第3章　最简单的C程序设计——顺序程序设计

1. 假如我国国民生产总值的年增长率为7%，计算10年后我国国民生产总值与现在相比增长多少倍。计算公式为

$$p=(1+r)^n$$

r 为年增长率，n 为年数，p 为与现在相比的倍数。

解：从主教材附录E(库函数)可以查到：可以用pow函数求 y^x 的值，调用pow函数的具体形式是 pow(x,y)。在使用 pow 函数时，需要在程序的开头用 #include 指令将<math.h>头文件包含到本程序模块中。可以用下面的程序求出10年后国民生产总值是现在的多少倍。

```
#include <stdio.h>
#include <math.h>
int main( )
{   float p,r,n;
    r=0.07;
    n=10;
    p=pow(1+r,n);
    printf("p=%f\n",p);
    return 0;
}
```

运行结果：

```
p=1.967151
```

即10年后国民生产总值是现在的1.967151倍。

2. 存款利息的计算。有1000元，想存5年，可按以下5种办法存：

(1) 一次存5年期。

(2) 先存2年期，到期后将本息再存3年期。

(3) 先存3年期，到期后将本息再存2年期。

(4) 存1年期，到期后将本息再存1年期，连续存5次。

(5) 存活期存款。活期利息每一季度结算一次。

2017年的银行存款利息如下：

1年期定期存款利息为1.5%；

2年期定期存款利息为2.1%；

3年期定期存款利息为2.75%；

5年期定期存款利息为3%；

活期存款利息为 0.35%(活期存款每一季度结算一次利息)。

如果 r 为年利率,n 为存款年数,则计算本息和的公式为

1 年期本息和：$p=1000\times(1+r)$；

n 年期本息和：$p=1000\times(1+n\times r)$；

存 n 次 1 年期的本息和：$p=1000\times(1+r)^n$；

活期存款本息和：$p=1000\times\left(1+\frac{r}{4}\right)^{4n}$。

说明：$1000\times\left(1+\frac{r}{4}\right)$是一个季度结算后的本息和。

解：设 5 年期存款的年利率为 r5,3 年期存款的年利率为 r3，2 年期存款的年利率为 r2，1 年期存款的年利率为 r1，活期存款的年利率为 r0。

设按第 1 种方案存款 5 年得到的本息和为 p1,按第 2 种方案存款 5 年得到的本息和为 p2,按第 3 种方案存款 5 年得到的本息和为 p3,按第 4 种方案存款 5 年得到的本息和为 p4,按第 5 种方案存款 5 年得到的本息和为 p5。

程序如下：

```
#include <stdio.h>
#include <math.h>
int main( )
{   float r5,r3,r2,r1,r0,p,p1,p2,p3,p4,p5;
    p=1000;
    r5=0.03;
    r3=0.0275;
    r2=0.021;
    r1=0.015;
    r0=0.0035;

    p1=p*(1+r5*5);                  //一次存 5 年期
    p2=p*(1+2*r2)*(1+3*r3);         //先存 2 年期,到期后将本息再存 3 年期
    p3=p*(1+3*r3)*(1+2*r2);         //先存 3 年期,到期后将本息再存 2 年期
    p4=p*pow(1+r1,5);               //存 1 年期,到期后将本息再存 1 年期,连续存 5 次
    p5=p*pow(1+r0/4,4*5);           //存活期存款,活期利息每一季度结算一次
    printf("p1=%f\n",p1);           //输出按第 1 种方案得到的本息和
    printf("p2=%f\n",p2);           //输出按第 2 种方案得到的本息和
    printf("p3=%f\n",p3);           //输出按第 3 种方案得到的本息和
    printf("p4=%f\n",p4);           //输出按第 4 种方案得到的本息和
    printf("p5=%f\n",p5);           //输出按第 5 种方案得到的本息和
    return 0;
}
```

运行结果：

```
p1=1150.000000
p2=1127.964966
p3=1127.964966
p4=1077.284058
p5=1017.646240
```

讨论：

(1) 在使用 GCC(包括本书附赠的简易 C 语言编译环境)时，有可能遇到“/usr/bin/ld:/usr/bin/ld:ex_3_2.c:(.text+0x111):undefined reference to 'pow' collect2:error:ld returned 1 exit status”，这说明你的程序编译通过，但是在连接的时候，GCC 没有找到 pow 函数的具体实现(目标文件)，这时可以通过在 GCC 命令行增加“-lm”参数来告诉 GCC 要与 math 库连接(在简易 C 语言编译环境中，在“编译器标志”文本框添加“ -lm”)。

(2) 如果不想出现上面的警告，可以将第 4 行各变量改为 double 型，即

```
double r5,r3,r2,r1,r0,p,p1,p2,p3,p4,p5;
```

由于采用了双精度变量，得到的运算结果会更精确些，最后几位数字与上面的有些差别。

```
p1=1150.000000
p2=1127.965000
p3=1127.965000
p4=1077.284004
p5=1017.646235
```

(3) 输出运行结果时，得到 6 位小数，连同整数部分有 10 位数字，而一个 float 型变量只能保证 6 位有效数字，后面几位是无意义的。而且在输出款额时，人们一般只要求精确到两位小数(角、分)，因此可以在 printf 函数中用%10.2 格式符输出。最后 5 个语句可改为

```
printf("p1=%10.2f\n",p1);          //输出按第 1 种方案得到的本息和
printf("p2=%10.2f\n",p2);          //输出按第 2 种方案得到的本息和
printf("p3=%10.2f\n",p3);          //输出按第 3 种方案得到的本息和
printf("p4=%10.2f\n",p4);          //输出按第 4 种方案得到的本息和
printf("p5=%10.2f\n",p5);          //输出按第 5 种方案得到的本息和
```

这时的输出结果如下：

```
p1=1150.00
p2=1127.96
p3=1127.96
p4=1077.28
p5=1017.65
```

3. 购房从银行贷了一笔款 d，准备每月还款额为 p，月利率为 r，计算多少月能还清。设 d 为 300 000 元，p 为 6000 元，r 为 1%。对求得的月份取小数点后一位，对第 2 位小数按四舍五入处理。

提示：计算还清月数 m 的公式如下：

$$m=\frac{\lg p-\lg(p-d\times r)}{\lg(1+r)}$$

可以将公式改写为

$$m=\frac{\lg\dfrac{p}{p-d\times r}}{\lg(1+r)}$$

C 的库函数中有求对数的函数 log10，是求以 10 为底的对数。

解：根据以上公式可以很容易写出以下程序。

```
#include <stdio.h>
#include <math.h>
int main( )
{   float d=300000,p=6000,r=0.01,m;
    m=log10(p/(p-d*r))/log10(1+r);
    printf("m=%6.1f\n",m);
    return 0;
}
```

运行结果：

```
m=  69.7
```

即需要 69.7 个月才能还清。为了验证对第 2 位小数是否已按四舍五入处理，可以将程序第 6 行中的“%6.1f”改为“%6.2f”。此时的输出为

```
m= 69.66
```

可知前面的输出结果是对第 2 位小数按四舍五入处理的。

4. 分析下面的程序：

```
#include <stdio.h>
int main( )
{   char  c1,c2;
    c1=97;
    c2=98;
    printf("c1=%c,c2=%c\n",c1,c2);
    printf("c1=%d,c2=%d\n",c1,c2);
    return 0;
}
```

(1) 运行时会输出什么信息？为什么？

解：**运行时输出**

```
c1=a,c2=b
c1=97, c2=98
```

第 1 行是将 c1，c2 按%c 的格式输出，97 是字符 a 的 ASCII 码，98 是字符 b 的 ASCII 码。

第 2 行是将 c1，c2 按%d 的格式输出，所以输出两个十进制整数。

(2) 如果将程序第 4，5 行改为

```
c1=197;
c2=198;
```

运行时会输出什么信息？为什么？

解：多数 C 语言编译器把字符型数据作为 signed char 类型处理，它的有效范围为－128～127。但根据系统语言设置的不同，用%c 格式输出时，超出 0～127 区间的字符输

出结果是不可预料的。

用%d 格式输出时,输出 c1=-59,c2=-58。这是按补码形式输出的,内存字节中第 1 位为 1 时,作为负数。59 与 197 之和等于 256,58 与 198 之和也等于 256。对此可暂不深究。

只要知道:用 char 类型变量时,给它赋的值应在 0~127 的范围内。

(3) 如果将程序第 3 行改为

```
int c1,c2;
```

运行时会输出什么信息?为什么?

解:如果给 c1 和 c2 赋的值是 97 和 98,则输出结果与(1)相同。

如果给 c1 和 c2 赋的值是 197 和 198,则用%c 输出时是不可预料的字符。用%d 输出时,输出整数 197 和 198,因为它们在 int 类型的有效范围内。

5. 用下面的 scanf 函数输入数据,使 a=3,b=7,x=8.5,y=71.82,c1='A',c2='a'。问在键盘上如何输入。

```
#include <stdio.h>
int main( )
{   int a,b;
    float x,y;
    char c1,c2;
    scanf("a=%d b=%d",&a,&b);
    scanf("%f %e",&x,&y);
    scanf("%c%c",&c1,&c2);
    printf("a=%d,b=%d,x=%f,y=%f,c1=%c,c2=%c\n",a,b,x,y,c1,c2);
    return 0;
}
```

解:按如下方式在键盘上输入(见下面第 1,2 两行)。

```
a=3 b=7
8.5 71.82Aa
a=3,b=7,x=8.500000,y=71.820000,c1=A,c2=a
```

第 3 行是输出的结果。

注意:在输入 8.5 和 71.82 两个实数给 x 和 y 后,应紧接着输入字符 A,中间不要有空格,由于 A 是字母而不是数字,系统在遇到字母 A 时就确定输入给 y 的数值已结束。字符 A 就送到下一个 scanf 语句中的字符变量 c1。如果在输入 8.5 和 71.82 两个实数后输入空格符,会怎么样呢?情况如下:

```
a=3 b=7
8.5 71.82 Aa
a=3,b=7,x=8.500000,y=71.820000,c1= ,c2=A
```

这时 71.82 后面输入的空格字符就被 c1 读入,c2 读入了字符 A。在输出 c1 时就输出空格,输出 c2 的值为 A。

如果在输入 8.5 和 71.82 两个实数后按回车键,会怎么样呢?情况如下:

```
a=3 b=7
8.5 71.82
Aa
a=3,b=7,x=8.500000,y=71.820000,c1=
,c2=A
```

上面 3 行是输入，在输入 71.82 后按回车键。在这时“回车”被作为一个字符送到内存输入缓冲区，被 c1 读入（实际上 c1 读入的是回车符的 ASCII 码），字符 A 被 c2 读取，所以在执行 printf 函数输出 c1 时，就输出一个换行，在下一行输出逗号和 c2 的值 A。

在用 scanf 函数输入数据时往往会出现一些意想不到的情况，例如在连续输入不同类型的数据（特别是数值型数据和字符数据连续输入）的情况。要注意回车符是可能被作为一个字符读入的。

通过此例，可以了解怎样正确输入数据。这些知识不能靠枯燥地死记规则，必须善于在实践中注意分析现象，不断总结经验。

6. 请编程序将 China 译成密码，密码规律是：用原来的字母后面第 4 个字母代替原来的字母。例如，字母 A 后面第 4 个字母是 E，用 E 代替 A。因此，China 应译为 Glmre。请编一程序，用赋初值的方法使 c1，c2，c3，c4，c5 这 5 个变量的值分别为'C'，'h'，'i'，'n'，'a'，经过运算，使 c1，c2，c3，c4，c5 分别变为'G'，'l'，'m'，'r'，'e'。分别用 putchar 函数和 printf 函数输出这 5 个字符。

解：

```
#include <stdio.h>
int main( )
{   char c1='C',c2='h',c3='i',c4='n',c5='a';
    c1=c1+4;
    c2=c2+4;
    c3=c3+4;
    c4=c4+4;
    c5=c5+4;
    printf("password is %c%c%c%c%c\n",c1,c2,c3,c4,c5);
    return 0;
}
```

运行结果：

```
password is Glmre
```

7. 设圆半径 $r=1.5$，圆柱高 $h=3$，求圆周长、圆面积、圆球表面积、圆球体积、圆柱体积。用 scanf 输入数据，输出计算结果，输出时要求有文字说明，取小数点后 2 位数字。请编程序。

解：

```
#include <stdio.h>
int main( )
{   float h,r,l,s,sq,vq,vz;
    float pi=3.141526;
    printf("请输入圆半径 r,圆柱高 h:");
```

```
    scanf("%f,%f",&r,&h);                    //要求输入圆半径 r 和圆柱高 h
    l=2*pi*r;                                //计算圆周长 l
    s=r*r*pi;                                //计算圆面积 s
    sq=4*pi*r*r;                             //计算圆球表面积 sq
    vq=4.0/3.0*pi*r*r*r;                     //计算圆球体积 vq
    vz=pi*r*r*h;                             //计算圆柱体积 vz
    printf("圆周长为:        l=%6.2f\n",l);
    printf("圆面积为:        s=%6.2f\n",s);
    printf("圆球表面积为:    sq=%6.2f\n",sq);
    printf("圆球体积为:      vq=%6.2f\n",vq);
    printf("圆柱体积为:      vz=%6.2f\n",vz);
    return 0;
}
```

运行结果:

```
请输入圆半径r, 圆柱高h:1.5,3
圆周长为:       l=  9.42
圆面积为:       s=  7.07
圆球表面积为:   sq= 28.27
圆球体积为:     vq= 7.95
圆柱体积为:     vz= 21.21
```

说明:如果用中文操作系统下运行的 C 语言编译器(如本书附赠的简易 C 语言开发环境)对程序进行编译,在程序中可以使用中文字符串,在输出时也能显示汉字。如果用英文的 C 编译系统,则无法使用中文字符串,读者可以改用英文字符串。

8. 编写程序,用 getchar 函数读入两个字符给 c1 和 c2,然后分别用 putchar 函数和 printf 函数输出这两个字符。思考以下问题:

(1) 变量 c1 和 c2 应定义为字符型、整型还是二者皆可?

(2) 要求输出 c1 和 c2 值的 ASCII 码,应如何处理?用 putchar 函数还是 printf 函数?

(3) 整型变量与字符变量是否在任何情况下都可以互相代替?如:

```
char c1,c2;
```

与

```
int c1,c2;
```

是否无条件地等价?

解:

```
#include <stdio.h>
int main( )
{
    char c1,c2;
    printf("请输入两个字符 c1,c2:");
    c1=getchar( );
    c2=getchar( );
    printf("用 putchar 语句输出结果为:");
```

```
    putchar(c1);
    putchar(c2);
    printf("\n");
    printf("用 printf 语句输出结果为:");
    printf("%c %c\n",c1,c2);
    return 0;
}
```

运行结果：

```
请输入两个字符c1,c2:ab
用putchar语句输出结果为:ab
用printf语句输出结果为:a b
```

注意：若连续用两个 getchar 函数，输入字符时 a 和 b 之间没有空格，连续输入。

如果分两行输入：

a↙
b↙

结果会怎样？

运行结果：

```
请输入两个字符c1,c2:a
用putchar语句输出结果为:a

用printf语句输出结果为:a
```

第 1 行是输入数据，输入 a 后按回车键。结果还未来得及输入 b，程序马上输出了其下 4 行结果(包括两个空行)。

因为第 1 行将 a 和换行符输入内存的输入缓冲区，因此 c1 得到 a(ASCII 码为 97)，c2 得到换行符(ASCII 码为 10)。再用 putchar 函数输出 c1，就输出了字符 a，在输出 c2 时，就把换行符转换为回车和换行两个操作，输出一个换行，后面的 printf("\n")又输出一个换行，所以就相当于输出一个空行，此行不显示任何字符。后面用 printf 函数输出 c1 和 c2，同样也输出了字符 a 和一个空行。

注意：在用连续两个 getchar 输入两个字符时，只要输入了“a↙”，系统就会认为用户已输入了两个字符。所以应当连续输入 ab 两个字符，然后再按回车键，这样就保证了 c1 和 c2 分别得到字符 a 和 b。

下面回答思考问题：

(1) c1 和 c2 可以定义为字符型或整型，二者皆可。

(2) 可以用 printf 函数输出，在 printf 函数中用%d 格式符，即

```
printf("%d,%d\n",c1,c2);
```

(3) 字符变量在计算机内占 1 字节，而整型变量占 2 字节或 4 字节。因此整型变量在可输出字符的范围内(ASCII 码为 0～127 的字符)是可以与字符数据互相转换的。如果整数在此范围外，则不能代替。

为了进一步说明 char 型与 int 型数据的关系，请注意分析以下 3 个程序。

程序 1：

```
#include <stdio.h>
int main( )
{
    int c1,c2;                    //定义整型变量 c1,c2
    printf("请输入两个整数 c1,c2:");
    scanf("%d,%d",&c1,&c2);
    printf("按字符输出结果:\n");
    printf("%c,%c\n",c1,c2);
    printf("按 ASCII 码输出结果为:\n");
    printf("%d,%d\n",c1,c2);
    return 0;
}
```

运行结果：

```
请输入两个整数c1,c2:97,98
按字符输出结果:
a,b
按ASCII码输出结果为:
97,98
```

程序 2：

```
#include <stdio.h>
int main( )
{
    char c1,c2;                   //c1,c2 定义为字符型变量
    int i1,i2;                    //定义整型变量
    printf("请输入两个字符 c1,c2:");
    scanf("%c,%c",&c1,&c2);
    i1=c1;                        //赋值给整型变量
    i2=c2;
    printf("按字符输出结果:\n");
    printf("%c,%c\n",i1,i2);
    printf("按整数输出结果:\n");
    printf("%d,%d\n",c1,c2);
    return 0;
}
```

运行结果：

```
请输入两个字符c1,c2:a,b
按字符输出结果:
a,b
按整数输出结果:
97,98
```

程序 3：

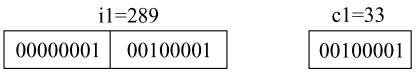
```
#include <stdio.h>
```

```
int main( )
{
    char c1,c2;                    //c1,c2 定义为字符型
    int i1,i2;                     //i1,i2 定义为整型
    printf("请输入两个整数 i1,i2:");
    scanf("%d,%d",&i1,&i2);
    c1=i1;                         //将整数赋值给字符变量
    c2=i2;
    printf("按字符输出结果:\n");
    printf("%c,%c\n",c1,c2);
    printf("按整数输出结果:\n");
    printf("%d,%d\n",c1,c2);
    return 0;
}
```

运行结果：

```
请输入两个整数i1,i2:289,330
按字符输出结果:
!,J
按整数输出结果:
33,74
```

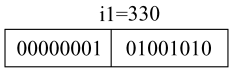
请注意 i,i1 和 i2 定义为整型，根据 C 语言标准，它们至少占 2 字节或 4 字节（GCC 和 MSVC 对它们分配 4 字节），而 c1 和 c2 是字符变量，只占 1 字节。如果是 unsigned char 类型，可以存放 0～255 的整数；如果是 signed char 类型，可以存放－128～127 的整数。而现在输入给 i1 和 i2 的值已超过 0～255 的范围，i1 的值为 289，在内存中 i1 的存储情况如图 3.1(a) 所示（为简单起见，用 2 字节表示），在赋给字符变量 c1 时，只将其存储单元中最后 1 字节（低 8 位）赋给 c1，见图 3.1(b)。而图 3.1(b)中的数据是整数 33，是字符'!'的 ASCII 码，所以用字符形式输出 c1 时，会输出字符'!'。图 3.2 表示 i2 和 c2 的情况，c2 的值为 74，是字符'J'的 ASCII 码，因此，按字符形式输出 c2 时就输出字符'J'。



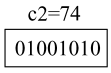
(a)

(b)

图　3.1



(a)



(b)

图　3.2