第3章 基本的标准输入与输出

华中科技大学计算机学院
甘早斌

本章内容

3.1 字符输入与输出

3.1.1 字符输出函数putchar

- 1. 函数原型
  - 函数原型是指对函数的名称、返回值类型、参数的数目及类型的说明。它规定了函数调用的语法格式，即函数调用形式。
  - putchar函数的原型：
    int putchar (int c);
  - 函数返回值：
    函数正确执行时返回该字符码，否则返回EOF。
- 2. 函数的调用形式
  - putchar函数的调用形式：
    putchar(c)
  - c为实际参数，它可为char、short与int类型的表达式，其值是要输出字符的字符码。

3.1.1 单字符输出函数putchar(续)

- 例3.1 使用putchar函数打印HUST。

```
1.  #include <stdio.h>
2.  int main(void)
3.  {
4.      char c1, c2, c3, c4;
5.      c1 = 'H';   c2 = 'U';
6.      c3 = 'S';   c4 = 'T';
7.      putchar(c1);  putchar(c2);
8.      putchar(c3);  putchar(c4);
9.      return 0;
10. }
```


3.1.1 单字符输出函数putchar(续)

- 例3.2 设变量说明为 char c = ' a'，则下列函数调用表达式都可输出字符' a'的图形符号。
  putchar(' a') putchar(c) putchar(97) putchar('\141')
- 例3.3 设变量说明为int i;下面的putchar函数调用以表达式的值为参数。
  putchar(i = ' '); 输出一个空格
  putchar(i = 32); 输出一个空格
  putchar(i ?' Y' :' N'); i≠0时输出字符Y，i＝0时输出字符N
  putchar('\a'); 系统扬声器响铃一次

3.1.2 字符输入函数getchar 1/3

- 1．函数原型
  - getchar函数的原型为：
    int getchar(void);
  - 返回值：函数返回值的类型为int，参数表中的void表示函数不需要参数。
- 2．函数调用形式
  - getchar函数的调用形式为：
    getchar()
  - 调用时，圆括号中不能带参数，但必须保留圆括号。函数执行时从输入流中读取一个字符，并将所读取的字符（类型为unsigned char）转换为int类型后作为函数的返回值。

3.1.2 字符输入函数getchar 2/3

- 3. getchar函数执行的执行流程
  - 首先检查输入流中是否有字符；若有则读取输入流中的第一个字符作为函数返回值，程序向后执行。
  - 若输入流中没有字符，getchar函数将进入等待输入状态。此时，可用键盘输入字符，按下回车键结束输入。
  - 比如，在键盘上按abc↙，所键入的字符’a’、’b’、’c’以及按下回车键产生的换行字符’\n’依次被送入输入流。
  - 按下回车键同时会激活处于等待输入状态的getchar函数，getchar函数从输入流读取第一个字符’a’，将其值转换为int类型后作为函数返回值返回，程序继续向后执行。剩余字符’b’、’c’和’\n’留存在输入流中。

3.1.2 字符输入函数getchar 3/3

- 例3.4 getchar函数执行流程分析

```
1.  #include<stdio.h>
2.  int main(void)
3.  {
4.    char  ch1, ch2, ch3;
5.    ch1=getchar();
6.    ch2=getchar();
7.    ch3=getchar();
8.    printf( “\n%c%c%c”, ch1, ch2, ch3);
9.    printf( “\n%d  %d  %d”, ch1, ch2, ch3);
10.   return 0;
11. }
```

程序执行时，如果输入：
a↙
b↙
那么将会输出：
a↙
b↙
97 10 98

3.2 字符串输入与输出

- 3.2.1 字符串输出函数puts
- 3.2.2 字符串输入函数gets

3.2.1 字符串输出函数puts 1/3

- 1. 函数原型
  int puts(const char *s);
- 2. 返回值
  - puts函数返回值类型为int，参数s是存放所要输出字符串的内存缓冲区的首地址，类型为字符指针。
  - puts函数从s所指定的地址读取字符串输出到标准输出设备，并在串尾输出一个换行符'\n'。字符串在内存缓冲区存储时串尾以空字符'\0'作为结束标志，puts取字符串时从s指定的内存区依次取字符直至取到空字符为止。
  - puts函数正确执行时返回一个非负整数值，如果出错，则返回EOF。参数表const char *s中的const表明字符指针s的值不会被该函数修改。

3.2.1 字符串输出函数puts 2/3

- 3. 函数的调用形式
  - puts函数的调用形式为： puts(s)
  - 其中，s为实际参数，可以是字符串常量、字符数组名、或指向某字符串的字符指针变量。
  - 在C语言中，字符串常量和字符数组名所表示的都是字符指针类型的常量值。
  - 字符串常量所表示的常量值是该字符串在内存缓冲区的首地址，字符数组名所表示的常量值是该字符数组在内存中的首地址（即数组的第一个字符元素的地址）。

3.2.1 字符串输出函数puts 3/3

- 4. 例3.6 使用puts函数在第一行输出“Hello,”，在第二行输出“World!”。

程序1：

```
#include <stdio.h>
int main(void)
{
    char s[20] = “Hello,”;
    char *pc = “World!”;
    puts(s);
    puts(pc);
    return 0;
}
```

程序2：

```
#include <stdio.h>
int main(void)
{
    puts(“Hello,\nWorld!”);
    return 0;
}
```


3.2.2 字符串输入函数gets 1/3

- 1．函数原型
  - gets函数的原型为：
    char *gets(char *s);
  - 函数的返回值的类型为char *，参数s是一个字符指针，指向存放输入字符串的内存缓冲区的首地址。
  - s所指向的内存缓冲区的长度应足够大，为满足极端情况下字符串输入的需要，缓冲区的长度应该不小于81个字节，以容纳输入的一行字符及一个空字符’\0’。

3.2.2 字符串输入函数gets 2/3

- 2、函数的调用形式
  - gets函数的调用形式为：
    gets(s);
  - gets从输入流中读取一行字符存放到s指定的内存缓冲区，结尾的换行字符’\n’被空字符’\0’所替换，以作为字符串的结束标志。
  - 输入流中的一行字符是从输入流的第一个字符开始直到其后的第一个换行字符’\n’为止的字符序列。
  - 函数正确执行时返回该内存缓冲区的首地址，即s的值；
  - 如果遇到文件尾或出错，则返回空指针NULL。

3.2.2 字符串输入函数gets 3/3

- 设有声明：char a[81]; 则gets(a);
  - 语句允许用户从键盘输入一个字符串，并将该输入串赋给字符数组a。
- 如果再执行：puts(a);
  - 则可以将用户输入的字符串在屏幕上显示出来。
- 注意：如果需要输入一个带空白字符的字符串，必须使用gets函数来实现。
- 需要说明的是：
  - 标准函数gets在C语言标准ISO-IEC 9899:2011（简称C11）之前的版本中是可以使用的。由于该函数隐含了编译器无法判断的安全隐患，因此在C11中删除了该函数。

3.3 格式化输入与输出

- 3.3.1 格式化输出函数printf
- 3.3.2 格式化输入函数scanf

3.3.1 格式化输出函数printf 1/18

- 1．函数原型
  int printf(const char *format, …);
  - 返回值：
    - printf函数的返回值类型为int。
    - 第一个形式参数format是一个字符串，称为格式字符串，用来指定输出数据的个数和输出格式；
    - 后面的“,…”表示该函数在调用时除前面的字符串参数必需之外，其余参数的数目可变，可以是0个到多个。
    - 其余参数是要被输出的数据，参数的个数和数据类型应与格式字符串中转换说明的个数和转换字符一致（参见表3.1）。
    - printf函数的返回值是函数调用时实际输出到标准输出设备的字符个数。

3.3.1 格式化输出函数printf 2/18

- 2．函数调用形式
  - printf函数的调用形式：
    printf(格式字符串, 数据项1, …, 数据项n);
  - 第一个参数（格式字符串）是必需的，其余参数（数据项1至数据项n）是要输出的数据，每个数据项是一个基本类型或指针类型的表达式；
  - 数据项的个数可以是0个到任意多个，但是在数目、数据类型和顺序上应与格式字符串中的格式转换说明一致；如果不一致，编译时不会报错，但是不能得到正确的输出结果。

3.3.1 格式化输出函数printf 3/18

- 3．printf函数的格式字符串
  - (1) 格式字符串的组成
    - 包含两类字符：普通字符和用于转换说明的格式字符。
    - 普通字符是作为提示信息来输出的文字，可以与转换说明交替出现，普通字符按原样输出。
    - 转换说明以%字符开始，以转换字符结尾，%字符和转换字符之间还可以带有域宽说明，用来指出数据输出时的对齐方向，输出数据域的宽度、小数部分的位数等格式要求。
    - 转换说明的语法形式为：
      %[域宽说明]转换字符

3.3.1 格式化输出函数printf 4/18

- 例子
  - 设变量说明为int i = 15; float x = -8.2;，则语句
    printf(“i=%d, x=%8.3f\n”, i, x);
  - 执行时输出
    i = 15, x =□□-8.200 (-8.200左边有两个空格)
  - 格式字符串“i=%d, x=%8.3f\n”中：
    - “i=”、“, ”、“x=”和“\n”是普通字符；
    - %d和%8.3f是两个转换说明，d和f是转换字符；
    - 8.3是域宽说明。转换说明%d对应于第二个参数i，%8.3f对应于第三个参数x；
    - 输出时格式字符串中的%d和%8.3f分别被i和x的值所代替。

3.3.1 格式化输出函数printf 5/18

- 4. printf的转换字符
  - 详见教材P59 表3.1
  - printf函数常用的转换字符：
    - d 输出十进制整数
    - u 输出十进制无符号整数
    - c 输出单个字符
    - s 输出字符串（必须以'\0'结束或在域宽说明中给出长度限制）
    - f 输出小数形式的浮点数，小数部分位数由精度确定，默认为6位

3.3.1 格式化输出函数printf 5/18

- 例子
  - 设变量说明为：
    char c=65; int x1=65;
    unsigned x2=65535; float y=65;
    char name[100]= “The C programming”;
  - 则
    printf(“%c,%d,%d,%c,%u,%f,%s”,
    c, c, x1, x1, x2, y, name);
  - 执行时输出为：
    A,65,65,A,65535,65.000000,The C programming

3.3.1 格式化输出函数printf 7/18

- printf函数常用格式字符用法的几点说明
  - ①转换说明%d和%u的区别是，对于二进制最高位为0的整数，两者输出相同的结果。例如：
    int i = 100; unsigned j = 100;
    printf(“%d,%u”, i, i); 输出100,100
    printf(“%d,%u”, j, j); 输出100,100
  - 对于二进制最高位为1的整数，%d输出相应负数值，%u输出相应正数值。例如：
    int i = ~0; unsigned j = ~0;
    printf(“%d,%u”, i, i); 输出-1,65535
    printf(“%d,%u”, j, j); 输出-1,65535

3.3.1 格式化输出函数printf 8/18

- ②%f，%e，%g三种转换说明在缺省域宽说明时的输出精度
  - %f输出小数形式的浮点数，小数部分为6位，多于6位采用四舍五入，少于6位则在末位补零，以保证6位小数；
  - %e输出标准指数形式的浮点数，尾数部分为6个有效数字，包括1位非零的整数部分和5位小数部分，采用四舍五入和末位补零的方法确保6个有效数字；
  - %g将按%f和%e两种格式输出的数据去掉无效零后进行比较，选取输出宽度较小的那种格式进行输出。

3.3.1 格式化输出函数printf 9/18

- 例子
  - 设变量说明为

    double x=12.00004951, y=12.4951, z=12000000.4951;
  - 则

    printf(“%f\t%e\t%g\n%f\t%e\t%g\n”, x, x, x, y, y, y);

    printf(“%f\t%e\t%g\n”, z, z, z);
  - 输出

    12.000050 1.200000e+001 12

    12.495100 1.249510e+001 12.4951

    12000000.495100 1.200000e+007 1.2e+007

3.3.1 格式化输出函数printf 10/18

- ③ 几个特殊转换字符的用法举例：
  - 例3.7 转换字符n的用法。设变量说明为int x = 1;，依次执行：

    printf(“abc%n”, &x);

    printf(“x=%d”, x);
    - 第一个语句输出abc，并将格式字符串“abc%n”中到%n为止已输出的字符数目3按对应参数所表示的内存地址写到内存中去，&x表示变量x的地址，那么x的值被改写为3；
    - 第二个语句输出x=3。

3.3.1 格式化输出函数printf 11/18

- 例3.8 转换字符%的用法。

  printf(“%%d”, 100);
  - 输出%d(相邻的两个%输出一个%，后面的d作为普通字符输出，由于格式字符串中没有转换说明，所以后面的数据项100不被输出)

    printf(“%%%d”, 100);
  - 输出%100(相邻的两个%输出一个%，%%后面的%d是输出十进制整数转换说明，所以输出后面的数据项100)

3.3.1 格式化输出函数printf 12/18

- 注意:
  - 如果%后面的字符是一个%，则输出一个%；
  - 如果%与其后的字符不能构成一个合法的转换说明，大多数编译系统将%与其后面的字符一起作为普通字符输出。例如:

    int i=-6; double x=5.7, y=123.4567;

    printf(“%d%%100”, -i/2);
  - 输出3%100（相邻的两个%输出一个%）

    printf(“%a”, i);
  - 输出%a（因为%后面的a不是转换字符，%a被当作普通字符输出，由于格式字符串中没有转换说明，所以后面的数据项i不被输出）。

3.3.1 格式化输出函数printf 13/18

- ④ 转换说明与输出参数不一致时可能导致的后果：
  - 因为printf函数根据格式字符串中的转换说明来决定输出数据项的数目和类型，如果转换字符使用不当，或转换说明的个数多于数据项个数，则会输出错误的数据；
  - 如果转换说明的个数少于数据项个数，则多出的数据项不被输出。

3.3.1 格式化输出函数printf 14/18

- 例如：

```
int i=-6; double x=5.7, y=123.4567;
printf(“%d,%g\n”, i, y, x);
```

- 输出 -6,123.457，数据项x因无对应的转换说明而未被输出。

```
printf(“%d\n”,x);
```

- 输出一个错误的值-13107，因为转换说明%d与输出数据项x类型不匹配所致。

```
printf(“i=%d\n”);
```

- 输出i=4109，4109是一个不确定的值，`。

3.3.1 格式化输出函数printf 15/18

- (2) 域宽说明
  - 在%和转换字符之间可以有域宽说明字符，用来指出输出数据的对齐方式、输出数据域的宽度、小数部分的位数等要求。可用作域宽说明的字符见表3.2，域宽说明可以是其中一个或多个字符的组合。
  - 最常用的printf函数域宽说明字符有以下几种形式：
    - %：表示格式说明的起始符号，不可缺少。
    - -：有-表示左对齐输出，如省略表示右对齐输出。
    - 0：有0表示指定空位填0,如省略表示指定空位不填。
    - m.n：m指输出数据的最小宽度。若数据实际宽度(位数)小于m，则左端补空格或零，若数据实际宽度大于m，则按实际位数输出。

      n指输出数据的精度。对于e/E/f格式的为小数部分的位数，对于g/G格式为有效数字的个数；对于整数位至少应输出的数字个数(用前导0补足)；对于字符串为至多输出的字符数目。对于浮点数未指定n时，隐含的精度为6位。
    - l或h：l对整型指long型，对实型指double型。h用于将整型的格式字符修正为short型。

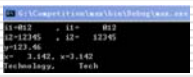
3.3.1 格式化输出函数printf 16/18

- 例3.9 -和m.n的用法

```
int i1=12, i2=12345;   float x=3.1416;
double y=123.4567;   char s1[20]= “Technology”;
printf(“i1=%-8.3d, i1=%8.3d\n”, i1, i1);
printf(“i2=%-8.3d, i2=%8.3d\n”, i2, i2);
printf(“y=%4.2f\n”, y);
printf(“x=%8.3f, x=%.3f\n”, x, x);
printf(“%8s, %8.4s\n”, s1, s1);
```

- 输出：

i1=012□□□□□, i1=□□□□□ 012
i2=12345□□□, i2=□□□ 12345
y=123.46
x=□□□3.142, x=3.142
Technology, □□□□Tech



3.3.1 格式化输出函数printf 17/18

- 例3.10 h，l，L的用法

```
short a;
long b;
double x;
long double y;
printf(“a=%hd,b=%ld,x=%lf,y=%Lf”, a, b, x, y);
```

- 输出短整数用%hd，输出长整数用%ld，
- 输出double类型的浮点数用%f和%lf都可以，
- 输出long double类型的浮点数用%Lf。

3.3.1 格式化输出函数printf 18/18

- 例3.11 *作为可变域宽的用法

```
int max=6;
char s[10]= “123456789”;
printf(“B%*cE\n”, max, ’ ’);
```

- 输出6个空格（*代表的域宽由对应参数max决定）

```
printf(“B%*sE\n”, 2*max, s);
```

- 输出123456789（左边有3个空格，*代表的域宽由对应参数2*max 决定）

```
printf(“B%.*s E\n”, max, s);
```

- 输出123456（*代表的输出精度由对应参数max决定）

3.3.2 格式化输入函数scanf 1/12

- 1．函数原型
  - scanf函数的原型为：

```
int scanf(const char *format, …);
```

  - 返回值
    - scanf函数的返回值类型为int。
    - 参数表中，format是格式字符串，用来指定输入数据的数目、类型和格式。
    - “, …”表示其他参数数目是可变的，其他参数必须是用来存放输入数据的内存地址。

3.3.2 格式化输入函数scanf 2/12

- 2、函数的调用形式

  scanf(格式字符串, 输入参数1, 输入参数2, ..., 输入参数n);

  - scanf从标准输入设备读取字符流，并按照格式字符串中由转换字符规定的格式转换成相应类型的值后，存放到输入参数指定的内存单元。
  - scanf函数正确执行时，返回值为被转换并赋值的数据的个数；遇到文件尾或出错时返回EOF。
  - 调用scanf函数一般至少需要读入一个数据，因此实际参数除格式字符串外至少应有一个输入参数。输入参数1至输入参数n可为基本类型或指针类型变量的地址（即指针）。
  - 用于输入字符串数据的参数应该为字符类型的指针，可以是字符数组名或指向字符数组首元素的指针变量。
  - 此外，输入参数在类型、数目和次序上应与格式字符串中的转换说明一致。

3.3.2 格式化输入函数scanf 3/12

- ① scanf函数的格式字符串
  - scanf函数的格式字符串与printf函数相似，在组成上可以包含普通字符和转换说明，具体包含以下三类字符：
    - 空白字符。包括如空格符、换行符、制表符（对数据的输入来说没有影响）；
    - 非%且非空白字符的普通字符（在输入流中相应位置必须有相同的字符与之匹配）；
    - 以%开头，以转换字符为结尾的转换说明，形式为：%[选项]转换字符。
  - 常用的scanf函数转换字符如表3.3所示。P63

3.3.2 格式化输入函数scanf 4/12

- 转换字符A,E,F,G,X与对应小写的转换字符在用法上相同。转换字符a、A在C99标准之前版本编译器中不能使用。
- 在实际使用中，scanf函数的格式字符串一般只需包含转换说明。**因为对于scanf函数格式字符串中除空格和制表符外的其他普通字符，在输入流中相应位置必须输入相同的字符，否则scanf函数读不到正确的数据**（系统不作检查）。
- 如果在scanf函数的格式字符串中加入了除空格和制表符以外的普通字符，不仅给数据输入带来麻烦，而且容易出错。可是，如果在格式字符串中每个转换说明之间适当添加空格或制表符（按Tab键输入），可以使转换说明看上去清晰明了，同时对数据的输入没有影响。

3.3.2 格式化输入函数scanf 5/12

- ② scanf函数常用的转换字符的用途
  - 例3.12 设变量说明为

    ```
    char c;
    int x1;
    unsigned x2;
    float y;
    char name[10];
    ```

  - 则输入以上类型数据的语句为：

    ```
    scanf(“%c%d%u%f%s”, &c, &x1, &x2, &y, name);
    ```


3.3.2 格式化输入函数scanf 6/12

- 例3.13 转换字符n的用法。
  - 设变量说明为int i, j, k;首先执行下面的输入语句：

    scanf(“%d %d %n”, &i, &j, &k);
  - 假定输入为：

    10 20 30↙
  - scanf函数在执行时，对应格式字符串中的第一个转换说明%d，10被赋予i；对应第二个转换说明%d，20被赋予j；
  - 对应第三个转换说明%n，将不从输入流中读取数据，而是把到此为止已从输入流中读过的字符数目5（包括10和20之间的一个空格字符）赋予k。输入流中还剩余字符’ ’，’3’，’0’和’\n’四个字符。

3.3.2 格式化输入函数scanf 7/12

- ③ 注意：
  - (i)如果转换说明为%%，则不从输入流读取数据，同时，对应参数指出的变量也将不被赋值；如果在其他完整的转换说明前面有两个相邻的%，则输入时需要一个%与之匹配。
  - 例如：设变量说明为int i=1;，依次执行下面两个语句：

    scanf(“%%d”,&i);

    printf(“i=%d”,i);
  - 若输入

    100↙（或%100↙）
  - 则输出

    i=1

3.3.2 格式化输入函数scanf 8/12

- (ii) 转换说明与输入参数不一致时可能导致的后果：如果转换说明与输入参数的类型不匹配，则导致读入的数据值不正确或程序非正常终止；如果转换说明的个数比输入数据的个数少，则无对应转换说明的变量不被赋值；如果转换说明的个数比输入参数的个数多，则可能死机。例如：

  int i, j;　double x;

  scanf("%d%d", &i, &j);

- 执行时若输入

  12a↙

- 则整数12被赋予i，由于第二个输入域是字符a，与第二个转换说明%d不匹配，因而不能被转换，j未被赋值；scanf函数执行时返回值为1，表明从输入流中读取并转换的数据个数是1。

3.3.2 格式化输入函数scanf 9/12

- 3．输入形式
  - 输入域
    - 构成一个被转换数据的字符序列称为一个输入域。即从输入流中当前位置开始，直到其后的第一个空白字符为止，或直至根据转换说明不能被转换的字符之前，或直至指定域宽范围内的所有字符，都是一个输入域。
  - 输入域的格式：
    - (1) 输入域之间一般可用空白字符（空格符、换行符、制表符）隔开；在整型、浮点型或字符型后面的字符型数据不用分隔符；在整型、浮点型或字符型数据后面的字符串数据可以有或无空白字符；一个字符串内部不能有空白字符，因为空白字符是输入域的分隔符。
    - 例3.14~例3.18

3.3.2 格式化输入函数scanf 10/12

- 输入域的格式：
  - (2)当输入完转换说明规定数目的数据并键入回车键时scanf函数开始执行；当用完格式字符串中的转换说明，或当某个输入域与转换说明不能匹配时，scanf停止执行；下一次调用scanf函数时从上一次调用已经匹配转换的最后一个字符的下一个字符（也就是当前输入流中的首字符）开始读取。

    例3.19

  - (3)当scanf函数的格式字符串中包含有非空白字符的普通字符时，在输入流中相应位置必须有相同的字符与之一一对应。

    例3.20~例3.21

3.3.2 格式化输入函数scanf 11/12

- 4．转换说明中的可选项
  - scanf函数转换说明中的可选项包括：用于指定输入域宽度的整数(m)，用于和整数或浮点数转换字符一起输出各种类型的整型数或浮点数的转换修饰字符（h，l，L），用于跳过某个输入域的赋值抑制符（*）。
  - (1)如果指定了域宽，则从自然输入域首字符开始取指定宽度的字符作为实际输入域。自然输入域的宽度若小于或等于指定的域宽时，将整个自然输入域作为实际输入域；自然输入域的宽度若大于指定域宽时，从自然输入域取完指定宽度的字符后，剩下的字符作为下一个自然输入域。

    例3.22~例3.23

3.3.2 格式化输入函数scanf 12/12

- (2)如果输入短整数、长整数、双精度浮点数、长双精度浮点数，则相应的转换字符前须分别加h, l和L（l是字母L的小写）。

  例3.24

- 注意：输入double类型数据时转换说明是%lf，不能用%f或%Lf。
- (3)如果在某个转换字符前面使用了“*”，则与该转换说明对应的输入域被跳过。这种情况称为“虚读”。虚读用于从输入流中有选择地读取部分内容。

  例3.25、例3.26

本章小结

- 本章介绍了几个基本的标准输入与输出函数，包括:
  - 单个字符输入输出函数getchar与putchar
  - 字符串输入输出函数gets和puts
  - 格式化输入输出函数scanf和printf的用法
  - 在编程时，这些函数的使用频度非常高，需要理解和掌握这些函数的参数、调用形式及函数返回值的用法，在编程时做到灵活应用。
  - 格式化输入输出中常用转换说明%c、%d、%s、%f等的用法需要记住。
  - printf转换说明的域宽说明字符和scanf转换说明可选项的用法，不必死记硬背，在使用时这些细节可以查阅书籍和手册。

Assignments:

- 必做题：
  - 3.2，3.3，3.4，3.5，3.6，3.7，3.8，3.9，3.10，3.11
- 我建议：
  - 后面习题，每题都做，并且搞懂！