第3章

选择结构程序设计

前两章编写的程序都是按语句书写的次序从上到下依次执行，每条语句都被执行到了，这样的程序结构称为顺序结构。但是在实际编写程序时，有些语句经常需要在符合某些条件时才能执行，例如，输入3条边计算三角形的面积，首先应判断输入的3条边是否符合构成三角形的条件（任意两条边之和大于第三条边），符合此条件计算三角形面积，否则提示无法构成三角形，这种类型的程序结构被称为选择结构，也称分支结构。

3.1 双分支结构

上面提到的判断能否构成三角形的程序结构就是典型的双分支结构。图3.1是双分支结构的基本形式。

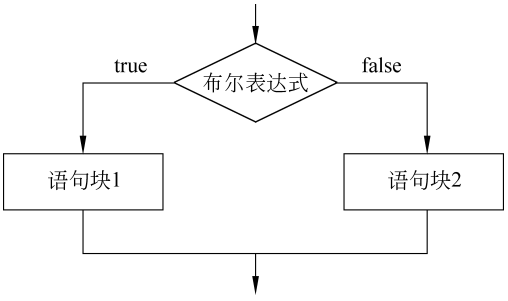

图3.1 双分支结构的基本形式

实现选择结构的最常用的语句是if语句。使用if语句的基本形式可构造双分支结构，双分支结构的if语句的基本形式如下：

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

语句的布尔表达式是最终运算结果为布尔值的表达式，通常是关系表达式

或逻辑表达式。当布尔表达式的值为真(true)时执行语句块 1 中的语句,当布尔表达式的值为假(false)时执行语句块 2 中的语句。

例 3.1　输入两个整数,输出其中较大的数。

```
import java.util.*;
public class Compare {
    public static void main(String[] args) {
        int a,b;
        Scanner reader=new Scanner(System.in);
        //输入两个整数
        a=reader.nextInt();
        b=reader.nextInt();
        //利用 if 语句输出较大的数
        if(a>b){
            System.out.println(a);
        }else{
            System.out.println(b);
        }
    }
}
```

运行程序,如果输入的是 5 3,因为布尔表达式 a>b 的值为 true,因此执行 if 后花括号中的语句,输出变量 a 的值 5;如果输入的是 3 5,则布尔表达式 a>b 的值为 false,因此执行 else 后花括号中的语句,输出变量 b 的值 5。

例 3.2　输入 3 条边的长度值,如果这 3 条边能构成三角形,则输出三角形的面积,否则输出不能构成三角形的提示信息。

```
import java.util.*;
public class Triangle {
    public static void main(String[] args) {
        double a,b,c;
        Scanner reader=new Scanner(System.in);
        //输入 3 条边的长度值,double 类型
        a=reader.nextDouble();
        b=reader.nextDouble();
        c=reader.nextDouble();
        //判断能否构成三角形:任意两条边之和大于第三条边
        if(a+b> c&&b+c> a&&c+a> b){
            //计算三角形面积并输出
            double m,area;
            m=(a+b+c)/2;
            area=Math.sqrt(m*(m-a)*(m-b)*(m-c));
            //在输出时,可利用"+"将多个数据连接起来,其中有一个数据通常是字符串
            System.out.println("三角形的面积为"+area);
```

```
        }else{
            System.out.println("不能构成三角形");
        }
    }
}
```

运行程序,输入 3 条边的值后,如果符合三角形的判定条件(两条边之和大于第三条边)则使用海伦公式计算三角形的面积并输出,否则输出不能构成三角形的提示信息。

在 if 语句中,用花括号括起来的语句块 1 和语句块 2 都是复合语句,可以包含一条语句或多条语句。当语句块中只有一条语句时,可以省略语句外面的这对花括号;而当语句块中有多条语句时,则不能省略花括号。例如,例 3.2 中,if 后的语句块中有 4 条语句,不能省略语句外面的这对花括号,否则编译时会提示没有 if 与 else 匹配的错误;else 后的语句块中只有一条语句,因此可以省略该条语句外面的这对花括号。

例 3.3 划船问题:一个教师带着 x 个学生去划船,每条船最多可装 4 人,问最少需要多少条船?

```
import java.util.*;
public class Boating {
    public static void main(String[] args) {
        int x,n;
        Scanner reader=new Scanner(System.in);
        //输入学生人数
        x=reader.nextInt();
        //计算总人数
        x=x+1;
        //判断人数是否恰好是 4 的倍数:n=x/4
        if(x%4==0)
            n=x/4;
        else
            n=x/4+1;
        System.out.println("最少需要"+n+"条船。");
    }
}
```

在这个例子中,if 和 else 后都只有一条语句,因此都省略了这些语句外面的花括号。没有花括号的约束,else 只能控制其后的一条语句,最后一行的输出语句不受其控制,属于 if 语句之后的句子,我们即便将其缩进到与其上的语句“n=x/4+1;”一样,仍不能改变它不属于 if 语句的事实。

注意:虽然只有一条语句时可省略其外面的一对花括号,但对初学者来说,在熟练掌握 if 语句之前尽量不要省略,以免在应该使用复合语句的情况下忘记加上这对花括号,造成程序流程控制上的错误。

对于双分支结构,如果 else 后的语句块 2 中没有要执行的语句时,可以省略整个 else 部分。例如,也可以使用下面的代码完成上例的划船问题。

```
import java.util.*;
public class Boating2 {
    public static void main(String[] args) {
        int x,n;
        Scanner reader=new Scanner(System.in);
        x=reader.nextInt();
        x=x+1;
        n=x/4;
        if(x%4!=0)
            n=n+1;
        System.out.println("最少需要"+n+"条船。");
    }
}
```

程序的逻辑是至少需要 x/4 条船(人数是 4 的倍数),如果人数不是 4 的倍数,则还要再加上一条船。

3.2　多分支结构

在某些问题中,可能根据条件的不同有两种以上的不同选择,这就是多分支结构,如图 3.2 所示。例如,比较两个数字的大小,实际上有 3 种情况:第一个数大于第二个数,第一个数小于第二个数,两个数相等。Java 语言对于处理这类问题通常使用多分支的 if 语句。

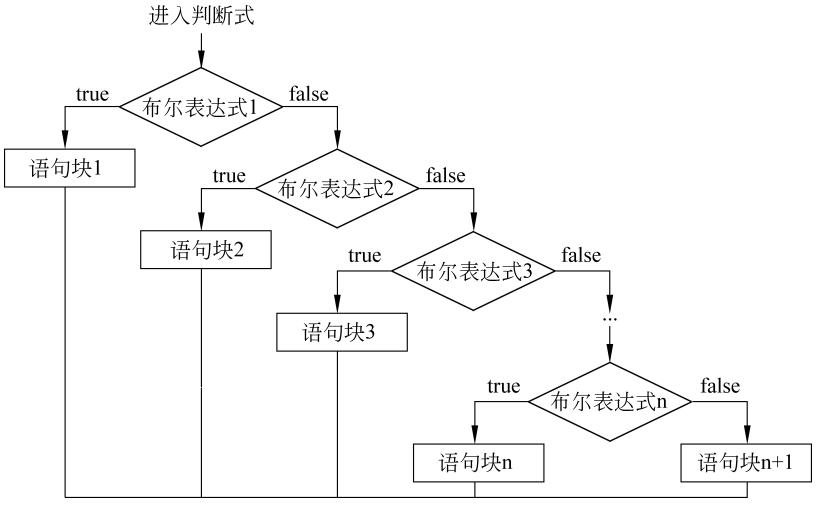

图 3.2　多分支结构的一种形式

多分支 if 语句的基本结构如下:

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

```
    语句块 2;
}else if(布尔表达式 3){
    语句块 3;
}else{
    语句块 4;
}
```

可以根据分支的多少增减 else if 语句的个数,当某个语句块中只有一条语句时可以省略包含语句块的这对花括号。

例 3.4 输入两个数及运算符,根据运算符输出这两个数的四则运算结果。

```
import java.util.*;
public class Arithmetic {
    public static void main(String[] args) {
        double x,y,result;
        char symbol;
        Scanner reader=new Scanner(System.in);
        //输入"数字 1 运算符 数字 2",注意输入时用空格分隔,或者每个输入占一行
        x=reader.nextDouble();
        symbol=reader.next().charAt(0);
        y=reader.nextDouble();
        //不考虑不合理输入,有 4 种情况,因此采用多分支结构处理
        if(symbol=='+')
            result=x+y;
        else if(symbol=='-')
            result=x-y;
        else if(symbol=='*')
            result=x*y;
        else
            result=x/y;
        System.out.println(result);
    }
}
```

根据输入运算符的不同,程序共有 4 个分支,如果考虑到除了输入四则运算符之外,用户还有可能输入其他字符,那么可以编写含 5 个分支的 if 语句。

注意:由于 Scanner 使用空格分隔输入的数据,所以应按"数字、空格、运算符、空格、数字"的方式输入,例如我们输入 3+5,则输出结果是 8.0;如果输入的是 3*5,则输出结果是 15.0。

例 3.5 计算分段函数的值:输入变量 x,当 x 大于 1 时,y 等于 1;当 x 为−1～1 时,y 等于 x;当 x 小于−1 时,y 等于−1。

```
import java.util.*;
public class Subsection {
    public static void main(String[] args) {
```

```
        double x,y;
        Scanner reader=new Scanner(System.in);
        x=reader.nextDouble();
        if(x> 1)
            y=1;
        else if(x> =-1 && x<=1)
            y=x;
        else
            y=-1;
        System.out.println(y);
    }
}
```

程序共有 3 个分支，每个分支都只有一条语句，这也是一个简单的多分支结构程序。

3.3　switch 语 句

多分支结构是根据逻辑判断结果值从多个分支中选择一个分支执行，虽然可以使用带有多个 else if 分支的 if 语句来解决，但当嵌套层数较多时，程序的可读性大大降低。如图 3.2 所示，从图的绘制上来说，感觉布尔表达式 1 比布尔表达式 2 重要，布尔表达式 2 比布尔表达式 3 重要，依次类推。实际上每个表达式代表一种情况，其重要性是相同的。使用 switch 语句也可以处理多分支问题。switch 语句根据表达式的值来执行多个操作中的一个，从图形上看结构更为清晰，switch 语句的通用形式如图 3.3 所示。

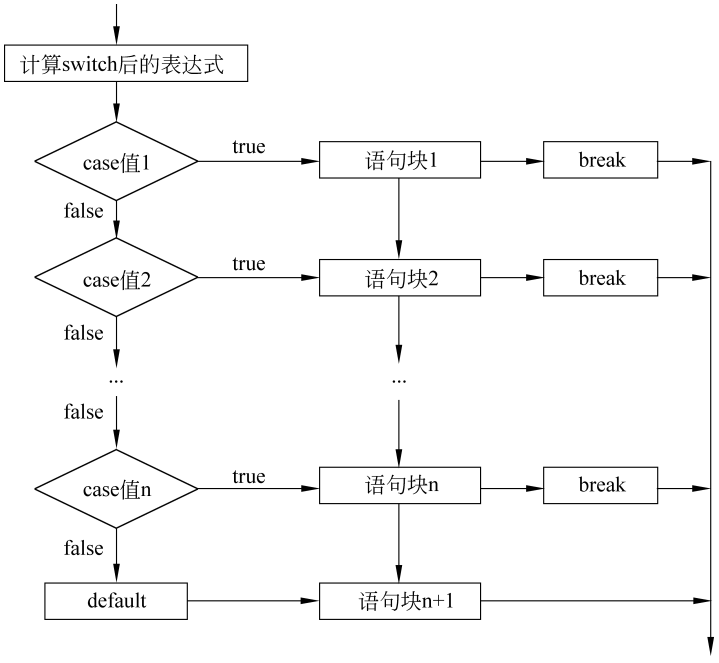

图 3.3　switch 语句的通用形式

switch 语句的基本结构如下：

```
switch(表达式){
    case  值 1:语句块 1;[ break; ]                    //分支 1
    case  值 2:语句块 2;[ break; ]                    //分支 2
    …

    case  值 n:语句块 n;[ break; ]                    //分支 n
    [ default :  语句块 n+1; ]                        //分支 n+1
}
```

注意：switch 后面的表达式的值类型可以是 byte、char、short 和 int 类型，也可是字符串类型，不允许是实数类型(如 double)。

例 3.6 利用 switch 语句完成例 3.4 的四则运算，输入两个数及运算符，根据运算符输出这两个数的四则运算结果。

```
import java.util.*;
public class ArithmeticSwitch {
    public static void main(String[] args) {
        double x,y,result=0;
        char symbol;
        Scanner reader=new Scanner(System.in);
        x=reader.nextDouble();
        symbol=reader.next().charAt(0);
        y=reader.nextDouble();
        switch(symbol){
            case '+':
                result=x+y;break;
            case '-':
                result=x-y;break;
            case '*':
                result=x*y;break;
            case '/':
                result=x/y;break;
        }
        System.out.println(result);
    }
}
```

运行程序，输入 3－5 后，switch 程序执行过程如下：因为 symbol 值为'－'，因此执行 case '－'后的语句块，计算两个数的差，遇到 break 语句，结束 switch 语句的执行，执行其后的输出语句，输出结果为－2.0。

switch 语句的 case 块可以直接写上多条语句，不需要花括号来界定语句块。break 语句是可选的，语句一旦进入某一个 case 分支，程序将一直向下执行，直到遇到 break 语

句或执行完最后一条语句后结束。例如，如果删除上述程序中的 4 个 break 语句，输入 3－5 后，程序在执行完语句“result＝x－y;”后，会接着执行“result＝x * y;”和“result＝x/y;”，因此输出结果为 0.6。在通常情况下，除非特殊需要，在每个语句块的最后都应添加 break 语句。

例 3.7　输入 2024 年任意一个月份，输出该月的天数。

解题思路：2024 年是闰年，2 月是 29 天，其他月份天数明确，根据输入月份不同输出该数据。

```
import java.util.*;
public class DaysInMonth {
    public static void main(String[] args) {
        int month;
        Scanner reader=new Scanner(System.in);
        month=reader.nextInt();
        switch(month){
            //以下月的天数一致，故只在 12 月后使用 break 语句
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12:
                    System.out.println("There are 31 days in that month.");
                    break;
            case 2:
                    System.out.println("There are 29 days in that month.");
                    break;
            case 4:
            case 6:
            case 9:
            case 11:
                    System.out.println("There are 30 days in that month.");
                    break;
            //对输入的错误月份给出错误提示信息
            default:
                    System.out.println("Invalid month.");
                    break;
        }
    }
}
```

在程序中由于多个 case 块要执行的语句是一样的，这时省略前面 case 块的 break 语句，仅在最后一个块上写上必要语句，为程序的编写带来了方便，也提高了程序的可读性。

3.4 选择结构的嵌套

在 if 语句的花括号内和 switch 语句 case 后的语句块中可以放置任何合法的 Java 语句,当然也包括 if 语句和 switch 语句,这就构成了选择结构的嵌套。

例 3.8 按由大到小的次序输入 3 条边的长度值,判断它们是否能构成三角形,如果能构成三角形,则判断构成的是锐角三角形、直角三角形还是钝角三角形。

```
import java.util.*;
public class TriangleShape {
    public static void main(String[] args) {
        double a,b,c;
        Scanner reader=new Scanner(System.in);
        a=reader.nextDouble();
        b=reader.nextDouble();
        c=reader.nextDouble();
        if(b+c> a){
            if(a*a<b*b+c*c)
                System.out.println("能构成锐角三角形");
            else if(a*a==b*b+c*c)
                System.out.println("能构成直角三角形");
            else
                System.out.println("能构成钝角三角形");
        }else{
            System.out.println("不能构成三角形");
        }
    }
}
```

程序先判断能否构成三角形,在能构成三角形的情况下再判断三角形的形状,从而形成了一个 if 语句的嵌套结构。

例 3.9 输入某年某月,输出该月的天数。

解题思路:除了 2 月,其他月份天数是固定的,在输出 2 月的天数时,需要对该年是平年还是闰年做出判断,闰年的判定方式是如果某年份能被 4 整除并且不能被 100 整除,或者能被 400 整除,那么这年是闰年。

```
import java.util.*;
public class DaysInMonth2 {
    public static void main(String[] args) {
        int year,month;
        Scanner reader=new Scanner(System.in);
        year=reader.nextInt();
        month=reader.nextInt();
```

```
        switch(month){
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12:
                    System.out.println("There are 31 days in that month.");
                    break;
            case 2:
                    if(year%4==0&&year%100!=0||year%400==0)
                        System.out.println("There are 29 days in that month.");
                    else
                        System.out.println("There are 28 days in that month.");
                    break;
            case 4:
            case 6:
            case 9:
            case 11:
                    System.out.println("There are 30 days in that month.");
                    break;
            default:
                    System.out.println("Invalid month.");
                    break;
        }
    }
}
```

这是一个 switch 语句中含 if 语句的选择结构的嵌套。

3.5　养成良好的程序设计习惯

在编写程序时编程人员应注意养成良好的程序设计习惯，这样才能设计出符合自己最高水平的程序，也能更快地提升自己的程序设计水平。就现阶段而言，良好的程序设计习惯应包括以下 5 方面。

（1）当面对一个问题时，要进行充分的思考，在感觉确有把握的情况下再进行编程实践。

（2）编写的程序应保持良好的书写习惯，如命名的标识符应尽量有意义，用好空白、缩进、注释等。

（3）编写的程序尽量反映编程人员解决这一问题的思路，也就是说，程序应忠实于编程人员解决问题的想法。

(4) 程序的书写尽量简单明了,少用复杂的句子。

(5) 程序的扩展性好,最好能直接用于其他程序中,或经过极少修改就能解决同类型的其他问题。

例 3.10 3 种方法找最大的数:输入 3 个整数,输出其中最大的数。

解题思路一:例 3.1 中编写了输出两个数中较大的数的程序。很容易对该程序进行修改,从而完成本问题。例 3.1 中已经找到了两个数中较大的数,用这个数与第三个数进行比较,这又是一个从两个数中找最大数的程序。按此思路编写的第一个程序代码如下:

```
import java.util.*;
public class CompareThree1 {
    public static void main(String[] args) {
        int a,b,c;
        Scanner reader=new Scanner(System.in);
        a=reader.nextInt();
        b=reader.nextInt();
        c=reader.nextInt();
        if(a> b){
            if(a> c)
                System.out.println(a);
            else
                System.out.println(c);
        }else{
            if(b> c)
                System.out.println(b);
            else
                System.out.println(c);
        }
    }
}
```

解题思路二:也可以用第一个数与第二个数和第三个数进行比较,从而判断第一个数是否为最大数,是则输出,否则剩下的就是第二个数与第三个数两个数的比较问题了。按此思路编写的第二个程序代码如下:

```
import java.util.*;
public class CompareThree2 {
    public static void main(String[] args) {
        int a,b,c;
        Scanner reader=new Scanner(System.in);
        a=reader.nextInt();
        b=reader.nextInt();
        c=reader.nextInt();
        if(a> =b&&a> =c)
            System.out.println(a);
```

```
        else if(b> c)
            System.out.println(b);
        else
            System.out.println(c);
    }
}
```

如果仔细研究一下就会发现第二个的程序更好，因为如果是从 4 个数中找最大数、从 5 个数中找最大数，在第二个程序的基础上更容易修改完成，也就是第二个程序的扩展性更好。

解题思路三：可是实际上如果从若干数中找最大的数，我们处理方式与上面两种方法都不同。查找的过程大体上是这样的，看第一个数，记住它，依次看后面的每个数，与我们记住的数进行比较，如果比我们记住的数大，则记住这个更大的数，否则什么都不做。这样当看完所有的数后，我们记住的就是最大的数。根据这一思路写出的第三个程序代码如下：

```
import java.util.*;
public class CompareThree3 {
    public static void main(String[] args) {
        int a,b,c,max;
        Scanner reader=new Scanner(System.in);
        a=reader.nextInt();
        b=reader.nextInt();
        c=reader.nextInt();
        //记住第一个数
        max=a;
        //与第二个数比较
        if(max<b)
            max=b;
        //与第三个数比较
        if(max<c)
            max=c;
        System.out.println(max);
    }
}
```

显然，这个程序不仅扩展性比第二个程序更好，而且更符合解决这一问题的思路，因此是这 3 个程序里最好的。

3.6　程序设计实例

例 3.11　输入两个整数，判断第一个数是不是第二个数的约数。

解题思路：如果 a 是 b 的约数，则 b 能被 a 整除，即 b%a 的值为 0。程序代码如下：

```
import java.util.*;
public class Divisor {
    public static void main(String[] args) {
        int a,b;
        Scanner reader=new Scanner(System.in);
        a=reader.nextInt();
        b=reader.nextInt();
        if(b%a==0)
            System.out.println("a 是 b 的约数。");
        else
            System.out.println("a 不是 b 的约数。");
    }
}
```

例 3.12 输入一个 1000 以内的正整数,将其反转后输出。如输入 123,则输出 321,输入 12,则输出 21,否则输出“数据输入范围错误。”的提示信息。

解题思路:因为输入的数有一位、两位、三位、不在题目要求范围 4 种情况,使用多分支的 if 结构完成。

```
import java.util.*;
public class Reverse {
    public static void main(String[] args) {
        int n,rev;
        Scanner reader=new Scanner(System.in);
        System.out.println("请输入 1~999 的整数:");
        n=reader.nextInt();
        if(n<1 || n> 999)
            System.out.println("数据输入范围错误。");
        else if(n> =100){
            rev=n%10*100+n/10%10*10+n/100;
            System.out.println(n+" 的反转数为"+ rev);
        }else if(n> =10){
            rev=n%10*10+n/10;
            System.out.println(n+" 的反转数为"+ rev);
        }else{
            rev=n;
            System.out.println(n+" 的反转数为"+ rev);
        }
    }
}
```

例 3.13 编写程序,按例 2.11 对扑克牌的编码方式,输入不包含大小王的一张扑克牌的编码,输出其花色和扑克牌上数值。例如,输入 22,输出“红心 10”;输入 39,输出“方块 A”。

解题思路：第 2 章编写的程序输出的是花色对应值，可使用 switch 语句按不同的花色值输出对应文字，在输出牌面值时注意牌面值为 1 时应输出的是字母 A，牌面值为 11、12、13 时输出对应的字母 J、Q、K。

```
import java.util.Scanner;
public class Poker1 {
    public static void main(String args[]) {
        int a;
        Scanner reader=new Scanner(System.in);
        System.out.println("请输入 0~51 的一个整数:");
        a=reader.nextInt();
        switch(a/13){
            case 0:
                System.out.print("黑桃");break;
            case 1:
                System.out.print("红心");break;
            case 2:
                System.out.print("梅花");break;
            case 3:
                System.out.print("方块");break;
        }
        switch(a%13){
            case 10:
                System.out.println("J");break;
            case 11:
                System.out.println("Q");break;
            case 12:
                System.out.println("K");break;
            case 0:
                System.out.println("A");break;
            default:
                System.out.println(a%13+1);break;
        }
    }
}
```

例 3.14　编写程序，随机获取两张不包括大小王的扑克牌 puke1、puke2，按附录 F“斗地主程序要求和玩法规则”比较两张牌的大小。

解题思路：随机获取一张牌即产生 0～51 的随机数，可使用 random 函数，该函数的返回值是双精度实数，对其直接取整，取整后有可能获得的两个随机数是相同的，程序需对其进行处理。

```
public class Poker2 {
    public static void main(String args[]) {
```

```
        int puke1,puke2;
        puke1=(int)(Math.random() * 52);
        puke2=(int)(Math.random() * 52);
        System.out.println("两张扑克牌的数值分别是"+puke1+","+puke2);
        System.out.println("puke1 花色值是"+puke1/13+",数字值是"+(puke1%13+1));
        System.out.println("puke2 花色值是"+puke2/13+",数字值是"+(puke2%13+1));
        if(puke1==puke2)
            System.out.println("发牌错误!");
        else {
            int a,b;
            //将牌面值 0~12 按其比较时的大小转换为 11、12、0~10,方便比较
            a=(puke1+11)%13;
            b=(puke2+11)%13;
            if(a > b)
                System.out.println("puke1 大");
            else if(a < b)
                System.out.println("puke2 大");
            else
                System.out.println("两张牌一样大。");
        }
    }
}
```

因为牌面值 0～12 对应 A、2～10、J、Q、K,而牌的大小次序为 3～10、J、Q、K、A、2,因此使用(牌面值+11)%13 获得牌面大小的比较值,即牌面 3～10、J、Q、K、A、2 分别对应数值 0～12,这样数值的大小就是牌的大小,从而方便比较。

3.7 知识补充

在 Java 语言中简单的选择结构可以使用条件运算符来完成。条件运算符(?:)是一个三元运算符,它涉及三个表达式。表达式采取下面形式:

```
布尔表达式?表达式 1:表达式 2;
```

若布尔表达式的计算结果为 true,就计算表达式 1 的值并返回;若布尔表达式的结果为 false,就计算表达式 2 的值并返回。

例 3.15 条件表达式示例:使用条件表达式完成例 3.1,比较两个整数的大小。代码如下:

```
class CompareTJ{
    public static void main(String[] args) {
        int a,b,max;
        Scanner reader=new Scanner(System.in);
        //输入两个整数
```

```
        a=reader.nextInt();
        b=reader.nextInt();
        //利用条件运算符找到较大的数
        max= a> b ? a: b;
        System.out.println(max);
    }
}
```

使用了条件运算符的表达式被称为条件表达式，也称问号表达式。

本章小结

1. 对于初学者来说，if 语句中复合语句的花括号尽量不要省略。

2. if 语句可对数据是否在某个区间进行判断，switch 语句只针对一个个的单值进行判断。

3. 若没有充分理由，switch 语句中的 break 语句尽量不要省略。

4. if 语句是更常用的多分支结构语句，而 switch 语句在阅读上更为清晰。

5. switch 的 default 块在用法上相当于 if 语句的 else 块。

6. 选择结构的嵌套是将一个选择结构完整地放置在其他选择结构的复合语句块中，注意不允许两个选择结构语句有交叉情况。

7. 平时应注意养成良好的编程习惯，良好的编程习惯有助于解决更复杂的问题，而且问题越复杂，其效果越明显。

概念测试

1. 在 Java 语言中，if 语句的判定表达式的计算结果必须是_____类型的。

2. switch 语句中除关键字 switch 外，使用的关键字还有_____、_____和_____。

3. 已知“int n;”，判断 n 是不是一个奇数的表达式为_____，判断 n 是不是一个偶数的表达式为_____。

4. 已知“int n;”，判断 n 是不是一个三位数的表达式为_____。

5. 判断 x 小于 3 或者 x 大于或等于 10 的表达式为_____。

6. 已知“char ch;”，判断 ch 是否为一个小写字母的表达式为_____，判断 ch 是否为一个数字字符的表达式为_____。

编程实践

1. 判断某年是不是闰年：如果某年能被 4 整除并且不能被 100 整除，或者能被 400 整除，那么这年是闰年。

2. 输入 3 条边的值(带小数),如果能构成三角形,则输出三角形的面积(保留两位小数),否则输出不能构成三角形的提示信息。

3. 输入两个整数,如果这两个整数同号(均为正或均为负),输出它们的积,否则输出它们的和。

4. 生成两个最高分为 100 分的成绩(整数),然后按由大到小的次序输出这两个数。

5. 输入一个数字,如果该数是负数,输出其立方根,否则输出其平方根,输出结果保留三位小数。

6. 输入一个实数,输出与之最邻近的两个整数。例如,输入 4.4,则输出 4 和 5;输入 −4.4,则输出 −5 和 −4。

7. 输入一个五位整数,判断其是否为回文数。回文数:正读、反读数字都一样的数,如 12321、43434 是回文数,12345、12323 不是回文数。

8. 输入一元二次方程 $ax^2+bx+c=0$ 的系数 a、b、c,输出其根的个数(0、1 或 2)。

9. 输入一元二次方程 $ax^2+bx+c=0$ 的系数 a、b、c,输出其根的情况,如果有实根,则计算并输出其根,输出结果保留两位小数。

10. 输入一个实数,如果该数大于 0 则输出数字 1,等于 0 则输出数字 0,小于 0 则输出数字 −1。

11. 输入一个成绩,输出其对应的分数等级:优秀(90～100)、良好(80～89)、中(70～79)、及格(60～69)、不及格(0～59),分别用 if 和 switch 两种语句来完成。