第 3 章 程序的控制结构

在第 2 章中介绍如何编写最基本的 C 语言程序，所编程求解的问题一般是按顺序逐步执行的，这种程序结构称为顺序结构，顺序结构就像一条流水线，将程序的语句逐一执行。计算机的程序也可以判断特定的条件，根据判断的结果，决定下一步要执行的命令，即程序产生了分支；还有一些工作需要重复很多次才能完成，需要对某条语句或某段程序重复执行多次。若实现这两类问题，就要用到本章介绍的分支结构和循环结构。

3.1 关系运算与逻辑运算

人们在工作生活中经常需要进行判断“某个事件是否发生”。用计算机程序进行“某个事件是否发生”的判断就是所谓的关系运算，也就是比较运算。某个事件是否发生，可能只有一个条件，也可能有多个条件。对于某一条件，有时需要采用其成立（或不成立）的反过程，也就是对条件“求反”。对于多个条件，有时要求多个条件只要成立一个即可，是一种“或”的连接关系；有时要求其同时成立，是一种“并且”的连接关系。这种“求反”和对多个条件“或”“并且”进行连接的运算称为逻辑运算。掌握关系运算和逻辑运算对于正确编写分支程序和循环程序是非常重要的。

3.1.1 关系运算


3-1 关系运算

【问题描述 3.1】 猜数字游戏，事先给出一个确定的数字（如 123），让游戏者从键盘输入他猜想的数字，如果游戏者猜对了则显示 Right，如果猜错了则显示 Wrong。

分析：本问题的关键是将游戏者猜的数字与给出的数字进行比较。在计算机程序中进行这种比较需要将事件符号化。假设程序设定的数字为 123，所猜的数字放入变量 guess，判断猜测的数字是否正确，就是比较 guess 的值是否等于 123，这就是所谓关系运算。关系运算是对两个表达式进行比较，并产生运算结果。

1. 关系运算符

C 语言提供的关系运算符有 6 种，如表 3.1 所示。

表 3.1　C 语言中的关系运算符

运　算　符	含　　义	运　算　符	含　　义
>	大于	<=	小于或等于
>=	大于或等于	==	等于
<	小于	!=	不等于

2. 关系表达式

(1) 关系表达式：用关系运算符将两个表达式(算术、关系、逻辑、赋值表达式等)连接起来所构成的表达式称为关系表达式。关系表达式的值是一个逻辑值，只有两种取值“真”或“假”。C 语言没有逻辑型数据，关系运算的结果是 int 型，以 1 表示关系表达式成立，代表“真”；0 表示关系表达式不成立，代表“假”。

(2) 关系运算的特点：

① 关系运算符都是双目运算符，并且都是从左向右结合。

② 关系运算符的优先级比算术运算符低，但都比赋值运算符高。在关系运算符内部，>、>=、<、<=运算符的优先级相同；==和!=两种运算符的优先级相同，且低于其他 4 种关系运算符。

例如：

c>a+b　　等价于　c>(a+b)　　关系运算符的优先级低于算术运算符。

a>b==c　等价于　(a>b)==c　>优先级高于==。

a==b<c　等价于　a==(b<c)　<优先级高于==。

a=b>c　　等价于　a=(b>c)　　关系运算符的优先级高于赋值运算符。

为了结构清晰易读，在程序设计中建议采用加括号方式。

3-2 逻辑运算

3.1.2　逻辑运算

平常所说的“求反”“或”“并且”分别对应逻辑非、逻辑或、逻辑与。

1. 逻辑运算符

C 语言中的逻辑运算符有 3 种，如表 3.2 所示。

表 3.2　C 语言中的逻辑运算符

运　算　符	含　　义
&&	逻辑与
||	逻辑或
!	逻辑非

2. 逻辑表达式

逻辑表达式：用逻辑运算符(逻辑与、逻辑或、逻辑非)将关系表达式或逻辑量连接起来，构成逻辑表达式。

逻辑表达式的值是一个逻辑量“真”或“假”。C 语言编译系统在给出逻辑运算结果时，以 1 代表“真”，以 0 代表“假”，但在判断一个表达式的结果是否为“真”时，以 0 代表“假”，以

非 0 代表“真”(即认为一个非 0 的数值是“真”)。

在一个逻辑表达式中如果包含多个逻辑运算符,则优先顺序如下。

(1) ! (非)→&&(与)→||(或),! 是三者中最高的。

(2) 逻辑运算符中的 && 和|| 的优先级低于关系运算符,而! 不仅高于关系运算符,而且高于算术运算符。

例如:

```
a>b&&x>y      等价于  (a>b)&&(x>y)
a==b||x==y    等价于  (a==b)||(x==y)
!a||a>b       等价于  (!a)||(a>b)
```

逻辑运算的真值表如表 3.3 所示。

表 3.3 逻辑运算的真值表

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

在逻辑表达式的求解中,并不是所有的逻辑运算符都被执行,只是在必须执行下一个逻辑运算符才能求出表达式的解时,才执行该运算符。

(1) “与”表达式:a&&b&&c,只要 a、b、c 有一个为“假”则整个表达式的值就为假。因此,只有 a 为真,才需要判别 b 的值;只有 a、b 都为真,才需要判别 c 的值。只要 a 为假,此时整个表达式已经确定为假,就不必判别 b 和 c;如果 a 为真,b 为假,则不必判断 c。

设 a=1, b=2, c=3, d=4, m=n=1,则执行表达式(m=a>b)&&(n=c>d)后,m、n 的值是多少?

说明: 因 a>b 为假(0),所以 m=0,赋值表达式 m=a>b 的值为 0,又因整个表达式是“与”表达式,所以不需要再计算 n=c>d,n 保持原值 1 未变,而整个表达式的结果就是 0。

(2) “或”表达式:a||b||c,只要 a、b、c 有一个为“真”则整个表达式的值就为真。因此,只要 a 为真,整个表达式就可以确定为真,就不必判断 b 和 c;只有 a 为假,才需要判断 b;当 a、b 都为假时,才需要判断 c。

【问题描述 3.2】 判断某一年是否为闰年。所谓闰年,是指符合下面两个条件之一:①能被 4 整除,但不能被 100 整除;② 能被 4 整除,又能被 400 整除。

分析: 对于条件①,能被 4 整除写作 year%4==0,不能被 100 整除写作 year%100!=0。要求两者同时满足,内部是一个“逻辑与”的关系,可合并为 year%4==0&&year%100!=0。

对于条件②,因为能够被 400 整除一定能被 4 整除,所以第二个条件可以简化为能够被 400 整除,写作 year%400==0。

条件①和条件②满足任何一个都是闰年,两者之间是“逻辑或”的关系。因此,判断闰年条件的逻辑表达式表示为((year%4==0)&&(year%100!=0))||(year%400==0)。表达式为“真”,闰年条件成立,是闰年;否则,不是闰年。

【练习 3.1】 输入3条边长,判断这3条边是否能构成一个三角形。假设用3个变量x、y、z存放3条边的边长,以下哪一个表达式是正确的?

A. x+y>z　　B. x+y>z, x+z>y, y+z>x

C. x+y>z && x+z>y && y+z>x　　D. x+y>z || x+z>y || y+z>x

3.2 分支结构

在生活中经常要做出选择,例如填报大学志愿时,有非常多的大学和非常多的专业,需要根据自己的高考分数、自己的兴趣爱好等很多因素综合做出选择。根据一定的条件做出不同的选择,在C语言中是用分支结构来实现的。

3.2.1 单分支结构

3-3 单分支 if

【例 3.1】 编写一个程序,判断某一年是否为闰年。

分析:判断闰年的表达式已在问题描述3.2中给出,如果表达式值为真,则是闰年,输出"xxxx年是闰年";否则,不输出。

这类问题是根据条件是否成立来决定是否执行相应的命令,其执行过程如图3.1所示。C语言提供了单分支选择结构解决上述问题,其一般形式:

```
if (表达式)
  语句;
```

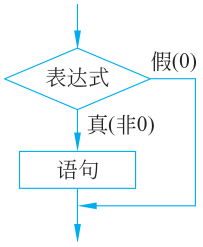

图 3.1 单分支选择结构条件语句执行过程

表达式是判断条件,只要表达式的值不为0,就认为条件成立。

例3.1的参考程序如下。

```
1.  #include <stdio.h>
2.  int main( )
3.  {
4.      int Year;
5.      printf("Input year:");
6.      scanf("%d", &Year);
7.      if (((Year%4==0)&&(Year%100!=0))||(Year%400==0)) //判断是否为闰年
8.          printf("%d年是闰年\n",Year);                  //条件成立,执行输出语句
9.      return 0;
10. }
```

说明:表达式结果非0,执行printf语句,否则程序结束。

【例 3.2】 简单的字符加密，输入一个小写字母字符，按 a→d，b→e，…，x→a，y→b，z→c 的加密规则对这个字符加密，然后输出加密后的字符。

分析：对于 a～w 这 23 个字母，加密方法很简单，直接对字符数据加 3 即可。关键是最后 3 个字母 x、y、z 怎么处理。x＋3 后对应的 ASCII 码字符是{，怎么才能让它变成 a 呢？字母就 26 个，并且是连续的，是不是减掉 26 就可以回到起始的字母 a 了？另外，还要注意怎样表示字母的范围，这里就用到了上面讲到的关系运算与逻辑运算。

例 3.2 的参考程序如下。

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      char ch;
5.      scanf("%c", &ch);
6.      if(ch>='a' && ch<='w')  printf("%c\n", ch+3);
7.      if(ch>='x' && ch<='z')  printf("%c\n", ch-23);
8.      return 0;
9.  }
```

想一想：第 6 行的条件是否可以写成 if('a'<＝ch<＝'w')，这样能否得到正确的输出？

【练习 3.2】 输入两个数，求最大数并输出，测试数据如下。

```
输入:12  35
输出:35
```

【例 3.3】 输入 3 个数 x1、x2、x3，按从小到大的顺序输出这 3 个数。

分析：实际这就是一个排序的问题。为实现顺序输出，可以把最小数放到 x1。如果 x1＞x2，则交换两个数，同理 x1 和 x3 比较，x2 和 x3 比较。这样经过交换，3 个数按从小到大的顺序分别存储在 x1、x2、x3 中。

交换两个数一般需要一个中间变量，可用 3 条赋值语句实现：

```
temp=x1;
x1=x2;
x2=temp;
```

if 条件成立时，应该执行这 3 条语句，为此需要使用复合语句。

例 3.3 的参考程序如下。

```
1.  #include <stdio.h>
2.  int main( )
3.  {
4.      int x1,x2,x3,temp;
5.      scanf("%d%d%d",&x1,&x2,&x3);
6.      if (x1>x2)
7.        { temp=x1;  x1=x2;  x2=temp; }      //括号部分构成复合语句
8.      if (x1>x3)
9.        { temp=x1;  x1=x3;  x3=temp; }      //经过两次交换,x1 为三者中最小的
10.     if (x2>x3)
11.       { temp=x2;  x2=x3;  x3=temp; }
12.     printf("%d,%d,%d", x1,x2,x3);
13.     return 0;
14. }
```

想一想：第7、9、11行的代码如果去掉花括号“{}”，对程序结果会不会造成影响？

3.2.2 双分支结构


3-4 双分支 if

【例3.4】 编写一个密码判断程序，如果密码正确则给出正确提示，否则给出错误提示。

分析：根据密码正确与否，需要分别执行不同的语句，其执行过程见图3.2。

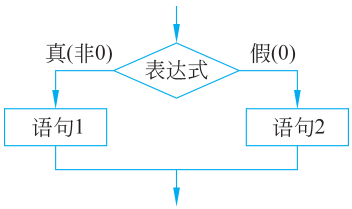

图3.2 双分支选择结构条件语句执行过程

C语言解决这类双分支选择结构的语句形式：

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

说明：if语句中的“表达式”一般为关系表达式或逻辑表达式，但不限于这两种表达式。是执行语句1，还是执行语句2，取决于“表达式”运算的结果。

如果表达式的值不为0，则执行语句1；否则，执行语句2。同样，语句1和语句2既可以是单语句，也可以是复合语句。

例3.4的参考程序如下。

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int key;
5.      printf("请输入密码");
6.      scanf("%d", &key);
7.      if(key==110)  printf("密码正确,欢迎使用!\n");
8.      else  printf("对不起,密码错误!\n");
9.      return 0;
10. }
```

【练习3.3】 一个整数，判断它的奇偶性，若是奇数则输出odd，若是偶数则输出even，测试数据如下。

测试数据1	测试数据2
输入：	输入：
23	32
输出：	输出：
odd	even

【练习 3.4】 输入 3 个整数，求最大数并输出，测试数据如下。

```
输入:12  35  24
输出:35
```

3.2.3 多分支结构


3-5 多分支结构

【问题描述 3.3】 身体质量指数(Body Mass Index，BMI)简称体指数，是目前国际相对常用的一种衡量人体胖瘦程度以及健康状况的指标，计算公式是：体指数＝体重(kg)÷身高(m)^2。

例如，若体重＝55.8kg，身高＝1.70m，则 BMI＝55.8÷(1.70)^2＝19.31。

成人 BMI 指标分为 4 种状态，对应的中国标准如图 3.4 所示。

表 3.4 成人 BMI 指标

状态	中国标准
偏瘦	BMI＜18.5
正常	18.5≤BMI＜24
偏胖	24≤BMI＜28
肥胖	BMI≥28

如果编程实现 BMI 的判断，会发现这里分了 4 种状态，一个 if…else…是不够用的，所以需要用到多分支的选择结构。

多分支结构的语句形式：

```
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;
```

多分支选择结构的功能是，按顺序求各表达式的值。如果某一表达式的值为真(非 0)，那么执行其后相应的语句，执行完后整个 if 语句结束，其余语句则不被执行；如果没有一个表达式的值为真，那么执行最后的 else 语句，执行过程如图 3.3 所示。

问题描述 3.3 的计算体指数的参考程序如下。

```
1.  #include<stdio.h>
2.  int main( )
3.  {
4.      double w, h, b;                    //w 表示体重,h 表示身高,b 表示 BMI
5.      scanf("%lf%lf", &w, &h);
6.      b=w/(h*h);
7.      printf("BMI=%.2lf\n", b);
8.      if (b<18.5)  printf("偏瘦!\n");
9.      else
10.         if (b<24)  printf("正常!\n");
11.         else
12.             if (b<28)  printf("偏胖!\n");
13.             else
14.                 printf("肥胖!\n");
15.     return 0;
16. }
```

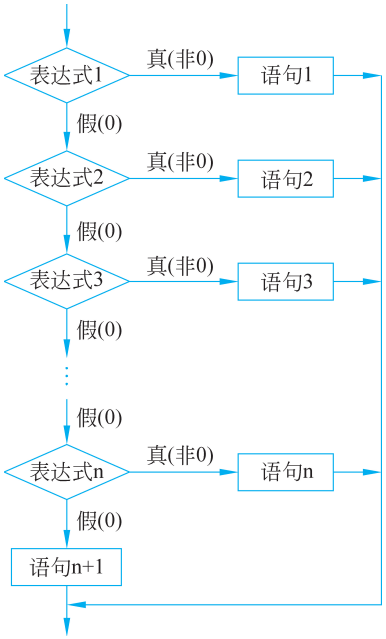

图 3.3　多分支选择结构条件语句执行过程

想一想： 第 6 行，如果去掉小括号，写成 b=w/h * h;，那么计算结果还对吗？

第 10 行，if 的条件为什么不写成 if(b>=18.5 && b<24)？

第 12 行，类似的问题，为什么不写成 if (b>=24 && b<28)？

第 13 行，为什么 else 后不写成 if (b>=28)？

再来看一个多分支结构在数学问题上的应用。

【例 3.5】 设有分段函数如下，编写程序，输入 x，计算并输出 y 值。

$$y=\begin{cases}-e^{2x+1}+3 & x<-2\\ 2x-1 & -2\leqslant x\leqslant 3\\ 3\lg(3x+5)-11 & x>3\end{cases}$$

分析： 需要对输入的 x 值进行判断，若 x < −2，则计算 $y=-e^{2x+1}+3$，程序结束；否则，若 x≤3(此处不用判断 x 是否大于或等于−2，为什么?)，则计算 y=2x−1；否则，计算 y=3lg(3x+5)−11。在这里，涉及多个条件的判断，并根据判断结果选择不同的公式计算 y 值，因此需要使用多分支结构。

例 3.5 的参考程序如下。

```
1.  #include <stdio.h>
2.  #include<math.h>                          //包含数学函数的头文件
3.  int main( )
4.  {
5.      double x, y;
6.      printf("Input x:");
7.      scanf("%lf", &x);                      //这里需用%lf,因 x 是 double 型
8.      if (x<-2)   y=-exp(2*x+1)+3;           //exp 是以 e 为底的指数函数
9.      else  if (x<=3)   y=2*x-1;
```

```
10.         else  y=3*log10(3*x+5)-11;    //log10是以10为底的对数函数,即lg
11.     printf("y=%.2lf\n", y);
12.     return 0;
13. }
```

说明：在本例中用到 exp 和 log10 两个数学函数，分别是以 e 为底的指数函数和以 10 为底的对数函数，它们都只有一个参数，计算结果均为 double 型。这两个函数是系统提供的，为了使用这些数学函数，需要包含头文件 math.h，该文件还包含了许多常用的数学函数，具体可参见附录 C.2 中的 math.h 头文件。

【例 3.6】 某公园的票价是每人 10 元，一次购票满 30 张，每张票可以少收 1 元，试编写一个自动计费程序。

分析：该问题根据购票数是否大于或等于 30 计算相应的费用。设购票数用 number 表示，金额用 money 表示。若购票数少于 30，money＝number×10；否则，money＝number×9，这可以用分支语句实现。但是，若购 29 张票，money＝number×10＝290 元，若购 30 张，money＝30×9＝270 元。显然，这时购 30 张票更省钱。因此 number＜30 时，需要考虑人数少于 30 人的情况下，number×10 是否大于 270，若 money＞270，则应按 30 张票收费，这时需要再次进行判断。为了完善功能程序功能，增强实用性，应计算出实收金额与门票的差额。

例 3.6 的参考程序如下。

```
1.  #include <stdio.h>
2.  int main( )
3.  {
4.      int  iNumber, iSum, iMoney, iBalance;
5.      printf("Input the number of entering park:");
6.      scanf("%d",&iNumber);
7.      if (iNumber>=30)  iMoney=iNumber*9;
8.      else  if (10*iNumber<270)  iMoney=10*iNumber;
9.            else  {  iMoney=270;  iNumber=30;  }
10.     printf("cost:%d yuan\n", iMoney);
11.     printf("Input Money:");
12.     scanf("%d", &iSum);
13.     iBalance=iSum-iMoney;
14.     printf("Tickets Money Cost Balance\n");
15.     printf("%d,%d,%d,%d\n", iNumber, iSum, iMoney, iBalance);
16.     return 0;
17. }
```

【例 3.7】 编程计算个人所得税。目前我国实施的个人所得税率征收规定：个税起征点为 5000 元，适用 7 级超额累进税率，具体税率如表 3.5 所示。

应纳税所得额的计算公式：

应纳税所得额＝月度收入－5000 元(起征点)－专项扣除(三险一金等)－专项附加扣除－依法确定的其他扣除

三险一金是指养老保险、医疗保险、失业保险和住房公积金。

为了编程计算简单，可以对公式进行简化：

应纳税所得额＝月度收入－5000元

个税＝应纳税所得额×税率－速算扣除数

表3.5　个人所得税税率表

应纳税额	税率/%	速算扣除数/元
① 不超过3000元的部分	3	0
② 超过3000元至12000元的部分	10	210
③ 超过12000元至25000元的部分	20	1410
④ 超过25000元至35000元的部分	25	2660
⑤ 超过35000元至55000元的部分	30	4410
⑥ 超过55000元至80000元的部分	35	7160
⑦ 超过80000元的部分	45	15160

分析：由于个人所得税采取分段税率，逐段叠加，因此可以从最高税率段开始计算，采用if的多分支结构实现。为了简化，在程序中暂时忽略三险一金，只输入工资，若工资小于或等于5000元，则不交税，可以直接结束程序；若工资大于5000元，则用工资减去5000，得到应纳税所得额，然后根据表3.5进行计算。

例3.7的参考程序如下。

```
#include <stdio.h>
int main()
{
    double fIncome, fTax;   int k;
    printf("请输入你的工资:");
    scanf("%lf",&fIncome);                    //double型数据用%lf输入
    if(fIncome<=5000)
    {  printf("免征个人所得税!\n");  return 0; } //直接结束main函数
    else  fIncome=fIncome-5000;               //工资减5000后的金额为应纳税额
    if(fIncome>80000)  fTax=fIncome*0.45-15160;
    else  if(fIncome>55000)  fTax=fIncome*0.35-7160;
          else  if(fIncome>35000)  fTax=fIncome*0.30-4410;
                else  if(fIncome>25000)  fTax=fIncome*0.25-2660;
                      else  if(fIncome>12000)  fTax=fIncome*0.20-1410;
                            else  if(fIncome>3000)  fTax=fIncome*0.10-210;
                                  else  fTax=fIncome*0.3;
    printf("你应交个人所得税为:%.2lf\n", fTax);
    return 0;
}
```

【例3.8】　在学生成绩管理中，成绩经常要在百分制与等级制之间进行转换。90分以上为A等，80～89分为B等，70～79分为C等，60～69分为D等，其余为E等。编写程序，根据输入的百分制，输出对应的等级。

分析：设用变量fscore表示成绩，而等级是依据fscore的值变化的，共有5种情况，如表3.6所示。

表 3.6　百分制成绩对应等级的判断方法

成　　绩	等　　级	判 断 方 法
fscore＞＝90	A	fscore/10＝10 或 9
80＜＝fscore＜90	B	fscore/10＝8
70＜＝fscore＜80	C	fscore/10＝7
60＜＝fscore＜70	D	fscore/10＝6
fscore＜60	E	fscore/10＝其他值

很明显这是一个多分支问题，利用多分支 if 语句完全可以解决这类问题。但是，如果 if…else…语句过多，则会令人眼花缭乱。幸运的是，C 语言提供了另一种多分支语句 switch 语句。

switch 语句的基本格式：

```
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 ]
}
```

说明：① switch 后圆括号中的表达式，只能是整型、字符型或枚举型表达式。其中枚举数据类型在以后学习。

② 当表达式的值与某个 case 后面的常量表达式的值相等时，就执行此 case 后面的语句。执行完后，流程控制转移到下一个 case(包括 default)中的语句继续执行。如果不想继续执行，就需要使用 break 语句使流程跳出 switch 结构，即终止 switch 语句的执行，最后一个分支可以不用 break 语句。这里的[ ]表示该项是可选项，可以不写。

③ 如果表达式的值与所有常量表达式都不匹配，就执行 default 后面的语句。如果没有 default 就跳出 switch，执行 switch 语句后面的语句。

④ 各个常量表达式的值必须互不相同，否则会出现矛盾。

⑤ case 后面允许有多条语句，且可以不用花括号“{ }”括起来。

例 3.8 的参考程序如下。

```
1.  #include <stdio.h>
2.  int main( )
3.  {
4.      int  iScore, temp;
5.      printf("Input score of student:");
6.      scanf("%d",&iScore);
7.      temp=iScore/10;
8.      switch(temp)
```

```
9.      {   case 10:
10.         case 9: printf("A"); break;
11.         case 8: printf("B"); break;
12.         case 7: printf("C"); break;
13.         case 6: printf("D"); break;
14.         default: printf("E");
15.     }
16.     return 0;
17. }
```

说明：本例中，temp=10 和 temp=9 都输出 A，共用输出语句 printf("A");，所以 case 10：后面不使用 break 语句，就会继续向下执行，直到遇到 break 语句终止，达到了共用 prinft("A")的目的。由于以后的语句不再共享，每次输出后，都必须使用 break 终止。否则，程序将继续执行下面的语句。

另外，default 可以放在 switch 语句中的任意位置，但要注意，如果 default 不是放在最后，则它后面也要写上 break 语句。上例中的 switch 语句也可以写成以下形式。

```
switch(temp)
{   default: printf("E"); break;        //这里需要写上 break 语句
    case 10:
    case 9: printf("A"); break;
    case 8: printf("B"); break;
    case 7: printf("C"); break;
    case 6: printf("D"); break;
}
```

3-6 if 语句的嵌套

3.2.4 if 语句的嵌套

【例 3.9】 编程实现一个猜数游戏。程序给出一个数字，如果游戏者猜对了则显示 You are right!，否则显示 You are wrong!，并提示游戏者是猜大了还是猜少了。

分析：该问题与前面几个问题的不同之处在于，如果条件不成立，先显示提示信息，然后再判断猜大了还是猜少了。也就说，第二个判断是嵌在第一个判断条件不成立语句中，这是一种 if 语句的嵌套形式。

所谓 if 语句的嵌套，是指 if 语句的 if 块或 else 块中又包含一个完整的 if 语句。

if 语句的嵌套的一般形式：

```
if (表达式 1)
    if (表达式 2) 语句 1;
    else 语句 2;
else
    if (表达式 3) 语句 3;
    else 语句 4;
```

对于嵌套结构，必须注意 else 与 if 的配对关系。**C 语言规定 else 总是与它前面最近的，而且没有与其他 else 配对的 if 进行配对**。特别是 if…else 子句数目不一样时(if 的数量只会大于或等于 else 的数量)。例如：

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

根据 C 语言规定，上面 else 应与第二个 if 配对。如果希望 else 与第一个 if 配对，可以将第二个 if 用一对花括号“{}”括起来，即写成下面的形式：

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

例 3.9 的参考程序如下。

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int iOrigNum=555, iInputNum;      //iOrigNum 表示要猜的数,初始设定为 555
5.      printf("Please input a integer number:\n ");
6.      scanf("%d",& iInputNum);          //输入游戏者猜的数
7.      if (iInputNum ==iOrigNum)         //判断是否相等
8.          printf("You are right!\n");
9.      else                              //else 表示两个数不相等,即猜错了
10.     {   printf("You are wrong!\n");
11.         if (iInputNum > iOrigNum)  printf("Your number is bigger!\n");
12.         else  printf("Your number is smaller!\n");
13.     }
14.     return 0;
15. }
```

一元二次方程求根是初中数学知识，现在可以用 C 语言编程来进行求解。

【例 3.10】 求一元二次方程 $ax^2+bx+c=0$ 的根，a、b、c 由键盘输入。

分析： 对于一元二次方程，有以下几种可能。

如果 a=0，不是二次方程，不需要求解。

如果 a!=0，是二次方程，求根又分为以下 3 种情况：

$b^2-4ac=0$，有两个相等的实根；

$b^2-4ac>0$，有两个不等的实根；

$b^2-4ac<0$，有两个共轭复数根。

该问题有多个条件，这就涉及条件语句的嵌套问题；另外，由于该问题涉及开平方运算，有可能出现无限小数，为此需要设置一个最小值，小于或等于该值就认为是 0。这里使用 C 语言提供的开平方函数 sqrt、取绝对值函数 fabs，所以在程序开始必须包含 math.h 头文件。

因为实数在内存中是不精确表示的，所以一般不用==判断一个实数是否等于 0。常用的方法是判断一个实数是否小于一个足够小的数(如 0.000001，即 1E-6)，若是，即可认定该实数等于 0。用 N-S 图描述的求解二元一次方程的算法如图 3.4 所示。

例 3.10 的参考程序如下。

```
1.  #include <stdio.h>
2.  #include <math.h>                      //包含数学函数头文件
```

```
3.  int main( )
4.  {
5.      double a, b, c, disc, x1, x2, rpart, ipart;
6.      scanf("%lf%lf%lf", &a, &b, &c);
7.      if (fabs(a)<=1E-6)                  //如果 a=0,则输出不是一元二次方程
8.          printf("It isn't a quadratic.\n");
9.      else                                //若 a 不等于 0,则计算方程的根
10.     {   disc=b*b-4*a*c;                 //计算 disc
11.         if (fabs(disc)<=1E-6)           //若 disc 等于 0,则输出两个相等的实根
12.             printf("two equal roots: \n%.2lf\n", -b/(2*a));
13.         else
14.         {   if (disc>1E-6)              //若 disc 大于 0,则输出两个不等的实根
15.             {   x1=(-b+sqrt(disc))/(2*a);
16.                 x2=(-b-sqrt(disc))/(2*a);
17.                 printf("two distinct real roots: \n%.2lf, %.2lf\n", x1, x2);
18.             }
19.             else                        //若 disc 小于 0,则输出两个不等的虚根
20.             {   rpart=-b/(2*a);
21.                 ipart=sqrt(-disc)/(2*a);
22.                 printf("two complex roots: \n%.2lf+%.2lfi, %.2lf-%.2lfi\n",
23.                     rpart, ipart, rpart, ipart);
24.             }
25.         }
26.     }
27.     return 0;
28. }
```

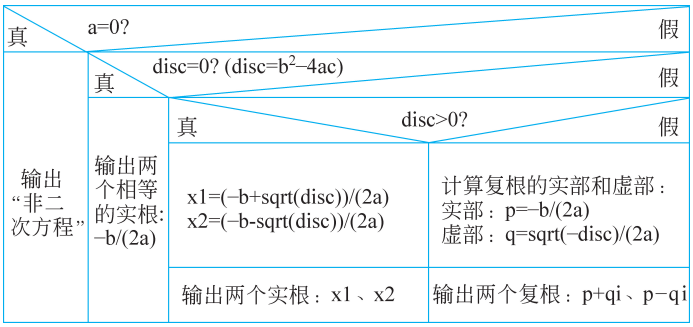

图 3.4　用 N-S 图描述的求解二元一次方程的算法

说明：在本例中，要特别注意花括号“{ }”的使用，左括号“{”一定与最近的右括号“}”结合成一对，而不能交叉。if…else…在逻辑上是一条语句。另外，格式缩进规范很容易看清楚 if 与 else 的配对关系，有利于减少出错概率。

程序运行输出：

测试数据 1	测试数据 2	测试数据 3	测试数据 4
输入：	输入：	输入：	输入：
1 2 1	1 0 -1	2 1 2	0 1 2
输出：	输出：	输出：	输出：
two equal roots: -1.00	two distinct real roots: 1.00, -1.00	two complex roots: -0.25+0.97i, -0.25-0.97i	It isn't a quadratic.

3.2.5 条件运算符

条件运算在程序设计过程中经常遇到，有些 if 语句非常简单，可直接用条件运算来实现。例如，两个数取最大数，用 if 语句可以写成如下形式。

```
if (a>b)  max=a;
else  max=b;
```


3-7 条件运算符

对于这种非常简单的 if 语句，C 语言提供了一种方便格式，即条件运算符。

条件运算符“？：”是 if 语句的缩写形式，是唯一的三目运算符。

条件表达式的一般形式：

```
表达式 1? 表达式 2:表达式 3
```

条件表达式的功能是，先计算表达式 1 的值，若为真(非 0)，则取表达式 2 的值为整个条件表达式的值；若表达式 1 的值为假(0)，则取表达式 3 的值为整个条件表达式的值。其执行过程如图 3.5 所示。

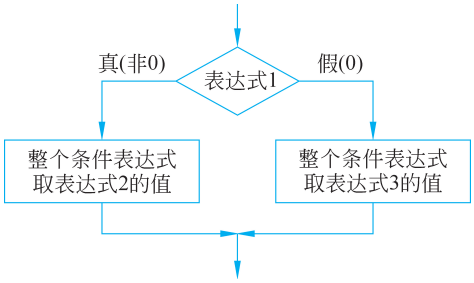

图 3.5　条件运算符执行过程

若 if 语句中，在表达式为“真”和“假”时，都只执行一个赋值语句且给同一个变量赋值时，可以使用简单的条件运算符来处理。

上面两个数取最大数的 if 语句可以用条件运算符写成：

```
max=a>b? a:b;
```

该式中，如果 a＞b 成立，则 max＝a；否则，max＝b。

说明： ① 条件运算符的优先级高于赋值运算符，低于关系运算符和算术运算符。

例如，max＝a＞b? a:b 等价于 max＝((a＞b)? a:b)。

② 条件运算符的结合性为“自右向左”。

例如，a＞b? a:c＞d? c:d 等价于(a＞b)? a:((c＞d)? c:d)。

③ 表达式 2 和表达式 3 不仅可以是数值表达式，还可以是赋值表达式或函数表达式。

例如，a＞b? (a＝100) : prinf("%d\n",b)。

④ 表达式 1、表达式 2 和表达式 3 的类型都可以不同。条件表达式值的类型是表达式 2 和表达式 3 中类型较高的类型。

例如，x＞y? 1:1.5，当 x＞y 时条件表达式的值为 double 型数据 1.0。

【例 3.11】 输入一个字符，如果是大写字母则转换为小写，否则不转换。输出最后得到

的字符。

分析：判断字符变量 ch 是否是大写字母，可以用 ch>='A'&&ch<='Z'实现，若是大写字母则需要转换，否则 ch 不变。大写字母转换为小写字母采用 ch=ch+32。

例 3.11 的参考程序如下。

```
1.  #include<stdio.h>
2.  int main( )
3.  {
4.      char ch;
5.      printf("请输入一个字符:\n");
6.      ch=getchar();
7.      ch= (ch>='A'&&ch<='Z') ?(ch+32):ch;
8.      printf("%c",ch);
9.      return 0;
10. }
```

说明：表达式 ch=（ch>='A'&&ch<='Z'）?（ch+32）：ch 中的圆括号可以不要，但有圆括号程序看上去会更加清晰。

3.3 循环结构

3.3.1 循环的引出

例 3.9 猜数游戏，只能猜一次，这样玩起来不过瘾。既然猜错时程序可以提示是猜大了还是猜小了，我们多猜几次应该就可以猜对了，如果要实现只有猜对了才结束程序，那么程序该如何编写？先看看例 3.9 的程序代码：

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int iOrigNum=555, iInputNum;      //iOrigNum 表示要猜的数,初始设定为 555
5.      printf("Please input a integer number:\n ");
6.      scanf("%d",& iInputNum);          //输入游戏者猜的数
7.      if (iInputNum ==iOrigNum)         //判断是否相等
8.          printf("You are right!\n");
9.      else                              //else 表示两个数不相等,即猜错了
10.     {   printf("You are wrong!\n");
11.         if (iInputNum > iOrigNum)  printf("Your number is bigger!\n");
12.         else  printf("Your number is smaller!\n");
13.     }
14.     return 0;
15. }
```

如果将程序重复写下去是可以的，但是我们不知道该写多少次，因为不知道多少次才会猜对。即使知道，但这种方式显然也是不可取的，太麻烦了。事实上，每次猜数的操作都是一样的，如果猜错了则回到程序的第 5 行再继续运行就可以了，如果猜对了则结束运行。

在解决实际问题的过程中，许多问题的求解都归结为重复执行的操作，例如数值计算中的方程迭代求根、非数值计算中的对象遍历。重复执行就是循环，循环是计算机特别擅长的

工作之一。循环并不是简单地重复，每次循环，操作的数据（状态、条件）都可能发生变化。循环的动作是受控制的，例如满足一定条件才继续做，一直做到某个条件满足或者做多少次才能结束。也就是说，重复工作需要进行控制——循环控制。C 语言提供了 3 种循环控制语句（不考虑 goto 和 if 构成的循环），构成了 3 种基本的循环结构。

（1）while 循环：先判断条件，再执行循环体。

（2）do-while 循环：先执行循环体，再判断条件。

（3）for 循环：先判断条件，再执行循环体。

3.3.2 while 循环

在第 2 章编程输出星号组成的图形，如一个星号矩形，那时写的程序是这样的，有没有觉得这程序实在是“低级”。

```
#include<stdio.h>
int main( )
{
    printf("******\n");
    printf("******\n");
    printf("******\n");
    printf("******\n");
    printf("******\n");
    return 0;
}
```

现在还是输出这个星号矩形，可以把程序写得“高级”一点。循环是重复做一件事，上面这个程序中，printf("******\n"); 重复了 5 次，现在这个重复的语句只需要写一次，然后控制它执行 5 次。为了控制 5 次，还需要一个计数器。计数器其实就是一个整型变量，每执行一次星号输出，这个整型变量就加 1，当计数器等于 5 时就可以结束循环。现在可以把上面的程序改写如下。

```
#include<stdio.h>
int main( )
{
    int  i ;                          //i 就是计数器
    i=0;                              //i 初值赋为 0
    while ( i<5 )
    {   printf("******\n");
        i=i+1;
    }
    return 0;
}
```

虽然还没有讲 while 循环，但是这个代码也基本能看懂。while (i<5)就是当 i<5 成立时，输出一行星号，然后计数器 i 加 1；当 i<5 不成立时，就结束循环。


3-8 while 循环

while 循环的一般形式：

```
while ( 表达式 )
    语句;
```

这里的表达式也称为“循环条件”,语句则称为“循环体”。

while 循环的执行过程:先计算 while 后面表达式的值,如果其值为“真”(非 0)则执行循环体。执行完循环体后,再次计算 while 后面表达式的值,如果值为“真”,则继续执行循环体,如果表达式值为“假”;则退出循环。while 的执行过程如图 3.6 所示。

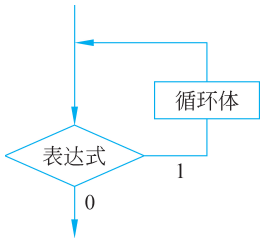

图 3.6 while 的执行过程

使用 while 语句需要注意以下几点。

① 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中的语句。因此,如果表达式的值一开始就为“假”,那么循环体一次也不执行。

② 当循环体为多个语句组成时,必须用花括号“{ }”括起来,构成复合语句。while(表达式)后面不能有分号“;”,因为分号代表一条空语句,这样写循环体就成空语句了。

③ 在循环体中使用的语句一般应保证产生满足循环终止条件的结果,以避免“无限循环”的发生。

下面是例 3.9 猜数游戏升级版的参考程序。

```
#include <stdio.h>
#include <stdlib.h>              //包含该文件才能调用 exit 函数
int main()
{
    int iOrigNum=555, iInputNum;
    while (1)                    //while 的表达式为 1,表示条件始终满足,即永真循环
    {   printf("Please input a integer number:\n ");
        scanf("%d", &iInputNum);
        if (iInputNum==iOrigNum)
        {   printf("You are right!\n");
            exit(0);             //exit 函数的作用是退出程序,返回操作系统
        }
        else
        {   printf("You are Wrong!\n");
            if (iInputNum>iOrigNum)  printf("Your number is bigger!\n");
            else  printf("Your number is smaller!\n");
        }
    }  //end while
    return 0;
}
```

说明:本程序里使用了 while(1)循环,循环条件直接用常数 1,因为 1 就表示“真”,即循环条件永远是成立的,因为这里我们不知道猜几次才能猜对,所以循环条件这里没法用具体的次数来控制。但是一个程序不能永远循环下去,猜对时应该结束程序,这里用了一个退出函数 exit(0);直接结束程序。当然也可以直接用 return 0;语句,效果是一样的。

下面来看一个经典的例题,它“红”的程度不亚于“Hello world!”程序。只要讲循环结构,第一个例题基本都会用这个小学一年级的加法题目。

【例 3.12】 编写程序计算:1+2+3+…+100。

分析:该程序是 100 个数的累加,加数每次增 1,如果不采用等差数列求和公式来计算,

可用循环结构来实现。设 sum 的初值为 0，加数 i 的初值为 1，算法描述如下。

step1：sum＝0，i＝1(循环初值)。

step2：当 i＜＝100 时，重复执行 sum＝sum＋i；i＝i＋1；(即循环体语句)；否则，结束循环。

step3：输出 sum 的值(即 1＋2＋…＋100 的结果)。

算法的 N-S 图表示如图 3.7 所示。

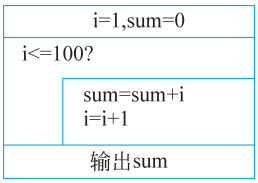

图 3.7　求和运算 N-S 图

例 3.12 的参考程序如下。

```
1.  #include <stdio.h>
2.  int main( )
3.  {
4.      int i=1, sum=0;
5.      while (i<=100)
6.      {   sum=sum+i;
7.          i++;
8.      }
9.      printf("sum=%d\n",sum);
10.     return 0;
11. }
```

编写循环程序要注意以下几个问题。

① 遇到数列求和、求积这类问题，一般可以考虑使用循环解决。

② 注意循环初值的设置，一般对于累加器常设置为 0，累乘器常设置为 1。

③ 循环体中为需要重复做的工作，同时要保证使循环逐渐趋于结束。循环的结束由 while 中的表达式(条件)控制。

从这个问题可以看出，循环给编程带来很大方便，对于例 3.4 密码检查程序，可以利用循环修改，在密码输入错误时，给用户 3 次输入机会。对于例 3.1 的闰年判断程序、例 3.6 的公园门票计算程序、例 3.8 的学生成绩分级程序，都可以利用循环实现多次运行。读者可以自行改写这些程序。

3.3.3　do-while 循环


3-9 do-while 循环

do-while 循环的一般形式：

```
do
{
    语句;
}while(表达式);
```

这里的表达式称为“循环条件”，语句称为“循环体”。

do-while 循环的执行过程：先执行 do 后面的循环体语句。然后计算 while 后面表达式的值，如果其值为“真”(非 0)，则继续执行循环体；如果表达式的值为“假”(0)，则退出循环。do-while 的执行过程如图 3.8 所示。

说明：① do-while 循环与 while 循环十分相似，它们的主要区别是：while 循环先判断循环条件再执行循环体，循环体可能一次也不执行。do-while 总是先执行一次循环体，然后

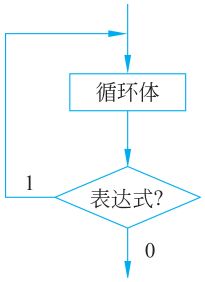

图 3.8 do-while 的执行过程

再求表达式的值,因此,无论表达式是否为“真”,循环体至少执行一次。

② 当 do-while 语句的循环体语句中只有一条语句时也可不用花括号,但是加上花括号可增加程序的可读性。

【例 3.13】 编写程序计算:1+2+3+…+n,其中 n 是由键盘输入的任意正整数。

分析:该题与例 3.12 相似,只是将 100 改成了 n,而 n 的值由键盘输入,这样具有更好的灵活性。这里对例 3.12 的代码进行一些修改,并使用 do-while 编程实现。

例 3.13 的参考程序如下。

```
#include <stdio.h>
int main( )
{
    int n, i=1, sum=0;
    printf("input n: ");
    scanf("%d", &n);
    do
    {   sum=sum+i;
        i++;
    } while (i<=n);
    printf("sum=%d\n",sum);
    return 0;
}
```

【例 3.14】 编写一个简单教学程序,训练小学生的加减法,能够随机产生两位数的加减法测试题。若结果错误则提示练习者重新计算,若结果正确则给予表扬,并询问练习者是否继续。

分析:根据问题要求,首先需要使用随机数函数,利用随机数函数,产生 100 以内的两个数;再产生 2 以内的一个随机数,当随机数为 0 时运算符为减号,随机数为 1 时运算符为加号。在屏幕上输出加法或减法的算式,学生输入计算结果,判断学生计算结果是否正确,给出相应答复。询问是否继续做题,如回答 y 或 Y 则进入下一次循环,否则退出。需要注意的是,如果减数大于被减数,需要将两个数交换。该问题至少要进行一次运算才可以退出,因此是一种直到型循环。

例 3.14 的参考程序如下。

```
#include<stdio.h>
#include<stdlib.h>              //该头文件中有 rand、srand、system 函数的函数声明
#include<time.h>                //该头文件中有 time 函数的函数声明
int main()
{
    int x, y, comp, result, temp;
    char ch, choice;
    srand(time(0));             //获取系统时间作为随机数的种子
    do
    {   system("cls");          //调用系统命令——清除屏幕
        x=rand()%100;           //rand 函数产生一个 0~32767 的随机整数
        y=rand()%100;           //通过取余运算可将产生的随机数控制在 0~99
```