第3章 顺 序 结 构

C 语言是一种结构化程序设计语言，具有结构化程序设计语言的一切特点。顺序结构是最简单的 C 语言程序结构，它不需要专门的语句来控制流程。本章所介绍的语句，将按它们在程序中出现的先后顺序逐条执行，在执行这些语句的过程中不会发生程序流程的转移，由这样的语句构成的程序结构就是顺序结构。

3.1 C 语言程序的语句

在第 1 章中介绍过，C 语言程序由函数构成，一个 C 语言函数通常由函数的首部和函数体两部分组成，而函数体一般包括说明部分（由若干条说明语句组成）和执行部分（由若干条执行语句组成）。C 语言的任何语句都必须以“;”作为语句的结束标志，“;”是 C 语句的必要组成部分。

3.1.1 说明语句

C 语言规定，函数中使用的所有变量（或数组）必须在使用前进行定义，否则会在编译时出错。如果程序中不使用变量（或数组），当然也可以没有变量定义语句。说明语句包括变量（或数组）定义语句和函数声明语句两种，最常用的是变量（或数组）定义语句。通常是通过变量定义语句来确定变量的类型与初值的。例如：

```
char ch1,ch2;           /*定义 ch1,ch2 为字符型*/
int x,y,z=1;            /*定义 x,y,z 为整型,z 初值为 1*/
float a,b,c;            /*定义 a,b,c 为单精度浮点型*/
double d1,d2;           /*定义 d1,d2 为双精度型*/
```

3.1.2 执行语句

程序的功能是由若干条执行语句实现的。执行语句可分为 5 类：表达式语句、函数调用语句、复合语句、空语句和控制语句。

1．表达式语句

表达式语句由任意表达式加上语句结束符分号“;”组成，其一般形式为：

```
表达式;
a+b;                    /*加法运算语句。但计算结果不能保留,无实际意义*/
a=10,b=20,c=a+b;        /*逗号表达式语句*/
```

在 C 语言中，表达式语句的表达能力很强，使用也很方便。

2．函数调用语句

由函数调用表达式加上分号即构成了函数调用语句。例如：

```
printf("What are you doing?");      /*输出语句*/
scanf("%d",&x);                     /*输入语句*/
```

3．复合语句

在 C 语言中，复合语句也可称为“语句块”，将若干条语句用一对花括号“{}”括起来便构成了复合语句。花括号内可以包含任何 C 语言语句，其一般形式如下：

```
{
   语句1
   语句2
   ……
   语句n
}
```

例如：

```
{ i=5; j*=i; }
```

说明：

① 一条复合语句在语法上作为一条语句处理，在一对花括号中的语句数量不限。在 C 语言程序中，凡是可以出现单语句的地方，都可以使用复合语句。

② 在书写复合语句时，要注意花括号必须配对。复合语句中右花括号的后面不加分号。

③ 在复合语句中，不仅可以有执行语句，还可以有说明语句，说明语句应该出现在可执行语句的前面。例如：

```
{ int a=5; b=a*a; }
```

4．空语句

只有一个分号“;”组成的语句，被称为空语句。例如：

```
main( )
{ ; }
```

空语句的语义是什么也不执行。在程序设计中有时需要加上一个空语句来表示存在一条语句，有待后续开发。但是随意加上分号会造成逻辑上的错误，所以应该慎用。

5．控制语句

顾名思义，控制语句的作用是控制程序的流程，以实现程序的分支结构和循环结构。C 语言只有 9 种控制语句，可分成以下 3 类：

① 条件语句，用于控制分支结构的语句，例如，if 语句、switch 语句；

② 循环语句，用于控制循环结构的语句，例如，while 语句、do while 语句、for 语句；

③ 转移语句，用于调整程序流程的语句，例如，goto 语句、break 语句、continue 语句、return 语句。

3.2　赋值运算

3.2.1　赋值运算符和赋值表达式

1．赋值运算符

在 C 语言中，称“=”为简单赋值运算符，简称赋值运算符。在赋值运算符之前加上双目运算符（中间不能有空格）构成的新的自运算符称为复合赋值运算符，参见表 3.1。

表 3.1 赋值运算符

类　别	运 算 符	名　称	示例表达式	等价表达式
（简单）赋值运算符	=	赋值运算	a=10	
（算术）复合赋值运算符	+=	加赋值	a+=b	a=a+b
	−=	减赋值	a−=b	a=a−b
	*=	乘赋值	a*=b	a=a*b
	/=	除赋值	a/=b	a=a/b
	%=	取余赋值	a%=b	a=a%b

说明：赋值运算符均为双目运算符，优先级仅高于逗号运算符，其结合性为右结合性。

2．赋值表达式

用赋值运算符将一个变量和一个表达式连接起来的式子称为赋值表达式，参见表 3.1 中的示例。赋值表达式的一般形式为：

变量 赋值运算符 表达式

例如：a=10;

a+=10;

赋值运算的功能：将赋值运算符右边表达式的值存放到以左边变量名为标识的存储单元中。对于（算术）复合赋值运算来说，它实现的操作是将其左边变量的值与右边表达式的值进行加（或减、乘、除、求余等）运算后再存放到以左边变量名为标识的存储单元中。

赋值表达式的值就是被赋值后赋值号左边变量的值。

例如，赋值表达式“a=10”运算后，有两层意思：一是使变量 a 的值为 10，即将 10 放到变量 a 对应的存储单元中，不论变量 a 的原值是多少，执行上述赋值操作后（或称赋值运算），a 的值更新为 10；二是求得赋值表达式“a=10”的值为 10。

将赋值表达式作为表达式的一种，不仅可以形成赋值表达式语句，而且可以以表达式形式出现在其他语句（如循环语句）中，这是 C 语言灵活性的一种表现。

3.2.2 赋值语句

赋值语句实际上就是在赋值表达式末尾加上分号，它的一般形式为：

赋值表达式;

例如，分析下列程序段的输出结果：

```
float lengh,width,area;      /*定义变量*/
length=12.5;                 /*下面通过 3 条赋值语句为 3 个变量赋值*/
width=7.6;
area=length*width;
printf("area=%f\n",area);
```

关于赋值语句说明几点：

① 赋值运算符“=”的左边必须是变量，右边的表达式可以是单一的常量、变量、函数调用或表达式。例如，下面都是合法的赋值语句：

```
x=10;
y=x+10;
```

```
z=sqrt(2);
```

② 赋值运算符“=”不同于数学中使用的等号，它没有相等的含义。

例如，x=x+1；其含义是取出变量 x 中的值加 1 后，再存入变量 x 中。

③ 赋值运算符的结合性为“右结合性”。

例如：　x=y=z=8；　　等价于　　x=(y=(z=8))；

运算时，先求赋值表达式“z=8”的值（得 8），其值再赋给变量 y（得 8），再把表达式“y=8”的值赋给变量 x（得 8）。

④ 赋值语句的主要功能就是给变量赋值，在程序中可给一个变量多次赋值，但应注意变量的当前值。

例如，分析下列程序段执行后，注意变量 a 的值的变化：

```
int a, b, c;
a=5 ;                    /*a 的值为 5*/
a=(b=a+2);               /*a 的值为 7*/
a=(b=b+1)+(c=3);         /*a 的值为 11*/
```

⑤ 关于复合赋值运算，请注意是将赋值运算符左侧变量的值和右边整个表达式的值进行适当运算后再赋值。

例如，赋值语句：x*=y+8；　等价于　　x=x*(y+8)；

对于初学者来说，容易错误地认为等价于语句：　x=x*y+8；

又如，分析下列程序段执行后，各变量的值更新为多少？

```
int i=2，j=12，k=10;
k+=j+=i+8;
```

分析：上述语句中“+”优先级高于“+=”，故先运算 i+8 值为 10；同一优先级的两个复合赋值运算符“+=”因结合性为右结合性，所以先运算右面的“+=”（即 j+=10 的值为 22），再运算左面的“+=”（即 k+=22 的值为 32）。

C 语言采用这种复合赋值运算符，一是为了简化程序，使程序精练，二是为了提高编译效率，因为从编译的角度来看，它可以生成更短小的汇编代码。

⑥ 赋值运算中的类型转换。

当赋值运算符左边变量的类型和右边表达式值的类型一致时，直接赋值。如果类型不一致，首先要把赋值运算符右边表达式值的类型强制转换为左边变量的类型，然后再进行赋值。转换规则见表 3.2。

表 3.2　赋值运算中数据类型的转换规则

左边变量的类型	右边表达式值的类型	转 换 说 明
int	double	将实型数据的小数部分截去后再赋值
double	int	将整型数据转换成实型数据后再赋值
int	char	值不变，高 24 位补 0，或进行符号扩展
long,int	short	值不变，高 16 位进行符号扩展
int,short	long	右侧的值不能超过左侧数据值的范围，否则将导致意外的结果
unsigned	signed	按原样赋值。但是如果数据范围超过相应整型数据的范围，将导致意外的结果
signed	unsigned	

3.2.3 自增、自减运算符

当复合赋值运算是自加 1 或自减 1 的时候，C 语言提供了更为优化的运算符——自增（++）、自减（––）运算符。

设整型变量 a，初值为 10。要实现对其加 1，已知可以有以下两种写法。

方法 1：a=a+1;

方法 2：a+=1;

现在还有方法 3，并且是最好的方法，即“++a;”或者“a++;”。

也就是说，只有在自加 1 的情况下，代码 a++或++a 可以生成最优化的汇编代码。

同样，自减 1 操作也有对应的运算符：––a 或 a––。

设 a 原值为 10，则执行––a 或者 a––后，a 的值都为 9。

所以，自增“++”和自减“––”运算符的作用是使运算对象的值增 1 或减 1，它们是两个单目运算符，可置于运算对象的左侧或右侧，例如，++i、i++、––i、i––等都属于合法的表达式。参加自增、自减运算的运算对象只能是变量而不能是表达式或常量。表 3.3 列出了自增、自减运算符的种类和功能。

表 3.3 自增、自减运算符的种类和功能

运 算 符	名 称	例 子	相 当 于
++	自增	i++或++i	i=i+1 或 i+=1
––	自减	i––或––i	i=i−1 或 i−=1

自增、自减运算符出现在变量之前（如++i 和––i），称为前置（prefix）运算，出现在变量之后（如 i++和 i––），称为后置（postfix）运算。对一个变量 i 实行前置运算或后置运算，其运算结果是一样的，即都使变量的值增 1 或减 1，但作为表达式来说却有着不同的值。例如，假设 i 的初值等于 3，则：

```
++i   /*先使 i 的值加 1,然后再使用 i,表达式的值是 i 加 1 之后的值为 4*/
i++   /*先使用 i 的值,然后再使 i 的值加 1,表达式的值是 i 加 1 之前的值为 3*/
```

所以赋值表达式 j=++i 相当于 i=i+1 和 j=i，赋值表达式的值是 4，变量 i 的值也是 4。而赋值表达式 j=i++则相当于 j=i 和 i=i+1，赋值表达式的值是 3，变量 i 的值是 4。

例如，已知定义语句：

```
int a=3,b=5,c;
```

则有：

（1）c=(++a)*b;　　等价于：a=a+1;
　　c=a*b;

结果：c 的值为 20。

（2）c=(a++)*b;　　等价于：c=a*b;
　　a=a+1;

结果：c 的值为 15。

【例 3.1】 自增、自减运算符使用示例。

程序代码如下：

```
#include "stdio.h"
int main( )
{  int i=5,j,k;
```

```
    j=++i;                          /*i 的值自加 1 变为 6, j 的值为 6*/
    i=5;
    k=i++;                          /*k 的值为 5,i 的值为 6*/
    printf("i=%d,j=%d,k=%d\n",i,j,k);
    return 0;
}
```

程序运行结果如下：

```
i=6, j=6, k=5
```

说明：

① 自增、自减运算符（++、––）只能用于变量，不能用于常量和表达式，例如，2++或(x+y)++都是不合法的。

② ++和––的结合方向是“右结合性”，其优先级高于基本算术运算符，与负号运算符为同一优先级。例如，–i++，因为“–”运算符和“++”运算符优先级相同，而结合方向为“自右至左”，即它相当于–(i++)。

3.3　数据的输入和输出

将数据通过计算机外部设备送到计算机内存中的操作称为输入，反之，将数据从计算机内存送到计算机外部设备的操作称为输出。在这里，操作系统默认的标准输入/输出设备是键盘和显示器。

首先说明，C 语言本身不提供输入/输出语句，它的输入/输出操作都是通过调用 C 语言系统提供的输入/输出标准函数来实现的。例如：

```
printf("What are you doing?");
scanf( “%d” ,&x);
```

是函数调用语句，而不是 C 语言提供的输入/输出语句。

C 语言提供的输入/输出标准函数被存放在标准函数库中，输入/输出类函数的头文件为 stdio.h，如果要使用这些输入/输出函数，必须在源程序的开始处使用编译预处理命令：#include "stdio.h"（关于 C 语言的预处理功能详见第 6 章）。

本节主要介绍两类常用的标准输入/输出函数：一类是用于单个字符的输入/输出函数 getchar( )和 putchar( )；另一类是用于各类数据的格式化输入/输出函数 scanf( )和 printf( )，其中最后一个字符 f 是 format 的缩写，即格式的意思。

3.3.1　单个字符的输入和输出函数

1．单个字符输入函数 getchar( )

getchar( )函数调用的一般形式为：

```
变量=getchar( );
```

该函数的功能是从标准输入设备（键盘）上输入一个字符，按回车键后，getchar( )将接收该字符作为函数的值。通常将 getchar( )函数得到的值通过赋值运算符赋给某个字符型变量。

例如：

```
char ch;
ch=getchar( );
ch++;
```

当程序执行到“ch=getchar( );”语句时，程序会暂停，等待用户从键盘输入一个字符，然后继续执行后面的语句。

注意：getchar( )只能接收一个字符，而且只有在用户按回车键<Enter>后，读入才开始执行。

2．单个字符输出函数 putchar ( )

putchar( )函数调用的一般形式为：

```
putchar(ch)
```

函数的功能是向标准输出设备（显示器）输出一个字符（即 ch 的值），其中，ch 可以是字符型常量、变量或整型变量。

【例 3.2】 从键盘输入一个字符并显示该字符。

```
#include "stdio.h"
int main( )
{  char ch;
   ch=getchar( );
   putchar(ch);
   return 0;
}
```

该程序在运行到“ch=getchar( );”语句时会暂停，等待用户从键盘上输入一个字符。如果用户从键盘输入：c <Enter>，就会在屏幕上看到字符 c。执行情况如下：

```
输入：c <Enter>
输出：c
```

注意：getchar( )函数得到的字符可以赋给一个整型变量或字符型变量，也可以不赋给任何变量而作为表达式的一部分。例如，可以将例 3.2 程序中的后两行用下面的语句代替：

```
putchar(getchar( ));
```

需要说明的是，getchar( )函数可接收任何键盘输入，请仔细阅读并理解下面程序示例。

【例 3.3】 单个字符输入、输出函数示例。

```
#include "stdio.h"
int main( )
{  char a,b;
   a=getchar();
   b=getchar();
   putchar(a);putchar(b);
   return 0;
}
```

第 1 次运行程序：

（1）输入：AB<Enter>

输出：AB

第 2 次运行程序：

（2）输入：A□B<Enter>

输出：A□

说明：“□”表示空格。

解释：第 1 次运行程序，通过键盘用户输入了 3 个字符 AB<Enter>，两次 getchar( )调用，顺序读取 A 和 B 赋给变量 a 和 b; 第 2 次运行程序，通过键盘用户输入了 4 个字符 A□B<Enter>，两次 getchar( )调用，顺序读取 A 和□赋给变量 a 和 b。

putchar( )函数也可以输出转义字符，例如，putchar('\n') 输出一个换行符。如果将例 3.3 中程序的最后一行改为：

```
putchar(a); putchar ('\n');putchar (b);
```

则运行结果为：

```
输入：AB<Enter>
输出：A
      B
```

putchar( )函数还可以输出其他转义字符，例如：

```
putchar ('\102');            /*输出字符“B”，显示内容为：B   */
putchar ('\");               /*输出单引号字符“ ' ”，显示内容为：'   */
putchar ('\012');            /*输出换行符*/
```

3.3.2　格式化输入和输出函数

1．格式化输出函数 printf( )

putchar( )函数只能输出单个字符。如果用户在程序中需要输出若干个任意类型的数据，就要使用在前面章节中用到的格式输出函数 printf( )来实现。printf( )函数在整个 C 语言程序设计中应用非常广泛，希望大家能够很好地掌握。

printf( )函数调用的一般形式如下：

```
printf("控制字符串",输出项列表);
```

printf( )函数的功能是，按控制字符串规定的输出格式，将输出项列表中的各输出项的值依次输出到系统指定的标准输出设备（显示器）上。

下面分别介绍 printf( )函数中各参数的含义。

（1）**控制字符串**

控制字符串是用双引号括起来的字符串，也称转换控制字符串，它包含格式说明和普通字符两种信息。

① 格式说明

格式说明由“%”和跟随其后的一个格式字符组成。它的作用是将要输出的数据转换为指定的格式输出。格式说明总是由“%”字符开始，以一个格式字符作为结束，对不同类型的数据应使用不同的格式字符控制其输出格式。C 语言提供的 printf( )格式字符及其功能如表 3.4 所示。

表 3.4　printf( )格式字符及其功能

格 式 字 符	意　义
d	以十进制有符号形式输出整型数据
o	以八进制无符号形式输出整型数据（不带前导 0）
x	以十六进制无符号形式输出整型数据（不带前导 0x）
u	以十进制无符号形式输出整型数据
c	输出一个字符

续表

格 式 字 符	意　义
s	输出字符串中的字符，直到遇到“\0”，或者输出由精度指定的字符数
f	以小数形式输出单精度和双精度数据，隐含的小数位数为 6
e	以规格化的指数形式输出单精度和双精度数据，隐含的小数位数为 6
g	按 e 和 f 格式中宽度较短的一种输出，不输出无意义的 0

在一些系统中，这些格式字符只允许使用小写字母，因此建议读者统一使用小写字母，使程序具有通用性。此外，还可以根据需要在“%”和格式字符之间插入“宽度说明”“左对齐符号-”“前导零符号 0”等附加格式说明，格式为：

```
%[+][-][0][#][m.n][l]格式字符;
```

printf( )附加格式说明符及其功能如表 3.5 所示。

表 3.5　printf( )附加格式说明字符及其功能

字　符	意　义
字母 l	用于 long 和 double 型数据的输出，可加在格式符 d、o、x、u、e、f 前面
m（正整数）	指定输出数据所占的宽度，若输出的数据位数大于 m，为保证数据的正确性，则按实际位数输出；如果数据的位数小于 m，则多出的位数补空格
.n（正整数）	.n 称为精度。对于实数，表示输出 n 位小数；对于字符串，表示截取的字符个数；对于整数，指定必须输出的数字个数，若输出的数字少于指定的个数，则前面补 0，否则按原样输出
-	输出的数字或字符左对齐
+	使输出的数字总是带＋或 –号
0	在输出的数据前加前导 0
#	使输出的八进制数（或十六进制数）带前导 0（或 0x）。注意：#只对 o 格式字符和 x 格式字符起作用

② 普通字符

普通字符是需要原样输出的字符，它包含可打印的字符和不可打印的字符。可打印的字符在“控制字符串”中直接用字符符号表示，如 a、b 等；不可打印的字符用转义字符表示，如换行'\n'，横向跳格'\t'、响铃'\a'等。

（2）**输出项列表**

输出项列表是需要输出的一些数据，可以是一个或者多个输出项。当有多个输出项时，各输出项之间用逗号“,”隔开。输出项可以是常量、变量或表达式。

输出项与控制字符串中的格式说明从左到右在类型上必须一一对应匹配。如果不匹配将导致数据不能正确输出。

另外，输出项的个数与控制字符串中格式说明的个数应该相同。如果输出项的个数多于格式说明的个数，则多余的输出项不输出；如果输出项的个数少于格式说明的个数，则对于多余的格式说明将输出不定值（或 0 值）。

例如，以下程序段：

```
int x=10;double y=2.5;
printf("x=%d,y=%lf,s=%s\n",x+5,y, "End");
```

输出结果为：

```
x=15,y=2.500000,s=End
```

其中：输出项 x+5 与%d 相对应；y 与%lf 相对应，"End"与%s 相对应；按照相应的格式输出；而"x="、"y="、"s="、"\n "及逗号为普通字符，将原样输出。

pritnf( )中如果省略输出项，且控制字符串中也没有格式说明，其结果是将该字符串原样显示。

例如，以下程序段：

```
printf("What is your name? \n");
printf("My name is Li li. \n");
```

该程序段执行后，屏幕显示：

```
What is your name?
My name is Li li.
```

【例 3.4】　格式输出函数示例。

```
#include "stdio.h"
int main( )
{  int a=5,b=8;
   printf("%d%d%d\n",a,b,a+b);
   printf("%d %d %d\n",a,b,a+b);
   printf("%d,%d,%d\n",a,b,a+b);
   printf("a=%d,b=%d,a+b=%d\n",a,b,a+b);
   return 0;
}
```

程序运行结果如下：

```
5813
5 8 13
5,8,13
a=5,b=8,a+b=13
```

从输出结果可以看出，清晰度越来越好，恰当地设计输出样式可以得到友好的输出结果。

另外，如果希望输出字符“%”，使用特殊格式说明“%%”。例如：

```
printf("%d%%\n",10);
```

执行后将输出：

```
10%
```

2．格式化输入函数 scanf( )

scanf( )函数调用的一般形式为：

```
scanf ("控制字符串"，输入项地址列表)
```

scanf( )函数的功能是，按控制字符串规定的输入格式，从系统指定的标准输入设备（键盘）上将输入的数据依次存到输入项地址列表所指定的内存单元中。

下面分别介绍 scanf( )函数中各参数的含义。

（1）**控制字符串**

控制字符串是用双引号括起来的字符串，也称转换控制字符串，它规定了输入数据的输入格式。它包含格式说明和普通字符两种信息。

① 格式说明

与 printf( )函数中的格式说明类似，scanf( )函数的格式说明也是以“%”开始，以一个格式字符结束的，中间可以插入附加格式说明符。表 3.6 列出了 scanf( )函数用到的格式字符，表 3.7 列出了 scanf( )函数常用的附加说明符（修饰符）。

表 3.6 scanf( )格式字符

格 式 字 符	意 义
d	输入一个十进制整型数据
i	输入一个整型数据，该数据也可以是带前导 0 的八进制数或带前导 0x 的十六进制数
o	以八进制形式输入一个整型数据（可以带前导 0，也可以不带前导 0）
x	以十六进制形式输入一个整型数据（可以带前导 0x，也可以不带前导 0x）
u	输入一个无符号十进制整型数据
c	输入一个字符型数据
s	输入一个字符串，将字符串送到一个字符数组中，在输入时以非空格字符开始，以第一个空格字符结束
f	以小数形式或指数形式输入单精度数
e	与 f 的作用相同

表 3.7 scanf( )附加的格式说明符

字 符	意 义
l	用于输入 long 型数据（%ld,%lo,%lx）和 double 型数据（%lf 或%le）
m（正整数）	指定输入数据所占的宽度（列数）

② 普通字符

普通字符在输入数据时要求原样输入相同的字符。一般情况下，最多只在格式字符之间加逗号普通字符。

（2）**输入项地址列表**

输入项地址列表是由若干个地址组成的列表，可以是变量的地址或字符串的首地址等。若为多项，各项之间用逗号隔开。

例如：
```
int a,b;
scanf ("%d,%d",&a ,&b);
```
初学者往往写成：
```
scanf ("%d,%d",a ,b);
```
这是错误的，因为 a 和 b 不是地址形式。需要说明的是，虽然语句是错的，但是编译时并不报错，只是变量 a 和 b 会得到一个错误的值。

为了能正确地通过键盘输入数据，说明如下：

① 当调用 scanf( )函数从键盘输入数据时，最后一定要按下回车键，scanf( )函数才开始接收从键盘上输入的数据，并且输入项地址列表与控制字符串中的格式说明从左到右在个数和类型上必须一一对应匹配，如果不匹配将得不到正确的数据。

② 当指定输入数据所占的宽度为 m 时，系统自动按宽度 m 截取所需数据，但不能对实型数据指定小数位的宽度。例如，以下程序段：
```
int a,b;
scanf ("%2d%3df",&a，&b );
printf ("a=%d,b=%d\n",a，b );
```
键盘输入：123456<Enter>

输出：a=12,b=345

而“scanf ("%6.3f",&a); ”是不合法的，不能企图输入以下信息而使 a 的值为 123.456：

```
123456<Enter>
```

③ 在输入数据时，遇到下列情况之一时认为该数据项结束：

- 空格符（空格键）、制表符（Tab 键）或回车符（回车键）；
- 宽度结束，如“%2d”，则只取两列；
- 非法输入。

④ 当输入的数据少于输入项的个数时，程序等待输入，直到输入的数据个数与输入项的个数相同为止；当输入的数据多于输入项的个数时，多于的数据将留作下一个输入函数的输入数据。

⑤ 特别强调：

- 地址表列问题，如：

```
scanf( “%d,%f” ,a,b);                    //忘记&运算符
```

编译时系统并不报错，只是变量获得的是不确定的错误值

- 普通字符问题，如：

```
scanf( “%d,%f\n” ,&a,&b);                //换行符\n 在这被当做两个普通字符\和 n
```

输入的数值序列应该是：

```
12,12.5\n
```

【例 3.5】　格式输入函数示例。

```
#include "stdio.h"
int main( )
{  int a,b; char c;
   scanf("%d%c%d",&a,&c,&b);
   printf("%d,%d,%c\n",a,b,c);
   return 0;
}
```

程序运行结果如下：

```
输入：33a66 <Enter>   (输入 a、c、b 的值)
输出：33,66,a         (输出 a、b、c 的值)
```

其中，根据 scanf( )函数的功能，将从键盘上输入的 3 个数据 33、66 和字母 a 分别存入系统为变量 a、b、c 分配的内存单元中，如图 3.1 所示。第一个数据对应格式%d 输入 22 之后遇字母 a，因此认为数据项 22 后已没有数字了，第 1 个数据到此结束，把 22 赋给变量 a；第 2 个数据对应格式%c，因此将字符 a 赋给变量 c；第 3 个数据 66 后面是回车键，认为此数据项到此结束，则将 66 赋给变量 b。

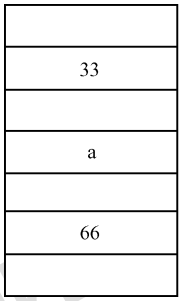

图 3.1　变量 a、b、c 中的值

3.4　程 序 举 例

【例 3.6】　已知 *a*=5.0，*b*=2.5，*c*=7.8，计算

$$y=\frac{\pi ab}{a+bc}$$

分析：由已知条件定义变量 a、b、c 为实型且赋初值，定义符号常量 PI 表示圆周率π，定义 y 为实型变量来存储以上公式的计算结果。

程序代码如下：

```
#include "stdio.h"
#define PI 3.14
int main( )
{  double a=5.0,b=2.5,c=7.8,y;
   y=PI*a*b/(a+b*c);              /*适当使用括号写出正确的 C 表达式*/
   printf("y=%lf\n",y);
   return 0;
}
```

程序运行结果为：

```
y=1.602041
```

【例 3.7】 从键盘输入两个整数赋给变量 a 和 b，要求交换变量 a 和 b 中的值。

分析：定义 3 个整型变量 a、b 和 t，a 和 b 两个变量分别存储从键盘输入的两个整数，这两个整数可以调用 scanf( )函数实现输入。

交换 a 和 b 中值的方法是，首先将 a 中的值用临时变量 t 保存起来（在此可通过赋值语句“t=a;”来实现），然后将 b 的值赋给 a（即“a=b;”），再把保存在临时变量 t 中的值赋给 b（即“b=t;”）。根据此思路编写程序代码如下：

```
#include "stdio.h"
int main( )
{   int a,b,t;
    scanf("%d%d",&a,&b);
    printf("a=%d,b=%d\n",a,b);
    t=a;a=b;b=t;       /*注意 3 条赋值语句的顺序*/
    printf("a=%d,b=%d\n",a,b);
    return 0;
}
```

程序运行结果为：

```
输入: 22  66<Enter>
输出: a=22,b=66
      a=66,b=22
```

【例 3.8】 从键盘输入一个大写字母，要求改用小写字母输出，并输出大写字母和小写字母的 ASCII 码值。

分析：定义 c1、c2 两个字符型变量来分别存储大写字母和小写字母，大写字母可以用 getchar( )函数实现输入。大小写字母间转换的方法前面已经介绍过。根据此思路编写程序代码如下：

```
#include "stdio.h"
int main( )
{  char c1,c2;   c1=getchar( );
   printf("%c,%d\n",c1,c1);
```

```
    c2=c1+32;
    printf("%c,%d\n",c2,c2);
    return 0;
}
```

程序运行结果为：

```
输入: A<Enter>
输出: A, 65
      a, 97
```

【例 3.9】 从键盘输入一个 5 位正整数，求出个位数和百位数并输出，如输入 12345，则输出 35。

分析：定义 3 个整型变量 a、b 和 m。m 变量用于存储从键盘输入的 5 位正整数，这个整数可以调用 scanf( )函数实现输入；a 和 b 两个变量分别存储求出的个位数和百位数。

求个位数的方法：用 m 除以 10 的余数就是个位数（在此可通过表达式“m%10”来实现）。

求百位数的方法：用 m 除以 100 的商再除以 10 的余数就是百位数（在此可通过表达式“m/100%10”来实现）。根据此思路编写程序代码如下：

```
#include "stdio.h"
int main( )
{  int  a, b, m ;
   scanf("%d",  &m );
   a=m%10;  b=m/100%10;
   printf("%d%d\n", b, a);
   return 0;
}
```

程序运行结果为：

```
输入: 12345<Enter>
输出: 35
```

【例 3.10】 设一元二次方程为 $ax^2+bx+c=0$，输入 3 个系数 a、b、c（设 a 不为 0，且 $b^2>4ac$），求两个实根。

分析：定义变量 a、b、c 为实型，代表方程的 3 个系数，可以用 scanf( )函数实现数据的输入；定义 $x1$、$x2$ 两个实型变量来存储两个实数根。

一元二次方程的求根公式为：

$$x1=\frac{-b+\sqrt{b^2-4ac}}{2a} \qquad x2=\frac{-b-\sqrt{b^2-4ac}}{2a}$$

程序代码如下：

```
#include "stdio.h"
#include "math.h"
int main( )
{  double a,b,c,x1,x2;
   scanf("%lf%lf%lf",&a,&b,&c);
   x1=(-b+sqrt(b*b-4*a*c))/(2*a);
   x2=(-b-sqrt(b*b-4*a*c))/(2*a);
   printf("x1=%5.2lf\nx2=%5.2lf\n",x1,x2);
   return 0;
}
```

程序运行结果为：

```
输入: 2.3  6.7   3.1<Enter>
输出: x1=-0.58
      x2=-2.34
```

本例中，sqrt( )是求平方根函数，其头文件为 math.h，所以在程序的开头加#include "math.h"语句。

从本节的例题可以看出，顺序程序结构是一种按照语句书写顺序执行的简单的程序结构，可解决一些简单的问题。

本 章 小 结

1．C 语言的任何语句都必须以“;”作为语句的结束标志，“;”是 C 语句的必要组成部分。

2．C 语言的语句可分为说明语句和可执行语句两类，其中可执行语句分为表达式语句、函数调用语句、复合语句、空语句和控制语句 5 种。

3．赋值语句是程序设计中使用最频繁的语句，主要功能是给变量赋值。

4．C 语言中没有提供专门的输入/输出语句，所有的输入/输出操作都是由调用标准函数库中的输入/输出函数来实现的。

- getchar( )和 scanf( )函数是输入函数，用于接收来自键盘的输入数据。getchar( )函数是字符输入函数，只能接收单个字符；scanf( )是格式输入函数，可按指定的格式输入若干个任意类型的数据。
- putchar( )和 printf( )函数是输出函数，在显示器屏幕上输出信息。putchar( )函数是字符输出函数，只能显示单个字符；printf( )是格式输出函数，可按指定的格式输出若干个任意类型的数据。

习 题 3

3.1 思考题


选择题讲解视频

1．赋值的含义是什么？请给出下列赋值语句（序列）的功能。

（1）`a=10;` 功能：_____

（2）`a*=3+5;` 功能：_____

（3）`a+=b; b=a-b; a=a-b;` 功能：_____

（4）`a=b%10*10+b/10;` 功能：_____

（5）`a='A'+32;` 功能：_____

（6）`a=36%10+'0';` 功能：_____


填空题讲解视频

2．给定下列变量的定义语句。

```
int num;
double score, sum;
```

判定并修改下列 scanf( )函数调用语句的错误。

（1）`scanf("%d",num);`

（2）`scanf("%d,%d",&score,sum);`

（3）`scanf("%d,%f,%lf",&num,&score,sum);`

（4）`scanf(&num,"%d");`

3．请为下列 scanf( )函数调用编写适当的键盘输入数据序列（要求给 i 赋 10，给 j 赋 20，给 c 赋'A'）。

（1）`scanf("%d,%d",&i,&j);`

（2）`scanf("%d%d",&i,&j);`

（3）`scanf("i=%d,j=%d",&i,&j);`

（4）`scanf("%d,%c,%d",&i,&c,&j);`

4. 按要求写出正确的语句。

（1）输入 3 个 double 型数据给变量 a,b,c。输入数据时用一个空格间隔，写出正确的 scanf( )语句。

（2）输出 double 型变量 a,b,c 的平均值，保留 3 位有效位，写出正确的 printf( )语句。

（3）设用 char 型变量 ch 保存一大写字母，输出该大写字母本身及其 ASCII 码值，写出正确的 printf( )语句。

（4）设有 double a=1.25, b=2.32; 编写 printf( )语句（多个），输出下面的算式。

```
  1.25
+1.32
------
  2.57
```

3.2　读程序写结果题

1.
```
char c1='b',c2='e';
printf("%d,%c\n",c2-c1,c2-'a'+'A');
```

输出结果是：_____。

2.
```
double x1, x2;
 x1=3/2;
 x2=x1/2;
 printf("%d,%f", (int)x1, x2) ;
```

输出结果是：_____。

3.
```
#include "stdio.h"
int main( )
{ double d=3.2; int x,y;
  x=1.2;y=(x+3.8)/5.0;
  printf("%d \n", d*y);
  return 0;
}
```

输出结果是：_____。

4.
```
#include "stdio.h"
int main( )
{ int a=1, b=2;
  a=a+b; b=a-b; a=a-b;
  printf("%d,%d\n", a, b );
  return 0;
}
```

输出结果是：_____。

5.
```
#include "stdio.h"
int main()
{ char a='4',b='6',c='3';
  int x,y,z,m=0;
  x=a-'0';
  y=b-'0';
  z=c-'0';
```

```
    m=m*10+x;
    m=m*10+y;
    m=m*10+z;
    printf("x=%d,y=%d,z=%d,m=%d\n",x,y,z,m);
    return 0;
}
```

输出结果是：_____。

3.3 程序填空题：按照要求，请在下面画线处填写适当的内容。

1. 要求下列程序的输出结果是 16.00。

```
#include "stdio.h"
int main( )
{ int a=9, b=2;   double x=____, y=1.1,z;
  z=a/2+b*x/y+1/2;
  printf("%6.2lf\n", z );
  return 0;
}
```

2. 以下程序的功能是输入两个整型数后计算其商。例如，输入 5/2<回车>，则输出“5/2=2.50”。

```
#include "stdio.h"
int main( )
{ int a,b;  double c;
  scanf("_______",&a,&b);
  c=(double)a/b;
  printf("_________",a,b,c);
  return 0;
}
```

3.4 编程题

1. 编写程序实现从键盘输入两个十进制整型数据 10 和 8 给变量 x 和 y，并按下列格式输出。

	x	y
十进制数	10	8
八进制数	12	10
十六进制数	A	8

2. 编写一个程序，输入一个大写英文字母（'B'～'Y'），输出它的前导字母、该字母本身及其后续字母。

3. 编写一个程序，输入一个 3 位正整数，要求反向输出对应的整数，如输入 123，则输出 321。

4. 某工种按小时计算工资：

总工资=每月的劳动时间×每小时工资

从总工资中扣除 10%公积金，剩余的为应发工资。编程从键盘输入劳动时间和每小时工资，打印出应发工资数额。

5. 编写程序，读入 3 个整数给变量 a、b、c，然后交换它们的值，把 a 原来的值给 b，把 b 原来的值给 c，把 c 原来的值给 a。