中国矿业大学（北京）
China University of Mining and Technology (Beijing)

面向对象技术与C++程序设计

第二章 数据输入与输出

授课教师：张潇

机电与信息工程学院

计算机系






第二章 数据输入与输出

- 本章主要介绍C++程序数据的输入和输出

C++程序的结构

流与标准输入输出设备

cin、cout

输出格式控制

文件数据的建立与读取

2.1 C++程序的结构

1、C++程序的构成

- 声明部分
- 主函数部分
- 函数定义部分

2、程序结构的一个例程如下

```
0       //CH2-1   .cpp
1       #include<iostream.h>
2       #define N 10
3       void sort(int a[],int n);
4       void print(int a[],int);
5
6        void main(){
7           int a[N];
8           cout<<"input 10 numbers:\n";
9           for (int i=0;i<N;i++)
10              cin>>a[i];
11          sort(a,N);
12          print(a,N);
13      }
14
15      void sort(int a[],int n) {
16          for(int i=0;i<n-1;i++)
17             for(int j=i+1;j<n;j++){
18                 if(a[i]<a[j]){
19                    intt=a[i];
20                    a[i]=a[j];
21                    a[j]=t;
22                 }
23             }
24      }
25
26      void print(int a[],int n) {
27          for(int i=0;i<n;i++)
28             cout<<a[i]<<"   ";
29          cout<<endl;
30      }
```

声明部分

主函数部分

函数 sort()

函数 print()

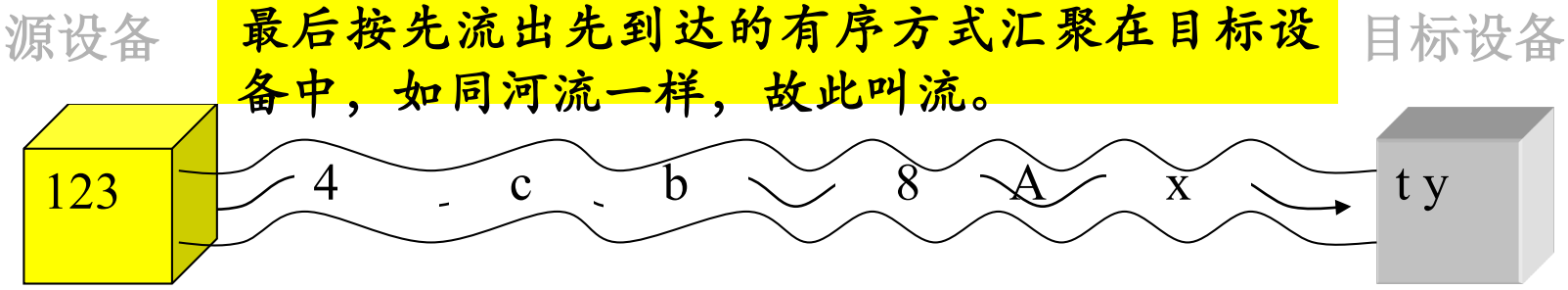
2.2 流与标准输入与输出设备

- 本节主要介绍C++程序中数据输入输出的方法，是C++程序设计的基础。应该掌握
  - 流的概念
  - iostream.h
  - cin
  - cout
  - endl

1、C及C++中的流概念

- I/O（input/ouput，输入/输出）数据是一些从源设备到目标设备的字节序列，称为字节流。除了图像、声音数据外，字节流通常代表的都是字符，因此在多数情况下的流（stream）是从源设备到目标设备的字符序列，

字符序列从源设备连续不断地流向目标设备，最后按先流出先到达的有序方式汇聚在目标设备中，如同河流一样，故此叫流。



- 输入流
  - 输入流（input stream）是指从输入设备流向内存的字节序列。
- 输出流
  - 输出流（output stream）是指从内存流向输出设备的字节序列。
- iostream
  - 输入流istream
  - 输出流ostream
- C++中输入输出数据的流变量
  - cin　输入流对象，C++已将其与键盘关联
  - cout　输出流对象，C++已将其与显示器关联

2.3 cin和析取运算符

1、cin的用途

cin读作(see-in）是一个输入流对象，用于从键盘输入数据。在C++程序中，也可以使用c语言中常用的scanf函数输入数据，但cin更简单。

2、cin的用法

- 输入单个变量的值

  cin>>x;

- 输入多个变量的值

  cin>>x1>>x2>>x3>>x4……>>xn;

其中x,x1……x2可是以内置数据类型如int，char，float，double等。

3、用cin时的注意事项

- **在一条cin语句中同时为多个变量输入数据。在输入数据的个数应当与cin语句中变量个数相同，各输入数据之间用一个或多个空白（包括空格、Tab）作为间隔符，全部数据输入完成后，按Enter键结束。**
- **在>>后面只能出现变量名，下面的语句是错误的。**

```
cin>>"x=">>x; //错误，>>后面含有字符串"x="
cin>>12>>x;   //错误，>>后面含有常数12
cin>>'x'>>x;
```

- cin具有自动识别数据类型的能力，析取运算>>将根据它后面的变量的类型从输入流中为它们提取对应的数据。比如：

  cin>>x;

  - 假设输入数据2，析取运算符>>将根据其后的x的类型决定输入的2到底是数字还是字符。若x是char类型，则2就是字符；若x是int，float之类的类型，则2就是一个数字。

  - 再如，若输入34，且x是char类型，则只有字符3被存储到x中，4将继续保存在流中；若x是int或float，则34就会存储x中。

– **数值型数据的输入**

在读取数值型数据时，析取运算符>>首先略掉数据前面的所有空白符号，如果遇到正、负号或数字，就开始读入，包括浮点型数据的小数点，并在遇到空白符或其他非数字字符时停止。例如：

```
int x1;
double x2;
char x3;
cin>>x1>>x2>>x3;
```

- **假如输入“35.4A”并按Enter键，x1是35；x2 是0.4；x3是'A'**

4、输入数据案例分析

【例2-2】 假设有变量定义语句如下：

```
int a,b;
double z;
char ch;
```

下面的语句说明数据输入的含义。

	语句	输入	内存变量的值
1	cin>>ch;	A	ch='A'
2	cin>>ch;	AB	ch='A'，而'B'被保留在输入流中等待被读取
3	cin>>a;	32	a=32
4	cin>>a;	32.23	a=32，.23留在输入流中等待被读取
5	cin>>z;	76.21	z=76.21
6	cin>>z;	65	z=65.0
7	cin>>a>>ch>>z	23 B 3.2	a=23，ch='B',Z=3.2
8	cin>>a>>ch>>z	23B3.2	a=23，ch='B',Z=3.2
9	cin>>a>>b>>z	23 32	a=23，b=32，等待输入下一个数据存入z
10	cin>>a>>z	2 3.2 24	a=2，z=3.2，24被保留在输入流中等待被读取
11	cin>>a>>ch	132	a=132，计算机等待输入 ch的值
12	cin>>ch>>a	132	ch='1'，a=32

4、get输入空白字符

- 用cin 输入数据时,空白作为数据之间的间隔,无法输入

```
char c1,c2;
int n;
std::cin>>c1>>c2>>n;
```

若输入：X 5

则X将存入c1，5被存入c2，n没有输入值

- get输入流函数完成单个空白字符（包括空格、回车换行、Tab等）的输入，
- get函数的用法如下：

```
std::cin.get();
std::cin.get(char varChar);
```

例如:

```
std::cin.get(c1);
std::cin.get(c2);
std::cin>>n;
```

若输入 1 3,则c1为1,c2为空白,c2为3

5、getline输入包括空白字符的长字符串

– getline函数一次读取一行字符，其用法如下

std::cin.getline(char *c ,int n ,char ='\n');

【例2-3】 getline读取一行输入存入字符串中

```
#include<iostream>
int main()
{
   char s1[100];
   std::cout<<"use getline input char: ";
   std::cin.getline(s1,50);
   std::cout<<s1<<std::endl;
    return 0;
}
```

- Test题目：编写程序实现从标准输入每次读入一行文本，然后改写程序，每次读入一个单词

2.4 cout和数据输出

1、cout的用途

cout（读作see-out）是一个输出流对象，已被C++默认关联到显示器，用于在屏幕上输入数据。

在C++程序中，也可使用C语言的printf输出数据，但cout更简单。

2、cout的用法

```
cout<<x;
```

其中x可是以内置数据类型如int，char，float，double等。

- 输出字符类型的数据

对于字符变量和字符串变量，cout将把变量的值输出到显示屏幕上。对于字符常量和字符串常量，cout将把它们原样输出在屏幕上

【例2-4】 用cout输出字符数据。

```
//CH2-4.cpp
#include<iostream>
int main()
{
    char ch1='c';
    char ch2[]="Hello C++!";
    std::cout<<ch1;
    std::cout<<ch2;
    std::cout<<"C";
    std::cout<<"Hello everyone!";
    return 0;
}
```

3、连续输出

- cout能够同时输出多个数据，用法如下：

  cout<<x1<<x2<<x3<<…;

  例：

  ```
  cout<<ch1<<ch2<<"C"<<"Hello everyone!";
  ```

- 与C语言一样，在C++程序中也可以将一条命令写在多行上。比如，上面的语句也可写成下面的形式：

  ```
  cout<<ch1
      <<ch2
      <<"C"
      <<"Hello everyone!";
  ```

4、输出换行

在cout语句中换行可用：\n或endl

【例2-5】 在例2-4的输出语句中增加换行符。

```
// CH2-5.cpp
#include<iostream>
int main(){
    char ch1='c';
    char ch2[]="Hello C++!";
    std::cout<<ch1<<std::endl;
    std::cout<<ch2<<'\n';
    std::cout<<"C"<< std:: endl;
    std::cout<<"Hello everyone!\n";
     return 0;
}
```

2.5 cout输出格式控制符

1、输入数据类型数据

在连续输入多个数据时，应注意在数据之间插入间隔符。如

```
int x1=23;
float x2=34.1;
double x3=67.12;
cout<<x1<<x2<<x3<<900;
```

其中的cout语句将在屏幕上输出，

2334.167.12900

2、数制基数

hex：16进制，oct：8进制，dec：10进制

【例2-6】 输出不同进制的数据。

```
//ch2-6.cpp
#include<iostream>
int main()
{
   int x=34;
   std::cout<<std::hex<<17  <<"  "<<x<<"
   "<<18<<std::endl;
   std::cout<<17   <<"  "<<std::oct <<x<<"
   "<<18<<std::endl;
   std::cout<<std::dec<<17  <<"  "<<x<<"
   "<<18<<std::endl;
    return 0;
}
```

3、其它输出格式控制符

#include <iomanip>

- 设置浮点数的精度
  - setprecision(n)
- 设置输出域
  - setw(n)
- 设置对齐方式
  - setiosflags(long f);
  - resetiosflags(long f);

【例2-7】用setiosflags和 resetiosflags设置 和取消输出数据的对齐方式。

```
//CH2-7.cpp
#include<iostream>
#include<iomanip>
int main(){
std::cout<<“123456781234567812345678”
                    << std:: endl;
std:: cout<<std::setiosflags(std::ios::left)<<std::setw(8)
          <<456<<std::setw(8)<<123<< std::endl;
std:: cout<<std::resetiosflags(std::ios::left)<<std::setw(8)
          <<123<< std:: endl;
return 0;
}
```

【例2-8】用fill和setfill设置输出填充字符。

```
//ch2-8.cpp
#include<iostream>
#include<iomanip>
int main()
{
    std::cout<<"123456781234567812345678"<<std::endl;
    std::cout<<std::setw(8)<<123<<std::setw(8)<<456<<std::setw(8)<<789<<std::endl;
    std::cout.fill('@');
    std::cout<<std::setw(8)<<123<<std::setw(8)<<456<<std::setw(8)<<789<<std::endl;
    std::cout<<std::setfill('^');
    std::cout<<std::setw(8)<<123<<std::setw(8)<<456<<std::setw(8)<<789<<std::endl;
    return 0;
}
```

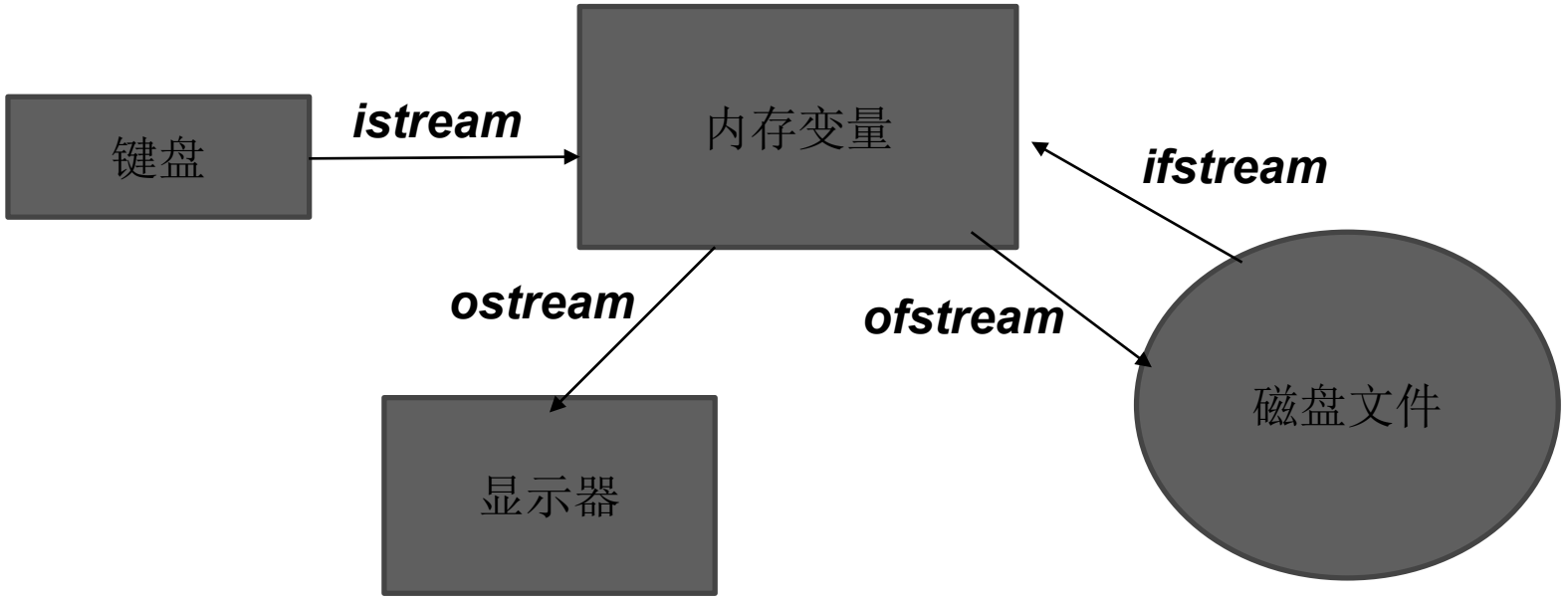
4. 输出填充字符(用指定字符填充空白)

```
std::cout.fill(ch);
std::cout<< std::setfill(ch);
```

2.6 文件数据输入与输出

1.文件数据读取基理

程序与文件的数据交换方法同它与标准输入/输出设备的数据交换方法相同，从文件读取数据与从键盘输入数据的方法相似，将数据写入文件与将数据输出到显示器的方法相似。



2、文件数据读取的过程

(1) 在程序中包含头文件fstream

```
#include <fstream>
```

(2) 定义文件流变量

```
ifstream inData;        //定义输入文件流变量
ofstream outData;       //定义输出文件流变量
```

(3) 将文件流变量与磁盘文件关联起来

```
inData.open(filename，mode);
outData.open(filename，mode);
```

(4)

ios::in 输入模式
ios::out 输出模式
ios::app 增加模式
ios::truc 如果文件已存在，先删除该文件
ios::binary 二进制模式

(5)

```
inData.close();
outData.close();
```

3、文件操作案例

【例2-9】在C根目录下建立一文件student.dat，并从键盘输入3个学生的数据到该文件中。每个学生的数据包括姓名、学号，以及数学、英语、计算机等课程的成绩。。

	学号	数学	英语	计算机
学生1	110101	76	87	89
学生2	110102	87	78	90
……				

```
//CH2-9.cpp
#include <iostream>
#include <fstream>
using namespace std;
int main() {
ofstream outfile("student.dat");
char name[8], id[8];
int math, eng, computer;
for (int i = 0; i<3; i++) {
cout << "name: "; cin>>name;
cout << "stno : "; cin>>id;
cout << "math : "; cin>>math;
cout << "eng : "; cin>>eng;
cout << "computer : "; cin>>computer;
outfile << name << "  " << id << "  " << math << "  " << eng << " " << computer <<
    endl;}
outfile.close();
return 0;}
```

定义文件变量,对outfile的操作实际是对C根目录中的student.dat磁盘文件的操作

将内存变量的值写入outfile,实际上写出到磁盘文件student.dat中

3、文件操作案例

【例2-10】编写一程序建立一文件student.dat，并从键盘输入3个学生的数据到该文件中。每个学生的数据包括姓名、学号，以及数学、英语、计算机等课程的成绩。将文件student.dat中的数据读出来，计算每个同学的总分，并显示在屏幕上。输出格式如下：。

	学号	数学	英语	计算机	总分
学生1	1	1	1	1	3
学生2	2	2	2	2	6

```
//ch2-10.cpp
#include<iostream>
#include<iomanip>
#include<fstream>
int main() {
    std::ifstream infile("student.dat");
    char name[8], id[8];
    int math, eng, computer, sum;
     std::cout << std::setw(10) << "name" << std::setw(10) << "stno"
     << std::setw(10) << "math"<<std::setw(10)<<"eng“
     << std::setw(12) << "computer"<<std::setw(10)<<"sum"
     << std::endl << std::endl;
```

定义文件变量,对infile的操作实际是对当C根目录中的student.dat磁盘文件的操作

```
infile>>name;
    while (!infile.eof()){
       infile>>id>>math>>eng>>computer;
       sum=math + eng + computer;
       std::cout<<std::setw(10)<<name<<std::setw(10
                <<id<<std::setw(10)<<math
               <<std::setw(10)<<eng
                    <<std::setw(12)<<computer
                <<std::setw(10)<<sum <<std::endl;
       infile>>name,
    }
    infile.close();
}
```

将文件变量中的数据读入到内存变量中，实际上是将磁盘文件student.dat中值读入到内存变量中.

- Test题目：建立两个磁盘文件f1.dat和f2.dat，编程序实现以下工作：

①从键盘输入16个整数，分别存放在两个磁盘文件中(每个文件中放8个整数)；

②从f1.dat读入8个数，然后存放到f2.dat文件原有数据的后面；

③从f2.dat中读入16个整数，将它们按从小到大的顺序存放到f2.dat(不保留原来的数据)。

④分别输出文件f1.dat和f2.dat的内容。

Test题目：编程序，在显示屏上显示一个由字母B组成的三角形。（用控制符或流成员函数控制输出格式）

B

BBB

BBBBB

BBBBBBB

BBBBBBBBB