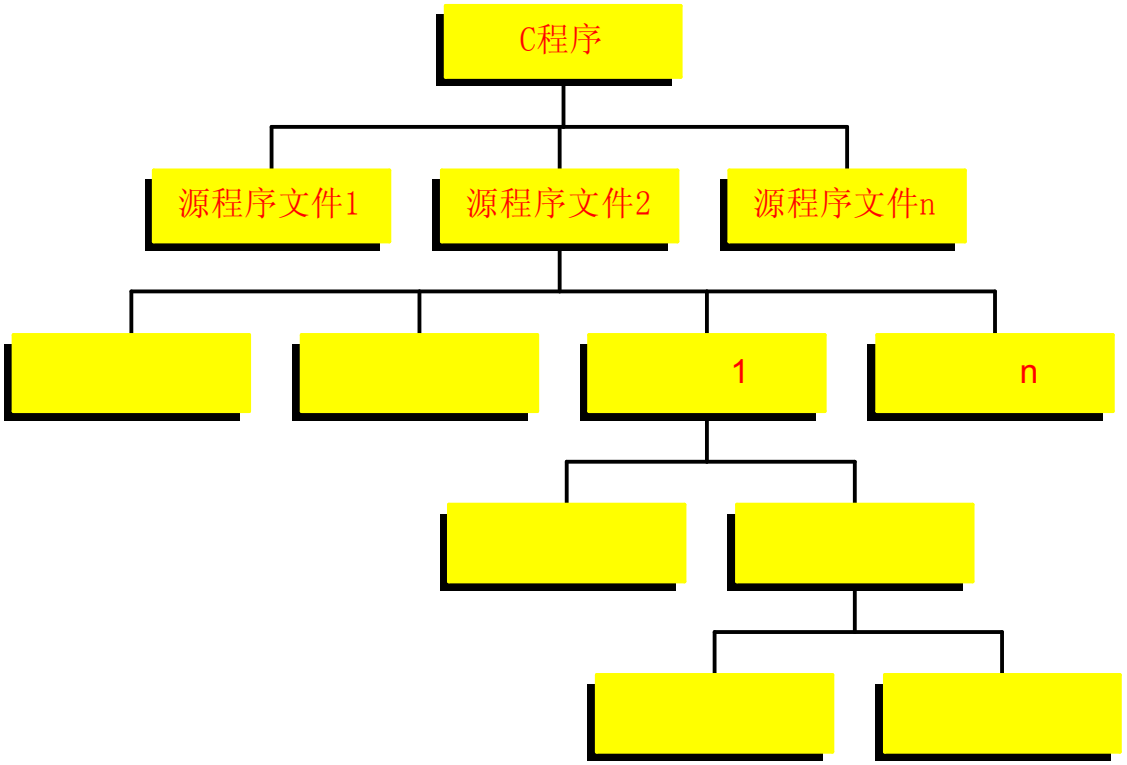
4 最简单的 C 程序设计—顺序程序设计

从程序流程的角度来看，程序可以分为三种基本结构， 即顺序结构、分支结构、循环结构。 这三种基本结构可以组成所有的各种复杂程序。C 语言提供了多种语句来实现这些程序结构。 本章介绍这些基本语句及其在顺序结构中的应用，使读者对 C 程序有一个初步的认识， 为后面各章的学习打下基础。

1.1 C 语句概述

C 程序的结构：



C 程序的执行部分是由语句组成的。 程序的功能也是由执行语句实现的。

C 语句可分为以下五类：

1) 表达式语句
2) 函数调用语句
3) 控制语句
4) 复合语句
5) 空语句

1. 表达式语句：表达式语句由表达式加上分号“;”组成。

   其一般形式为：

   表达式;

   执行表达式语句就是计算表达式的值。

   例如：

   x=y+z; 赋值语句；
   y+z; 加法运算语句，但计算结果不能保留，无实际意义；
   i++; 自增 1 语句，i 值增 1。

2. 函数调用语句：由函数名、实际参数加上分号“;”组成。

   其一般形式为：

   函数名(实际参数表);

   执行函数语句就是调用函数体并把实际参数赋予函数定义中的形式参数，然后执行被调函数体中的语句，求取函数值（在后面函数中再详细介绍）。

   例如：

   printf("C Program");调用库函数，输出字符串。

3. 控制语句：控制语句用于控制程序的流程， 以实现程序的各种结构方式。它们由特定的语句定义符组成。C 语言有九种控制语句。 可分成以下三类：

   1) 条件判断语句：if 语句、switch 语句；
   2) 循环执行语句：do while 语句、while 语句、for 语句；
   3) 转向语句：break 语句、goto 语句、continue 语句、return 语句。

4. 复合语句：把多个语句用括号{}括起来组成的一个语句称复合语句。

   在程序中应把复合语句看成是单条语句，而不是多条语句。

   例如：

   ```
   { x=y+z;
     a=b+c;
     printf("%d%d", x, a);
    }
   ```

   是一条复合语句。

   复合语句内的各条语句都必须以分号“;"结尾，在括号"}"外不能加分号。

5. 空语句：只有分号“;”组成的语句称为空语句。空语句是什么也不执行的语句。在程序中空语句可用来作空循环体。

   例如

   ```
   while(getchar()!='\n')
   ;
   ```

   本语句的功能是，只要从键盘输入的字符不是回车则重新输入。

   这里的循环体为空语句。

1.1 赋值语句

赋值语句是由赋值表达式再加上分号构成的表达式语句。

其一般形式为：

变量=表达式;

赋值语句的功能和特点都与赋值表达式相同。 它是程序中使用最多的语句之一。

在赋值语句的使用中需要注意以下几点：

1. 由于在赋值符“=”右边的表达式也可以又是一个赋值表达式，

   因此，下述形式

   变量=(变量=表达式);

   是成立的，从而形成嵌套的情形。

   其展开之后的一般形式为：

   变量=变量=…=表达式;

   例如：

   ```
   a=b=c=d=e=5;
   ```

   按照赋值运算符的右接合性，因此实际上等效于：

   ```
   e=5;
   d=e;
   c=d;
   b=c;
   a=b;
   ```

2. 注意在变量说明中给变量赋初值和赋值语句的区别。

   给变量赋初值是变量说明的一部分，赋初值后的变量与其后的其它同类变量之间仍必须用逗号间隔，而赋值语句则必须用分号结尾。

   例如：

   ```
   int a=5,b,c;
   ```

3. 在变量说明中，不允许连续给多个变量赋初值。

   如下述说明是错误的：

   ```
   int a=b=c=5
   ```

   必须写为

   ```
   int a=5,b=5,c=5;
   ```

   而赋值语句允许连续赋值。

4. 注意赋值表达式和赋值语句的区别。

   赋值表达式是一种表达式，它可以出现在任何允许表达式出现的地方，而赋值语句则不能。

   下述语句是合法的：

   ```
   if((x=y+5)>0) z=x;
   ```

   语句的功能是，若表达式 x=y+5 大于 0 则 z=x。

   下述语句是非法的：

   ```
   if((x=y+5;)>0) z=x;
   ```

   因为 x=y+5;是语句，不能出现在表达式中。

1.1 数据输入输出的概念及在 C 语言中的实现

1) 所谓输入输出是以计算机为主体而言的。
2) 本章介绍的是向标准输出设备显示器输出数据的语句。
3) 在C语言中，所有的数据输入 / 输出都是由库函数完成的。 因此都是函数语句。
4) 在使用C语言库函数时，要用预编译命令

```
#include
```

将有关“头文件”包括到源文件中。

使用标准输入输出库函数时要用到 “stdio.h”文件，因此源文件开头应有以下预编译命令：

```
#include< stdio.h >
```

或

```
#include "stdio.h"
```

stdio 是 standard input &outupt 的意思。

5) 考虑到 printf 和 scanf 函数使用频繁，系统允许在使用这两个函数时可不加

```
#include< stdio.h >
```

或

```
#include "stdio.h"
```

1.1 字符数据的输入输出

1.1.1 putchar 函数（字符输出函数）

putchar 函数是字符输出函数， 其功能是在显示器上输出单个字符。

其一般形式为：

putchar(字符变量)

例如：

```
putchar('A');      (输出大写字母 A)
putchar(x);        (输出字符变量 x 的值)
putchar('\101');   (也是输出字符 A)
putchar('\n');     (换行)
```

对控制字符则执行控制功能，不在屏幕上显示。

使用本函数前必须要用文件包含命令：

```
#include<stdio.h>
```

或

```
#include "stdio.h"
```

【例 4.1】输出单个字符。

```
#include<stdio.h>
main(){
  char a='B',b='o',c='k';
  putchar(a);putchar(b);putchar(b);putchar(c);putchar('\t');
  putchar(a);putchar(b);
```

```
  putchar('\n');
  putchar(b);putchar(c);
}
```

1.1.1 getchar 函数（键盘输入函数）

getchar 函数的功能是从键盘上输入一个字符。

其一般形式为：

```
getchar();
```

通常把输入的字符赋予一个字符变量，构成赋值语句，如：

```
char c;
c=getchar();
```

【例 4.2】输入单个字符。

```
#include<stdio.h>
void main(){
  char c;
  printf("input a character\n");
  c=getchar();
  putchar(c);
}
```

使用 getchar 函数还应注意几个问题：

1) getchar 函数只能接受单个字符，输入数字也按字符处理。输入多于一个字符时，只接收第一个字符。
2) 使用本函数前必须包含文件“stdio.h”。
3) 在 TC 屏幕下运行含本函数程序时，将退出 TC 屏幕进入用户屏幕等待用户输入。输入完毕再返回 TC 屏幕。
4) 程序最后两行可用下面两行的任意一行代替：

```
putchar(getchar());
printf("%c",getchar());
```

1.1 格式输入与输出

1.1.1 printf 函数（格式输出函数）

printf 函数称为格式输出函数，其关键字最末一个字母 f 即为“格式”(format)之意。其功能是按用户指定的格式，把指定的数据显示到显示器屏幕上。在前面的例题中我们已多次使用过这个函数。

1. printf 函数调用的一般形式

printf 函数是一个标准库函数，它的函数原型在头文件"stdio.h"中。但作为一个特例，不要求在使用 printf 函数之前必须包含 stdio.h 文件。

printf 函数调用的一般形式为：

printf("格式控制字符串", 输出表列)

其中格式控制字符串用于指定输出格式。格式控制串可由格式字符串和非格式字符串两种组成。格式字符串是以%开头的字符串，在%后面跟有各种格式字符，以说明输出数据的类型、形式、长度、小数位数等。如：

"%d"表示按十进制整型输出；

"%ld"表示按十进制长整型输出；

"%c"表示按字符型输出等。

非格式字符串在输出时原样照印，在显示中起提示作用。

输出表列中给出了各个输出项，要求格式字符串和各输出项在数量和类型上应该一一对应。

【例 4.3】

```
main()
{
  int a=88,b=89;
  printf("%d %d\n",a,b);
  printf("%d,%d\n",a,b);
  printf("%c,%c\n",a,b);
  printf("a=%d,b=%d",a,b);
}
```

本例中四次输出了 a,b 的值，但由于格式控制串不同，输出的结果也不相同。第四行的输出语句格式控制串中，两格式串%d 之间加了一个空格(非格式字符)，所以输出的 a,b 值之间有一个空格。第五行的 printf 语句格式控制串中加入的是非格式字符逗号， 因此输出的 a,b 值之间加了一个逗号。第六行的格式串要求按字符型输出 a,b 值。第七行中为了提示输出结果又增加了非格式字符串。

2. 格式字符串

在 Turbo C 中格式字符串的一般形式为：

[标志][输出最小宽度][.精度][长度]类型

其中方括号[]中的项为可选项。

各项的意义介绍如下：

1) 类型：类型字符用以表示输出数据的类型，其格式符和意义如下表所示：

格式字符	意　　义
d	以十进制形式输出带符号整数(正数不输出符号)
o	以八进制形式输出无符号整数(不输出前缀 0)
x,X	以十六进制形式输出无符号整数(不输出前缀 0x)
u	以十进制形式输出无符号整数
f	以小数形式输出单、双精度实数
e,E	以指数形式输出单、双精度实数
g,G	以%f 或%e 中较短的输出宽度输出单、双精度实数
c	输出单个字符

s	输出字符串

2) 标志：标志字符为-、+、#、空格四种，其意义下表所示：

标 志	意　　义
-	结果左对齐，右边填空格
+	输出符号(正号或负号)
空格	输出值为正时冠以空格，为负时冠以负号
#	对 c,s,d,u 类无影响；对 o 类,在输出时加前缀 o；对 x 类,在输出时加前缀 0x；对 e,g,f 类当结果有小数时才给出小数点

3) 输出最小宽度:用十进制整数来表示输出的最少位数。若实际位数多于定义的宽度,则按实际位数输出，若实际位数少于定义的宽度则补以空格或 0。

4) 精度:精度格式符以"."开头，后跟十进制整数。本项的意义是：如果输出数字，则表示小数的位数；如果输出的是字符，则表示输出字符的个数；若实际位数大于所定义的精度数，则截去超过的部分。

5.长度:长度格式符为 h,l 两种，h 表示按短整型量输出，l 表示按长整型量输出。

【例 4.4】

```
main()
{
  int a=15;
  float b=123.1234567;
  double c=12345678.1234567;
  char d='p';
  printf("a=%d,%5d,%o,%x\n",a,a,a,a);
  printf("b=%f,%lf,%5.4lf,%e\n",b,b,b,b);
  printf("c=%lf,%f,%8.4lf\n",c,c,c);
  printf("d=%c,%8c\n",d,d);
 }
```

本例第七行中以四种格式输出整型变量 a 的值，其中"%5d "要求输出宽度为 5，而 a 值为 15 只有两位故补三个空格。 第八行中以四种格式输出实型量 b 的值。其中"%f"和"%lf "格式的输出相同，说明"l"符对"f"类型无影响。"%5.4lf"指定输出宽度为 5，精度为 4,由于实际长度超过 5 故应该按实际位数输出，小数位数超过 4 位部分被截去。第九行输出双精度实数，"%8.4lf "由于指定精度为 4 位故截去了超过 4 位的部分。第十行输出字符量 d,其中"%8c "指定输出宽度为 8 故在输出字符 p 之前补加 7 个空格。

使用 printf 函数时还要注意一个问题，那就是输出表列中的求值顺序。不同的编译系统不一定相同，可以从左到右，也可从右到左。Turbo C 是按从右到左进行的。请看下面两个例子：

【例 4.5】

```
  main(){
  int i=8;
  printf("%d\n%d\n%d\n%d\n%d\n%d\n",++i,--i,i++,i--,-i++,-i--);
}
```

【例 4.6】

```
main(){
int i=8;
printf("%d\n",++i);
printf("%d\n",--i);
printf("%d\n",i++);
printf("%d\n",i--);
printf("%d\n",-i++);
printf("%d\n",-i--);
}
```

这两个程序的区别是用一个 printf 语句和多个 printf 语句输出。但从结果可以看出是不同的。为什么结果会不同呢？就是因为 printf 函数对输出表中各量求值的顺序是自右至左进行的。在第一例中，先对最后一项"-i--"求值，结果为-8,然后 i 自减 1 后为 7。 再对"-i++"项求值得-7，然后 i 自增 1 后为 8。再对"i--"项求值得 8，然后 i 再自减 1 后为7。再求"i++"项得 7，然后 i 再自增 1 后为 8。 再求"--i"项，i 先自减 1 后输出，输出值为 7。最后才求输出表列中的第一项"++i"，此时 i 自增 1 后输出 8。

但是必须注意，求值顺序虽是自右至左，但是输出顺序还是从左至右，因此得到的结果是上述输出结果。

1.1.1 scanf 函数(格式输入函数)

scanf 函数称为格式输入函数，即按用户指定的格式从键盘上把数据输入到指定的变量之中。

1. scanf 函数的一般形式

scanf 函数是一个标准库函数，它的函数原型在头文件"stdio.h"中，与 printf 函数相同，C 语言也允许在使用 scanf 函数之前不必包含 stdio.h 文件。

scanf 函数的一般形式为：

scanf("格式控制字符串"，地址表列)；

其中，格式控制字符串的作用与 printf 函数相同，但不能显示非格式字符串，也就是不能显示提示字符串。地址表列中给出各变量的地址。地址是由地址运算符"&"后跟变量名组成的。

例如：

&a, &b

分别表示变量 a 和变量 b 的地址。

这个地址就是编译系统在内存中给 a,b 变量分配的地址。在 C 语言中，使用了地址这个概念，这是与其它语言不同的。 应该把变量的值和变量的地址这两个不同的概念区别开来。变量的地址是 C 编译系统分配的，用户不必关心具体的地址是多少。

变量的地址和变量值的关系如下：

在赋值表达式中给变量赋值，如：

a=567

则，a 为变量名，567 是变量的值，&a 是变量 a 的地址。

但在赋值号左边是变量名，不能写地址，而 scanf 函数在本质上也是给变量赋值，但要

求写变量的地址，如&a。 这两者在形式上是不同的。&是一个取地址运算符，&a 是一个表达式，其功能是求变量的地址。

【例 4.7】

```
main(){
  int a,b,c;
  printf("input a,b,c\n");
  scanf("%d%d%d",&a,&b,&c);
  printf("a=%d,b=%d,c=%d",a,b,c);
}
```

在本例中，由于 scanf 函数本身不能显示提示串，故先用 printf 语句在屏幕上输出提示，请用户输入 a、b、c 的值。执行 scanf 语句，则退出 TC 屏幕进入用户屏幕等待用户输入。用户输入 7　8　9 后按下回车键，此时，系统又将返回 TC 屏幕。在 scanf 语句的格式串中由于没有非格式字符在"%d%d%d"之间作输入时的间隔，因此在输入时要用一个以上的空格或回车键作为每两个输入数之间的间隔。如：

7 8 9

或

7
8
9

2. 格式字符串

格式字符串的一般形式为：

%[*][输入数据宽度][长度]类型

其中有方括号[]的项为任选项。各项的意义如下：

1) 类型：表示输入数据的类型，其格式符和意义如下表所示。

格式	字符意义
d	输入十进制整数
o	输入八进制整数
x	输入十六进制整数
u	输入无符号十进制整数
f 或 e	输入实型数(用小数形式或指数形式)
c	输入单个字符
s	输入字符串

2) "*"符:用以表示该输入项,读入后不赋予相应的变量，即跳过该输入值。

如:

```
scanf("%d %*d %d",&a,&b);
```

当输入为：1　2　3 时，把 1 赋予 a，2 被跳过，3 赋予 b。

3) 宽度:用十进制整数指定输入的宽度(即字符数)。

例如：

```
scanf("%5d",&a);
```

输入：12345678

只把 12345 赋予变量 a，其余部分被截去。

又如：

scanf("%4d%4d",&a,&b);

输入：12345678

将把 1234 赋予 a，而把 5678 赋予 b。

4) 长度:长度格式符为 l 和 h，l 表示输入长整型数据(如%ld) 和双精度浮点数(如%lf)。h 表示输入短整型数据。

使用 scanf 函数还必须注意以下几点：

1) scanf 函数中没有精度控制，如：scanf("%5.2f",&a);是非法的。不能企图用此语句输入小数为 2 位的实数。

2) scanf 中要求给出变量地址，如给出变量名则会出错。如 scanf("%d",a);是非法的，应改为 scnaf("%d",&a);才是合法的。

3) 在输入多个数值数据时，若格式控制串中没有非格式字符作输入数据之间的间隔则可用空格，TAB 或回车作间隔。C 编译在碰到空格，TAB，回车或非法数据(如对"%d"输入"12A"时，A 即为非法数据)时即认为该数据结束。

4) 在输入字符数据时，若格式控制串中无非格式字符，则认为所有输入的字符均为有效字符。

例如：

scanf("%c%c%c",&a,&b,&c);

输入为：

d e f

则把'd'赋予 a, ' ' 赋予 b,'e'赋予 c。

只有当输入为：

def

时，才能把'd'赋于 a,'e'赋予 b,'f'赋予 c。

如果在格式控制中加入空格作为间隔，

如:

scanf ("%c %c %c",&a,&b,&c);

则输入时各数据之间可加空格。

【例 4.8】

```
main(){
  char a,b;
  printf("input character a,b\n");
  scanf("%c%c",&a,&b);
  printf("%c%c\n",a,b);
}
```

由于 scanf 函数"%c%c"中没有空格，输入 M　N，结果输出只有 M。而输入改为 MN 时则可输出 MN 两字符。

【例 4.9】

```
main(){
  char a,b;
  printf("input character a,b\n");
  scanf("%c %c",&a,&b);
  printf("\n%c%c\n",a,b);
```

```
}
```

本例表示 scanf 格式控制串"%c %c"之间有空格时，输入的数据之间可以有空格间隔。

5) 如果格式控制串中有非格式字符则输入时也要输入该非格式字符。

例如：

```
scanf("%d,%d,%d",&a,&b,&c);
```

其中用非格式符" , "作间隔符，故输入时应为：

```
5,6,7
```

又如：

```
scanf("a=%d,b=%d,c=%d",&a,&b,&c);
```

则输入应为：

```
a=5,b=6,c=7
```

6) 如输入的数据与输出的类型不一致时，虽然编译能够通过，但结果将不正确。

【例 4.10】

```
main(){
  int a;
  printf("input a number\n");
  scanf("%d",&a);
  printf("%ld",a);
}
```

由于输入数据类型为整型，而输出语句的格式串中说明为长整型，因此输出结果和输入数据不符。如改动程序如下：

【例 4.11】

```
main(){
    long a;
    printf("input a long integer\n");
    scanf("%ld",&a);
    printf("%ld",a);
}
```

运行结果为：

```
input a long integer
1234567890
1234567890
```

当输入数据改为长整型后，输入输出数据相等。

【例 4.12】

```
main(){
  char a,b,c;
  printf("input character a,b,c\n");
  scanf("%c %c %c",&a,&b,&c);
```

```
  printf("%d,%d,%d\n%c,%c,%c\n",a,b,c,a-32,b-32,c-32);
 }
```

输入三个小写字母，输出其 ASCII 码和对应的大写字母。

【例 4.13】

```
main(){
 int a;
 long b;
 float f;
 double d;
 char c;
  printf("\nint:%d\nlong:%d\nfloat:%d\ndouble:%d\nchar:%d\n",sizeof(a),sizeof(b)
  ,sizeof(f),sizeof(d),sizeof(c));
}
```

输出各种数据类型的字节长度。

1.1 顺序结构程序设计举例

【例 4.14】输入三角形的三边长，求三角形面积。

已知三角形的三边长 a,b,c，则该三角形的面积公式为：

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

其中 s = (a+b+c)/2

源程序如下：

```
#include<math.h>
main()
{
 float a,b,c,s,area;
 scanf("%f,%f,%f",&a,&b,&c);
 s=1.0/2*(a+b+c);
 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);
}
```

【例 4.15】求 $ax^2+bx+c=0$ 方程的根，a,b,c 由键盘输入，设 $b^2-4ac>0$。

求根公式为：

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

令 $q=\frac{\sqrt{b^2-4ac}}{2a}$，$q=\frac{\sqrt{b^2-4ac}}{2a}$

则 $x_1=p+q$

$x_2=p-q$

源程序如下：

```
#include<math.h>
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("\nx1=%5.2f\nx2=%5.2f\n",x1,x2);
}
```