第 3 章　选择结构与循环结构

C 语言程序设计是面向过程的结构化程序设计。在结构化的程序中，可以执行的流程分为 3 种：顺序结构、选择结构和循环结构。顺序结构是程序流程控制的基本结构；同时，为了实现各种功能的算法，除了按照正常的流程顺序执行外，还需要改变顺序控制流程以实现某种功能，这种情况下要用到选择语句、循环语句等控制语句，这些控制语句根据不同的情况分别执行相应的动作，或者重复执行某个动作。本章介绍在程序中怎样实现以上选择和循环控制程序结构。

在第 1 章中曾介绍过，一个函数包含声明部分和执行部分，执行部分是由语句组成的，语句的作用是向计算机系统发出操作指令，要求执行相应的操作。C 语言中的语句主要有控制语句、表达式语句、空语句、复合语句、函数调用语句等，下面分别进行简要介绍。

(1) 控制语句。用于完成一定的控制功能。C 语言有以下 9 种控制语句。

① if…else　　(条件语句)
② for　　(循环语句)
③ while　　(循环语句)
④ do…while　　(循环语句)
⑤ break　　(中断 switch 或循环语句)
⑥ continue　　(结束当前循环语句)
⑦ switch　　(多分支选择语句)
⑧ return　　(返回语句)
⑨ goto　　(转向语句)

(2) 表达式语句。表达式是通过运算符连接操作数得到的式子，包括算术运算表达式、关系运算表达式、逻辑运算表达式、赋值表达式等。一个表达式加上分号就构成表达式语句。例如：

```
x=1;
a=3*b-c/4;
++i;
c=a>b?a:b;
```

都是合法的表达式语句。

在各种表达式语句中，赋值语句是最基本的表达式语句，也是用得最多的语句。与其他高级语言相比，C 语言的赋值语句有以下不同。

① C 语言的赋值符号“=”是一个运算符，而在其他大多数语言中赋值符号不是运算符。

② 赋值表达式可以包括在其他表达式中，但赋值语句不能包含在其他语句和表达式中。例如：

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

在该语句中，while 中的条件表达式首先执行赋值运算 ch＝getchar()，然后判断 ch 是否为换行符\n。while 中的条件表达式可以包含赋值表达式 ch＝getchar()，但不能包含赋值语句“ch＝getchar();”。

任何表达式都可以加上分号成为语句，例如，“x＋y＊5－z;”是一条合法语句，但是没有将运算结果赋值给任何一变量，所以单独执行没有意义。

（3）空语句。空语句是由一个分号(;)构成的语句。

与其他语句不同的是，空语句不执行任何操作，通常与 while 和 for 语句一起使用，作为循环语句的循环体(循环体什么也不执行)，例如：

```
while((c=getchar())==' ');
```

该语句表示将跳过输入字符串中的空白字符。

（4）复合语句。用花括号将多条语句括起来得到的就是复合语句，又称分程序。复合语句通常用于选择结构、循环结构中。例如：

```
while(grade>CUTOFF){
    printf("请输入分数:");
    scanf("%d",&grade);
    total+=grade;
}
```

3 条语句用花括号括起来形成一条复合语句，注意最后一条语句的分号不能省略。

对于简单的语句块，有时用逗号表达式替代复合语句。例如，

```
if(a<b)
    temp=a,a=b,b=temp;
```

（5）函数调用语句。在函数调用之后加上分号形成的命令语句。例如，调用输出函数：

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

3.1 选择结构

在许多算法实现中要使用选择结构，例如，只有在除数不为 0 的情况下才能执行除法。选择结构用于根据条件成立与否选择下一步要执行的动作，可以从两种或多种候选动作中选择一种执行。C 语言中的选择语句可以根据表达式的值选择要执行的分支语句，从而实现流程控制。

在 C 语言中，提供两种选择语句实现流程控制。

（1）if 语句：单分支选择，也可以通过一系列嵌套的 if…else 语句实现多分支选择。

（2）switch 语句：多分支选择，其中表达式的值与一组常量比较。

C 语言中的 if 语句用来判断给定条件是否满足，根据判定结果决定执行相应的操作。

C语言提供了3种形式的if语句，分别是简单if语句、if…else语句及嵌套的if…else语句。

3.1.1 简单if语句

简单if语句的一般形式如下：

```
if(表达式)
    语句段
```

例如：

```
if(x>y)
    printf("x>y");
```

if语句圆括号中的表达式一般是关系表达式或逻辑表达式，但也可以是其他表达式。if语句的流程如图3-1所示。

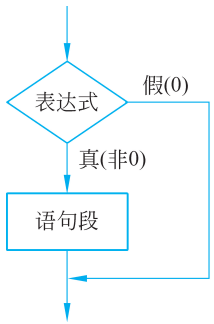

图3-1 if语句的流程图

首先对圆括号中的表达式求值，当表达式的值为真(非0)时，执行圆括号后的语句段；否则跳过该语句段，继续执行后面的语句。其中，语句段可以是一条语句，也可以是包含在花括号之间的复合语句。

例3-1 求绝对值。

【问题描述】

输入任意一个整数，然后输出它的绝对值。

【输入格式】

输入任意一个整数。

【输出格式】

输出它的绝对值。

【样例输入】

```
-4
```

【样例输出】

```
4
```

【问题分析】

正数和零的绝对值是它自身，负数的绝对值是它的相反数，因此只需要对负数进行处理即可。

【参考代码】

```
1.  #include<bits/stdc++.h>
2.  using namespace std;
3.  int main()
```

```
{
    int n;
    cin>>n;
    if(n<0)
        n=-n;
    cout<<n;
    return 0;
}
```

例 3-2 3 个数字排序。

【问题描述】

输入 3 个整数 a,b,c,要求将输出的数据按从大到小排序后输出。

【输入格式】

输入 3 个整数 a,b,c,每个整数之间用空格隔开。

【输出格式】

从大到小顺序输出 a,b,c 的值。

【样例输入】

```
4 3 5
```

【样例输出】

```
5 4 3
```

【问题分析】

对 3 个数进行排序,可以将最大的放在 a 中,最小的放在 c 中,按照 a,b,c 的顺序输出即可。将最大的放在 a 中,需要将 a 分别与 b 和 c 进行两次比较,如果 a 小,进行相应的交换即可;将最大的数换在 a 中后,再比较一下 b 和 c,将较小的放在 c 中即可。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a,b,c,t;
    cin>>a>>b>>c;
    if(a<b){
        t=a; a=b; b=t;
    }
    if(a<c){
        t=a; a=c; c=t;
    }
    if(b<c){
        t=b; b=c; c=t;
    }
```

```
15.     cout<<a<<" "<<b<<" "<<c;
16.     return 0;
17. }
```

【代码分析】

使用 if 语句时，如果表达式的值为非 0，其后只能执行一条语句。为了完成交换，有两种方式：一种是参考代码中采取复合语句，另一种是采取逗号表达式：

```
if(a<b)
    t=a, a=b, b=t;
```

【拓展思考】

如何对 4 个数进行排序？5 个数呢？

3.1.2 if…else 语句

if…else 语句可以根据表达式结果选取两路指令序列中的一路进行操作。if…else 语句的一般形式如下：

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

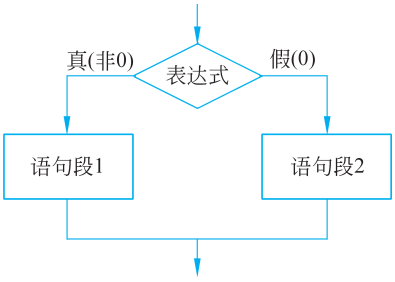

图 3-2 if…else 语句的流程图

首先对表达式求值，如果表达式的值为真（非 0），执行语句段 1；如果表达式的值为假（0），则执行语句段 2。if…else 语句的流程如图 3-2 所示。

例 3-3 求两个整数中的较大者。

【问题描述】

输入两个整数，求其中的较大者。

【输入格式】

在一行中输入用空格隔开的两个整数，如 5 9。

【输出格式】

输出两个整数中的较大者，输出形式举例：max=9。

【样例输入】

```
5 9
```

【样例输出】

```
max=9
```

【问题分析】

本题旨在考查 if…else 结构，对给定的两个数进行判断，输出较大的数。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a,b,c;
    cin>>a>>b;
    if(a>b)
        c=a;
    else
        c=b;
    cout<<"max="<<c;
    return 0;
}
```

3.1.3 嵌套的 if…else 语句

在 if…else 语句中可以执行一条语句，也可以包含任何有效的 C 语言语句块。当该语句块又是一个 if…else 语句时，则形成了 if…else 语句的嵌套。使用嵌套的 if…else 语句可以实现按照不同条件选择两个以上的分支流程。C 语言中的 if 语句嵌套形式很灵活，可以在 if 语句中包含 if…else 语句结构，也可以在 else 语句中包含 if…else 结构，或者在 if 及 else 中都包含 if…else 结构。图 3-3 给出了嵌套的 if…else 语句一般形式。

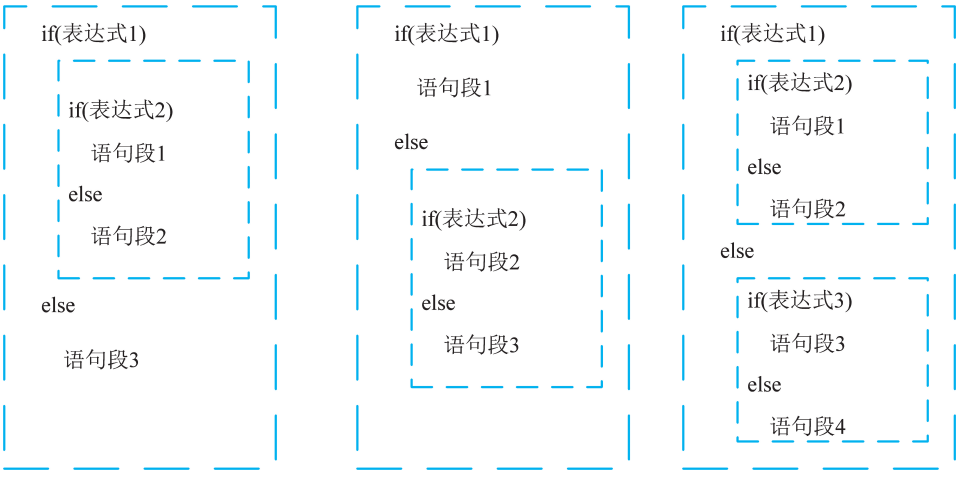

图 3-3 嵌套的 if…else 语句一般形式

例 3-4 分段函数。

【问题描述】

定义如下的分段函数，编程实现。

例 3-4

$$y=\begin{cases}-x^2, & x<-1\\ \sqrt{x}+2x-1, & x>5\\ x^2+3x+1, & \text{其他}\end{cases}$$

【输入格式】

一个实数 x。

【输出格式】

实数 y，保留到小数点后两位。

【样例输入】

```
-3
```

【样例输出】

```
-9.00
```

【问题分析】

本题旨在考查 if…else 的嵌套结构，需要注意的是，构造的条件需要涵盖分段函数的定义域。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    double x,y;
    cin>>x;
    if(x<-1)
        y=-x*x;
    else{
        if(x>5)
            y=sqrt(x)+2*x-1;
        else
            y=x*x+3*x+1;
    }
    printf("%.2lf",y);
    return 0;
}
```

对于 if…else 语句的嵌套使用，进行以下 3 点说明。

(1) else 总是与 if 成对出现。可以单独使用 if 语句，但是不能单独使用 else 语句。

(2) else 总是与它上面最近的未配对的 if 匹配。

例如，如果将上例程序中 if…else 语句的嵌套关系改为如下程序，思考修改后的程序是否正确。

```
y=x*x+3*x+1;
if(x>=-1)
    if(x>5)
        y=sqrt(x)+2*x-1;
else y=-x*x;
```

编程者把 else 写在与第一个 if 同一列上，意图是使 else 与第一个 if 对应，但实际上 else 是与第二个 if 配对的，因为它们相距最近。应该计算的是 x＜－1 的情况，但此时表达式 y＝－x*x 是 x＜＝5 的情况，所以结果是错误的。

如果 if 与 else 的数目不一样，为避免错误，习惯上将嵌套中的 if 语句用花括号括起来。例如，上述错误的程序可以做如下修改：

```
y=x*x+3*x+1;
if(x>=-1){
    if(x>5)
        y=sqrt(x)+2*x-1;
}
else
    y=-x*x;
```

对于初学者来说，为了避免使用上的错误，最好将嵌套的 if 语句用花括号括起来，不管该语句是不是复合语句，都写成如下形式：

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

这样，花括号限定了嵌套 if 语句的范围，因此 else 与第一个 if 配对。

（3）在嵌套的 if…else 语句的使用中，将第二个 if…else 语句嵌套在其上层 if…else 语句的 else 部分内时，会产生最常用的嵌套的 if…else 语句，其嵌套的一般形式如下：

```
if(表达式 1)
    语句段 1
else if(表达式 2)
    语句段 2
else if(表达式 3)
    语句段 3
…
else if(表达式 n)
    语句段 n
else
    语句段 n+1
```

这种嵌套形式又称 else…if 结构。else…if 结构依次计算表达式 i(i=1,2,…,n)的值,当表达式 i 为真时,执行与之相关的语句段 i,并以此结束整个嵌套链。else…if 结构的流程图如图 3-4 所示。

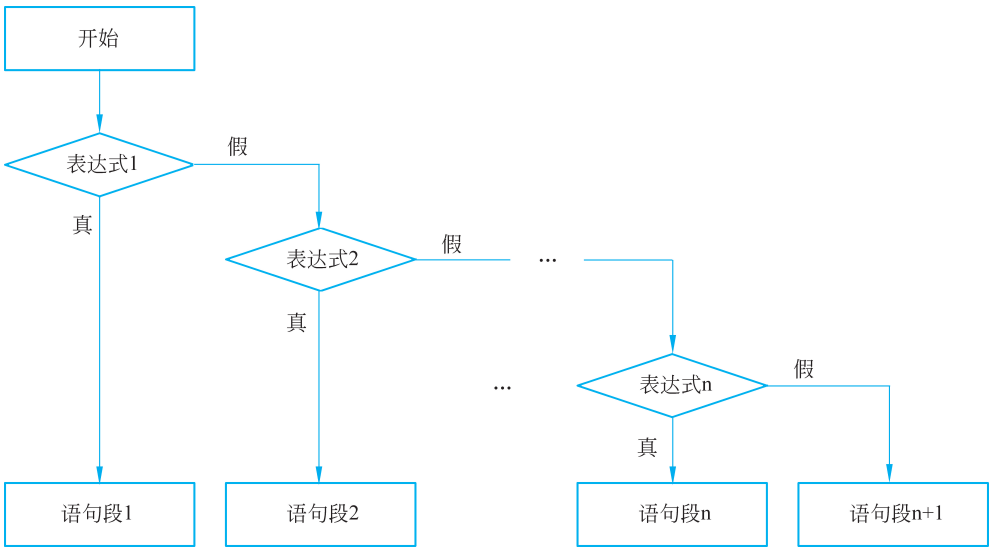

图 3-4 else…if 结构的流程图

例 3-5 等级成绩。


例 3-5

【问题描述】

学院的考试采用等级制,即将百分制转换为 A、B、C、D、E 5 个等级,设成绩为 X,则 90≤X≤100 为 A,80≤X<90 为 B,70≤X<80 为 C,60≤X<70 为 D,否则为 E。编写一个程序,将输入的分数转换成 A、B、C、D 和 E 5 个等级。

【输入格式】

输入一个单精度数。

【输出格式】

输出相应等级。

【样例输入】

```
100
```

【样例输出】

```
A
```

【问题分析】

本题考查 if…else 语句的嵌套结构。对于这种多条件的问题,建议从一个方向进行判断,可以从高到低,也可以从低到高。

【参考代码】

```
1.  #include<bits/stdc++.h>
```

```
2.  using namespace std;
3.  int main()
4.  {
5.      double x;
6.      cin>>x;
7.      if(x>=90)
8.          cout<<"A"<<endl;
9.      else if (x>=80)
10.         cout <<"B"<<endl;
11.     else if (x>=70)
12.         cout<<"C"<<endl;
13.     else if (x>=60)
14.         cout<<"D"<<endl;
15.     else
16.         cout<<"E"<<endl;
17.     return 0;
18. }
```


例 3-6

例 3-6 求一元二次方程的根。

【问题描述】

利用公式 $x=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$ 求一元二次方程 $ax^2+bx+c=0$ 的根，其中 a 不等于 0。

【输入格式】

输入一行，包含 3 个浮点数 a，b，c（它们之间以一个空格分开），分别表示方程 $ax^2+bx+c=0$ 的系数。

【输出格式】

输出一行，表示方程的解。

若 $b^2=4ac$，则两个实根相等，输出为 x1=x2=…。

若 $b^2>4ac$，则两个实根不等，输出为 x1=…;x2=…，其中 x1>x2。

若 $b^2<4ac$，则有两个虚根，输出为“x1=实部+虚部 i；x2=实部−虚部 i”，即 x1 的虚部系数大于或等于 x2 的虚部系数，实部为 0 时不可省略。实部=−b/(2 * a)，虚部=sqrt(4 * a * c−b * b) / (2 * a)。

所有实部要求精确到小数点后 5 位，数字、符号之间没有空格。

【样例输入 1】

```
1.0 2.0 8.0
```

【样例输出 1】

```
x1=-1.00000+2.64575i;x2=-1.00000-2.64575i
```

【样例输入 2】

```
1 0 1
```

【样例输出 2】

```
x1=-0.00000+1.00000i;x2=-0.00000-1.00000i
```

【问题分析】

本题旨在考查 if…else 结构和输出的格式控制，需要根据不同的情况进行处理。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    double a,b,c,delta;
    double x1,y1,x2,y2,t;
    cin>>a>>b>>c;
    delta=b*b-4*a*c;
    if(delta==0)
        printf("x1=x2=%.5lf",-b/(2*a));
    else{
        if(delta>0){
            x1=(-b+sqrt(delta))/(2*a); x2=(-b-sqrt(delta))/(2*a);
            if(x1<x2)                  //交换两个实根
                t=x1,x1=x2,x2=t;
            printf("x1=%.5lf;x2=%.5lf",x1,x2);
        }
        else{
            x1=(-b)/(2*a); y1=sqrt(-delta)/(2*a); y2=-sqrt(-delta)/(2*a);
            if(y1<y2)                  //交换两个虚部
                t=y1,y1=y2,y2=t;
            printf("x1=%.5lf",x1);
            if(y1>0)                   //正的虚根用+显示
                printf("+%.5lfi;x2=%.5lf",y1,x1);
            else                         //负的虚根用-显示,若数是负数,直接输出
                printf("%.5lfi;x2=%.5lf",y1,x1);
            if(y2>0)
                printf("+%.5lfi",y2);
            else
                printf("%.5lfi",y2);
        }
    }
    return 0;
}
```

3.1.4 多分支选择结构——switch 语句

if 语句可以处理二分支问题，采用嵌套的 if…else 语句还可以处理多分支问题。但如果分支太多，嵌套的层数就会很深，if 与 else 的配对就容易出现问题，还会影响程序的可读性。为此，C 语言提供了专门用于处理多分支选择结构的条件语句——switch 语句，又称开关语句。在某些情况下使用 switch 语句，程序结构清晰，使得代码具有更好的可读性。switch 语句的一般形式如下：

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

switch 语句的流程：首先计算 switch 表达式的值，如果该表达式的值等于某个 case 的常量表达式的值，则程序的控制转向该 case 后面的语句，即执行该 case 后的语句段。如果 case 语句段后有 break 语句，执行完语句段后则控制跳出 switch 语句，执行 switch 之后的程序；如果 case 语句段后没有 break 语句，则执行完该语句段后，将继续往下执行，直到遇到 break 语句，如果没有 break 语句，则将执行到 switch 语句的最后。如果 switch 表达式的值不等于任何一个 case 的常量表达式的值，但 switch 语句有 default 语句，则执行 default 后面的语句段，执行后，退出 switch 语句，执行 switch 之后的语句。switch 语句的流程如图 3-5 所示。

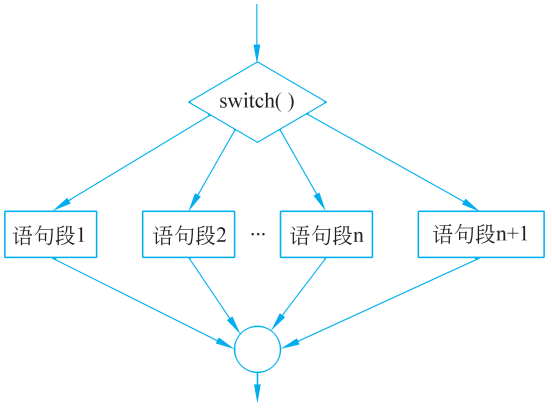

图 3-5 switch 语句的流程图

使用 switch 语句时，有 8 点需要注意。

(1) switch 关键字后面的表达式只能是整型、字符型或枚举型表达式。

（2）case 关键字后面的常量表达式必须是与表达式相对应的整型、字符型或枚举型常量，不能是变量和表达式，并且 case 和常量表达式之间要有空格。

（3）每个 case 关键字后面的常量表达式的值必须互不相同，即同一个常量在 switch 语句中只能对应一种处理方案，否则会出现互相矛盾的现象。

（4）语句段可以是简单语句也可以是复合语句。

（5）如果表达式的值与所有常量表达式的值都不匹配，就执行 default 后面的语句。如果没有 default 语句，则不执行任何语句，流程转到 switch 语句的下一条语句。

（6）case 和 default 可以出现在任何位置，其前后次序不影响执行结果，但习惯上将 default 放在 switch 语句的底部。

（7）通常在 case 后面的语句中包含 break 语句，当程序执行与表达式匹配的 case 语句段时，遇到 break 语句后跳出整个 switch 语句。如果不存在 break 语句，则执行完后，流程控制转到下一个 case（包括 default）中的语句继续执行。

（8）case 提供了执行某一语句段的入口，起着标号的作用；多个 case 可以执行同一语句段。

用 switch 语句可以代替嵌套的 if…else 语句实现多分支选择结构，下面用 switch 语句编写前面用嵌套的 if…else 语句实现的对成绩进行分类的程序。

例 3-7 等级成绩。


例 3-7

【问题描述】

同例 3-5。

【输入格式】

输入一个单精度数。

【输出格式】

输出相应等级。

【样例输入】

```
100
```

【样例输出】

```
A
```

【问题分析】

本题旨在考查 switch 语句的应用。对问题进行分析，将成绩分为 5 个区间：[0,60)，[60,70)，[70,80)，[80,90)，[90,100]。考虑到 switch 后面的表达式只能是整型，因此需要将输入的浮点数转换成整型。因此，可以有效利用整数的整除，将输入的成绩转换为 0～10 的整数。

【参考代码】

```
1.  #include<bits/stdc++.h>
2.  using namespace std;
3.  int main()
4.  {
```

```
5.      float score;
6.      cin>>score;
7.      switch((int)score/10){
8.          case 10:
9.          case 9: cout<<"A";break;
10.         case 8: cout<<"B";break;
11.         case 7: cout<<"C";break;
12.         case 6: cout<<"D";break;
13.         default: cout<<"E";
14.     }
15.     return 0;
16. }
```

【代码分析】

(int)score 是将输入的浮点数转换为整数，整除 10 会产生 0～10 的相关整数。考虑到 0～5 这 6 种情况对应等级 E，因此将其放在 default 中。当结果是 10 和 9 时，均为等级 A，因此，case 10 后面不增加任何代码，只是作为一个标号存在。输出相应的等级后，增加 break 语句，结束相应的判断。

例 3-8 写出下列程序的运行结果。

```
1.  #include<stdio.h>
2.  int main()
3.  {
4.      int i;
5.      for(i=1;i<=5;i++){
6.          switch(i%2){
7.              case 0: i++;   printf("#\n");
8.              case 1: i+=2;  printf("*");
9.              default: printf(" ");
10.         }
11.     }
12.     return 0;
13. }
```

【代码分析】

本题对 i=1～5 的 5 种情况进行判断，进行以下分析。

当 i=1 时，i%2=1，因此，执行 case 1 后的内容，i=3，并在屏幕输出 *，由于后面没有 break 语句，因此会继续执行 default 标号的内容，输出空格。

此时 i=3，执行 for 循环中的 i++，i=4，i%2=0，执行 case 0 后面的内容，输出 # 后换行；由于后面没有 break 语句，继续执行 case 1 的内容，i=6，输出 *；没有 break 语句，继续执行 default 标号后的内容，输出空格。

因此上述程序的运行结果如下（其中□表示空格）：

*□#
*□

3.1.5 选择结构程序举例

例 3-9 闰年。

【问题描述】

任意给出的一个年份，判断是不是闰年。

【输入格式】

只有一个整数 year，代表年份。

【输出格式】

如果是闰年输出 Yes，否则输出 No。

【样例输入】

```
2000
```

【样例输出】

```
Yes
```

【问题分析】

闰年的基本规则是“四年一闰，百年不闰，四百年再闰”。因此可以分为两种情况：①年份是 4 的倍数，但不是 100 的倍数；②年份是 400 的倍数。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    int year;
    cin>>year;
    if((year%4==0 && year%100!=0) || (year%400==0))
        cout<<"Yes";
    else
        cout<<"No";
    return 0;
}
```

例 3-10 3 个数排序。

【问题描述】

输入 3 个数 a，b，c，按由小到大的顺序输出。

【输入格式】

标准输入，3 个无序的单精度数，以空格隔开。

【输出格式】

标准输出，输出 3 个由小到大排好序的浮点数，以空格隔开。

【样例输入】

```
3 2 1
```

【样例输出】

```
1 2 3
```

【问题分析】

虽然可以通过简单的数据交换完成，本题旨在考查 if…else 的嵌套结构，可以不改变 3 个数的数值完成数据从小到大的输出。对于 3 个数，共有 3!＝6 种情况，如图 3-6 所示。

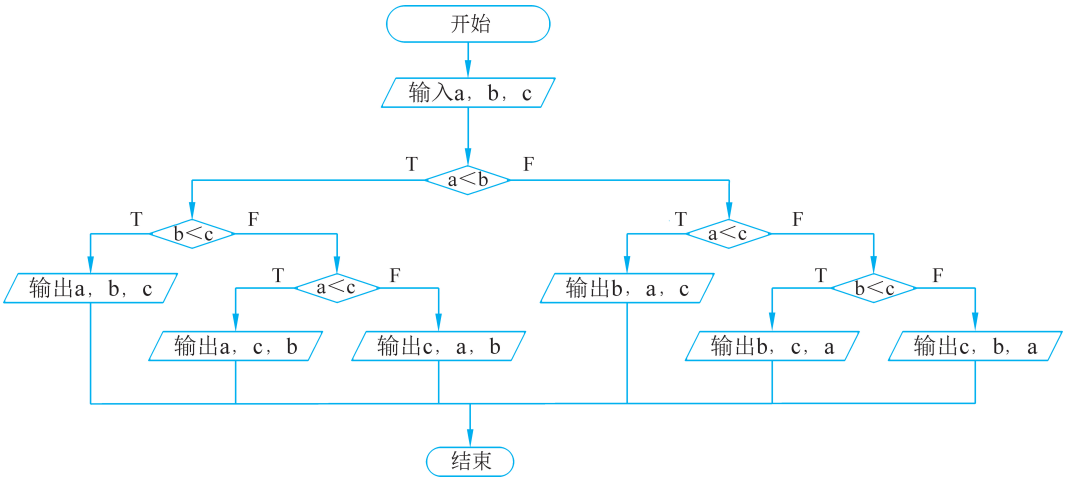

图 3-6　输入 3 个数，按从小到大的顺序排序

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    float a,b,c;
    cin>>a>>b>>c;
    if(a<b){
        if(b<c)
            cout<<a<<" "<<b<<" "<<c;
        else{
            if(a<c)
                cout<<a<<" "<<c<<" "<<b;
            else
                cout<<c<<" "<<a<<" "<<b;
        }
```

```
    }else{
        if(b>c)
            cout<<c<<" "<<b<<" "<<a;
        else{
            if(a<c)
                cout<<b<<" "<<a<<" "<<c;
            else
                cout<<b<<" "<<c<<" "<<a;
        }
    }
    return 0;
}
```

【代码分析】

这是一个典型的具有三重对称嵌套的程序，该程序保持了3个变量在比较过程中其值不发生变化。但该程序对于多个变量的比较不适用，容易造成判断混乱。

例 3-11 模拟计算器。

【问题描述】

简单计算器模拟：输入两个整数和一个运算符，输出运算结果。

【输入格式】

第一行输入两个整数，用空格隔开。

第二行输入一个运算符(+、-、*、/)。

所有运算均为整数运算，保证除数不为0。

【输出格式】

输出对两个数运算后的结果。

【样例输入】

```
30 50
*
```

【样例输出】

```
1500
```

【问题分析】

本题旨在考查switch多分支选择结构，需要根据输入的运算符进行相应的操作。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    int a,b,ans;
    char ch;
```

```
    cin>>a>>b;
    cin>>ch;
    switch(ch){
        case '+': ans=a+b; break;
        case '-': ans=a-b; break;
        case '*': ans=a*b; break;
        case '/': ans=a/b; break;
    }
    cout<<ans;
    return 0;
}
```


例 3-12

例 3-12 判断某日是一年的第几天。

【问题描述】

按照年、月、日的格式输入年份、月份和日期，运行程序以后，判断这一天是这一年的第几天。

【输入格式】

标准输入，输入一个 8 位整数，表示年、月和日，其中年为四位数，月和日都为两位数，中间无空格。

【输出格式】

标准输出，输出这一天是这一年的第几天，数据保证合法。

【样例输入】

```
20160101
```

【样例输出】

```
1
```

【问题分析】

本题旨在考查 switch 的使用技巧。如果标号后没有 break 语句，将在执行完相应的语句后继续执行下一标号的相关语句，直至遇到 break 语句或 switch 结构结束为止，可以基于这一点，对参考代码 1 进行改进。

【参考代码 1】

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    int year,month,day,flag,ans=0;
    scanf("%4d%2d%2d",&year,&month,&day);
    if((year%4==0 && year%100!=0) || (year%400==0))
        flag=1;
    else
```

```
        flag=0;
    switch(month){
        case 1: break;
        case 2: ans+=31; break;
        case 3: ans+=31+28; break;
        case 4: ans+=31+28+31; break;
        case 5: ans+=31+28+31+30; break;
        case 6: ans+=31+28+31+30+31; break;
        case 7: ans+=31+28+31+30+31+30; break;
        case 8: ans+=31+28+31+30+31+30+31; break;
        case 9: ans+=31+28+31+30+31+30+31+31; break;
        case 10: ans+=31+28+31+30+31+30+31+31+30; break;
        case 11: ans+=31+28+31+30+31+30+31+31+30+31; break;
        case 12: ans+=31+28+31+30+31+30+31+31+30+31+30; break;
    }
    ans+=day;
    if(flag==1 && month>2)              //对闰年的情况进行处理
        ans++;
    cout<<ans;
    return 0;
}
```

【参考代码 2】

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    int year,month,day,flag,ans=0;
    scanf("%4d%2d%2d",&year,&month,&day);
    if((year%4==0 && year%100!=0) || (year%400==0))
        flag=1;
    else
        flag=0;
    switch(month){
        case 12: ans+=30;               //11 月的天数
        case 11: ans+=31;               //10 月的天数
        case 10: ans+=30;               //9 月的天数
        case 9: ans+=31;                //8 月的天数
        case 8: ans+=31;                //7 月的天数
        case 7: ans+=30;                //6 月的天数
        case 6: ans+=31;                //5 月的天数
        case 5: ans+=30;                //4 月的天数
        case 4: ans+=31;                //3 月的天数
```

```
20.         case 3: ans+=28;                //2 月的天数
21.         case 2: ans+=31;                //1 月的天数
22.         case 1: ans+=day;
23.     }
24.     if(flag==1 && month>2)
25.         ans++;
26.     cout<<ans;
27.     return 0;
28. }
```

【扩展思考】

如果数据不一定合法，该如何处理？

3.2 循环结构

在解决实际问题时，程序中除了使用前面介绍的输入输出和分支选择语句外，还需要重复执行相同的某些操作，如持续不断地接收输入数据、数值累加、重复验证密码的有效性等。在这些情况下使用能重复执行特定语句的循环结构，可以使问题变得简单。C 语言提供了 3 种循环语句：while 语句、do…while 语句和 for 语句。

在 C 语言中无论使用哪种循环语句，要构成循环控制结构，必须具有如下 4 个基本要素。

（1）循环控制表达式及循环控制变量。循环控制表达式决定是否进入循环，如果循环控制表达式为真，就执行循环体中的语句，否则不执行。循环控制表达式中若存在一个随循环次数变化的变量，该变量称为循环控制变量。该变量的值在循环中被改变，直到使得循环控制表达式为假，结束循环。

（2）循环体语句。循环体是循环重复执行的操作，可以是一条语句，也可以是由多条语句构成的语句段。如果为多条语句，需要使用花括号括起来，作为循环控制的循环体。

（3）初始条件设置语句。初始条件设置语句是一条或多条初始设置循环控制表达式的语句。该语句必须放置在循环控制表达式被计算之前，以确保正确地执行循环。

（4）循环变量修改语句。为了避免死循环，必须有控制循环结束的终止条件，即在重复的循环体语句中必须有一个改变循环控制表达式使其变为假的语句；否则如果循环控制表达式不能变为假，则会进行无限循环，也称死循环。一旦执行了无限循环，可以使用 Ctrl＋Break 键[①]终止执行中的程序。

while 语句在每次执行循环体之前测试循环条件。do…while 语句在每次执行循环体之后测试循环条件。for 语句提供特殊语法，可以初始化或更新一个或几个控制变量，同时测试循环条件。

① 现在有些笔记本计算机的键盘没有 Break 键，可以用 Ctrl＋C 键。