高等学校计算机应用规划教材

C 语言编程技巧分析

高　禹　主　编

张建科　乐　天　副主编

侯志凌　李　鑫

清华大学出版社

北　京

内 容 简 介

本书详细分析了 C 语言的编程技巧问题。本书分为 10 章，每章分析一个知识模块的编程技巧问题。各章首先介绍该章知识模块的要点，然后介绍运用该章知识时需要注意的问题，着重分析运用该章知识的编程技巧。第 1 章分析基础知识编程技巧，第 2 章分析选择结构编程技巧，第 3 章分析循环结构编程技巧，第 4 章分析数组知识编程技巧，第 5 章分析函数知识编程技巧，第 6 章分析预处理命令编程技巧，第 7 章分析指针知识编程技巧，第 8 章分析结构体、共用体和枚举类型以及链表知识编程技巧，第 9 章分析位运算知识编程技巧，第 10 章分析文件知识编程技巧。在每章的编程技巧分析部分，结合实例讲解与该章知识相关的编程技巧。对于每章知识所涉及到的典型问题的算法，通过实例进行了细致的分析。

本书的适用对象是具有初步的 C 语言知识的读者。本书既可作为高等院校的教材，也可供自学者及参加各级各类 C 语言考试者阅读使用，还可供使用 C 语言编程的人员及编程爱好者使用。

本书的电子教案和实例源文件可以到 http://www.tupwk.com.cn/downpage 网站下载。

图书在版编目(CIP)数据

C 语言编程技巧分析 / 高禹等 主编. —北京：清华大学出版社，2014

(高等学校计算机应用规划教材)

ISBN 978-7-302-34400-1

Ⅰ. ①C… Ⅱ. ①高… Ⅲ. ①C 语言—程序设计—高等学校—教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2013)第 260107 号

责任编辑：胡辰浩
装帧设计：孔祥峰
责任校对：曹　阳
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62794504

印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**15.5　　**字　　数：**358 千字
版　　次：2014 年 1 月第 1 版　　**印　　次：**2014 年 1 月第 1 次印刷
印　　数：1～3500
定　　价：29.00 元

产品编号：

前　言

C 语言是一种重要的计算机语言，它功能丰富、灵活性强、可移植性好、语言简洁，因而深受广大用户的喜爱。

许多人在初步学习了 C 语言之后，虽然掌握了 C 语言的基础知识和基本编程方法，但是，当动手解决实际问题时，会深切地感觉到：编程能力严重不足！特别是对于一些编程技巧(包括一些解决重要问题的典型算法)的运用力不从心。甚至有的人不清楚常用的重要编程技巧。

对于初步学习过 C 语言的人，为了促进他们在编程技巧方面的能力提高，我们编写了本书。参加本书编写的作者都是长期从事 C 语言教学工作以及运用 C 语言开发软件的教师，他们熟悉 C 语言的各种编程技巧，清楚提高初学者能力的教学方法。

为了达到促使读者在编程技巧方面能力提高的目的，我们在本书内容的设计上作了精心的思考。

根据 C 语言的知识结构，本书分为 10 章，每章分析一个知识模块的编程技巧问题。各章首先介绍该章知识模块的要点，然后介绍运用该章知识时需要注意的问题，着重分析运用该章知识的编程技巧，最后安排了一些习题。

在每章的编程技巧分析部分，我们精心挑选了许多实例。结合实例，讲解与该章知识相关的编程技巧。对于每个实例，我们首先给出利用该章知识与相关技巧所编写的程序，然后分析该程序的编写思路和所运用的技巧。对于每章知识所涉及到的典型问题的算法，我们通过实例进行了细致的分析。

每章在对某些实例进行分析之后，为了扩展读者的思路，我们提出了思考的问题，将对应的程序留给读者编写。相信读者经过思考将程序编写成功后，能力会有更大的提高。

在每章的习题部分，我们安排了不同类型的习题，以便读者深入理解该章知识与相关编程技巧。读者可以将其作为该章内容的必要的补充部分。对于每一道习题，为了方便读者学习该章知识与相关的编程技巧，我们给出了参考程序。

学习程序设计语言并提高编程能力的必要途径是实践，所以我们对应书中每章内容安排了上机实验。上机实验时，除了调试该部分安排的实验内容外，也应该调试书中例题和习题的程序。相信读者经过实践，能够提高动手能力和编程水平。

本书条理清楚、语言流畅、通俗易懂，实用性强。本书的适用对象是具有初步的 C 语言知识的读者，帮助他们提高编程能力。

除了主编和副主编外，参加本书编写的还有王广伟、亓常松、李慧、江有福、刘军、毕振波、杨永华、陈荣品、陈洪涛、吴远红、张威、张艳艳、黄海锋、崔振东、管林挺、谭小球、潘洪军等人。由于编者水平有限，书中难免存在错误与不足之处，诚恳欢迎读者批评指正。我们的电话是 01062796045，信邮箱是 huchenhao@263.net。

编　者

2013 年 8 月

目　　录

第1章　基础知识编程技巧分析

1.1　基础知识简要介绍

1.1.1　常量与变量

常量是在程序运行过程中其值不可被改变的量。

常量的类型包括：整型常量、实型常量、字符常量、字符串常量和符号常量 5 类。

- 整型常量有 3 种形式：十进制整型常量、八进制整型常量和十六进制整型常量。八进制数以数字“0”开头，十六进制数以“0x”开头(数字 0 与字母 x)。
- 实型常量有两种形式：小数形式和指数形式。如 271.828000 和 2.71828e+02。
- 一个字符常量代表 ASCII 字符集中的一个字符，在程序中用单引号(')括起来。字符常量可以采用转义字符的方式表示。

注意：

‘a’和‘A’是两个不同的字符常量。

- 字符串常量指的是用双引号括起来的一个或多个字符。如“How are you？”。
- 符号常量是使用“#define”定义的常量，即用一个标识符代表一个常量。符号常量可以是上述各常量类型的任何一种类型。

变量就是其值在程序运行过程中可以改变的量。

变量的实质是代表一定的存储单元，存储单元中存储的是该变量的值。变量要有变量名，通过使用变量名可以引用其所代表的存储单元中的内容。不同类型的变量存储单元的大小不同。

注意：

变量必须“先定义，后使用”。

整型变量有 6 种，分别是：有符号基本整型(signed int)、无符号基本整型(unsigned int)、有符号短整型(signed short int)、无符号短整型(unsigned short int)、有符号长整型(signed long int)、无符号长整型(unsigned long int)。

实型变量分为两种：单精度类型(float)和双精度类型(double)。

字符型变量用于存放一个字符，用关键字 char 来声明。在 C 语言中，字符型变量也有带符号与无符号之分，一般情况下，直接使用 char 声明的字符变量常常是带符号的，其数

值范围为-128~127；可以使用 unsigned char 声明无符号字符型变量，其数值范围为 0~255。

1.1.2　数据类型及其转换

C 语言的基本数据类型包括：整型、实型、字符型和枚举类型，C 语言的构造类型包括：数组、结构体和共用体，此外还有指针类型和空类型。

当同一表达式中各个数据的类型不同时，需要把它们转换成同一类型后再进行计算，这种转换可以由编译程序自动实现，即自动转换；也可以由程序人员在编程时使用类型转换运算符实现，即强制转换。

自动转换又可以分为两类。

一类是必然实现的转换，即不论参与运算的数据类型是否一致，某些类型的数据也必然转换为另一种类型，主要包括如下两种情况：

(1) 凡属于 char、short 类型的变量在运算时一律转换为 int 类型；

(2) 凡属于 float 类型的变量在运算时一律转换为 double 类型。

另一类是当运算对象的数据类型不同时，按照从低到高的方向进行转换。例如，若 a 是 int 型，b 是 double 型，计算 a/b 时，将 a 转换成 double 型后，再与 b 相除。

强制转换的格式如下：

```
(数据类型标识)表达式
```

其作用是把表达式的结果转换为由“数据类型标识”指定的数据类型。例如，(double)(a+b)，是将(a+b)的值强制转换为 double 类型。

注意：

其与“(double)a+b”不同。

1.1.3　算术运算符和算术表达式

算术运算符包括：加法(+)、减法(-)、乘法(*)、除法(/)、取模(%)(求余数)，其运算符合数学上的运算规则。需要注意的是：两个整型量相除时，所得的结果为整型，如 3/2 的结果为 1，而非 1.5；模运算要求两个运算量必须是整型数据。

算术表达式即指用算术运算符和括号将运算量连接起来、符合 C 语法规则的表达式。括号可以改变运算符的自然运算顺序，注意只能使用小括号。

1.1.4　赋值运算符和赋值表达式

赋值运算符(=)的一般使用形式为：“变量=表达式”，用于连接一个变量(准确说是内存单元)与一个表达式，其功能是把表达式的值赋予位于赋值运算符左边的变量。

赋值运算符左边的变量(内存单元)称为左值，右边的表达式称为右值。

在赋值运算符之前加上其他运算符可以构成复合赋值运算符，这主要是为了提高 C 编译器的编译效率。赋值运算符与算术运算符结合可以形成：+=，-=，*=，/=，%=。

例如，“a*=b+c;”等价于“a=a*(b+c);”。

注意：

“a*=b+c;”不等价于“a=a*b+c;”。

需要说明的是：复合赋值运算符的两个符号之间一定不能有空格，复合赋值运算符与赋值运算符(=)具有相同的优先级和结合性。

1.1.5　其他运算符和表达式

1. 自增(++)与自减(--)运算符

自增(++)运算可使变量的值增 1，而自减(--)运算可使变量的值减 1。

在自增运算符和自减运算符的使用中注意以下事项。

(1) 只有变量才能用自增(减)运算符连接，不可以将该类运算符用于表达式或常量。

(2) 自增(减)运算有前缀方式和后缀方式两种，使用时注意两者的区别。

(3) 自增(减)运算符为单目运算符，其结合性都是自右向左结合。

2. 逗号运算符与逗号表达式

用逗号运算符连接两个表达式所形成的表达式，称为逗号表达式，一般格式如下：

```
表达式 1, 表达式 2
```

逗号表达式的求值过程是：先求表达式 1 的值，再求表达式 2 的值，并将表达式 2 的值作为逗号表达式的值。逗号运算符连接的表达式 1 或表达式 2 也可以是逗号表达式。因此有下面的扩展形式，它的值等于表达式 n 的值。

```
表达式 1, 表达式 2，…, 表达式 n
```

3. 求字节数运算符(sizeof)

求字节数运算符 sizeof 是一个比较特殊的单目运算符，用它可以求各种数据类型所占的字节数。某一个数据类型在不同的计算机系统中可能占有不同长度的内存空间，使用求字节数运算符 sizeof，就可以了解在自己所使用的计算机系统中，各种数据类型所占用的内存空间大小。例如，sizeof(float)的值是 float 型在内存空间所占用的字节数。

1.1.6　运算符的优先级和结合性

运算符的优先级是指：当一个运算量的两边连接两个运算符时，根据运算符的优先级决定先运算其左边的还是先计算其右边的运算符。

运算符的结合性是指：当优先级相同的运算符出现在同一表达式中、且没有括号的时候，运算符和操作数的结合方式。通常有从左到右结合和从右到左结合两种方式。

一些常见的运算符的优先级和结合性如表 1-1 所示。

表 1-1　一些常见运算符的优先级和结合性

优　先　级	运　算　符	名　　称	结 合 方 向
1	++ -- - (类型) sizeof	增 1 运算符 减 1 运算符 负号运算符 类型转换运算符 长度运算符	自右至左
2	* / %	乘法运算符 除法运算符 取模运算符	自左至右
3	+ -	加法运算符 减法运算符	自左至右
4	=　+=　-=　*= /=　%=	赋值运算符	自右至左
5	,	逗号运算符	自左至右

1.1.7　数据的输入与输出

1. scanf 函数

scanf 函数的作用是以指定的格式从标准输入设备读取输入的信息。调用格式如下：

```
scanf("格式控制串",地址列表);
```

“格式控制串”是用双引号括起来的字符串，也称为“转换控制字符串”，由格式说明符和普通字符组成。格式说明符由“%”和格式字符组成，如%d、%c等。作用是以指定的格式输入数据。对于普通字符，在输入数据时，要在对应位置输入与它们相同的字符。

“地址列表”是所有变量的地址，也可以是字符串或数组的首地址，但不是变量本身。这与 printf()函数不同，要特别注意。各个变量的地址之间用逗号“,”分开。

scanf 函数不能为表达式赋值，如“scanf("%4d",&(x+3));”是错误的。

如果实际输入数据的宽度大于格式说明中规定的数据宽度，则系统自动从左到右按规定的数据宽度截取数据，多余的将被丢掉。

若在%后面加一个“*”修饰符，则表示要跳过此项。

在格式控制串中，如果格式说明的类型与输入项的类型不匹配，系统不会给出错误信息，但可能得不到正确结果。

使用 scanf 函数输入数据时，输入后一定要按回车键，否则 scanf 函数接收不到数据。

2. printf 函数

printf 函数用于向标准输出设备按规定格式输出信息。调用格式如下：

```
printf("格式控制字符串",输出项列表);
```

“格式控制字符串”必须用双引号括起来，其中的内容可以包括：格式说明符、普通字符和转义字符。格式说明符由“%”和格式字符组成，如%d、%c等，作用是以指定的格式输出数据。普通字符的作用是为提高输出结果的可读性。转义字符是以“\”开头的字符序列，功能是在输出时产生一个“特殊”操作。

“输出项列表”列出所要输出的数据，数据的形式可以是单变量、字符串、表达式等。数据个数必须与格式控制字符串所说明的输出参数个数相同，各数据之间用逗号“,”分开，且顺序一一对应，否则将会出现意想不到的错误。

在格式控制字符串中，格式说明符必须与输出项在个数和类型上相匹配。不然将导致数据不能正确输出，这时系统并不报错。

printf 函数的返回值通常是本次调用中输出字符的个数。

3. getchar 函数

getchar 函数的作用是输入一个字符作为函数值。函数调用格式如下：

```
getchar();
```

getchar 函数没有参数，且每次只能接收一个字符。

getchar 函数返回值是输入的字符，若输入的是数字，也视为字符，按字符处理。

getchar 函数的返回值可以作为表达式的一部分，如“printf("%c\n",getchar());”。

4. putchar 函数

putchar 函数的作用是输出一个字符。函数调用格式如下：

```
putchar(字符);
```

putchar 函数的参数可以是单个字符或表达式，putchar 函数具有计算功能。

putchar 函数的作用等同于“printf("%c", 字符);”。

5. getch 函数

getch()函数的作用与 getchar 函数基本相同。函数调用格式如下：

```
getch();
```

使用 getchar 函数时，用户输入一个字符后必须要按下回车键，而使用 getch()函数，用户输入一个字符后，不需要按下回车键，输入的字符作为 getch()函数值可以马上赋予变量，并且屏幕上看不到这个被输入的字符。

getch()函数经常用在程序运行时需要按下任意键继续的情况，不用按回车键，变量就

会接收输入的字符，程序就会往下执行。getch()函数也经常用于不希望看到所输入内容的时候，如输入密码等。

1.2　运用基础知识时需要注意的问题

1.2.1　关于变量的定义

任何变量必须先定义后使用。

在定义变量时，要根据问题的需要，为变量选择适当的类型。若类型选择不当，运算过程会出现问题，可能计算结果与理论上的预期不符或数据溢出或数据丢失。

当参与运算的数据以及结果都是整数时，定义变量的类型可在 6 种整型中适当选取某一种整型。但是要注意数据的取值范围，使用 Turbo C 系统时，当运算范围局限在-32768 至 32767 之间时，可将变量定义为基本整型(int 型)；若运算范围比较大，可将变量定义为长整型(long 型)。注意定义整型变量的各种省略形式，如 int 等价于 signed int，long 等价于 signed long int。

当参与运算的数据和结果不全是整数时，最好将变量定义为实型。

例如，在下面的程序段中，变量 x 和 y 必须被定义为实型，y 的值才能是 0.5。如果 x 和 y 被定义为整型，那么 x/6 的值就不是 0.5，而是 0；同样，y 的值不是 0.5，而是 0。

```
x=3; y=x/6;
```

例如，在下面的程序段中，变量 m 和 n 必须被定义为长整型，而不能定义为基本整型。因为定义为基本整型后，取值范围局限在-32768 至 32767 之间，而运算结果超出该范围。

```
m=3456; n=1000; m=m*n; n=2*m;
```

1.2.2　关于一些运算符号

1. 关于除法(/)运算符

如果两个实型数据相除，结果当然是实型数据；如果一个实型数据与一个整型数据相除，结果也是实型的数据。

需要注意的是：两个整型数据相除的结果仍然是整型数据。这样一来，可能会造成：运行程序计算两个整型数据相除的结果与人们通常的理解不相符。例如，执行下面的程序段后，a/b 的值是 2，而不是 2.5。如果 a 与 b 中有一个是实型变量，则 a/b 是 2.5。

```
int a, b; a=5; b=2; a=a/b;
```

同样，执行下面的程序段后，a/b 的值是 0，而不是 0.25。如果 a 与 b 中有一个是实型变量，则 a/b 是 0.25。

```
int a=1,b=4; a=a/b;
```

2. 关于取模(%)运算符

取模(%)运算就是求余数运算，参与运算的量必须是整型，取模(%)运算的结果也是整型。常用取模(%)运算来判断某个整数能否整除另一个整数，即余数是否为 0。

注意不要将取模(%)运算与除法(/)运算搞混。例如，执行下面的程序段后，a/b 的值是 2，而 a%b 的值是 3。

```
int a=13 b=5, c, d; c=a/b; d=a%b;
```

可以利用取模(%)运算将整数分类。例如，若 a 是整型变量，那么 a%2 的值分别是 0 和 1，这样将整数分成了两类：余数为 0 的是一类(偶数)，余数为 1 的是一类(奇数)。同样，若 a 是整型变量，那么 a%3 的值分别是 0、1 和 2，这样可将整数分成三类。

3. 关于赋值运算符与复合赋值运算符

赋值运算符(=)不表示两端相等(C 语言用符号“==”表示相等)。赋值运算符的左端必须是一个变量名，右端是表达式(单个变量或常量是表达式的特殊情况)。计算机在执行赋值运算时，计算右端表达式的值后赋给左端的变量。因此，在 C 语言中，像 k=k+3 这样的赋值表达式是正确的；而在数学中，像 k=k+3 这样的写法显然是不行的，因为在数学中，符号“=”表示相等。

下面写法显然是错误的，因为赋值运算符的左端不是一个变量名。

```
a+b=5*c+d;   a/b=d+a%b;
```

而下面写法是合理的，若 d 的值是 2，则 a、b、c 的值都是 8。

```
a=b=c=d+6;
```

上面的写法相当于“a=(b=(c=d+6));”，先计算 d+6 的值 8 赋给 c，则表达式 c=d+6 的值等于 c 的值 8；然后将表达式 c=d+6 的值 8 赋给 b，b 的值也是 8；则表达式 b=(c=d+6)的值也是 8，最后将表达式 b=(c=d+6)的值 8 赋给 a，a 的值也是 8。

对于复合赋值运算符，容易出现的错误是没有将复合赋值运算符右端的表达式当成一个整体看待。例如，对于复合赋值运算表达式“m/=d+a*b”，转换成使用普通赋值运算符的表达式应该是“m=m/(d+a*b)”，而不是“m=m/d+a*b”。

初学者为了避免出错，编程时尽量不使用复合赋值运算符，遇见使用复合赋值运算符的表达式时，可以将其转换成使用普通赋值运算符的表达式。

4. 关于++与--运算符

对于 int 型变量 i，++i 和 i++都等价于 i=i+1，--i 和 i--都等价于 i=i-1。

注意：

“++”(或“--”)放在变量的前面与放在变量的后面是不同的。++i 和--i 是前缀表示法，

i++和 i--是后缀表示法。前缀表示法是将 i 值先增 1(或减 1)，再使用在表达式中；后缀表示法是先在表达式中使用 i 的值，然后再将 i 值增 1 (或减 1)。

例如，执行下面的程序段后，变量 i、j、k 的值分别是 8、7、7。

```
int i,j,k;   i=6;
j=++i;     /*表达式++i 的值是 7，所以 j 的值是 7，同时 i 的值变为 7 */
k=i++;     /*表达式 i++的值是 7，所以 k 的值是 7，同时 i 的值变为 8*/
```

例如，执行下面的程序段后，变量 i、j、k 的值分别是 4、5、5。

```
int i,j,k;   i=6;
j=--i;     /*表达式--i 的值是 5，所以 j 的值是 5，同时 i 的值变为 5*/
k=i--;     /*表达式 i--的值是 5，所以 k 的值是 5，同时 i 的值变为 4*/
```

“++”(或“--”)经常用在循环语句中，作用是让循环变量的值增 1(或减 1)，此时使用前缀或后缀形式都可以。

1.2.3　关于混合运算

当使用整型、实型和字符型数据进行混合运算时，如果一个运算符两侧的操作数的数据类型不同，则系统按“先转换、后运算”的原则，首先将数据自动转换为同一类型，然后在同一类型数据间进行运算。

自动转换只是针对一个运算符两侧的两个运算对象，而不能对表达式中的所有运算符涉及到的运算对象全部做自动转换。容易出现的错误是：误以为表达式中的所有运算符涉及到的运算对象同时都转换。

例如，执行下面的程序段后，x 的值是 6.13。

```
float x, y=8.26
x=10/4+y/2;
```

对于表达式“10/4+y/2”，其中既有整型运算对象又有实型运算对象，计算机不是将所有的运算对象都自动转换为实型(如果那样的话，x 的值是 6.63)，只是将每个运算符两侧的两个运算对象自动转换为同一类型。所以，对于 10/4，计算的值为 2；而对于 y/2，将除法运算符两侧的自动转换实型，计算的值为 4.13，最后得到表达式“10/4+y/2”的值是 6.13。

如果改成“x=10/4.0+y/2;”，或改成“x=(float)10/4+y/2;”，那么 x 的值是 6.63。

请读者分析，执行下面的程序段后，x 的值是多少?

```
int x; float y=8.26;
x=10/4+y/2;
```

当进行混合运算时，还有一个需要注意的是运算的优先级和结合性问题。表 1-1 列出了一些常见运算的优先级，对于++、--、%等运算需要特别注意，因为在数学中没有对应的运算符号。表 1-1 列出了一些常见运算符的结合性。从表 1-1 中看出，结合方向是自右至左的运算包括赋值运算以及复合赋值运算，表 1-1 中的其他运算都是自左至右的结合方

向。编程时为了在优先级和结合性问题上不至于出错，可以适当多放几个圆括号。

1.2.4　关于一些输入输出函数

1. 关于 scanf 函数

scanf 函数的格式如下：

```
scanf("格式控制串",地址列表);
```

使用 scanf 函数时，需要注意的一个问题是：格式控制串中的每一项格式说明符与后面地址列表中的每一项地址必须一一对应，不但个数要对应(个数相同)，而且顺序要对应。例如，下面程序段有错误。一个错误是个数没有对应(格式说明符是 3 项，而地址是 4 项)，另一个错误是顺序没有对应(&a1 应该对应%d，&a2、&a3 和&a4 应该对应%f)。

```
int a1; float a2, a3, a4;
scanf("%f, %d, %c", &a1, &a2, &a3, &a4);
```

而改成如下形式后就正确了：

```
int a1; float a2,a3, a4;
scanf("%d, %f, %f, %f ", &a1, &a2, &a3, &a4);
```

使用 scanf 函数时，需要注意的另一个问题是：地址列表中的每一项必须是地址。就是说必须在变量名称的前面加上&符号。但是数组名的前面不用加上&符号，因为数组名代表数组的首地址。例如下面是错误的：

```
int a1; float a2; char name[10];
scanf("%d, %f, %s ", a1, a2, name);
```

而改成如下形式后就正确了：

```
scanf("%d, %f, %s ", &a1, &a2, name);
```

使用 scanf 函数时，在格式控制串中的两个格式说明符之间，可以没有任何间隔符号，也可以有间隔符号，在格式控制串中也可以有其他字母。执行 scanf 函数时，对以上各种情况必须区别对待。例如执行下面的程序段：

```
float a1, a2, a3;
scanf("%f %f %f", &a1, &a2, &a3);
```

从键盘输入的 3 个实数，可以用空格、tab 键或回车键分隔。

若改成下面形式：

```
scanf("%f, %f, %f", &a1, &a2, &a3);
```

那么输入的 3 个实数，必须用逗号分隔(上面若不用逗号而用其他字符，输入时也要做相应的变动，将逗号改为其他字符)。

若改成下面形式：

```
scanf("a1=%f, a2=%f, a3=%f", &a1, &a2, &a3);
```

那么应该按如下形式输入(假设 a1 是 3.14, a2 是 2.718, a3 是 9.8)：

```
a1=3.14, a2=2.718, a3=9.8 ↵
```

使用 scanf 函数时，在格式控制串中可以加修饰符(需注意修饰符的用法)。

英文字母 l(不是数码 1)和 h 是两个常用的修饰符。在从键盘为 int 型变量 x 输入值时，用%d 即可。而对于 long 型变量 y，必须用%ld。就是说，使用 scanf 函数为长整型变量输入值时，在%d 中要加修饰符 l(英文字母 l，不是数码 1)。对于短整型变量，使用 scanf 函数时，在%d 中要加修饰符 h。如下面程序段所示：

```
int x; long y; short z;
scanf("%d, %ld, %hd", &x, &y, &z);
```

可以使用修饰符指定输入数据所占的列数，如“%md”也是常用的修饰符，其中 m 代表一个整数，这个整数就是指定的列数。

如果从键盘输入的数据超过指定的列数，则变量只接收指定列数内的数据。例如执行下面的程序段：

```
int x;
scanf("%3d", &x);
```

由于指定输入数据所占的列数为 3，如果从键盘输入 12345，则 x 只接收了 123。

再如：

```
int x,y;
scanf("%3d%2d", &x, &y);
```

如果从键盘输入 1234567，则 x 接收了 123，y 接收了 45。

对于修饰符，不要随意添加或去掉，否则会出错。请读者自行上机试验。

2. 关于 printf 函数

printf 函数的格式如下：

```
printf("格式控制字符串",输出列表);
```

对于不同类型的数据，必须使用正确的格式说明符，否则输出结果错误。例如下面的程序段中，应该使用格式说明符%f 对应 float 型变量 a1，可是却错误地使用了%d；应该使用格式说明符%d 对应 int 型变量 a2，可是却错误地使用了%f；应该使用格式说明符%ld 对应 long 型变量 a3，可是却错误地使用了%d。当然无法实现正确的输出。

```
float a1=3.14159; int a2=3426; long a3=45678;
printf("%d, %f, %d ", a1, a2, a3);
```

格式控制字符串中的格式说明符与输出列表中的输出数据项必须一一对应。不但个数要对应(个数相同)，而且顺序要对应。这与前面对 scanf 函数的要求是一致的。

printf 函数具有计算功能，它能够计算输出列表中的表达式的值，计算之后，按照对应的格式控制串中的格式说明符进行输出。所以，编程序时，对于有些需要计算的表达式，可以直接将该表达式作为 printf 函数的输出列表中的一项即可。

在格式控制字符串中，允许放一些普通字符，这些普通字符被原样输出，这样可以使得输出的信息更丰富、更容易理解。如下所示：

```
printf("整型变量 num 的值为：%d。", num);
```

注意在 printf 函数中正确使用转义字符和修饰字符。

如果使用 printf 函数输出常用的英文字母或者阿拉伯数字，一般情况下是不会使用转义字符的。通常，使用转义字符都是为了输出一些特殊的字符。例如，回车换行符(\n)是最常用的一个转义字符，回车符(\r)、退格符(\b)、换页符(\f)、横向跳格符(\t)也是常用的转义字符。需要注意的是：使用转义字符输出一些特殊字符时，很容易出现错误。例如，当输出反斜线(\)、单撇号(’)、双撇号(”)时，这 3 种符号在格式控制串的编码形式应该分别是：\\、\’、\”，而不小心就会丢掉前面的引导符号(\)。

若在转义字符中采用 8 进制或 16 进制形式表示一个符号时，需注意符号的编码形式和不同数制的转换问题，同时应该记住一些符号的 ASCII 码值，如 A 的十进制 ASCII 码值为 65(其他大写英文字符的 ASCII 码值依次递增)，a 的十进制 ASCII 码值为 97(其他小写英文字符的 ASCII 码值依次递增)，数码 0 的十进制 ASCII 码值为 48(其他数码的 ASCII 码值依次递增)。

如果对输出有特殊要求，则在 printf 函数中可以使用修饰字符。例如，当使用格式说明符“%4d”输出一个 int 型变量的值时，该值占 4 列并且是向右对齐的。如果想让该值向左对齐，可以使用格式说明符“%-4d”。当输出许多数据并且想让输出格式统一时，可以使用格式说明符“%md”或“%-md”，其中 m 是一个整数。当输出长整型变量的值时，容易出现的错误是忘记加修饰符号 l(英文 L 的小写)。当输出实型变量的值时，有时会采用“%m.n”的格式(m 和 n 是整数)，m 值表示总的输出位数，n 值表示小数的位数，此时需注意：小数点也要占一位，因此设计 m 值应大于 n 值。

C 语言规定，如果格式说明中包含连续的两个%字符，即“%%”字符，则“%%”不作为格式符使用，而是输出一个字符“%”。

3. 关于 putchar 函数、getchar()函数和 getch()函数

putchar()函数的作用是输出一个字符，该字符是 putchar()函数的参数值。putchar()函数的参数可以是常量，也可以是变量，还可以是表达式。如下所示：

```
char ch1='B';
putchar('A'); putchar(ch1); putchar('A'+32);putchar(ch1+32);
```

输出为：ABab。

从上面可以看出：putchar()函数具有计算功能，它能计算表达式的值，然后输出对应的字符。

getchar()函数的作用是接收用户输入的一个字符，将该字符作为函数值。如果用户输入的字符超过一个，getchar()函数也只是接收第一个字符。例如运行下面的程序段：

```
char ch1;
ch1=getchar();   putchar(ch1);
```

输入 4 个字符 abcd，结果只是输出一个字符 a。

getch()函数的作用也是接收用户输入的一个字符，将该字符作为函数值。但是 getch()函数与 getchar()函数不同，它们的区别是：使用 getch()函数，从键盘上输入一个字符后不需要按下回车键，并且屏幕上看不到这个被输入的字符；而使用 getchar()函数从键盘上输入一个字符后需要按下回车键，并且屏幕上可以看到这个被输入的字符。

1.3　编程技巧分析

1.3.1　关于一些简单计算的编程技巧分析

例 1.1　求 m 除以 n 的商和余数。程序如下：

```
#include <stdio.h>
int main( )
{    int m,n;
     printf("请按顺序输入变量 m 和 n 的值，然后按回车键。\n ");
     scanf("%d, %d", &m ,&n);
     printf("m 除以 n 的商为:%d\n", m/n);
     printf("m 除以 n 的余数为:%d\n", m%n);
     return 0;
}
```

分析：两个整数相除的结果仍然是一个整数。取模运算可以求出两个整数相除的余数。本题是直接利用整数除法(/)运算求出商，利用取模(%)运算求出余数。

例 1.2　交换两个字符型变量的值。程序如下：

```
#include <stdio.h>
int main( )
{    char ch1, ch2, t;
     printf("请先为变量 ch1 输入一个字符，然后为变量 ch2 输入一个字符。\n ");
     ch1=getchar();       ch2=getchar();
     printf("交换之前，变量 ch1 的值是%c，变量 ch2 的值是%c。\n ", ch1, ch2);
     t=ch1;    ch1= ch2;    ch2=t;
     printf("交换之后，变量 ch1 的值是%c，变量 ch2 的值是%c。\n ", ch1, ch2);
     return 0;
}
```

分析：经常遇见交换两个变量值的情况，此时可以借助第 3 个变量，完成交换。

交换的方法是利用变量 t 做为临时的存储单元，利用赋值语句，首先将 ch1 的值存放在 t 中，再将 ch2 的值存放在 ch1 中，最后将 t 的值存放在 ch2 中。

例 1.3 求一个 3 位整数的各位数码之和。程序如下：

```
#include <stdio.h>
int main( )
{    int m, sum=0;
     printf("请输入一个 3 位整数：");
     scanf("%d", &m);
     sum=sum+m%10;    /*将个位数码加到 sum*/
     m=m/10;          /*将 m 缩小十倍，原来的十位数码成为个位数码*/
     sum=sum+m%10;    /*将十位数码加到 sum*/
     m=m/10;          /*将 m 缩小十倍，原来的百位数码成为个位数码*/
     sum=sum+m%10;    /*将百位数码加到 sum*/
     printf("%d 的 3 位数码之和为:%d。\n", m, sum);
  return 0;
}
```

分析：利用除法(/)运算和取模(%)运算，可以将一个整数的各位数码分离出来。

本题首先利用取模(%)运算分离出个位数码；然后，通过让整数缩小 10 倍，再利用取模分离出个位数码(实际是原来整数的十位数码)；最后，再让整数缩小 10 倍，再利用取模分离出个位数码(实际是原来整数的百位数码)。

利用循环语句即可将任意一个整数的各位数码分离出来，求出它的各位数码之和。

例 1.4 鸡与兔同笼，已知笼中有头 h 个、脚 f 条，问笼中鸡兔各有几只？程序如下：

```
#include<stdio.h>
main()
{    int chick,rabbit,h,f;
     printf("input head and foot");        scanf("%d,%d",&h,&f);
     rabbit=(f-2*h)/2;          chick=(4*h-f)/2;
     printf("chick=%d, rabbit=%d \n", chick, rabbit);
}
```

分析：根据题意，设有 chick 只鸡 rabbit 只兔，则可列出如下方程组：

```
2*chick+4*rabbit=f
chick+rabbit=h
```

解此方程组，可得出：

```
rabbit=(f-2*h)/2，chick=(4*h-f)/2
```

因此写出如上程序。

利用编写程序可以计算许多数学问题。但要注意，计算机解题的过程是将人解决数学问题的思维过程用程序设计语言表现出来，编程之前需要人来分析和设计解题过程。

1.3.2　关于一些简单输入输出的编程技巧分析

例 1.5 利用 getch()函数实现程序的暂停。程序如下：

```
#include <stdio.h>
int main( )
{    printf("演示暂停功能\n ");
     printf("现在程序暂停\n ");
     printf("若结束暂停, 请按任意键\n");
     getch();
     printf("程序继续进行\n");
     printf("程序正常结束\n");
     return 0;
}
```

分析：程序在运行过程中有时需要暂停，用户按任意键后程序又继续运行。此时，可以利用 getch()函数来实现此功能。

本题只是说明可以利用 getch()函数实现暂停，没有使用 getch()函数值。当然实现暂停也可以用其他方法，如计算机在执行 scanf()函数、getchar()函数时也要暂停。

例 1.6 分析 getchar()函数与 getch()函数的区别。程序如下：

```
#include <stdio.h>
int main( )
{    char ch;
     printf("请输入一个英文字符后按回车键。\n ");
     ch=getchar();
     printf("您输入的英文字符是：%c。\n", ch);
     printf("请输入一个英文字符, 不用按回车键。\n ");
     ch=getch();
     printf("您输入的英文字符是：%c。\n", ch);
     return 0;
}
```

分析：getchar()函数和 getch()函数都可以接收用户输入的一个字符作为函数值。但是两者有区别，根据这个区别，在不同情况下可以分别选择是使用 getchar()函数或 getch()函数。

例 1.7 输入一个十进制整数(介于 33 和 126 之间)，分别输出它的八进制、十进制、十六进制形式，以及对应的 ASCII 码字符。程序如下：

```
#include <stdio.h>
int main( )
{    int num;
     scanf("%d",&num);
     printf("\n 八进制形式为:%o，", num);
     printf("\n 十进制形式为:%d，", num);
     printf("\n 十六进制形式为:%x，\n ", num);
     printf("对应的 ASCII 码字符为%c。\n", num);
     return 0;
}
```

分析：可以利用 printf 函数，使用不同的格式说明符输出整数，显示该数在不同进位制下的表现形式。将整数从一种进位制形式转换为另一种进位制形式可以使用数学中给出的方法，但是由于 C 语言给出了%o、%d、%x 等不同的格式说明符，所以转换很容易。

例 1.8 计算两个 4 位数的和。要求在输入输出时加入适当的提示信息。

```
#include <stdio.h>
int main( )              /*本程序的功能是计算两个 4 位数的和*/
{    int n1,n2;
     printf("请输入第一个 4 位的十进制整数，然后按回车键。\n ");
     scanf ("%d", &n1);
     printf("您输入的第一个 4 位的十进制整数是：%d。\n", n1);
     printf("请输入第二个 4 位的十进制整数，然后按回车键。\n ");
     scanf ("%d", &n2);
     printf("您输入的第二个 4 位的十进制整数是：%d。\n", n2);
     printf("您输入的两个 4 位的十进制整数的和是：%d。\n", n1+n2);
     printf("计算完毕, 程序结束，再见。\n");
     return 0;
}
```

分析：编程时适当加入提示信息可方便用户操作，适当加入注释可使程序易读易懂。

1.4 习　　题

1. 编写程序，输入圆的半径值，输出圆的面积值。

2. 编写程序，交换两个实型变量的值。

3. 编写程序，输入一个 3 位整数，输出它除以 7 的商数和余数。

4. 编写程序，求一个 4 位整数的各位数码之和。

5. 编写程序，输入一个大写英文字母，将该字母对应的 ASCII 码分别按八进制、十进制和十六进制形式输出，同时输出小写的该英文字母。

6. 编写程序，输入长方形的长、宽、高，计算并输出长方形的体积和表面积，输出时显示 3 位小数。

7. 编写程序，输入梯形的上底、下底和高，计算其面积。

8. 编写程序，对于一元二次函数 $y=3x^2-4x+5$，输入一个 x 值，计算对应的 y 值。

9. 编写程序，输入平面直角坐标系中两点的坐标，计算并输出它们中点的坐标。

10. 编写程序，输入 x、y 的值，计算并输出两元函数 $z=4x+5y-6xy$ 的值。

1.5　上 机 实 验

1. 实验目的

熟悉数据类型、运算符和表达式、常用的输入输出函数等基础知识，掌握应用基础知识的主要技巧，掌握顺序结构程序设计的方法。

2. 实验内容

(1) 调试运行下面程序，回答程序后面的问题。

```
#include <stdio.h>
main( )
{    int x1, x2, t, n, m;
     scanf ("%d, %d", &x1, &x2);
     t=x1;   x1= x2;   x2=t;
     n=x1/x2; m=x1%x2
     printf("%4d, %4d \n", n, m);
     return 0;
}
```

运行上面程序时，从键盘输入 2 个整数分别是 3 和 26。请说明程序的执行过程，解释两个输出值的含义。

(2) 调试运行下面程序，回答程序后面的问题。

```
#include <stdio.h>
int main( )
{    int m;
     printf("请输入一个 3 位整数：");
     scanf("%d", &m);
     printf("你输入的 3 位数为:%d。\n", m);
     printf("%d ", m%10);   m=m/10;
     printf("%d ", m%10);   m=m/10;
     printf("%d ", m%10);
     return 0;
}
```

运行上面程序时，从键盘输入 345。请说明程序的执行过程，解释输出内容与输入值的关系。

(3) 调试运行下面程序，回答程序后面的问题。

```
#include <stdio.h>
int main( )
{    int m; float x, y,z; char ch;
     scanf("%f, %f", &x, &y);
     m=x+y; z=x+y; ch=m;
```

```
        print f("%d, %f , %c ", m, z, ch);
        return 0;
    }
```

运行上面程序时，从键盘输入 32.16 和 33.17。请说明程序的执行过程。为什么 m 与 z 的值不同？m 的值与字符'A'有何关系？

(4) 编写并上机调试运行完成下面任务的程序：

从键盘输入两个整数，计算这两个整数的算术平均值和几何平均值，最后输出这两个整数，输出算术平均值和几何平均值。要求按照占 6 列的格式输出每个整数，要求按照总共占 12 列且有 3 位小数的格式输出平均值。

(5) 编写并上机调试运行完成下面任务的程序：

已知方程 x^2-5x+6=0 有两个不相同的实根，根据求根公式计算并输出两个实根。

(6) 编写并上机调试运行完成下面任务的程序：

从键盘输入 3 个小写英文字母，计算每个小写字母的 ASCII 码的总和以及算术平均值，输出 3 个小写英文字母对应的大写英文字母，输出所计算的总和以及算术平均值。

第2章　选择结构编程技巧分析

2.1　选择结构知识简要介绍

2.1.1　关系运算符与逻辑运算符

1. 6 个关系运算符

见表 2-1。

表 2-1　6 个关系运算符

>	>=	<	<=	==	!=
大于	大于等于	小于	小于等于	相等	不相等

2. 3 个逻辑运算符

见表 2-2。

表 2-2　3 个逻辑运算符

!	&&	||
逻辑非	逻辑与	逻辑或

3. 优先级和结合性

(1) 算术运算符：*、/、%优先级相同且高于+、－。

(2) 关系运算符：>、>=、<、<=优先级相同且高于==、!=。

(3) 逻辑运算符：！高于&&，&&高于||。

(4) 不同运算符：！高于所有算术运算符, 所有算术运算符高于所有关系运算符, 所有关系运算符高于&&和||逻辑运算符，所有逻辑运算符高于赋值运算符。

除赋值运算符外，其他二元运算符都是左结合。

2.1.2　if 语句的几种表现形式

1. 单分支结构

语句的格式如下，流程图如图 2-1 所示。

```
if(表达式) 语句
```

执行过程为：若表达式的值为真，则执行语句；否则不进行任何操作。

2. 双分支结构

语句的格式如下，流程图如图 2-2 所示。

```
if(表达式)  语句 1
else  语句 2
```

执行过程为：若表达式的值为真，则执行语句 1；否则执行语句 2。

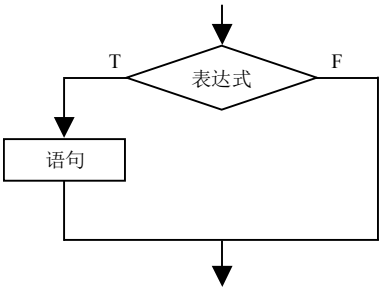

图 2-1　单分支结构流程图

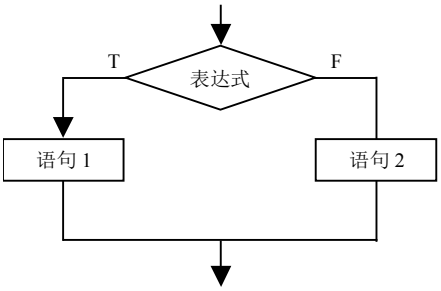

图 2-2　双分支结构流程图

3. 多分支结构

语句的格式如下，流程图如图 2-3 所示。

```
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
```

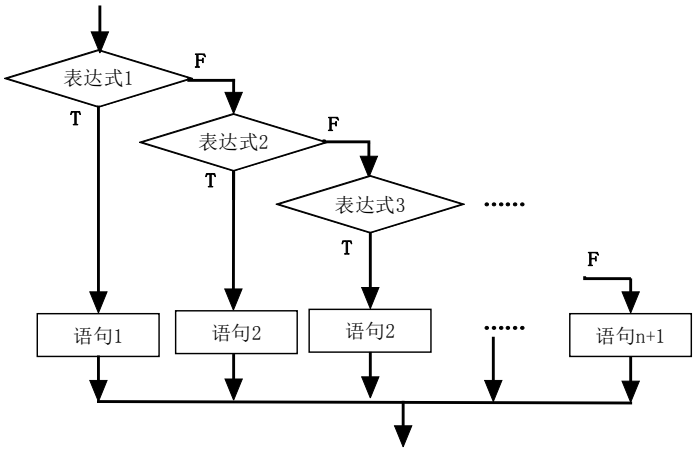

图 2-3　多分支结构流程图

执行过程为：若表达式 1 的值为真，则执行语句 1；否则若表达式 2 的值为真则执行语句 2；……；否则若表达式 n 的值为真则执行语句 n；否则执行语句 n+1。

2.1.3　switch 语句

switch 语句是专门为解决多分支问题设计的，格式如下：

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

执行过程：首先计算表达式的值，根据表达式的值，寻找 case 后面与表达式值相等的常量表达式，执行该常量表达式后面的语句块。若所有的 case 后面的常量表达式都与表达式的值不同，则执行 default 后面的语句。

注意：

(1) 每个语句块可以是一条简单语句，也可以是多条语句。多条语句可以用花括号构成复合语句，也可以不加花括号。

(2) 每个语句块里通常都含有一条 break 语句，作用是结束 switch 语句。

(3) case 后面一定是常量表达式，不可以包含变量。

(4) 允许某个“case 常量表达式 :”的后面没有语句块，此时向下顺序执行。

(5) default 可以省略。

2.1.4　条件运算符

条件运算符为“？ ：”，是 C 语言中唯一的三元运算符。由其组成的表达式为条件表达式。条件表达式一般格式为：

```
表达式 1 ？表达式 2 ： 表达式 3
```

执行过程：先求解表达式 1 的值，若其值为真，则条件表达式的值为表达式 2 的值，否则条件表达式的值为表达式 3 的值。

2.1.5　选择结构嵌套

如果要执行的分支语句本身是选择语句，则形成嵌套的选择结构。其形式多样，图 2-4 和图 2-5 列举了可能出现的几种情况。

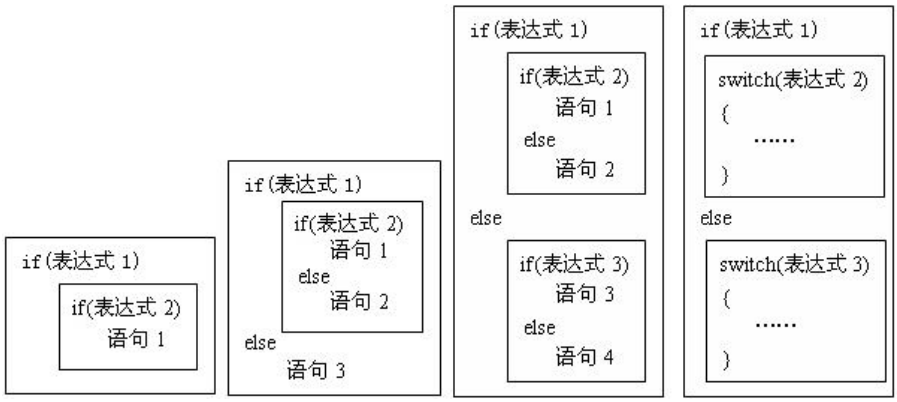

图 2-4　形式 1

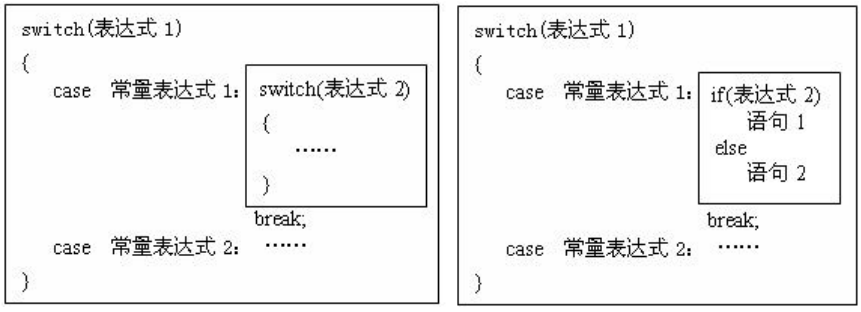

图 2-5　形式 2

2.2　运用选择结构知识时需要注意的问题

用选择结构编写程序时，要点 1 是：正确表达该问题所包含的条件；要点 2 是：选择合适的选择结构；要点 3 是：当需要选择结构有嵌套时，要设计正确的嵌套。

2.2.1　关于条件的表达

选择结构的条件通常是由逻辑运算符、关系运算符和算术运算符构成的表达式，但又不局限于这些运算符。只要构成的表达式正确，都可以表示条件。使用时注意以下几点：

1. 使用关系运算符时需要注意的一些问题

(1) 在>=、<=、==和!=每种运算符的两个符号的中间不要有空格。

(2) 不要将相等符号(==)写成赋值符号(=)。如果判断 a 与常量是否相等，如 a 与 1 是否相等，可写成 1==a。若写成 1=a，编译会报错。把相等==误写成=，这种错误排查很困难，用时要格外注意。

(3) 不要与数学运算符的写法混淆，如>=不能写成≥。

(4) 避免直接对实数作相等或不相等的判断。

计算机能够精确表示整数，但不能精确表示小数，因此在实数之间、实数与整数之间，进行相等比较时，不要直接用运算符==。例如，对于表达式“2.0/3.0*3.0==2.0”，在程序中一般应写为“fabs(2.0/3.0*3.0-2.0)<1e-6”。对于实型变量 a 和 3，作相等比较时，应写成“fabs(a-3)<1e-6”。一般情况下用 1e-6，若感觉精度不够，可将 1e-6 调整为更小的数。

(5) 字符可参与比较。比较时，以字符的 ASCII 码值与其他量进行比较。例如，'A'>100 为假，'A'<'C'为真。因为'A'的 ASCII 码值 65，'C'的 ASCII 码值 67。

2. 使用逻辑运算符时需要注意的一些问题

(1) 注意区分逻辑表达式中出现的数值是表示真假还是表示数值本身。

C 语言没有专门的逻辑值 true 和 false。一个条件成立时(即真)用 1 表示，不成立时(即假)用 0 表示。运算量用非 0 表示真，用 0 表示假。

例如，!56 的结果是 0。这里 56 是逻辑运算对象，56 被认为是真，被否定后为假。

例如，当 a=1，b=0 时，(a>b)&&b 的结果是 0。此时 a>b 中的 b 的取值是作为数值。但对于逻辑与符号后面的 b，b 的取值为 0，系统是将其作为逻辑假处理的。

注意：

!!a==a 不一定成立！只有 a 为 0 和 1 时成立。其他均不成立。例如，若 a=56，非 0 为真，则!a 的结果为 0，!!a 的结果为 1，和 a 不相等。

记住逻辑表达式和关系表达式的计算结果不是 0(假)就是 1(真)。

(2) 当表达多个条件时，需要用逻辑运算符连接各个表达式。

例如，3>2>1 虽然在数学中是成立的，但在 C 语言中却不成立。

原因在于：按从左到右的顺序，先计算 3>2，3>2 的结果为 1，再考虑该结果>1 吗？1>1 结果为 0，所以整个表达式的结果为 0。

若想表达 3>2 的同时 2>1，应该使用逻辑运算符连接，正确写法为：3>2&&2>1。

(3) 正确使用逻辑运算符。

正确使用逻辑运算符的前提是理解逻辑运算符所表示的含义。一般自然语言中的“和”、“同时”、“但”、“与”、“且”、“均”等应该用逻辑与(&&)运算符；自然语言中的“或者”、“条件之一”等应该用逻辑或(||)运算符；自然语言中的“非”、“不”等应该用逻辑非(！)运算符。

例如，“整数 m 和 n 均大于 0”，应该写为：m>0&&n>0。

例如，“并非整数 m 和 n 均大于 0”，应该写为：!(m>0&&n>0)。也可将!消去，表示为 m<=0||n<=0。

例如，“字符 ch 是英文字符”，即不论是大写英文字符或者是小写英文字符都可以，所以应该写为：(ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z')。例如，如果年数满足下列条件之一，则该年是闰年：①年数能被 4 整除，而不能被 100 整除。②年数能够被 400 整除。所以可用(year%4==0)&&(year%100!=0)||(year%400==0)判断该年是否闰年。

(4) 逻辑与(&&)和逻辑或(||)存在短路性质，导致并非所有运算量都参与运算。

例如，当 a=1，b=2，c=3，d=4 时，对于 a<b||c++>d，首先判断 a<b，其结果为 1，因此，此时已经知道整个表达式的结果为 1，所以系统对后面的 c++>d 不作运算，因此 c 的值仍为 3。

再例如，对于 a>b&&c++>d，由于 a>b 的结果为 0，因此可知整个表达式的结果为 0，所以系统对后面 c++>d 也不作判断。但如果是 a<b&&c++>d, a<b 结果为 1，此时需知道 c++>d 的结果，才可知整个表达式的结果，所以系统作 c++>d 运算，c 的值变为 4。

3. 使用多种运算符以及括号时需要注意的一些问题

在多种运算符出现在一个表达式里时，应注意运算符的优先级和结合性。

有时为了便于理解、方便阅读，可以适当增加括号。

例如，“a+b>c&&x+y<z”可以写成“(a+b>c)&&(x+y<z)”，因为两者等价。

例如，“!a==b||c>=d”可以写成“((!a)==b)||(c>=d)”，因为两者等价。

2.2.2　关于选择合适的选择结构

对于某个问题，若可以根据某个条件(取真或假)分成两个分支，则该问题适合使用双分支的 if 语句来解决；若可以根据多个条件(取真或假)分成多个分支，则该问题适合使用多分支的 if 语句来解决，或者适合使用嵌套的 if 语句来解决。

对于某个问题，若可以根据一个表达式的值(该表达式的值可以是整数，如 1、2、3 等；或者是字符，如'a'、'b'、'c'等)分成多个分支，则该问题适合用 switch 语句来解决。

1. 对于简单的 if 结构需要注意的问题

(1) if 后面的表达式别忘了加括号，而且括号要成对出现。

例如：

```
if (answer=='Y')    printf("The answer is Y");    √
if answer=='Y'    printf("The answer is Y");      ×
if ((answer=='Y')|| (answer=='y'))    printf("The answer is Y or y");    √
if (answer=='Y')|| (answer=='y')      printf("The answer is Y or y");    ×
```

(2) if 后面的括号里一定是个表达式，不能是一个语句。

例如，“if (x=y;)”是不可以的，括号里不可以是个语句。

(3) if 后面的复合语句是一个整体，“｛”和“｝”是这个整体的开始和结束。为了使代码层次清晰，应该使用缩进书写格式，这样不同层次的语句从不同的列处开始出现。

例如：

```
for(a=1; a<=10; a++)
   if (a%2==0)
      printf("a=%d",a);
```

(4) 不可随便添加分号。

例如，如果在上例中的 if 条件后面添加了分号，则条件为真时无法执行输出语句了，因为添加分号之后，if 语句条件为真时执行的是空语句。

```
for(a=1; a<=10; a++)
  if (a%2==0);
    printf("a=%d",a);
```

以上几点对其他形式的 if 语句同样适用。

2. 对于 if-else 结构需要注意的问题

(1) else 是对 if 条件的否定，else 不能独立存在，必须与 if 配对出现。

(2) else 后不能写成“else(条件表达式)”的形式，可以写为“else if(条件表达式)”，如此构成多分支 if 语句。

3. 对于 switch 结构需要注意的问题

(1) switch 后面的表达式的值可以为整型、字符型或枚举型，其他类型可以通过强制类型转换，转换成整型、字符型或枚举型。case 后的常量一定是整型常量、字符型常量或枚举型常量，不能是实型常量。关键词 case 和后面的常量之间用空格分隔。

例如，若 x 是 float 型变量，则必须转换成整型，如下：

```
switch((int)x)    /*强制转换成整型*/
{   case 5 : printf("%d\n",5); break;
    case 6 : printf("%d\n",6); break;
    default : printf("%d\n",15);
}
```

如果“case 5”被写为“case 5.8”，则编译报错。

(2) 若每个 case 后面都有 break，default 后面也有 break，则 case 和 default 语句的顺序不影响程序结果。一般 defualt 放在最后。

例如，下面两个 switch 语句是等价的。

```
switch(a)
{   case 5 : b=5;  break;
    case 6 : b=6;  break;
    default t:b=100;  break;
}
```

```
switch(a)
{   case 5 : b=5;  break;
     default t: b=100;  break;
    case 6 : b=6;  break;
}
```

对于下面两个 switch 语句，a=5 时，左边 switch 语句得 b=100，右边 switch 语句得 b=5。所以两个 switch 语句不等价。

```
switch(a)
{   case 5 :  b=5;
    case 6 :  b=6;
    default : b=100;  break;
}
```

```
switch(a)
{   case 6 : b=6;
    default : b=100;  break;
     case 5 : b=5;
}
```

(3) 多个 case 分支共同执行一组语句时，例如，a 取 1 或 2 或 3 时，都执行 b=2*a，可以如下编写：

```
switch(a)
{    case 1:
     case 2:
     case 3 : b=2*a; break;
     default : b=3*a; break;
}
```

注意，不能用逻辑运算符写成如下形式：

```
switch(a)
{    case 1||2||3 : b=2*a; break;
     default : b=3*a; break;
}
```

注意，不能用逗号分隔写成如下形式：

```
switch(a)
{    case 1,case 2,case 3 : b=2*a; break;
     default : b=3*a; break;
}
```

(4) 每个 case 后面的常量表达式的值不能与其他 case 后面的常量表达式相同，即要保证分支选择的唯一性。

2.2.3　关于选择结构的嵌套

(1) if 语句和 switch 语句可以出现在允许语句能够出现的任何地方。

若 if 或 switch 语句作为选择语句的组成部分，则构成嵌套，图 2-6 是嵌套的一种情况。

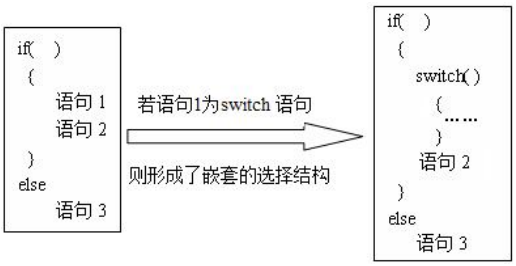

图 2-6　嵌套的一种情况

构成选择结构的嵌套并不难，而且嵌套形式比较自由。嵌套的实质是多分支选择。

(2) 对于嵌套的选择结构，要注意 if 和 else 的配对问题。

C 编译系统规定：else 总是与它前面最近的未配对的 if 匹配。并且由里向外逐对匹配。如果想要改变配对关系可添加大括号，以明确某个 if 与某个 else 的匹配关系。为了逻辑清晰，在书写时，使用缩进格式。

如图 2-7 所示的缩进的书写格式，可以使匹配更清晰，不易出错。

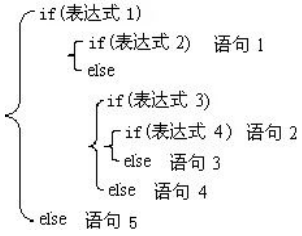

图 2-7　缩进的书写格式

(3) switch 语句嵌套时，break 语句只终止它所在的 switch 语句。

例如在下面的程序中，第 7 行中的 break 语句可以终止从第 6 行开始的 switch 语句，而第 9 行或第 10 行的 break 语句只能终止从第 8 行开始的 switch 语句。就是说，执行第 9 行或第 10 行的 break 语句后，只能跳出第 8 行到第 11 行的 switch 语句，执行第 12 行语句；不能直接跳到第 13 行后面去。而执行第 7 行中的 break 语句，才能跳到第 13 行后面去。

```
# include <stdio.h>
int main()
{    int x,y,z;
     printf("Input x,y:");
     scanf("%d, %d", &x, &y);
     switch (x)
     {    case  0  :  z=6*y; break;
          case  1  :  switch (y)
                      {    case  0 :  z=7*y+4*x; break;
                           case  1 :  z=2*x+3*y; break;
                      }
          case  2  z=4*z;
     }
     printf("z=%d\n", z);
     return 0;
}
```

(4) 当调试程序时，应该给出多个测量数据，尽可能测试到每个分支。

2.3　编程技巧分析

2.3.1　关于单分支和双分支的编程技巧分析

如果条件成立时执行某些操作，条件不成立时程序向下执行后面的语句，那么可以使用单分支 if 语句来完成程序的编写。

如果条件成立时执行某一类操作，条件不成立时执行另一类操作，并且两类操作不能同时被执行到，那么可以使用双分支 if 语句来完成程序的编写。

例 2.1　让计算机随机产生 3 个整数，将它们从小到大排序输出。程序如下：

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

```
#include<stdlib.h>
#include<time.h>
main()
{    int a,b,c,temp;
     srand(time(NULL));          /*设置随机数种子为当前时间*/
     a=rand();                    /*产生随机数*/
     b=rand();                   /*产生随机数*/
     c=rand();                   /*产生随机数*/
     printf("计算机随机产生的 3 个数为：%d, %d, %d\n",a,b,c);
     if(a>b)
       {    temp=a; a=b; b=temp; }
     if(a>c)
       {    temp=a; a=c; c=temp; }
     if(b>c)
       {    temp=b; b=c; c=temp; }
     printf("从小到大排序输出为：%d, %d, %d\n",a,b,c);
     return 0;
}
```

分析：函数 srand()是设置随机数种子，每次执行程序时该函数产生不同的整数序列，也就是传递给 srand()一个整数，以便决定 rand()函数从何处开始生成随机数。函数 srand()调用 time(NULL)返回一个自 1970 年 1 月 1 日以来经历的秒数。函数 rand()的值是取值为 0 到 32767 之间的随机整数。使用 rand()和 srand()函数需#include<stdlib.h>，使用 time()函数需#include<time.h>。

本程序是将最小的数存放在变量 a 中，方法是让变量 a 分别与变量 b、c 比较，若变量 a 中存放的数大于变量 b、c 中存放的数，则通过交换，使变量 a 中存放最小的数。最后通过变量 b 与 c 的比较，使变量 b 中存放 b 和 c 中最小的数(即 3 个数中位于中间的数)。然后可知，变量 c 中存放的数是最大的数。本程序用 3 个并列的单分支 if 语句实现。

请读者思考解决本问题的其他方法，例如使用其他的选择结构，然后编写对应的程序。

例 2.2 从键盘输入年和月的值，然后输出该月的天数。程序如下：

```
#include <stdio.h>
main()
{    int year,month,day=0;
     printf("input year ,month:\n");
     scanf("%d, %d", &year, &month);
     if (month==1||month==3||month==5||month==7||month==8||month==10||month==12)
        day=31;
     if (month==4||month==6||month==9||month==11)    day=30;
     if (month==2 && (year%4==0 && year%100!=0 ||year%400==0))    day=29;
     else    day=28;
     printf("days=%d",day);
     return 0;
}
```

分析：每年的 1、3、5、7、8、10、12 月是 31 天，4、6、9、11 月是 30 天，这两种情况使用单分支的 if 语句即可确定天数。闰年的 2 月份为 29 天，平年的 2 月份为 28 天。

闰年满足的条件如下。年份值能被 4 整除同时不能被 100 整除，或者年份值能被 400 整除，则此年为闰年，否则为平年。所以使用双分支的 if 语句即可确定 2 月份天数。

例 2.3 输入平面上三点的坐标，判断这三点是否共线。程序如下：

```
#include <stdio.h>
main()
{    float x1,y1,x2,y2,x3,y3,k1,k2;
     printf("请输入第 1 个点的坐标:");   scanf ("%f, %f,", &x1, &y1);
     printf("请输入第 2 个点的坐标:");   scanf ("%f, %f,", &x2, &y2);
     printf("请输入第 3 个点的坐标:");   scanf ("%f, %f,", &x3, &y3);
     k1=(y1-y2)/(x1-x2);   k2=(y3-y2)/(x3-x2);
     if (k1==k2)   printf("这三点共线！");
     else   printf("这三点不共线！");
     return 0;
}
```

分析：若直线方程为 y=kx+b，那么 k1=(y1-y2)/(x1-x2)是由(x1,y1)和(x2,y2)两点构成的直线的斜率，k2=(y3-y2)/(x3-x2)是由(x3,y3)和(x2,y2)两点构成的直线的斜率。若三点共线，则这两个斜率值应该相等，否则三点不共线。本题使用双分支的 if 语句完成判断。

读者可以根据几何知识思考其他的编程方法。例如由(x1,y1)和(x2,y2)得到一个直线方程的解析式，再由(x3,y3)和(x2,y2) 得到一个直线方程的解析式，判断两个解析式是否相等。

从上面的分析可以看到，利用计算机解数学题的过程，实际上是将人运用数学知识解题的过程用计算机语言表现出来的过程。例 2.4 也说明了这一点。

例 2.4 输入 3 个正数分别存于变量 a、b、c 中，判断这 3 个正数能否构成一个三角形。若能构成一个三角形，利用下面的数学公式计算三角形的面积 s。程序如下：

$$p=\frac{1}{2}(a+b+c) \qquad s=\sqrt{p(p-a)(p-b)(p-c)}$$

```
#include <stdio.h>
#include <math.h>
main()
{    float a,b,c,p,s;
     printf("请输入三个正数:");
     scanf ("%f, %f, %f", &a, &b, &c);
     if (a+b>c && a+c>b && b+c>a )
        {    printf("这三个正数能构成三角形。\n");
             p=(a+b+c)/2;   s=sqrt(p*(p-a)*(p-b)*(p-c));
             printf("三角形面积为：%f \n", s);
        }
     else   printf("这三个正数不能构成三角形。\n");
     return 0;
}
```

分析：构成三角形的条件如下。任意两边之和大于第三边(a+b>c && a+c>b && b+c>a)。本题使用双分支的 if-else 语句完成判断。

2.3.2　关于多分支的编程技巧分析

如果存在多个条件，每个条件成立时执行不同的操作，所有条件不成立时执行某一类操作。那么可以使用多分支 if 语句或者 switch 语句来完成程序的编写。

例 2.5 编写程序，计算如下的分段函数。程序如下：

$$p=\begin{cases} x & x<0 \\ 2x & 0\leqslant x\leqslant 5 \\ 3x & 5<x\leqslant 10 \\ 4x & x>10 \end{cases}$$

```
#include <stdio.h>
int main()
{    float x,y;
     printf("Please input x:");
     scanf("%f", &x);
     if(x<0)   y=x;
     else if (x<=5)   y=2*x;
     else if (x<=10)   y=3*x;
     else   y=4*x;
     printf("x=%f，y=%f\n", x, y);
     return 0;
}
```

分析：此例是计算分段函数，解决这类问题常用多分支结构(if-else-if)语句。按函数给出的变量 x 所满足的条件，顺序编写即可。

注意：

也可以不按变量 x 的取值顺序编写，只要保证 y 值计算正确即可。一般是按变量 x 的取值顺序，从最小值开始编写。也可从最大值开始编写，请读者来完成。

也可用嵌套的选择结构编写解决本问题的程序，请读者来完成。

例 2.6 输入一个不多于 5 位的正整数，(1)求出它的位数并输出；(2)按逆序输出它的每一位数码。例如，输入 6789，输出“输入的 6789 是 4 位数。逆序输出为 9876”。

```
#include <stdio.h>
int main()
{    long int shu1,shu2;   int ge,shi,bai,qian,wan,wei;
     printf("请输入一个正整数(1--99999): ");   scanf("%ld",&shu1);
     if   (shu1>9999)   wei=5;
     else if   (shu1>999)   wei=4;
     else if   (shu1>99)   wei=3;
     else if   (shu1>9)   wei=2;
     else   wei=1;
     printf("输入的%ld 是%d 位数。\n", shu1,wei);
     wan=shu1/10000;
     qian=(shu1-wan*10000)/1000;
     bai=(shu1- wan*10000- qian *1000)/100;
```

```
    shi=shu1%100/10;
    ge=shu1%10;
    printf("\n 逆序输出为");
    shu2=ge*10000+shi*1000+ bai*100+qian*10+wan;
    printf("%ld \n", shu2);
    return 0;
}
```

分析：在程序中，首先用 if-else-if 结构确定该数的位数，输出该数的位数；然后将每位数码分离出来，利用各数码，得到逆序排列的数，最后输出逆序排列的数。

能否用 switch 语句解决本问题？请读者思考。

例 2.7 假如今天是星期二，编写程序计算若干天后是星期几。程序如下：

```
#include <stdio.h>
int main()
{   int n;
    printf("请输入从今天开始过了多少天：");
    scanf("%d",&n);
    switch(n%7)
    {   case 0 : printf("星期二\n");break;
        case 1 : printf("星期三\n");break;
        case 2 : printf("星期四\n");break;
        case 3 : printf("星期五\n");break;
        case 4 : printf("星期六\n");break;
        case 5 : printf("星期天\n");break;
        case 6 : printf("星期一\n");break;
    }
    return 0;
}
```

分析：一个星期有 7 天，如果用 n 存放距离今天的天数，则让 n 与 7 作求余数运算，根据余数就可从星期二开始推算出 n 天后是星期几。余数有 7 种可能数字，即余数为 0 是星期二，余数为 1 是星期三，…，余数为 6 是星期一。所以本题适合用 switch 语句实现。

如果今天是星期五，程序如何修改？

例 2.8 给出某年某月某日，计算该日是当年的第几天，然后输出。程序如下：

```
#include <stdio.h>
int main()
{   int year,month,day,leap,number;
    printf("Input year, month, day:");
    scanf("%d%d%d", &year, &month, &day);
    if ((year%4==0)&&(year%100!=0)||(year%400==0))   leap=1;
    else    leap=0;
    switch(month)
    {   case 1 : number=day; break;
        case 2 : number=31+day; break;
        case 3 : number=31+28+leap+day; break;
        case 4 : number=31+28+leap+31+day; break;
        case 5 : number=31+28+leap+31+30+day; break;
```

```
            case 6 : number=31+28+leap+31+30+31+day; break;
            case 7 : number=31+28+leap+31+30+31+30+day; break;
            case 8 : number=31+28+leap+31+30+31+30+31+day; break;
            case 9 : number=31+28+leap+31+30+31+30+31+31+day; break;
            case 10 : number=31+28+leap+31+30+31+30+31+31+30+day; break;
            case 11 : number=31+28+leap+31+30+31+30+31+31+30+31+day; break;
            case 12 : number=31+28+leap+31+30+31+30+31+31+30+31+30+day; break;
        }
        printf("days=%d\n", number);
        return 0;
    }
```

分析：在程序中，变量 year、month、day 分别存放某年某月某日的值，变量 number 存放的值等于给定的年月日是当年的第几天。除 2 月外，其他月的天数是固定的。闰年时，2 月份的天数为 29 天，平年时为 28 天，程序中使用变量 leap 表现这点差别，闰年时 leap 取 1，平年时 leap 取 0。

在程序中，switch 语句中有 12 个 case 分支，对应 12 个月数，根据月数，为变量 number 赋值。若是 1 月份，number 值等于变量 day 的值；若是 2 月份，number 值等于 1 月的天数与 day 之和；若是 3 月份，number 值等于 1 月和 2 月的天数再加上 day；…，若是 11 月份，number 值等于 1 到 10 月的天数再加上 day；若是 12 月份，number 值等于 1 到 11 月的天数再加上 day。

在 switch 语句中，每个 case 的后面都有 break 语句，能否去掉 break？如果去掉 break，那么程序应该如何修改？请读者来完成。

这里的关键技巧是使用了变量 leap，利用 if 语句确定闰年或平年时 leap 的取值。如果不使用这个技巧，程序应该如何编写？请读者来完成。

例2.9 某公司产品检验系统规定：根据产品检验值确定产品等级。产品检验值(在0到100之间)在90及90以上的为优秀，产品检验值大于等于80且小于90的为良好，产品检验值大于等于60且小于80的为合格，产品检验值小于60的为不合格。编写程序，根据检验员输入的产品检验值，输出相应的产品等级。程序如下：

```
#include <stdio.h>
main()
{   int value,temp;
    printf("\n 请输入产品检验值：");
    scanf("%d",&value);
    temp=value/10;
    switch(temp)
    {   case 10 : case 9 :  printf("产品等级为优秀");   break;
        case 8 :            printf("产品等级为良好");   break;
        case 7 : case 6 :   printf("产品等级为合格");   break;
        default : printf("产品等级为不合格");
    }
}
```

分析：由于变量 value 取值在 0 到 100 之间，所以变量 temp 的值只有 11 种，分别是

10、9、8、7、6、5、4、3、2、1、0，根据题目的要求，这些值可以分成 4 类，分别对应优秀、良好、合格、不合格，所以可以使用 switch 语句完成程序的编写。

对于 switch(表达式)来说，当表达式取几个不同的值时，可以执行同一组语句。例如，本程序中变量 temp 取 7 和 6 时，都执行同一组语句“printf("产品等级为合格");break;”。

在本程序中，也可以不用 default，请读者思考程序应该如何修改。本题目也可以使用 if-else-if 语句编写，请读者思考并完成编写。

2.3.3　关于嵌套的编程技巧分析

如果某个问题需要对多个条件进行判断，即当判断某个条件成立或不成立后还要判断另一个或几个条件成立或不成立，那么可以使用嵌套的选择结构来完成程序的编写。

例 2.10 编程实现两个实数的加、减、乘、除四则运算。程序如下：

```
#include <stdio.h>
main()
{    float a,b,s;    char c;
     printf("请输入计算的表达式(形如 32+7 或 98-43 或 67*51 或 123/3 等)：");
     scanf("%f%c%f", &a,&c,&b);
     switch(c)
     {    case '+' : printf("\n%f+%f=%f",a, b, a+b);break;     /*做加法操作*/
          case '-' : printf("\n%f - %f=%f", a, b, a-b);break;   /*做减法操作*/
          case '*' : printf("\n%f*%f=%f", a, b, a*b);break;    /*做乘法操作*/
          case '/' :
             if (b==0) printf("分母不能为 0！");
             else    printf("\n%f / %f=%f", a, b, a/b);          /*做除法操作*/
             break;
          default : printf("\n 输入错误!");                     /*非以上字符, 输出错误提示信息*/
     }
     return 0;
}
```

分析：根据输入的四则运算表达式，变量 c 取 4 种不同的值(+、-、*、/)时分别执行不同的语句。在 switch 结构的内部，嵌套了一个 if- else 结构，处理分母为 0 的情况。

本题若不采用 switch 结构，程序应该如何编写？请读者来完成。

例 2.11 求方程 $ax^2+bx+c=0$ 的根(a、b、c 可取任意实数)。程序如下：

```
#include <stdio.h>
#include <math.h>
main()
{    float a,b,c,d,r,t;
     printf("请输入方程 ax2+bx+c=0 的系数 a b c:");
     scanf("%f%f%f",&a,&b,&c);
     if (fabs(a)<1e-6)                  /*若 a 等于 0*/
     {    if (fabs(b)<1e-6)             /*若 b 等于 0*/
               printf("方程无解\n");
          else
               printf("方程有唯一的解为%.3f\n",-c/b);
     }
```

```
    else
    {   d=b*b-4*a*c;   r=-b/(2*a);   t=sqrt(fabs(d))/(2*a);
        if (fabs(d)<1e-6)          /*若判别式 d 等于 0*/
            printf("方程有两个相等实根：%.3f\n",r);
        else if (d>1e-6)           /*若判别式 d 大于 0*/
            printf("方程有两个不等实根：%.3f,%.3f\n",r+t,r-t);
        else                       /*若判别式 d 小于 0*/
            printf("方程有两个虚根：%.3f+%.3fi,%.3f-%.3fi\n",r,t,r,t);
    }
    Return 0;
}
```

分析：若 a=0 且 b=0，则方程无解。若 a=0 且 b≠0，则方程有唯一解。若 a≠0，则根据判别式 b^2-4ac 的取值，方程的根有 3 种情况。

通过嵌套的 if 语句解决该问题。程序首先判断 a 是否等于 0，分两种情况：若 a 等于 0，程序执行一个 if-else 语句(第 8 行到第 11 行)；若 a 不等于 0，程序执行另一个 if-else-if 语句(第 14 行到第 21 行)。

注意：

实数与 0 的比较。不能简单地写(a==0)或(b==0 或(d==0)进行判断，因为计算机表示实数是近似的。因此一般让实数的绝对值与一个很小的正数去比较，如(fabs(a)<1e-6)或(fabs(b)<1e-6)或(fabs(d)<1e-6)，若成立则认为该实数的值等于 0。

本题可否选择其他的编写形式？若可以，请读者用其他的编写形式编写程序。

例 2.12 输入两个人的出生日期，判断两个人谁先出生。程序如下：

```
#include <stdio.h>
int main()
{   int year1,month1,day1, year2,month2,day2;
    printf("请输入第一个人的出生日期");
    scanf("%d%d%d",&year1,&month1,&day1);
    printf("请输入第二个人的出生日期");
    scanf("%d%d%d",&year2,&month2,&day2);
    if(year1==year2&&month1==month2&&day1==day2)
      printf("两人同年同月同日出生\n");
    else if(year1>year2)   printf("第二个人先出生");
    else if(year1<year2)   printf("第一个人先出生");
    else if(year1==year2)
    {   if(month1>month2)   printf("第二个人先出生");
        else if(month1<month2)   printf("第一个人先出生");
        else
        {   if(day1>day2)   printf("第二个人先出生");
            else    printf("第一个人先出生");
        }
    }
    return 0;
}
```

分析：因为本题需要比较 3 个条件，即年份的值是否相同、月份的值是否相同、日的

值是否相同，所以本题采用嵌套的 if 语句来编写。先比较年份的值，当年份相同时，再比较月份的值；当月份的值相同时，再比较日的值。

这里有一个比较的顺序问题。能否先比较日的值？或先比较月份的值？显然必须先比较年份的值，因为年份的值若不相同，则不需要比较月和日的值。在年份的值相同情况下，再去比较月份的值；最后考虑日的值的比较。

对于其他的多重比较，同样要注意顺序问题。

例2.13 某游轮公司规定游客乘坐游轮按如下标准收费：若是正常人，乘坐一等舱位收费300元，乘坐二等舱位收费260元，乘坐三等舱位收费230元，乘坐四等舱位收费200元。若是残疾人，乘坐一等舱位收费220元，乘坐二等舱位收费180元，乘坐三等舱位收费150元，乘坐四等舱位收费120元。编写程序，根据输入的游客信息(是否正常人，乘坐几等舱位)显示收费的标准数。程序如下：

```
#include <stdio.h>
main()
{    int man, cabin, charge;
     printf("若是正常人，请输入 1，若是残疾人，请输入 0：");
     scanf("%d", &man);
     printf("请输入舱位的等级(1、2、3、4)：");
     scanf("%d", &cabin);
     if (man==1)
     switch(cabin)
     {    case 1 : charge=300;   break;
          case 2 : charge=260;   break;
          case 3 : charge=230;   break;
          case 4 : charge=200;   break;
     }
     else
     switch(cabin)
     {    case 1 : charge=220;   break;
          case 2 : charge=180;   break;
          case 3 : charge=150;   break;
          case 4 : charge=120;   break;
     }
     printf("\n 收费标准为%d 元。", charge);
     return 0;
}
```

分析：变量 man==1 代表是正常人，man==0 代表是残疾人。变量 cabin 的值代表舱位等级。根据 man 和 cabin 的值，程序采用嵌套结构，首先按照 man 的值，分成两种情况，用 if 语句处理，对 man 的每一种情况，采用 switch 语句，分 4 种舱位为变量 charge 赋值。

本程序是将 switch 语句嵌套在 if 语句的里面，可否将 if 语句嵌套在 switch 语句里面来解决本问题？请读者思考并编写程序。

用不同的数值代表不同的事物或代表不同的现象，这是程序编写的常用技巧。本程序是用 1 代表正常人，用 0 代表残疾人。在其他情况下，例如可以用数值代表颜色：0 代表白色，1 代表黑色，2 代表红色，3 代表蓝色，等等。再如，可以用数值代表天气，0 代表

晴天，1 代表多云，2 代表小雨，3 代表中雨，4 代表大雨，等等。这样通过数值来表现不同的事物或不同的现象，然后编写程序处理问题。

通常情况下，当出现两个或两个以上条件需要进行判断时，程序往往采用嵌套的选择结构来处理。本题可否采用其他的嵌套方式，请读者编写对应的程序。

2.4 习　　题

1. 编写程序。将输入的 12 小时制的时间转换为 24 小时制。

2. 编写程序。输入平面上两点的坐标(x1,y1)和(x2,y2)，判断这两点的距离是否大于 9。

3. 编写程序。从键盘输入边长，分别计算正方形周长与面积，以及立方体体积。

4. 编写程序。输入的一个字符，如果该字符是大写字母则转换成小写字母，如果该字符是小写字母则转换成大写字母，其他字符不转换。

5. 编写程序。从键盘输入规定时速和机动车实际时速，根据交通违章超速罚款规定，输出该驾驶员需受的处罚。交通违章超速罚款规定如下：

(1) 超过规定时速 10%以下，罚款 50 元，记 3 分;

(2) 超过规定时速 10%以上不足 50%的，罚款 200 元，记 3 分;

(3) 超过规定时速 50%以上不足 70%的，罚款 500 元，记 6 分;

(4) 超过规定时速 70%以上不足 100%的，罚款 1000 元，记 6 分，可以并处吊销;

(5) 超过规定时速 100%的，罚款 2000 元，记 6 分，可以并处吊销。

6. 编写程序。从键盘输入一个不超过 30000 的正整数存放在变量 n 中，要求：

(1) 把 n 的各位数字倒着输出，如 n=258，应输出 852；

(2) 求出 n 是几位数。

7. 编写程序。已知年和月，求该月的天数。

8. 编写程序。已知 1~7 之间的数字与英文单词 Monday，Tuesday，Wednesday，Thursday，Friday，Saturday，Sunday 对应。由键盘输入数字，输出相对应的星期数。

9. 编写程序。求如下分段函数的值，要求分别用 if 嵌套语句和 switch 语句实现。

$$y=\begin{cases}\sqrt{x} & x>0\\ 8 & x=0\\ 1/x & x<0\end{cases}$$

10. 编写程序。从键盘上输入一个整数，判断其正负及奇偶性。

11. 编写程序。已知 A、B、C、D、E、F 共 6 个人站成一排，按如图 2-8 所示的形式开始报数，现从键盘敲入一个数，问由谁报到该数？

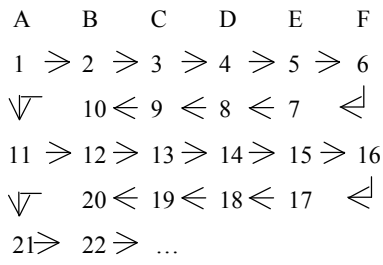

图 2-8　报数

12. 编写程序。现有 3 个大小相同的球，有一个球的重量和另外两个不同。编程实现用一架天平以最少的次数找出那个重量不一样的球。

13. 编写程序，计算工资。公司里的职工按其入职先后有不同的工资级别，5 年以下 40 元/h，5 年以上(含 5 年)10 年以下 60 元/h，10 年以上(含 10 年)80 元/h。一周工作时间需满 40h，不足部分按其工资级别的 0.5 倍扣钱，超出部分按其工资级别的 1.5 倍计算。按此方式计算职工一周的工资。

14. 编写程序，根据献血者的性别和体重确定献血量。男性体重超过 100 公斤(含 100 公斤)一次献血 400ml，低于 100 公斤一次献血 200ml；女性体重超过 60 公斤(含 60 公斤)一次献血 400ml，低于 60 公斤一次献血 200ml。

15. 编写程序。已知花店一支康乃馨 4 元，一支玫瑰 5 元，一支百合 6 元。将当前所有的钱全部用来买花，并且要求买的数量最多，钱也全部用光。输出各买了几支。

16. 编写程序。从键盘输入 x 的值，计算并打印下列分段函数 y 的值。

$$y=\begin{cases}0 & (\text{当 } x<60 \text{ 时})\\ 1 & (\text{当 } 60\le x<70 \text{ 时})\\ 2 & (\text{当 } 70\le x<80 \text{ 时})\\ 3 & (\text{当 } 80\le x<90 \text{ 时})\\ 4 & (\text{当 } x\ge 90 \text{ 时})\end{cases}$$

17. 编写程序。已知某公司销售人员的底薪为 1200 元，某月销售的利润(profit)和利润提成关系如下，计算销售人员的收入。

profit≤3000	没有提成
3000<profit≤5000	提成10%
5000<profit≤10000	提成15%
10000<profit≤20000	提成20%
profit>20000	提成25%

18. 编写程序。输入一百分制成绩，要求输出成绩的等级 A、B、C、D、E。90 及 90 分以上为 A，大于等于 80 且小于 90 分为 B，大于等于 70 且小于 80 分为 C，大于等于 60 且小于 70 分为 D，小于 60 分的为 E。

2.5　上 机 实 验

1. 实验目的

熟悉选择结构，熟悉选择结构的嵌套，掌握选择语句的使用方法，掌握与选择结构相关的一些常见问题的求解方法。

2. 实验内容

(1) 调试运行下面程序，回答程序后面的问题。

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

程序的功能是什么？如果将 if 后面的符号'<'改写成符号'>'，程序的功能是什么？将上面的程序用嵌套的选择结构实现，程序如何修改？

(2) 调试运行下面程序，若有编译错误，找出错误原因并修改正确，然后回答程序后面的问题。

```
#include <studio.h>
void main()
{    float score;
     printf("请输入成绩(1～100)：")
     scanf("%f", &score);
     if(score>100||score<0);       printf("成绩输入有误");
     else (0<=score<=100)
     switch((int)score/10)
     {    case 10||9 : printf("A\n");break;
          case 8 : printf("B\n");break;
          case 7 : printf("C\n");break;
          case 6 : printf("D\n");break;
          default :    printf("E \n");
     }
}
```

调试通过的程序的作用是什么？把 switch 语句替换为 if 语句，程序如何编写？