第3章 选择结构程序设计

大部分程序设计中都会用到选择结构。选择结构是指对给定的条件进行判断，然后根据判断结果去执行不同的操作，例如，根据考试成绩的高低来决定是否颁发合格证书，根据用户输入的密码来决定是否进入账户等。在C语言中，选择结构主要通过if语句和switch…case语句来实现。本章首先介绍关系运算和逻辑运算，然后介绍选择结构的不同实现方式。

3.1 关系运算符和关系表达式

在选择结构中通常需要判断表达式是“真”还是“假”。例如，对于表达式a＜b，真值将说明a是小于b的。在许多编程语言中，类似a＜b这样的表达式具有特殊的“布尔”类型或“逻辑”类型，其值为“真”或“假”。而在C语言中，诸如a＜b这样的比较运算会产生两个整数：0或1。0代表“假”(条件不成立)，1代表“真”(条件成立)。

C89中没有定义布尔类型，C99中提供了名为_Bool的布尔类型。

关系运算符和关系表达式

3.1.1 关系运算

关系运算实际上是一种“比较运算”，即用关系运算符对两个数进行比较，比较它们之间的“大小关系”。例如，关系表达式a＞3用于判断变量a的值是否大于3。

关系运算符可以用于比较整数和浮点数，也允许比较混合类型的操作数。C语言的关系运算符和数学上的＜、＞、≤、≥运算符号相对应，只是个别符号的书写形式有所不同，见表3-1。

表3-1 C语言中的关系运算符

符　号	含　义	示　例
＜	小于	3＜6　　运算结果为1
＞	大于	2.7＞5.4　　运算结果为0
＜=	小于等于	'A'＜='B'　　运算结果为1
＞=	大于等于	'a'＞='A'　　运算结果为1

字符按ASCII码值存储，根据ASCII码值大小进行比较，如'0'＞0的值为1。

【例3-1】 求x＝7＜3＜5的值。

表达式x＝7＜3＜5相当于：

$$\begin{aligned} x &= (7 < 3) < 5 \\ &= 0 < 5 \\ &= 1 \end{aligned}$$

因此，x 最后获得的值为 1。

需要注意的是，形如 x<y<z 的表达式在 C 语言中是合法的，这个表达式等价于(x<y)<z，即首先检测 x 是否小于 y，然后用比较后产生的结果(1 或 0)来和 z 进行比较。所以这个表达式并不是测试 y 是否介于 x 和 z 之间。假设要测试 y 是否介于 x 和 z 之间，则需要结合后面介绍的逻辑运算符才可以实现。

- 关系运算的结果为 0 或 1，0 代表假(条件不成立)，1 代表真(条件成立)。
- 在某些场合下，为了明确运算顺序，增强程序的可读性，最好在需要的地方添加括号。例如，c=(a<=b)表示把关系表达式 a<=b 的结果赋给变量 c，它显然比 c=a<=b 直观明确。

【例 3-2】 判断成绩是否及格。

```
#include<stdio.h>
int main()
{
  int grade;
  scanf("%d",&grade);
  if(grade>=60)          //判断成绩是否大于等于 60
    printf("pass\n");    //若条件成立，则输出及格信息
  return 0;
}
```

运行结果：

```
78↵
pass
```

本例通过关系表达式(grade>=60)来判断一个给定的成绩是否大于等于 60，如果条件成立，则输出及格的信息。

关系表达式主要用于选择结构中的条件判断，选择结构的内容将在本章后面详细介绍，由于 C 语言的书写比较接近自然语言，因此不会影响对本例的理解。

3.1.2 判等运算


关系运算符和关系表达式

C 语言中的判等运算符也属于关系运算符，但它们有着比较特殊的形式，见表 3-2。

表 3-2 C 语言中的判等运算符

符　号	含　义	示　例
==	等于	5==6　运算结果为 0
!=	不等于	3!=5　运算结果为 1

由于一个等号“=”在 C 语言中已经用来表示赋值运算符了，因此“等于”运算符就用相邻的两个等号“==”来表示。“不等于”运算符也是由两个字符!和=组成的“!=”。

【例 3-3】 已知 ch='a'，求 x=(ch=='b')的值。

表达式 x=(ch=='b')相当于：

$$x=('a'=='b')$$
$$=0$$

即，x最后获得的值为0。

本例首先判断表达式(ch==='b')的值，由于ch存储的是字符常量'a'，显然与字符常量'b'不相等，判断结果不成立，为0，因此变量x最终获得的结果为0。

【例3-4】 简单密码判断。从键盘输入一个数字，与内部设定的密码数字相比较，如果一致则给出“OK”信息。

```
#include <stdio.h>
int main()
{
  int password = 367,guess;        //内部设定密码值为 367
  scanf("%d",&guess);
  if(password == guess)            //判断输入的数字与内部设定的密码是否一致
    printf("OK\n");                //若一致则输出 OK 信息
  return 0;
}
```

运行结果：

```
367 ↵
OK
```

本例判断两个数字是否相等，需要使用等于运算符“==”，而不能写成赋值运算符“=”。语句if(password==guess)用来测试输入的guess是否等于password，如果写成语句if(password=guess)，则是先把guess的值赋给password，这样做的结果是修改了变量password中的数值，运行结果往往就背离了用户的期望。

不要混淆等于(==)运算符和赋值(=)运算符，这是最容易出现的C编程错误之一。

在第2章中曾经提到，浮点型数据在存储时会有误差，因此在用关系运算符直接对它们进行比较时，可能会得出错误的结果，见例3-5。

【例3-5】 浮点数的判等运算。

```
#include <stdio.h>
int main()
{
  double d0 = 0.3;
  double d1 = 0.1;
  double d2 = 0.2;
  printf ("%d\n", d0 == (d1 + d2));
  return 0;
}
```

运行结果：

```
0
```

本例运行结果为0，说明d0与(d1+d2)不相等，这显然不符合本题的情况，这种错误是由浮点数的存储误差造成的。因此，一般应避免对两个浮点数直接进行判等运算，而是采用判断两者的差的绝对值是否小于某个很小的数来实现，例如：

```
x= =y
```

可写成：

```
fabs(x-y)<1e-6
```

即如果 x 与 y 的差值非常小(小于 10^{-6})，就可以认为 x 与 y 是相等的。

fabs 是 C 语言标准库中求绝对值的函数，具体用法见附录。

逻辑运算符和逻辑表达式

3.2 逻辑运算符和逻辑表达式

3.2.1 逻辑运算符

关系表达式只适用于描述单一的条件，对于较复杂的复合条件需要将若干个关系表达式连接起来才能描述，如描述“x 大于 0 且不等于 5”，需要将两个关系表达式 x>0 和 x!=5 连接起来。

实现多个关系表达式的连接要用到逻辑运算符，C 语言提供了三个逻辑运算符，见表 3-3。其中，&&(逻辑与)和||(逻辑或)是双目运算符，它要求有两个操作数，!(逻辑非)是单目运算符，只要求有一个操作数。

表 3-3 C 语言中的逻辑运算符

符　号	含　义	示　例
!	逻辑非	!(math<=60)
&&	逻辑与	(math>90)&&(science>90)
||	逻辑或	(math>90)||(science>90)

逻辑运算符的操作规则如下。

(1) 逻辑非：当且只当 x 为零时，!x 的结果为 1。

(2) 逻辑与：当且只当 x 和 y 都为非零时，x&&y 的结果为 1。

(3) 逻辑或：当且只当 x 和 y 中至少有一个为非零时，x||y 的结果为 1。

表 3-4 为逻辑运算的真值表，表示当 a 和 b 的值为不同的组合时，各种逻辑运算所得到的值。

表 3-4 逻辑运算的真值表

a	b	!a	!b	a&&b	a||b
0	0	1	1	0	0
0	非 0	1	0	0	1
非 0	0	0	1	0	1
非 0	非 0	0	0	1	1

C 语言把非零数据当作“真”，把零当作“假”看待。

实际上，逻辑运算符两侧的运算对象可以是任何类型的数据。表 3-5 给出了几种逻辑运算的样例。

表 3-5　各种逻辑运算示例

逻辑表达式	结　果	说　明
!4	0	整数 4 为非 0,进行非运算后,结果为 0
5&&6	1	5 和 6 均为非 0,与运算后的结果为 1
'a'&&'b'	1	'a'与'b'的 ASCII 码都不为 0,即非 0,与运算后的结果为 1
4||0	1	4 为非 0,或运算中只要有一个非 0,结果就是 1
!8||0	0	8 为非 0,“非”运算后!8 的结果为 0,再与 0 进行或运算,结果为 0

3.2.2　用逻辑表达式表示条件

程序中经常需要将用文字或数学公式描述的条件转换为 C 语言的表达式。许多算法步骤需要检测某个变量的值是否位于指定的取值范围内。例如,假设用 min 表示取值范围的下限,max 表示取值范围的上限(min < max),如果要表示 x 的取值范围在 min 和 max 之间(含),则数学上可以用以下表达式来表示:

$$min \leqslant x \leqslant max$$

而写成 C 语言的表达式就应该是:

```
min<=x&&x>=max
```

图 3-1　表达式 min<=x&&x<=max 为 1 时 x 的取值范围

图 3-1 用阴影表示了 x 的取值范围,如果 x 位于该范围内,表达式值为 1,否则表达式值为 0。

表 3-6 给出了一些条件的描述以及对应的 C 语言表达式。

表 3-6　用 C 表达式表示条件

条　件	逻辑表达式
x 和 y 都大于 z	x>z&&y>z
x 位于 u 到 v 之间(含)(u<v)	u<=x&&x<=v
x 位于 u 到 v 之外(u<v)	x<u||x>v
x 能被 y 整除,但不能被 z 整除	x%y==0&&x%z!=0
x 能被 y 整除,又能被 z 整除	x%y==0&&x%z==0

【例 3-6】 判断大写字母。根据 ASCII 码的排列规律,如果某字符处于大写字母'A'以及大写字母'Z'之间,则可以确定它是一个大写字母,输出“Upper case”,否则不予处理。

```
#include<stdio.h>
int main()
{
  char ch;
  ch=getchar();                    //从键盘读入一个字符
  if(ch>='A'&&ch<='Z')             //判断字符 ch 是否大写字母
    printf("Upper case\n");
  return 0;
}
```

运行结果：

```
G↵
Upper case
```

本例需要判断字符 ch 是否在大写字母'A'和'Z'之间，即是否满足'A'≤ch≤'Z'。在 C 语言中可用“与”运算符将 ch >= 'A'和 ch <= 'Z'连接起来，即(ch >= 'A'&&ch <= 'Z')，或者('A'<=ch&&ch <= 'Z')也可以。但切记不能按数学上的习惯直接写成('A'<=ch <= 'Z')。表 3-7 对这两种写法进行了分析，用不同的测试数据来演示不同的写法对运算过程和运算结果的影响。

表 3-7 “判断字符 ch 是否在大写字母'A'和'Z'之间”的 C 表达式解析

逻辑表达式	运 算 步 骤	测 试 数 据	说 明
(正确写法) 'A'<=ch&&ch<='Z' 或 ch>='A'&&ch<='Z'	(1) 先计算'A'<=ch (2) 再计算 ch<='Z' (3) 最后将(1)和(2)的计算结果进行“与”运算	ch='G'	(1) 'A'<=ch 结果为 1。 (2) ch<='Z'结果为 1。 (3) 两个 1 进行“与”运算，结果为 1。 **输出 Upper case**
		ch='a'	(1) 'A'<=ch 结果为 1。 (2) ch<='Z'结果为 0。 (3) 1 和 0 进行“与”运算，结果为 0。 **不输出 Upper case**
(错误写法) 'A'<=ch<='Z'	(1) 先计算'A'<=ch (2) 再计算(1)的结果是否小于等于'Z' (3) 等价于('A'<=ch)<='Z'	ch='G'	(1) 'A'<=ch 结果为 1。 (2) 1<='Z'结果为 1。 **输出 Upper case**
		ch='a'	(1) 'A'<=ch 结果为 1。 (2) 1<='Z'结果为 1。 **输出 Upper case**

根据表 3-7 的分析可知，判断字符是否大写字母时，如果将表达式误写成'A'<=ch <='Z'，那么其运算步骤就不一样了，虽然当测试字符为大写字母'G'时，也能误打误撞地输出 Upper case，但当测试字符为小写字母时，输出结果也是 Upper case，这就明显错误了，因为把小写字母'a'判断为大写字母了。因此需要正确书写表达式，同时在测试程序时要多测试各种不同数据，以发现程序中隐藏的问题。

根据前面的分析可知，在 C 语言程序中，如果要判断某一字符 ch 是否为小写字母，正确的逻辑表达式应为'a'<=ch&&ch <='z'，而如果要判断某一字符是否为数字字符，正确的逻辑表达式为'0'<=ch&&ch <='9'。

3.2.3 短路求值

在逻辑表达式的求解中，并不是所有的逻辑运算符都需要执行，有时只需要执行一部分运算符就可得出逻辑表达式的最后结果，这时就不会继续运算下去，这一现象被称作“短路求值”。&& 和||就是所谓的“短路”运算符，例如表达式 x&&y，只有当 x 为真时，才需要判断 y 的值，若 x 为假，就立即得出整个表达式为假，不必再判断 y 的值了。又如表达式 x||y，只要 x 为真，就立即得出整个表达式为真，不必再判断 y 了，只有当 x 为假时，才需要判断 y 的值。

【例 3-7】 短路求值分析。

```
#include<stdio.h>
int main()
{
  int a=1,b=2,c=3,d=4;
  int m=1,n=1,t;
  t=(m=a>b)&&(n=c>d);
  printf("t=%d,m=%d,n=%d\n",t,m,n);
  return 0;
}
```

运行结果：

```
t=0,m=0,n=1
```

对于逻辑表达式(m=a>b)&&(n=c>d)，首先计算(m=a>b)，由于 a>b 的结果为 0，因此 m=0，这时可立即得出整个逻辑表达式的结果 t 为 0，因此(n=c>d)不被执行，n 的值就不会发生变换，依然保持 1。

短路求值：只要逻辑表达式的值能够确定，就停止对表达式求值。

用 if 语句实现选择结构

3.3 if 语句

if 语句通过对给定的条件进行分析、比较和判断来完成不同的操作。C 语言的 if 语句有三种表现形式：单分支 if 语句(if 语句)，双分支 if 语句(if…else 语句)，多分支 if 语句(if…else if 语句)。

3.3.1 单分支 if 语句

在前面的例 3-2 中出现过以下语句：

```
if(grade>=60)
  printf("pass\n");
```

这是单分支 if 语句。其流程图见图 3-2。

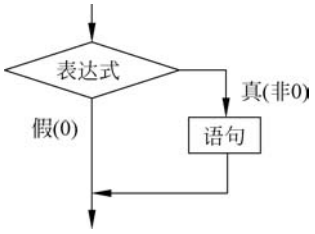

图 3-2 单分支 if 语句流程图

单分支 if 语句的用法见表 3-8。

表 3-8 单分支 if 语句的用法

语 法	示 例	说 明
if(表达式) 语句	if(grade>=60) printf("pass\n");	(1) 先计算表达式的值，如果表达式的值为真，则执行语句；否则不做任何操作。 (2) 此处的“语句”可以是单条语句，也可以是复合语句。 (3) 表达式可以是任意类型。若表达式值为 0，按“假”处理；若表达式值非 0，按“真”处理。例如： if(1) printf("That's OK"); 该 if 语句是合法的，因为表达式的值为 1，按“真”处理，最后输出 That's OK

【例 3-8】 求整数的绝对值。(nbuoj1035)

输入一个整数,输出它的绝对值。

由于正数和零的绝对值就是其本身,因此不需要处理,而负数的绝对值则可通过类似 x=－x 的形式求取。程序如下。

```
#include<stdio.h>
int main()
{
  int a,temp;
  scanf("%d",&a);
  temp=a;                          //将输入的数值复制到变量 temp 中
  if(temp<0)
      temp=-temp;                  //对 temp 进行判断,若是负数,则取反
  printf("%d\n",temp);
  return 0;
}
```

运行结果:

```
-9↵
9
```

在 if(表达式)的圆括号后面不能加分号,例如:

```
if(temp<0);                        //此处多加了分号,与设计初衷不符
    temp=-temp;
```

这样写语法检查时不会报错,但由于在 if(temp<0)后面加了分号,表示空操作,即当 temp<0 成立时执行空操作而不是执行 temp=－temp,因此代码的实现已经和原来的设计相背离了。

代码中用虚线框是为了突出选择结构的语句。

3.3.2 双分支 if 语句

【例 3-9】 成绩合格问题。(nbuoj1058)

输入一个整数表示课程成绩,判断学生成绩是否合格:当分数大于等于 60 分时,输出合格信息,小于 60 分的,输出不合格信息。

```
#include<stdio.h>
int main()
{
  int grade;
  scanf("%d",&grade);
  if(grade>=60)                    //判断成绩是否大于等于 60
    printf("pass\n");              //若条件成立,则输出及格信息
  else                             //否则说明表达式里的条件不成立,即成绩小于 60
    printf("failure\n");           //因此输出不合格信息
```

```
    return 0;
}
```

运行结果：

```
59 ↵
failure
```

本例用到双分支的 if 语句，即 if…else 语句，其流程图见图 3-3。

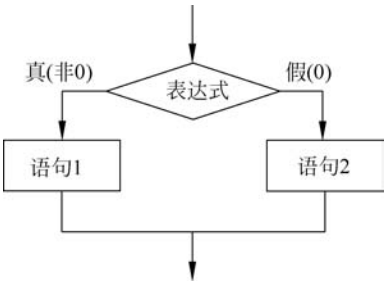

图 3-3　if…else 语句流程图

if…else 语句是选择结构的标准使用形式，其用法见表 3-9。

表 3-9　if…else 语句的用法

语　法	示　例	说　明

if(表达式)
　语句 1
else
　语句 2
	if(grade>=60)
	　printf("pass\n");
	else
	　printf("failure\n);
		(1) 先计算表达式的值，若值为真，则执行语句 1；否则执行语句 2。
		(2) 无论表达式的值为真还是假，只执行语句 1 和语句 2 中的某一个，不会两个都执行。
		(3) else 子句不能单独使用，它前面必须要有 if 配对使用

【例 3-10】 分段函数。(nbuoj1042)

输入一个整数 x，计算以下分段函数的值，输出保留 2 位小数。

$$y=\begin{cases} x^2-2 & x\geqslant 0 \\ \sqrt{5-x} & x<0 \end{cases}$$

```
#include<stdio.h>
#include<math.h>
int main()
{
  int x;
  double y;
  scanf("%d",&x);
  if(x>=0)
    y=x*x-2;
  else
    y=sqrt(5-x);
  printf("%.2f\n",y);
  return 0;
}
```

运行结果：

```
4 ↵
14
```

本例中，x 的取值范围有两个区间，x≥0 区间以及 x<0 区间。当 x≥0 时，取值范围如图 3-4 所示，而

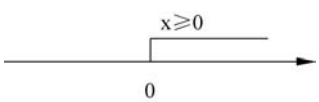

图 3-4　例 3-10 中 x≥0 的取值范围

当 x＜0 时，取值范围刚好是表达式 x≥0 的取反，即除去 x≥0 以外的所有区间。因此，在 if…else 语句里，如果表达式 x≥0 不成立，则说明 x 是小于 0 的，就不需要再刻意强调 x＜0 这一条件了。

【例 3-11】 两数求大值。(nbuoj1061)

从键盘输入任意两个整数 a 和 b，求出其中较大数的数值并输出。

可以再定义一个变量 max，用来存放 a，b 中较大数的值。算法流程图见图 3-5。

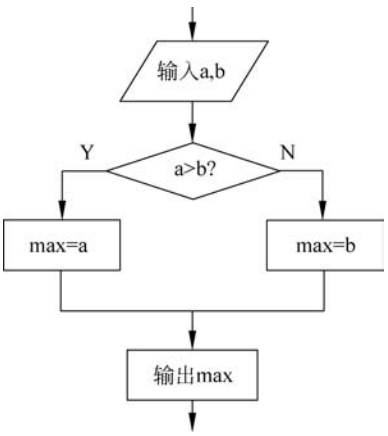

图 3-5 两数求大值的流程图

```
#include<stdio.h>
int main()
{
  int a,b,max;
  scanf("%d%d",&a,&b);
  if(a>b)                    //将较大数存到变量 max 中
    max=a;
  else
    max=b;
  printf("%d\n",max);
  return 0;
}
```

运行结果：

```
3 8↵
8
```

通过 if…else 语句对两数进行比较，将较大数的值存到变量 max 中并输出。

前面几个例子用了单一的关系运算符，下面这个例子用关系运算符和逻辑运算符构成一个逻辑表达式。

【例 3-12】 判断是否为英文字母。(nbuoj1046)

任意输入一个字符，判断其是否为英文字母，是则输出 YES，否则输出 NO。

```
#include<stdio.h>
int main()
{
  char ch;
  scanf("%c",&ch);
  if(ch>='a'&&ch<='z'||ch>='A'&&ch<='Z')   //判断 ch 是否属于小写字母或大写字母
    printf("YES\n");
  else
    printf("NO\n");
  return 0;
}
```

运行结果：

```
S↵
```

```
YES
```

【例 3-13】 是否闰年。(nbuoj1072)

输入一个整数 year 表示某一年,判断这一年是否闰年,是则输出 yes,否则输出 no。闰年的条件是符合下面两个条件之一:①能被 4 整除,但不能被 100 整除,如 2020;②能被 400 整除,如 2000。

```
#include<stdio.h>
int main()
{
  int year;
  scanf("%d",&year);
  if((year%4==0&&year%100!=0)||year%400==0)  //判断闰年的两个条件组成一个逻辑表达式
    printf("yes\n");
  else
    printf("no\n");
  return 0;
}
```

运行结果:

```
2021↵
no
```

对于关系比较复杂的逻辑表达式,建议适当地利用圆括号来明确运算的优先次序。

3.3.3 多分支 if 语句

实际编程时需要判断的条件往往不止两个,多分支的 if 语句可以解决多条件判断的问题。比如下面这个程序需要对学生成绩评级。

【例 3-14】 三级制成绩评级。(nbuoj1059)

输入一个整数形式的学生成绩(百分制),按以下规则计算并输出相应等级:[80,100]分为 A 等,[60,79]分为 B 等,小于 60 分为 C 等。(成绩范围为 0～100 分。)

```
#include<stdio.h>
int main()
{
  char ch;
  int score;
  scanf("%d",&score);        //输入百分制的学生成绩
  if(score>=80)
    ch='A';
  else if(score>=60)
    ch='B';
  else
    ch='C';
  printf("%c\n",ch);         //输出对应的成绩的等级
  return 0;
}
```

运行结果：

```
99↵
A
```

这个例子用到了多分支 if 语句，即 if…else if 语句，其流程图见图 3-6。

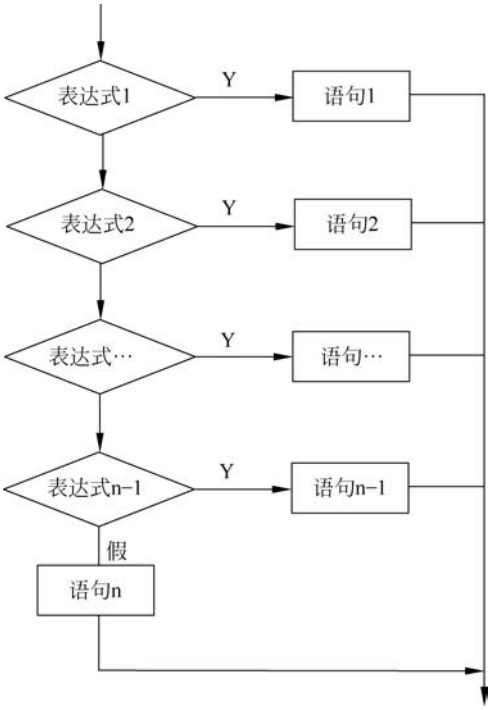

图 3-6　if…else if 语句流程图

if…else if 语句的用法见表 3-10。

表 3-10　if…else if 语句的用法

语　　法	示　　例	说　　明

if(表达式 1)
　语句 1
else if(表达式 2)
　语句 2
　…
else if(表达式 n－1)
　语句 n－1
else
　语句 n
	if(score＞＝80)
	　ch＝'A';
	else if(score＞＝60)
	　ch＝'B';
	else
	　ch＝'C';
		(1) 先计算表达式 1 的值，若表达式值为真，则执行语句 1，否则进行下一步判断。
		(2) 若表达式 2 的值为真，则执行语句 2，否则进行下一步判断。
		(3) 若前面所有的表达式都为假，则执行语句 n。
		(4) 只会执行语句 1 到语句 n 中的某一个，不会执行多个

如果有 n 个分支结构，则用最开始的 if 和最后一个 else 各处理一个分支，中间再连续写 n－2 个 else if 语句来处理剩余的 n－2 个分支。

【例 3-15】 单个字符类型判断。(nbuoj1049)

输入任意一个字符，判断该字符是小写字母、大写字母、数字字符或者其他类型字符，输

出对应提示信息。

```
#include <stdio.h>
int main()
{
  char ch;
  scanf("%c",&ch);
  if(ch>='A'&&ch<='Z')
    printf("upper\n");          //大写字母则输出 upper
  else if(ch>='a'&&ch<='z')
    printf("lower\n");          //小写字母则输出 lower
  else if(ch>='0'&&ch<='9')
    printf("digit\n");          //数字字符则输出 digit
  else
    printf("other\n");          //其他字符则输出 other
  return 0;
}
```

运行结果：

```
d↵
lower
```

本例运用字符 ASCII 码值的规律进行判断，一共有四个分支，用第一个 if 处理大写字母，最后一个 else 处理其他字符，中间的两个 else if 分别处理小写字母和数字字符。

多分支 if…else if 语句并不是新的语句类型，它依然是普通的 if 语句，只是刚好有另外一条 if 语句作为 else 的子句，而且这条 if 语句又有另外一条 if 语句作为它自己的 else 子句，以此类推。如本例的多分支选择语句可以写成如图 3-7 所示的形式。

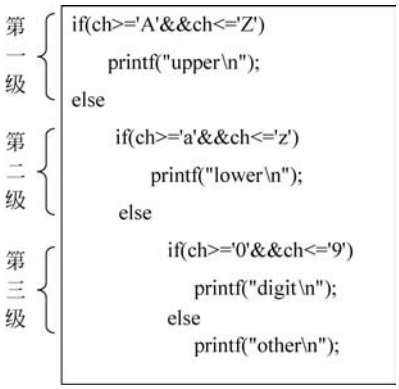

图 3-7　多分支 if 语句的缩进式书写

但是一般在书写多分支 if…else if 语句时不会对它进行缩进，避免需判断的条件太多引起过度缩进，而是如例 3-15 中的代码那样，把 else 与它后面的 if 写在同一行上，即：

```
if(表达式)
  语句
else if(表达式)
  语句
…
else if(表达式)
  语句
else
  语句
```

- 使用 if…else if 语句时，各个分支的条件一定要按照某种顺序书写。这样做既可以使程序条理清晰，而且也不容易出错。
- 用多分支 if…else if 语句时，最后的一个 else 语句不是总会出现的。

3.3.4 带复合语句的 if 语句

前面出现的 if 语句的各种形式中，语句部分都只有一条语句，如果想用 if 语句处理两条或多条语句，就需要使用复合语句。复合语句也称为语句块，通过在一组语句前后放置大括号，可以强制编译器将其作为一条语句来处理。

标准 if…else 语句中复合语句的书写形式见表 3-11。

表 3-11 选择结构中复合语句的用法

语 法	示 例	说 明

if(表达式)
{
语句 1(系列语句)
}
else
{
语句 2(系列语句)
}
	if(sum > 10000)
	{
	discount = 0.68;
	printf("Golden Card Discount = %.2f\n",
	discount);
	}
	else
	{
	discount = 0.88;
	printf("Ordinary card Discount = %.2f\n",
	discount);
	}
		(1) 如果消费金额超过 1 万，则给出 0.68 的折扣，并显示金卡折扣为 0.68。
		(2) 如果消费金额没有超过 1 万，则给出 0.88 的折扣，并显示普卡折扣为 0.88。
		(3) 当表达式为“真”或“假”时，对应的复合语句要么全部执行，要么全部不执行

【例 3-16】 两整数排序。(nbuoj1062)

输入两个整数，按从小到大的顺序输出这两个数。

```
#include <stdio.h>
int main()
{
  int a,b,t;
  scanf("%d%d",&a,&b);
  if(a>b)
  {                    //构成复合语句的左大括号
   t=a;                //①
   a=b;                //②
   b=t;                //③
  }                    //构成复合语句的右大括号
  printf("%d %d\n",a,b);   //已将较小数存入 a,较大数存入 b,输出
  return 0;
}
```

运行结果：

```
9 3↵
3 9
```

本例通过三变量法实现两数的交换，当 a>b 时，将较小数放入 a 变量，而将较大数放入 b 变量。三变量法由三条语句构成一个完整的交换过程，当 a>b 条件成立时，①②③这三条语句都被依次执行，而当条件 a>b 不成立时，①②③这三条语句都不被执行。因此需要

使这三条语句组成一个整体，即将这三条语句组成一个复合语句。

如果本例的语句写成如下形式：

```
if(a > b)
  t = a;
  a = b;
  b = t;
```

此时读者可以分析一下，分别用 9 和 3 以及 3 和 9 这两组数据分别进行测试，看运行结果将会有怎样的变化。

复合语句的主要作用是把多个语句组成一个可执行的单元。

【例 3-17】 三整数排序。(nbuoj1065)

输入三个整数 a，b，c，按从小到大的顺序输出这三个数。

```
#include < stdio.h >
int main()
{
  int a,b,c,t;
  scanf("%d%d%d",&a,&b,&c);
  if(a > b)
    {t = a;a = b;b = t;}              //实现 a、b 交换，较小数存入 a，较大数存入 b
  if(a > c)
    {t = a;a = c;c = t;}              //实现 a、c 交换，较小数存入 a，较大数存入 c
  if(b > c)
    {t = b;b = c;c = t;}              //实现 b、c 交换，较小数存入 b，较大数存入 c
  printf("%d %d %d\n",a,b,c); //最终变量 a、b、c 中的数据按从小到大顺序存放
  return 0;
}
```

运行结果：

```
3 9 7↵
3 7 9
```

解决三个数的排序问题，有许多种方案。本例采用了数据交换的方法，将三个变量相互间各做一次比较，根据比较结果将最小数放入 a，中间数放入 b，最大数放入 c。算法思想表示如下。

if(a > b)将 a 与 b 交换，则 a 是 a、b 中的较小者。

if(a > c)将 a 与 c 交换，则 a 是 a、c 中的较小者，并且 a 是三者中的最小者。

if(b > c)将 b 与 c 交换，则 b 是 b、c 中的较小者，也是三者中的中间数。

复合语句一般出现在选择和循环语句中。

3.4 条件运算符和条件表达式

C 语言中有条件运算符“?:”，它由两个符号“?”和“:”组成，可以用来构造条件表达式。条件运算符的使用形式如表 3-12 所示。

表 3-12　条件运算符的使用形式

语　法	表达式 1? 表达式 2：表达式 3
示　例	(a>b)?a:b
说　明	(1) 先求解表达式 1 的值，如果值为真，则计算表达式 2 的值来作为整个表达式的值； (2) 若表达式 1 的值为假，说明条件不成立，则计算表达式 3 的值来作为整个表达式的值； (3) 样例中的(a>b)? a:b 是一个条件表达式为，其执行方式为：如果 a>b 成立，则表达式取 a 的值，否则表达式取 b 的值

【例 3-18】 两数求大值。(nbuoj1061)

输入任意两个整数，用条件运算求出其中较大的数值并输出。

```
#include<stdio.h>
int main()
{
  int a,b,max;
  scanf("%d%d",&a,&b);
  max=(a>b)?a:b;      //用条件表达式,将 a、b 中较大的数值保存到变量 max 中
  printf("%d\n",max); //输出保存在 max 变量中的较大值
  return 0;
}
```

运行结果：

```
6 19↵
19
```

再看一个与字符操作有关的例子。

【例 3-19】 大写字母变小写字母。(nbuoj1430)

用条件运算表达式实现将大写字母转换为小写字母，如果是其他字符则保持不变。

```
#include<stdio.h>
int main()
{
  char ch,new_ch;
  scanf("%c",&ch);                          //输入一个字符
  new_ch=(ch>='A'&&ch<='Z')?ch+32:ch;       //对大写字母加 32 可得到对应的小写形式
  printf("%c\n",new_ch);
  return 0;
}
```

运行结果 1：

```
A↵
a
```

运行结果 2：

```
#↵
#
```

本例先判断 ch 是否是大写字母，如果是大写字母，则根据 ASCII 码的排列顺序可知，加 32 就可以得到该字母对应的小写形式，否则保持不变。

条件运算表达式语句相当于一个简单的选择结构。一般当 if 语句中需要执行的语句为赋值语句，并且两个分支都是给同一个变量赋值的时候，可以用条件表达式来达到相同的效果，如果语句比较复杂的情况下不建议刻意使用条件表达式。

选择结构的嵌套

3.5 选择结构的嵌套

在 if 语句中又包含一个或多个 if 语句的形式称为 if 语句的嵌套。C 语言中 if 语句嵌套的形式比较灵活，图 3-8 给出了 if…else 语句二重嵌套的一般形式。

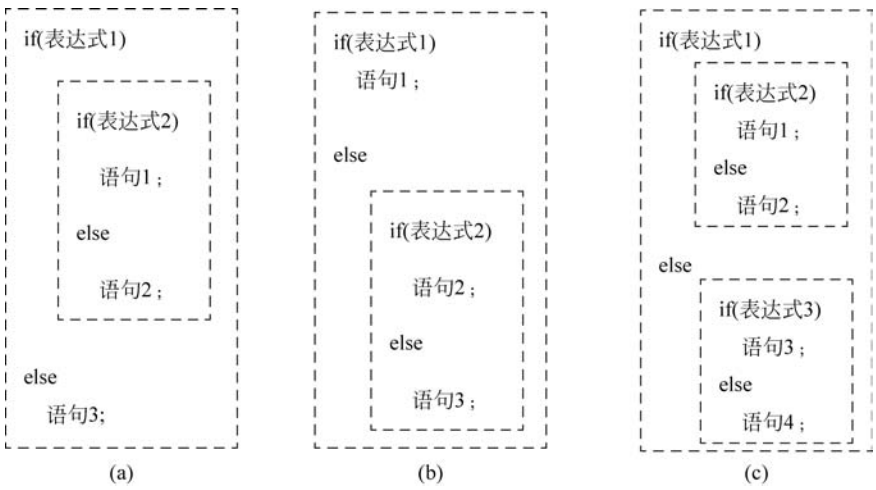

图 3-8 if…else 语句二重嵌套的一般形式

【例 3-20】 输入两个整数，比较它们的大小关系。

```
#include<stdio.h>
int main()
{
  int a,b;
  scanf("%d%d",&a,&b);
  if(a!=b)
    if(a>b)
      printf("%d>%d\n",a,b);
    else
      printf("%d<%d\n",a,b);
  else
    printf("%d=%d\n",a,b);
  return 0;
}
```

运行结果 1：

```
7 12↵
```

```
7＞12
```

运行结果 2：

```
9 －8↵
9＞－8
```

本例先将 a 与 b 的关系划分成两类，即“不等于”和“等于”，然后在“不等于”的情况下再进一步判断是“大于”还是“小于”，因此，在“不等于”的情况下出现了嵌套的 if 语句。

嵌套的 if 语句虽然占据多个书写行，但如果不构成复合语句的话，就无须用大括号括起来。

在嵌套的 if 语句中，特别要注意 if 与 else 的搭配问题，使用不当容易产生二义性。比如把嵌套关系写成如下形式：

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

设计者把 else 与第 1 个 if 写在同一列，试图以此表示它们是匹配的。但根据 if 与 else 的配对规则，else 实际上是与第 2 个 if 配对的（else 总是与离它最近的未匹配过的 if 匹配）。可见，嵌套内的 if 语句既可以是 if 语句形式也可以是 if…else 语句形式，这就会出现多个 if 和多个 else 重叠的情况，此时要特别注意 if 和 else 的配对问题，一般地，else 总是与它前面一个最近的未匹配过的 if 配对，如图 3-9 所示。

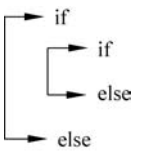

图 3-9　if 和 else 的配对原则

当然，为了保险起见，也可以用大括号来强制确定配对关系。对于上述例子，如果设计者的意图是让 else 和第 1 个 if 匹配，那么可做如下处理。

```
if(表达式 1)
{
  if(表达式 2)
    语句 1;
}
else
    语句 2;
```

C 语言不是以书写格式来分隔语句的，而是由逻辑关系决定的。

【例 3-21】 求一元二次方程 $ax^2+bx+c=0$ 的根。

本题可以分为以下两种情况来考虑。

（1）若 a 不等于 0，$d=b^2-4ac$，则需要进一步考虑 d 的取值，即：

① 若 $d\geqslant 0$，方程有两个实根：$x_{1,2}=\dfrac{-b\pm\sqrt{d}}{2a}$。

② 若 $d<0$，方程有两个虚根：$x_{1,2}=\dfrac{-b\pm\sqrt{-d}i}{2a}$。

（2）若 a 等于 0，则为非二次方程，因此根 $x=-c/b$。

流程图如图 3-10 所示。

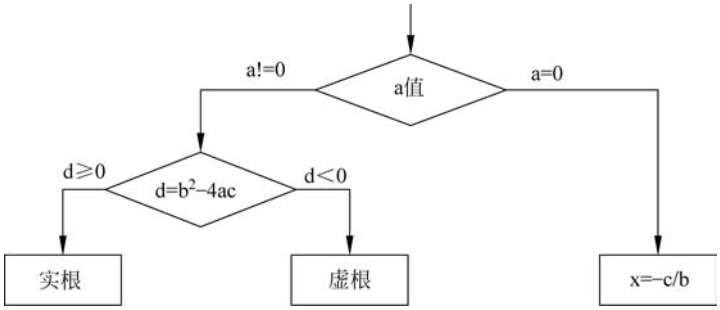

图 3-10　求一元二次方程根的流程图

```
#include <stdio.h>
#include <math.h>
int main()
{ double a,b,c,d,t1,t2;
  printf ("Input a b c:\n" );
  scanf("%lf%lf%lf",&a,&b,&c);
  if (a!=0)
  {
   d=b*b-4*a*c;
   t1=(-b)/(2*a);
   t2=sqrt(fabs(d))/(2*a);
      if(d>=0)
        printf("Two real roots:\nx1=%.1f\nx2=%.1f\n",t1+t2,t1-t2);
      else
        printf("Two complex roots:\nx1=%.1f+%.1fi\nx2=%.1f-%.1fi\n",t1,t2,t1,t2);
  }
  else
  printf("It's not quadratic,x=%.1f\n",-c/b);
  return 0;
}
```

运行结果 1：

```
Input a b c:
0 3 4↵
It's not quadratic,x=-1.3
```

运行结果 2：

```
Input a b c:
2 3 4↵
Two complex roots:
x1=-0.8+1.2i
x2=-0.8-1.2i
```

运行结果 3：

```
Input a b c:
```

```
1 4 1↵
Two real roots:
x1 = -0.3
x2 = -3.7
```

利用 C 语言提供的关系运算、逻辑运算和 if 语句，可以完成各种复杂的选择结构的控制流程描述。然而，为了保证程序的可读性，不提倡使用嵌套层次过多的运算。

用 switch 实现多分支选择

3.6 switch…case 语句

C 语言提供 switch…case 语句作为多分支选择结构的替代。switch…case 语句的用法如表 3-13 所示。

表 3-13 switch…case 语句的用法

语　法	示　例	说　明

switch(表达式)
{
case 常量表达式 1:
　　语句组 1;
　　break;
case 常量表达式 2:
　　语句组 2;
　　break;
　… …
case 常量表达式 n:
　　语句组 n;
　　break;
default:
　　语句组 n+1;
　　break;
}
	switch(grade)
	{
	case 'A':
	　　printf("80～100\n");
	　　break;
	case 'B':
	　　printf("60～79\n");
	　　break;
	case 'C':
	　　printf("<60\n");
	　　break;
	default:
	　　printf("Error\n");
	　　break;
	}
		(1) switch 后面的表达式通常是整型或字符型变量，也允许枚举类型数据。
		(2) case 后面的常量表达式起到标记作用，用来标志一个位置。
		(3) switch 语句的流程是：先计算 switch 后面表达式的值，然后用此值依次与各个 case 后的常量表达式比较，若与某个 case 后面的常量表达式的值相等，就执行这个 case 后面的语句组，执行后遇到 break 就退出 switch 语句；若表达式的值与所有 case 后面的常量表达式都不相等，则执行 default 后面的语句组 n+1，然后退出 switch 语句

【例 3-22】 编制菜单程序：在屏幕上显示问候信息表，根据用户的选择，显示不同的问候信息。

```
include <stdio.h>
int main()
{
  char ch;
  printf("1 Morning\n");
  printf("2 Afternoon\n");
  printf("3 Evening\n");
  printf("Input your choice:\n");
  ch = getchar();   //从键盘输入用户的选择
```

```
    switch(ch)      //根据选择进行不同的处理
    {
      case '1':printf("Good morning\n");break;
      case '2':printf("Good Afternoon\n");break;
      case '3':printf("Good Evening\n");break;
      default:printf("Selection Error\n");
    }
  return 0;
}
```

运行结果：

```
1 Morning
2 Afternoon
3 Evening
Input your choice
2↵
Good Afternoon
```

当用户输入数字2时，switch语句找到匹配的case子句，并执行后面的语句，输出问候语“Good Afternoon”，然后执行break语句，跳出switch语句。

使用switch…case语句需要注意以下几点。

(1) 关键字case和后面的常量表达式之间有空格间隔。

(2) default一般总是放在最后面，这时，default后面不需要break语句。default部分不是必需的。例如，在例3-22中如果没有default部分，当switch后面的表达式的值与case后面的常量表达式的值都不相等时，则不执行任何分支，直接退出switch语句。

(3) 各个case常量表达式不一定要按其值的大小顺序来书写语句，但要求各个case后的常量表达式必须是不同的值，以保证分支选择的唯一性。例如：

```
case '2':语句2;break;
case '1':语句1;break;
case '3':语句3;break;
case '1':语句4;break;
```

前3个case语句都是合法的，但最后一个case语句与第2个case语句的常量表达式的值相同了，这是不允许的。

建议尽量按照常量表达式值的大小顺序来书写语句，使语句条理更清晰。

(4) 如果在case后面包含多条执行语句，不需要加大括号。在进入case后，会自动顺序执行当前case后面的所有执行语句。

(5) 只有default中的break语句可有可无，而其余各分支中的break语句有或无时程序的流程是完全不同的，例如，在例3-22中如果case '1'，case '2'，case '3'后面没有break语句，则当用户输入数字1以后，程序的输出结果为：

```
Good Morning
Good Afternooon
Good Evening
Selection Error
```

这是因为 case 后面的常量表达式只起到标记的作用，而不起条件判断的作用。因此，一旦与 switch 后圆括号内表达式值匹配，就从这个标记开始执行，而且执行完一个 case 后面的语句后，若没有遇到 break 语句，会自动进入下一个 case 继续执行，而不再判断是否匹配。因此，若想执行一个 case 分支后立即跳出 switch 语句，就必须在此分支的最后添加一个 break 语句。

(6) 多个 case 可以共用一组执行语句。见例 3-23。

【例 3-23】 五级制成绩评级。(nbuoj1060)

输入一个整数表示百分制成绩(0～100)，将其转换为对应的等级制并输出。对应规则为：[90,100]分为 A,[80,89]分为 B,[70,79]分为 C,[60,69]分为 D,小于 60 分为 E。

如果直接对分数进行判断：100 分为 A,99 分为 A,……,1 分为 E,0 分为 E,则会有 101 个 case 分支，显然是不合理的。需要考虑如何减少 case 分支而又不影响程序的功能。首先将分数除以 10 取整，则将得到 10,9,8,7,6,5,4,3,2,1,0 这样 11 个分支，其中，10 和 9 这两个分支代表 90～100 分，分支 8 代表 80～89，分支 7 代表 70～79，分支 6 代表 60～69，分支 5,4,3,2,1,0 代表小于 60 分的。这样，分支数就大大减小了。最终的程序如下。

```
#include <stdio.h>
int main()
{
  int score;
  char grade;
  scanf("%d",&score);
  switch(score/10)
  {
    case 10:
    case 9: grade = 'A';break;     //case 10 和 case 9 两个 case 共用一组语句
    case 8: grade = 'B';break;
    case 7: grade = 'C';break;
    case 6: grade = 'D';break;
    case 5:
    case 4:
    case 3:
    case 2:
    case 1:
    case 0: grade = 'E';           //case 5 到 case 0 的六个 case 共用一组语句
  }
  printf("%c\n",grade);
  return 0;
}
```

运行结果：

```
90↵
A
```

本例还可以进一步减少 case 分支，如将 5,4,3,2,1,0 这 6 个分支都归入到 default 分支，则对应代码段可改写成如下形式。

```
switch(grade)
{
  case 10:
  case 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");break;
}
```

下面这个例子也是多个 case 子句共用一组语句的。

【例 3-24】 模拟万年历。(nbuoj1073)

输入两个整数表示年和月的数值,打印出这一年的这一个月的天数。如用户输入的信息是 2020 年的 2 月,则打印出该月的天数为 29。假设数据都是有效的。

根据历法,凡 1、3、5、7、8、10、12 月每月为 31 天,凡 4、6、9、11 月每月为 30 天;2 月份闰年 29 天,平年 28 天。闰年的判断条件需满足以下两个条件中的一个即可:①能被 4 整除但不能被 100 整除;②能被 400 整除。

```
#include<stdio.h>
int main()
{
  int year,month,days;
  scanf("%d%d",&year,&month);      //输入两个整数表示年和月
  switch(month)
  {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
      days=31;
      break;
    case 4:
    case 6:
    case 9:
    case 11:
      days=30;
      break;
    case 2:                                          //2 月的天数要根据闰年还是平年来定
    if(year%4==0&&year%100!=0||year%400==0)
      days=29;
    else
      days=28;
    break;
  }
```

```
    printf("%d\n",days);
    return 0;
}
```

运行结果：

```
2020 2 ↵
29
```

关于闰年的判断可以有很多的写法，请读者根据自己所掌握的知识尝试不同的设计思路。

3.7 实例研究

第2章的实例研究中介绍了一个加、减、乘、除的简单程序，输入两个数据，根据固定的模式计算相应的结果。如果希望程序灵活一些，能根据用户输入的运算符号来决定进行加、减、乘、除中的哪一种运算，则需要对输入的运算符号进行判断。选择结构提供了这方面的支持。

3.7.1 四则运算

【例 3-25】 简单计算器。(nbuoj1084)

设计一个简单计算器程序，可根据输入的表达式，对两个数进行加、减、乘、除运算。输入形式为AopB，其中，A和B表示参加运算的两个浮点数，op代表算术运算符+、-、*、/中的一种，如2+5。计算结果保留两位小数。假设不会出现除数为0的情况。

```
#include<stdio.h>
int main()
{
    char op;
    double a,b,answer;
    scanf("%lf%c%lf",&a,&op,&b);      //输入计算公式，变量op存储运算符
    switch(op)                        //判断运算符op
    {
        case '+':answer=a+b;break;    //若是加号，则执行加法运算
        case '-':answer=a-b;break;
        case '*':answer=a*b;break;
        case '/':answer=a/b;break;
    }
    printf("%.2f\n",answer);
    return 0;
}
```

运行结果：

```
7/2 ↵
3.50
```

本例所设计的计算器是由用户出题，由机器来回答，机器按照运算规则计算得到的肯定是正确答案，这个工作原理跟日常使用的计算器类似。

下面进一步模拟小学生四则运算的学习过程，需要在给出题目后，由用户来回答，然后由机器来判断用户的回答是否正确。

【例 3-26】 小学生四则运算练习系统。编制一个可以完成加、减、乘、除运算的程序，输入一个算术表达式，如果运算符号是“+”，则执行加法操作；为“-”，则执行减法操作；为“*”，则执行乘法操作；为“/”，则执行除法操作。给出表达式后，从键盘再输入一个运算结果，若答案正确则输出正确信息，否则给出错误提示。（假设除法都是能整除的。）

```
#include <stdio.h>
int main()
{
  int a,b,user_ans,res;              //user_ans 表示用户答案,res 表示标准答案
  char op;
  scanf("%d%c%d",&a,&op,&b);    //输入表达式
  switch(op)                                    //先计算标准答案,存入变量 res
  {
    case '+': res = a + b;break;
    case '-': res = a - b;break;
    case '*': res = a * b;break;
    case '/':
      if(b!= 0)
      res = a/b;
      else
      printf("Division by zero,ERROR\n");  //若除数为 0 则给出错误提示
      break;
  }
  if(!(op == '/'&&b == 0))
  {
    printf(" = ");
    scanf("%d",&user_ans);          //输出等号,提醒用户输入答案
    if(res == user_ans)             //输入用户答案
      printf("Sucess:)\n");         //将用户答案 user_ans 与标准答案 res 进行比较
    else
      printf("Error:(\n");
  }
  return 0;
}
```

运行结果 1：

```
4 + 5 ↵
 = 9 ↵
Success:)
```

运行结果 2：

```
7 * 8 ↵
 = 67 ↵
Error:(
```

本例用了两个选择结构，一个由 switch 语句构成，判断算术运算符号然后计算出对应

的标准答案；一个由嵌套的 if 语句构成，在此输入用户答案，并将用户答案与标准答案进行对比，然后输出相应的提示信息。

3.7.2 随机数

【例 3-27】 随机数比大小。由系统随机产生两个随机数，比较两者的大小。

```
#include<stdio.h>
#include<stdlib.h>                    //包含库函数 rand 和 srand 的原型
#include<time.h>
int main()
{
  int a,b;
  srand((unsigned)time(NULL));        //使随机函数 rand 的值随时间变化
  a=rand();                           //调用随机数函数 rand 产生一个数存入 a
  b=rand();                           //调用随机数函数 rand 产生一个数存入 b
  if(a>b)
    printf("%d>%d\n",a,b);
  else
    printf("%d<%d\n",a,b);
  return 0;
}
```

运行结果：

```
17035>14484
```

本题用到了随机数的产生，这是比较实用的一个功能，关于随机数的介绍如下。

在计算机中并没有一个真正的随机数发生器，但是可以做到使产生的数字重复率很低，看起来好像是真正的随机数，实现这一功能的程序叫伪随机数发生器。

不管用什么方法实现随机数发生器，都必须给它提供一个名为“种子”的初始值。而且这个值最好是随机的。现在的 C 编译器都提供了一个基于 ANSI C 标准的伪随机数发生器函数，用来生成随机数，它们就是 rand 和 srand 函数。这两个函数的工作过程如下。

(1) 首先给 srand 提供一个种子，它是一个 unsigned int 类型，其取值范围为 0～65 535，如语句 srand((unsigned)time(NULL))。有了 srand()函数，程序每次运行时产生的随机数都会不同；如果删除这条语句，则每次运行结果都是一样的。

(2) 然后调用 rand()函数，它会返回一个随机数(0～32 767)。

掌握了随机数的产生方法后，就可以对小学生四则运算练习系统进行进一步修改，由系统随机产生两个整数来参加四则运算。

【例 3-28】 改进的小学生四则运算练习系统。生成两个 100 以内的随机数，从键盘输入一个运算符号(+，-，* 或/)，对这两个随机数进行对应的计算并输出结果。

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

```
    int a,b,answer;
    char op;
    srand((unsigned)time(NULL));
    a = rand() % 100;            //生成 0～99 的随机数
    b = rand() % 100 + 1;        //生成 1～100 的随机数
    printf("请输入一个算术运算符号( + , - , * 或/)");
    scanf(" %c",&op);            //输入算术运算的符号
    switch(op)                   //根据运算符号执行对应的计算
    {
      case '+':answer = a + b;break;
      case '-':answer = a - b;break;
      case '*':answer = a * b;break;
      case '/':answer = a/b;break;
    }
    printf(" %d %c %d = %d\n",a,op,b,answer);
    return 0;
}
```

运行结果：

```
请输入一个算术运算符号( + , - , * 或/):/↵
95/37 = 2
```

本例运行时只需输入运算符(＋,-, * 或/),而参加运算的两个数由程序中通过 rand()函数自动生成。

由于 C 程序可能会在不同的集成环境下运行,所以在使用时要特别注意相关函数在不同环境下是否适用。本书代码都在 VC6.0 环境下运行,因此本题采用 rand()函数。

3.8 习　　题

选择结构常见错误解析

3.8.1 选择题

1. C 语言中,关系表达式和逻辑表达式的值是(　　)。

   A. 真或假　　B. 0 或 1　　C. T 或 F　　D. True 或 False

2. 设 a 为整型变量,不能正确表达数学关系 10＜a＜15 的 C 语言表达式是(　　)。

   A. 10＜a＜15

   B. a＝＝11||a＝＝12||a＝＝13||a＝＝14

   C. !(a＜＝10)&&!(a＞＝15)

   D. a＞10&&a＜15

3. 如果 int a＝3,b＝4; 则条件表达式 a＜b? a:b 的值是(　　)。

   A. 3　　B. 4　　C. 0　　D. 1

4. 逻辑运算符两侧运算对象的数据类型(　　)。

   A. 只能是 0 或 1　　B. 只能是 0 或非 0 正数

   C. 只能是整型或字符型数据　　D. 可以是任何类型的数据

5. 在嵌套使用 if 语句时,C 语言规定 else 总是(　　)。

A. 和之前与其具有相同缩进位置的 if 配对

B. 和之前与其最近的 if 配对

C. 和之前与其最近的且不带 else 的 if 配对

D. 和之前的第 1 个 if 配对

6. 多分支选择语句 switch(表达式)中的“表达式”不允许是(　　)。

A. 整型变量　　B. 字符型变量　　C. 常量表达式　　D. 浮点型变量

7. 下列关系表达式中结果为假的是(　　)。

A. 3<=7　　B. (a=2*2)==2

C. 0!=1　　D. y=2+2

8. 判断字符 ch 是否是大写英文字母,正确的逻辑表达式是(　　)。

A. 'A'<=ch&&ch<='Z'　　B. 'A'<=ch||ch<='Z'

C. A<=ch&&ch<=Z　　D. A≤ch≤Z

9. 在 C 语言中,紧跟在关键字 if 后一对圆括号里的表达式(　　)。

A. 只能是逻辑表达式　　B. 只能是关系表达式

C. 只能是逻辑表达式或关系表达式　　D. 可以是任意合法的表达式

10. 下列语句中,输出结果与其他语句不同的是(　　)。

A. if(a) printf("%d\n",x);　else printf("%d\n",y);

B. if(a==0) printf("%d\n",y);　else printf("%d\n",x);

C. if(a==0) printf("%d\n",x);　else printf("%d\n",y);

D. if(a!=0) printf("%d\n",x);　else printf("%d\n",y);

11. 在以下各语句序列中,能够将变量 x 和 y 中较大值赋值到变量 t 中的是(　　)

A. if(x>y) t=x; t=y;　　B. t=x; if(x>y) t=y;

C. t=y; if(x>y) t=x;　　D. if(x>y) t=y; else t=x;

12. 设 x,y,z,t 均为 int 型变量,则执行以下语句后,t 的值为(　　)。

```
x=y=z=1;
t=++x||++y&&++z;
```

A. 不定值　　B. 4　　C. 1　　D. 0

13. 对于条件表达式(k)?(i++):(i--)来说,其中的表达式 k 等价于(　　)。

A. k==0　　B. k==1　　C. k!=0　　D. k!=1

14. 以下程序的运行结果为(　　)。

```
#include<stdio.h>
int main()
{
  char c='a';
  if('a'<c<='z') printf("LOW");
  else printf("UP");
  return 0;
}
```

A. LOW　　B. UP

C. LOWUP　　D. 程序语法错误

15. 以下程序的运行结果为(　　)。

```
#include<stdio.h>
int main()
{ int a=100,x=10,y=20,flag1=5,flag2=0;
  if(x<y)
  if(y!=10)
  if(!flag1)a=1;
  else if (flag2)a=10;
  else a=-1;
  printf("%d\n",a);
  return 0;
}
```

A. −1　　B. 100　　C. 1　　D. 10

16. 以下程序的运行结果为(　　)。

```
#include<stdio.h>
int main()
{
  int s=15;
  switch(s/4)
  {
    case 1:    printf("One ");
    case 2:    printf("Two ");
    case 3:    printf("Three ");
    default:    printf("Over ");
  }
  return 0;
}
```

A. Three　　B. Over

C. Three Over　　D. One Two Three Over

3.8.2 在线编程题

NBUOJ 上的选择结构编程

1. 符号属性判断。(nbuoj1036)

输入任意一个浮点数 x,根据其符号属性,输出对应的 sign 值。

$$\text{sign}=\begin{cases}1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases}$$

2. 判断奇数偶数。(nbuoj1038)

写一程序判断输入的整数的奇偶性,若是奇数则输出 odd,若是偶数则输出 even。

3. 计算分段函数。(nbuoj1043)

输入浮点数 x,计算并输出下面分段函数 y 的值(保留两位小数)。

$$y=\begin{cases}(x+1)^2+2x+\dfrac{1}{x}, & (x<0)\\ \sqrt{x}, & (x\geqslant 0)\end{cases}$$

4. 第几象限。(nbuoj1044)

输入两个整数 x,y 值表示平面上的一个坐标点,判断该坐标点处于第几象限,并输出相应的结果,用数字 1,2,3,4 分别对应四个象限。假设坐标点不会处于 x 轴和 y 轴上。

5. 圆内圆外。(nbuoj1045)

有一个半径为 10 的圆,圆心坐标为(0,0),从键盘输入任意点的坐标(a,b),判断该点在圆内,在圆外,还是恰巧在圆周上。输出 in 表示在圆内,out 表示在圆外,on 表示在圆周上。

6. 单个字母大小写互换。(nbuoj1047)

输入一个字符,如果该字符是小写字母,则输出其大写形式。如果该字符是大写字母,则输出其小写形式。若是其他字符则原样输出。

7. 三数求大值。(nbuoj1064)

从键盘输入三个整数 x,y 和 z,求出最大数的值。

8. 平面上的三角形判断。(nbuoj1012)

输入三个数 a,b,c,请问以这三个数作为边长能否构成一个三角形? 如果可以构成三角形,则输出该三角形的面积,否则输出 Error。

9. 求 5 和 7 的整数倍。(nbuoj1070)

判断输入的正整数是否既是 5 又是 7 的整数倍。若是,则输出 yes,否则输出 no。

10. 鸡兔同笼。(nbuoj1066)

已知笼子里鸡和兔的总数量为 n,总的腿数为 m,请计算笼子里鸡的数目和兔的数目并输出;如果无解则输出 No answer。

11. 加油站加油。(nbuoj1078)

某加油站提供三种汽油和一种柴油,售价分别如下。

90 号汽油:5.14 元/升。

93 号汽油:5.54 元/升。

97 号汽油:5.90 元/升。

0 号柴油:5.13 元/升。

另外,加油站还提供“自助加油”和“协助加油”两个服务等级,如果是自助加油,则可以获得 5%的优惠;如果是工作人员协助加油,则只有 2%的优惠。输入三个浮点数分别表示加油量、油品类型(如 90)、加油类型(1 表示自助加油,2 表示协助加油),计算用户应付的金额。

12. 一元二次方程根。(nbuoj1081)

输入方程系数 a、b、c,求一元二次方程的根:$ax^2+bx+c=0$,假设 $b^2-4ac\geqslant 0$。

13. 求点的高度。(nbuoj1082)

假设有四个圆塔,圆心坐标分别为(2,2)(−2,2)(−2,−2)(2,−2),如图 3-11 所示。圆塔直径都为 1 米,圆塔高 50 米,其他都为平地(高度为 0)。输入任一坐标值(x,y),打印出该点的高度。

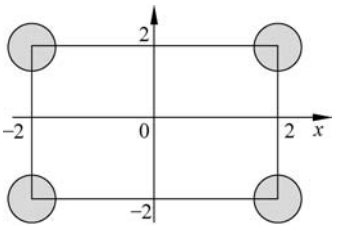

图 3-11 圆塔坐标图

14. 求 1～10 的英文单词。(nbuoj1083)

输入 1～10 中任意一个数字,输出相应的英文单词(首字母大写)。如果输入其他数字则输出 Error。

15. 石头剪刀布。(nbuoj1232)

CoCo 和 Tom 玩石头剪刀布的游戏,规则是石头砸剪刀、剪刀剪布、布包石头。他们用

数字代替手势来完成石头剪刀布的游戏。假设0表示石头,1表示剪刀,2表示布,每人在纸上写一个数字(数字范围局限于0、1、2),然后同时展示所写的数字,如果CoCo的数字胜出了,则输出Win,否则一律输出Lose。

16. 正方形还是圆形。(nbuoj1218)

首先从键盘读入一个浮点数x,然后再读入一个小写字母(s或c),如果读入的字母是s,则计算并输出正方形面积(此时x作为边长);如果读入的字母是c,则计算并输出圆面积(此时x作为半径)。

17. 今天星期几。(nbuoj1198)

输入一个正整数表示一个星期中的某一天,若此数字在[1,7]内,则输出对应英文星期名,否则输出错误提示。例如,输入2,则输出"Tuesday";输入7,则输出"Sunday";输入非法数值16,则输出"Illegal day"。(输出不包括双引号。)

18. 四数比大小。(nbuoj1230)

输入4个整数,将这4个数从大到小输出。

19. 计算个人所得税(老版算法)。(nbuoj1048)

已知个人所得税有如下的计算公式,输入一个浮点数表示某人本月的计税依据,输出本月应交的个人所得税,保留两位小数。计税依据如表3-14所示。

表3-14 计税依据

级 数	全月应纳税所得额(TL)	税率/%	速算扣除数	计算公式
1	TL≤1500	3	0	TL×0.03
2	1500≤TL≤4500	10	105	TL×0.1−105
3	4500≤TL≤9000	20	555	TL×0.2−555
4	9000≤TL≤35000	25	1005	TL×0.25−1005
5	35000≤TL≤55000	30	2755	TL×0.3−2755
6	55000≤TL≤80000	35	5505	TL×0.35−5505
7	TL≥80000	45	13505	TL×0.45−13505

计税方法:

(1) 计税依据=(工资、津贴等各项收入应发数之和)−(公积金、失业保险、养老保险、医疗保险之和)

(2) 全月应纳税所得额=计税依据−3500

(3) 所得税额=应纳税所得额×适用税率−速算扣除数

例如,某人当月9号计税依据为13500元,则其应交个人所得税税额为:

(13500−3500)×25%−1005=1495元。

20. 计算火车运行时间。(nbuoj1492)

根据火车的出发时间和到达时间,计算整个旅途经过的时间。输入为两个四位的整数,分别表示火车出发时间和到达时间(只考虑出发时间和到达时间是同一天的情况),其中前两位数表示小时(00～23),后两位表示分钟(00～59)。如出发时间为4点21分,到达时间为21点8分,则输入为0421 2108。输出整个旅途的时间,用小时和分钟表示。如针对输入数据0421 2108,输出为:16 hour 47 minute。