第 3 章 顺序结构程序设计

数据描述和动作描述是程序设计的两项主要内容。前面介绍的基本数据类型、常量定义和变量说明，初步完成了第一项任务——对数据进行描述。而对动作的描述是由语句来实现的。本章从简单的程序入手，掌握语句的基本概念，逐步理解用 C 语言进行程序设计的方法。

3.1 C 程序的基本结构及 C 语句的种类

C 语言是一种结构化程序设计语言。它包含了实现各种操作的丰富语句。学习 C 语言编程首先应掌握这些语句的格式和用法。

3.1.1 结构化程序的 3 种基本结构

结构化程序的 3 种基本结构分别是顺序结构、选择结构和循环结构。这是一般的结构化程序所具有的通用结构。

C 语言的语句完全可以满足结构化程序的 3 种基本结构的要求。使用 C 语言所提供的语句可以方便地实现顺序结构、选择结构和循环结构的程序设计。

顺序结构是由若干条按先后顺序执行的语句构成的。

选择结构是由一个或多个条件来确定所要执行的语句，又称为分支结构。根据实际情况可分为二支或多支，通常使用选择语句来实现。选择结构增加了编程的灵活性，它是 C 语言编程中常用的一种语法结构。

循环结构是当满足某种循环条件时反复执行某段程序，直到不满足循环条件为止。这种结构可使程序简洁明了，它也是 C 语言编程中常用的一种方法。

C 语言程序充分体现了结构化程序的三种基本结构。

3.1.2 C 语句的种类

下面介绍语句的种类及特点。

1．表达式语句和空语句

表达式语句是由表达式加分号（;）构造的一种简单语句。该语句在 C 语言程序中出现得最多，例如：

```
int a=1,b=2;
a*b+5;              （算术表达式语句）
a!=b*2;             （关系表达式语句）
!b||a&& b;          （逻辑表达式语句）
b=a−2;              （赋值表达式语句）
b<a?a:b;            （条件表达式语句）
a=5,b=7,a+b;        （逗号表达式语句）
```

总之，任何一种合法的表达式加分号都可构成表达式语句。

表达式语句与表达式的区别在于一个分号。在实际使用中，一定要区分表达式和表达式语句的用法。在要求使用表达式时，一定不能用表达式语句，反之亦然。例如，在 if 的条件语句中，要求用表达式，这时不能给出表达式语句。

空语句是一种只有分号而无表达式的特殊语句。空语句的功能是不做任何操作，它只是形式上的语句，它是 C 语句中最简单的语句。

空语句通常用在需要一条不做任何操作的语句的地方。例如，循环语句的循环体，goto 语句所转向的语句等。

2．复合语句

复合语句是由两条或两条以上的语句用花括号（{}）括起来的语句序列。复合语句是相对于单条语句而言的。复合语句通常可以出现在允许单语句出现的地方，复合语句可以等价于一条语句。

在 C 语言程序中，复合语句可以嵌套使用，即在复合语句中还可以包含复合语句。下面就是在复合语句中嵌套复合语句的格式：

```
…
{
   <语句序列>
   {
      <语句序列>
   }
   <语句序列>
}
…
```

复合语句和函数体都采用花括号括起来的若干条语句的格式，但二者是不同的。复合语句被定义在函数体内，即函数体内可以包含若干个复合语句，而复合语句中不可能出现函数体。另外，函数体内可以是一条语句，也可以没有语句。

3．分支语句

该语句可以实现选择结构，C 语言提供了两种分支语句，一种是条件语句，另一种是开关语句，使用它们可以实现多路分支。

4．循环语句

循环语句可以实现循环结构。C 语言提供了三种循环语句：while 循环语句、do…while 循环语句和 for 循环语句。

5．转向语句

执行转向语句时，系统转去执行指定的语句。C 语言提供的转向语句有退出语句 break、继续语句 continue、无条件转向语句 goto。使用这些语句将给程序增加灵活性和方便性。

3.2　数据输入输出的实现

所谓输入输出是以计算机主机为主体而言的。从计算机向外部输出设备（如显示器、打印机、磁盘等）输出数据称为“输出”，从输入设备（如键盘、磁盘、光盘、扫描仪等）向计

算机输入数据称为“输入”。

C 语言本身不提供输入输出语句，输入和输出操作是由函数来实现的。在 C 标准库函数中提供了一些输入输出函数，例如，printf 函数和 scanf 函数。不把输入输出作为 C 语言提供的语句的目的是使 C 语言编译系统简单，因为将语句翻译成二进制的指令是在编译阶段完成的，没有输入输出语句就可以避免在编译阶段处理与硬件有关的问题，可以使编译系统简化，而且通用性强，可移植性好，对各种型号的计算机都适用，便于在各种计算机上实现。各种版本的 C 语言库函数是各计算机厂商（或软件开发公司）针对某一类型计算机的情况编写的，并且已编译成目标文件（.obj）。它们在连接阶段与由源程序经编译而得到的目标文件相连接，生成一个可执行的目标程序。如果在源程序中有 printf 函数，在编译时并不把它翻译成目标指令，而是在执行阶段调用已被连接的函数库中的 printf 函数。

由于 C 编译系统与 C 函数库是分别进行设计的，因此不同的计算机系统所提供函数的数量、名字和功能是不完全相同的。不过，有些通用的函数（如 printf 和 scanf 等），各种计算机系统都提供，称为各种计算机系统的标准函数。

C 语言库函数中有一批“标准输入输出函数”，它是以标准的输入输出设备（一般为终端设备）为输入输出对象的。其中有：putchar（输出字符），getchar（输入字符），printf（格式输出），scanf（格式输入），puts（输出字符串），gets（输入字符串）。在本章中介绍前面 4 个最基本的输入输出函数。

在使用 C 语言库函数时，要用预编译命令“#include”将有关的“头文件”包含到用户源文件中。在头文件中包含了与用到的函数有关的信息。例如，使用标准输入输出库函数时，要用到“stdio.h”文件。文件后缀中“h”是 head 的缩写，#include 命令都是放在程序的开头，因此这类文件被称为“头文件”。在调用标准输入输出函数时，文件开头应有以下预编译命令：

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

或 `#include "stdio.h"`

stdio.h 是 standard input & output 的缩写，它包含了与标准 I/O 库有关的变量定义和宏定义。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 命令。

3.3 标准输出函数——printf 函数

C 语言中最基本的数据输出函数是 printf()，它是标准输出函数，其作用是在终端设备（或系统隐含指定的输出设备）上，按指定格式进行数据输出。

3.3.1 printf 函数的一般调用形式

printf 函数的一般调用形式如下：

```
printf(<格式控制字符串>,<输出项表>)
```

其中，<格式控制字符串>的作用是将要输出的数据转换为指定的格式输出，包含格式编辑符和原样输出的字符串，格式编辑符用于指定输出格式，其形式是：

```
%[修饰符]格式字符
```

[修饰符]包括：标志、类型修饰、输出最小宽度和精度等，可根据需要取舍。

<输出项表>中的各输出项要用逗号隔开，输出项可以是合法的常量、变量或表达式。格式转换说明的个数要与输出项的个数相同，使用的格式字符也要与它们一一对应且类型匹配。

如果在 printf()函数调用之后加上“;”，就构成了输出语句。

例如：printf("x=%d,y=%f \n",x,y);

其中的%d 和%f 是对应的 x 和 y 的输出格式说明。输出过程是：在当前光标位置处先原样输出“x=”，接下来用“%d”格式输出变量 x 的值，再原样输出字符串“,y=”，然后以“%f”格式输出 y 的值，最后输出转义字符“\n”（换行），使输出位置移到下一行的开头处。当 x=1，y=2.0 时，上述语句的输出结果为：x=1,y=2.000000。

3.3.2　printf 函数中常用的格式控制

不同的数据类型输出时所用的格式也是不同的。表 3-1 中列出了 C 语言中常用的格式字符。表 3-2 中列出了几种修饰符。

表 3-1　printf 格式字符

格 式 字 符	说　　明
d, i	以带符号的十进制形式输出整数（正数不输出符号）
o	以八进制无符号形式输出整数
x，X	以十六进制无符号形式输出整数，若用 x，则输出的十六进制数为小写形式 a~f，若用 X，则输出的十六进制数为大写形式 A~F
u	以无符号的十进制形式输出整数
c	输出一个字符
s	输出字符串
f	以小数形式输出实数，隐含输出 6 位小数
e，E	以指数形式输出实数，数字部分小数位含小数点为 6 位，若用 e，则输出时，指数以小写 e 表示，若用 E，则输出时，指数以大写 E 表示
g，G	由系统决定采用%f 或%e 格式，以使输出宽度最小，无意义的 0 不输出。g 和 G 的区别在于若以指数形式输出时，指数以小写 e 还是大写 E 表示

表 3-2　printf 的修饰符

字　　符	说　　明
字母 l	可加在格式符 d、o、x、u 前面，表示各种长整型
m.n	m、n 都是一个正整数。m 表示数据的输出列宽；对于字符串，*n* 表示截取字符的个数，对于实数，*n* 表示输出的小数位数
-	输出时靠左对齐，右边补空格
+	输出时靠右对齐，输出符号位（正数输出正号，负数输出负号）
#	对于 o 格式输出时加前缀 0；对于 x 格式输出时加前缀 0x；对于 e、g、f 格式，当结果有小数时才给出小数点；对其他格式符无影响

对于修饰符的使用，有以下几点说明。

1．输出宽度

如果直接用%d、%f、%c 等格式输出数据，则都是按照数据实际宽度输出显示，并采用

右对齐方式。也可以根据需要，用十进制整数限定输出数据的位数。例如，printf("x=%5d",24);表示整数 24 以 5 位宽度右对齐输出显示，即输出 x=□□□24。实际数据若超过定义宽度，则按实际位数输出；若少于定义宽度，则补空格或 0。

2．精度

对于 float 或 double 类型的实型数，可以用“m.n”的形式指定数据的输出宽度和小数位数（即精度）。m、n 为正整数，其中 m 指数据总宽度，n 对 e、f 格式符而言，指小数位数。当小数位数大于 n 时，自动四舍五入截去右边多余的小数；当小于指定宽度时，在小数部分最右边自动添 0。若宽度大于 m，整数部分不丢失，小数部分仍按上述规则处理。例如，“printf("x=%8.1f",123.45);”的输出结果为：x=□□□123.5。

3．输出字符“%”

可在格式控制字符串中连用两个%。例如，“printf("x=%4.2f%%",0.5);”的输出结果是：x=0.50%。

【例 3.1】printf 函数应用举例。

```
#include <stdio.h>
void main()
{
   printf("1234567890\n");
   printf("-------------\n");
   printf("%s,%c,%d\n","abcd",'A','A');
   printf("a=%-3d,a=%+3d\n",2+3,5);
   printf("%#o,%#x\n",12,12);
   printf("%6.3s\n","abcde");
   printf("%.3s\n","abcde");
}
```

输出结果是：

```
1234567890
-------------
abcd,A,65
a=5  ,a= +5
014,0xc
    abc
abc
```

3.3.3 调用 printf 函数时的注意事项

在调用 printf 函数进行输出时需要注意：

（1）在格式控制字符串中，格式说明与输出项从左到右在类型上必须一一对应匹配。如不匹配，将导致数据不能正确输出，且这时系统并不报错。特别要注意的是：在输出长整型数据时，一定要使用%ld 格式说明，如果遗漏了字母“l”，只用了%d，将输出错误的数据。

（2）在格式控制字符串中，格式说明与输出项的个数应该相同。如果格式说明的个数少于输出项的个数，多余的输出项不予输出；如果格式说明的个数多于输出项的个数，则对于多余的格式将输出不定值（或 0 值）。

（3）在格式控制字符串中，除了合法的格式说明外，可以包含任意的合法字符（包括转义字符），这些字符在输出时将“原样输出”。

（4）printf 中各输出表达式的求值是从右向左进行的。例如：

```
int i=0;
printf("%d,%d,%d\n",i+=3,i+=2,i+=1);
```

i 先赋值为 0，由于 printf 中各输出表达式的求值是从右向左进行，即先求最后一个表达式 i+=1 的值，i=1，它返回 i 的值为 1；然后求倒数第 2 个表达式 i+=2 的值，i=3，它返回 i 的值为 3；最后求第 1 个表达式 i+=3 的值，i=6，它返回 i 的值为 6。所以输出为：

```
6,3,1
```

3.4　标准输入函数——scanf 函数

scanf 函数是 C 语言提供的标准输入函数，它的作用是在终端设备（或系统隐含指定的输入设备）上输入数据。

3.4.1　scanf 函数的一般调用形式

scanf 函数的一般调用形式如下：

```
scanf (<格式控制字符串>,<输入项表>)
```

格式控制字符串通常只包含格式转换说明符，其含义与 printf()类似。

输入项表中的各输入项用逗号隔开，各输入项只能是合法的地址表达式，也就是说，输入项是某个存储单元的地址。如果不是地址，要使用“&”求地址运算符。

如果在 scanf 函数调用之后加上“;”，就构成了输入语句。例如：

【例 3.2】用 scanf 函数输入数据。

```
#include <stdio.h>
void main()
{
   int t1,t2;
   scanf("%d,%d",&t1,&t2);
   printf("%d,%d",t1,t2);
}
```

运行时应先从键盘输入 t1 和 t2 的值：

```
12,34↙
12,34    (输出 t1 和 t2 的值)
```

上述 scanf 语句“%d,%d”表示按十进制的形式输入数据，而且输入时数据间应使用逗号作为分隔，不能使用空格或其他符号。如果写成“%d%d”的形式则不能使用逗号，而要使用空格（一个或多个）、回车、Tab 键。例如：

```
scanf("%d%d",&t1,&t2);
```

则输入应为：

```
12□34↙
```

或

```
12□□34↙
```

或

12↙
34↙
或
12（按 Tab 键）34
以上均为正确的输入方式。

3.4.2 scanf 函数中常用的格式控制

scanf 函数的格式说明符和 printf 相类似，都是以%开始，以一个格式字符结束，中间可以插入修饰符（见表 3-3、表 3-4）。

表 3-3 scanf()格式转换说明符

格 式 字 符	说　明
d，i	按有符号的十进制整数输入
u	按无符号的十进制整数输入
o	输入无符号的八进制整数
x，X	输入无符号的十六进制整数（不分大小写）
c	输入单个字符
s	输入一个字符串
f，e，E，g，G	输入实数，可以是小数或指数形式

表 3-4 修饰符

字符 l	用于输入长整型数据（可用%ld，%lo，%lx，%lu）以及 double 型数据（用%lf 或%le）
h	用于输入短整型数据（可用%hd，%ho，%hx）
*	表示本输入项在读入后不赋给任何变量
域宽	指定输入数据所占宽度（列数），域宽应为正整数

说明：

（1）对于 unsigned 型变量所需的数据，可以用%u、%d 或%o、%x 格式输入。

（2）可以指定输入数据所占列数，系统自动按它截取所需要的数据。例如：

```
scanf("%3d%3d",&a,&b);
```

输入：123456↙

系统自动将 123 赋给 a，456 赋给 b。

（3）如果在%后有一个“*”修饰符，表示跳过它指定的列数。例如：

```
scanf("%2d□%*3d□%2d",&a,&b);
```

如果输入如下信息：

12□345□67↙

将 12 赋给 a，%*3d 表示读入 3 位整数但不赋给任何变量。然后再读入 2 位整数 67 赋给 b。也就是说第 2 个数据“345”被跳过。在利用现成的一批数据时，有时不需要其中某些数据，可用此方法“跳过”它们。

（4）输入数据时不能规定精度，例如：

```
scanf("%7.2f",&a);
```

是不合法的，不能企图利用这样的 scanf 函数并输入以下数据而使 a 的值为 12345.67。

```
1234567↙
```

3.4.3　调用 scanf 函数时的注意事项

（1）scanf 函数中的“格式控制字符串”后面应当是变量地址，而不应是变量名。例如，如果 a、b 为整型变量，则

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

是不对的，应将“a,b”改为“&a,&b”。这是 C 语言与其他高级语言的不同之处。许多初学者常在此出错。

（2）在“格式控制字符串”中除格式转换说明符以外还有其它字符（如空格、分号、冒号等），则在输入数据时还应输入与这些字符相同的字符，如表 3-5 所示。

表 3-5　scanf 的各种输入形式

输入语句形式	输 入 形 式
scanf("%d,%d",&t1,&t2);	3,4
scanf("%d%d",&t1,&t2);	3□4 或 3□□□4 或 3（Tab 键）4 或 3（回车）4
scanf("%d:%d",&t1,&t2);	3:4
scanf("t1=%d,t2=%d",&t1,&t2);	t1=3,t2=4

表 3-5 所列不可能涵盖全部输入形式，希望读者以此为启发推而广之。

（3）在用%c 格式输入字符时，一次只能接受一个字符，且空格和各种转义字符都作为有效字符输入，例如：

```
scanf("%c%c%c",&c1,&c2,&c3);
```

若输入：

```
a↙
bc↙
```

则 c1='a'，c2='\r'（回车字符），c3='b'，因为%c 只要求读入一个字符，而此处输入了 5 个字符（回车也算一个字符号），故只将前 3 个'a'，'\r'，'b'分别赋给 c1，c2 和 c3。

（4）在输入数据时，遇到以下情况时认为输入数据结束。

① 遇空格，或按回车键或按跳格键（Tab）。

② 按指定的宽度结束，如“%3d”，只取 3 列。

③ 遇到非法输入。

如

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

若输入

```
1234a1230.26↙
```

第一个数据对应%d 格式在输入 1234 之后遇到字母 a，因此认为数值 1234 后已没有数字了，第一个数据到此结束，把 1234 送给变量 a。字符'a'送给变量 b，由于%c 只要求输入一个字符，因此输入字符 a 之后不需要加空格，后面的数值应送给 c。如果由于疏忽把本应为 1230.26 错打成 123o.26，由于后面出现字母'o'，就认为该数值数据到此结束，将 123 送给 c。

3.5 字符输入输出函数

putchar 函数和 getchar 函数都是标准 I/O 库中的函数，被定义在“stdio.h”中。其功能分别是在显示器上输出一个字符和通过键盘输入一个字符。

3.5.1 字符输出函数 putchar

putchar 函数调用的一般形式是：

```
putchar(参数);
```

参数可以是一个字符型变量、整型变量或一个字符型常量（包括控制字符和转义字符）等。

【例 3.3】使用 putchar 输出字符。

```
#include <stdio.h>
void main()
{
   char ch1,ch2;
   int i;
   ch1='C';
   ch2='h';
   i=105;
   putchar(ch1);          /*输出字符 C*/
   putchar(ch2);          /*输出字符 h*/
   putchar(i);            /*以字符形式输出整型变量 i 的值,105 是字符"i"的 ASCII 码*/
   putchar('n');          /*输出字符 n*/
   putchar('\141');       /*输出字符 a,'\141'是转义字符*/
   putchar('\n');         /*输出换行*/
}
```

运行该程序，输出结果是：

```
China
```

注意，若将上例“putchar(ch1); putchar(ch2);”两句合并成“putchar(ch1,ch2);”是错误的。因为 putchar 函数只能带一个参数，即一次只能传送一个字符到屏幕上。

3.5.2 字符输入函数 getchar

和 putchar 相反，getchar()函数的功能是从终端接受一个字符。其一般格式是：

```
getchar()
```

注意，getchar()是一个无参函数，即圆括号中没有参数，但圆括号不能省略。

【例 3.4】输入单个字符。

```
#include <stdio.h>
void main()
{
   char ch;
   ch=getchar();
   putchar(ch);
   putchar(getchar());
}
```

输入 AB 并按回车键，就看到输入为：

```
AB↙
AB
```

在此注意 getchar 函数一次只能接受一个字符，上例中计算机执行到第一个 getchar()语句时就停下等待从键盘输入，在键盘输入“AB”，而没有按回车键之前，“AB”存放在键盘缓冲区中，getchar()得不到数据，程序仍无法向下执行。按回车键之后，第一个 getchar()得到“A”，将“A”赋值给 ch，putchar(ch)将 ch 中“A”输出，接着第二个 getchar()将存放在键盘缓冲区中的“B”读出来，再通过 putchar()将“B”在“A”后面输出，见上面的输出结果。

getchar 函数接受的字符可以赋给一个字符型或整型变量，也可以不赋给任何变量，直接输出。

3.6　顺序结构程序设计举例

【例 3.5】输入一个华氏温度，要求输出摄氏温度。公式为：

```
C=(5/9)(F-32)
```

输入要有文字说明，结果取两位小数。

```
#include <stdio.h>
void main()
{
   float c,F;
   printf("请输入一个华氏温度\n");
   scanf("%f",&F);
   c=(5.0/9)*(F-32);
   printf("华氏温度%f 转换为摄氏温度后的结果为: %.2f\n",F,c);
}
```

一次的运行情况如下：

```
请输入一个华氏温度
67↙
华氏温度 67.000000 转换为摄氏温度后的结果为: 19.44
```

【例 3.6】从键盘输入一个大写字母，要求改用小写字母输出。

```
#include <stdio.h>
void main()
{
   char c1,c2;
   c1=getchar();
   printf("%c,%d\n",c1,c1);
   c2=c1+32;
   printf("%c,%d",c2,c2);
}
```

一次的运行情况如下：

```
A↙
A,65
a,97
```

【例 3.7】将键盘输入的两个整数赋给变量 a 和 b，输出 a、b 的值以及它们交换后的值。下面是用两种不同的方法编写的程序。

（1）借助第三个变量实现将两个变量对调的操作。

```
#include <stdio.h>
void main()
{
   int a,b,c;
   printf("Input a and b:");
   scanf("%d%d",&a,&b);
   printf("old a=%d b=%d\n",a,b);
   c=a;
   a=b;
   b=c;
   printf("new a=%d b=%d\n",a,b);
}
```

一次的运行情况如下：

```
Input a and b:5 8↙
old a=5 b=8
new a=8 b=5
```

在程序中，交换 a、b 两个变量的值，不能简单地用“a=b;b=a;”两条语句实现。因为，执行 a=b，是把 b 的值赋给了 a，a 原来的值已丢失，接下去执行 b=a，又把新的 a 的值（即原来 b 的值）赋给了 b，结果是 a 和 b 的值都是原来 b 的值。因此，借用第三个变量 c，先将 a 的值转移到 c 中，再将 b 的值给 a，最后把 c 的值（原来 a 的值）给 b。

（2）不用第三个变量实现的交换。

```
#include <stdio.h>
void main()
{
   int a,b;
   printf("Input a and b:");
   scanf("%d%d",&a,&b);
   printf("old a=%d b=%d\n",a,b);
   a=a+b;
   b=a-b;
   a=a-b;
   printf("new a=%d b=%d\n",a,b);
}
```

一次的运行情况如下：

```
Input a and b:5 8↙
old a=5 b=8
new a=8 b=5
```

在 a、b 交换的过程中，变量单元中值的变化情况如表 3-6 所示。

表 3-6　a、b 交换过程中变量值变化情况

	A 的 值	B 的 值
输入值	5	8
a=a+b	5+8	8
b=a−b	5+8	5+8−8
a=a−b	5+8−5	5

【例 3.8】运用头文件“math.h”求 sin(x)

```
#include<stdio.h>
#include<math.h>
#define Pi 3.1415926
void main()
{
   float x;
   printf("请输入 x 的值: \n");
   scanf("%f",&x);
   printf("sin(%f 弧度)=%f\n",x,sin(x));
   printf("sin(%f 度)=%f\n",x,sin(x*Pi/180));
}
```

运行该程序：

```
请输入 x 的值:
30↙
sin(30.000000 弧度)=-0.988032
sin(30.000000 度)=0.500000
```

习　题

一、选择题

1. 用 getchar 函数可以从键盘读入一个_____。
   A．整型变量表达式值　　B．实型变量值
   C．字符串　　D．字符或字符型变量值
2. scanf 函数被称为_____输入函数。
   A．字符　　B．整数　　C．格式　　D．浮点
3. scanf 函数按_____串规定的格式输入数据。
   A．格式控制　　B．特殊　　C．具体安排　　D．功能
4. scanf 函数是一个_____函数。
   A．用户定义的　　B．标准库　　C．字符　　D．地址
5. 在 scanf 函数语句中，地址表列由_____组成。
   A．表达式　　B．变量　　C．常量　　D．地址项
6. 设有以下语句

   ```
   Char ch1,ch2, scanf（”%c%c”,&ch1,&ch2）;
   ```

   若要为变量 ch1 和 ch2 分别输入字符 A 和 B，正确的输入形式应该是
   A、A 和 B 之间用逗号间隔　　B、A 和 B 之间不能有任何间隔符
   C、A 和 B 之间可以用回车间隔　　D、A 和 B 之间用空格间隔

二、简答题

1. C 语言中的语句有哪几类？

2. 怎样区分表达式和表达式语句？
3. C 语言为什么要把输入输出的功能作为函数，而不作为语言的基本部分？
4. 若有程序如下:

```
#include <stdio.h>
void main()
{
   int i,j;
   scanf("i=%d,j=%d",&i,&j);
   printf("i=%d,j=%d\n",i,j);
}
```

要求给 i 赋 10，给 j 赋 20，则应该从键盘输入什么？

三、读程题

写出下面程序的输出结果。

1.
```
#include <stdio.h>
void main()
{
   int a=5,b=6;
   float x=67.8564,y=-789.124;
   char c='A';
   long n=1234567;
   unsigned u=65535;
   printf("%d%d\n",a,b);
   printf("%3d%3d\n",a,b);
   printf("%f,%f\n",x,y);
   printf("%-10f,%-10f\n",x,y);
   printf("%8.2f,%8.2f,%.4f,%.4f,%3f,%3f\n",x,y,x,y,x,y);
   printf("%e,%10.2e\n",x,y);
   printf("%c,%d,%o,%x\n",c,c,c,c);
   printf("%ld,%lo,%x\n",n,n,n);
   printf("%u,%o,%x,%d\n",u,u,u,u);
   printf("%s,%5.3s\n","COMPUTER","COMPUTER");
}
```

2.
```
#include <stdio.h>
void main()
{
   int x=10,y=10;
   printf("%d,%d\n",x--,--y);
}
```

3.
```
#include <stdio.h>
void main()
{
   int i=8;
   printf("%d,%d,%d,%d\n",++i,--i,i++,i--);
}
```

4.
```
#include <stdio.h>
```

```
void main()
{
   char c1='6',c2='0';
   printf("%c,%c,%d\n",c1,c2,c1-c2);
}
```

5.
```
#include <stdio.h>
void main()
{
   int n=65535;
   n++;
   printf("%d\n",n);
}
```

四、编程题

1. 编写一个程序，输入弧度值，将弧度换算成角度值（度、分、秒的形式）输出。
2. 编写一个程序，用 getchar()分别输入两个字符给 c1、c2，然后输出这两个字符以及它们所对应的 ASCII 码。
3. 输入一个小于 10000 的正整数，输出该整数各位上的数字各是多少？
4. 输入一个 3 位正整数，然后反向输出对应的数。如输入“123”，则输出“321”。
5. 把十进制数“97”转化为八进制数、十六进制数。
6. 用 getchar 函数读入两个字符 c1、c2，然后分别用 putchar 函数和 printf 函数输出。
7. 编写程序，打印下列图形。

```
          A
        B   C
      D       E
    F           G
H   I   J   K   L   M   N   O   P
```