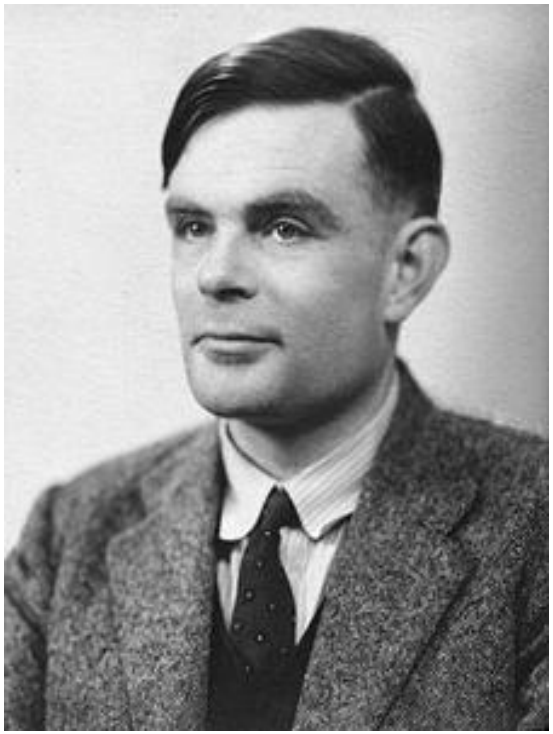

Allan Turing (阿伦•图灵)

Allan Turing（阿伦•图灵）

- 1912年生于英国伦敦
- 1931年进入剑桥大学国王学院学习。1934年毕业后，凭借其关于概率论的毕业论文，1935年被选为国王学院的研究员（Fellow of King's College）
- 1936年发表论文“On computable numbers with an application to the Entscheidungs problem”，提出图灵机“Turing Machine”模型
- 1936年前往普林斯顿大学攻读博士学位，38年毕业，导师 是阿隆佐·邱奇（Alonzo Church）
  - Thesis: Systems of Logic Based on Ordinals (基于序数的逻辑系统)
- 1939年服役，从事密码破译研究
- 1945-1948年在英国国家物理实验室，研制ACE计算机
- 此后，加盟曼切斯特大学皇家学会计算实验室，1950年发表著名论文“Computing Machine and Intelligence”（*Mind*, 59(236): 433-460），提出“图灵测试”
- 1951年被选为英国皇家学会院士
- 1952年，因同性恋遭政府刑事起诉
- 1954年，在其42岁生日前服毒自杀身亡
- 2009年9月10日，当时的英国首相戈登·布朗代表英国政府对图灵当年遭受到的不公正处置公开表示正式道歉
- 2013年，英国女王伊丽莎白二世向图灵追加了“皇家赦免令”

ACM Turing Award（图灵奖）

- 国际（美国）计算机协会（Association for Computing Machinery）于1966年设立的第一个奖项
- 宗旨：奖励在计算机科学研究中做出创造性贡献、推动计算机科学技术发展的杰出计算机科学家
- 奖金额：最初2万美金；1989年增至2万5千美金；2007年在 Intel和Google资助下，奖金提升到25万美金；从2014年开始，在Google资助下，奖金提升到100万美金
- 到2026年，总共有81位计算机科学家获得该奖项

第二讲
基本数据及其运算

（第二部分）

周水庚

2026年9月24日

提要

- 数据输入输出基础
- 实例讲解

提要

- **数据输入输出基础**
- 实例讲解

字符输出函数: putchar()

- 字符输出函数的功能
  - 函数调用putchar(ch)将实参ch对应的字符到标准输出设备上
- putchar()的使用方法
  - 调用函数时，需提供一个实参，实参可以是字符型或整型数据，包括字符型常量(包括控制字符和转义字符)，字符型变量，整型变量等

```
#include <stdio.h>
int main()
{ char ch;  int  i;
  ch = 'h';  i = 'i';
  putchar('C');  /* 输出字符 C */
  putchar(ch);  /* 输出字符 h */
  putchar(i);    /* 以字符形式输出整型变量值 */
  putchar('n'); putchar('\141');/* 输出字符 a */
  putchar('\n');
  return 0;
}
```

运行该程序将输出：China

字符输入函数：getchar()

- 从标准输入设备上读取一个字符。该函数没有参数，对它的每次调用，就返回下一个输入字符的ASCII码值
- getchar()读取字符的时候，是从stdin的“缓存”中读的。用户输入的字符被存放在键盘缓冲区中，当用户键入回车(回车字符也放在缓冲区中)之后，getchar ()才开始从stdin流中每次读入一个字符。getchar ()函数的返回值是用户输入的第一个字符的ASCII码，如出错返回-1
- 如用户在按回车之前输入了不止一个字符，其它字符会保留在键盘缓存区中，等待后续getchar ()调用读取。也就是说，后续的getchar ()调用不会等待用户按键，而直接读取缓冲区中的字符，直到缓冲区中的字符读完为后，才等待用户下一次按键
- 执行语句：“ch = getchar(); ”使变量ch得到输入字符的ASCII码值。变量ch为char型或int型

```
#include <stdio.h>
int main()
{ char c ;
  c = getchar(); /* 调用getchar()不要参数*/
  putchar(c);       /* 输出读入的字符 */
  putchar('\n');
}
```

程序运行时，如果从键盘键入字符 Z 和回车：

```
Z
```

程序输出 c 的值‘Z’：

```
Z
```

```
/*输入字符，输出字符及其代码*/
#include <stdio.h>
int main()
{ char c1, c2;
  c1 = getchar();  /* 输入一个字符    */
  c2 = getchar();  /* 再输入一个字符 */
  putchar(c1);  putchar(c2);
  printf("code1 = %d  code2 = %d\n", c1, c2);
  return 0;
}
```

```
#include<stdio.h>
void test_GetChar()
{
    char c;
    c = getchar();
    printf("1: %c\n\n", c);
    c = getchar();
    printf("2: %c\n\n", c);
    c = getchar();
    printf("3: %c\n\n", c);

    printf("end\n\n");
}
```

输入：
y+回车
y+回车

输出：
1: y

2:

3: y

end

```
#include <stdio.h>
int main(void)
{
    int c;
    while ((c = getchar()) != '\n')
    {
      printf("%c", c);
    }
    return 0;
}
```

利用getchar()函数让程序等待用户按下“回车键”（Enter键）后才退出，在调试程序的时候很有用。用法：需要停顿的地方，加上：while ((c = getchar()) != '\n');

关于字符输出

- 在输出字符串中，出现以转义符\+数字表示的字符时，转义符\将结合尽可能多的有效数字作为一个字符输出

- 譬如
  - printf(“\06711”);输出：711　/*‘7’的ASCII码为55，即067
  - printf(“\081111”);输出：　/* \0是字符串结束符
  - printf(“\1081111”);输出：81111　/* \10 表示退格符\b
  - printf(“\x7ag”);输出：zg　/* \x7a表示122，‘z’的ASCII码
  - printf(“\7gc”);输出：先嘟一下，然后输出gc

- 计算机键盘上的enter键，表示\n

printf()

- printf()函数的功能是将输出项按指定的格式排版输出到标准输出设备上
- 调用printf()函数的一般形式为：

  printf(格式控制字符串，输出项，输出项， …)
- 格式控制字符串是字符串表达式，通常是由用一对双引号括起来的字符串常量。格式控制字符串包含三类字符：普通字符、转义字符和格式转换说明
  - 普通字符，要求按原样输出
  - 转义字符，要求按转义字符的意义输出，如‘\n’，表示输出时回车换行，‘\b’表示退格等
  - 格式转换说明，以字符%开头至输出格式符结束的字符列组成

printf()（续）

- 格式转换说明的一般形式

  %[-][+][ ][#][w][.p][h/l/L]输出格式符

  - 其中，用方括号括住的内容是格式修饰说明，可以缺省，如“%d”、“%7.5f”等。每个格式转换说明对应一个输出项，输出项可以是常量、变量或表达式
  - 格式转换说明的作用是将对应输出项的内容按要求产生出字符列，并排版输出
- 例如: printf("a=%d,b=%d,a+b=%d\n", a, b, a+b)
  - 字符串“a=%d,b=%d,a+b=%d\n”为格式控制字符串
    - %d为格式转换说明，a=、,b=、,a+b=都为普通字符，\n是转义字符。格式控制字符串后的a、b、a+b为输出项。若格式控制字符串中没有格式转换说明，输出项也就不再需要

printf(格式控制字符串，输出项，输出项， …)

↓

“普通字符/转义字符/格式转换说明”

↓

%[格式修饰说明] 输出格式符

↓

[-][+][ ][#][w][.p][h/l/L]

输出格式符共有16个，有12种不同的格式

格式符	意　　义
d或i	整型数据以十进制形式输出
o	整型数据以八进制形式输出
x或X	整型数据以十六进制形式输出
u	无符号整型数据以十进制形式输出
c	字符的ASCII码数据，输出对应的字符
s	输出字符串
f	以“整数部分.小数部分”形式输出实型数据
e或E	以[-]n .nnnnne±xx输出实型数据
g或G	以f或e格式输出实型数据
p	指针值转换成一串可显示的字符输出
n	记录已输出的字符个数
%	输出一个字符%

printf()（续）

- x(或X)和o格式符把符号位作为数的一部分输出。x(或X)用字符a、b、c、d、e、f(或A、B、C、D、E、F)表示 9 之后的6个十六进制数字符
- 一个整数，只要它的值在0～255 范围内，也可以用字符形式输出，输出以该整数为ASCII代码的字符。反之，一个字符数据也可以用整数形式输出，输出该字符的ASCII代码值
- f、e(或E)和g(或G)格式符用于输出实型数据，格式转换时有四舍五入处理
- 对于f 格式，小数点后的数字个数可由格式修饰说明p指定，若p为0，不显示小数点

printf()（续）

- 用e(或E)格式输出时，对于非0实数，小数点前有一位非零数字，输出格式中的有效位数可由格式修饰说明p指定；字符e(或E)之后是指数，指数部分至少包含两个数字。若输出值的绝对值不小于 1E+100，则指数部分多于两位数字

- g（或G）格式能根据表示数据所需字符的多少自动选择f格式或e(或E)格式输出实数，选择是以输出时所需字符少为标准

- n格式符不产生输出，对应输出项是一个整型变量地址(int *)，即整型指针；printf()把本次调用截至解释该格式符为止已输出字符个数放在对应变量中

printf()（续）

- **格式修饰说明有7种: %[-][+][ ][#][w][.p][h/l/L]**
  - **w域宽说明，w 是一个十进制整数，表示输出字段的字符数。若转换后需要的字符个数比给出的宽度w多，以实际需要为准；若转换后需要的字符数比w少，就在左边用填充字符补足(若左边对齐(-)，则在右边补填充字符)。通常用空白符作填充字符，若十进制整数w有前导0(此0不表示以八进制数给出字段宽度)，则以字符0作填充字符**
  - **域宽说明也可以是字符*，这时域宽大小由下一个输出项的整数值给出，又若该值为负值，相当于有左对齐标志-。例如printf("%*d",iw,123)显示123占用的域宽由整型变量iw给出，并当iw为负值时，表示以左对齐方式编排输出的格式**

printf()（续）

- - 左对齐标志，当转换后字符个数少于w时，在w所限定的字段宽度内，转换所得字符列左对齐，右边补填充符。缺省时，右对齐，左边补填充符
- + 正数也带符号输出，适用于带符号的数值数据输出。根据数值的正、负，在数值字符列之前加上符号‘+’或‘-’。缺省时，只对负数输出有负号‘-’
- 空格 若对应输出的数值数据是一个正数，符号用空格代替。若‘ +’和空格同时出现，空格格式修饰说明被略去

printf()（续）

- #用于八进制数，十六进制数和浮点数格式输出
  - 当八进制或十六进制数输出时，前面加字符0(八进制数)，或加双字符0x(十六进制数)。缺省时，不加字符0，或双字符0x
  - 对浮点数输出(e、E、f、g、G)总显示小数点，不管小数点后是否没有数字
  - 当用g或G格式输出时，无意义的小数点和小数部分的无意义的0根据精度的要求输出；缺省时，无意义的小数点和0不输出

printf()（续）

- .p 其中p也是十进制整数
  - 对于g(或G)或e(或E)格式输出，p指明输出精度(有效数字位数)，缺省时，p=6
  - 对于f格式输出，p指明输出字符列的小数点之后的数字个数，缺省时，p=6
  - 对于s格式输出，p指明最多输出字符串的前p个字符，多余截断。缺省时，字符串的内容全部输出
  - 对于d、i、o、u、x和X，表示至少出现的数字个数
- 同域宽说明一样，p也可以是字符*，而实际值由后面一个输出项的整数值给出，若该值为负值，相当于没有给出 p

printf()（续）

- h/l/L 指明输出项的类型
  - 长度修饰符h用于格式符d、i、o、u、x和X，表示对应的输出项是短整型(short int)或无符号短整型(unsigned short int)
  - 长度修饰符l用于格式符d、i、o、u、x和X，表示对应的输出项是长整型或无符号长整型
  - 长度修饰符L用于格式符e、E、f、g、G，表示对应的输出项是long double型

```
int i = 1234; long j = 1234567L;
printf("%d,%+6d,%06d,%-6d,%5ld", i, i, i, i, j)
```

将输出:1234, +1234,001234,1234 ,1234567

注意，对于long型数据输出，必须在格式符之前有长度修饰符l，表明输出long型数据。

若 `int k = 045; long p = -1L;`

`printf("%#o,%4o,%6lo", k, k, p)`

将输出：045, 45,37777777777

而printf("%#x,%4x,%6lX", k, k, p)将输出

0x25, 25,FFFFFFFF

若 unsigned int u = 65535u; long p=-1L
printf("%d,%4u,%lu",u,u,p)
将输出:-1,65535,4294967295

若 char ch1 = 045, ch2 = 'a';
printf("%c,%-3c,%2c", ch1, ch2, ch2)
将输出:%,a , a

若 char s[] = "ABCDEF";
printf("%3s,%4.2s,%-7.4s,%.5s", s, s, s, s)
将输出:ABCDEF, AB,ABCD ,ABCDE

若float f = 123.4567f; double d = 123.456789;
printf("%f,%8.3f,%-7.2f,%.7f", f, f, f, d)
将输出:123.456703, 123.457,123.46 ,123.4567890

而printf("%e,%10.2e,%-10.2e,%.2e",f,f,f,f,d)
将输出:1.23457e+02, 1.2e+02,1.2e+02 ,1.2e+02

printf()（续）

- 实型数据的有效位数
  - 不要以为凡是打印(显示)的数字都是准确的
  - 一般地，float型只有7位有效数字，double型有15位有效数字。实际上，因计算过程中的误差积累，通常不能达到所说的有效位数
- %g格式的特殊性
  - 当它选择“整数部分.小数部分”形式时，因格式修饰说明.p在e格式中的意义是指精度，所以p的值是整数部分位数与小数部分位数之和(不是f格式中的小数位数)
  - 格式修饰说明#有无也对输出形式有影响
    - 如#缺省，输出时，小数部分无意义的0及小数点不输出；如有#，则无意义的0及小数点照常输出

如有 float g1 = 12.34f, g2 = 0.0f;

double d = 123.456789, g = 123456.789;

int count;

```
printf("%g,%g,%#g,%#g%n",g1,g2,g1,g2,&count);
printf("COUNT = %d\n", count);
printf("%f,%g,%g,%g,%.8g", g1, g1, d, g, g)
```

将输出：

12.34,0,12.3400,0.000000

COUNT = 24

12.340000,12.34,123.457,123457,123456.79

scanf()

- scanf()函数的功能是从标准设备读入字符序列，按格式控制字符串所包含的格式解释输入字符序列，并将解释结果存贮到对应的数据存贮地址中
- 调用格式输入函数scanf()的一般形式为：

scanf(格式控制字符串，输入数据存贮地址，输入数据存贮地址， …)

- 格式控制字符串的一般形式是字符串表达式，通常是由一对双引号括起来的字符串常量，直接用于解释输入字符序列

printf(格式控制字符串，输入项存储地址，输入项存储， …)

↓

“空白类字符/普通字符/格式转换说明”

↓

%[格式修饰说明] 输入格式符

↓

[*][w][h/l/L]

scanf()(续)

- 格式控制字符串可以包含
  - 空白类字符（空格符或制表符或换行符），使输入掠过空白类字符，直到遇到下一个非空白类字符
  - 普通字符(不包括%)，要求输入字符流下一个字符与它相同
  - 格式转换说明，以字符‘%’开头至输入格式符结束的字符序列组成。格式转换说明引导对下一输入字符段进行转换
- 例如，设变量 i，j，k 为整型变量，函数调用 scanf(“%d%d%d”, &i, &j, &k); 为变量 i，j，k 输入数据。其中&i，&j，&k分别表示变量i，j，k的存贮地址

scanf()(续)

- 格式转换说明的一般形式
  - %[*][w][h/l/L] 输入格式符
- 输入格式符共有14个，12种不同输入格式
- 格式修饰说明：方括号括住的内容，可以缺省，它们的意义是
  - * 赋值抑制符，对应的输入数据项按要求被输入，但结果不存贮。 带星号的格式转换说明不对应输入数据存贮地址，用它来跳过一个输入数据项
  - w 整型常数(域宽说明)，输入数据项的字符段宽度。若实际输入字符段宽度小于w，以实际宽度为准

scanf()(续)

- 除输入格式符c和[]外，输入域定义为从下一个非空白类字符起(因此可能跳过空格符，制表符，换行符)，到一个与所解释类型相矛盾的字符为止，或直到转换了域宽说明所指定的字符个数为止
- h/l/L 长度修饰符，指明输入数据项的存储类型
  - h修饰格式符d、i、o、u、x，输入整数以短整型存贮
  - l修饰格式符d、i、o、u、x，输入整数以长整型存贮；修饰格式符e，f，g，输入实数以double型存贮
  - L修饰格式符e，f，g，表示输入的实数是按long double 型存贮
  - 缺省时，对于格式符d、i、o、u、x，输入整数以int型存贮；对于格式符e，f，g，输入实数以float型存贮

输入格式符表

格式符	意　义
d	输入十进制整型数据
i	输入十进制整型数据。若输入数据以字符0开始，则以八进制形式输入；若以0x或0X开头，则以十六进制形式输入
o	以八进制形式输入整型数据
x	以十六进制形式输入整型数据
u	输入无符号整型数据
c	输入字符数据
s	输入字符串
e、f、g	输入实型数据
p	输入指针值
n	记录已输入的字符个数
[]	输入匹配字符串
%	匹配输入字符

scanf()(续)

- 格式控制字符串之后给出的是变量地址，而不是变量名(除非是指针)。如要为整型变量n输入数据，
  - 写成 scanf(“%d”, n) 是不正确的
  - 应写成 scanf("%d", &n)
- 如果在格式控制字符串中除格式转换说明和空白符之外，还有其他字符，则在输入数据时应输入与这些字符相同的字符。例如scanf(“%d,%d”, &i, &j), 则在为 i，j 输入数据时，紧接在第一个整型数据之后，必需需要有一个逗号字符
  - 输入 1，2是正确的
  - 输入 1 2等其他形式都是不正确的

scanf()(续)

- 输入数据时，将字符流转换成内部表示后，存贮到对应数据存贮地址中
- 例如: scanf(“%3d%*4d%d”, &i, &j);如输入字符行为:123456 78, 将使变量 i=123， j=78。其中数据 456 因赋值抑制符*的作用被跳过
- 一般从键盘读入数据，不指定输入数据项的字段宽度，数据项与数据项之间用空白符、或制表符、或回车符分隔
- 为整型变量输入整数时，若变量类型为短整型，可在格式符之前加长度修饰说明h；若变量类型为长整型，则必须在格式符之前加长度修饰说明l

scanf()(续)

- 输入数值数据时，输入字符流中的前导空白类字符会被自动掠过，从空白类字符后的字符开始输入。构成数值数据的字符被输入转换成计算机的内部表示，并存储结果。若第一个非空白类字符不能构成数值字符，则立即结束输入

- s 格式用来输入字符串，对应的数据存贮地址为字符列表(数组)的首地址，该数组必须大到足以容纳可能输入的最长字符串

- 在输入字符流中，以非空白类字符开始，以后随的第一个空白类字符结束的非空白类字符的字符序列作为一个字符串。scanf() 函数在输入的字符序列之后自动添加字符串结束标志符\0（因此，存储输入字符序列的字符数组的长度必须比实际最长字符串的字符数多1）

scanf()(续)

- s格式符
  - 数组变量定义：s[100];
  - 格式输入函数调用: scanf("%4s", s)
  - 格式“%4s”要掠过开头的空白类字符，最多输入4个非空白类字符，输入的非空白类字符存于数组s[]，并在输入字符序列之后还有字符\0

scanf()(续)

- e、f、g格式用来输入实数。如格式转换说明中含有长度修饰说明l，则为double型变量地址；含有长度修饰说明L，则为long double型变量地址；否则，为float型变量地址

- []格式由左方括号[后面跟随一系列字符及右方括号]组成。一对方括号之间的字符定义了一个匹配字符集。例如，
  - 格式“%[xy]”定义一个由字符x和字符y 两个字符构成的字符集。输入字符序列中，所有只由字符x和y组成的字符序列都能与之匹配
  - 若左方括号之后的第一个字符是字符^，表示一个由其后继字符定义的字符集的补集字符集

scanf()(续)

- 例如：“ %[^xy]” 定义了这样一个匹配字符集，所有除字符x和字符y之外的其它字符都能与之匹配
- 输入时，从当前输入字符开始到输入字符序列中第一个不能与之匹配的字符为止。注意，格式[]不跳过开头的空白类字符
- 该格式符对应的数据存贮地址为字符列表的首地址。所有匹配的字符都被输入存贮，并在最后添加字符串结束标志符‘\0’

输入输出流cin和cout

- 在c++程序中，使用cin和cout流对象进行基本的输入输出，它们在头文件iostream.h中定义，在程序中，应使用如下预处理命令包含该头文件
  - #include <iostream.h>

- 头文件iostream.h中含有cin、cout、cerr、clog 4个对象，对应于标准输入流、标准输出流、非缓冲和经缓冲的标准错误流。
  - cin对应于标准输入流设备（如键盘）
  - cout对应于标准输出流设备（如显示器）

输出流cout（1）

- 使用cout输出数据的一般格式为
  - cout << 表达式 1 << 表达式 2 … << 表达式 n;
  - 功能是：从屏幕当前光标位置开始，将各表达式的值依次输出
  - <<为流插入运算符，表示将表达式的数据依次插入到内存缓冲区中。当遇到缓冲区满、输出换行符endl（或者\n）、清除缓冲区的流格式控制符flush时，才将缓冲区中的数据输出到屏幕上，并清除缓冲区

输出流cout（2）

- 一个<<后只能有一个输出项；如有多个输出项，则需要多个流插入运算符
  - cout << x, y, z; /* 出错！只输出一项x */
  - cout << x << y << z;

```
#include <iostream.h>
int main(){
  cout << "Welcome to C++!\n";

  return 0;
}
```

输出结果：Welcome to C++!

输入流cin（1）

- 使用cin输出数据的一般格式为
  - cin >> 表达式 1 >> 表达式 2 … >> 表达式 n;
  - 功能是：程序暂停执行，等待用户从键盘上输入数据。用户输入了所有数据后在以回车键表示输入结束。程序将用户输入的数据依次传给各变量，并继续运行后续语句

输入流cin（2）

- >>称为流提取运算符，表示将内存缓冲区中的数据提取出来，并赋值给变量
- 输入数值数据时，前导的空白类字符被自动忽略，符合类型要求的数据被接受，遇到不符合类型要求的字符结束一个数值数据的输入
  - 通常输入多个数值数据时，数据之间以空白符或回车键作为数值数据的分隔符

输入流cin（3）

- 输入字符数据时，前导的空白类字符被自动忽略，以后输入的每个字符都被输入和存储
  - 字符变量只接受一个字符
  - 字符数组变量接受一个字符串

- 与scanf不同，变量名前没有取地址运算符&

例子（1）

```
#include <iostream.h>
int main(){
 int x; char c1, c2;
 cin >> c1 >> x >> c2;
 cout << "c1=" << c1 << ", c2=" << c2 << ", x="
   << x << endl;
 return 0;
}
```

如果输入：a123b

则输出为：c1=a, c2=b, x=123

例子（2）

```
#include <iostream.h>
int main(){
 int x, y=0;
 cin >> x >> y;
 cout << "x=" << x << ", y=" << y << endl;
 return 0;
}
```

如果输入：123 456　则输出为：x=123, y=456

如果输入：123, 456　则输出为：x=123, y=0

例子（3）

```
#include <iostream.h>
int main(){
 int x; float y; char z;
 cin >> x >> y >>z;
 cout << "x=" << x << ", y=" << y << ", z=" << z
   <<endl;
 return 0;
}
```

如果输入：123.4 567.89 a

则输出为：x=123, y=0.4, z=5

提要

- 数据输入输出基础
- **实例讲解**

实例1

- 输入三角形的三边长，求三角形面积
  - 设输入的三边长a、b、c能构成三角形。从数学知识已知求三角形面积的公式为

$$\text{area}=\sqrt{s(s-a)(s-b)(s-c)}$$

  - 其中s=(a+b+c)/2

```
#include <stdio.h>
#include <math.h>
int main() {
  float a, b, c, s, area;
  scanf("%f, %f, %f", &a, &b, &c);
  s= (a+b+c)/2.0;
  area= sqrt(s*(s-a)*(s-b)*(s-c));
  printf("a=%7.2f, b=%7.2f, c =%7.2f, s =%7.2f\n",
                a, b, c, s);
  printf("area=%7.2f\n", area);
  return 0;
}
```

实例2

- 从键盘输入一个大写字母，要求改用小写字母输出

```
#include<stdio.h>
int main(){
        char cl，c2；
        cl=getchar()；
        printf("%c，%d\n"，cl，cl)；
        c2=cl+32；
        printf("%c，%d\n"，c2，c2)；
}
```

实例3

- 求ax2+bx+c=0方程的根。a，b，c由键盘输入，设b2-4ac＞0，方程的根x_1, x_2分别是：

$$x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a} \qquad x_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}$$

```
#include<math.h>
#include<stdio.h>
int main(){
  float a，b，c，disc，x1，x2，p，q;
  scanf("a=%f，b=%f，c=%f"，&a，&b，&c);
  disc=b*b-4*a*c;
  p=-b/(2*a);
  q=sqrt (disc)/(2*a);
  x1=p+q;    x2=p-q;

  printf("\n\nx1=%5.2f\nx2=%5.2f\n"，x1，x2);
  return 0;
}
```

实例4

- 计算1-1/3+1/5-1/7+…直到最后一项的绝对值小于10^{-6}

```
#include<stdio.h>
int main(){
  int n=1;
  float x=1, t=1, s=0;
  while(t>=1e-6){
    t=1.0/(2*n-1)
    s=s+x*t;
    x=-1*x; n=n+1;
  }
  printf("1-1/3+1/5-1/7+…=%f\n", s);
}
```

实例5：求水仙花数

- 一个三位数的百位数、十位数和个位数的立方和等于这个数，则该数为水仙花数

```
#include<stdio.h>
int main(){
  int n=100, i, j, k;
  do {
    i=n/100;
    j=(n/10)%10;
    k=n%10;
    if (n==i*i*i+j*j*j+k*k*k)
       printf("%d= %d^3+ %d^3+ %d^3\n", n,i,j,k)
    n=n+1;
  }while(n<=999)
}
```

实例6：编写一个程序，示意前缀++和后缀++的区别，前缀--和后缀--的区别

```
#include <iostream.h>
int main(){
        int i, j;
        i=4;
        cout << "执行j=++i前" << "i = "<<i<<endl;
        j=++i;
        cout << "执行j=++i后" << "i = "<<i<<"\tj = "<<j<<endl;
        i=4;
        cout << "执行j=i++前" << "i = "<<i<<endl;
        j=i++;
        cout << "执行j=i++后" << "i = "<<i<<"\tj = "<<j<<endl;
        i=4;
        cout << "执行j=--i前" << "i = "<<i<<endl;
        j=--i;
        cout << "执行j=--i后" << "i = "<<i<<"\tj = "<<j<<endl;
        i=4;
        cout << "执行j=i--前" << "i = "<<i<<endl;
        j=i--;
        cout << "执行j=i--后" << "i = "<<i<<"\tj = "<<j<<endl;
    return 0;
}
```

实例7：编写输入两个整数，输出这两个整数的和、差、积、商和余数的程序

```
#include <iostream.h>
int main(){
        int i, j;
        cout <<"输入两个整数！ \n";
        cin >> i>>j;
        cout << i << "+"<<j<<"="<<i+j<<endl;
        cout << i << "-"<<j<<"="<<i-j<<endl;
        cout << i << "*"<<j<<"="<<i*j<<endl;
        cout << i << "/"<<j<<"="<<i/j<<endl;
        cout << i << "%"<<j<<"="<<i%j<<endl;
    return 0;
}
```

实例8：编写输入三个整数，输出这三个数的和、平均值、最小值和最大值的程序

```
#include <iostream.h>
int main(){
        int i, j, k, sum, max, min;
        double ave;
        cout <<"输入三个整数！ \n";
        cin >> i >> j >> k;
        sum = i + j + k;
        ave = sum/3.0;
        max = min = i;
        if(j > max)  max = j ;
        else if(j < min) min = j;
        if(k > max)  max = k ;
        else if(k < min) min = k;
        cout << i << "+"<<j<<"+"<<k<<"="<<sum<<endl;
        cout << "("<<i << "+"<<j<<"+"<<k<<")/3.0="<<ave<<endl;
        cout << "Min("<<i << ","<<j<<","<<k<<")="<<min<<endl;
        cout << "Max("<<i << ","<<j<<","<<k<<")="<<max<<endl;
    return 0;
}
```

实例9：编写输入两个整数，输出它们之间的以下关系：小于、大于、等于、不等于、整除等

```
#include <iostream.h>
int main(){
        int i, j;
        cout <<"输入两个整数！ \n";
        cin >> i >> j;
        if(i < j)
           cout << i << "小于"<<j<<endl;
        else
                cout << i << "不小于"<<j<<endl;
        if(i == j)
           cout << i << "等于"<<j<<endl;
        else
                cout << i << "不等于"<<j<<endl;
        if(i > j)
           cout << i << "大于"<<j<<endl;
        else
                cout << i << "不大于"<<j<<endl;
        if(j % i == 0)
           cout << i << "能整除"<<j<<endl;
        else
                cout << i << "不能整除"<<j<<endl;
    return 0;
}
```

第2章作业

- 第2章习题
  - 第1-12题