第 3 章 结构化程序设计

3.1 结构化程序设计概述

程序设计方法是影响程序设计成败以及程序设计质量的重要因素之一。目前，程序设计的方法有两大类，一类是面向过程的结构化程序设计方法；另一类是面向对象的程序设计方法。这里主要介绍结构化程序设计方法，其是进行各类程序设计的基础，有助于程序设计思想的形成和理解。

结构化程序设计方法是基于模块化、自顶向下、逐步求精和结构化程序设计等程序设计技术而发展起来的。用这种方法设计的程序称为结构化程序，其强调程序设计风格和程序结构的规范化，提倡清晰的结构。

早期的程序设计是非结构化的，所编写的程序中含有大量的 goto 语句，可以方便地从程序的一个地方直接跳到另一个地方。这样做的好处是程序设计十分方便灵活，但其缺点也十分突出，就是程序的流程非常混乱，不便于对程序的阅读和理解，也不便于程序中错误的排除，更不便于程序的维护和扩展。

经过研究人们发现，解决任何复杂问题的方法（即算法）都可以由顺序结构、选择（分支）结构和循环结构这 3 种基本结构组成，这些基本结构之间可以并列、可以相互包含（嵌套），但不允许交叉，也不允许从一个结构直接跳转到另一个结构的内部去。这样只用 3 种基本结构所设计的算法结构清晰，易于阅读和理解，也易于纠错，这种设计方法就是结构化方法。

1965 年，荷兰学者 E.W.Dijkstraz 在一次会议上指出“可以从高级语言中取消 goto 语句”,“程序的质量与程序中所包含的 goto 语句的数量成反比”。1966 年，Boehm 和 Jacopini 证明“只用三种基本的控制结构就能实现任何单入口、单出口的程序”。Boehm 和 Jacopini 的证明为结构化程序设计技术奠定了理论基础。经过多年的实践，结构化程序设计的理论和方法日益完善，并已被广泛接受和使用，也总结出了在总体设计、详细设计和编码阶段应该遵循的一些原则。

（1）在总体设计阶段采用“自顶向下，逐步求精”的模块化设计方法，该方法把一个复杂的待求解问题根据总需求划分成若干个相对独立的模块，每个模块完成一个特定的功能。一般来说，一个模块在 C 语言程序设计时对应一个函数，该函数中包含的语句行数大概在 50 行左右。如果一个模块的规模太大（即对应的函数中的语句行数远大于 50），可根据该模块应完成的功能再细分成若干个子模块，而每个子模块又可根据其应完成的功能细分成若干个更小的子模块。如此“自顶向下，逐步求精”，直到每个模块都足够小，不必对模块再细分为止，如图 3.1 所示。

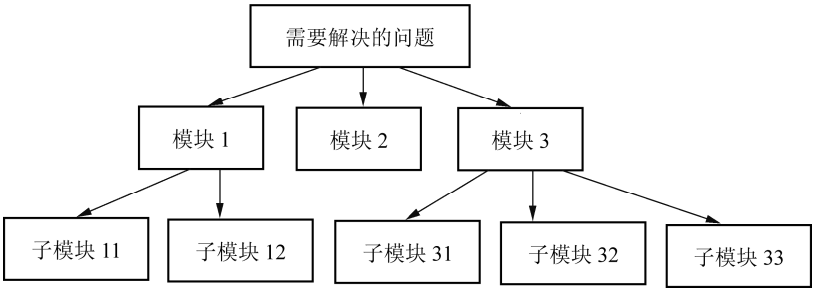

图 3.1　模块化设计方法

（2）在详细设计阶段采用“基本结构，组合而成”的方法，就是程序不论大小、简单还是复杂，程序的结构由 3 种基本结构（即顺序结构、选择结构和循环结构）组合而成，程序各个部分之间做到“一个入口，一个出口”，没有随意地跳转。这样的程序结构清晰，易于发现错误。

（3）在最后的编码阶段应做到“清晰第一，效率第二”，并采用良好的程序设计风格，从而提高程序的可读性，便于调试时改正错误，也便于程序的维护。也就是说，在编写代码具体实现某个功能时，首先应考虑采用正确且清晰的方法，然后再考虑实现的效率问题。应主动放弃那些看似效率高，但日后或者其他人很不容易理解的方法。所谓良好的程序设计风格，就是应做到“书写规范，缩进格式”。写代码时不要把语句写成“左对齐”，而是按代码的结构层次向右缩进，形成锯齿形的代码格式。这样的程序清晰易读，纠错容易。

3.2　顺序结构程序设计

3.2.1　C 语言语句概述

C 程序中每个函数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 语言中语句可分为以下 5 类：表达式语句、函数调用语句、控制语句、复合语句和空语句。

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 */
```

注意：表达式能构成语句是 C 语言的重要特色，因此有人称 C 语言是“表达式语言”。

赋值语句是由赋值表达式再加上分号构成的表达式语句，其功能和特点都与赋值表达式相同，是程序中用得最多的语句之一。但要注意赋值语句与赋值表达式的区别，赋值表达式是一种表达式，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。例如，语句“if ((x=y+5)>0) z=x;”是合法的，而语句“if ((x=y+5;)>0) z=x;”是非法的。

2．函数调用语句

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);
```

执行函数调用语句就是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，求取函数的返回值。

例如：

```
printf("C Program");  /* 调用库函数实现输出字符串，但不保留函数的返回值 */
```

注意：函数调用语句本质上也是一种表达式语句。

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，由特定的语句定义符构成。C 语言有 9 种控制语句，可分成以下 3 类。

- 条件判断语句：if 语句、switch 语句。
- 循环执行语句：while 语句、do while 语句、for 语句。
- 转向语句：break 语句、continue 语句、goto 语句、return 语句。

4．复合语句

把多个语句用大括号{}括起来组成的一个语句称为复合语句。在程序中应把复合语句看成是一条语句，而不是多条语句。复合语句用在语法上只能有一条语句，但逻辑上需要多条语句的场合。

例如：

```
if (x>y) { t=x;  x=y;  y=t; }
```

这里用到了复合语句来实现当 $x>y$ 时，交换两变量 x 和 y 中的值。由于实现交换需要 3 条语句，而 C 语言又规定 if 语句当条件成立时只能做一条语句，这时就可用复合语句。

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，空语句是什么也不做的语句。在程序中空语句用在语法上需要有一条语句，但逻辑上又没有什么要做的场合。

例如：

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

本语句的循环体为空语句，其功能是只要从键盘输入的字符不是回车符则继续输入。

3.2.2 常用的输入和输出函数

所谓输入输出是以计算机主机为主体而言的，从计算机向输出设备（如显示器、打印机、磁盘等）输出数据称为输出，从计算机输入设备（如键盘、光盘、磁盘等）输入数据称为输入。C 语言本身没有输入输出语句，输入输出是靠库函数来实现的，C 语言不提供输入输出语句的原因是，编译系统简单、高效、通用性强、可移植性好。在 C 语言中使用输入输出库函数（包括 printf()、scanf()等），要用#include <stdio.h>命令。

1．printf()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，把指定的数据显示在显示器上。在前面的例题中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，函数原型在头文件 stdio.h 中，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", 输出项列表)
```

其中格式控制字符串用于指定输出格式。该字符串中的字符有以下两种：

（1）普通字符：包括可打印的西文字符、汉字和转义字符。可打印的西文字符和汉字按原样显示在显示器上，起到提示的作用。转义字符往往是一些控制字符，控制产生特殊的输出效果。

（2）格式说明项：由%与格式字符组成，其作用是将数据按指定的格式输出，不同类型的数据有不同的格式字符，后面将专门讨论。

注意：格式控制字符串中的格式说明项与输出项（输出项可以是表达式）在数量和类型上应该一一对应。若格式控制字符串中没有格式说明项，则输出项也就不再需要。例如:

```
int a=3,b=8;
printf("a=%d  b=%d\n", a,b);
```

2）格式说明项

格式说明项的一般形式为：

```
[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。各项的意义如下。

（1）类型：表示输出项数据的类型，其类型格式字符和意义如表 3.1 所示。

表 3.1　printf()函数中的类型格式字符

类型格式符	意　　义
d	以十进制形式输出带符号整数（正数不输出符号+）
u	以十进制形式输出无符号整数
o	以八进制形式输出无符号整数（不输出前缀 0）

续表

类型格式符	意　义
x 或 X	以十六进制形式输出无符号整数（不输出前缀 0x）
c	输出一个字符
s	输出字符串
f	以小数形式输出单、双精度实数，默认输出 6 位小数
e 或 E	以规格化指数形式输出单、双精度实数
g 或 G	以%f%e 中较短的输出宽度输出单、双精度实数

（2）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。

（3）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；如果输出的是字符串，则表示输出的字符个数；若字符串实际字符数大于所定义的精度数，则截去超过的部分。

（4）长度：长度格式符为 l，可加在 d、o、x、u 前，表示按长整型量输出。

（5）标志：标志格式符用于控制输出项输出时的具体格式，最常用的标志格式字符和意义如表 3.2 所示。

表 3.2　printf()函数中最常用的标志格式字符

标志格式符	意　义
-	数据宽度小于最小宽度时，结果左对齐，右边填空格
0	数据宽度小于最小宽度时，在数据的前面以“0”填充

例 3.1　printf()函数中的格式控制字符示例。

```
#include <stdio.h>
int main()
{   int a=5683;
    long b=5944568;
    float f=48.357621;
    double d=35648256.3645287;
    char c='k';

    printf("a=%d,%6d,%3d\n", a,a,a);          /* %3d实际输出4位 */
    printf("b=%ld,%lo,%lx,%lX\n", b,b,b,b);   /* %lX输出大写的ABF */
    printf("today=%4d-%02d-%02d\n", 2012,3,18);
    printf("f=%f,%8.3f,%-8.3f,%e\n", f,f,f,f);/* %8.3f输出时进行了四舍五入 */
    printf("d=%f,%8.4f,%8.10f,%E\n", d,d,d,d);
                                          /* %8.10f输出的最后3位小数无意义 */
    printf("c=%c,%4c,string=%6.3s\n", c,c,"program");
    return 0;
}
```

运行结果：

```
a=5683,  5683,5683
b=5944568,26532370,5ab4f8,5AB4F8
today=2012-03-18
f=48.357620,  48.358,48.358  ,4.835762e+001
d=35648256.364529,35648256.3645,35648256.3645287010,3.564826E+007
c=k,   k,string=   pro
```

3）注意事项

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。不同的编译系统顺序不一定相同，可以从左到右，也可从右到左。Dev-C++ 5.5.3 是按从右到左进行的。例如：

```
int i=8;
printf("%d  %d\n", i,i--);    /* 输出7  8，不是8  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。

2．scanf()函数

scanf()函数称为格式输入函数，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，从键盘上输入数据放到对应的变量中。

1）scanf()函数调用的一般形式

scanf()函数是一个标准库函数，函数原型在头文件 stdio.h 中，在使用 scanf()函数之前必须包含 stdio.h 文件。scanf()函数调用的一般形式为：

```
scanf("格式控制字符串", 地址列表)
```

其中格式控制字符串用于指定输入格式。该字符串中的字符也有普通字符和格式说明项两种，但不能显示普通字符，也就是不能显示提示字符串。地址列表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。scanf()函数在本质上是给变量赋值，但要求写变量的地址。

注意：格式控制字符串中的格式说明项与变量在数量和类型上应该一一对应。

例如：

```
int a,b,c;
printf("Input a、b、c:");
scanf("%d%d%d", &a,&b,&c);
```

输入为：

```
7  8  9↙    （↙表示回车，下文同 ）
```

或

```
7↙
8↙
9↙
```

由于格式控制字符串（即%d%d%d）中没有普通字符，因此在输入时要用空白符（空格、回车或 Tab 键）作为数与数之间的分隔符。

2）格式说明项

格式说明项的一般形式为：

```
%[*][输入数据宽度][长度]类型
```

其中有方括号[ ]中的项为可选项。各项的意义如下。

（1）类型：表示输入数据的类型，其类型格式符和意义如表 3.3 所示。

表 3.3　scanf()函数中的类型格式字符

类型格式符	意　　义
d	输入十进制整数
u	输入无符号十进制整数
o	输入无符号八进制整数
x 或 X	输入无符号十六进制整数
c	输入一个字符
s	输入字符串
f	输入实型数（用小数形式或指数形式）
e,E,g,G	作用与 f 相同

（2）“*”符：表示该输入项读入后不赋予相应的变量，即跳过该输入值。例如：

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

当输入为 7　8　9↙时，则把 7 被赋予 *a*，8 被跳过，9 被赋予 *b*。

（3）宽度：用十进制整数指定输入的宽度（即字符数）。例如：

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

当输入为 1234789↙时，则把 1234 被赋予 *a*，789 被赋予 *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。

3）注意事项

（1）与 printf()函数不同，scanf()函数中 double 型变量必须用%lf 输入，short 型变量必须用%hd 输入。变量必须给出变量的地址，而不仅仅是变量名。例如，“scanf("%d", a);”是非法的。

（2）scanf()函数中对实数没有精度控制。例如，“scanf("%5.2f", &a);”是非法的，不能企图用此语句输入小数为两位的实数。

（3）在输入数值数据时，输入字符流中的前导空白符（空格、回车或 Tab 键）会被自动丢弃，从空白符后的字符开始输入。构成数值数据的字符被转换成计算机的内部表示形式，并存储到对应的变量中。遇到空白符、已读入由宽度所指定的字符数、非法字符（例如，对“%d”输入“12A”时，A 即为非法字符）3 种情况即认为该数据输入完毕。所以，从键盘输入数据，若不指定输入的宽度，空白符（空格、回车或 Tab 键）可以作为数与数

之间的分隔符。

（4）若格式控制字符串中有普通字符，则输入时也要输入该普通字符。例如：

```
scanf("%d-%d-%d", &y,&m,&d);
```

则输入应为：

```
2012-3-18↙
```

又如：

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

则输入应为：

```
a=5,b=6,c=7↙
```

这时普通字符不是起到提示的作用，而是给输入制造了麻烦，应加以避免。

（5）在输入字符数据时，若格式控制字符串中没有普通字符，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↙时，则把'd'赋予 *a*，空格赋予 *b*，'e'赋予 c。

输入为：de↙时，则把'd'赋予 *a*，'e'赋予 *b*，回车符赋予 c。

只有当输入为：def↙时，才能把'd'赋予 *a*，'e'赋予 *b*，'f'赋予 c。

如果在格式控制字符串中加入空格作为分隔，例如：

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

则输入时各数据之间可加空格。

（6）若输入的数据个数多于 scanf()函数中变量的个数时，则多余的数据会被后继的 scanf()函数继续使用。

3．putchar()函数

putchar()函数是字符输出函数，其功能是在显示器上输出一个字符，函数原型在头文件 stdio.h 中，在使用 putchar()函数之前必须包含 stdio.h 文件。putchar()函数调用的一般形式为：

```
putchar(字符变量或常量)
```

例如：

```
putchar('A');    /* 输出大写字母A */
putchar(x);      /* 输出字符变量x的值 */
putchar('\n');   /* 换行，对控制字符则执行控制功能，不在屏幕上显示字符 */
```

4．getchar()函数

getchar()函数是字符输入函数，其功能是从键盘上输入一个字符，函数原型在头文件

stdio.h 中，在使用 getchar()函数之前必须包含 stdio.h 文件。getchar()函数调用的一般形式为：

```
getchar()
```

通常把输入的字符赋予一个字符变量，构成赋值语句。例如：

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

注意：

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。

3.2.3 顺序结构程序设计举例

在顺序结构的程序中，所有语句都按照它们在程序中出现的顺序从上到下、从左到右逐条执行，这种程序结构是 3 种基本结构中最简单的一种。

例 3.2 交换两变量值的方法。

```
#include <stdio.h>
int main()
{   int a=3, b=8, temp;

    temp=a;  a=b;  b=temp;        /* 交换方法1 */
    printf("a=%d  b=%d\n", a,b);
    a=a+b;  b=a-b;  a=a-b;        /* 交换方法2，不提倡该方法 */
    printf("a=%d  b=%d\n", a,b);
    return 0;
}
```

运行结果：

```
a=8  b=3
a=3  b=8
```

例 3.3 四舍五入方法。

```
#include <stdio.h>
int main()
{   float a=3.1415926, b, c;

    b=(int)(a*100+0.5)/100.0;         /* a保留两位小数后赋予b */
    printf("b=%f\n", b);
    c=(int)(a*10000+0.5)/10000.0;     /* a保留4位小数后赋予c */
    printf("c=%f\n", c);
    printf("a=%.2f\n", a);            /* a输出时保留两位小数，但a的数值没有变 */
    printf("a=%.7f\n", a);            /* a只有7位有效数字，所以最后一位有误差 */
    return 0;
```

```
}
```

运行结果：

```
b=3.140000
c=3.141600
a=3.14
a=3.1415925
```

例 3.4　求一元二次方程的根。

```
#include <stdio.h>
#include <math.h>
int main()
{   float a,b,c,disc,p,q;

    printf("请输入一元二次方程的三个系数：");
    scanf("%f%f%f", &a,&b,&c);
    disc=b*b-4*a*c;
    p=-b/(2*a);
    q=sqrt(disc)/(2*a);
    printf("x1=%5.2f  x2=%5.2f\n", p+q,p-q);
    return 0;
}
```

运行结果：

```
请输入一元二次方程的三个系数：1 8 15
x1=-3.00  x2=-5.00
```

本例中不考虑 disc<0 的情况，要编写更为通用的求根程序，就要用到 3.3 节中的分支结构。

3.3　选择结构程序设计

选择结构（又称分支结构）是根据运行时的情况自动选择要执行的语句。在 C 语言中，用 if 语句或 switch 语句可以实现选择结构。如何表示条件，如何使用选择结构控制程序流程，是本节要掌握的主要内容。

3.3.1　关系运算符和关系表达式

关系运算就是比较运算，用于比较两个量之间的大小关系，如大于、小于等。在程序中经常需要比较两个量之间的大小关系，以决定程序下一步的工作。C 语言中共有<（小于）、<=（小于或等于）、>（大于）、>=（大于或等于）、==（等于）、!=（不等于）6 种关系运算符，其中前 4 种运算符（<、<=、>、>=）的优先级相同，后两种运算符（==、!=）的优先级也相同，且前 4 种的优先级高于后两种的优先级。关系运算符的优先级低于算术运算符，高于赋值运算符。关系运算符都是双目运算符，其结合性均为左结合。

用关系运算符将两个表达式连接起来的式子称为关系表达式，关系表达式的一般形式为：

```
表达式1  关系运算符  表达式2
```

其中的表达式 1 或表达式 2 可以是算术表达式、赋值表达式、关系表达式等。例如：

```
b*b-4*a*c>0     (a=b+c)<(d=50)     ch>'A'
```


关系表达式的值是逻辑值“真”或“假”，若关系表达式成立，其值为“真”，用整数“1”表示；若关系表达式不成立，其值为“假”，用整数“0”表示。例如，23>18 成立，其值为“真”，即为 1；而'A'>'a'不成立，其值为“假”，即为 0。

使用关系运算符时务必要注意以下几点：

（1）进行相等比较时一定要用双等号“==”，因为 C 语言中的单个等号“=”是赋值运算符。如果误用表达式“a=5”来判断整型变量 *a* 的值是否等于 5，编译系统不会报错，因为编译器将其理解为赋值运算，即把 5 赋给变量 *a*。

（2）不要把 C 语言表达式“5<a<10”理解为数学关系式“5<a<10”，因为 C 语言表达式 5<a<10 的值永远为真（因为 10 总是大于 5<a 的比较结果）；而数学关系式 5<a<10 只有当变量 *a* 的值在 5～10 之间时才成立。

（3）尽量避免实型表达式与数值常量进行“==”或“!=”比较，因为实型量有精度限制。若用 e 表示某个实型表达式，则可以将表达式 e==5 转化为表达式 fabs(e−5)<1e−6。

3.3.2 逻辑运算符和逻辑表达式

为了表达数学关系式 5<a<10，这时就要用到逻辑运算符。C 语言中共有 ！（非运算）、&&（与运算）、||（或运算）3 种逻辑运算符。其中 ！运算符优先级最高，&&运算符优先级次之，|| 运算符优先级最低。&& 和 || 均为双目运算符，具有左结合性；!为单目运算符，具有右结合性。

逻辑运算符进行运算时的运算规则如下：

（1）&& 参与运算的两个量都为真时，结果才为真，否则为假。例如，5>0 && 4>2，由于 5>0 为真，4>2 也为真，相与的结果也为真。

（2）|| 参与运算的两个量都为假时，结果才为假，否则为真。例如，5>0 || 5>8，由于 5>0 为真，相或的结果也就为真。

（3）！参与运算量为真时，结果为假；参与运算量为假时，结果为真。例如，!(5>0) 的结果为假。

用逻辑运算符将两个表达式连接起来的式子称为逻辑表达式，逻辑表达式的一般形式为：

```
表达式1  逻辑运算符  表达式2
```

其中的表达式 1 或表达式 2 可以是逻辑表达式、关系表达式等。下面举几个逻辑表达式实例。

（1）数学关系式“5<a<10”对应的 C 语言表达式：5<a && a<10 。

（2）判别某一个字符 ch 是否是英文字母的表达式：ch>='A' && ch<='Z' || ch>='a' && ch<='z' 。

（3）判断某人是否是男性（用字母 M 或 m 表示）且年龄为 20 岁：(sex=='M' || sex=='m') && age==20 。

逻辑表达式的值是逻辑值“真”或“假”，与关系表达式一样，若逻辑表达式的值为“真”，用整数“1”表示；为“假”，用整数“0”表示。但反过来在判断一个量是为“真”还是“假”时，“0”就表示“假”，非“0”的数值都表示“真”。例如，由于 5 和 3 均为非“0”，因此 5 && 3 的值为“真”，即为 1。又如，由于字符'0'的 ASCII 码是 48 非“0”，因此 '0' || 0 的值为“真”，即为 1。再如，表达式! a 等价于表达式 a==0，而表达式 a 等价于表达式 a!=0 。

由于数值 0 表示假，非 0 表示真，所以逻辑运算符两边的表达式也可以是算术表达式、赋值表达式等。这时要注意各种运算符的优先级，部分已学过的运算符的优先级关系如图 3.2 所示。按照运算符的优先级可以得到：

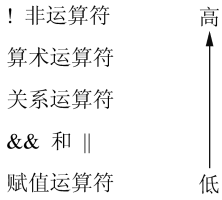

图 3.2　部分运算符的优先级关系图

a>b && c>d　　等价于　(a>b) && (c>d)
!b==c || d<a　　等价于　((!b)==c) || (d<a)
a+b>c && x+y<z　等价于　((a+b)>c) && ((x+y)<z)

最后要特别说明的是，逻辑表达式中的求值采用“短路求值法”，即：

- a && b　只有当 *a* 为真时，才对 *b* 求值，否则不再对 *b* 求值。
- a || b　只有当 *a* 为假时，才对 *b* 求值，否则不再对 *b* 求值。

例如：设有“int a=1,b=2,c=3,d=4,m=1,n=1;”，则对表达式(m=a>b) && (n=c>d)求值后 *n* 仍然保持值 1 而不是 0，因为(m=a>b)的值为假，所以根本没有对(n=c>d)求值。

3.3.3　if 语句

1．if 语句

选择结构最常用的语句是 if 语句，其根据给定的条件进行判断，以决定是否执行某个分支程序段。if 语句的一般形式为：

```
if (表达式) 语句1;
 [ else 语句2; ]
```

其中，表达式一般为关系表达式或逻辑表达式，理论上可以是任何表达式。方括号[ ]里面的内容是可选项，可以省略，即形成不带 else 的 if 语句。

不带 else 的 if 语句的执行过程是：如果表达式的值为真，则执行语句 1，否则不执行语句 1，如图 3.3（a）所示。而带 else 的 if 语句的执行过程是：如果表达式的值为真，则执行语句 1，否则执行语句 2，如图 3.3（b）所示。

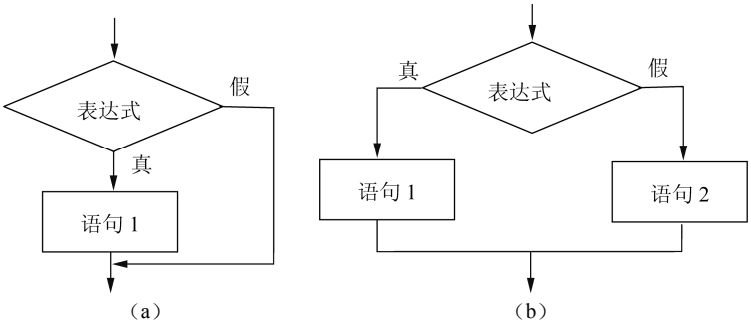

图 3.3　if 语句的执行流程

例 3.5 输入两个整数，输出其中的较大数。

```
#include <stdio.h>
int main()
{   int a,b,max;

    printf("Input two integers: ");
    scanf("%d%d", &a,&b);
    max=a;
    if (max<b) max=b;
    printf("Max=%d\n", max);
    return 0;
}
```

运行结果：

```
Input two integers: 21 56
Max=56
```

本例中，输入两个数存入变量 *a* 和 *b*，先把 *a* 赋予变量 max，再用 if 语句判断 max 是否小于 *b*，若小于，则把 *b* 赋予 max，因此 max 中总是较大数，最后输出 max 的值。

例 3.6 输入两个整数，先输出其中的较小数，再输出较大数。

```
#include <stdio.h>
int main()
{   int a,b,t;

    printf("Input two integers: ");
    scanf("%d%d", &a,&b);
    if (a>b) {  t=a;  a=b;  b=t;  }
    printf("Min=%d  Max=%d\n", a,b);
    return 0;
}
```

运行结果：

```
Input two integers: 56 21
Min=21  Max=56
```

在 if 语句的一般形式中，语句 1 或语句 2 都应是单个语句，如果在满足条件时想要执行一组（多条）语句，则必须把这一组语句用 {} 括起来构成一个复合语句，本例中就是。

例 3.7 用带 else 的 if 语句重做例 3.5（输入两个整数，输出其中的较大数）。

```
#include <stdio.h>
int main()
{   int a,b,max;

    printf("Input two integers: ");
    scanf("%d%d", &a,&b);
    if (a>b) max=a;
    else max=b;
```

```
    printf("Max=%d\n", max);
    return 0;
}
```

运行结果：

```
Input two integers: 21 56
Max=56
```

例 3.8 输入一个年份，判断该年份是否是闰年后输出。

分析：地球围绕太阳公转一圈的实际时间是 365 天 5 小时 48 分 46 秒。如果一年只有 365 天，每年就多出 5 个多小时，4 年多出 23 小时 15 分 4 秒，差不多就是一天，于是人们决定每 4 年增加 1 天，即产生了闰年。但这样做时间实际上又少了约 45 分，每 100 年有 25 个闰年会导致少了 18 小时 43 分 20 秒，于是又决定每 100 年只有 24 个闰年（世纪年不作为闰年）。

但这样又产生了新的问题，因为每个平年多出 5 小时 48 分 46 秒，100 个平年多出 581 小时 16 分 40 秒，而 24 天即为 576（24×24）小时，每 100 年只有 24 个闰年又导致时间多出 5 小时 16 分 40 秒，于是最后又决定每 400 年增加一个闰年，这样就比较接近实际情况了。

```
#include <stdio.h>
int main()
{   int year;

    printf("请输入一个年份: ");
    scanf("%d", &year);
    if (year%100!=0 && year%4==0 || year%400==0)
        printf("是闰年\n");
    else
        printf("不是闰年\n");
    return 0;
}
```

运行结果：

```
请输入一个年份: 2012
是闰年
```

2．if-else-if 语句

前面的 if 语句只能用于两个分支的情况。当有多个分支选择时，可采用 if-else-if 语句，其一般形式为：

```
if (表达式1)
    语句1;
else if (表达式2)
    语句2;
else if (表达式3)
    语句3;
    …
```

```
else if (表达式n)
    语句n;
else
    语句n+1;
```

其执行过程是：依次判断各表达式的值，当某个表达式的值为真时，则执行其对应的语句，然后跳到整个 if 语句之后继续执行程序。如果所有的表达式均为假，则执行语句 *n*+1，然后继续执行后续程序，如图 3.4 所示。格式中的语句 1、语句 2、…、语句 *n*、语句 *n*+1 都只能是单个语句。

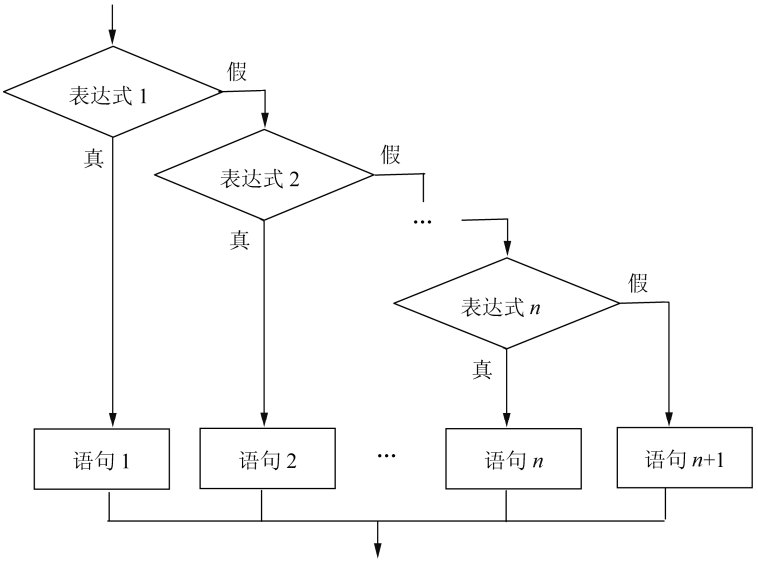

图 3.4　if-else-if 语句的执行流程

说明：最后的“else 语句 n+1”常起到检查是否出错的作用，这一部分也可以省略。若省略，则当 n 个条件均不成立时，就什么也不执行。注意，语句中的 *n* 个条件应当是互斥的，即不应当出现有两个或两个以上条件同时成立的情况。

例 3.9　判断键盘输入字符的类别并输出。

分析：要判断键盘输入字符的类别，可根据输入字符的 ASCII 码值来进行。由 ASCII 码表知 ASCII 码值小于 32 的为控制字符，在字符'0'和'9'之间的为数字，在字符'A'和'Z'之间为大写字母，在字符'a'和'z'之间为小写字母，其余则为其他字符。这是一个多分支选择的问题，可用 if-else-if 语句编程，判断输入字符 ASCII 码所在的范围，分别给出不同的输出。例如，输入字符'e'，输出其为小写字母。

```
#include <stdio.h>
int main()
{   char c;

    printf("Input a character: ");   c=getchar();
    if (c<32)
```

```
        printf("This is a control character.\n");
    else if (c>='0'&&c<='9')
        printf("This is a digit.\n");
    else if (c>='A'&&c<='Z')
        printf("This is a capital letter.\n");
    else if (c>='a'&&c<='z')
        printf("This is a small letter.\n");
    else
        printf("This is an other character.\n");
    return 0;
}
```

运行结果：

```
Input a character: #
This is an other character.
```

例 3.10 输入一个百分制成绩，判断这一百分制成绩，输出对应的五级制成绩，若输入的百分制成绩错误，应报错。

分析：百分制成绩的范围为 0～100，五级制成绩为优、良、中、及格、不及格，分别用字母 A、B、C、D、E 来表示。判断标准为：90 分以上为优；80～89 为良；70～79 为中；60～69 为及格；60 分以下为不及格。这是一个多分支选择的问题，可用 if-else-if 语句编程。

```
#include <stdio.h>
int main()
{   int score;  char ch;

    printf("请输入百分制成绩(0～100): ");
    scanf("%d", &score);
    if (score<0 || score>100)
        {  printf("百分制成绩超出范围!\n");  return 1;  }
    if (score>=90) ch='A';
    else if (score>=80) ch='B';
    else if (score>=70) ch='C';
    else if (score>=60) ch='D';
    else ch='E';
    printf("对应的五级制成绩: %c\n", ch);
    return 0;
}
```

运行结果：

```
请输入百分制成绩(0～100): 86
对应的五级制成绩: B
```

本例中，首先排除出错情况，保证 score 在 0～100 之间，这样条件 score>=90 && score<=100 就可以简化为 score>=90，由于 score>=90 不成立时才会判断 score>=80，因此条件 score>=80 就等价于条件 score>=80 && score<=89，依此类推。

3．if 语句的嵌套

当 if 语句中的语句 1 或语句 2 又是 if 语句时，则构成了 if 语句嵌套的情形。如图 3.5 所示给出了几种 if 语句的嵌套形式。

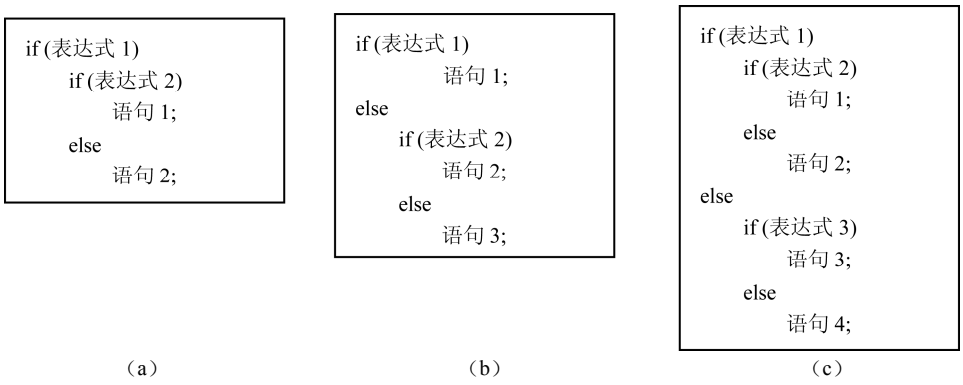

图 3.5　if 语句的嵌套形式

在写 if 语句的嵌套结构时，要特别注意 if 和 else 的配对问题。例如，对于图 3.5（a），因为有两个 if，这时 else 究竟与哪一个 if 配对呢？即对于图 3.5（a）中 if 语句的嵌套结构，是否也能理解为图 3.6（a）形式的 if 语句的嵌套结构？如果能，就会产生了二义性。为此，C 语言规定，else 总是与同一层最接近它的 if 配对，即图 3.6（a）的理解是错误的。如果要使 else 与 if (表达式 1)配对，必须将“if (表达式 2) 语句 1;”用大括号括起来，形成复合语句，即写成图 3.6（b）所示的形式。

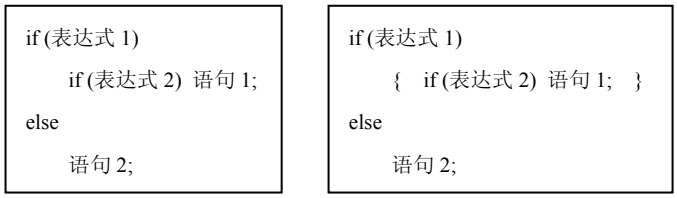

图 3.6　if 语句嵌套时 else 的配对问题

为了能区分嵌套的层次，常采用缩进的书写格式来表达不同的层次，使同一层次具有相同的缩进位置，这样写出的程序便于阅读，也易于查错。当然，缩进仅仅是为了改善可读性，程序的语义还是要靠语法来保证，图 3.6（b）中为了 else 与 if (表达式 1)配对，还是要加大括号才行。

例 3.11　输入 x 的值，根据下列函数，输出对应 y 的值。

$$y=\begin{cases}-1 & (x<0)\\ 0 & (x=0)\\ 1 & (x>0)\end{cases}$$

分析：if 语句嵌套结构的实质是为了进行多分支选择，本例中有 3 种选择，即 $x<0$、$x=0$ 或 $x>0$，可以采用 if 语句的嵌套结构实现（程序 1），显然也可以用 if-else-if 语句实现（程序 2）。

程序1

```
#include <stdio.h>
int main()
{   float x;  int y;

    printf("x=? ");
    scanf("%f", &x);
    if (x<=0)
        if (x<0) y=-1;
        else y=0;
    else y=1;
    printf("y=%d\n", y);
    return 0;
}
```

程序2

```
#include <stdio.h>
int main()
{   float x;  int y;

    printf("x=? ");
    scanf("%f", &x);
    if (x<0) y=-1;
    else if (x>0) y=1;
    else y=0;
    printf("y=%d\n", y);
    return 0;
}
```

实现多分支选择除了可以用 if 语句的嵌套结构和 if-else-if 语句结构外，有时用并列的 if 语句也可以实现，下面的程序段 1 就可以实现例 3.11 中的要求。对 if 语句和 if 语句嵌套结构的错误理解都会导致程序的错误，如下面的程序段 2 和程序段 3。

程序段 1

```
scanf("%f", &x);
y=0;
if (x<0) y=-1;
if (x>0) y=1;
printf("y=%d\n", y);
```

程序段 2

```
scanf("%f", &x);
y=-1;
if (x>0)
    y=1;
else y=0;
printf("y=%d\n", y);
```

程序段 3

```
scanf("%f", &x);
y=0;
if (x<=0)
    if (x<0) y=-1;
else y=1;
printf("y=%d\n", y);
```

由此可以看出，if 语句嵌套时，if 语句一般形式中语句 1 又是一个 if-else 结构，非常容易出错；而语句 2 又是一个 if-else 结构，则不易出错。因此建议读者尽量不要嵌套在语句 1 分支上，尽量嵌套在语句 2 分支上，从而形成 if-else-if 的结构形式。

例 3.12　求 3 个不同整数中的最大值。

分析：本例是一个多分支选择问题，为了求出最大值，可以用 if 语句的嵌套结构，也可以用 if-else-if 语句结构，甚至用并列的 if 语句结构。如果将问题改为“求 4 个不同整数中的最大值”，请读者自行比较各种方法的优劣。

```
#include <stdio.h>
int main()
{   int a,b,c,max;

    printf("请输入3个整数: ");
    scanf("%d%d%d", &a,&b,&c);
    max=a;
    if (c>b)
       {  if (c>a) max=c;  }
    else
       {  if (b>a) max=b;  }
    printf("Max=%d\n", max);
    return 0;
}
```

```
if-else-if语句结构
    if (a>b && a>c) max=a;
    else if (b>a && b>c) max=b;
    else max=c;
```

```
并列的if语句结构
    max=a;
    if (max<b) max=b;
    /* max中已是a和b中的较大数 */
    if (max<c) max=c;
```

例 3.13 求一元二次方程的根。

分析：例 3.4 没有考虑求根时各种可能的情况。一元二次方程 $ax^2+bx+c=0$ 的根有以下几种情况：

（1）当 $a=0$ 时，如果 $b=0$ 时，方程无解，否则方程退化为一元一次方程 $bx+c=0$，只有一个实根 $-c/b$。

（2）当 $a\neq0$ 时，如果 $b^2-4ac<0$ 时，方程有两个共轭复根，否则方程有两个实根。

```
#include <stdio.h>
#include <math.h>
int main()
{   float a,b,c,disc,p,q;

    printf("请输入一元二次方程的三个系数：");
    scanf("%f%f%f", &a,&b,&c);
    if (a==0)    /* 根据3.3.1节中的解释，条件a==0可改为fabs(a)<1e-6 */
       if (b==0)  printf("方程无解！\n");   /* 同样，条件也可改为fabs(b)<1e-6 */
       else  printf("方程退化为一次方程，只有一个实根：%f\n", -c/b);
    else
    {   disc=b*b-4*a*c;
       p=-b/(2*a);
       q=sqrt(fabs(disc))/(2*a);
       if (disc<0)  printf("有两个共轭复根：实部=%f, 虚部=%f\n", p,q);
       else  printf("有两个实根：根1=%f, 根2=%f\n", p+q,p-q);
    }
    return 0;
}
```

运行结果：

```
请输入一元二次方程的三个系数：0 2 5
方程退化为一次方程，只有一个实根：-2.500000
请输入一元二次方程的三个系数：1 -5 6
有两个实根：根1=3.000000，根2=2.000000
请输入一元二次方程的三个系数：1 4 5
有两个共轭复根：实部=-2.000000，虚部=1.000000
```

3.3.4 条件运算符

条件运算符为“?”和“:”，是 C 语言中唯一的一个三目运算符，即有 3 个参与运算的量。由条件运算符组成条件表达式的一般形式为：

```
表达式1 ? 表达式2 : 表达式3
```

其求值规则为：先对表达式 1 求值，若值为真（即非 0），则以表达式 2 的值作为条件表达式的值；若值为假（即 0），则以表达式 3 的值作为条件表达式的值，而运算结果的类型由表达式 2 的值和表达式 3 的值中较高的那个类型来决定。例如，当 $a>b$ 成立时，表达式“(a>b) ? 2 : 3.5”的运算结果是 2.0，而非 2，这一点务必注意。

使用条件运算符时，还应注意以下几点：

（1）条件运算符的优先级低于关系运算符和算术运算符，但高于赋值运算符。因此“max=(a>b)?a:b”等价于“max=a>b?a:b”。

（2）条件运算符的结合方向是自右至左。例如，“a>b?a:c>d?c:d”应理解为“a>b?a:(c>d?c:d)”，这也就是条件表达式嵌套的情形，即其中的表达式 3 又是一个条件表达式。

由条件运算符组成的表达式是条件表达式。合理使用条件表达式，不但使程序简洁，也提高了运行效率。

例 3.14 用条件表达式重做例 3.5（输入两个整数，输出其中的较大数）。

```
#include <stdio.h>
int main()
{   int a,b,max;

    printf("Input two integers: ");
    scanf("%d%d", &a,&b);
    max=a>b?a:b;
    printf("Max=%d\n", max);
    return 0;
}
```

例 3.15 用条件表达式重做例 3.8（输入一个年份，判断该年份是否是闰年后输出）。

```
#include <stdio.h>
int main()
{   int  year,leap;

    printf("请输入一个年份: ");
    scanf("%d", &year);
    leap = (year%100!=0 && year%4==0 || year%400==0);
    printf("%s\n", (leap) ? "是闰年" : "不是闰年" );
    return 0;
}
```

3.3.5 switch 语句

多分支选择结构可以用 if 语句的嵌套结构实现，但如果分支较多，嵌套的 if 语句层次也增多，程序变得复杂冗长，可读性降低。C 语言还提供了另一种用于多分支选择的 switch 语句，其一般形式为：

```
switch (表达式)
{   case 常量表达式1:  语句1;
    case 常量表达式2:  语句2;
        …
    case 常量表达式n:  语句n;
    [  default  :  语句n+1;  ]
}
```

其中表达式的值必须是整型、字符型或枚举型，常量表达式的值类型必须与“表达式”的值类型相容。switch 语句的执行过程是：计算表达式的值，并逐个与其后的常量表达式值相比较，当表达式的值与某个常量表达式的值相等时，即执行其后的语句，然后不再进行判断，继续执行后面所有 case 后的语句，直到右大括号前的最后一条语句。default 是可选的，如果表达式的值与所有 case 后的常量表达式的值均不相等时，则执行 default 后的语句，假如没有 default，则这时 switch 语句什么也不做。

例 3.16 输入整数 1～7，输出对应星期几的英文。

```
#include <stdio.h>
int main()
{   int a;

    printf("Input integer number: ");
    scanf("%d", &a);
    switch (a)
    {   case 1 : printf("Monday\n");
        case 2 : printf("Tuesday\n");
        case 3 : printf("Wednesday\n");
        case 4 : printf("Thursday\n");
        case 5 : printf("Friday\n");
        case 6 : printf("Saturday\n");
        case 7 : printf("Sunday\n");
        default: printf("Error\n");
    }
    return 0;
}
```

运行结果：

```
Input integer number: 6
Saturday
Sunday
Error
```

从运行结果看，程序没有达到所希望的目的，这是用 switch 语句时最常见的错误。为什么会出现这种情况呢？这恰恰反映了 switch 语句的一个特点。在 switch 语句中，“case 常量表达式”只相当于一个语句标号，表达式的值和某个标号相等则转向该标号执行，但不能在执行完该标号后的语句后自动跳出 switch 语句，所以出现了继续执行后面所有语句的情况。这是与前面介绍的 if 语句完全不同的，应特别注意。为了避免上述情况，C 语言还提供了一种 break 语句，可用于跳出 switch 语句，break 语句只有关键字 break。

修改例 3.16 中的程序，在每个 case 后的语句之后增加 break 语句，使执行完每一个分支之后均可跳出 switch 语句，从而避免输出不应输出的内容。

```
#include <stdio.h>
int main()
{   int a;

    printf("Input integer number: ");
```

```
    scanf("%d", &a);
    switch (a)
    {   case 1 : printf("Monday\n");  break;
        case 2 : printf("Tuesday\n");  break;
        case 3 : printf("Wednesday\n");  break;
        case 4 : printf("Thursday\n");  break;
        case 5 : printf("Friday\n");  break;
        case 6 : printf("Saturday\n");  break;
        case 7 : printf("Sunday\n");  break;
        default: printf("Error\n");  break;
    }
    return 0;
}
```

运行结果：

```
Input integer number: 6
Saturday
```

在使用 switch 语句时还应注意以下几点：

- 在 case 后的各常量表达式的值不能相同，否则会出现错误。
- 在 case 后，允许有多个语句，可以不用{}括起来。
- 各 case 和 default 子句的先后顺序可以变动，而不会影响程序执行结果。
- 多个 case 可以共用一组执行语句。

例 3.17 用 switch 语句重做例 3.10（输入一个百分制成绩，判断这一百分制成绩，输出对应的五级制成绩，若输入的百分制成绩错误，应报错）。

分析：百分制成绩 score 的合理范围为 0～100，表达式 score/10 求值后，共有 11 种情况，其中值 10 和值 9 是同一个分支，值 5～0 可通过 default 后的语句实现。

```
#include <stdio.h>
int main()
{   int score;  char ch;

    printf("请输入百分制成绩(0～100): ");
    scanf("%d", &score);
    if (score<0 || score>100)
        {  printf("百分制成绩超出范围!\n");  return 1;  }
    switch (score/10)
    {   case 10 :
        case 9 : ch='A';  break;   /* 两个case 共用的语句 */
        case 8 : ch='B';  break;
        case 7 : ch='C';  break;
        case 6 : ch='D';  break;
        default: ch='E';  break;
    }
    printf("对应的五级制成绩: %c\n", ch);
    return 0;
```

```
}
```

运行结果：


```
请输入百分制成绩(0～100): 100
对应的五级制成绩: A
```

例 3.18 计算器程序。输入一个简单的表达式（只能做加、减、乘或除 4 种运算），输出计算结果。

```
#include <stdio.h>
int main()
{   float a,b,s;
    char c;

    printf("请输入表达式：");
    scanf("%f%c%f", &a,&c,&b);
    switch (c)
    {   case '+' : s=a+b;  break;
        case '-' : s=a-b;  break;
        case '*' : s=a*b;  break;
        case '/' : if (b==0) {  printf("除数为0错误！\n");  return 1;  }
                   else s=a/b;
                   break;
        default : printf("运算符非法！\n");  return 1;
    }
    printf("表达式的值为：%f\n", s);
    return 0;
}
```

运行结果：

```
请输入表达式：3*5
表达式的值为：15.000000
```

```
请输入表达式：5/0
除数为0错误！
```

```
请输入表达式：2&3
运算符非法！
```

本例说明两点：①switch 语句中表达式的值为字符类型；②switch 语句中又可以嵌套 if 语句。

3.4 循环结构程序设计

循环结构是程序中一种很重要的结构，其特点是：在给定条件成立时，反复执行某程序段，直到条件不成立为止。给定的条件称为循环条件，反复执行的程序段称为循环体。C 语言提供了 3 种循环语句，分别是 while 语句、do-while 语句和 for 语句。如何表示循环条件，如何构造循环体并避免无限循环（即“死循环”），是本节要掌握的主要内容。

3.4.1 while 循环结构

while 语句的一般形式为：

```
while (表达式)
    语句;
```

其中表达式是循环条件，语句为循环体。当循环体是多个语句时，应当用{}括起来构成一句复合语句。while 语句的执行流程如图 3.7 所示。很显然，循环体中应当有语句来改变循环条件的取值，否则就有可能产生死循环。while 语句的特点是：先计算表达式的值，然后根据表达式的值决定是否执行循环体。当表达式的值一开始就为假（即 0）时，则循环体一次也不执行。

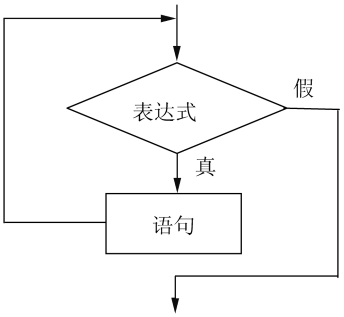

图 3.7　while 语句的执行流程

例 3.19　输入一行字符，统计并输出字符的个数。

分析：一行字符以回车符结束，输入回车符时用 getchar()函数读到的是字符'\n'。因此如果程序读到的字符不是'\n'就不断计数，否则结束循环输出读到的字符个数（不包括回车符）。

```
#include <stdio.h>
int main()
{   char c;  int n;

    printf("请输入一行字符：\n");
    n=0;    /* 计数器赋初值0 */
    c=getchar();          /* 读第一个字符，也就为循环控制变量赋初值 */
    while (c!='\n')       /* c为循环控制变量 */
    {   n++;              /* n称为计数器 */
        c=getchar();      /* 读下一个字符，也就改变了循环控制变量的值 */
    }
    printf("字符个数为：%d\n", n);
    return 0;
}
```

用赋值运算符改写后

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

运行结果：

```
请输入一行字符：
Expo 2010 Shanghai
字符个数为：18
```

本例中循环条件的取值取决于变量 *c* 的取值，所以称 *c* 为循环控制变量，即变量 *c* 起到了控制循环体是否继续被执行的作用。注意，在进入循环之前不要忘记给循环控制变量赋初值，循环体中也不要忘记改变循环控制变量的值，以避免死循环的发生。程序中变量 *n* 起到计数的作用，通常称它为计数器，当然不要忘记给计数器赋初值 0。由于在 C 语言中赋值是运算符，所以读字符并计数的程序段可以改写得更为简洁。

例 3.20　输入一个班级某门课程的成绩，统计并输出全班该门课程的平均成绩。

分析：要求出平均成绩，首先要求出全班的总成绩和学生数。由于不知道全班有多少

个学生，也就不知道要读多少次成绩。但考虑到成绩没有负数，因此可以用输入一个负数来表示成绩输入完毕，如果程序读到的成绩大于等于 0 就不断累加和计数，否则结束循环，输出平均成绩。


```
#include <stdio.h>
int main()
{   float score, sum;  int n;

    printf("请输入所有学生的成绩，负数表示输入完毕：\n");
    sum=n=0;              /* 累加器，计数器赋初值0 */
    scanf("%f", &score);
    while (score>=0)      /* 这里假定只要score>=0，就是一个有效的成绩 */
    {   sum+=score;       /* sum称为累加器 */
        n++;              /* n称为计数器 */
        scanf("%f", &score);
    }
    if (n>0) printf("平均成绩为：%6.2f\n", sum/n);
    else printf("没有输入有效的成绩！");
    return 0;
}
```

运行结果：

```
请输入所有学生的成绩，负数表示输入完毕：
71 64 82 77 83 76 95 78 -1
平均成绩为：  78.25
```

本例中变量 score 为循环控制变量，同例 3.19 一样变量 *n* 为计数器，而变量 sum 起到累加的作用，称为累加器，在进入循环之前不要忘记给累加器赋初值 0。

3.4.2 do-while 循环结构

do-while 语句的一般形式为：

```
do
    语句;
while (表达式);
```

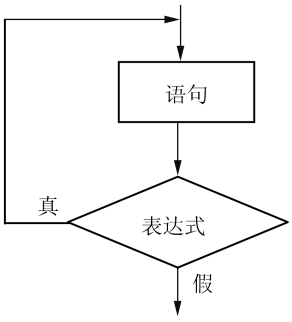

图 3.8　do-while 语句的执行流程

其中语句为循环体，表达式是循环条件，注意在表达式的右圆括号)后有一个分号。当循环体是多个语句时，应当用{}括起来构成一句复合语句。建议即使循环体只有一句语句也用{}括起来，且将右大括号}放在同一行的 while 关键字之前，这样在看到 while 时可以很容易地区分是 do-while 语句还是 while 语句。do-while 语句的执行流程如图 3.8 所示。比较图 3.7 与图 3.8 可以很容易看出 do-while 语句与 while 语句的区别：do-while 语句是先执行后判断，因此 do-while 语句至少要执行一次循环体；而 while 语句是先判断后执行，如果条件一开始就不满足，则循环体一次也不执行。因此 do-while 语句一般用在循环体至少要执行一次的情况。下面就是一段判断口令是否正确的程序段：

```
long password;
do {
   printf("请输入口令：\n");
   scanf("%ld", &password);
}while (password!=123456);   /* 假定口令是123456 */
```

例 3.21　输入一个长整数，输出它是一个几位数。

分析：设长整数为 a_0，则进行 $a_1=a_0/10$ 运算后，a_1 仍然是一个长整数，只不过 a_1 的位数比 a_0 少了一位，只要 a_1 不为 0，就继续进行 $a_2=a_1/10$ 运算，这样一直进行下去，直到某次除法运算后商为 0 为止。如果另设一个计数器 n，用它来统计除法的次数，则循环结束后 n 中存放的就是 a_0 的位数。由于一个长整数至少是 1 位数，循环至少要做一次，因此适合用 do-while 语句来实现循环。因为求出 a_1 后 a_0 就没有用了，所以程序中 a_0，a_1，a_2，… ，可以用同一个变量 a 。

```
#include <stdio.h>
int main()
{   long a;  int n;

   printf("请输入一个长整数：");
   scanf("%ld", &a);
   n=0;
   do {
      n++;
      a/=10;
   }while (a!=0);
   printf("它是一个%d位数\n", n);
   return 0;
}
```

运行结果：

```
请输入一个长整数：56789
它是一个5位数
```

例 3.22　输入一个大于等于 3 的正整数 m，判断它是否是素数，并输出判断结果。

分析：素数数学上也称质数，特点是除了能被 1 和数本身（本题为 m）整除外，不能被其他数（本题为 n）整除。根据素数的概念可以采用穷举法来检测 m 不能被 2～m–1 中任何一个数整除时，才能确定它是素数。设变量 n 的初值为 2，如果 $m\%n$ 等于 0，则退出循环（说明 m 能被 n 整除，m 不是素数），否则 n 加 1 后继续做 $m\%n$ 运算，直到 $m\%n$ 等于 0 为止（当 n 等于 m 时，必有 $m\%n$ 等于 0，这说明 m 是素数）。由于 $m\%n$ 运算至少要做一次，因此可以用 do-while 语句来实现循环。

```
#include <stdio.h>
int main()
{   int m,n;

   printf("请输入一个大于等于3的正整数：");
```

```
    scanf("%d", &m);
    n=1;    /* 保证第1次做m%n运算时，n的值为2 */
    do {
       n++;
    }while (m%n!=0);
    printf("%d%s素数\n", m,(n<m)? "不是":"是");  /* 使用条件运算符简化了程序 */
    return 0;
}
```

运行结果：

```
请输入一个大于等于3的正整数：11
11是素数
```

3.4.3 for 循环结构

for 语句是 C 语言所提供的功能更强，使用更广泛的一种循环语句。其一般形式为：

```
for (表达式1; 表达式2; 表达式3)
    语句;
```

其中，表达式 1 通常用来给循环变量赋初值，一般是赋值表达式；表达式 2 通常是循环条件，一般为关系表达式或逻辑表达式；表达式 3 通常可用来修改循环变量的值，一般是赋值表达式。这 3 个表达式都可以是逗号表达式，即每个表达式都可由多个表达式组成。一般形式中的语句即为循环体，当循环体是多个语句时，应当用{}括起来构成一句复合语句。

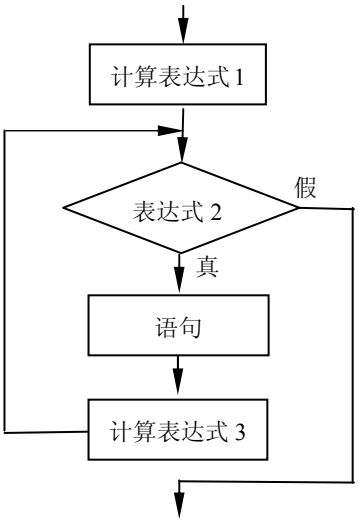

图 3.9　for 语句的执行流程

for 语句的执行流程（如图 3.9 所示）是：

（1）计算表达式 1 的值。

（2）计算表达式 2 的值，若值为真（非 0），则执行循环体一次，否则跳出循环。

（3）计算表达式 3 的值，转回第（2）步继续执行。

在整个 for 循环过程中，表达式 1 只计算一次，表达式 2 和表达式 3 则可能计算多次。循环体可能多次执行，也可能一次都不执行。

需要说明的是，3 个表达式都是任选项，都可以省略。在循环变量已赋初值的情况下，可省去表达式 1。如果省去表达式 2 或表达式 3 则将造成死循环，这时应在循环体内设法结束循环。下面列出的几种 for 语句形式都是合法的，但分号间隔符不能少。

```
for ( ；表达式；表达式)  省去了表达式1
for (表达式；  ；表达式)  省去了表达式2
for (表达式；表达式；)  省去了表达式3
for ( ；  ；)  省去了全部表达式
```

由图 3.9 可知，表达式 1 是在执行循环之前完成的，因此可以写在 for 语句的前面，而表达式 3 是每次执行了循环体以后再求值的，因此可以放在循环体语句的后面，作为循环体的一部分，即 for 语句可以写成如下形式：

```
表达式1;
for ( ; 表达式2; )
{   语句;
    表达式3;
}
```

其中的“for (; 表达式 2;)”也可以用“while (表达式 2)”来代替，即 for 循环也可写成 while 循环的形式。由此可见，for 循环结构本质上跟 while 循环结构类似，都是先判条件，后决定是否执行循环体。在完成同样功能的情况下，for 循环结构简洁、方便，for 语句不仅可用于循环次数已经确定的情况，也可用于循环次数不确定而只给出循环结束条件的情况。

例 3.23 求正整数 n 的阶乘 $n!$，其中 n 由用户输入。

分析：$n!=1\times2\times\cdots\times n$，设 $f_{n-1}=(n-1)!$，则利用公式 $f_n=f_{n-1}\times n$，初始值 $f_1=1$，可算出 $n!$。

```
#include <stdio.h>
int main()
{   int i,n;  double fn;

    printf("n=? ");
    scanf("%d", &n);
    for (fn=1,i=2; i<=n; i++)    /* 累乘器赋初值1 */
        fn*=i;                   /* fn称为累乘器 */
    printf("%d!=%.0lf\n", n,fn);
    return 0;
}
```

运行结果：

```
n=? 10
10!=3628800
```

注意本例中表达式 1 是一个逗号表达式，因为累乘器 fn 和循环控制变量 i 都要初始化。

例 3.24 求 n 个数中的最大数，其中 n 由用户输入。

分析：例 3.12 中给出了求 3 个数中的最大值的 3 种方法，其中 if-else-if 语句结构比较好理解，但当数的个数增加后该方法就不可行了，而并列的 if 语句结构就很容易推广到多个数，甚至 n 个数的情况。设前 $n-1$ 个数的最大值为 $\max_{n-1}$，则利用 $max_n=\max(\max_{n-1}, x)$，初始值 $\max_0$ 为尽可能小的数，可求出 n 个数中的最大数。

```
#include <stdio.h>
int main()
{   int i,n;  float x,max;

    printf("请输入数的个数：");
    scanf("%d", &n);
    printf("请输入%d个数：\n", n);
    max=-1e30;           /* 最大值的初值是尽可能小的数 */
```

```
    for (i=1; i<=n; i++)
    {   scanf("%f", &x);
        if (x>max) max=x;   /* max中是到目前为止的最大值 */
    }
    printf("最大值: %f\n", max);
    return 0;
}
```

运行结果：

```
请输入数的个数: 8
请输入8个数:
71 64 82 77 83 76 95 78
最大值: 95.000000
```

例 3.25 用 for 语句重做例 3.19（输入一行字符，统计并输出字符的个数）。

```
#include <stdio.h>
int main()
{   int n=0;

    printf("请输入一行字符: \n");
    for ( ; getchar()!='\n'; n++)
        ;     /* 循环体是空语句 */
    printf("字符个数为: %d\n", n);
    return 0;
}
```

本例中，省去了 for 语句的表达式 1，表达式 3 也不是用来修改循环变量，而是用作输入字符的计数。这样，就把本应在循环体中完成的计数放在表达式 3 中完成了，因此循环体是空语句。应注意的是，空语句后的分号不可少，如果缺少此分号，则就把后面的 printf 语句当成循环体来执行了。反过来说，如循环体不为空语句时，决不能在表达式 3 的括号后加分号，否则就不能反复执行循环体了。这些都是编程中常见的错误，初学者要特别注意。

3.4.4 循环结构的嵌套

在循环体语句中又包含有另一个完整的循环结构的形式，称为循环结构的嵌套。嵌套在循环体内的循环结构称为内循环，外面的循环结构称为外循环。如果内循环体中又有嵌套的循环语句，则构成多重循环。while、do-while、for 这 3 种循环都可以互相嵌套。如图 3.10 中列出的几种循环结构的嵌套形式都是合法的。

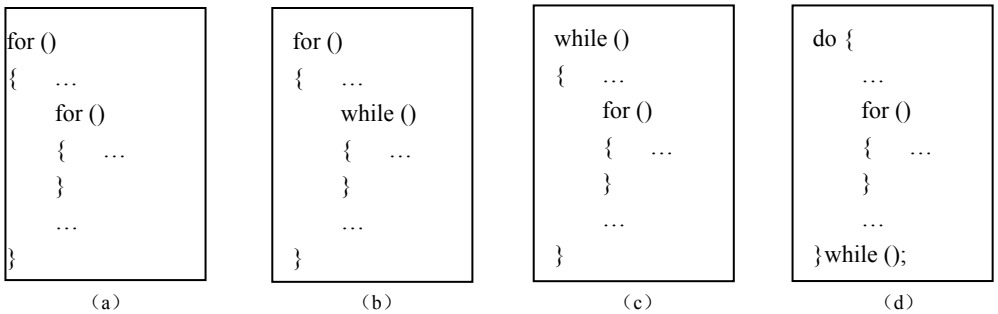

图 3.10 循环结构的嵌套形式

例 3.26　打印下列形式的九九乘法口诀表。

```
1×1= 1
1×2= 2  2×2= 4
1×3= 3  2×3= 6  3×3= 9
1×4= 4  2×4= 8  3×4=12  4×4=16
1×5= 5  2×5=10  3×5=15  4×5=20  5×5=25
1×6= 6  2×6=12  3×6=18  4×6=24  5×6=30  6×6=36
1×7= 7  2×7=14  3×7=21  4×7=28  5×7=35  6×7=42  7×7=49
1×8= 8  2×8=16  3×8=24  4×8=32  5×8=40  6×8=48  7×8=56  8×8=64
1×9= 9  2×9=18  3×9=27  4×9=36  5×9=45  6×9=54  7×9=63  8×9=72  9×9=81
```

分析：这是一个典型的循环嵌套结构的程序，因为要打印 9 行，要一个循环结构，而打印第 *i* 行时要打印 *i* 个等式，又要一个循环结构。

```
#include <stdio.h>
int main()
{   int i,j;

    for (i=1; i<=9; i++)
    {   for (j=1; j<=i; j++)
           printf("%d×%d=%2d  ", j,i,i*j);
        putchar('\n');
    }
    return 0;
}
```

循环嵌套结构的执行过程是这样的：因为内循环是外循环的循环体的一部分，所以外循环的循环控制变量每取一个值，内循环就要从头到尾执行一遍，即内循环的循环控制变量的值要从“初值”变化到“终值”。

思考题：如何修改程序，打印出下列形式的九九乘法口诀表。

```
1×1= 1  1×2= 2  1×3= 3  1×4= 4  1×5= 5  1×6= 6  1×7= 7  1×8= 8  1×9= 9
        2×2= 4  2×3= 6  2×4= 8  2×5=10  2×6=12  2×7=14  2×8=16  2×9=18
                3×3= 9  3×4=12  3×5=15  3×6=18  3×7=21  3×8=24  3×9=27
                        4×4=16  4×5=20  4×6=24  4×7=28  4×8=32  4×9=36
                                5×5=25  5×6=30  5×7=35  5×8=40  5×9=45
                                        6×6=36  6×7=42  6×8=48  6×9=54
                                                7×7=49  7×8=56  7×9=63
                                                        8×8=64  8×9=72
                                                                9×9=81
```

例 3.27　打印出所有的水仙花数。

分析：水仙花数是一个 3 位整数且满足条件“各位数字的立方和等于该数本身”，如 153 等。设变量 *i*、*j* 和 *k* 分别表示 3 位整数的百位、十位和个位，其中 *i* 的取值范围为 1～9，*j* 和 *k* 的取值范围都为 0～9，所有的 3 位整数 *i**100+*j**10+*k* 可以通过 3 重的循环嵌套结构来生成。本例实际上也是穷举法，即每一个 3 位整数都测试一次。

```
#include <stdio.h>
int main()
{   int i,j,k,i3,i100,j3,j10;

    for (i=1; i<10; i++)
    {   i3=i*i*i;  i100=i*100;    /* 引进变量i3和i100可以减少运算量 */
        for (j=0; j<10; j++)
```

```
        {   j3=j*j*j;  j10=j*10;
            for (k=0; k<10; k++)
                if (i3+j3+k*k*k==i100+j10+k)
                    printf("%d  ", i100+j10+k);
        }
    }
    putchar('\n');
    return 0;
}
```

运行结果：

```
153  370  371  407
```

3.4.5 无条件转移语句

无条件转移语句是指当程序执行到该语句时，程序立即转移到程序的其他地方执行，至于转移到什么地方，要看使用何种无条件转移语句。C 语言提供 3 个无条件转移语句：break 语句、continue 语句和 goto 语句。

break 语句主要用于循环结构和 switch 语句结构中，continue 语句主要用于循环结构中。结构化程序设计要求少用或尽量不用 goto 语句，但考虑到有些场合还在应用，本节也作简单介绍。

1．break 语句

break 语句由关键字 break 后加分号“;”组成。其一般形式是：

```
break;
```

在前面讨论 switch 多分支结构时，已经介绍过 break 语句，其是用来跳出 switch 结构，使程序继续执行该结构的下一句语句。当 break 语句用在循环结构中时，被用来跳出循环体，提前结束循环，把程序流程无条件转移到循环结构的下一句语句继续执行。使用 break 语句可以使循环语句有多个出口，在一些场合下使编程更加灵活、方便。需要说明如下：

（1）在循环结构中，break 语句总是与 if 语句一起使用，即满足某条件时便跳出循环。

（2）在循环嵌套结构中，break 语句只能跳出包含它的最里面的那一层循环。

例 3.28　找出 3～100 的全部素数，每输出 10 个素数后换行。

分析：从例 3.22 可知，判断一个数是否是素数要用到循环结构，而现在要找出 3～100 的全部素数又要用到循环结构，所以本题是一个循环嵌套结构。例 3.22 中是用穷举法通过验证 2～m–1 中的数都不是 m 的因子来判断出 m 是素数的。根据分析，完全可以将穷举的范围从 2～m–1 缩小到 2～$\sqrt{m}$。另外，本例中还应设置一个计数器来统计已经输出的素数的个数。

```
#include <stdio.h>
#include <math.h>
int main()
{   int m,k,n,c;

    for (c=0,m=3; m<100; m+=2)          /* 因为偶数不是素数，所以m每次增加2 */
```

```
    {   k=(int)sqrt(m);
        for (n=2; n<=k; n++)
            if (m%n==0) break;              /* 条件成立，m不是素数，退出内循环 */
        if (n>k)
        {   printf("%5d", m);
            c++;                            /* 统计素数个数的计数器加1 */
            if (c%10==0) putchar('\n');     /* 输出10个数后换行 */
        }
    }
    putchar('\n');
    return 0;
}
```

运行结果：

```
    3    5    7   11   13   17   19   23   29   31
   37   41   43   47   53   59   61   67   71   73
   79   83   89   97
```

2．continue 语句

continue 语句由关键字 continue 后加分号“;”组成。其一般形式是：

```
continue;
```

continue 语句只能用在循环体中，该语句的功能是提前结束本次循环，即跳过循环体中 continue 语句之后尚未执行的部分循环体语句，继续进行下一次循环条件的判断。需要说明如下：

（1）continue 语句只能用于 while、do-while 和 for 等循环语句的循环体中，总是与 if 语句一起使用，起到加速循环的作用。

（2）在循环嵌套结构中，continue 语句只能结束包含它的最里面的那一层循环的本次循环。一个较好说明 continue 语句用处的例子是例 4.26。

（3）break 语句和 continue 语句的区别是：break 语句是跳出循环结构并结束整个循环，不再进行循环条件的判断；而 continue 语句只是结束本次循环，继续进行循环条件的判断来决定循环是否继续进行下去。

例 3.29　continue 语句应用示例：求输入的 10 个整数中正数的个数及其平均值。

```
#include <stdio.h>
int main()
{   int i,n,a;  float sum;

    printf("请输入10个整数:\n");
    for (sum=n=i=0; i<10; i++)
    {   scanf("%d", &a);
        if (a<=0)  continue;    /* 如果为负数则结束本次循环 */
        n++;  sum+=a;        /* 对输入的正数计数并求和 */
    }
    printf("%d个正整数的平均值是: %6.2f\n", n,sum/n);
```

```
    return 0;
}
```

运行结果：


```
请输入10个整数:
12 21 -1 -18 30 15 -11 22 -7 -13
5个正整数的平均值是:  20.00
```

3．goto 语句

goto 语句的一般形式是：

```
goto 语句标号;
```

其中的语句标号是一个标识符，这个标识符与冒号（:）一起出现在函数内某语句的前面。goto 语句的作用是程序无条件转移到该语句标号处并执行其后的语句。所以与 goto 语句相对应，程序中必有一个带语句标号的语句，其形式是：

```
语句标号:语句;
```

此语句就是 goto 语句的目标语句。注意目标语句与 goto 语句必须处于同一个函数中，但可以不在一个循环层中，即 goto 语句只能在当前函数内无条件转移，不可以转到本函数以外。

但是，在结构化程序设计中一般不主张使用 goto 语句，以免破坏“单入口、单出口”的结构化程序设计风格，造成程序流程的混乱，不便于对程序的阅读和理解，也不便于程序中错误的排除。过去，goto 语句通常与条件语句配合使用实现条件转移（构成循环、跳出循环体）等功能。现在，有了 break 语句和 continue 语句后，通常只用在从多重循环嵌套的内层循环跳到外层循环的外面时才用到 goto 语句。所以应该少用、慎用 goto 语句，而不是完全禁用。

例 3.30　goto 语句应用示例：用 if…goto 实现求 1～100 之和。

分析：求 1～100 之和完全可以用 for 语句来实现，本例仅仅是为了说明 goto 语句的用法，并不是说以后在编写程序时要用 goto 语句去实现循环结构。

```
#include <stdio.h>
int main()
{   int i,sum;

    sum=0;  i=1;
  loop:     /* 语句标号 */
    if (i<=100)
    {   sum+=i;  i++;
        goto loop;
    }
    printf("sum=%d\n", sum);
    return 0;
}
```

运行结果：

```
sum=5050
```

思考题：如何用 goto 语句修改例 3.27，使得程序找出第一个水仙花数后就结束。

3.4.6 循环程序设计方法举例

数学上的“递推”方法在程序设计中应用很多，递推方法解题的关键是确定递推公式和初始条件。根据初始条件的不同，递推又可分为“顺推”和“倒推”两种形式。所谓顺推是数列后面的值要根据前面的值才能推算出来; 所谓倒推是数列前面的值要通过后面的值才能推算出来。

1．顺推法

有一类问题是求解一系列数据，或者是求解一系列数据的最终结果，而该系列数据中的相邻数据的变化有一定的规律性。问题的模型可以抽象成递推公式 $x_n=f(x_{n-1})$，就是从前向后推出系列数据中的每一项，所以称为“顺推”。

最基本的数据累加、累乘算法就是顺推算法的典型例子，累加过程的递推公式是 $s_n=s_{n-1}+a_n$，其中 s_{n-1} 表示前 $n-1$ 项的和，a_n 表示第 n 个累加对象。当只需要累加的最终结果，而不需要累加过程中的部分和时，递推公式在程序中可表示为 $s=s+a_n$ 。

例 3.31 利用公式 $\pi/4\approx1-1/3+1/5-1/7+\cdots$ 的前 10^6 项，求π的近似值。

分析：在例 3.20 中 a_n 就是输入的成绩 score，本题的关键是如何得到 a_n。很显然分母是一个个奇数（最大值为 $2\times10^6-1$），而分子是 1 和-1 交叉出现，本例中通过引进变量 sign 和语句 sign = -sign;很巧妙地解决了这一问题。

```
#include <stdio.h>
int main()
{   double pi,t,sign;
    long n;

    pi=0.0;
    for (sign=n=1; n<2000000; n+=2)
    {   t = sign/n;    /* sign不能为int类型，否则t总为0 */
        pi += t;
        sign = -sign;
    }
    printf("Л=%lf\n", pi*4);
    return 0;
}
```

运行结果：

```
Л=3.141592
```

例 3.32 兔子繁殖问题。著名的意大利数学家 Fibonacci 曾提出一个有趣的问题：有一对新生兔子，从第三个月开始繁殖，每个月都生产一对小兔子。小兔子到第三个月又开始繁殖，每个月都生产一对下一代小兔子。按此规律，在没有兔子死亡的情况下，那么一年

后共有多少对兔子？

分析：显然第 1 个月和第 2 个月都只有一对兔子，从第 3 个月开始的兔子数是上个月和上上个月的兔子数之和。这是因为，在没有兔子死亡的情况下，当月的兔子数由两部分组成：上个月的老兔子和当月出生的小兔子。而当月出生的小兔子数恰好是上上个月的兔子数，因为上个月的兔子中还有部分在这个月不能生育，只有上上个月的兔子才能每对生产一对小兔子。数学模型为：

$$fib_1=fib_2=1 \qquad (n=1,2) \quad 初始值$$

$$fib_n=fib_{n-1}+fib_{n-2} \quad (n\geqslant 3) \quad 递推公式$$

这就是著名的 Fibonacci 数列。顺便说一下，当 n 趋向于∞时，数列前后两项的商（fib_{n-1}/fib_n）趋向于黄金分割数 0.618 。

```
#include <stdio.h>
int main()
{   int n;  long fib1,fib2,fib3;

    fib1=fib2=1;
    printf("%6ld%6ld", fib1,fib2);
    for (n=3; n<=12; n++)
    {   fib3=fib1+fib2;
        printf("%6ld", fib3);
        if(n%6==0)printf("\n");
        fib1=fib2;
        fib2=fib3;
    }
    return 0;
}
```

另一种可行的方法，请读者自行分析其正确性。

```
for (n=2; n<=6; n++)
{   fib1=fib1+fib2;
    fib2=fib2+fib1;
    printf("%6ld%6ld", fib1,fib2);
    if (n%3==0)  printf("\n");
}
```

运行结果：

```
     1     1     2     3     5     8
    13    21    34    55    89   144
```

2．倒推法

所谓“倒推法”就是从后向前递推，来求解问题的初始数据，即由结果倒过来推解它的前提条件。

例 3.33　猴子吃桃问题。猴子第 1 天摘下若干个桃子, 当即吃了一半, 还不过瘾, 又多吃了一个。第 2 天又将剩下的桃子吃掉一半, 又多吃了一个。以后每天都吃前一天剩下的一半多一个，到第 10 天想吃时, 只剩一个桃子了。问第 1 天有多少个桃子？

分析：设第 n 天有桃子 x_n 个，显然有 $x_n=x_{n-1}/2-1$，即 $x_{n-1}=2(x_n+1)$。这样就可以用倒推法从第 10 天的 1 个桃子算出第 9 天的桃子数，再算出第 8 天的桃子数，直到算出第 1 天的桃子数。由于求出 x_{n-1} 后 x_n 就没有用了，所以程序中 x_{n-1} 和 x_n 可以用同一个变量 x，即 $x=2*(x+1)$。因为 $x=2*(x+1)$至少要做一次，所以适合用 do–while 语句来实现循环。

```
#include <stdio.h>
int main()
```

```
{   int n,x;

    n=10;  x=1;
    do {
        x=2*(x+1);
        n--;
    }while (n>1);
    printf("第1天有%d个桃子", x);
    return 0;
}
```

运行结果：

```
第1天有1534个桃子
```

3．迭代法

“迭代”法也称“辗转”法，是一种不断用变量的旧值递推新值的解决问题的方法，是递推方法的一个特例。在递推过程中前后数值的逻辑意义有顺序上的不同，而迭代过程中新旧值的逻辑意义一般是完全相同的。迭代法可分为“精确迭代”和“近似迭代”，下面的例 3.34 属于精确迭代，而例 3.35 属于近似迭代。

例 3.34　求两个正整数的最大公约数。

分析：求两个正整数的最大公约数有许多种方法。

（1）“更相减损之术”方法是我国古代数学家对公约数求解问题进行了研究并提出的算法。该方法是以两数中较大的数减去较小的数，得到的差与原来较小的数构成新的一对数，再以较大的数减去较小的数，如此进行下去，直到产生一对相等的数，该数即为最大公约数。例如，求 12 和 16 的最大公约数，具体过程为：(12,16)→(12,4)→(8,4)→(4,4)，所以 4 是 12 和 16 的最大公约数。

（2）“辗转相除法”方法是古希腊数学家对公约数求解问题研究后提出的算法。该方法是以两数中较大的数除以较小的数，得到的余数与原来较小的数构成新的一对数，再以较大的数除以较小的数，如此进行下去，直到余数为 0 为止，则较小的数就是最大公约数。例如，求 288 和 123 的最大公约数，具体过程为：(288,123)→(42,123)→(42,39)→(3,39)，所以 3 是 288 和 123 的最大公约数。

设两个整数为 *a* 和 *b*，则运算过程如下：

① *a* 除以 *b* 取余得 *c*，若 *c*=0，则 *b* 为两数的最大公约数，并退出循环。

② 若 *c*≠0，则 *a*=*b*，*b*=*c* 再回去执行①。

```
#include <stdio.h>
int main()
{   int a,b,c;
    /* 用辗转相除法求两个正整数的最大公约数 */
    printf("请输入两个正整数: ");
    scanf("%d%d", &a,&b);
    if (a<b) {  c=a;  a=b;  b=c;  }   /* 本句可以省略，无非是下面的循环多做一次 */
    while ( (c=a%b)!=0 )              /* 同例3.19，充分利用赋值运算符简化程序 */
```

```
        {  a=b;  b=c;  }
    printf("最大公约数是: %d\n", b);
    return 0;
}
```

运行结果：

```
请输入两个正整数: 123 288
最大公约数是: 3
```

例 3.35 用牛顿迭代法 $x_{n+1}=x_n-f(x_n)/f'(x_n)$求方程 $2x^3-4x^2+3x-6=0$ 在 1.5 附近的根，直到前后两项差的绝对值小于 10^{-6} 为止。

分析：一般来说，求一元五次（或更高次）方程的根或微分方程的数值解等计算问题，很难或无法用像一元二次方程的求根公式那样的解析法去求解，于是发明了很多数值计算方法（简称数值法）来求出问题的近似解，牛顿迭代法就是方程求根的一种数值计算方法。

牛顿迭代法要求方程 $f(x)=0$ 中的函数 $f(x)$是可导的。假定方程 $f(x)=0$ 有一个根 r，牛顿迭代法求根时首先估计一个初始值 x_0（可以是猜测的），然后通过迭代公式求出下一个估计值 x_1，只要 x_1 没有满足精度要求，就继续用迭代公式求出下一个估计值 x_2，…，直到 x_{n+1} 非常接近根 r（如满足 x_{n+1} 和 x_n 差的绝对值小于 10^{-6}）为止。显然，本例适合用 do-while 语句来实现循环，程序中用变量 $x1$ 和 $x0$ 分别表示迭代公式中的 x_{n+1} 和 x_n。

```
#include <stdio.h>
#include <math.h>
int main()
{   double x0,x1,f,fd;
    /* 考虑到下面的x0=x1;语句，这里不是赋值给x0，而是x1 */
    x1=1.5;
    do {
        x0=x1;
        f=((2*x0-4)*x0+3)*x0-6;        /* 这里采用了秦九韶算法求多项式的值 */
        fd=(6*x0-8)*x0+3;              /* 秦九韶是我国南宋时期的数学家 */
        x1=x0-f/fd;
    }while(fabs(x1-x0)>=1e-6);
    printf("The root is %lf\n", x1);
    return 0;
}
```

运行结果：

```
The root is 2.000000
```

4．穷举法

穷举法是另一种常用的问题求解方法，前面的判断素数和找水仙花数问题中已经使用过该方法。其基本思想是：对问题的所有可能答案进行一一测试，直到找到正确答案或测试完全部可能答案。穷举法对于人来说是一件单调而烦琐，甚至可能无法完成的工作，但对于计算机来说，重复性的工作非常适合用循环解决，并能发挥计算机高速运算的优势。当然也不能让计算机做无用功，应当排除那些不合理的情况，尽可能减少问题可能解的数目。

例 3.36　搬砖问题（中国古典算术问题）。某工地有 45 块砖需要搬运，已知男人一次搬 3 块，女人一次搬 2 块，两个小孩一次抬 1 块砖。要求 45 人一次搬完所有的 45 块砖，问男人、女人、小孩各几人？

分析：这是一个组合问题，由男人、女人和小孩的人数决定组合的数量。设 men 为男人数，women 为女人数，children 为小孩数，根据题意，求解该题必须使下面的不定方程组成立：

men + women + children = 45

3×men + 2×women + children/2 = 45

进一步分析该题中的条件，可以确定 3 个变量的取值范围（穷举算法这一步不可忽视，其可以大大降低循环次数，从而提高程序执行效率）。

- men 的取值范围为 0～15；
- women 的取值范围为 0～22；
- children 的人数应为 45-men-women，因为两个小孩抬 1 块砖，所以小孩数必须为偶数。

程序如下：

```
#include <stdio.h>
int main()
{   int men,women,children;

    printf("男人  女人  小孩\n");
    for (men=0; men<=15; men++)
    {   for (women=0; women<=22; women++)
        {   children = 45-women-men;
            if (3*men+2*women+children/2==45 && children%2==0)
                printf("%4d  %4d  %4d\n", men,women,children);
        }
    }
    return 0;
}
```

运行结果

```
男人  女人  小孩
   0    15    30
   3    10    32
   6     5    34
   9     0    36
```

最后要说明的是，本例中 if 语句的条件也可以改写为：6*men+4*women+children==90，原来的条件 children%2==0 可以不要了，因为上述条件成立，children 必为偶数，这样可进一步减少判断的运算量。

练　习　3

一、选择题

1．以下选项中，不是 C 语言语句的是（　　）。

A．{ int i;　i++;　printf("%d\n", i); }　　B．　;

C．a=5,c=10　　D．{　;　}

2. 执行以下程序时，若输入 1234567↙（这里↙表示回车），程序的输出结果为（　　）。

```
#include <stdio.h>
int main()
{   int x,y;
    scanf("%2d%2d",&x,&y);
    printf("%d\n",x+y);
}
```

A．17　　B．46　　C．15　　D．9

3．若有定义：“int a, b;”，则用语句“scanf("%d%d", &a,&b);”输入 *a* 和 *b* 的值时，不能作为输入数据分隔符的是（　　）。

A．,　　B．空格　　C．回车　　D．Tab 键

4．语句“printf("%f", 2.5+1*7%2/4);”的输出结果是（　　）。

A．2.500000　　B．2.750000　　C．3.375000　　D．3.000000

5．若有定义：“float f1, f2;”，假定数据的输入方式为 4.52□3.5↙（这里□代表空格），则正确的输入语句是（　　）。

A．scanf("%f,%f", &f1,&f2);　　B．scanf("%f%f", &f1,&f2);

C．scanf("%3.2f%2.1f", &f1,&f2);　　D．scanf("%3.2f,%2.1f", &f1,&f2);

6．以下选项中，不合法的 C 语言表达式是（　　）。

A．x+1=x+1　　B．0<=x<100　　C．i=j==0　　D．(char)(65+3)

7．以下选项中，能正确表示 a≥10 或 a≤0 的关系表达式是（　　）。

A．a>=10 or a<=0　B．a>=10 | a<=0　C．a>=10 && a<=0　D．a>=10 || a<=0

8．以下选项中，能正确表示 *a* 和 *b* 同时为正数或同时为负数的表达式是（　　）。

A．(a>=0||b>=0) && (a<0||b<0)　　B．(a>=0 && b>=0) && (a<0 && b<0)

C．(a+b>0) && (a+b<=0)　　D．a*b>0

9．若有定义：“int x=3, y=1;”，则表达式(!x || y--)的值是（　　）。

A．0　　B．1　　C．2　　D．−1

10．若有定义：“int a=−2, b=−1, c=0, m=2, n=2;”，则计算逻辑表达式(m=a<b<c) && (++n)后，*n* 的值为（　　）。

A．0　　B．1　　C．2　　D．3

11．若有定义：“int a, b, c=246;”，则依次执行语句“a=c/100%9;　b=(−1) && (−1);”后，*a* 和 *b* 的值分别为（　　）。

A．2 和 1　　B．3 和 2　　C．4 和 3　　D．2 和−1

12．假定所有变量均已正确定义，下列程序段运行后，*x* 的值是（　　）。

```
a=b=c=0;  x=35;
if (!a) x--;
else if (b) ;
```

```
if (c) x=3;
else x=4;
```

A．34　　B．4　　C．35　　D．3

13．若有定义：“int a=1,b=3,c=5,d=5,x;”，下列程序段运行后，*x* 的值是（　　）。

```
if (a<b)
   if (c<d) x=1;
   else if (a<c)
          if (b<d) x=2;
          else  x=3;
       else  x=6;
else  x=7;
```

A．1　　B．2　　C．3　　D．6

14．若有定义：“int a=−1, b=1;”，则执行以下 if 语句后的输出结果为（　　）。

```
if ( (++a<0) && !(b--<=0) )
   printf("%d%d\n",a,b);
else
   printf("%d%d\n",b,a);
```

A．–11　　B．01　　C．10　　D．00

15．若有定义：“int a=12, b=5, c=−3;”，则执行以下程序段后的输出结果为（　　）。

```
if (a>b)
if (b<0) c=0;
else  c++;
printf("%d\n", c);
```

A．0　　B．1　　C．−2　　D．−3

16. 如果两次运行下列程序时，从键盘上分别输入 6 和 4，则输出的结果分别是(　　)。

```
#include <stdio.h>
int main()
{   int x;
   scanf("%d", &x);
   if (x++>5) printf("%d", x);
   else printf("%d\n", x--);
}
```

A．7 和 5　　B．6 和 3　　C．7 和 4　　D．6 和 4

17．下列关于 switch 语句和 break 语句的结论中，正确的是（　　）。

A．break 语句是 switch 语句中的一部分

B．在 switch 语句中可以根据需要使用或不使用 break 语句

C．在 switch 语句中必须使用 break 语句

D．break 语句只能用于 switch 语句中

18．若有定义：“int a=1,b=0;”，则执行以下 switch 语句后的输出结果为（　　）。

```
switch(a)
{   case 1: switch(b)
          {   case 0: printf("**0**");  break;
              case 1: printf("**1**");  break;
          }
    case 2: printf("**2**");  break;
}
```

A．**0**　　B．**0****2**

C．**0****1****2**　　D．有语法错误

19．若有定义：“int x=1, a=0, b=0;”，则执行以下程序段后的输出结果为（　　）。

```
switch(x)
{   case 0: b++;
    case 1: a++;
    case 2: a++;  b++;
}
printf("a=%d,b=%d\n",a,b);
```

A．a=2,b=1　　B．a=1,b=1　　C．a=1,b=0　　D．a=2,b=2

20．设 *w*、*x*、*y*、*z*、*m* 均为 int 型变量，则以下程序段运行后，*m* 的值是（　　）。

```
w=1;  x=2;  y=3;  z=4;
m=(w<x)?w:x;  m=(m<y)?m:y;  m=(m<z)?m:z;
```

A．4　　B．3　　C．2　　D．1

21．若有定义：“int a=5,b=4,c=6,d;”，则语句“printf("%d\n", d=a>b?(a>c?a:c):(b));”的输出结果是（　　）。

A．5　　B．4　　C．6　　D．不确定

22．在语句“while (x) ;”中的条件 *x* 等价于（　　）。

A．x==0　　B．x==1　　C．x!=1　　D．x!=0

23．在语句“while (!e) ;”中的条件!e 等价于（　　）。

A．e==0　　B．e!=1　　C．e!=0　　D．e==1

24．以下程序段运行后的输出结果是（　　）。

```
int n=9;
while (n>6)
   { n--;  printf("%d",n); }
```

A．987　　B．876　　C．8765　　D．9876

25．以下程序段中 while 循环执行的次数是（　　）。

```
int  k=0;
while (k=1)
```

```
    k++;
```

A．无限次　　B．执行一次

C．一次也不执行　　D．有语法错，不能执行

26．以下程序段运行后的输出结果是（　　）。

```
int a=1, b=2, c=2, t;
while (a<b<c)
    { t=a; a=b; b=t; c--; }
printf ("%d,%d,%d", a,b,c);
```

A．1,2,0　　B．2,1,0　　C．1,2,1　　D．2,1,1

27．以下程序段运行后的输出结果是（　　）。

```
int x=23;
do
{   printf("%d",x--);
}while(!x);
```

A．321　　B．23　　C．没有任何内容　　D．陷入死循环

28．以下程序段中 for 循环的执行次数是（　　）。

```
for (x=0,y=0; (y!=123) && (x<4); x++ ) ;
```

A．是无限循环　　B．循环次数不定　　C．循环执行 4 次　　D．循环执行 3 次

29．以下程序段运行后的输出结果是（　　）。

```
int x=10, y=10, i;
for (i=0; x>8; y=++i)
    printf("%d %d ", x--,y);
```

A．10 1 9 2　　B．9 8 7 6　　C．10 9 9 0　　D．10 10 9 1

30．以下程序段中 for 循环体的执行次数是（　　）。

```
int i, j;
for (i=0,j=1; i<=j+1; i+=2,j--)
    printf("%d\n",i);
```

A．3　　B．2　　C．1　　D．0

31．以下程序的输出结果是（　　）。

```
#include <stdio.h>
int main()
{   int k,j,m;
    for (k=5; k>=1; k--)
    {   m=0;
        for (j=k; j<=5; j++)
            m=m+k*j;
```

```
    }
    printf ("%d\n",m);
}
```


A．124　　B．25　　C．36　　D．15

32．以下程序中，while 循环的循环次数是（　　）。

```
#include <stdio.h>
int main()
{   int  i=0;
    while (i<10)
    {  if (i<1) continue;
       if (i==5) break;
       i++;
    }
}
```

A．1　　B．10　　C．6　　D．死循环

33．以下程序的输出结果是（　　）。

```
#include<stdio.h>
int main()
{   int i=0,a=0;
    while (i<20)
    {   for ( ; ; )
        {   if ( (i%10)==0 ) break;
            else i--;
        }
        i+=11;  a+=i;
    }
    printf("%d\n",a);
}
```

A．21　　B．32　　C．33　　D．11

34．若有定义：“int n;”，则以下选项中构成死循环的是（　　）。

A．n=100;　do { n++; }while (n>100);

B．for (n=100 ;　; n=n%100+1)　if (n>100) break;

C．n=100;　while (n) --n;

D．n=100;　while (n--) ;

二、填空题

1．若执行以下程序段时，从键盘输入 10□20□30□40↙，则输出结果是_____。

```
int a1,a2,a3;
scanf("%d%*d%d%d",&a1,&a2,&a3);
printf("%d  %d  %d",a1,a2,a3);
```

2．若执行以下程序段时，从键盘输入 100↙，则输出结果是_____和_____。

```
int m;
printf("输入一个整数:");
scanf("%d",&m);
printf("%d(10)<=>%o(8)\n",m,m);
printf("%d(10)<=>%x(16)\n",m,m);
```

3．若有定义：“int a=10,b=9,c=8;”，则依次执行下列语句后，变量 *c* 中的值是_____。

```
c=(a-=(b-5));    c=(a%11)+(b=3);
```

4．表示“整数 x 的绝对值大于 5”时，值为“真”的 C 语言表达式是_____。

5．设 *x*、*y*、*z* 均为 int 型变量，请写出描述“*x* 或 *y* 中至少有一个小于 *z*”的表达式_____。

6．能正确表示 20<*x*<30 或 *x*<−100 的 C 语言关系表达式是_____。

7．已知 *a*=7.5, *b*=2, *c*=3.6，表达式 a>b && c>a || a<b && !c>b 的值是_____。

8．若有定义：“int a=5, b=4, c=3, d;”，则依次执行下列语句后的输出结果是_____。

```
d=(a>b>c);    printf("%d\n",d);
```

9．若运行以下程序段时，从键盘输入 58↙，则输出结果是_____。

```
int a;
scanf("%d", &a);
if (a>50) printf("%d", a);
if (a>40) printf("%d", a);
if (a>30) printf("%d", a);
```

10．若运行以下程序段时，从键盘输入 12↙，则输出结果是_____。

```
int x,y;
scanf("%d", &x);
y=x>12?x+10:x-12;
printf("%d\n",y);
```

11．若有定义：“int n=10;”，则运行以下程序段后，变量 *n* 的值是_____。

```
switch(n)
{   case 9: n++;
    case 10: n++;
    case 11: n++;
    default: n++;
}
```

12．若运行以下程序段时，从键盘输入 Y?N?↙，则输出结果为_____。

```
char c;
while ((c=getchar())!='?')
    putchar(--c);
```

13．以下程序段的输出结果是_____。

```
int x=2;
while (x--) ;
printf("%d\n", x);
```

14．以下程序段的输出结果是_____。

```
int a=2, s=0, n=1, count=1;
while (count<=7)
    { n=n*a;  s=s+n;  ++count; }
printf("s=%d",s);
```

15．以下程序段的输出结果是_____。

```
int x=2;
do {
   printf("*");  x--;
}while(x);
```

16．以下程序段中循环体的执行次数是_____。

```
int a=10, b=0;
do {
   b+=2;  a-=2+b;
}while(a>=0);
```

17．以下程序段的输出结果是_____。

```
int i=10, s=0;
do {
   s=s+i;
   i--;
}while(i>2);
printf("%d\n",s);
```

18．以下程序的功能是：计算 1~10 之间的奇数之和与偶数之和，请填空。

```
#include <stdio.h>
int main()
{   int a, b, c, i;
    a=c=0;
    for (i=0; i<=10; i+=2)
    {   a+=i;
        _____;
        c+=b;
    }
    printf("偶数之和=%d\n",a);
    printf("奇数之和=%d\n",c-11);
}
```

19．以下程序的输出结果是_____。

```
#include <stdio.h>
int main()
{   int y,a;
    y=2;  a=1;
    while (y--!=-1)
    {   do {
           a*=y;  a++;
        }while (y--);
    }
    printf("%d,%d",a,y);
}
```

20．以下程序的功能是：输出 100 以内能被 3 整除且个位数为 6 的所有整数，请填空。

```
#include <stdio.h>
int main()
{   int  i, j;
    for ( i=0; _______; i++)
    {   j=i*10+6;
        if (_____)  continue;
        printf("%d",j);
    }
}
```

三、思考题

1．什么是结构化程序设计方法？结构化程序设计应遵循哪些原则？

2．C 语言的语句有哪几类？为什么说 C 语言是表达式语言？

3．C 语言中如何表示逻辑值“真”和“假”？一个值参加逻辑运算时又如何判断其是“真”还是“假”？

4．设有变量定义：“int a=3, b=8;”，表达式 (a<5) || (b=5) 求值后，*b* 的值是多少？

5．设有 *n*(*n*>0)个学生要分班，每班 *k*(*k*>0)个学生，最后不足 *k* 个学生也编一班，试用条件表达式表示班级数。

6．请消除下列程序中的两个语法错误。

```
#include <stdio.h>
int main()
{   int a,b;  float x=5.7;

    a=10;
    scanf("%d", b);
    printf("a+b=%d\n", a+b);
    printf("x=%d\n", x);
}
```

7．如果执行下列程序时，数据的输入格式如下，请解释程序的运行结果。为了使得 c1='A'和 c2='a'，假定程序不做修改，应如何修改数据的输入格式？假定数据的输入格式不变，又应如何修改程序？从中体会一般情况下 scanf 的格式控制字符串中的普通字符不但起不到提示的作用，反而给用户的输入制造了麻烦。

```
#include <stdio.h>
int main()
{   int  a, b;  float  x,y;  char  c1,c2;

    scanf("a=%d  b=%d", &a,&b);
    scanf("%f%f", &x,&y);
    scanf("%c%c", &c1,&c2);

    printf("a=%d□b=%d\n", a,b);    /* □表示空格 */
    printf("x=%0.1f□y=%0.1f\n", x,y);
    printf("c1=%c□c2=%c\n", c1,c2);
}
```

输入为：

```
a=3□b=7↙     （ ↙表示回车 ）
8.5□6.1↙
A□a↙
```

输出为：

```
a=3□b=7
x=8.5□y=6.1
c1=
□c2=A
```

四、编程题

1．编写一个程序，输入一个华氏温度 *f*，输出对应的摄氏温度 *c*。计算公式是：*c*=5/9*(*f*–32) 。

2．假定某种手机套餐规定：月租费 10 元，可免费发送短信 60 条，超出部分每条 0.10 元；可免费与本地手机通话 20 分钟（包括打入与打出），超出部分每分钟 0.15 元；与本地固定电话通话可享受每分钟 0.2 元的优惠（没有免费通话时间）。在不考虑长途通话的情况下，输入某用户一个月发送短信的条数、与本地手机通话的分钟数和与本地固定电话通话的分钟数，编写程序计算并输出该用户这个月的手机通信费用。

3．编写一个程序，输入两个两位正整数，它们各位上的数字互不相等，程序判断这两个两位数的乘积是否等于把它们各自位上的数字交换后所得的新的两位数的乘积，并输出相应的信息。

```
输入为：12□63↙
输出为：12*63=21*36
输入为：12□34↙
```

输出为：12*34<>21*43

4．编写一个程序，输入某乘客先后两次乘坐公交车的上车时间（假定在同一天内），程序判断时间间隔是否大于 2 小时并输出相应的信息。输入格式以及输出信息的格式如下所示，程序不考虑输入时间错误（如 8:68:72）。提示：可将时间转换为以秒为单位后再做减法。

输入为：9:31:4□11:8:25↙
输出为：时间间隔=01:37:21　你能享受公交优惠1元。
输入为：12:58:37□15:2:49↙
输出为：时间间隔=02:04:12　对不起，你不能享受公交优惠。

5．编写一个程序，输入一个三角形的 3 条边的边长，程序判断其是否能够构成一个三角形，能则输出三角形的面积，否则输出信息“不能构成三角形”。三角形面积计算公式：面积=$\sqrt{s(s-a)(s-b)(s-c)}$，其中，a、b、c 为三角形的 3 条边长，$s=(a+b+c)/2$ 。

6．计算个人所得税征税问题。应纳税工资=工资总额−3500 元，应纳税工资在 1500 元以下（含 1500 元，以下同）的税率为 3%；1500 元以上 4500 元以下的部分税率为 10%；4500 元以上 9000 元以下的部分税率为 20%；9000 元以上 35 000 元以下的部分税率为 25%。输入工资总额（假定小于 38 500 元），编写程序计算并输出个人所得税。个税计算公式如下：

个税＝应纳税工资×对应的税率−速算扣除数

全月应纳税工资	税率%	速算扣除数（元）
不超过 1500 元	3	0
超过 1500 元至 4500 元	10	105
超过 4500 元至 9000 元	20	555
超过 9000 元至 35 000 元	25	1005

7．假定银行整存整取存款的存期有 1、2、3、5 年 4 种，年利率分别为 3.5%、4.4%、5%、6.5%。输入存款的本金和存期（若输入的存期错误，应报错），编写程序计算并输出到期后的利息（利息=本金×存期×年利率）。

8．编写一个程序，输入年份和月份，输出该月的天数。

9．编写一个程序，输入一个正整数，要求以相反的顺序输出该数。例如，输入 12345，则输出为 54321。

10．编写一个程序，输入一行字符，输出其中的英文字母、数字字符和其他字符各有多少个。

11．编写一个程序，用 e≈1+1/1!+1/2!+1/3!+……+1/n!，求 e 的近似值，直到 1/n!<10^{-6} 为止。

12．编写一个程序，用迭代法求数 a 的平方根，迭代公式为：$x_{n+1}=(x_n+a/x_n)/2$，要求前后两次求出的 x 的差的绝对值小于 10^{-5} 。数 a 可从键盘上输入，x_0 可取 $a/2$ 。

13．假定某一大型比赛中有 10 名裁判同时为一名体操运动员打分，编写一个程序，输入这 10 名裁判的打分，输出去掉一个最高分和一个最低分后该运动员的平均得分。

14．编写一个程序，输入 10 个互不相同的数，输出其中的最大数和次大数。

15.编写一个程序，找出连续整数之和是 500 的所有整数序列，如 500=98+99+100+101+102 。

16．“百钱百鸡”问题。我国古代数学家张丘建在《算经》中出了一道题：“鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。百钱买百鸡，问鸡翁、母、雏各几何？”意思是：公鸡 5 元钱 1 只，母鸡 3 元钱 1 只，小鸡 1 元钱 3 只。用 100 元钱买 100 只鸡，问公鸡、母鸡和小鸡各买多少只？编写程序解此题。

17．爱因斯坦阶梯问题。设有一个阶梯，每步跨 2 阶，最后余 1 阶；每步跨 3 阶，最后余 2 阶；每步跨 5 阶，最后余 4 阶；每步跨 6 阶，最后余 5 阶；每步跨 7 阶，正好全部跨完。问该阶梯至少有几阶？编写程序解此题。提示：设 ladders 表示阶梯数，根据题意，可知①ladders 为奇数；②ladders 的起始值为 7 且为 7 的整数倍；综合①②可知③ladders 取值的步长为 14。

18．假定某学校有近千名学生在操场上排队，7 人一行多 3 人，5 人一行多 2 人，3 人一行多 1 人。问该校有多少名学生？编写程序解此题。

19．趣味填空题。给出用等号连接的两个正整数，如“1234＝127”，当然，这个等式是不成立的。现在让你在左边的整数中间某个位置插入一个加号，看是否有可能让等式成立。如上面的式子如果写成 123+4=127，等式就成立了。编写程序解此题，若不可能让等式成立，则输出“Impossible!”。

20．数学黑洞问题。这个名词是由一个数学游戏开始的，随机写出一个三位正整数（要求三位数字不完全相同），然后按照数字从大到小顺序，把三位数字重新排列得到一个新数，接下来，再把所得的数字顺序颠倒一下，又得到一个新数，把两个新数的差作为一个新的三位数，再重复上面的步骤，不停地重复下去。例如：

476 变 764 和 467　　764–467=297

297 变 972 和 279　　972–279=693

693 变 963 和 369　　963–369=594

594 变 954 和 459　　954–459=495

有趣的是，经过几次迭代之后，三位数最后都会停在 495 这个数上。编写一个程序输入一个三位数（要求三位数字不完全相同），输出最后达到 495 的过程（输出格式同上面的举例）。