东南大学吴健雄学院
CHIEN-SHIUNG WU COLLEGE OF SEU

C++程序设计

——流类库与输入输出

内容

C++中流的概念及流类体系

标准设备的输入输出

文件的输入输出

文件与对象

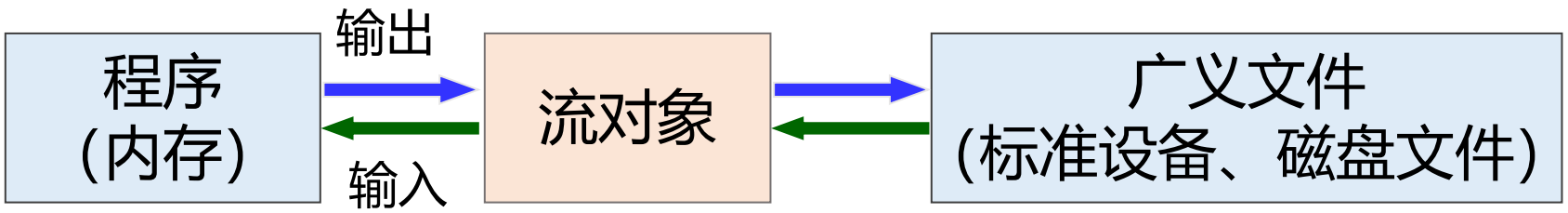
C++中流的概念及流类体系

流（stream) ——为什么用“流”

现象：数据从一个对象流动到另一个对象。 cin >> x >> y; file << val;

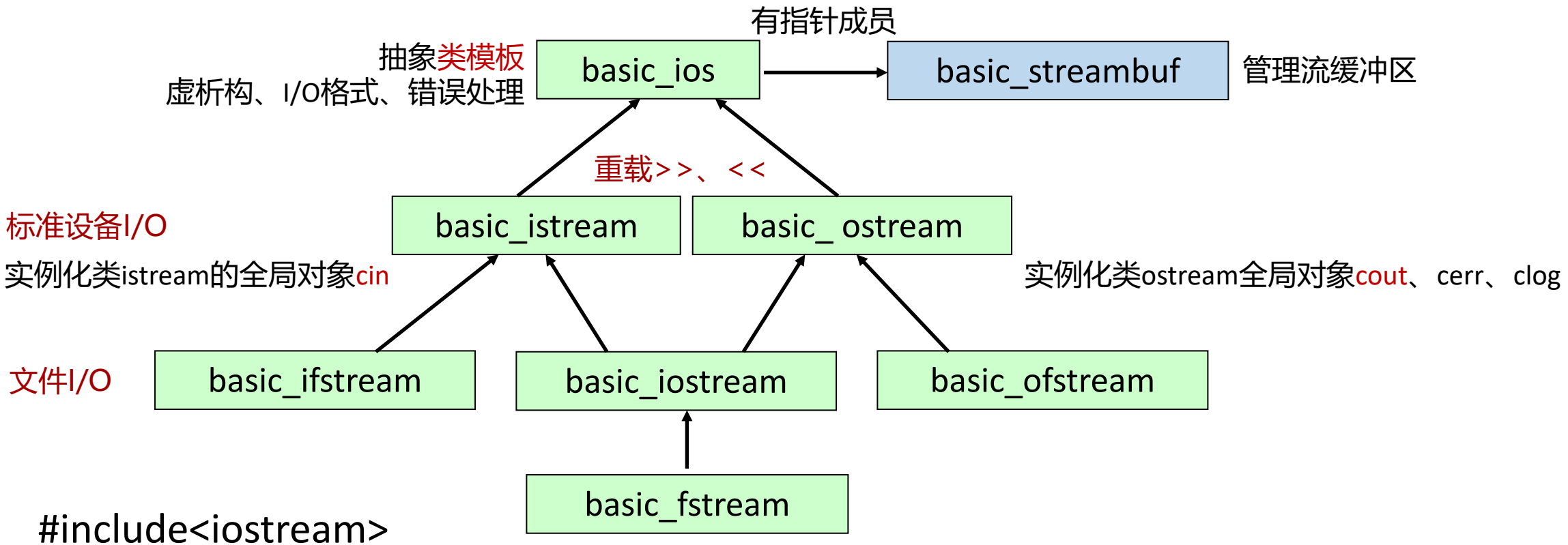
抽象：封装专门处理数据流动的类，称“流类”，提供接口。

输入输出流类库：流类形成派生体系，有一系列的类定义。



编程者使用流提供的方法

C++的流类体系（了解）



#include<iostream>

自学了解basic_ios提供的格式化输入输出控制及一般使用方法。9.2*

标准设备的输入输出*

IO健壮性

basic_ios中定义了流操作状态字及针对状态字的操作函数，用以监控输入输出状态。

istream中的部分成员

```
int istream :: get();                          //提取一个字符，包括空格等，若干重载函数
istream& istream : :getline(char*, int, char = '\n');    //提取字符串，有重载函数
int istream :: gcount();                       //最后一次提取的字符数（包括回车）
istream& istream :: ignore( int = 1, int = EOF);          //用于清空缓冲区
```

ostream中的部分成员

```
ostream&  ostream :: put(char);        //输出参数字符，有若干重载函数
```

重载<<、>>运算符 —— 实现对象的输入输出

以复数类为例:

Complex c1, c2;　要实现cin >> c1 >> c2;　　cout << c1 << “ “ << c2 << endl;

```
class Complex
{     double re, im;
public:
      complex(int = 0, int = 0);          //构造
      friend istream& operator >>(istream &is, complex &c)          //重载 > >
      {     is >> c.re >> c.im;      return is;    }
      friend ostream& operator <<(ostream &os, complex &c)      //重载 < <
      {    os << c.re << “+i” << c.im;     return os;    }
};
```

为什么只能友元重载？为什么返回流对象？

文件的输入输出

文件操作回顾

文件（标准设备、磁盘文件）分类

- 文本文件（字符文件） —— 内容可见
- 二进制文件（数据文件） —— 内容不可见

文件操作三部曲

- 打开：建立文件流对象与实际磁盘文件的关联，有多种打开方式
- 读/写：程序与流对象之间读写，文本文件与二进制文件读写操作有区别
- 关闭：断开流对象与磁盘文件关联

文件打开方式

ios中定义了文件打开方式：

```
enum open_mode
{     in    = 0x01,
      out = 0x02,
      ate  = 0x04,          //配合输入/输出（清空），指针置于文件尾
      app = 0x08,           //配合输出,新数据添加在尾部
      trunc = 0x10,         //打开文件并清空,以建立新文件
      binary = 0x80         //二进制文件
};
```

例如：ofstream ofile(“文件名”， ios :: binary | ios :: app);

【例9.9】将对象的数据保存到文本文件中。

```
class inventory
{    string Name, IdNo;          //货物名称与货号
      int Quantity;              //数量
     double Cost, TotalValue;    //单价与总价
public:
     inventory(string = "#", string = "0", int = 0, double = 0);
     friend ostream& operator<<(ostream&, inventory&);
     friend istream& operator>>(istream&, inventory&);
};

ostream & operator<< (ostream& dest, inventory& iv)
{     dest << iv.Name << '\t' << … << …;        //省略代码
       return dest;
}
```

```
istream& operator >> (istream& sour, inventory& iv)
{     sour >> iv.Name >> iv.No >> ……;        //省略代码
      return sour;
}

int main()
{      inventory car1("夏利2000","805637928",156,80000),
                  motor1(“金城125”, “93612575”, 302, 10000),
                  car2, motor2;
       ofstream destfile(“f1.txt");
       destfile << car1 << motor1;
       destfile.close();
       ifstream sourfile(“f1.txt”);       //重新作为数据源打开
      sourfile >> car2 >> motor2;
      sourfile.close();
      return 0;
}
```

<<、>>运算符可实现向标准设备I/O，也可实现向文本文件I/O。为什么?

二进制文件操作

文件打开：以二进制格式打开 ios :: binary

读写：用流提供的成员函数read/write实现读/写

istream& istream :: read(char * , int);

ostream& ostream :: write(const char * , int);

【例】将1 - 500之间的偶数写入文件data.dat中，再将文件中数据读到一个数组中。

```
int main() {
    ofstream outfile("data.dat", ios::binary);     //打开
     if(!outfile)  {    cout<<"can't open file"<<endl;   exit(1);    }
     for(int n = 2; n <= 500; n+=2)
          outfile.write((char*)&n, sizeof(n));    //写入文件
    outfile.close();                                //关闭

    ifstream infile("data.dat", ios::binary);       //重新打开
    if(!infile) {    cout<<"can't open file"<<endl;   exit(1);   }
    int a[250];
    infile.read((char*)a, sizeof(int)*250);         //读到数组
    infile.close();                                 //关闭
    return 0;
}
```

```
for(i = 0; !infile.eof(); i++)
    infile.read((char*)&a[i], sizeof(a[i]);
```

提示：读文件一般通过文件尾检测eof()完成。

如何实现将对象中的数据写入二进制文件 —— 定义写文件成员。（例9.10）

```
void Inventory :: bDatatofile(ofstream& dest)
{     dest.write((char*)&Name,sizeof(Name));     dest.write((char*)IdNo, sizeof(IdNo));
      dest.write((char*)&Quantity, sizeof(int));
      dest.write((char*)&Cost, sizeof(double));
      dest.write((char*)&TotalValue, sizeof(double));
}
void inventory::Bdatafromfile(ifstream&sour)
{     sour.read((char*)&Name, sizeof(Name);     sour.read ((char*)IdNo, sizeof(IdNo));
      sour.read((char*)&Quantity, sizeof(int));
      sour.read((char*)&Cost, sizeof(double));
      sour.read((char*)&TotalValue, sizeof(double));
  }
```

```
void main() {
    Inventory  car1(…), motor1(…) , car2, motor2 ;
    ofstream datafile(“f1.data", ios::binary);
    car1.bdatatofile(datafile);      //不同于文本文件的处理方法
    motor1.bdatatofile(datafile);
    datafile.close();
    ifstream sdatafile(“f1.data”, ios::binary);    //重新打开
    car2.bdatafromfile(sdatafile); cout<<car2<<endl;
    motor2.bdatafromfile(sdatafile);
    cout<<motor2<<endl;
    sdatafile.close();
}
```

文件的随机访问*

指控制文件指针移动，实现在任意位置读写数据，多用于二进制文件。

ios类中定义了枚举类型seek_dir，来表示文件指针位置：

```
enum seek_dir { beg = 0,  cur = 1,  end = 2  };
```

输入流操作文件指针的成员函数

```
istream& istream :: seekg(ios :: seek_dir);                 //直接定位
istream& istream :: seekg(long, ios :: seek_dir);           //相对定位
long istream :: tellg();                                    //返回当前指针位置
```

输出流操作文件指针的成员函数

```
ostream& ostream :: seekp(ios :: seek_dir);
ostream& ostream :: seekp(long, ios :: seek_dir);
long ostream :: tellp();
```

Q&A

文件与对象

设想一个需求

- 报名进行3天；
- 每天报名结束后将数据记录到文件中保存
- 第二天开始工作时将历史信息载入程序，重复第2-3两步

设计框架——

- 构造函数：打开文件、用文件数据初始化对象、关闭；
- 析构函数：打开文件、当前数据保存到文件、关闭
- 运行过程中，如遇信息修改，适时保存文件。

【例9.13】管理商店的一批货物。定义一个货物数组类。首次运行时，尚无数据文件，需由键盘输入数据，而后数据写入文件；此后每次构造数组时，都从文件初始化对象，恢复至前一次数据状态，开始新的工作。

```
template <typename T>
class Array
{     T *elements;        //数组首地址
      int last;           //最后一个有效数据下标
      int maxSize;        //数组容量
public:
      Array(string, int = 20);          //构造缺省容量20，初始化数据来自参数文件
      Array(int = 20);                  //构造缺省容量20，无初始化数据
      ~Array();                         //数据保存到文件
      friend ostream& operator <<(ostream&, const Array<T>&);    //重载<<
      bool isFull() const {    return last == maxSize-1;   }     //判断表是否满
      void renew(int);                  //增大表容量
      void insert (const T&, int );     //数据插入在下标位置处，可用于创建数组
      void insert (constT&);            //升序插入，可用于创建升序数组
};
```

```
template <typename T>
 Array<T> :: Array(string fname, int maxs)     //fname为关联文件的文件名
{     maxSize = maxs;
      last = -1;                                //空表标志
      elements = new T[maxSize];                //创建表空间
      T temp;
       ifstream infile(fname);                  //若用fstream，则需指明ios :: in
      if(infile)                                //如文件打开成功，则读数据
     {     while(!infile.eof())
           {     last++;
                 infile >> temp;                //处理文件尾
                 if(!infile.eof())    elements[last] = temp;
           }
           datafile.close();          //必须放此处，打开成功才能关闭
      }
     datafile.clear(0);              //清状态字，流不能恢复曾经的异常置位
}
```

```
template <typename T>
 Array<T> :: Array(int maxs)
{     maxSize = maxs;
      last = -1;
      elements = new T[maxSize];
}
```

```
template <typename T>
ostream& operator<< (ostream& os, const Array<T>& arr)
{     for(int i = 0; i <= arr.last; i++)
          os << arr.elements[i] << “  “;
      return os;
}
```

```
template <typename T>
Array<T> :: ~Array()
{     if(elements)
    {     string fname;    cout << “输入文件名：”;     cin >>  fname;
          ofstream outfile(fname);                  //写状态打开关联文件
          for(int i = 0; i <= last; i++)
               outfile << elements[i] << “  “;
          outfile.close();
          delete []elements;
    }
}
```

其他成员定义略

Array推演为商品类数组

```
class inventory
{     string Name;              //商品名称
      string No;                //货号
      int Quantity;             //数量
      double Cost;              //价格
      double TotalValue;        //总价
public:
      inventory(string = "#", string = "#", int = 0, double = 0);
      friend ostream&operator <<(ostream&, inventory&);
      friend istream&operator  >>(istream&, inventory&);
      bool operator <(inventory &);       //可能推演所需，货号为关键字
      bool operator >(inventory &);       //可能推演所需，货号为关键字
};
```

Array推演为商品类数组进行测试

```
int main()
{       Array<inventory> mylist(“mydata.txt”, 50);        //参数给出文件名
        inventory temp;
        for(int i = 0; i < 10; i++)            //再读入10个数据并有序插入
        {       cin >> temp;
                mylist. insertOrder (temp);
      }
      cout << mylist;                          //输出表中数据
      return 0;                                //退出前数据会写入文件
}
```

Q&A