设备中读取数据和向设备写入数据。

例如之前用到的： cin、 cout、 >> 。

10.1.1 IO 类对象

流 (stream)

数据从数据源到目的端的流动过程称之为流。

常用的 IO 类

常用的 IO 类有 itream 和 ostream ，同时包括文件流类型和 string 流类型。

ios 是抽象基类，由它派生出 istream 类和 ostream 类，两个类名中第 1 个字母 i 和 o 分别代表输入（input）和输出（output）。ifstream 和 ofstream 类用于文件输入输出，类名中第 2 个字母 f 代表文件（file）。

类型 ifstream 和 istringstream 继承自 istream，类型 ofstream 和 ostringstream 继承自 ostream。

10.1.1 IO 类对象

常用 IO 类库简介：

头文件	类名	功能
iostream	ios	抽象基类
iostream	istream	通用输入流和其他输入流的基类
	ostream	通用输出流和其他输出流的基类
	iostream	通用输入输出流和其他输入输出流的基类
fstream	ifstream	输入文件流类
	ofstream	输出文件流类
	fstream	输入输出文件流类
sstream	istringstream	输入字符串流类
	ostringstream	输出字符串流类
	stringstream	输入输出字符串流类

10.1.1 IO 类对象

常用对象 cin ， cout ：

cin 是 istream 类的对象，它从标准输入设备 (键盘) 获取数据，通过输入运算符 >> 从流中提取数据，提取的时候会根据对象的类型从输入流中提取相应长度的字节，cin>> 从流中提取数据时通常跳过输入流中的空格、制表符、换行符等空白字符。

cout 是 ostream 类对象，它向控制台窗口输出数据。

和普通对象的区别：

和普通对象不同，IO 对象不支持赋值操作。

10.1.1 IO 类对象

常用对象 cin ， cout ：

cin 是 istream 类的对象，它从标准输入设备 (键盘) 获取数据，通过输入运算符 >> 从流中提取数据，提取的时候会根据对象的类型从输入流中提取相应长度的字节，cin>> 从流中提取数据时通常跳过输入流中的空格、制表符、换行符等空白字符。

cout 是 ostream 类对象，它向控制台窗口输出数据。

和普通对象的区别：

和普通对象不同，IO 对象不支持赋值操作。

示例：

```
ifstream in1, in2;//定义两个文件输入流对象
in1 = in2; //错误: 不能对流对象赋值
//同样, IO对象也不支持复制操作:
ostream print(ostream);//错误: 不能按值方式返回或传递ostream对象
```

10.1.2 状态条件

请看如下情况：

```
double x;
cin>>x;
```

10.1.2 状态条件

请看如下情况：

```
double x;
cin>>x;
```

当输入为 char 类型的字符，会怎么样呢?

10.1.2 状态条件

请看如下情况：

```
double x;
cin>>x;
```

当输入为 char 类型的字符，会怎么样呢?

代码中的 cin>> 期待读取一个 double，但却提供了字符数据，cin 会进入错误状态。一旦 cin 进入到错误状态，它就变成无效的，无法再执行后续的输入。因此，在使用 cin 时，要确保它的状态是有效的。

10.1.2 状态条件

那么又该如何来判断状态的有效性呢?

10.1.2 状态条件

那么又该如何来判断状态的有效性呢?

```
while(cin>>x)//遇到错误状态循环将退出;
if(!cin)cin.clear();//clear()函数执行后，cin变为有效状态;
```

10.1.3 刷新缓冲区

缓冲区刷新的原因：

导致缓冲区刷新有很多原因，比如缓冲区满、程序正常结束、遇到 endl 等。缓冲区刷新完成后，原来的数据被清空。

示例：

```
cout<<"endl"<<endl;//输出endl和一个换行，然后刷新缓冲区
cout<<"flush"<<flush;//输出flush（无额外字符），然后刷新缓冲区
cout<<"ends"<<ends;//输出ends和一个空字符，然后刷新缓冲区
```

10.2 标准输入输出

输入输出的控制是 C++ 程序当中最常用的一些操作，字符数据的格式化控制尤为重要。

10.2.1 字符数据的输入

使用 » 运算符：

数据的输入一般以空白字符结束（包括空格符、制表符和回车符等），而这些空白字符会被系统过滤掉。

使用 cin.get() 函数：

cin.get() 函数可以获取空白字符。其功能是从输入流中获取一个字符，并将其返回。

示例：

```
for(char c;(c=cin.get())!='\n';)
   cout<<c;
cout<<endl;
```

10.2.1 字符数据的输入

使用 cin.getline() 函数：

getline 函数以回车符作为输入结束的标志符，把从输入流 cin 中提取的字符序列（不包括回车符）放到 string 类对象 s 中，并返回 cin 的引用。

示例：

```
string s;
getline(cin,s);
```

10.2.2 格式化控制

整形值的进制

默认格式按照十进制输入输出，其他进制可以用进制说明符进行转换。

示例:

```
cout<<showbase<<uppercase;//显示进制信息，十六进制数以大写形式输出
cout<<"default:"<<26<<endl;
cout<<"octal:"<<oct<<26<<endl;
cout<<"decimal:"<<dec<<26<<endl;
cout<<"hex:"<<hex<<26<<endl;
cout<<noshowbase<<nouppercase<<dec;
int i,j;
cin>>oct>>i;//输入格式为八进制;
cin>>hex>>j;//输入格式为十六进制;
```

10.2.2 格式化控制

整形值的进制

默认格式按照十进制输入输出，其他进制可以用进制说明符进行转换。

示例:

```
cout<<showbase<<uppercase;//显示进制信息，十六进制数以大写形式输出
cout<<"default:"<<26<<endl;
cout<<"octal:"<<oct<<26<<endl;
cout<<"decimal:"<<dec<<26<<endl;
cout<<"hex:"<<hex<<26<<endl;
cout<<noshowbase<<nouppercase<<dec;
int i,j;
cin>>oct>>i;//输入格式为八进制;
cin>>hex>>j;//输入格式为十六进制;
```

注意:

上面最后一条语句执行完之后，后续输入的数据为十六进制，用户可以利用 dec 将进制形式恢复为默认的十进制。

10.2.2 格式化控制

控制打印精度:

```
double x=1.2152;
cout.precision(3);//使用precision成员函数指定打印精度;
cout<<"precision:"<<cout.precision()<<",x="<<x<<endl;
cout<<setprecision(4);//使用setprecision函数指定打印精度;
cout<<"precision:"<<cout.precision()<<",x="<<x<<endl;
cout<<"scientific:"<<scientific<<10*exp(1.0)<<endl;//用科学计数法控制输
  出格式;
cout<<"fixed␣decimal:"<<fixed<<10*exp(1.0)<<endl;//定点十进制默认格式;
cout<<"default␣float:"<<defaultfloat<<10*exp(1.0)<<endl;
```

10.2.2 格式化控制

控制打印精度:

```
double x=1.2152;
cout.precision(3);//使用precision成员函数指定打印精度;
cout<<"precision:"<<cout.precision()<<",x="<<x<<endl;
cout<<setprecision(4);//使用setprecision函数指定打印精度;
cout<<"precision:"<<cout.precision()<<",x="<<x<<endl;
cout<<"scientific:"<<scientific<<10*exp(1.0)<<endl;//用科学计数法控制输
  出格式;
cout<<"fixed decimal:"<<fixed<<10*exp(1.0)<<endl;//定点十进制默认格式;
cout<<"defaul  at:"<<defaultfloat<<10*exp(1.0)<<endl;
```

注意:

在执行 scientific 或者 fixed 操纵符后，精度控制的是小数点后面的数值位数，而不是默认的数值总位数。defaultfloat 为 C++11 新特性，它将流恢复到默认状态。

10.2.2 格式化控制

利用 setw 指定占用宽度

```
int i=-10;
double x=1.2152;
cout<<"i:"<setw(10)<<i<<endl;
cout<<"x:"<<setw(10)<<x<<endl;
```

输出结果:

```
i:       -10
x:      1.2152
```

10.3 文件输入输出与 string 流

文件流：

和磁盘进行数据交换需要文件流，其中文件流包括： iftream，ofstream，fstream， 它们分别可以从指定文件读取数据，向指定文件写入数据和读写文件。

10.3.1 使用文件流对象

文件流对象的创建和关联：

```
ifstream in(ifname);//创建输入文件流对象，提供文件名;
ofstream out;//创建输出文件流对象，没有提供文件名;
```

10.3.1 使用文件流对象

文件流对象的创建和关联:

```
ifstream in(ifname);//创建输入文件流对象，提供文件名;
ofstream out;//创建输出文件流对象，没有提供文件名;
```

文件流的打开与关闭:

```
out.open(name);//调用open 函数，使之与一个文件关联;
if(out);//用于检测open操作是否成功;
out.close();//调用close函数关闭文件;
```

10.3.2 文件模式

每个文件都有一些文件模式，用来指定如何使用文件。

常用的文件模式

- ios::in 以读方式打开文件；
- ios::out 以写方式打开文件（默认方式）。如果已有此文件，则将其原有内容全部擦除，如文件不存在，则建立新文件；
- ios::app 以写方式打开文件，写入的数据追加到文件末尾；
- ios::ate 打开一个已有的文件，并定位到文件末尾；
- ios::binary 以二进制方式打开一个文件，如不指定此方式则默认为 ASCII 方式。

10.3.2 文件模式

每个文件都有一些文件模式，用来指定如何使用文件。

常用的文件模式

- ios::in 以读方式打开文件；
- ios::out 以写方式打开文件（默认方式）。如果已有此文件，则将其原有内容全部擦除，如文件不存在，则建立新文件；
- ios::app 以写方式打开文件，写入的数据追加到文件末尾；
- ios::ate 打开一个已有的文件，并定位到文件末尾；
- ios::binary 以二进制方式打开一个文件，如不指定此方式则默认为 ASCII 方式。

说明：

每一个文件流类型都设置了一个默认的文件模式，如果没有指定具体的文件模式，则以默认模式打开。ios::in 是 ifstream 流的默认模式，ios::out 是 ofstream 流的默认模式。fstream 的默认模式为 ios::in 和 ios::out。

10.3.2 文件模式

文件读取示例：将百鸡问题中结果保存，然后读出计算结果并且打印输出。

```
#include<iostream>
#include<string>
#include<iomanip>//使用setw函数
#include<fstream>//文件输入输出
using namespace std;
int main() {
int max_rst = 100 / 5, max_hen = 100 / 3;
ofstream out("result.txt");//在当前目录创建文件;
if (out) { //判断文件是否成功打开;
        out <<setw(10)<<"公鸡"<<setw(10)<<"母鸡"<<setw(10)<< "小鸡";
        for (int i = 0; i < max_rst; ++i) {
            for (int j = 0; j < max_hen; ++j) {
                int k = 100 - i - j;
                if (k % 3) continue;
                if (5 * i + 3 * j + k / 3 == 100)//向文件写入数据;
                    out<<'\n'<<setw(10)<<i<<setw(10)<<j<<setw(10)<<k;
            }
        }
       out.close();//关闭文件;
}
```

10.3.2 文件模式

文件读取示例：将百鸡问题中结果保存，然后读出计算结果并且打印输出。(续)

```
ifstream in("result.txt");//打开当前目录下的文件;
if (in) {//判断文件是否成功打开;
    string head;
    getline(in, head);
    cout << head << endl;
    int r[3];
    while (!in.eof()) {//成员函数eof用来判读文件流是否结束;
        in>>r[0]>>r[1]>>r[2];//从文件读取数据;
        cout<<setw(10)<<r[0]<<setw(10)<<r[1]<<setw(10)<<r[2]<<endl;
   }
   in.close();//关闭文件;
}
return 0;
}
```

10.3.2 文件模式

文件读取示例：将百鸡问题中结果保存，然后读出计算结果并且打印输出。(续)

```
ifstream in("result.txt");//打开当前目录下的文件;
if (in) {//判断文件是否成功打开;
    string head;
    getline(in, head);
    cout << head << endl;
    int r[3];
    while (!in.eof()) {//成员函数eof用来判读文件流是否结束;
        in>>r[0]>>r[1]>>r[2];//从文件读取数据;
        cout<<setw(10)<<r[0]<<setw(10)<<r[1]<<setw(10)<<r[2]<<endl;
   }
   in.close();//关闭文件;
}
return 0;
}
```

说明:

在打开文件时，可以指定文件的具体路径，例如 “d:/result.txt”；如缺省路径，则默认为当前目录下的文件

string 流

string 流可以向 string 类对象写入数据，也可以从 string 类对象读取数据。string 流定义在 sstream 头文件中，它包含三个类型：istringstream、ostringstream 和 stringstream。

string 流

- istringstream　从 string 对象读取数据；
- ostringstream　向 string 对象写入数据；
- stringstream　既可以从 string 对象读取数据也可以向 string 对象写入数据。

istringstream 流

当从设备读取一行文本时，往往需要对整行文本中的单个单词进行处理，这时可以使用 istringstream 流对象。比如，需要获取一行文本中的所有单词，并把它们存放到一个 vector 里面。

istringstream 流

当从设备读取一行文本时，往往需要对整行文本中的单个单词进行处理，这时可以使用 istringstream 流对象。比如，需要获取一行文本中的所有单词，并把它们存放到一个 vector 里面。

使用示例：

```
vector<string>wds;//保存读取的单词;
string line,word;
while(getlien(cin,line)){
    istringstream iss(line);//创建输入的string流对象，保存line的副
      本;
    while(iss>>word)
        wds.push_back(word);//将读取到的单词尾插;
}
```

ostringstream 流

当需要一次打印不同数据类型的数据时，使用 ostringstream 流可以很容易实现。比如，在上一节的例子中，在获取所有单词之后，一次性输出每个单词和他们的长度。

ostringstream 流

当需要一次打印不同数据类型的数据时，使用 ostringstream 流可以很容易实现。比如，在上一节的例子中，在获取所有单词之后，一次性输出每个单词和他们的长度。

使用示例：

```
ostringstream out;//创建流对象;
for(auto &i;wds)
   out<<i<<":"<<i.lengthe()<<'\n';//处理单词;
cout<<out.str();
```

ostringstream 流

当需要一次打印不同数据类型的数据时，使用 ostringstream 流可以很容易实现。比如，在上一节的例子中，在获取所有单词之后，一次性输出每个单词和他们的长度。

使用示例：

```
ostringstream out;//创建流对象;
for(auto &i;wds)
   out<<i<<":"<<i.lengthe()<<'\n';//处理单词;
cout<<out.str();
```

注意：

注意，ostringstream 的另外一个版本的成员函数 str 接受一个 string 类型的参数，用来覆盖原有的数据，例如：

如下所示：

```
out.str("");//清空原有数据,调用此函数时，out~里面的数据将被清空。
```

本章结束