计算机程序设计基础(1)
03 数据类型（下）

清华大学电子工程系

杨昉

E-mail: fangyang@tsinghua.edu.cn

上节内容

- 计算机**数据**表示
  - 计算机计数制：数制及进制转换，定点整数，定点小数
  - 计算机中数的表示：**原码**、**反码**、**补码**、**偏移码**
- 常量与变量介绍
- 基本数据类型**常量**
  - 整型常量
  - 实型（浮点型）常量：**舍入误差**
  - 字符型常量：ASCII表，转义符

本节目录

基本数据类型变量的定义

3.1 基本数据类型变量的定义

- 整型常量的定义
- 实型(符点型)常量的定义
- 字符型常量的定义

基本数据类型变量：整型变量

- 基本整型变量
  - int 变量表列;
- 长整型变量
  - long [int] 变量表列;　　其中int可以省略
- 短整型变量
  - short [int] 变量表列;　　其中int可以省略
- 无符号整型
  - unsigned [int] 变量表列;　　其中int可以省略
  - 也可以是： unsigned long 或 unsigned short

```
1  int a;
2  long b;
3  short c;
4  unsigned d;
5  unsigned long e;
6  unsigned short f;
```

基本数据类型变量：整型变量

- 定义变量
  - 一个类型说明语句可以定义多个同类型变量，各变量间用逗号“,”分隔多个同类型的变量也可以用多个类型说明语句定义
  - C语言允许在定义变量的同时为变量**赋初值**
- 例3-1：阅读下列C程序

```
x=-65535 y=  -255 z=    -1
x=-65535 y=  -255 z=    -1
请按任意键继续. . .
```

```
1  #include <stdio.h>
2  void main(){
3    long x, y, z;
4    x = -0xffffL; y = -0xffL; z = 0xffffffffL;
5    printf("x=%6ld y=%6ld z=%6ld\n", x, y, z);
6    x = -0xffff; y = -0xff; z = 0xffffffff;
7    printf("x=%6ld y=%6ld z=%6ld\n", x, y, z);
8  }
```

在为长整型变量初始化或赋值时，如果被赋数据为基本整型常量，则C编译系统自动将被赋数据转换成与相应变量的**类型一致**

基本数据类型变量：整型变量

- 定义变量
  - 类型说明语句定义的变量为变量分配了**存储空间**，在未对变量赋值前，这些变量中（即存储空间中）的值是**随机的**
  - 各种整型变量所占字节数有限，所能存放的整数有一定**范围**
- 例3-2：阅读下面C程序

```
1  #include <stdio.h>
2  void main(){
3     short x = 65535;
4     unsigned short y = 65535;
5     long z = 65535;
6     printf("x=%d\n", x);
7     printf("y=%u\n", y);
8     printf("z=%ld\n", z);
9  }
```

```
x=-1
y=65535
z=65535
请按任意键继续. . .
```

%d为基本整型输出格式说明符
%u为无符号基本整型输出格式说明符
%ld为长整型输出格式说明符

计算机中二进制表示：
x 1111111111111111
y 1111111111111111
z 00000000000000001111111111111111

课堂练习

- 练习1：修改例3-2中的65535为75535，请写出代码运行结果
  - 提示：short型和unsigned short型变量分别可表示范围-32768~32767和0~65535

```
#include <stdio.h>
void main(){
   short x = 75535;
   unsigned short y = 75535;
   long z = 75535;
   printf("x=%d\n", x);
   printf("y=%u\n", y);
   printf("z=%ld\n", z);
}
```

对于短整型变量x来说，2个字节存放不下75535的二进制值，而只能存放后16位的二进制值，其十进制值为9999

```
x=9999
y=9999
z=75535
请按任意键继续. . .
```

基本数据类型变量：实型变量

- **单精度**实型变量
  - 定义形式：float 变量列表;
  - 具有6～7位有效数字（23位尾数转换为10进制）
  - 分配了4B的存储空间
- **双精度**实型变量
  - 定义形式： double 变量列表;
  - 具有15～16位有效数字（52位尾数转换为10进制）
  - 分配了8B的存储空间

基本数据类型变量：实型变量

●例3-3：阅读下面C程序

```
1  #include <stdio.h>
2  void main(){
3    float a = 1.0f, b = 2.5f, c;
4    double d = 1.0, e, f;
5    c = 755.35f;
6    e = 5.35e-5;
7    f = 1e2;
8    printf("a=%f\n", a);
9    printf("b=%f\n", b);
10   printf("c=%f\n", c);
11   printf("d=%f\n", d);
12   printf("e=%f\n", e);
13   printf("f=%f\n", f);
14 }
```

```
a=1.000000
b=2.500000
c=755.349976
d=1.000000
e=0.000053
f=100.000000
请按任意键继续. . .
```

基本数据类型变量：字符型变量

- 字符型变量用于存放**字符型常量**
  - 定义形式： char 变量表列;
  - C编译系统为字符型变量分配一个字节，存放字符的**ASCII码**
- 例3-4：阅读下面C程序

```
x=A
y=B
y=66
请按任意键继续. . .
```

```
1  #include <stdio.h>
2  void main(){
3     int x;
4     char y;
5     x = 65;
6     y = 'B' ;
7     printf("x=%c\n", x);
8     printf("y=%c\n", y);
9     printf("y=%d\n", y);
10 }
```

基本数据类型变量：字符型变量

●字符型变量

□一个int型数据占4B，而字符型数据占1B，因此，在将整型数据以字符形式输出时，只取低字节中的数据作为ASCII码字符输出

```
1  #include <stdio.h>
2  void main(){
3     int x;
4     x = 1348; /* 0x544 */
5     printf("x=%c\n", x);
6  }
```

```
x=D
请按任意键继续. . .
```

由于C语言中的字符数据与整数之间可以通用，因此，字符型数据与整型数据之间可以进行混合运算

可以将字符型数据赋给整型变量，也可以将整型数据赋给字符型变量

基本数据类型变量：字符型变量

●例3-5：英文大小写字母的转换

□每一个英文小写字母比它相应的大写字母的ASCII码大32

```
1  #include <stdio.h>
2  void main(){
3    char x, y, c1, c2;
4    x = 'A' ;
5    y = 'B' ;
6    c1 = x + 32;
7    c2 = y + 32;
8    printf("x=%c, y=%c\n", x, y);
9    printf("c1=%c, c2=%c\n", c1, c2);
10 }
```

```
x=A, y=B
c1=a, c2=b
请按任意键继续. . .
```

课堂练习

练习2：如果并不知道大写字母和小写字母ASCII码之间的关系，还有没有办法实现大小写的转换？

```
1  #include <stdio.h>
2  void main()
3  {
4    char x, y, c1, c2;
5    x = 'a';
6    y = 'b';
7    c1 = x + 'C' - 'c';
8    c2 = y + 'C' - 'c';
9    printf("x=%c, y=%c\n", x, y);
10   printf("c1=%c, c2=%c\n", c1, c2);
11 }
```

提示：考虑字符型数据和整型数据间的相互转化

```
x=a, y=b
c1=A, c2=B
请按任意键继续. . .
```

基本变量类型总结

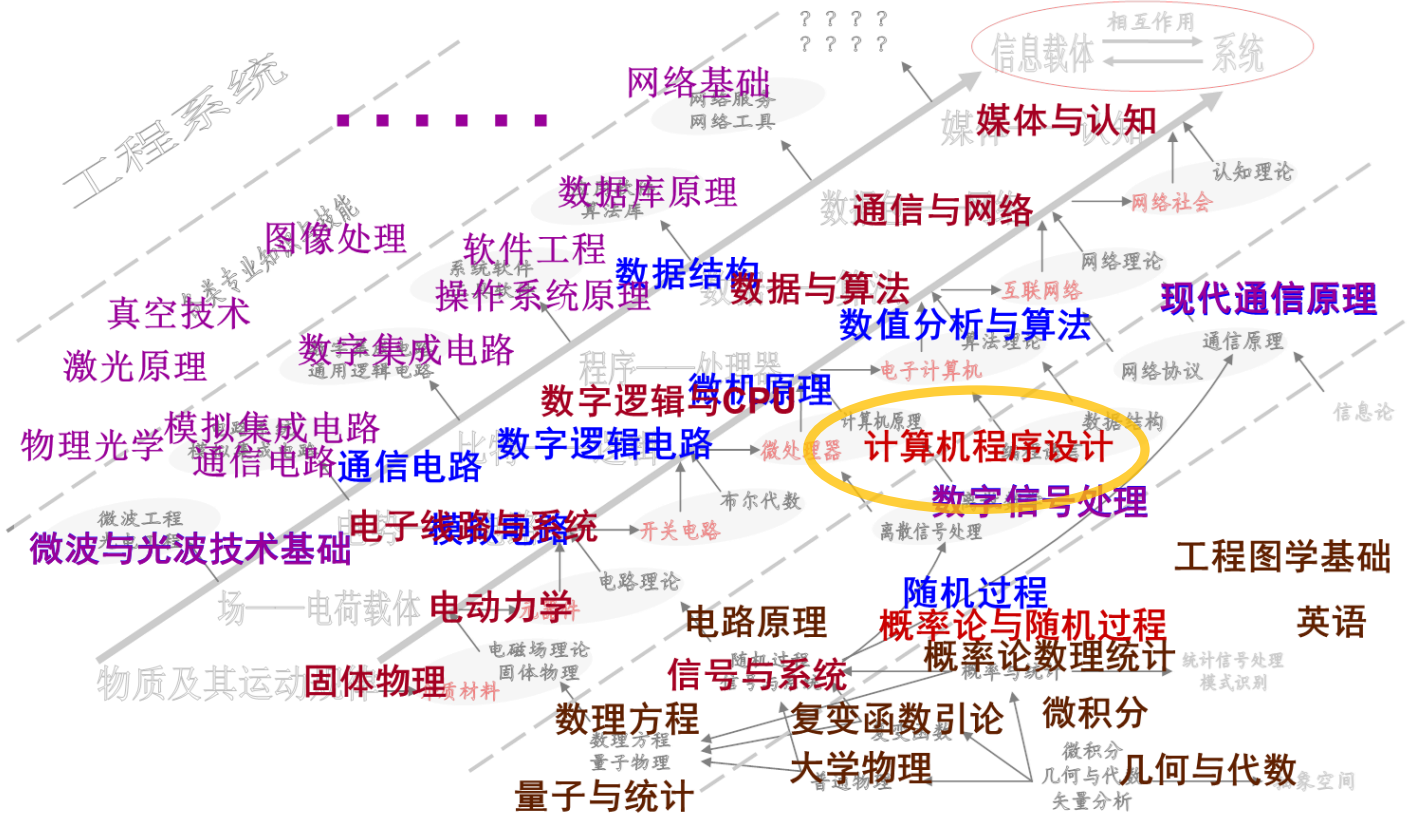
类型		16位系统		32位系统	
		存储空间	取值范围	存储空间	取值范围
整型	short	2B	-32768~32767 (-2^{15}~$2^{15}-1$)	2B	-32768~32767 (-2^{15}~$2^{15}-1$)
	unsigned short	2B	0~65535 (0~$2^{16}-1$)	2B	0~65535 (0~$2^{16}-1$)
	int	2B	-32768~32767 (-2^{15}~$2^{15}-1$)	**4B**	-2^{31}~$2^{31}-1$
	unsigned int	2B	0~65535 (0~$2^{16}-1$)	**4B**	0~4294967295 (0~$2^{32}-1$)
	long	4B	-2^{31}~$2^{31}-1$	4B	-2^{31}~$2^{31}-1$
	unsigned long	4B	0~4294967295 (0~$2^{32}-1$)	4B	0~4294967295 (0~$2^{32}-1$)
	long long	8B	-2^{63}~$2^{63}-1$	8B	-2^{63}~$2^{63}-1$

基本变量类型总结

类型		16位系统		32位系统	
		存储空间	取值范围	存储空间	取值范围
字符型	char	1B	-128~127 (-2^7~2^7-1)	1B	-128~127 (-2^7~2^7-1)
	unsigned char	1B	0~255 (0~2^8-1)	1B	0~255 (0~2^8-1)
实型	float	4B	-10^{38}~10^{38}	4B	-10^{38}~10^{38}
	double	8B	-10^{308}~10^{308}	8B	-10^{308}~10^{308}

- 注意：处理**中文**信息要使用**unsigned char**类型，GBK表用两个unsigned char存放一个汉字

为什么要学习基本变量类型?



- 计算机程序设计对于后续的专业课学习具有**基础性**的作用
- 在程序设计中，基本变量类型又具有**基础性**的地位
- 九层之台，起于垒土——只有掌握好了基本变量类型，才能更深入学习程序设计的知识，进而更好地服务于自己的专业

sizeof 运算符

3.2 sizeof 运算符

sizeof 运算符


- 给出一个变量或某种数据类型的量在计算机内存中所占**字节数**
  - 1. 求表达式计算结果所占内存的字节数

    一般形式： sizeof (表达式) 或 sizeof 表达式
  - 2. 求某种数据类型的量所占内存的字节数

    一般形式： sizeof (类型名)
  - sizeof运算符可以出现在表达式中

    例：x = sizeof(float) - 2;
- 结构体的位段（bit-field）变量，**不能用**sizeof求所占内存的字节数。关于位段，将在后续课程中详细讲

sizeof 运算符


●例3-6：输出各种类型的量在计算机中占的字节数

```
1  #include <stdio.h>
2  void main(){
3    printf("sizeof(char)=%d\n", sizeof(char));
4    printf("sizeof(int)=%d\n", sizeof(int));
5    printf("sizeof(long)=%d\n", sizeof(long));
6    printf("sizeof(short)=%d\n", sizeof(short));
7    printf("sizeof(float)= %d\n", sizeof(float));
8    printf("sizeof(double)=%d\n", sizeof(double));
9    printf("sizeof(3)=%d\n", sizeof(3));
10   printf("sizeof(3L)=%d\n", sizeof(3L));
11   printf("sizeof(3.46)=%d\n", sizeof 3.46);
12   printf("sizeof(3.46f)=%d\n", sizeof(3.46f));
13 }
```

```
sizeof(char)=1
sizeof(int)=4
sizeof(long)=4
sizeof(short)=2
sizeof(float)= 4
sizeof(double)=8
sizeof(3)=4
sizeof(3L)=4
sizeof(3.46)=8
sizeof(3.46f)=4
请按任意键继续. . .
```

sizeof 运算符

●例3-7：输出各种类型的量在计算机中占的字节数

```
1  #include <stdio.h>
2  void main(){
3    printf("%d\n", sizeof('\n' ));
4    printf("%d\n", sizeof((char)'\n'));
5  }
```

```
4
1
请按任意键继续. . .
```

□所有整数常量是int型，浮点数常量是double型

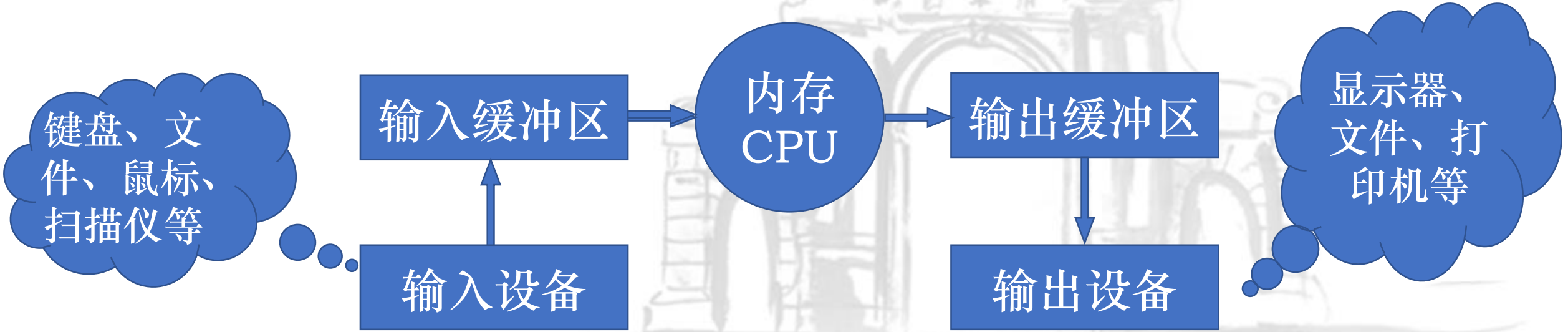
□'\n'虽然在我们看来是字符型，但对于C编译器，看到的是这个字符的ASCII值10，也就是int型10，因此sizeof('\n')的值为4

3.3 数据的输入输出函数

- 格式输出函数
- 格式输入函数
- 字符输出函数
- 字符输入函数

数据的输入输出函数

- 输出
  - 把数据从计算机内部**送到**计算机外部设备上
- 输入
  - 从计算机外部设备将数据**送入**计算机内部



数据的输入输出函数

- 数据的输入与输出应包括
  - 用于输入或输出的**设备**
  - 输入或输出数据的**格式**
  - 输入或输出的具体**内容**
- 输入与输出函数中，键盘是**标准输入设备(stdin)**，显示器是**标准输出设备(stdout)**
- **#include <stdio.h>**
  - 如果要使用C语言中的输入函数或输出函数，则在使用前（即在每个文件的开头）应该使用**包含(引用)命令**，将C语言中进行输入与输出的**库函数**和**常量变量**等说明的(头)文件包含进来

格式输出函数

●格式输出语句

□printf ("格式控制", 输出列表);

□格式控制：说明输出项目所采用的格式，要用一对双引号括起来

- 其中用于说明输出数据格式的格式说明符总是以%开头
- 后面紧跟具体格式字符

□输出列表：指出要输出的内容。

- 其中可以有多个输出项目，但各输出项目之间要用“,”分隔
- 各输出项目可以是常量、变量以及表达式
- 格式是：输出项1，输出项2，…输出项n

格式说明符与输出列表中的量一一对应，类型要一致，个数应相同

格式输出函数：整型格式说明符

进制	格式说明符	对应变量类型
十进制	%d 或 %md	int
	%ld 或 %mld	long
	%hd 或 %mhd	short
	%u 或 %mu	unsigned int
	%lu 或 %mlu	unsigned long
八进制	%o 或 %mo	int
	%lo 或 %mlo	long
十六进制	%x 或%mx	int
	%lx 或%mlx	long

- m表示输出的整型数据所占**总宽度（即列数）**
- 格式说明符中没有用m来说明数据所占的宽度，则以输出数据的实际位数为准
- 实际数据位数<m时，数据前面将用**空格补满**
- 实际数据位数>m时，以输出数据实际位数进行输出，**自动突破场宽限制**

格式输出函数：实型格式说明符

数据形式	格式说明符
十进制数形式	%f 或 %m.nf
指数形式	%e 或 %m.ne
	%E 或 %m.nE

- m表示整个数据所占宽度，n表示小数点后面所占位数
- 省略m和n，则%f, %e或%E将输出**6位小数**(不包括小数点)，不足补0
- 在小数点后取n位后，所规定的数据宽度m不够输出数据前面的整数部分（包括小数点），则按实际位数输出，**自动突破场宽限制**
- 在C语言中，用于输出单精度与双精度实型**数据格式说明符相同**

格式输出函数：字符型格式说明符

- 字符型格式说明符：%c 或%mc
  - 其中m表示**输出宽度**，即在这种情况下，在输出字符的前将会补m-1个空格
- 例3-8：阅读以下程序

```
1  #include <stdio.h>
2  void main(){
3    int a, b;
4    float x, y, s;
5    a = 34;  b = -56;
6    x = 2.5f; y = 4.51f;    s = x * x + y * y;
7    printf("a=%d,b=%d\n", a, b);
8    printf("x=%6.2f,y=%6.2f,s=%6.2f\n", x, y, s);
9  }
```

去掉f会怎么样?

```
a=34,b=-56
x=  2.50,y=  4.51,s= 26.59
请按任意键继续. . .
```

两个空格　两个空格　一个空格

格式输出函数：案例

- 例3-9：阅读以下程序

```
1  #include <stdio.h>
2  void main(){
3    int a, b;
4    float x, y, s;
5    a = 34;  b = -56;
6    x = 2.5; y = 4.51;  s = x * x + y * y;
7    printf("a=%d, b=%d\n", a, b);
8    printf("x=%6.2f, y=%6.2f, s=%6.2f\n", x, y, s);
9  }
```

warning C4305: “=”：从“double”到“float”截断

Why?

- C语言默认所有浮点常数是double类型，以便以可能的最大精度存储数据
- 因此2.5, 4.51都会以double格式存储，当赋值给x, y这些float型变量时，需要把double类型数据截断成float类型，从15位有效数字截断为7位有效数字，有精度损失

解决方案

- 2.5，4.51以float格式存储
  - x=2.5f; y=4.51f;
- 强制转换成float类型
  - x=(float)2.5; y= (float)4.51;
- x,y定义为double类型
  - double x,y;

格式输出函数：执行过程


- 1. 在计算机内存中开辟**输出缓冲区**，存放输出项目表中各项目数据
- 2. 依次计算项目表中各项目（**常量**或**变量**或**表达式**）的值，并按各项目数据类型应占的字节数依次将它们存入输出缓冲区中
- 3. 根据**“格式控制”**字符串中的各格式说明符依次从输出缓冲区中取出若干字节的数据（**如果是非格式说明符，则将按原字符输出**），转换成对应的十进制数据进行输出。
  - 其中从输出缓冲区中取多少个字节的数据是按照对应格式说明符说明的数据类型

格式输出函数：案例

- 例3-10：阅读以下程序

```
1  #include <stdio.h>
2  void main(){
3    int xx, yy, zz;
4    xx = 1; yy = -65535; zz = 1;
5    printf("xx=%ld, yy=%ld, zz=%ld\n", xx, yy, zz);
6    printf("xx=%hd, yy=%hd, zz=%hd\n", xx, yy, zz);
7    printf("xx=%d, yy=%d, zz=%d\n", xx, yy, zz);
8  }
```

```
xx=1, yy=-65535, zz=1
xx=1, yy=1, zz=1
xx=1, yy=-65535, zz=1
请按任意键继续. . .
```

格式说明符	对应变量类型
%d 或 %md	int
%ld 或 %mld	long
%hd 或 %mhd	short

格式输出函数：案例

- 例3-11：阅读以下程序

```
1  #include <stdio.h>
2  void main(){
3     double x = 34.567;
4     printf("x=%f\n", x);
5     printf("x=%d\n", x);
6     printf("x=%d\n", (int)x);
7  }
```

强制转化

第二行输出错误，原因是格式说明符与输出项目类型不匹配

```
x=34.567000
x=1958505087
x=34
请按任意键继续. . .
```

强制转化时不四舍五入

格式输出函数：输出左对齐

- 输出左对齐
  - 输出的数字默认是右对齐，如果在宽度说明前加一个**负号(-)**，则输出为**左对齐**，即在右边补空格
- 例3-12：修改例3-8，在输出格式说明符中加负号(-)

```
1  #include <stdio.h>
2  void main(){
3    int a, b;
4    float x, y, s;
5    a = 34; b = -56;
6    x = 2.5f; y = 4.51f; s = x * x + y * y;
7    printf("a=%d, b=%d\n", a, b);
8    printf("x=%-6.2f,y=%-6.2f,s=%-6.2f\n",x,y,s);
9  }
```

修改前

```
a=34,b=-56
x=  2.50,y=  4.51,s= 26.59
请按任意键继续. . .
```

修改后

```
a=34,b=-56
x=2.50  ,y=4.51  ,s=26.59
请按任意键继续. . .
```

格式输出函数：格式字符

格式字符	说明
c	输出一个字符
d或i	输出带符号的十进制形式整型数 %ld为长整型，%hd为短整型
o	输出八进制形式整型数 %o不带先导0；%#o加先导0
x或X	输出十六进制形式整型数，但不带先导0x或0X %#x或%#X输出带先导0x或0X的十六进制数
u	输出无符号十进制形式整型数
f	以带小数点的数学形式输出浮点数（单精度和双精度数）

格式输出函数：格式字符

格式字符	说明
e或E	以**指数形式输出浮点数(单精度和双精度数)** 格式是：[-]m.dddddd[e/E]±xxx 小数位数(d的个数)由输出精度决定，隐含是6 若指定的精度为0，则连小数点在内的小数部分都不输出
g或G	在满足输出精度的情况下，由系统决定用%f还是用%e(或%E)输出,未指定输出精度则默认6位有效数字
s	输出一个**字符串**，直到遇到"\0" 若字符串长度超过指定的精度则自动突破，不会截断字符串
p	**输出变量的内存地址**
%	输出一个%

格式输出函数：格式字符

- 格式说明都必须用“%”开头，以一个**格式字符**作为结束
  - 可插入**宽度说明**“m”、**左对齐符号**“-”、**前导零符号**“0”等
- 长度修饰符
  - 在%和格式字符之间，可以加入**长度修饰符**
  - 对于长整型数（long）应该加**l**，如%ld
  - 对于短整型数（short）可以加**h**，如%hd
- 格式字符
  - 在某些系统中，可能不允许使用大写字母的格式字符
  - 因此为了使程序具有通用性，**尽量不用大写字母的格式字符**

格式输出函数：输出宽度

- 不指定输出宽度
  - 按照数据实际宽度输出，前后不加空格，采用右对齐的形式
- 人为控制输出数据宽度
  - 在%和格式字符之间**插入整数常数**来指定输出宽度m（例如%4d）
  - 如果**m<数据实际宽度**，输出时将会**自动突破宽度**
  - 如果**m>数据实际宽度**，输出时将会右对齐，左边补空格，达到指定宽度

```
1  #include <stdio.h>
2  void main(){
3    int a = 10000;
4    printf("%4d\n", a);
5    printf("%8d\n", a);
6  }
```

```
10000
  10000
请按任意键继续...
```

格式输出函数：输出宽度


- 对于float和double
  - 可用“m.n”的形式指定输出宽度（m和n分别是一个整常数），其中m指定输出数据宽度（包括小数点），n指定小数点后小数位位数，n也称为精度（例如%12.4f）
  - 对于f、e或E，当输出数据的小数位多于n位时，截去右边多余的小数，并对截去部分的第一位小数做四舍五入处理
  - 当输出数据的小数位少于n时，在小数的最右边补0，使得输出数据的小数部分宽度为n

```
1  #include <stdio.h>
2  void main() {
3    double a = 3.456;
4    printf("%8.2f\n", a);
5    printf("%8.5f\n", a);
6  }
```

```
    3.46
3.45600
请按任意键继续...
```

格式输出函数：输出宽度


- 对于float和double
  - 若给出的总宽度m小于n加上整数位数和小数点（e或E格式还要加上指数的5位），则自动突破m的限制;
  - 反之，数字右对齐，左边补空格
  - 可以用“.n”格式（例如%.6f），仅指定小数部分输出位数
  - 如果指定“m.0”或“.0”格式（例如%12.0f或%.0f），则不输出小数点和小数部分

```
1  #include <stdio.h>
2  void main() {
3    double a = 3.456;
4    printf("%4.4f\n", a);
5    printf("%.5f\n", a);
6    printf("%.0f\n", a);
7  }
```

```
3.4560
3.45600
3
请按任意键继续...
```

格式输出函数：输出宽度

- 对于g或G
  - 可以用 %m.ng 或 %m.nG的形式来输出,其中m指定输出数据宽度（包括小数点），n指定输出的有效数字位数
  - 宽度>数字的有效数字位数，则左边自动补空格;
  - 宽度不足，则自动突破
  - 若%g、%G、%mg、%mG形式不指定输出有效数字位数，将按照最多6位有效数字输出，按照四舍五入原则截去右边多余小数
  - 若指定或默认的输出有效数字位数超过实际数字的有效位数，则把实际数字原样输出

格式输出函数：输出宽度

- 对于整型数
  - 若输出格式是“0n_1”或“.n_2”格式（例如%05d或%.5d），如果指定的宽度超过输出数据的实际宽度，输出时将会右对齐，左边补0

```
1  #include <stdio.h>
2  void main() {
3    int a = 10;
4    printf("%05d\n", a);
5    printf("%.5d\n", a);
6  }
```

```
00010
00010
请按任意键继续...
```

格式输出函数：输出宽度

- 对于字符串
  - 格式“n”指定字符串的输出宽度
  - 若n<字符串实际长度，则自动突破，输出整个字符串
  - 若n>字符串实际长度，则右对齐，左边补空格
  - 若用“.n”格式指定字符串输出宽度，则若n<字符串的实际长度，将只输出字符串的前n个字符(**将字符串截断！**)

```
1  #include <stdio.h>
2  void main() {
3    char a[12] = "hello world";
4    printf("%6s\n", a);
5    printf("%15s\n", a);
6    printf("%.5s\n", a);
7  }
```

```
hello world
    hello world
hello
请按任意键继续. . .
```

格式输出函数：输出精度

- 输出数据的实际精度取决于
  - 格式控制中的**域宽**
  - 数据在计算机内的**存储精度**
  - 通常系统保证float类型有7位有效数字，double类型有15位有效数字
  - 若指定域宽和小数域宽**超过**相应类型数据的有效数字，则输出的多余数字没有意义

格式输出函数：符号位

- 使输出数据总带+号或-号
  - 输出数据若为负数，前面有符号位“-”，但正数的“+”被省略
  - 如果要每一个数前面都带正负号，可以在%和格式字符间加一个“+”号来实现
- 各种情况下的输出示例
  - int k=1234; double f =123.456;

```
1  printf("%d\n", k);
2  printf("%6d\n", k);
3  printf("%2d\n", k);
4  printf("%f\n", f);
5  printf("%12f\n", f);
6  printf("%12.6f\n", f);
7  printf("%2.6f\n", f);
8  printf("%.6f\n", f);
9  printf("%12.2f\n", f);
```

```
1234
 1234
1234
123.456000
  123.456000
  123.456000
123.456000
123.456000
      123.46
```

格式输出函数：案例

- 各种情况下的输出示例
  - int k=1234; double f =123.456;

```
1  printf("%12.0f\n", f);
2  printf("%.f\n", f);
3  printf("%e\n", f);
4  printf("%13e\n", f);
5  printf("%13.8e\n", f);
6  printf("%3.8e\n", f);
7  printf("%.8e\n", f);
8  printf("%13.2e\n", f);
9  printf("%13.0e\n", f);
10 printf("%.0e\n", f);
11 printf("%g\n", f);
12 printf("%5g\n", f);
13 printf("%10g\n", f);
14 printf("%g\n",123.456789);
```

```
 123
123
1.234560e+02
 1.234560e+02
1.23456000e+02
1.23456000e+02
1.23456000e+02
    1.23e+02
       1e+02
1e+02
123.456
123.456
  123.456
123.457
```

```
1  printf("%06d\n", k);
2  printf("%.6d\n", k);
3  printf("%012.6f\n", f);
4  printf("%013.2e\n", f);
5  printf("%s\n", "abcdefg");
6  printf("%10s\n", "abcdefg");
7  printf("%5s\n", "abcdefg");
8  printf("%.5s\n", "abcdefg");
9  printf("%-6d\n", k);
10 printf("%-12.2f\n", f);
11 printf("%-13.2e\n", f);
12 printf("%+-6d%+-12.2f\n",k,-f);
13 printf("%4.1f%%\n", 12.5);
```

```
001234
001234
00123.456000
000001.23e+02
abcdefg
   abcdefg
abcdefg
abcde
1234
123.46
1.23e+02
+1234 -123.46
12.5%
```

格式输出函数：printf注意事项


- 1. printf的输出格式为自由格式，是否在两个数之间留**逗号**、**空格**或**回车**，完全取决于格式控制

```
1  printf("%d%d%f\n", k, k, f);
2  printf("%d %d %f\n", k, k, f);
```

```
12341234123.456000
1234 1234 123.456000
```

- 2. 格式说明符与输出列表中的量必须**一一对应**，**类型匹配**。否则不能正确输出，且编译时不会报错
  - 格式说明个数少于输出项个数，则多余的输出项不予输出
  - 格式说明个数多于输出项个数，则将输出一些毫无意义的数字乱码
- 3. printf函数有返回值，返回值是**本次调用输出字符的个数**，包括回车等控制符

格式输出函数：printf注意事项

- 4. 不要在输出语句中改变输出变量的值，会造成输出结果**不确定**
  - C语言国际标准中的未定义行为（undefined behavior）

```
1  int k = 8;
2  printf("%d,%d\n", k, ++k);
```

```
9,9
请按任意键继续. . .
```

  - 调用函数printf时，其参数是从右至左进行传递的，将先进行++k运算
- 5. 变场宽输出
  - 将按照m指定的场宽输出i的值，并不输出m的值

printf("%*d", m, i);

  - 按照m和n指定的场宽输出浮点型变量f的值，并不输出m、n的值

printf("%*.*f", m, n, f);

格式输入函数：输入语句


- 格式输入语句

scanf ("格式控制",内存地址表);

- 格式控制：说明输入数据所采用的格式，要用一对**双引号**括起来
- 格式控制中用于说明输入数据格式的**格式说明符**总是以%开头，后面紧跟具体**格式字符**
- 输入列表：指出各输入数据所存放的**内存地址**。其中可以有多个输入项目，但各输出项目之间要用“,”分隔
- **取地址运算符&**
  - 例如，&a表示变量a在内存中的地址

格式输入函数：格式说明符

进制	格式说明符	对应变量类型
十进制	%d 或 %md	int
	%ld 或 %mld	long
	%hd 或 %mhd	short
	%u 或 %mu	unsigned int
	%lu 或 %mlu	unsigned long
八进制	%o 或 %mo	int
	%lo 或 %mlo	long
十六进制	%x 或%mx	int
	%lx 或%mlx	long

- m表示输入的整型数据所占**总宽度（即列数）**
- 与输出情形一样，对于八进制形式与十六进制形式的输入格式，主要用于输入无符号整型的数据

格式输入函数：格式说明符

数据形式	格式说明符
单精度实型	%f 或 %e
双精度实型	%lf 或 %le
字符型	%c

- 在用于输入时，无论是单精度实型还是双精度实型，都**不能**用m.n来指定输入的宽度和小数点后的位数

- 例3-13：格式输入

```
1  #include <stdio.h>
2  void main(){
3    int a;
4    float b;
5    char c;
6    scanf("%d%f%c", &a, &b, &c);
7  }
```

取地址运算符&

格式输入函数：说明

- 当用于输入整型数据的格式说明符中**没有宽度说明**时
  - 如果各格式说明符之间没有其他字符，则在输入数据时，两个数据之间用"**空格**"、"**Tab**"、"**回车**"来分隔
  - 如果各格式说明符之间包含其他字符，则在输入数据时，应输入与**这些字符相同的字符**作为间隔

格式输入函数：案例


●例3-14：设有如下定义

int a, b; float c, d;

现要利用格式输入函数输入数据，使得：

a=12, b=78, c=12.5, d=7.6

(1) 格式说明符中**没有宽度说明**，各格式说明符之间**没有其他字符**

□两个数据之间用空格来分隔，也可用“Tab”或“回车”来分隔

```
1  #include <stdio.h>
2  void main(){
3    int a, b;
4    float c, d;
5    scanf("%d%d%f%f",&a,&b,&c,&d);
6  }
```

```
12 78 12.5 7.6
请按任意键继续. . .
```

格式输入函数：案例

●例3-14：设有如下定义

int a, b; float c, d;

现要利用格式输入函数输入数据，使得：

a=12, b=78, c=12.5, d=7.6

(2) 格式说明符中**没有宽度说明**，各格式说明符之间**有逗号**

□两个数据之间输入格式控制中对应的**逗号**

```
1  #include <stdio.h>
2  void main(){
3    int a, b;
4    float c, d;
5    scanf("%d, %d, %f, %f", &a, &b, &c, &d);
6  }
```

```
12, 78, 12.5, 7.6
请按任意键继续. . .
```

格式输入函数：案例

●例3-14：设有如下定义

int a, b; float c, d;

现要利用格式输入函数输入数据，使得：

a=12, b=78, c=12.5, d=7.6

(3) 格式说明符中**没有宽度说明**，各格式说明符之间**有其他字符**

□两个数据之间输入格式控制中**对应的字符**

```
1  #include <stdio.h>
2  void main(){
3    int a, b;
4    float c, d;
5    scanf("a=%d, b=%d, c=%f, d=%f",&a,&b,&c,&d);
6  }
```

```
a=12, b=78, c=12.5, d=7.6
请按任意键继续. . .
```

格式输入函数：宽度说明

- 当格式说明符中有宽度说明时，**按宽度说明截取数据**
  - 例3-15：阅读以下代码

```
1  #include <stdio.h>
2  void main(){
3    int a, d;
4    char b, c;
5    printf("input a, b, c, d:");
6    scanf("%3d%3c%2c%2d", &a, &b, &c, &d);
7    printf("a=%d, b=%c, c=%c, d=%d\n",a,b,c,d);
8  }
```

```
input a, b, c, d:1234567890123456
a=123, b=4, c=7, d=90
请按任意键继续. . .
```

3d 3c 2c 2d

1234567890123456

  - 按照3c格式把字符456给b赋值，虽然字符‘4’赋值给了b，但随后的字符5和6放到了b之后的内存中，造成了**内存越界**，引起致命错误
  - 不能使用**%mc** (m>1)格式给char型变量读入字符值

格式输入函数：宽度说明

- 当格式说明符中有宽度说明时，**按宽度说明截取数据**
  - 例3-16：阅读右边代码

```
1  #include <stdio.h>
2  void main(){
3     int k;
4     float a;
5     double y;
6     scanf("%3d%5f%5le", &k, &a, &y);
7     printf("%d %.4f %.4f\n", k, a, y);
8  }
```

```
123456.789.123
123 456.7000 89.1200
请按任意键继续. . .
```

  - 除非数字一开始就是“**粘联**”在一起，否则不提倡指定输入数据所占的宽度

格式输入函数：执行过程


- 1. C程序开始执行时，系统就在计算机内存中开辟了一个**输入缓冲区**，用于暂存从键盘输入的数据。开始时该输入缓冲区是空的
- 2. 执行到一个输入函数时，就检查输入缓冲区中是否有数据：
  - 若输入缓冲区**没有数据**，则等待用户从键盘输入数据并依次存放到输入缓冲区中
  - 若输入缓冲区**已经有数据**（上一个输入函数读剩下的），则依次按照“格式控制”中的格式说明符从输入缓冲区中取出数据
  - 当输入一个<回车>或<换行>符后，将依次按照“格式控制”中还未用过的格式说明符从输入缓冲区中取出数据
- 3. 数据被转换成计算机中的表示形式（**二进制**），最后存放到内存地址表中指出的对应地址中

格式输入函数：注意事项

- 从键盘输入数据是以**<回车>**或**<换行>**作为结束符的
  - 当输入的数据一行不够时，可以在下一行**继续输入**
  - 当一行上的数据用不完时，可以留给**下一个输入函数**使用
- 在输入函数的"格式控制"中，最后**不能**加换行符'\n '
  - **<回车>**或**<换行>**是作为键盘输入数据的**结束符**

格式输入函数：格式字符

格式字符	说明
c	输入一个字符
d	输入带符号的十进制整型数
i	输入整型数，整型数可以是带先导0的八进制数，也可以是带先导0x(或0X)的十六进制数
o	输入八进制格式整型数，可以带先导0，也可以不带
x	输入十六进制格式整型数,可以带先导0x或0X,也可以不带
u	输入无符号十进制形式整型数
f (lf)	输入以小数形式浮点数（单精度用f，双精度数用lf)
e (le)	输入以数学形式或指数形式浮点数（单精度用e，双精度数用le)
s	输入一个字符串，直到遇到"\0“ 若字符串长度超过指定的精度则自动突破，不会截断字符串

格式输入函数：注意事项

- **输入项**与**格式说明符**必须一一对应，否则不能正确输入，而且编译时**不会报错**
  - 若格式说明符个数少于输入项个数，scanf函数结束输入，则多余的输入项将**无法得到**正确的输入值
  - 若格式说明符个数多于输入项个数，scanf函数结束输入。多余的数据作废，不会作为**下一个**输入语句的数据
- **输入数据**少于**输入项**时，运行程序**等待输入**，直到满足要求为止
- **输入数据**多于**输入项**时，多余的数据在输入流中没有作废，而是**等待下一个**输入操作语句继续从此输入流读取数据
- scanf函数返回值是本次scanf调用**正确输入的数据项的个数**

格式输入函数：键盘输入

●1．输入数值数据

- 输入整数或实数这类数值型数据时，输入的数据间必须用空格、回车符、制表符（TAB键）等间隔符隔开，间隔符个数不限。即使在格式说明中人为指定了输入宽度，也可以用此方式输入
- 用scanf函数从键盘输入数据时，每行数据在未按下回车（Enter）之前，可以任意修改。按下回车后，数据就送入了输入缓冲区，不能再修改

```
1  #include <stdio.h>
2  void main(){
3     int k;
4     float a;
5     double y;
6     scanf("%d%f%le", &k, &a, &y);
7  }
```

```
10 12.3 1234567.89
请按任意键继续. . .
```

```
10
12.3
1234567.89
请按任意键继续. . .
```

格式输入函数：键盘输入

- 2. 跳过某个输入数据
  - 在%和格式字符之间加入“*”号
- 例3-17：阅读以下代码

```
1  #include <stdio.h>
2  void main() {
3     int x, y, z;
4     scanf("%d%*d%d%d", &x, &y, &z);
5     printf("%d %d %d\n", x, y, z);
6  }
```

34被跳过

```
12 34 56 78
12 56 78
请按任意键继续. . .
```

格式输入函数：键盘输入


- 3.在格式控制字符串中插入其它字符
  - scanf函数中的格式控制字符串是为了输入数据用的，无论其中有什么字符，也**不会输出**到屏幕上
- 例3-18：阅读以下代码

```
1  #include <stdio.h>
2  void main() {
3    int x, y, z;
4    scanf("Please input x,y,z: %d%d%d", &x, &y, &z);
5    printf("%d %d %d\n", x, y, z);
6  }
```

```
Please input x,y,z: 12 34 56
12 34 56
请按任意键继续. . .
```

  - 需全部手动输入，包括Please input x,y,z:、字符的大小写、字符间的间格等必须与scanf中的**完全一致**。这些字符又被称为**通配符**

格式输入函数：键盘输入

- 3.在格式控制字符串中插入其它字符

使用printf进行输入提示

```
1  #include <stdio.h>
2  void main() {
3     int x, y, z;
4     printf("Please input x,y,z: ");
5     scanf("%d%d%d", &x, &y, &z);
6     printf("%d %d %d\n", x, y, z);
7  }
```

```
Please input x,y,z: 12 34 56
12 34 56
请按任意键继续. . .
```

```
Please input x,y,z: 12,34,56
12 34 56
```

✓

空格是间隔符，将全部被忽略掉

```
Please input x,y,z: 12, 34, 56
12 34 56
```

✓

格式控制中加入逗号

```
1  scanf("%d,%d,%d", &x, &y, &z);
```

逗号没有紧跟在输入数据后面

```
Please input x,y,z: 12 , 34 , 56
12 -858993460 -858993460
```

✗

字符输出函数

- putchar(c)
  - 在屏幕显示的当前光标位置处输出项目c所表示的一个字符
  - c可以是字符型常量、字符型变量、整型变量或整型表达式
  - 执行过程与格式输出函数的执行过程完全相同
  - putchar( )也可以输出转义字符，ASCII中59为‘;’

```
1  #include <stdio.h>
2  void main(){
3    int x = 68;
4    char y = 'B' ;
5    putchar('A' );
6    putchar(y);
7    putchar(67);
8    putchar(x);
9    putchar(34 + 25);
10 }
```

```
ABCD;请按任意键继续. . .
```

```
1  #include <stdio.h>
2  void main(){
3    int x = 68;
4    char y = 'B' ;
5    putchar('A');   putchar('\n');
6    putchar(y);     putchar('\n');
7    putchar(67);    putchar('\n');
8    putchar(x);     putchar('\n');
9    putchar(34+25); putchar('\n');
10 }
```

```
A
B
C
D
;
请按任意键继续. . .
```

字符输入函数

- getchar(c)
  - 接收从键盘输入的一个字符
  - 在执行字符输入函数时，由键盘输入的字符依次存放在输入缓冲区中，同时也在屏幕上显示，并且以回车结束
  - 一个字符输入函数只顺序接收一个字符，输入缓冲区中剩下的字符数据（包括回车符）将留给下面的字符输入函数或格式输入函数使用

```
1  #include <stdio.h>
2  void main(){
3     char x;
4     x = getchar();
5  }
```

字符输入函数

- _getch( ) 或_getche( )
  - 在按下相应键的同时接收从键盘输入的一个字符
  - 都不等待<回车>符，只要按下相应键的同时，此函数就读取(接收)了相应字符
  - 使用_getch( )时，由键盘输入的字符不在屏幕上显示
  - 使用_getche( )时，由键盘输入的字符同时在屏幕上显示

```
1  #include <conio.h>
2  //注意：这里引用的不是stdio.h
3  void main(){
4     char x;
5     x = _getch();
6  }
```

字符输入函数

- _ungetch(c)
  - 把刚从键盘接收(输入)的一个字符c回写到输入流。使得输入流看起来未读过

```
1  #include <stdio.h>
2  #include <conio.h>
3  void main(){
4     char x, y;
5     x = _getche(); //读入一个字符
6     _ungetch(x);   //将读入的字符放回输入流中
7     y = _getche(); //读入一个字符
8     putchar(y);
9  }
```

```
xx请按任意键继续. . .
```

前一个x为输入，后一个x为输出

本节总结

- 基本数据类型**变量**的定义
  - 整型（int, short, long）
  - 实型(符点型)（float, double）
  - 字符型变量（char）
- **sizeof**运算符：sizeof(int)，sizeof(233)
- 数据的输入输出函数
  - 格式输出/输入函数：printf, scanf
  - 字符输出/输入函数：putchar, getchar, _getch, _getche, _ungetch

本节作业

- 作业3
  - 课本第二章习题3
  - 课本第三章习题4,5,6,8
  - 完成后将word文档提交到网络学堂
- 附加题：
  - 输入一个浮点数a，以整型输出其四舍五入后的结果b（不要使用math.h中现成的round函数）
  - 输入任意一个字符，输出其ASCII码的十六进制表示

本节作业

- 上机实验2（示例代码见下一页）
  - 1. 编写程序打印查看97.6875和-97.6875的double型值在计算机内的IEEE 754存储格式按字节的十六进制值，验证是否与教材所给结果一致。同时也打印查看97.6875和-97.6875的float型值在计算机内存储按字节的十六进制值
  - 2. 编写程序分别打印100与-100的double、float、long、int、short、char型值在计算机内的存储按字节的十六进制值。分析结果为什么各自不同
  - 上机实验请将**运行结果截图**粘贴到**word文档**提交到网络学堂

本节作业

- 程序代码示例
  - 打印double和float型变量内部表示十六进制值，以97.6875为例

```
1  #include <stdio.h>
2  main(){
3    double x;
4    float y;
5    unsigned char *p;//
6    x = 97.6875;
7    p = (unsigned char *)&x;//
8    printf("%02X %02X %02X %02X %02X %02X %02X %02X\n",
9        *p, *(p+1), *(p+2), *(p+3), *(p+4), *(p+5), *(p+6), *(p+7));//
10   y = 97.6875;
11   p = (unsigned char *)&y;//
12   printf("%02X %02X %02X %02X\n", *p, *(p+1), *(p+2), *(p+3));//
13 }
```

程序中末尾有//的行读不懂没有关系，学完后续知识就会理解


THANKS