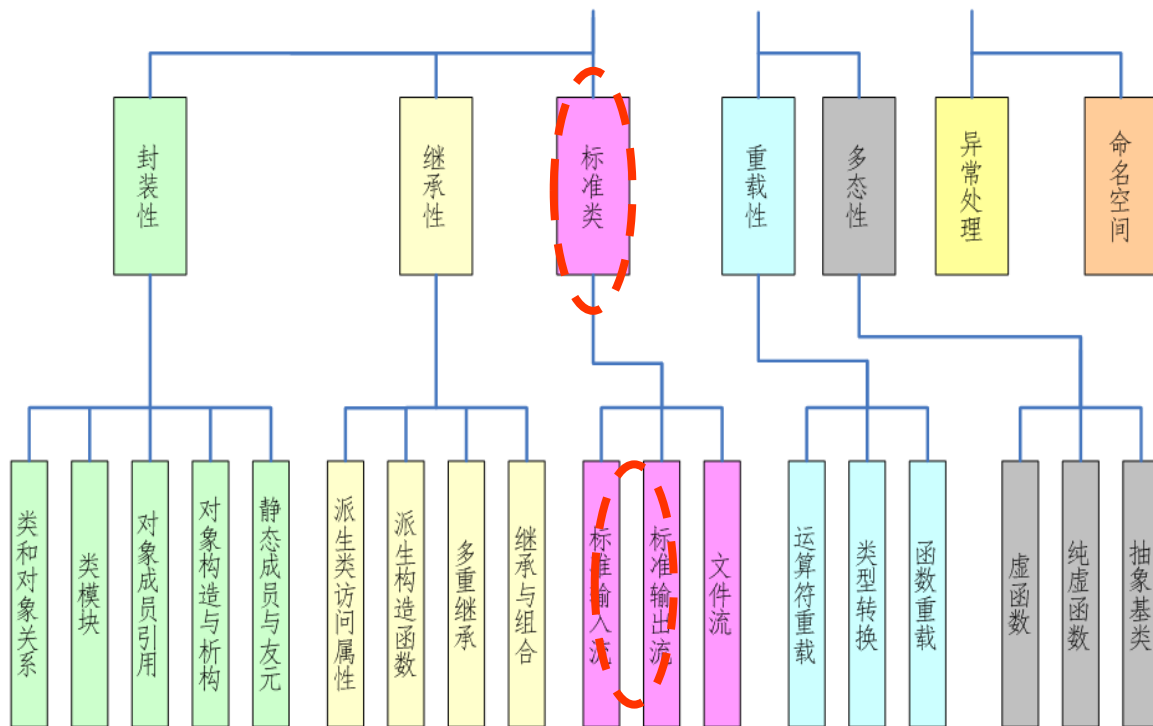


第9讲 输入输出流

9.1 C++的输入和输出流

9.2 标准I/O流操作函数

C++的OOP程序 = 对象 + 消息 + 工具



计算机的I/O操作小结



scanf()
cin
getchar()
gets()



printf()
cout
putchar()
puts()



fprintf()
fputw()
fputc()
fputs()
fwrite()



fscanf()
fgetc()
fgetw()
fgets()
fwrite()

9.1 C++的输入和输出

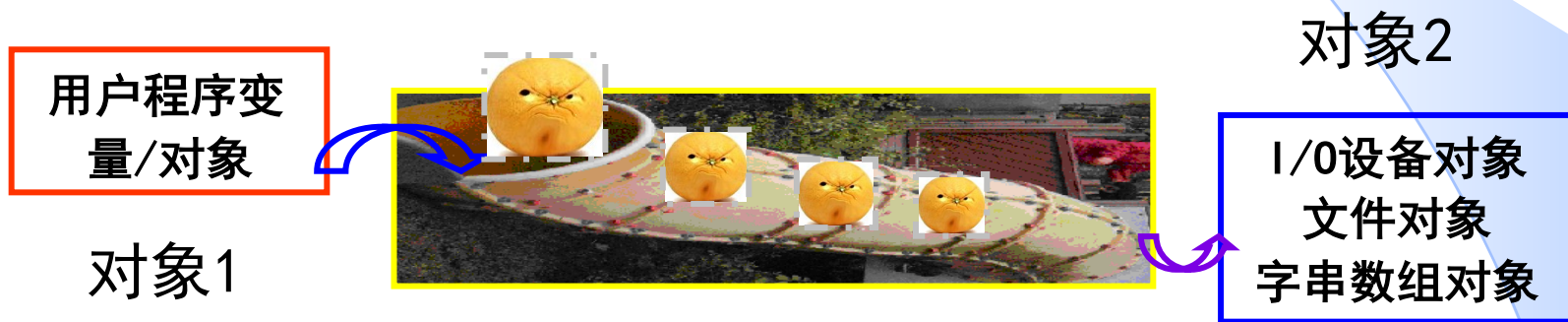
□ C++ I/O类型

- (1) 标准I/O：对系统指定的标准设备的I/O操作；
- (2) 文件I/O：以磁盘文件为对象进行I/O，即从磁盘文件输入、输出数据到磁盘文件；
- (3) 字符串I/O：对内存中指定空间（通常为一个字符数组）进行输入和输出。

- C++ I/O操作的“**类型安全** (type safe)” 。编译系统对数据类型进行严格检查，凡是类型不正确数据都不能通过编译；
- C++ I/O操作的“**可扩展性**” 。不仅可以用来输入、输出标准类型的数据，也可**I/O用户自定义类型数据**。

□ 流的概念

- 外延：从一个对象到另一个对象的数据流动的字节序列；
- 内涵：内存中为每个流开辟一个内存缓冲区，用来存放流中的数据，即流是与内存缓冲区相对应。流的内容可是ASCII字符、二进制数，图像和视音频等数据。



□ 流的类型

- 标准流：用于程序与I/O设备之间的数据交互；
- 文件流：用于程序与磁盘文件之间的数据交互；
- 字符串流：用于程序与内存字符串数组之间的数据交互。

注意：<iostream.h>是旧的写法，标准化后的C++推荐使用<iostream>的写法！

□ C++ I/O 类库结构

- C++的I/O流被定义为不同的流类(stream class)。用流类定义的对象称为流对象。cout和cin是iostream类的全局对象；
- ios是抽象基类，派生类族如图。istream类支持I操作，ostream类支持O操作，iostream类支持I/O操作；
- ifstream和ofstream类是支持文件操作。ifstream支持对文件的I操作，ofstream支持对文件的O操作。

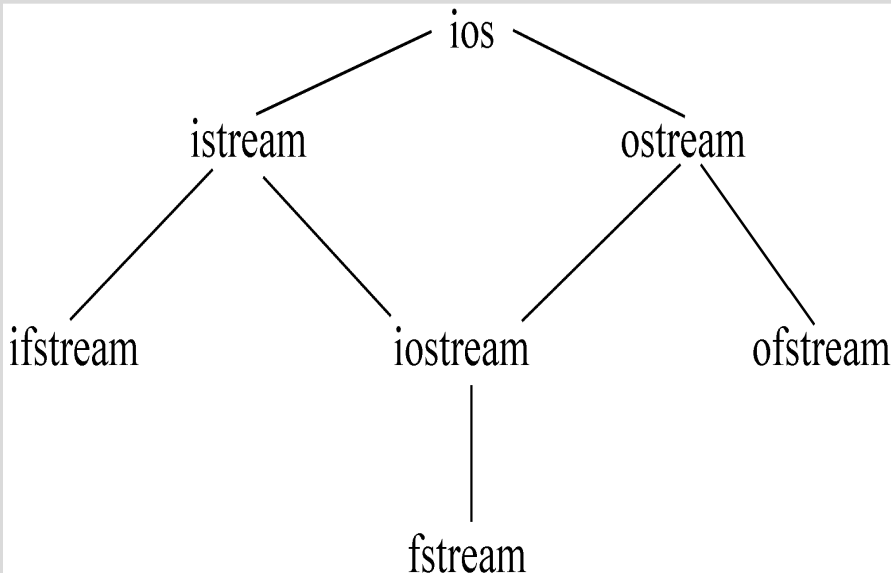
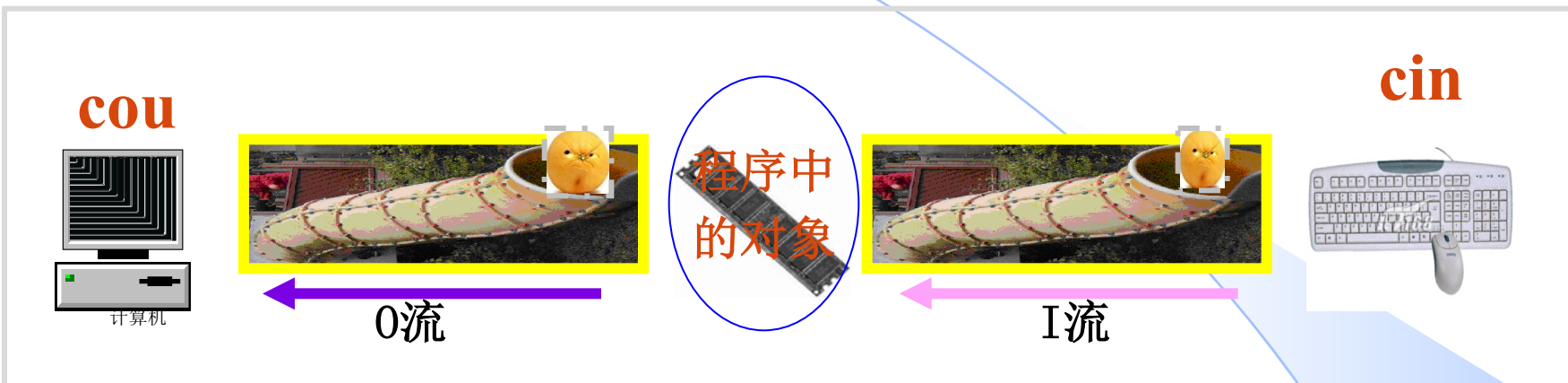


表 13.1 I/O 类库中的常用流类

类 名	作 用	在哪个头文件中声明
ios	抽象基类	iostream
istream	通用输入流和其他输入流的基类	iostream
ostream	通用输出流和其他输出流的基类	iostream
iostream	通用输入输出流和其他输入输出流的基类	iostream
ifstream	输入文件流类	fstream
ofstream	输出文件流类	fstream
fstream	输入输出文件流类	fstream
istrstream	输入字符串流类	strstream
ostrstream	输出字符串流类	strstream
strstream	输入输出字符串流类	strstream

9.2 C++的标准I/O流

□ C++的标准I/O流的概念



□ 标准I/O流：实现程序中对象与标准I/O设备（对象）之间的数据交换；

□ 缓存规则

- 输出操作时，程序对象数据先预存在缓冲区中，直到满或者endl，缓冲区的全部数据自动送到屏幕或磁盘文件；
- 输入时，键盘键入的数据先预存到缓冲区，当按回车键，才读入到程序中的对象。

9.2 C++的标准I/O流

□ 在iostream头文件中重载运算符

istream类已将“>>”重载为标准类型的提取运算符：char, signed char, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, float, double, long double, char*, signed char*, unsigned char*, void*类型；

```
istream& operator>>(bool );
istream& operator>>(char );
istream& operator>>(short);
istream& operator>>(int );
istream& operator>>(float);
istream& operator>>(char*);
```

这些函数都是以成员函数的方式定义的，支持下列使用方式：
istream对象 >> 基本类型变量
ostream对象 << 基本类型变量

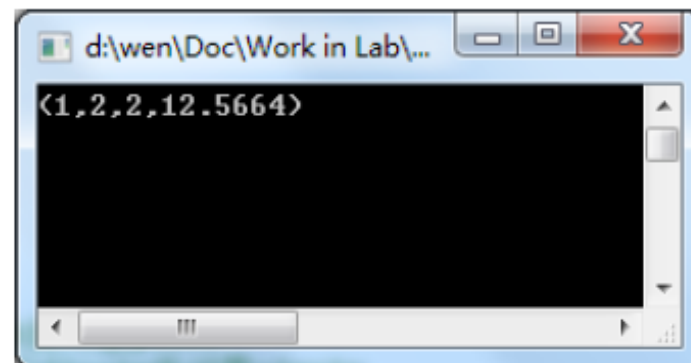
□ “<<”和“>>”还可用于自定义类型数据，必须在程序中对“<<”和“>>”进行重载，并申明为类友元（WHY?）

- ✓ istream & operator >> (istream &, 自定义类 &);
- ✓ ostream & operator << (ostream &, 自定义类 &);


```
class Circle : public Point {
    int x, y;
    float r;
public:
    Circle(float xx=0, float yy=0, float rr=0);
    //...
    friend ostream& operator <<(ostream& out, const Circle& cr);
};

ostream& operator <<(ostream& out, const Circle& cr)
{   out << "(" << cr.getX() << "," << cr.getY() << ","
        << cr.r << "," << cr.area() << ")" << endl;
    return out;
}
```

```
Circle cr(1, 2, 2.0);
cout << cr;
```



▣ **cout流**： ostream类中构建的输出流全局对象（cout）

（1）“cout<<” 输出数据时，系统会判断数据类型，根据其类型选择调用与之匹配的运算符重载函数；

（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流插入一个endl时，终止一行字符串并刷新缓冲区；插入ends时，终止字符串并插入空格符。

▣ **输出格式控制**

（1）使用控制符控制输出格式：在头文件iomanip中定义；

（2）用流对象的成员函数控制输出格式：流成员函数setf和控制符setiosflags括号中的参数表示格式状态。

表 13.3 输入输出流的控制符

控 制 符	作 用
dec	设置整数的基数为 10
hex	设置整数的基数为 16
oct	设置整数的基数为 8
setbase(n)	设置整数的基数为 n(n 只能是 8,10,16 三者之一)
setfill(c)	设置填充字符 c,c 可以是字符常量或字符变量
setprecision(n)	设置实数的精度为 n 位。在以一般十进制小数形式输出时 n 代表有效数字。在以 <u>fixed(固定小数位数)</u> 形式和 <u>scientific(指数)</u> 形式输出时 n 为小数位数
setw(n)	设置字段宽度为 n 位
setiosflags(ios::fixed)	设置浮点数以固定的小数位数显示
setiosflags(ios::scientific)	设置浮点数以科学记数法(即指数形式)显示
setiosflags(ios::left)	输出数据左对齐
setiosflags(ios::right)	输出数据右对齐
setiosflags(ios::skipws)	忽略前导的空格
setiosflags(ios::uppercase)	在以科学记数法输出 E 和以十六进制输出字母 X 时以大写表示
setiosflags(ios::showpos)	输出正数时给出“+”号
resetioflags()	终止已设置的输出格式状态,在括号中应指定内容

□ 用控制符控制输出格式

```

1 #include <iostream>
2 #include <iomanip> //不要忘记包含此头文件
3 using namespace std;
4 int main()
5 {int a;
6  cout<<"input a:";
7  cin>>a;
8  cout<<"dec:"<<dec<<a<<endl; //以十进制
9  cout<<"hex:"<<hex<<a<<endl; //以十六进
10 cout<<"oct:"<<setbase(8)<<a<<endl; //以八进制形式输出整数a
11 char *pt="China"; //pt指向字符串"China"
12 cout<<setw(10)<<pt<<endl; //指定域宽为10，输出字符串
13 cout<<setfill('*')<<setw(10)<<pt<<endl; //指定域宽10，输出字符串，空白处以'*'填充
14 double pi=22.0/7.0; //计算pi值
15 cout<<setiosflags(ios::scientific)<<setprecision(8); //按指数形式输出，8位小数
16 cout<<"pi="<<pi<<endl; //输出pi值
17 cout<<"pi="<<setprecision(4)<<pi<<endl; //改为4位小数
18 cout<<"pi="<<setiosflags(ios::fixed)<<pi<<endl; //改为小数形式输出
19 return 0;}

```

C:\WINDOWS\system32\cmd.exe

```

input a:14
dec:14
hex:e
oct:16
      China
*****China
pi=3.14285714e+000
pi=3.1429e+000
pi=3.143
请按任意键继续. . .

```

表 13.4 用于控制输出格式的流成员函数

流成员函数	与之作用相同的控制符	作 用
precision(n)	setprecision(n)	设置实数的精度为 n 位
width(n)	setw(n)	设置字段宽度为 n 位
fill(c)	setfill(c)	设置填充字符 c
setf()	setiosflags()	设置输出格式状态, 括号中应给出格式状态, 内容与控制符 setiosflags 括号中的内容相同, 如表 13.5 所示
unsetf()	resetiosflags()	终止已设置的输出格式状态, 在括号中应指定内容

表 13.5 设置格式状态的格式标志

格式标志	作 用
ios: :left	输出数据在本域宽范围内向左对齐
ios: :right	输出数据在本域宽范围内向右对齐
ios: :internal	数值的符号位在域宽内左对齐, 数值右对齐, 中间由填充字符填充
ios: :dec	设置整数的基数为 10
ios: :oct	设置整数的基数为 8
ios: :hex	设置整数的基数为 16
ios: :showbase	强制输出整数的基数(八进制数以 0 打头, 十六进制数以 0x 打头)
ios: :showpoint	强制输出浮点数的小点和尾数 0
ios: :uppercase	在以科学记数法格式 E 和以十六进制输出字母时以大写表示
ios: :showpos	对正数显示“+”号
ios: :scientific	浮点数以科学记数法格式输出
ios: :fixed	浮点数以定点格式(小数形式)输出
ios: :unitbuf	每次输出之后刷新所有的流
ios: :stdio	每次输出之后清除 stdout, stderr

□ 用流控制成员函数输出数据

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  {int a=21;
5   cout.setf(ios::showbase);//显示基数符号(0x或0)
6   cout<<"dec:"<<a<<endl;      //默认以十进制形式输出a
7   cout.unsetf(ios::dec);      //终止十进制的格式设置
8   cout.setf(ios::hex);        //设置以十六进制输出的状态
9   cout<<"hex:"<<a<<endl;      //以十六进制形式输出a
10  cout.unsetf(ios::hex);      //终止十六进制的格式设置
11  cout.setf(ios::oct);         //设置以八进制输出的状态
12  cout<<"oct:"<<a<<endl;      //以八进制形式输出a
13  cout.unsetf(ios::oct);
14  char *pt="China";           //pt指向字符串"China"
15  cout.width(10);              //指定域宽为10
16  cout<<pt<<endl;              //输出字符串
17  cout.width(10);              //指定域宽为10
18  cout.fill('*');              //指定空白处以'*'填充
19  cout<<pt<<endl;              //输出字符串
20  double pi=22.0/7.0;          //输出pi值
21  cout.setf(ios::scientific);  //指定用科学记数法输出
22  cout<<"pi=";                  //输出"pi="
23  cout.width(14);              //指定域宽为14
24  cout<<pi<<endl;              //输出pi值
25  cout.unsetf(ios::scientific); //终止科学记数法状态
26  cout.setf(ios::fixed);       //指定用定点形式输出
27  cout.width(12);              //指定域宽为12
28  cout.setf(ios::showpos);     //正数输出"+"号
29  cout.setf(ios::internal);    //数符出现在左侧
30  cout.precision(6);           //保留6位小数
31  cout<<pi<<endl;              //输出pi, 注意数符"+"的位置
32  return 0;}

```



```

C:\WINDOWS\system32
dec:21
hex:0x15
oct:025
    China
*****China
pi=*3.142857e+000
+***3.142857
请按任意键继续. . .

```

```
#include <iostream>
```

```
#include <iomanip> //格式控制
```

```
using namespace std;
```

```
void main()
```

```
{
```

```
double amount = 22.0 / 7;
```

```
cout << amount << endl;
```

//(1) 以默认格式输出

```
cout << setprecision(0) << amount << endl
```

//(2) 设置有效位数为0等于默认格式

```
<< setprecision(1) << amount << endl
```

//(3) 设置有效位数为1

```
<< setprecision(2) << amount << endl
```

//(4) 设置有效位数为2

```
<< setprecision(3) << amount << endl
```

//(5) 设置有效位数为3

```
<< setprecision(4) << amount << endl;
```

//(6) 设置有效位数为4

```
cout << setiosflags(ios::fixed);
```

```
cout << setprecision(8) << amount << endl;
```

//(7) 设置有效位数为8

```
cout << resetiosflags(ios::fixed);
```

// 由于fixed与scientific不互斥，同时设置会

输出出错，所以这里取消fixed设置

```
cout << setiosflags(ios::scientific) << amount << endl; // (8) 设置以科学计数法显示
```

示

```
cout << setprecision(6);
```

```
}
```

C:\WINDOWS\system32\cmd.exe

3.14286

3.14286

3

3.1

3.14

3.143

3.14285714

3.14285714e+000

请按任意键继续. . .

9.2 C++的标准I/O流

▣ ostream类中构建2个流对象（cerr和clog）

- ① cerr/clog：标准错误流对象，被指定与显示器关联。其作用是向标准错误设备输出有关出错信息
- ② 调试程序时，往往不希望程序运行时的出错信息被送到文件，而要求在显示器上及时输出，这时应该用Cerr/clog。Cerr/clog流中的信息是用户根据需要指定
- ③ Clog与cerr区别：cerr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


```

1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4 int main()
5 { float a,b,c,disc;
6   cout<<"please input a,b,c:";
7   cin>>a>>b>>c;
8   if (a==0)
9     cerr<<"a is equal to zero,error!"<<endl; //将有关出错信息插入cerr流, 在屏幕输出
10  else
11    if ((disc=b*b-4*a*c)<0)
12      clog<<"disc=b*b-4*a*c<0"<<endl; //将有关出错信息插入cerr流, 在屏幕输出
13  else
14    { cout<<"x1="<<(-b+sqrt(disc))/(2*a)<<endl;
15      cout<<"x2="<<(-b-sqrt(disc))/(2*a)<<endl;}
16  return 0;}

```

C:\WINDOWS\system32\cmd.exe

please input a,b,c:0 3 2
a is equal to zero,error!
请按任意键继续. . .

C:\WINDOWS\system32\cmd.exe

please input a,b,c:5 1 5
disc=b*b-4*a*c<0
请按任意键继续. . .

□ ostream类的成员函数put

ostream类还提供了专用于输出单个字符的成员函数put。如cout.put('a'); put函数的参数可是字符或字符的ASCII代码(也可是一个整型表达式)。

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  { char *a="BASIC";
5    for(int i=4;i>=0;i--)
6      cout.put(*(a+i)); //putchar( *(a+i));
7    cout.put('\n'); //putchar('\n');
8    return 0;}
9

```

C:\WINDOWS\system32\cmd.exe

CISAB

请按任意键继续. . .

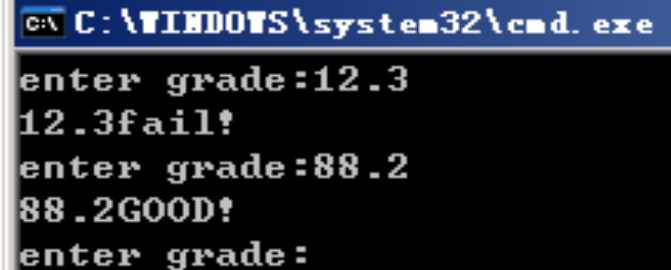
□ cin流对象

- ① cin是istream类对象。变量通过“>>”从流中提取数据。提取时以输入流中的空格、tab键、换行符等为数据分隔符。
- ② 只有在输入完数据按回车键，该数据才从键盘缓冲区被送入到流缓冲区，形成输入流，“>>”才能提取数据。
- ③ cin的输入状态：正常状态，cin>0，否则，cin=0。

```

1  #include <iostream>
2  using namespace std;
3  int main( )
4  { float grade;
5    cout<<"enter grade:";
6    while(cin>>grade)//能从cin流读取数据
7    {if(grade>=85) cout<<grade<<"GOOD!"<<endl;
8     if(grade<60) cout<<grade<<"fail!"<<endl;
9     cout<<"enter grade:";}
10   cout<<"The end."<<endl;
11   return 0;}

```



```

C:\WINDOWS\system32\cmd.exe
enter grade:12.3
12.3fail!
enter grade:88.2
88.2GOOD!
enter grade:

```

□ `istream`类字符输入的流成员函数`get()`

3种调用形式：无参数、一个参数、3个参数

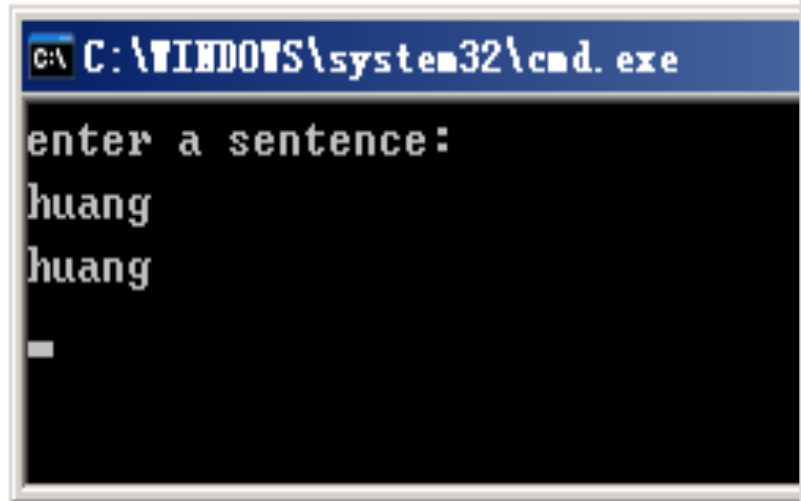
(1) 不带参数`get`函数：调用形式为`cin.get()`；

用来从输入流中读取一个字符，返回值是读入字符。若遇到输入流中的文件结束符，则返回文件结束标志`EOF`。

```

1  #include <iostream>
2  using namespace std;
3  int main( )
4  {int c;
5   cout<<"enter a sentence:"<<endl;
6   while((c=cin.get())!=EOF)
7     cout.put(c);
8   return 0;}

```



```

C:\WINDOWS\system32\cmd.exe
enter a sentence:
huang
huang
_

```

(2) 一个参数调用形式: `cin.get(ch)`

从输入流中读取一个字符，赋给字符变量ch。如果读取成功则函数返回非0值，如失败(遇文件结束符)则函数返回0；

(3) 3个参数调用形式: `cin.get(字符数组/指针, 字符个数n, 终止字符)`；

从输入流中读取n-1个字符，如果在读取n-1个字符之前遇到指定的终止字符，则提前结束读取。如果读取成功则函数返回非0值，如失败(遇文件结束符)则函数返回0值。

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 { char ch[20];
5   cout<<"enter a sentence:"<<endl;
6   cin.get(ch,10,'\n');//指定换行符为终止字符
7   cout<<ch<<endl;
8   return 0;}
```

C:\WINDOWS\system32\cmd.exe

enter a sentence:

huangyongfeng

huangyong

请按任意键继续. . .

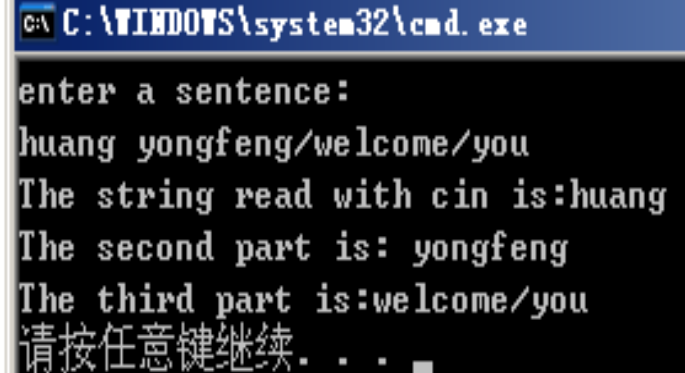
□ 成员函数getline () 函数读入一行字符

其用法与带3个参数的get函数类似, 即: cin.getline(字符数组/字符指针), 字符个数n, 终止标志字符)

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  { char ch1[20],ch2[20],ch3[20];
5    cout<<"enter a sentence:"<<endl;
6    cin>>ch1;
7    cout<<"The string read with cin is:"<<ch1<<endl;
8    cin.getline(ch2,10,'/');
9    cout<<"The second part is:"<<ch2<<endl;
10   cin.getline(ch3,20);
11   cout<<"The third part is:"<<ch3<<endl;
12   return 0;}

```



```

C:\WINDOWS\system32\cmd.exe
enter a sentence:
huang yongfeng/welcome/you
The string read with cin is:huang
The second part is: yongfeng
The third part is:welcome/you
请按任意键继续. . .

```

□ `cin.get(buff, size, ' \n')`和`cin.getline(buff, size, \n')`

➤ 功能基本相同;

➤ 区别

(1)`cin.get ()` 终止符不会保存在数组中, 但仍然保留在输入流中 (终止符就是读取的下一个字符, 除非分隔符从输入流中删除 (可使用`cin.ignore()`). 否则紧接着的第2个`get ()`操作结果为空。

(2)`cin.getline ()` 要从输入流中删除终止符 (即读取它并删除), 也不将它保存在数组中。


```

1  #include <iostream>
2  using namespace std;
3  int main()
4  { char ch1[20],ch2[20],ch3[20];
5    cout<<"enter a sentence:"<<endl;
6    cin>>ch1;
7    cout<<"The string read with cin is:"<<ch1<<endl;
8    cin.getline(ch2,10,'/');
9    cout<<"The second part is:"<<ch2<<endl;
10   cin.getline(ch3,20);
11   cout<<"The third part is:"<<ch3<<endl;
12   return 0;}

```

C:\WINDOWS\system32\cmd.exe

```

enter a sentence:
huang yongfengwelcomeyou!!!!!!!!!!!!!!
The string read with cin is:huang
The second part is: yongfeng
The third part is:
请按任意键继续. . .

```

- 注意: `getline()` 在读入 $n-1$ 字符后会清除缓存, 但遇到结束符提前结束不清空

全局范围)

```

1
2 #include<iostream>
3 #include<string>
4 using namespace std;
5 int main()
6 {char ch[10], ch1[10];
7  cout<<"enter a string:"<<endl;
8  cin.getline(ch,10,"");
9      cout<<ch<<endl;
10 cin.getline(ch1,10,"");
11      cout<<ch1<<endl;
12 }
13
14
15
```

C:\WINDOWS\system32\cmd.exe

```

enter a string:
huangyong\fenghuangyongfeng
huangyong
fenghuang
请按任意键继续. . .

```

C:\WINDOWS\system32\cmd.exe

```

enter a string:
huang\yongfenghuangyongfeng
huang
yongfengh
请按任意键继续. . .

```

C:\WINDOWS\system32\cmd.exe

```

enter a string:
huangyongfeng huangyongfeng
huangyong

请按任意键继续. . .

```

□ 成员函数Eof ()

从输入流读取数据，如果到达文件末尾或CTRL+Z，eof () 为非零值；否则为0

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  { char c;
5    while(!cin.eof())
6      if((c=cin.get())!=' ')
7        cout.put(c);
8    return 0;
9  }
```

C:\WINDOWS\system32\cmd.exe

```

huang yongfeng
huangyongfeng
huang
huang
yongfeng
yongfeng
end
end
^Z
```

请按任意键继续. . .

(b)

cin.ignore(n, 终止字符): 跳过输入流中n个字符, 或在遇到指定的终止字符时提前结束(此时跳过包括终止字符在内若干字符)。如 `ignore(5, 'A')` // 跳过输入流中5个字符, 遇 'A' 后就不再跳了。 `ignore()` (n默认值为1, 终止字符默认为EOF), 相当于 `ignore(1, EOF)`。

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  { char ch[20];
5    cin.get(ch,20,'/');
6    cout<<"The first part is:"<<ch<<endl;
7    // cin.ignore();
8    cin.getline(ch,20,'/');
9    cout<<"The second part is:"<<ch<<endl;
10   return 0;
11  }
  
```

```

C:\WINDOWS\system32\cmd.exe
huang/huang/
The first part is:huang
The second part is:huang
请按任意键继续. . .
  
```

```

C:\WINDOWS\system32\cmd.exe
huang/huang/
The first part is:huang
The second part is:
请按任意键继续. . .
  
```

第9次作业(必做题)

- 要求：本次作业2道题目，第10周末交作业，要求同前面
- 第1题. 在第8次作业第1题基础上，给“xx管理系统”设计一个主界面，采用C++I/O操作在屏幕打印一个漂亮的图案和菜单。例如，可参考如下界面，或另外设计。
- 说明：程序只要显示菜单，不要求将菜单与相关程序模块链接。

```

+++++
      XXX 公司人事管理系统

+++++

#####

      主菜单
      1. 数据录入
      2. 数据查询
      3. 数据保存
      4. 退出
      请选择序号 (1-4)

#####
  
```

2. 阅读下列程序，写出执行结果

```
#include<iostream.h>
void main()
{double x=123.456;
cout.width(10);
cout.setf(ios::dec,ios::basefield);
cout<<x<<endl;
cout.setf(ios::left);
cout<<x<<endl;
cout.width(15);
cout.setf(ios::right);
cout<<x<<endl;
cout.setf(ios::showpos);
cout<<x<<endl;
cout<<-x<<endl;
cout.setf(ios::scientific);
cout<<x<<endl;
}
```

- 选做题目：设计一道程序，程序中要求分别使用“>>、cin.get()、cin.getline()”等来输入一行字符串，并统计字符的个数。（注意：字符串中可能包含空格、或制表符等符号）。

重 要 通 知

- 5月2号课程暂停，5月9号照常上课。请各位同学相互转达。